# Auditable Attribute-Based Credentials Scheme and Its Application in Contact Tracing

Pengfei Wang[1], Xiangyu Su[1], Mario Larangeira[1,2], and Keisuke Tanaka[1]

[1] Department of Mathematical and Computing Sciences, School of Computing,
Tokyo Institute of Technology. Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
pengfei.w@outlook.com, su.x.ab@m.titech.ac.jp,
mario@c.titech.ac.jp, keisuke@is.titech.ac.jp.
[2] Input Output, Global. mario.larangeira@iohk.io.

**Abstract.** During the pandemic, the limited functionality of existing privacy-preserving contact tracing systems highlights the need for new designs. Wang et al. proposed an environmental-adaptive framework (CSS '21) but failed to formalize the security. The similarity between their framework and attribute-based credentials (ABC) inspires us to reconsider contact tracing from the perspective of ABC schemes. In such schemes, users can obtain credentials on attributes from issuers and prove the credentials anonymously (i.e., hiding sensitive information of both user and issuer). This work first extends ABC schemes with auditability, which enables designated auditing authorities to revoke the anonymity of particular *issuers*. For this purpose, we propose an "auditable public key (APK)" mechanism that extends the updatable public key by Fauzi et al. (AsiaCrypt '19). We provide formal security definitions regarding auditability and build our auditable ABC scheme by adding a DDH-based APK to Connolly et al.'s ABC construction (PKC '22). Note that the APK mechanism can be used as a plug-in for other cryptographic primitives and may be of independent interest. Finally, regarding contact tracing, we refine Wang et al.'s framework and present a formal treatment that includes security definitions and protocol construction. An implementation is provided to showcase the practicality of our design.

**Keywords:** Attribute-Based Credentials · Auditable Public Keys · Contact Tracing

## 1 Introduction

Contact tracing, a method that prevents diseases from spreading, faces new challenges considering new findings in epidemiology research. Proposed in [38], the environmental-adaptive contact tracing (EACT) framework took different transmission modes (*i.e.*, droplet and airborne) and virus distribution (*e.g.*, lifespan and region size, which depends on environmental factors) into consideration (Appendix A recalls the rationale behind embedding environmental factors in contact tracing systems). However, their framework are based on an informal threat model and failed to unify the system syntax for different transmission modes, hence, leaving a gap between theoretical proofs and implementations.

The similarity between their framework and a self-issuing decentralized credentials scheme [28] inspires us to turn our eyes to credentials schemes, typically the *attribute-based* ones (ABC). Note that we consider ABC schemes, *e.g.*, [26, 18], instead of more general anonymous credentials, *e.g.*, [11–13]. This is because the capability of embedding attributes in credentials empowers contact tracing systems to manage environmental factors as attributes. To the best of our knowledge, this approach has seen limited exploration or association with contact tracing in previous works despite its inherent viability[1]. We explain the reason as follows. Recall that an ABC scheme involves issuers, users, and verifiers. In the issuance phase, an issuer grants a credential to a user on the user's attributes. The user can then prove possession (showing) of the credential on their attributes without revealing identities, but they *cannot* prove attributes that are not embedded in their credentials. Hence, by building contact tracing systems atop ABC schemes: (1) users can take environmental factors and local information as attributes; (2) users can issue others credentials on these attributes as contact records; (3) users can anonymously prove their records to potentially malicious verifiers (in contact tracing, medical agencies). It is also convenient to bring the broad spectrum of functionalities in ABC to contact tracing, *e.g.*, selective showing [26], proof of disjoint attributes [18], issuer-hiding [6, 18], delegation [4], *traceability* [32], etc.

Moreover, the security of ABC, *i.e.*, anonymity and unforgeability, can also be adapted to contact tracing (as we will show in Section 4.2)[2]. Intuitively, given any honest user's showing of contact records, anonymity prevents other users and medical agencies (even if they collude) from identifying the user or learning anything except what is intentionally revealed by the user. Whereas, unforgeability guarantees that no user can perform a valid showing if she does not possess a corresponding contact record (*i.e.*, a credential issued by another user according to some committed attributes, *e.g.*, environmental factors). The two properties resemble the "(pseudonym and trace) unlinkability" and "integrity" of contact tracing systems proposed in [19] (more discussion in Section 4.2).

However, existing ABC schemes cannot be utilized directly to build contact tracing systems due to the lack of tracing capability. Note that the traceability in [32] is similar to group signatures, *i.e.*, to revoke the anonymity of *regular users*. In contrast, the traceability of contact systems should enable the issuer of a contact record to be notified whenever the record is being shown. For example, when two users (A and B) have contact, they first exchange contact records by issuing each other a credential based on the contact. Then, when one user (say,

[1] Silde and Strand [37] proposed a contact tracing system based on anonymous tokens, *i.e.*, an anonymous credential variant that does not support attributes.

[2] Notably, game-based and simulation-based security definitions of contact tracing systems have been proposed in [19, 5], respectively. This work will focus on the game-based ones because we proceed from the perspective of ABC schemes with game-based definitions.

user A) is diagnosed and presents her credential issued by her counterparty (user B) to medical agencies, user B should be able to check if the presented credential is issued by herself. If so, user B can confirm that she had close contact with user A. We formalize this functionality as auditability of issuers in the underlying credentials scheme, which we call an auditable ABC.

**Our approach and contributions.** We show a brief image of our approach. In order to build an auditable ABC scheme, we first propose a cryptographic tool called the "auditable public keys (APK)" mechanism, which extends the updatable public key given in [23]. The APK embeds extra structure in the secret and public key pair with a new auditing key. The structure will be preserved even after updating the public key and can be verified (we call it audit) by the auditing key. That is, a participant who holds the auditing key corresponding to some key pair can audit if a given public key is updated from the corresponding public key without knowing the secret randomness in the update algorithm. Like the updatable public key, our APK can be used as a plug-in for many different cryptographic primitives, hence, not being limited to credentials schemes (we show a concrete example in Appendix D).

Next, we adapt APK to the existing ABC scheme [26] and define the formal syntax of our auditable ABC. We show a concrete construction for the APK mechanism based on the matrix Diffie-Hellman problem over matrix distributions [22, 35]. We prove that our APK construction can be inserted into the structure-preserving signatures on equivalence classes (SPS-EQ) scheme [18] without breaking the security of the original SPS-EQ (though incurring a slight reduction loss). By employing our modified SPS-EQ, a set-commitment scheme from [26], and a zero-knowledge proofs of knowledge protocol [24], we present a construction for the auditable ABC scheme.

Finally, we refine the EACT framework [38] and provide a construction based on our auditable ABC scheme. Hence, we can unify the tracing process of the conventional Bluetooth Low Energy (BLE)-based setting for droplet mode and their discrete-location-tracing setting (DLT) for airborne mode. Then, we argue that the security of the refined EACT can be derived from our auditable ABC scheme but requires sufficient adaptions, *e.g.*, in contact tracing, the verifier of credentials may be malicious and approve falsely shown credentials. We explain these adaptions and finally show an implementation (in Section 4.3) of our construction on real-life Android devices to demonstrate practicality.

*Our contributions.* Our contributions are threefold: (1) we propose an APK mechanism that can be used as a plug-in tool for many cryptographic primitives; (2) we propose an auditable ABC scheme that inherits auditability from APK. Then, we show concrete constructions for APK and the auditable ABC scheme; (3) we refine and construct the EACT framework [38] based on credentials schemes. We also provide formal security definitions and implement the construction. Additionally, we add algorithms to jPBC library [15] to support matrix-based bilinear pairing operations during implementation.

**Related works.** Despite the broad functionalities of ABC schemes, no existing work considers the same traceability (revoking issuer's anonymity) as in contact tracing systems. Regarding auditability of ABC schemes, existing works [7, 17] focused on auditing the credentials back to their holders, *i.e.*, regular users. Instead, our auditability intends to identify issuers. This is because, as shown in Section 3.2, modifications are made into ABC syntax so that verifiers cannot identify *the issuer* of shown credentials even if they collude with the original issuer. However, such a property opens the gate of fabricating issuers. Hence, it is crucial to let issuers (or designated third parties chosen by the issuer, called auditors) check if a shown credential originates from the issuer herself.

To compare with existing contact tracing systems, we consider three aspects: (1) security (*i.e.*, anonymity/unlinkability and tracing-soundness/integrity); (2) extensibility (*e.g.*, the capability of handling different transmission modes and environmental factors); (3) efficiency (*e.g.*, requiring BLE handshakes during the recording phase or not). We notice that these requirements may contradict each other (in fact, unlinkability and integrity may also have contradictions [19]). For example, as mentioned above, revealing more data (extensibility) inevitably incurs breaches in anonymity (security). Then, to fix such breaches, we have to rely on heavy mechanisms that burden the system's efficiency. In the following, we evaluate several cryptographic contact tracing systems to prove our observation.

A simple paradigm of contact tracing utilizes symmetric primitives (*e.g.*, pseudo-random permutations/functions/generators (PRP/PRF/PRG) [2, 1] and hash functions [14]) to generate period-specific keys and pseudonyms (here, the period can be several hours or days). As shown in [19], these systems can achieve *unlinkability* (*i.e.*, period-specific pseudonyms are unlinkable) due to the pseudo-randomness of the underlying building blocks; and *integrity* (*i.e.*, no adversary can forge recorded pseudonyms to trigger users' tracing) due to the pre-image resistance of these primitives; but suffer from *the relay (and replay) attacks* (*i.e.*, the adversary can relay or replay previous records to break integrity) [17] because users cannot tell if a pseudonym has been presented or not (without checking timestamps). The simplicity of this paradigm allows us to construct highly efficient systems. However, the simplicity also prevents us from recording anything but pseudonyms, hence, limiting the extension capability.

One method to enhance the aforementioned systems is to use re-randomizable primitives (*e.g.*, signature schemes) as in [34, 38] and in our work. Concretely, a user obtains a piece of authorized information (in most cases, a signature) from her counterparty (in [38], the counterparty can be regarded as the user herself) during a close contact, and then presents an updated (re-randomized) signature to medical agencies when she is diagnosed. This approach achieves *unlinkability* from the re-randomizability of the building blocks; and *integrity* from unforgeability. Previous works [34, 38] consider semi-honest verifiers (medical agencies) who only approve valid signatures to extend the bulletin board. Hence, they can prevent *the relay and replay attacks* by requiring additionally the freshness of signatures.

Moreover, Wang et al. [38] demonstrate with the environmental-adaptive framework that contact tracing systems can handle more useful information to enhance tracing precision (as explained in Appendix A). Their drawback is that users must reveal all attributes to verifiers during the tracing phase. We push forward their idea of utilizing credential schemes and add selective showing capability to our system. Hence, users in our system only reveal what is necessary for deciding close contacts without leaking any other information. However, extensions come with associated costs: our system requires handshakes during BLE scanning (same to [34]) and is built atop pairing-based primitives (same to [38] and the third construction given in [14]).

Inherently shown in [1, 14], where authors present various constructions of contact tracing systems with varying levels of security and efficiency, the trade-off among these requirements (security, extensibility, and efficiency) prevents us from finding the ultimate solution for contact tracing. We argue that our system is *secure* despite handling more sensitive data; is *extensible* to tackle new epidemiology findings; and is *efficient enough* to be implemented in real life.

**Organization.** We organize the content as follows. First, we present the necessary general building blocks and assumptions in Section 2. Section 3 formally introduces our first contribution, *i.e.*, an APK mechanism and an auditable ABC scheme. We show constructions and give security proofs to these schemes. Section 4 shows a construction for our refined EACT framework based on auditable ABC, argues its security, and provides implementation results. Finally, Section 5 concludes this work.

## 2 Preliminaries

*Notation.* Throughout this paper, we use $\lambda$ for the security parameter and $\mathsf{negl}(\cdot)$ for the negligible function. PPT is short for probabilistic polynomial time. For an integer $q$, $[q]$ denotes the set $\{1, \ldots, q\}$. Given a set $A$, $x \overset{\$}{\leftarrow} A$ denotes that $x$ is randomly and uniformly sampled from $A$; whereas, for an algorithm $\mathsf{Alg}$, $x \leftarrow \mathsf{Alg}$ denotes that $x$ is assigned the output of an algorithm $\mathsf{Alg}$ on fresh randomness. Let $\mathsf{Alg}_1, \mathsf{Alg}_2$ be two algorithms, $\langle \mathsf{Alg}_1, \mathsf{Alg}_2 \rangle$ denotes a potentially interactive protocol between the two algorithms. Let $\mathsf{H}$ denote a collision-free hash function. For an additive group $\mathbb{G}$, $\mathbb{G}^*$ denotes $\mathbb{G} \setminus \{0_{\mathbb{G}}\}$. For a set $A \subseteq \mathbb{Z}_p$, we refer to a monic polynomial of order $|A|$ defined over $\mathbb{Z}_p[X]$, $\mathsf{Ch}_A(X) \overset{\Delta}{=} \mathit{\Pi}_{x \in A}(X - x) = \sum_{i=0}^{|A|} c_i \cdot X^i$ as $A$'s characteristic polynomial.

We denote the asymmetric bilinear group generator as $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$ where $\mathsf{BG} \overset{\Delta}{=} (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. Here, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are additive cyclic groups of prime order $p$ with $\lceil \log_2 p \rceil = \lambda$, $P_1, P_2$ are generators of $\mathbb{G}_1, \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a type-3, *i.e.*, efficiently computable non-degenerate bilinear map with no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. For an element $a \in \mathbb{Z}_p$ and $i \in \{1, 2\}$, $[a]_i$ denotes $aP_i \in \mathbb{G}_i$ as the representation of

$a$ in group $\mathbb{G}_i$. As mentioned in [18], for vectors or matrices $\mathbf{A}, \mathbf{B}$, the bilinear map $e$ computes $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T \in \mathbb{G}_T$.

**General building blocks and assumptions.** This work takes the black-box use of three cryptographic primitives: (1) a digital signature scheme $\mathsf{SIG} \overset{\Delta}{=} (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies correctness and existentially unforgeability under adaptive chosen-message attacks (EUF-CMA) [29]; (2) a set-commitment scheme $\mathsf{SC} \overset{\Delta}{=} (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenSubset}, \mathsf{VerifySubset})$ that satisfies correctness, binding, subset-soundness, and hiding [26]; (3) a zero-knowledge proofs of knowledge (ZKPoK) protocol $\Pi$ that satisfies completeness, perfect zero-knowledge, and knowledge-soundness [24]. The formal definitions of these primitives can be found in Appendix B.

Moreover, we assume the following assumptions hold over matrix distribution: the matrix decisional Diffie-Hellman (MDDH) assumption [22] and the kernel matrix Diffie-Hellman (KerMDH) assumption [35]. We also assume the Diffie-Hellman (DDH) assumption and the $q$-co-discrete-logarithm ($q$-co-DL) assumption holds over bilinear groups.

**Definition 1 (Matrix Distribution).** *Let $l, k \in \mathbb{N}$ with $l > k$. $\mathcal{D}_{l,k}$ is a matrix distribution that outputs matrices in $\mathbb{Z}_p^{l \times k}$ of full rank $k$ in polynomial time. We further denote $\mathcal{D}_k \overset{\Delta}{=} \mathcal{D}_{k+1,k}$.*

Let $\mathsf{BGGen}$ be the bilinear group generator that outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ and $\mathcal{D}_{l,k}$ be a matrix distribution.

**Definition 2 ($\mathcal{D}_{l,k}$-MDDH Assumption).** *The $\mathcal{D}_{l,k}$-MDDH assumption holds in group $\mathbb{G}_i \in \mathsf{BG}$ where $i \in \{1, 2, T\}$ relative to $\mathsf{BGGen}$, if for all $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), \mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_{l,k}, \mathbf{w} \overset{\$}{\leftarrow} \mathbb{Z}_p^k, \mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_p^l$ and all PPT adversary $\mathcal{A}$, the following advantage is negligible of $\lambda$:*

$$\mathsf{Adv}^{MDDH}_{\mathcal{D}_{l,k},\mathbb{G}_i} = |\Pr[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_i, [\mathbf{Aw}]_i) = 1] - \Pr[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_i, [\mathbf{u}]_i) = 1]|.$$

**Definition 3 ($\mathcal{D}_{l,k}$-KerMDH Assumption).** *The $\mathcal{D}_{l,k}$-KerMDH assumption holds in group $\mathbb{G}_i \in \mathsf{BG}$ where $i \in \{1, 2\}$ relative to $\mathsf{BGGen}$, if for all $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), \mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_{l,k}$ and all PPT adversary $\mathcal{A}$, the following advantage is negligible of $\lambda$:*

$$\Pr[[\mathbf{x}]_{3-i} \leftarrow \mathcal{A}(\mathsf{BG}, [\mathbf{A}]_i]) : e([\mathbf{x}^\top]_{3-i}, [\mathbf{A}]_i) = [\mathbf{0}]_T \wedge \mathbf{x} \neq \mathbf{0})].$$

**Definition 4 (DDH Assumption).** *The DDH assumption holds in $\mathbb{G}_i \in \mathsf{BG}$ where $i \in \{1, 2\}$ for $\mathsf{BGGen}$, if for all $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), x, y, z \overset{\$}{\leftarrow} \mathbb{Z}_p$ and all PPT adversary $\mathcal{A}$, the following advantage is negligible of $\lambda$:*

$$|\Pr[\mathcal{A}(\mathsf{BG}, xP_i, yP_i, xyP_i) = 1] - \Pr[\mathcal{A}(\mathsf{BG}, xP_i, yP_i, zP_i) = 1]|.$$

**Definition 5 ($q$-co-DL Assumption).** *The $q$-co-DL assumption holds for $\mathsf{BGGen}$, if for all $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), a \overset{\$}{\leftarrow} \mathbb{Z}_p$ and all PPT adversary $\mathcal{A}$, the following advantage is negligible of $\lambda$:*

$$\Pr[a' \leftarrow \mathcal{A}(\mathsf{BG}, ([a^j]_1, [a^j]_2)_{j \in [q]} : a' = a].$$

## 3 Auditable Attribute-Based Credentials Scheme

This section first presents an auditable public key (APK) mechanism, then an APK-aided ABC scheme, which will be the main building block of our refined environmental-adaptive contact tracing framework.

Conventionally, an attribute-based credentials (ABC) scheme involves three types of participants: Issuer (also called organization), user, and verifier. An issuer grants credentials to a user on the user's attributes. The user can then prove possession of credentials with respect to her attributes to verifiers. The basic requirements of a secure ABC include correctness, anonymity, and unforgeability [26]. On a high level, correctness guarantees that verifiers always accept the showing of a credential if the credential is issued honestly; Anonymity prevents verifiers and (malicious) issuers (even by colluding) from identifying the user or exposing information during a showing against the user's will; Unforgeability requires that users (even by colluding) cannot perform a valid showing of attributes if the users do not possess credentials for the attributes.

The recent specifications of decentralized identifiers and verifiable credentials [36, 33] refueled the interest of the community in researching ABC schemes. New functionalities, as shown in Section 1, have been proposed to broaden the application of ABC schemes. Abstracted from the demands of contact tracing systems, we propose yet another functionality, *i.e.*, the auditability, that enables designated users to verify the particular *issuer* of a shown credential. In order to present our scheme, we first introduce the notion of the auditable public key (APK) mechanism that extends the updatable public key [23]. Then, we employ APK and present our auditable ABC scheme.

### 3.1 Auditable Public Keys

Proposed in [23], the updatable public key mechanism is a generic tool that can be integrated into many cryptographic primitives, *e.g.*, digital signature and public key encryption schemes. The mechanism enables public keys to be updated in a public fashion, and updated public keys are indistinguishable from freshly generated ones. The verification of public keys either requires the corresponding secret key (verifying the key pair) or the randomness used in the updating algorithm. However, these approaches are insufficient in multi-user cases, *e.g.*, in credentials schemes and contact tracing systems. The reasons are: (1) secret keys should only be known to their holders; (2) asking the user who runs the updating algorithm to store its random value or keep the value secret may require impractical assumptions (*e.g.*, assuming *every* user to be honest).

Therefore, we propose an APK mechanism to extend the updatable public key by embedding a structure represented by an auditing key into public keys. The structure enables designated third parties who hold the auditing key, the auditors, to decide whether a public key is updated from the corresponding public key of the auditing key. Moreover, we require that no auditor can learn the corresponding *secret* key of its auditing key. Hence, we separate the role of users, *i.e.*, a user can delegate her capability of auditing to an auditor without revealing the secret key, and a user who performs the updating algorithm can discard her randomness without the concern of being asked to provide it.

The formal syntax and security definitions of APK are given in the following. We recall and extend the definitions from [23].

**Definition 6 (Auditable Public Key Mechanism).** *An auditable public key (APK) mechanism involves a tuple of algorithms* $\mathsf{APK} \overset{\Delta}{=} (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Update}, \mathsf{VerifyKP}, \mathsf{VerifyAK}, \mathsf{Audit})$ *that are performed as follows.*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* $\lambda$ *and outputs the public parameter* $\mathsf{pp}$ *that includes secret, public and auditing key space* $\mathcal{SK}, \mathcal{PK}, \mathcal{AK}$*. These are given implicitly as input to all other algorithms;*
- $\mathsf{KGen}(\mathsf{pp})$ *takes as input the public parameter* $\mathsf{pp}$ *and outputs a secret and public key pair* $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{SK} \times \mathcal{PK}$*, and an auditing key* $\mathsf{ak} \in \mathcal{AK}$*. Later, we omit* $\mathsf{pp}$ *in algorithm inputs;*
- $\mathsf{Update}(\mathsf{pk}; r)$ *takes as input a public key* $\mathsf{pk}$ *and a randomness* $r$*. It outputs a new public key* $\mathsf{pk}' \in \mathcal{PK}$*;*
- $\mathsf{VerifyKP}(\mathsf{sk}, \mathsf{pk}', r)$ *is deterministic, and takes as input a secret key* $\mathsf{sk} \in \mathcal{SK}$*, a value* $r$ *and a public key* $\mathsf{pk}' \in \mathcal{PK}$*. It outputs* $1$ *if* $\mathsf{pk}' \leftarrow \mathsf{Update}(\mathsf{pk}; r)$ *given* $(\mathsf{sk}, \mathsf{pk}, \cdot) \leftarrow \mathsf{KGen}(\mathsf{pp})$*, or* $0$ *otherwise;*
- $\mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak})$ *is deterministic, and takes as input a secret key* $\mathsf{sk} \in \mathcal{SK}$ *and an auditing key* $\mathsf{ak} \in \mathcal{AK}$*. It outputs* $1$ *if* $(\mathsf{sk}, \cdot, \mathsf{ak}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*, or* $0$ *otherwise;*
- $\mathsf{Audit}(\mathsf{ak}, \mathsf{pk}', \mathsf{pk})$ *is deterministic and is performed by a designated auditor who holds the auditing key* $\mathsf{ak} \in \mathcal{AK}$ *of a secret and public key pair* $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{SK} \times \mathcal{PK}$*.* $\mathsf{Audit}$ *takes as input a public key* $\mathsf{pk}' \in \mathcal{PK}$*, the auditing key* $\mathsf{ak}$ *and the public key* $\mathsf{pk}$*. It outputs* $1$ *if* $\mathsf{pk}'$ *is updated from* $\mathsf{pk}$*, i.e., there exists* $r$ *such that* $\mathsf{pk}' \leftarrow \mathsf{Update}(\mathsf{pk}; r)$*, or* $0$ *otherwise.*

APK mechanism satisfies correctness, indistinguishability, and unforgeability.

**Definition 7 (Correctness).** *An APK mechanism satisfies perfect correctness if the following properties hold for any* $\lambda > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $(\mathsf{sk}, \mathsf{pk}, \mathsf{ak}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: (1) the update process verifies, i.e.,* $\mathsf{VerifyKP}(\mathsf{sk}, \mathsf{Update}(\mathsf{pk}; r), r) = 1$*; (2) the auditing key verifies, i.e.,* $\mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak}) = 1$*; (3) the auditing process verifies, i.e.,* $\mathsf{Audit}(\mathsf{ak}, \mathsf{pk}', \mathsf{pk}) = 1$ *for any* $\mathsf{pk}' \leftarrow \mathsf{Update}(\mathsf{pk})$*.*

The indistinguishability of APK follows [23], *i.e.*, no adversary can distinguish between an updated known public key and a freshly generated one. Note that (also applies in unforgeability) the adversary can query to $\mathsf{KGen}$ and $\mathsf{Update}$ since these algorithms are publicly available.

**Definition 8 (Indistinguishability).** *An APK mechanism satisfies indistinguishability if for any PPT adversary* $\mathcal{A}$*, the following probability holds for any* $\lambda > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $(\mathsf{sk}^*, \mathsf{pk}^*, \mathsf{ak}^*) \leftarrow \mathsf{KGen}(\mathsf{pp})$

$$\left| \Pr\left[ \begin{array}{l} b \overset{\$}{\leftarrow} \{0,1\}; \mathsf{pk}_0 \leftarrow \mathsf{Update}(\mathsf{pk}^*); \\ (\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{ak}_1) \leftarrow \mathsf{KGen}(\mathsf{pp}); \\ b^* \leftarrow \mathcal{A}(\mathsf{pk}^*, \mathsf{pk}_b) \end{array} : b^* = b \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

We formalize two types of unforgeability, *i.e.*, for secret key and auditing key. Concretely, the former requires that given an auditing key with its corresponding public key, the adversary cannot produce a secret and public key pair, and a randomness, such that: (1) the output public key is updated from the secret key's corresponding public key with respect to the randomness; (2) the output secret key and the given auditing key pass the verification given by $\mathsf{VerifyAK}$; (3) the auditing key, the output public key and the given public key pass the auditing given by $\mathsf{Audit}$. This property captures adversarial auditors who hold an auditing key and intend to recover the corresponding secret key. Hence, it covers the one given in [23], in which the adversary is only given a public key.

Next, the auditing key unforgeability requires that given a public key, the adversary cannot produce an auditing key such that the corresponding secret key of the public key verifies the auditing key. This property captures adversarial participants who intend to trigger the auditing algorithm to output 1 for arbitrary public keys. The formal definitions are as follows.

**Definition 9 (Secret Key Unforgeability).** *An APK mechanism satisfies secret key unforgeability if for any PPT adversary $\mathcal{A}$, the following probability holds for any $\lambda > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, and $(\mathsf{sk}, \mathsf{pk}, \mathsf{ak}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*

$$\Pr\left[(\mathsf{sk}', \mathsf{pk}', r) \leftarrow \mathcal{A}(\mathsf{ak}, \mathsf{pk}) \begin{array}{r} \mathsf{VerifyKP}(\mathsf{sk}', \mathsf{pk}', r) = 1 \wedge \\ : \mathsf{VerifyAK}(\mathsf{sk}', \mathsf{ak}) = 1 \wedge \\ \mathsf{Audit}(\mathsf{ak}, \mathsf{pk}', \mathsf{pk}) = 1 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

**Definition 10 (Auditing Key Unforgeability).** *An APK mechanism satisfies auditing key unforgeability if for any PPT adversary $\mathcal{A}$, the following probability holds for any $\lambda > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, and $(\mathsf{sk}, \mathsf{pk}, \mathsf{ak}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*

$$\Pr\left[\mathsf{ak}' \leftarrow \mathcal{A}(\mathsf{pk}) : \mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak}') = 1\right] \leq \mathsf{negl}(\lambda).$$

For constructions, similar to the updatable public key [23], our APK can be constructed from the DDH problem and its variants. Section 3.3 will show a concrete construction based on the MDDH problem, which will further serve as a building block for our auditable ABC scheme[3].

### 3.2 Formal Definitions of Auditable ABC

The starting point of our auditable ABC is [26] which supports selective showing on subsets of attributes. Then, we integrate APK by modifying the key generation algorithm of issuers and adding the auditing algorithm. Given a credential showing, the auditing algorithm with an auditing key outputs 1 or 0 to indicate whether the shown credential is issued by a secret key corresponding to the auditing key. We show the formal syntax of auditable ABC in the following.

[3] We will also give a DDH-based construction in Appendix D. There, we show an example that utilizes the DDH-based APK to extend the famous B(G)LS signature scheme [9, 8].

**Definition 11 (Auditable ABC Scheme).** *An auditable ABC scheme* $\mathsf{AABC}$ *consists of PPT algorithms* $(\mathsf{Setup}, \mathsf{OrgKGen}, \mathsf{UsrKGen})$*, two potentially interactive protocols* $\langle \mathsf{Obtain}, \mathsf{Issue} \rangle$ *and* $\langle \mathsf{Show}, \mathsf{Verify} \rangle$*, and a deterministic algorithm* $\mathsf{Audit}$*. The participants in* $\mathsf{AABC}$ *perform as follows.*

- $\mathsf{Setup}(1^\lambda, q)$ *takes as input the security parameter* $\lambda$ *and the size upper bound* $q$ *of attribute sets. It outputs the public parameter* $\mathsf{pp}$*;*
- $\mathsf{OrgKGen}(\mathsf{pp})$ *is executed by issuers.* $\mathsf{OrgKGen}$ *takes as input the public parameter* $\mathsf{pp}$*. It outputs an issuer-secret and issuer-public key pair* $(\mathsf{osk}, \mathsf{opk})$ *with an auditing key* $\mathsf{ak}$*. The issuer delegates* $\mathsf{ak}$ *to users (auditors) selected by herself (if there is none, the issuer is the auditor);*
- $\mathsf{UsrKGen}(\mathsf{pp})$ *is executed by users.* $\mathsf{UsrKGen}$ *takes as input the public parameter* $\mathsf{pp}$*. It outputs a user-secret and user-public key pair* $(\mathsf{usk}, \mathsf{upk})$*. Later, we omit* $\mathsf{pp}$ *in algorithm inputs;*
- $\langle \mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}, A), \mathsf{Issue}(\mathsf{upk}, \mathsf{osk}, A) \rangle$ *are PPT algorithms executed between a user and an issuer, respectively.* $\mathsf{Obtain}$ *takes as input the user-secret key* $\mathsf{usk}$*, the issuer-public key* $\mathsf{opk}$ *and an attribute set* $A$ *of size* $|A| \leq q$*;* $\mathsf{Issue}$ *takes as input the user-public key* $\mathsf{upk}$*, the issuer-secret key* $\mathsf{osk}$ *and the attribute set* $A$*.* $\mathsf{Obtain}$ *returns* $\mathsf{cred}$ *on* $A$ *to the user, and* $\mathsf{cred} = \perp$ *if protocol execution fails. The protocol outputs* $(\mathsf{cred}, I)$ *where* $I$ *denotes the issuer's transcript;*
- $\langle \mathsf{Show}(\mathsf{opk}, A, D, \mathsf{cred}), \mathsf{Verify}(D) \rangle$ *are executed between a user and a verifier, respectively, where* $\mathsf{Show}$ *is a PPT algorithm, and* $\mathsf{Verify}$ *is deterministic.* $\mathsf{Show}$ *takes as input an issuer-public key* $\mathsf{opk}$*, an attribute set* $A$ *of size* $|A| \leq q$*, a non-empty set* $D \subseteq A$ *representing the attributes to be shown, and a credential* $\mathsf{cred}$*;* $\mathsf{Verify}$ *takes as input the set of shown attributes* $D$*.* $\mathsf{Verify}$ *returns* $1$ *if the credential showing is accepted, or* $0$ *otherwise. The protocol outputs* $(S, b)$ *where* $S$ *denotes the showing (user's transcript), and* $b \in \{0, 1\}$*. For convenience, we also write* $b \leftarrow \langle \mathsf{Show}, \mathsf{Verify} \rangle(S)$*;*
- $\mathsf{Audit}(\mathsf{ak}, S, \mathsf{opk})$ *is executed by a designated auditor with an auditing key* $\mathsf{ak}$ *such that corresponding issuer-key pair is* $(\mathsf{osk}, \mathsf{opk})$*.* $\mathsf{Audit}$ *also takes as input a showing of credential* $(S, \cdot) \leftarrow \langle \mathsf{Show}, \mathsf{Verify} \rangle$ *and the issuer-public key* $\mathsf{opk}$*. It outputs* $1$ *if the shown credential is issued with* $\mathsf{osk}$*, or* $0$ *otherwise.*

In addition to the auditing process, we make two modifications to the ABC scheme from [26]. First, we write protocol transcriptions of $\langle \mathsf{Obtain}, \mathsf{Issue} \rangle$ and $\langle \mathsf{Show}, \mathsf{Verify} \rangle$ explicitly in our syntax concerning that the application in contact tracing may involve non-interactive proofs and require some transcripts to be publicly accessible (Section 4.1). In contrast, the previous works [26, 18] only mentioned them in security definitions.

Second, our $\mathsf{Verify}$ algorithm of $\langle \mathsf{Show}, \mathsf{Verify} \rangle$ takes as input only the attribute sets to be shown. In contrast, the original scheme also takes the issuer-public key $\mathsf{opk}$ of the $\mathsf{Show}$ algorithm. Their purpose is to prevent credentials from being issued by unidentified issuers. However, as shown in [18], the exposure of the issuer identity affects the users' anonymity. Although some previous works [6, 18] proposed the issuer-hiding property so that users can hide their credential issuers' identities within a list of identified issuers, achieving such security incurs heavy mechanisms. Here, we rely on the $\mathsf{Audit}$ algorithm to provide an extra verification layer. That is, given an updated issuer-public key in a credential showing, the auditor who holds an auditing key *corresponding to an identified public key* must prove whether the shown credential is issued by the corresponding secret key.

**Security properties.** We formally define correctness, anonymity, and unforgeability (two types) for our auditable ABC scheme. Concretely, correctness requires auditors to output 1 on any valid showing of credentials if the credential was issued by the corresponding secret key of the auditing key. The unforgeability game grants its adversary access to auditing keys. In the following, we omit $\mathsf{pp}$ if the algorithm takes as input other variables.

**Definition 12 (Correctness).** *An* $\mathsf{AABC}$ *scheme satisfies perfect correctness, if the following properties hold for any* $\lambda > 0, q > 0$*, any non-empty sets* $A, D$ *such that* $|A| \leq q$ *and* $D \subseteq A$*, and* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q), (\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{OrgKGen}(\mathsf{pp}), (\mathsf{usk}, \mathsf{upk}) \leftarrow \mathsf{UsrKGen}(\mathsf{pp}), (\mathsf{cred}, \cdot) \leftarrow \langle \mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}, A), \mathsf{Issue}(\mathsf{upk}, \mathsf{osk}, A)\rangle$*: (1) the credential showing verifies, i.e.,* $(\cdot, 1) \leftarrow \langle \mathsf{Show}(\mathsf{opk}, A, D, \mathsf{cred}), \mathsf{Verify}(D)\rangle$*; (2) if the credential showing is accepted, the auditing verifies, i.e.,* $\mathsf{Audit}(\mathsf{ak}, S, \mathsf{opk}) = 1$ *for any* $(S, 1) \leftarrow \langle \mathsf{Show}, \mathsf{Verify}\rangle$*.*

For anonymity and unforgeability, we follow the approach given by [26], in which adversaries can corrupt some participants. We first introduce the following lists and oracles to model the adversary.

*Lists and oracles.* At the beginning of each experiment, either the experiment generates the key tuple $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak})$, or the adversary outputs $\mathsf{opk}$. The sets $\mathsf{HU}, \mathsf{CU}$ track all honest and corrupt users. We use the lists $\mathsf{USK}, \mathsf{UPK}, \mathsf{CRED}, \mathsf{ATTR}, \mathsf{OWNER}$ to track user-secret keys, user-public keys, issued credentials with the corresponding attribute sets, and the users who obtain the credentials. In the anonymity games, we use $J_{\mathsf{LoR}}, I_{\mathsf{LoR}}$ to store the issuance indices and the corresponding users that have been set during the first query to the left-or-right oracle. The adversary is required to guess a bit $b$.

Considering a PPT adversary $\mathcal{A}$, the oracles are listed in the following. Note that we add the $\mathcal{O}_{\mathsf{Audit}}$ oracle for the unforgeability experiment.

- $\mathcal{O}_{\mathsf{HU}}(i)$ takes as input a user index $i$. If $i \in \mathsf{HU} \cup \mathsf{CU}$, the oracle returns $\perp$; Otherwise, it creates a new honest user $i$ with $(\mathsf{USK}[i], \mathsf{UPK}[i]) \leftarrow \mathsf{UsrKGen}(\mathsf{pp})$ and adds the user to the honest user list $\mathsf{HU}$. It returns $\mathsf{UPK}[i]$ to the adversary.
- $\mathcal{O}_{\mathsf{CU}}(i, \mathsf{upk})$ takes as input $i$ and (optionally) a user public key $\mathsf{upk}$. If $i \in \mathsf{CU}$ or $i \in I_{\mathsf{LoR}}$, the oracle returns $\perp$; If $i \in \mathsf{HU}$, it moves $i$ from $\mathsf{HU}$ to $\mathsf{CU}$ and returns $\mathsf{USK}[i]$ and $\mathsf{CRED}[j]$ for all $j$ such that $\mathsf{OWNER}[j] = i$; If $i \notin \mathsf{HU} \cup \mathsf{CU}$, it adds $i$ to $\mathsf{CU}$ and sets $\mathsf{UPK}[i] = \mathsf{upk}$.
- $\mathcal{O}_{\mathsf{ObtIss}}(i, A)$ takes as input $i$ and a set of attributes $A$. If $i \notin \mathsf{HU}$, the oracle returns $\perp$; Otherwise, it generates a credential with $(\mathsf{cred}, \top) \leftarrow \langle \mathsf{Obtain}(\mathsf{USK}[i], \mathsf{opk}, A), \mathsf{Issue}(\mathsf{UPK}[i], \mathsf{osk}, A)\rangle$. If $\mathsf{cred} = \perp$, the oracle returns $\perp$; Otherwise, it adds $(i, \mathsf{cred}, A)$ to $(\mathsf{OWNER}, \mathsf{CRED}, \mathsf{ATTR})$ and returns $\top$.
- $\mathcal{O}_{\mathsf{Obtain}}(i, A)$ takes as input $i$ and $A$. If $i \notin \mathsf{HU}$, the oracle returns $\perp$; Otherwise, it runs $(\mathsf{cred}, \cdot) \leftarrow \langle \mathsf{Obtain}(\mathsf{USK}[i], \mathsf{opk}, A), \cdot\rangle$ by interacting with the

adversary $\mathcal{A}$ running $\mathsf{Issue}$. If $\mathsf{cred} = \perp$, the oracle returns $\perp$; Otherwise, it adds $(i, \mathsf{cred}, A)$ to $(\mathsf{OWNER}, \mathsf{CRED}, \mathsf{ATTR})$ and returns $\top$.

- $\mathcal{O}_{\mathsf{Issue}}(i, A)$ takes as input $i$ and $A$. If $i \notin \mathsf{CU}$, the oracle returns $\perp$; Otherwise, it runs $(\cdot, I) \leftarrow \langle \mathsf{Obtain}(\mathsf{USK}[i], \mathsf{opk}, A), \cdot \rangle$ by interacting with the adversary $\mathcal{A}$ running $\mathsf{Obtain}$. If $I = \perp$, the oracle returns $\perp$; Otherwise, it adds $(i, \perp, A)$ to $(\mathsf{OWNER}, \mathsf{CRED}, \mathsf{ATTR})$ and returns $\top$.
- $\mathcal{O}_{\mathsf{Show}}(j, D)$ takes in the index $j$ and a set of attributes $D$. Let $i = \mathsf{OWNER}[j]$, if $i \notin \mathsf{HU}$, the oracle returns $\perp$; Otherwise, it runs $(S, \cdot) \leftarrow \langle \mathsf{Show}(\mathsf{opk}, \mathsf{ATTR}[j], D, \mathsf{CRED}[j]), \cdot \rangle$ by interacting with the adversary $\mathcal{A}$ running $\mathsf{Verify}$.
- $\mathcal{O}_{\mathsf{Audit}}(S)$ is an oracle that holds public and auditing keys for all identified *issuers.* Given a showing transcript of a credential $S$, it runs $b \leftarrow \langle \mathsf{Show}, \mathsf{Verify} \rangle(S)$. If there exists $\mathsf{opk}$ and $\mathsf{ak}$ pair such that $\mathsf{Audit}(\mathsf{ak}, S, \mathsf{opk})$=1, the oracle returns $(\mathsf{opk}, b, 1)$ to the adversary; Otherwise, it returns $\perp$.
- $\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, D; b)$ takes as input two issuance indices $j_0, j_1$, a set of attributes $D$ and a challenge bit $b \xleftarrow{\$} \{0,1\}$. If $J_{\mathsf{LoR}} \neq \emptyset$ and $J_{\mathsf{LoR}} \neq \{j_0, j_1\}$, the oracle returns $\perp$. Let $i_0 = \mathsf{OWNER}[j_0], i_1 = \mathsf{OWNER}[j_1]$. If $J_{\mathsf{LoR}} = \emptyset$, it sets $J_{\mathsf{LoR}} = \{j_0, j_1\}, I_{\mathsf{LoR}} = \{i_0, i_1\}$. If $i_0, i_1 \notin \mathsf{HU}$ or $D \nsubseteq (\mathsf{ATTR}[j_0] \cap \mathsf{ATTR}[j_1])$, the oracle returns $\perp$; Otherwise, it runs $(S_b, \cdot) \leftarrow \langle \mathsf{Show}(\mathsf{opk}_b, \mathsf{ATTR}[j_b], D, \mathsf{CRED}[j_b]), \cdot \rangle$ by interacting with the adversary $\mathcal{A}$ running $\mathsf{Verify}$.

Then, the formal definitions are as follows.

**Definition 13 (Anonymity).** *An* AABC *scheme satisfies anonymity if for any PPT adversary $\mathcal{A}$ that has access to oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{LoR}}\}$, the following probability holds for any $\lambda, q > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$:*

$$\left| \Pr\left[ \begin{array}{ll} (\mathsf{opk}_0, \mathsf{opk}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}); & \\ b \xleftarrow{\$} \{0,1\}; & : b^* = b \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{st}) & \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Note that we modify the $\mathsf{Verify}$ in $\langle \mathsf{Show}, \mathsf{Verify} \rangle$ so that it does not take as input issuer-public keys. Hence, our anonymity also captures the indistinguishability of these keys. The definition above is arguably more close to the unlinkability from [6] because the $\mathcal{O}_{\mathsf{LoR}}$ oracle runs the $\mathsf{Show}$ algorithm with $\mathsf{opk}_b$ according to the challenge bit $b \xleftarrow{\$} \{0,1\}$.

**Definition 14 (Unforgeability).** *An* AABC *scheme satisfies unforgeability, if for any PPT adversary $\mathcal{A}$ that has access to oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{Audit}}\}$, the following probability holds for any $\lambda > 0, q > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{OrgKGen}(\mathsf{pp})$*

$$\Pr\left[ \begin{array}{ll} (D, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{opk}, \mathsf{ak}); & : b = 1 \wedge \mathit{If}\ \mathsf{OWNER}[j] {\in} \mathsf{CU}, \\ (S, b) \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathsf{Verify}(D) \rangle & \qquad\qquad D {\notin} \mathsf{ATTR}[j] \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Like APK, unforgeability regarding to auditing keys is needed. A user should not recover the auditing key of a given public key even after querying the auditing oracles on other key tuples for polynomial times. Since the adversary can run key generation on its own in APK, the auditing unforgeability of auditable ABC is equivalent to the auditing key unforgeability in Definition 10.

### 3.3 Our Constructions and Analysis

Our auditable ABC construction has the same approach of [18], relying on a structure-preserving signatures on equivalence classes (SPS-EQ) and a set-commit schemes. We extend their ABC construction with our APK mechanism.

**An MDDH-based APK construction.** In order to work with the ABC scheme (precisely, the SPS-EQ) given in [18], the setup algorithm $\mathsf{Setup}$ runs $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$ and samples a matrix $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_1$. It outputs $\mathsf{pp} \overset{\Delta}{=} (\mathsf{BG}, [\mathbf{A}]_2, \ell)$ where $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$, and $\ell$ is a parameter for message size in the SPS-EQ. We present a construction of APK based on group $(\mathbb{G}_2, P_2, p)$ where the MDDH and KerMDH assumptions are believed to hold.

**Construction 1 (MDDH-Based APK** $\mathsf{APK}$**)** *The rest of the algorithms are:*

- $\mathsf{KGen}(\mathsf{pp})$*: Sample matrices* $\mathbf{K}_0 \overset{\$}{\leftarrow} \mathcal{D}_{\ell,2}$ *and* $\mathbf{K}_1 \overset{\$}{\leftarrow} \mathbb{Z}_p^{2\times 2}$ *of full rank* 2*. Set* $\mathbf{K} = \mathbf{K}_0\mathbf{K}_1$*. Then, compute* $[\mathbf{B}]_2 = [\mathbf{K}_1\mathbf{A}]_2$ *and* $[\mathbf{C}]_2 = [\mathbf{K}\mathbf{A}]_2$*. Finally, set* $\mathsf{sk} = (\mathbf{K}_1, \mathbf{K}), \mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2), \mathsf{ak} = \mathbf{K}_0$ *and output* $(\mathsf{sk}, \mathsf{pk}, \mathsf{ak})$*;*
- $\mathsf{Update}(\mathsf{pk}; r)$*: Sample* $r \overset{\$}{\leftarrow} \mathbb{Z}_p$ *and compute* $[\mathbf{B}']_2 = r \cdot [\mathbf{B}]_2, [\mathbf{C}']_2 = r \cdot [\mathbf{C}]_2$*. Output* $\mathsf{pk}' = ([\mathbf{B}']_2, [\mathbf{C}']_2)$*;*
- $\mathsf{VerifyKP}(\mathsf{sk}, \mathsf{pk}', r)$*: Parse* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1)$*. Output* 1 *if* $\mathsf{pk}'_0 = r \cdot \mathsf{sk}_0 \cdot [\mathbf{A}]_2 \wedge \mathsf{pk}'_1 = r \cdot \mathsf{sk}_1 \cdot [\mathbf{A}]_2$*, or* 0 *otherwise;*
- $\mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak})$*: Parse* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$*. Output* 1 *if* $\mathsf{sk}_1 = \mathsf{ak} \cdot \mathsf{sk}_0$*, or* 0 *otherwise;*
- $\mathsf{Audit}(\mathsf{ak}, \mathsf{pk}', \mathsf{pk})$*: Parse* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1), \mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$*. Output* 1 *if* $\mathsf{pk}_1 = \mathsf{ak} \cdot \mathsf{pk}_0 \wedge \mathsf{pk}'_1 = \mathsf{ak} \cdot \mathsf{pk}'_0$*, or* 0 *otherwise.*

Hence, we have the following theorem.

**Theorem 1.** *The APK mechanism* $\mathsf{APK}$ *given by Construction 1 satisfies the following properties.*

- *Correctness (Definition 7);*
- *Indistinguishability (Definition 8) if the* $\mathcal{D}_{l,1}$*-MDDH assumption where* $l \in \{2, \ell\}$ *holds on* $\mathbb{G}_2$*;*
- *Secret key and auditing key unforgeability (Definition 9 and 10) if the* $\mathcal{D}_1$*-KerMDH holds on* $\mathbb{G}_2$*.*

*Proof.* On the additive cyclic group $\mathbb{G}_2$, APK *correctness* can be yielded directly from our construction. To prove *indistinguishability*, let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ where $\mathsf{pp} = (\mathsf{BG}, [\mathbf{A}]_2, \ell)$ are given as above. The reduction receives an MDDH challenge over $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{X}]_2, [\mathbf{z}]_2)$ where $\mathbf{X} \overset{\$}{\leftarrow} \mathcal{D}_{l,1}$. According the challenge bit $b \in \{0, 1\}$, $\mathbf{z}$ is set to $\mathbf{X}y$ with $y \overset{\$}{\leftarrow} \mathbb{Z}_p$ (when $b = 0$) or $\mathbf{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^l$ (when $b = 1$). $l$ takes its value from $\{2, \ell\}$ because the two components in a public key, $[\mathbf{B}]_2$ and $[\mathbf{C}]_2$, are matrices of size 2×1 and $\ell$×1, respectively. Note that the reduction needs to prepare both components of the public key. That is, it samples a full-ranked

$\mathbf{X}' \xleftarrow{\$} \mathbb{Z}_p^{l' \times l}$ such that $l' \in \{2, \ell\} \wedge l' \neq l$, and embeds the MDDH challenge $\mathsf{chl}$ by setting $\mathsf{pk}^* \overset{\Delta}{=} ([\mathbf{X}]_2, \mathbf{X}'[\mathbf{X}]_2)$ and $\mathsf{pk}' \overset{\Delta}{=} ([\mathbf{z}]_2, \mathbf{X}'[\mathbf{z}]_2)$. The indistinguishability adversary $\mathcal{A}$ takes as input $(\mathsf{pk}^*, \mathsf{pk}')$. If the challenge tuple satisfies $[\mathbf{z}]_2 = [\mathbf{X}y]_2$ (when $b = 0$), then $\mathsf{pk}'$ is distributed identically to $\mathsf{pk}_0$ ($\mathsf{pk}_0 \leftarrow \mathsf{Update}(\mathsf{pk}^*)$, the adversary will output $b^* = 0$). Otherwise (when $b = 1$), $\mathsf{pk}'$ is distributed identically to $\mathsf{pk}_1$ (a freshly generated public key, the adversary outputs $b^* = 1$). Therefore, the reduction has the same advantage in the $\mathcal{D}_{l,1}$-MDDH ($l \in \{2, \ell\}$) game as the adversary in the indistinguishability game of Definition 8.

The proofs of two types of unforgeability are similar. For *secret key unforgeability*, the reduction receives a KerMDH challenge over $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{A}]_2)$ where $\mathbf{A} \xleftarrow{\$} \mathcal{D}_1$. The reduction prepares the inputs for the unforgeability adversary $\mathcal{A}$. That is, it samples $\mathbf{K}_0 \xleftarrow{\$} \mathcal{D}_{\ell,2}$ and $\mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ of full rank 2. Then, let $[\mathbf{X}]_2 = [\mathbf{K}_1\mathbf{A}]_2$, the reduction embeds the challenge $\mathsf{chl}$ by setting $\mathsf{ak} \overset{\Delta}{=} \mathbf{K}_0$ and $\mathsf{pk} \overset{\Delta}{=} ([\mathbf{X}]_2, [\mathbf{K}_0\mathbf{X}]_2)$. Hence, the input to the adversary in the reduction is distributed identically as in the definition of unforgeability. Suppose the adversary $\mathcal{A}$ breaks secret key unforgeability, which means that $\mathsf{VerifyKP}, \mathsf{VerifyAK}, \mathsf{Audit}$ verify the output tuple $(\mathsf{sk}', \mathsf{pk}', r)$. More precisely, parse $\mathsf{sk}' = (\mathsf{sk}'_0, \mathsf{sk}'_1)$, it holds that $\mathsf{sk}'_0[\mathbf{A}]_2 = [\mathbf{X}]_2 = [\mathbf{K}_1\mathbf{A}]_2$ and $\mathsf{sk}'_1[\mathbf{A}]_2 = [\mathbf{K}_0\mathbf{X}]_2$. If the adversary can find a non-zero vector $\mathsf{sk}'_0 - \mathbf{K}_1$ in the kernel of $\mathbf{A}$, it can break the secret key unforgeability. However, finding $\mathsf{sk}'_0$ is equivalent to solving a $\mathcal{D}_1$-KerMDH problem (with $\mathsf{sk}'_0$, the reduction outputs $[\mathsf{sk}'_0 - \mathbf{K}_1]_1$ to the KerMDH challenge and $e([\mathsf{sk}'_0 - \mathbf{K}_1]_1, [\mathbf{A}]_2) = [0]_T$). Thus, the reduction advantage in the $\mathcal{D}_1$-KerMDH game is the same as the adversary in the secret key unforgeability game.

Similarly, the *auditing key unforgeability* reduction receives a KerMDH challenge over $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{X}]_2)$ where $\mathbf{X} \in \mathcal{D}_1$. The reduction samples $\mathbf{K}_0 \in \mathbb{Z}_p^{\ell\times 2}$ of full rank 2 and relays $(P_2, [\mathbf{X}]_2, [\mathbf{K}_0\mathbf{X}]_2)$ to the adversary. Hence, the input of the adversary, *i.e.*, $\mathsf{pk} = ([\mathbf{X}]_2, [\mathbf{K}_0\mathbf{X}]_2)$, distributes identically to the definition. Suppose the adversary $\mathcal{A}$ breaks auditing key unforgeability, which means it finds $\mathsf{ak}'$ such that $\mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak}') = 1$. Note that although the reduction cannot prepare the corresponding secret key, the structure preserves in the public key, *i.e.*, the adversary must output a non-zero $\mathsf{ak}' - \mathbf{K}_0$ in the kernel of $\mathbf{X}$. As explained before, the reduction cannot gain advantages in the $\mathcal{D}_1$-KerMDH game by invoking the auditing key unforgeability adversary.

**An auditable ABC construction.** Before we present the full construction of our auditable ABC scheme, we first recall briefly the SPS-EQ scheme from [18] (the construction and security definitions can be found in Appendix C). We will note that the key generation in their construction differs from our $\mathsf{APK.KGen}$. Hence, by further proving that the change only incurs slightly more advantage to the adversary in the original scheme, we show that our modification preserves the security definitions of the SPS-EQ. Moreover, as proven before, our modification also satisfies the security of the APK mechanism.

*Extending the SPS-EQ [18].* We show the original key generation of the SPS-EQ in the following. Recall that the $\mathsf{Setup}$ algorithm outputs $\mathsf{pp} = (\mathsf{BG}, [\mathbf{A}]_2, \ell)$.

- $\mathsf{SPSEQ.KGen}(\mathsf{pp})$: Sample matrices $\mathbf{K}_1 \overset{\$}{\leftarrow} \mathbb{Z}_p^{2\times 2}$ and $\mathbf{K} \overset{\$}{\leftarrow} \mathcal{D}_{\ell,2}$ of full rank 2. Then, compute $[\mathbf{B}]_2 = [\mathbf{K}_1\mathbf{A}]_2$ and $[\mathbf{C}]_2 = [\mathbf{K}\mathbf{A}]_2$. Finally, set $\mathsf{sk} = (\mathbf{K}_1, \mathbf{K}), \mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2)$ and output $(\mathsf{sk}, \mathsf{pk})$.

The only difference here is that we further sample $\mathbf{K}_0 \overset{\$}{\leftarrow} \mathcal{D}_{\ell,2}$ of full rank 2 and compute $\mathbf{K}$ by the multiplication of $\mathbf{K}_0$ and $\mathbf{K}_1$. In the following lemma, we show that this change only increases the SPS-EQ adversary's advantage by at most the advantage of solving a $\mathcal{D}_{\ell,2}$-MDDH problem over $\mathbb{G}_2$.

**Lemma 1.** *Replacing* $\mathsf{SPSEQ.KGen}$ *with* $\mathsf{APK.KGen}$ *in the* $\mathsf{SPSEQ}$ *scheme given in Construction 4 (Appendix C) preserves the correctness, EUF-CMA and perfect adaption of signatures with respect to message space of the original scheme.*

*Proof.* Correctness is straightforward as proven in Theorem 1. We unify the proofs of EUF-CMA and perfect adaption of signatures with respect to message space by considering a sequence of two games: $\mathsf{Game}_1$ is the EUF-CMA and perfect adaption of signatures with respect to message space games for the original SPS-EQ scheme with $\mathsf{SPSEQ.KGen}$ given in Definition 28 and 29 (in Appendix C); and $\mathsf{Game}_0$ substitutes $\mathsf{SPSEQ.KGen}$ with our APK mechanism's $\mathsf{APK.KGen}$. Hence, $\mathsf{Game}_0$ is the game for our modified scheme. We further denote the adversary $\mathcal{A}$'s advantage with $\mathsf{Adv}_i$ for each game $\mathsf{Game}_i$ where $\in \{0, 1\}$. In the transition of $\mathsf{Game}_0 \to \mathsf{Game}_1$, $\mathsf{pk}_{\mathsf{Game}_1} = ([\mathbf{K}_1\mathbf{A}]_2, [\mathbf{K}_{\mathsf{Game}_1}\mathbf{A}]_2)$ replaces $\mathsf{pk}_{\mathsf{Game}_0} = ([\mathbf{K}_1\mathbf{A}]_2, [\mathbf{K}_0\mathbf{K}_1\mathbf{A}]_2)$. Note that all matrices are of full rank 2, hence, distinguishing $\mathsf{pk}_{\mathsf{Game}_1}$ and $\mathsf{pk}_{\mathsf{Game}_0}$ is equivalent to solve a challenge of $\mathcal{D}_{\ell,2}$-MDDH problem (because $\mathbf{K}_{\mathsf{Game}_1}$ is an $\ell\times 2$ matrix of full rank 2). That is, $|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}^{\mathrm{MDDH}}_{\mathcal{D}_{\ell,2},\mathbb{G}_2}$. Moreover, as shown in [18], the original SPS-EQ scheme satisfies EUF-CMA and perfect adaption of signatures with respect to message space. We conclude Lemma 1.

*Constructing the auditable ABC.* Let $\mathsf{BGGen}$ be the bilinear group generation, $\mathsf{SC}$=($\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenSubset}, \mathsf{VerifySubset}$) be the set-commitment [26] that satisfies correctness, binding, subset-soundness and hiding (definitions given in Appendix B.2), and $\Pi$ be a general ZKPoK protocol that satisfies completeness, perfect zero-knowledge and knowledge-soundness (definitions given in Appendix B.3). With the necessary algorithms from our APK mechanism and the SPS-EQ [18], *i.e.*, ($\mathsf{KGen}, \mathsf{Update}, \mathsf{Audit}$)$\in$$\mathsf{APK}$ and ($\mathsf{Setup}, \mathsf{Sign}, \mathsf{ChgRep}, \mathsf{Verify}$)$\in$ $\mathsf{SPSEQ}$, we show an auditable ABC $\mathsf{AABC}$ in the following. Note that the SPS-EQ scheme [18] utilizes a non-interactive zero-knowledge argument (which we take as a black-box) under the common reference string (CRS) model.

**Construction 2 (Auditable ABC** $\mathsf{AABC}$**)** *The algorithms are as follows.*

- $\mathsf{Setup}(1^\lambda, q)$*: Run* $\mathsf{BG}$$\leftarrow$$\mathsf{BGGen}(1^\lambda)$ *where* $\mathsf{BG}$=$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. *Sample* $a \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ *and compute* $([a^i]_1, [a^i]_2)_{i\in[q]}$. *Sample matrices* $[\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1$

$\xleftarrow{\$} \mathcal{D}_1$*, and a common reference string* $\mathsf{crs}$ *for* $\mathsf{SPSEQ}$*. Output* $\mathsf{pp}{=}(\mathsf{BG}, ([a^i]_1, [a^i]_2)_{i\in[q]}, ([\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1), \mathsf{crs}, \ell{=}3)$*;*

- $\mathsf{OrgKGen}(\mathsf{pp})$*: Output* $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{APK.KGen}(\mathsf{BG}, [\mathbf{A}]_2, \ell)$ *and delegate* $\mathsf{ak}$ *to auditors selected by the issuer;*
- $\mathsf{UsrKGen}(\mathsf{pp})$*: Sample* $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$ *and output* $(\mathsf{usk}, \mathsf{upk} = \mathsf{usk}P_1)$*;*
- $\langle\mathsf{Obtain}, \mathsf{Issue}\rangle$ *and* $\langle\mathsf{Show}, \mathsf{Verify}\rangle$*: See Figure 1. In* $\langle\mathsf{Obtain}, \mathsf{Issue}\rangle$*, following the arguments in [18], we consider malicious issuer-keys and user-keys. Hence, both the issuer and the user should run a ZKPoK protocol to prove their public keys to each other, i.e.,* $\Pi^{\mathsf{osk}}(\mathsf{opk})$ *and* $\Pi^{\mathsf{usk}}(\mathsf{upk})$ *in Figure 1; Whereas, in* $\langle\mathsf{Show}, \mathsf{Verify}\rangle$*, the ZKPoK protocol, i.e.,* $(\pi_1, \pi_2, \pi_3){\leftarrow}\Pi^{(C, rC, P_1)}(C_1, C_2, C_3)$*, proves freshness to prevent transcripts of valid showings from being replayed by someone not in possession of the credential [26];*
- $\mathsf{Audit}(\mathsf{ak}, S, \mathsf{opk})$*: Parse* $S{=}(\mathsf{opk}', \mathsf{cred}', W; D)$*. Return* $\mathsf{APK.Audit}(\mathsf{ak}, \mathsf{opk}', \mathsf{opk})$*.*

Therefore, we have the following result.

**Theorem 2.** *The auditable ABC scheme* $\mathsf{AABC}$ *given by Construction 2 satisfies the following properties.*

- *Correctness (Definition 12);*
- *Anonymity (Definition 13) if the DDH assumption holds, the ZKPoK protocol has perfect zero-knowledge, the underlying APK satisfies indistinguishability and auditing key unforgeability, and the SPS-EQ scheme perfectly adapts signatures with respect to message space;*
- *Unforgeability (Definition 14) if the q-co-DL assumption holds, the ZKPoK protocol has perfect zero-knowledge, the set-commitment scheme* $\mathsf{SC}$ *satisfies subset-soundness, the APK satisfies secret key unforgeability, and SPS-EQ satisfies EUF-CMA;*
- *Auditing unforgeability (Definition 10) if the* $\mathcal{D}_1$*-KerMDH holds on* $\mathbb{G}_2$*.*

*Proof.* We show a brief proof here. *Correctness* follows directly from the correctness of building blocks. From now, we describe the proof rationale for anonymity, unforgeability, and auditing unforgeability properties, respectively. In general, we rely on [18] (Theorem 6 and 7) for properties of the ABC scheme and Theorem 1 for the APK part of our construction.

The proof for anonymity (Definition 13) is an adaptation of the one given in Theorem 7 of [18]. Note there are two slight modifications to the anonymity definition in our work: (1) the adversary in the anonymity game generates two issuer-public keys $(\mathsf{opk}_0, \mathsf{opk}_1)$; (2) the challenge oracle $\mathcal{O}_{\mathsf{LoR}}$ requires the adversary also to distinguish under which issuer-public key is the credential issued. To adjust the previous proof for these modifications, we first consider the two issuer-public keys $(\mathsf{opk}_0, \mathsf{opk}_1)$ and the updated issuer-public key in the credential showing, *i.e.*, $\mathsf{opk}_b' \leftarrow \mathsf{APK.Update}(\mathsf{opk}_b)$ where $b \in \{0, 1\}$ is the challenge bit in $\mathcal{O}_{\mathsf{LoR}}$. The adversary can win our introduced anonymity game if: (1) it can distinguish $\mathsf{opk}_0'$ from $\mathsf{opk}_1'$, then, the adversary can also win the *indistinguishability* game of the APK mechanism given in Definition 8; (2) it can recover the corresponding auditing keys $(\mathsf{ak}_0, \mathsf{ak}_1)$ from the issuer-public keys (hence, winning the anonymity game trivially by running $\mathsf{APK.Audit}(\mathsf{ak}_b^*, \mathsf{opk}'_b, \mathsf{opk}_b))$, which means the adversary wins the *auditing key unforgeability* game of the APK mechanism given in Definition 10. However, since the aforementioned APK properties have been proven in Theorem 1, our defined anonymity adversary gains no advantage over the anonymity adversary of [18].

Next, by the proof in [18], the anonymity of ABC requires that the DDH assumption holds, the ZKPoK protocol has perfect zero-knowledge (which is taken as a black-box in this work), and the SPS-EQ perfectly adapts signatures with respect to message space (which has been proven in Lemma 1), thus we conclude the anonymity part for our auditable ABC scheme.

Similarly, our proof of unforgeability (Definition 14) adapts the one given in Theorem 6 of [18]. The only modification we make is that the unforgeability

| $\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, A)$ | | $\mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, A)$ |
|---|---|---|
| | $\xleftrightarrow{\pi \leftarrow \Pi^{\mathsf{usk}}(\mathsf{upk})}$ | If $\Pi$ fails, return $\perp$ |
| If $\Pi$ fails, return $\perp$<br>$(C, O) \leftarrow \mathsf{SC.Commit}(A; \mathsf{usk})$; | $\xleftrightarrow{\pi \leftarrow \Pi^{\mathsf{osk}}(\mathsf{opk})}$ | |
| $r \xleftarrow{\$} \mathbb{Z}_p^*; R \overset{\Delta}{=} rC$; | $\xrightarrow{(C,R)}$ | If $e(C, P_2) \neq e(\mathsf{upk}, \mathsf{Ch}_A(a) P_2$<br>and $\forall a' \in A{:}[a']_1 = [a]_1$:<br>return $\perp$. Else return:<br>$(\sigma, \tau) \leftarrow \mathsf{SPSEQ.Sign}(\mathsf{osk}, (C, R, P_1))$ |
| | $\xleftarrow{(\sigma,\tau)}$ | |
| Check<br>$\mathsf{SPSEQ.Verify}(\mathsf{opk}, (C, R, P_1), (\sigma, \tau))$;<br>Return $\mathsf{cred} \overset{\Delta}{=} (C, (\sigma, \tau), r, O)$ | | |

| $\mathsf{AABC.Show}(\mathsf{opk}, A, D, \mathsf{cred})$ | | $\mathsf{AABC.Verify}(D)$ |
|---|---|---|
| Parse $\mathsf{cred} = (C, \sigma, r, O)$;<br>$\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$;<br>$((C_1, C_2, C_3), \sigma') \leftarrow \mathsf{SPSEQ.ChgRep}($<br>$(C, rC, P_1), (\sigma, \tau), \mu, \rho, \mathsf{opk})$;<br>$(C_1, C_2, C_3) \overset{\Delta}{=} \mu \cdot (C, rC, P_1)$;<br>$\mathsf{cred}' \overset{\Delta}{=} (C_1, C_2, C_3, \sigma')$;<br>$\mathsf{opk}' \leftarrow \mathsf{APK.Update}(\mathsf{opk}, \rho)$;<br>$O' \overset{\Delta}{=} (b, \mu \cdot O)$ where $b \in \{0, 1\}$;<br>$W \leftarrow \mathsf{SC.OpenSubset}(\mathsf{SC.pp}, \mu C, A, O', D)$<br>$S \overset{\Delta}{=} (\mathsf{opk}', \mathsf{cred}', W)$;<br>$(\pi_1, \pi_2, \pi_3) \leftarrow \Pi^{(C, rC, P_1)}(C_1, C_2, C_3)$ | | |
| | $\xleftrightarrow{(S, \pi_1, \pi_2, \pi_3)}$ | |
| | | If $\Pi$ fails, return 0; Else return:<br>$\mathsf{SPSEQ.Verify}(\mathsf{opk}', \mathsf{cred}') \wedge$<br>$\mathsf{SC.VerifySubset}(C_1, D, W)$ |

**Fig. 1.** $\langle \mathsf{Obtain}, \mathsf{Issue} \rangle$, $\langle \mathsf{Show}, \mathsf{Verify} \rangle$ in AABC.

game provides its adversary with auditing keys. With additional auditing keys, the adversary can win the unforgeability game if it can recover the secret key of the underlying APK mechanism, which violates the *secret key unforgeability* of APK as proven in Theorem 1. Then, by the proof in [18], the unforgeability of ABC requires that the $q$-co-DL assumption holds, the ZKPoK protocol has perfect zero-knowledge, the set-commitment scheme $\mathsf{SC}$ satisfies subset-soundness (which is also taken as a black-box in this work), and SPS-EQ satisfies EUF-CMA (which has been proven in Lemma 1), thus we conclude unforgeability for our auditable ABC scheme.

Finally, for the auditing unforgeability property, the adversary in the auditing unforgeability game aims to recover the issuer-public key's corresponding auditing key. Hence, the definition is equivalent to the auditing key unforgeability of APK (we use the same definition), which has been proven in Theorem 1.

## 4 Application: Contact Tracing

From the perspective of credentials, we review the environmental-adaptive contact tracing (EACT) framework proposed in [38]. We provide a construction based on our auditable ABC scheme and argue that the game-based security definitions of auditable ABC suffice the requirements in contact tracing systems. Finally, we implement our construction to showcase its practicality.

**Overview.** We start by recalling the EACT framework [38]. It utilizes a bulletin board to store contact records, which can be instantiated by a blockchain protocol satisfying the robust ledger properties [27], *i.e.*, the capability of achieving immutable consensus atomically. Concerning different virus transmission modes (droplet and airborne), EACT considered tracing approaches via Bluetooth Low Energy (BLE) and self-reported discrete location (DLT). However, the framework cannot unify the tracing approach in both settings because the recorded data are of different structures. As we will show later, ABC schemes enable us to circumvent this problem by regarding environmental and location data as *attributes*. Here, for completeness, we define a comparison algorithm to decide close contacts for BLE and DLT, *i.e.*, $\mathsf{Compare}_{\{\mathrm{BLE},\mathrm{DLT}\}}(\mathsf{envpp}, D, A)$ takes as input the environmental parameters $\mathsf{envpp}$, an opened attributed set $D$ (from other users, potentially downloaded from the bulletin board) and an attribute set $A$ (of the user who runs the algorithm). We say the algorithm is "*well-defined*" if it outputs 1 when attributes in $D$ and $A$ are regarded as close contact concerning the tracing setting in $\{\mathrm{BLE}, \mathrm{DLT}\}$, and 0 otherwise.

The EACT framework involves three phases: key management, recording, and tracing, with two types of participants: user $\mathcal{U}$ and medical agency $\mathcal{M}$. We refine the algorithms with respect to our auditable ABC scheme. Intuitively, in the key management phase, users generate key pairs for participating as both issuers and regular users. The medical agency generates its key pair from a signature scheme. Users need to register their *issuer-public keys*, and medical agencies need to register their public keys. Then, in the recording phase, when users contact (two users in BLE or one user in DLT), we consider a pairwise executed $\langle \mathsf{Obtain}, \mathsf{Issue} \rangle$, *i.e.*, each user performs as an ABC issuer to grant its counterparty (itself in DLT) a credential on the attributes of current environmental data (or location data). This approach in the DLT setting can be easily adapted to the case in which the user can communicate to BLE beacons, providing additional evidence for the user's location data. Finally, in the tracing phase, whenever a user searches for medical treatment, she shows her records while the agency performs verification. The medical agency uploads records to the bulletin board (as we will show in tracing-soundness (Definition 16), we assume a malicious agency who can upload invalid records). Hence, users can refer to the bulletin board and audit if any shown credential is issued by themselves. Then, by comparing the environmental factors, they can detect close contact. The following section presents the full construction, including our modifications to the original framework.

### 4.1 An Auditable ABC-Based Construction

Let $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme that satisfies correctness and EUF-CMA (definitions given in Appendix B.1), and let $\mathsf{AABC}$ be our auditable ABC construction given in Construction 2.

**Construction 3 (Refined EACT** $\mathsf{REACT}$**)** *Our refined EACT framework involves three phases, i.e., Key management:* $(\mathsf{Setup}, \mathsf{OrgKGen}, \mathsf{UsrKGen}, \mathsf{MedKGen}, \mathsf{KReg})$*; Recording:* $\mathsf{Exchange}$*; Tracing:* $(\langle \mathsf{Show}, \mathsf{Verify} \rangle, \mathsf{Merge}, \mathsf{Trace})$*. The algorithms are performed as follows.*

- $\mathsf{Setup}(1^\lambda, q, \mathsf{envpp})$ *is run by the system where* $\mathsf{envpp}$ *denotes the environmental parameters. It runs* $\mathsf{AABC.pp} \leftarrow \mathsf{AABC.Setup}(1^\lambda, q)$ *and outputs* $\mathsf{pp} = (\mathsf{AABC.pp}, \mathsf{envpp})$.
- $\mathsf{OrgKGen}(\mathsf{pp})$ *is run by a user and outputs* $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{AABC.OrgKGen}(\mathsf{pp})$. *Note that in contact tracing, we consider the user auditing for herself;*
- $\mathsf{UsrKGen}(\mathsf{pp})$ *is run by a user and outputs* $(\mathsf{usk}, \mathsf{upk}) \leftarrow \mathsf{AABC.UsrKGen}(\mathsf{pp})$*;*
- $\mathsf{MedKGen}(\mathsf{pp})$ *is run by a medical agency. It outputs a medical agent key pair with* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{SIG.KGen}(1^\lambda)$. *Later, we omit* $\mathsf{pp}$ *in algorithm inputs;*
- $\mathsf{KReg}(\mathsf{pk}, \mathsf{misc}, \mathbb{B})$ *is a DID [36] black-box, which takes as input a public key* $\mathsf{pk} \in \{\mathsf{opk}, \mathsf{mpk}\}$*, auxiliary information* $\mathsf{misc}$*, and a bulletin board* $\mathbb{B}$*.* $\mathsf{KReg}$ *registers* $\mathsf{pk}$ *with the corresponding* $\mathsf{misc}$ *on* $\mathbb{B}$*.*
- $\mathsf{Exchange}(\{(\mathsf{osk}_i, \mathsf{opk}_i), (\mathsf{usk}_i, \mathsf{upk}_i), A_i\}_{i \in \{0,1\}})$ *is an interactive protocol executed between two users* $\mathcal{U}_0, \mathcal{U}_1$*, who may be identical, e.g., in the DLT setting. For* $i \in \{0, 1\}$*, both users perform* $(\mathsf{cred}_i, \cdot) \leftarrow \langle \mathsf{Obtain}(\mathsf{usk}_i, \mathsf{opk}_{1-i}, A_i), \mathsf{Issue}(\mathsf{upk}_i, \mathsf{osk}_{1-i}, A_i) \rangle$ *to grant each other a credential. The protocol outputs* $\mathsf{cred}_0$ *and* $\mathsf{cred}_1$ *for each user, respectively.*
- $\langle \mathsf{Show}, \mathsf{Verify} \rangle$ *is the showing and verification protocol in our auditable ABC, which here, is executed between a user* $\mathcal{U}$ *and a medical agency* $\mathcal{M}$*. The protocol outputs* $(S, b) \leftarrow \mathsf{AABC}.\langle \mathsf{Show}, \mathsf{Verify} \rangle$ *where* $S$ *is a showing of the credential and* $b \in \{0, 1\}$*. Note that we explicitly add revealed attributes to* $S$*, i.e.,* $S \overset{\Delta}{=} (\mathsf{opk}', \mathsf{cred}', W, D)$*. Moreover, we enable this protocol to process in batches, i.e., it can take a list of* $n$ *credentials and verifies for each entry;*

- $\mathsf{Merge}(\mathsf{msk}, (S, b), \mathbb{B})$ *is run by a medical agency* $\mathcal{M}$. *If* $b = 1$, $\mathsf{Merge}$ *runs* $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{msk}, S)$ *and outputs* $\mathbb{B}||(\mathsf{mpk}, S, \sigma)$, *or aborts otherwise;*
- $\mathsf{Trace}(\mathsf{ak}, A, \mathbb{B})$ *is run by a user* $\mathcal{U}$ *with issuer-public and auditing keys* $\mathsf{opk}, \mathsf{ak}$. *It parses* $\mathbb{B} = \{(\mathsf{mpk}_j, S_j, \sigma_j)\}_{j\in[|\mathbb{B}|]}$, *and for each entry, parses* $S_j = (\mathsf{opk}'_j, \mathsf{cred}'_j, W_j, D_j)$. *Then, for each entry, it runs* $b \leftarrow \mathsf{SIG.Verify}(\mathsf{mpk}_j, S_j, \sigma_j)$, *and* $b' \leftarrow \mathsf{AABC.Audit}(\mathsf{ak}, S_j, \mathsf{opk})$ *(which is* $\mathsf{APK.Audit}(\mathsf{ak}, \mathsf{opk}'_j, \mathsf{opk})$*). For all* $j{\in}[|\mathbb{B}|]$ *such that* $b{=}1{\wedge}b'{=}1$, *it compares according to environmental parameters and tracing settings, i.e.,* $b_j \leftarrow \mathsf{Compare}_{\{BLE,DLT\}}(\mathsf{envpp}, D_j, A)$. *If there exists any* $j$ *that satisfies* $b_j = 1$, $\mathsf{Trace}$ *outputs* $1$*; Otherwise, it outputs* $0$.

### 4.2 Security and Analysis

We directly employ the cryptographic game-based security definitions from our auditable ABC scheme given in Section 3.2, including correctness, anonymity, and unforgeability. Moreover, we consider two separate properties for tracing, *i.e.*, traceability and tracing-soundness. At the end of this section, we will compare our ABC-based security definitions to the ones in [19].

The refined EACT requires signatures from medical agencies in $\mathsf{Merge}$ and on the bulletin board $\mathbb{B}$ (satisfying robust ledger properties [27]). Hence, we first formalize the tracing process correctness to capture these new requirements.

**Definition 15 (Traceability).** *Given the bulletin board* $\mathbb{B}$, *a* $\mathsf{REACT}$ *system satisfies traceability, if for any* $\lambda > 0, q > 0$, *any non-empty sets* $A$ *with* $|A| \leq q$, *and for any honest user* $\mathcal{U}$ *with a key tuple* $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \overset{\$}{\leftarrow} \mathsf{OrgKGen}(\mathsf{pp})$ *where* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^q)$, *if there exists* $(\mathsf{mpk}, S, \sigma) \in \mathbb{B}$ *such that* $\langle \mathsf{Show}, \mathsf{Verify}\rangle(S) = 1$, $\mathsf{SIG.Verify}(\mathsf{mpk}, S, \sigma) = 1$, $D \in S$ *such that* $\mathsf{Compare}_{\{BLE,DLT\}}(\mathsf{envpp}, D, A) = 1$, *then* $\Pr[\mathsf{Trace}(\mathsf{ak}, A, \mathbb{B}) = 1] = 1$ *where* $A$ *is the attribute set of* $\mathcal{U}$ *when she issues the credential being shown in* $S$.

Then, we have the following lemma.

**Lemma 2.** *Let the bulletin board satisfy the robust ledger properties [27]. The refined EACT* $\mathsf{REACT}$ *given by construction 3 satisfies traceability if* $\mathsf{AABC}$ *and* $\mathsf{SIG}$ *satisfy correctness, and the* $\mathsf{Compare}$ *algorithm is well-defined.*

The proof follows directly from the correctness of $\mathsf{AABC}$ and $\mathsf{SIG}$, and the well-defined comparing algorithm $\mathsf{Compare}_{\{\mathrm{BLE,DLT}\}}(\mathsf{envpp}, \cdot, \cdot)$. Moreover, we require the bulletin board to satisfy the robust ledger properties [27] so that any entry on it cannot be erased or modified after a period of time.

Next, we consider the soundness of tracing, *i.e.*, the situation in which an honest user's $\mathsf{Trace}$ outputs 1 falsely. The PPT adversary $\mathcal{A}$ either: (1) forges a valid credential on behalf of honest users; or (2) colludes with a malicious medical agency so that arbitrary showings can be uploaded to the bulletin board. The first case has been captured by our unforgeability game in the auditable ABC scheme (Definition 14) with additional assumptions for the bulletin board, signature scheme, and comparing algorithm (like in Lemma 2).

However, the second one is dedicated to contact tracing. The reason lies in the different use cases, *i.e.*, in auditable ABC, auditors audit credential showings on behalf of the original issuer, hence, triggering the auditing algorithm of another auditor gains the adversary no benefits; whereas, in contact tracing, it will cause false positive errors to the original issuer. In order to prevent such an attack, we require the proof in $\mathsf{AABC}.\langle\mathsf{Show},\mathsf{Verify}\rangle$ to be non-interactive. Concretely, since our bulletin board is instantiated by a blockchain, users can apply the Fiat-Shamir transformation on the head block of the blockchain, *i.e.*, embedding the hash of the blockchain's head block into their proof. Note that this approach also guarantees the freshness of the proof that prevents the relay and replay attacks [17]. This is because the adversary cannot guess the head of the blockchain priorly due to the security of blockchain protocols.

As shown in Theorem 2, the anonymity and unforgeability of $\mathsf{AABC}$ (also for $\mathsf{REACT}$ in Theorem 3) requires perfect zero-knowledge of $\Pi$. Hence, we must rely on heavy mechanisms, *e.g.*, [30], to make such a protocol non-interactive. An alternative way is to prove these theorems with computational zero-knowledge with a looser security reduction. The transformation to a non-interactive protocol with computational zero-knowledge can be achieved with the Fiat-Shamir heuristic [25] to trade security tightness for efficiency. Then, the showing of a credential becomes publicly verifiable so that even if a malicious medical agency falsely uploads credential showings to the bulletin board, every user (including the one who runs $\mathsf{Trace}$) can verify the showing.

Compared to the unforgeability of auditable ABC in Definition 14, due to the malicious $\mathcal{A}_{\mathcal{M}}$ setting, tracing-soundness removes the requirement of $b = 1$ (the credential showing can be invalid) but embeds the proof of freshness (the showing must be presented at most once). We formally define tracing-soundness (with respect to malicious $\mathcal{A}_{\mathcal{M}}$).

**Definition 16 (Tracing-Soundness).** *Given the bulletin board $\mathbb{B}$, a $\mathsf{REACT}$ system satisfies tracing-soundness (with respect to malicious $\mathcal{A}_{\mathcal{M}}$), if for any PPT adversary $\mathcal{A}$ that has access to oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{Audit}}\}$, the following probability holds for any $\lambda > 0, q > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda}, q, \mathsf{envpp})$, and $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{OrgKGen}(\mathsf{pp})$:*

$$\Pr\left[\begin{array}{ll}(D, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{opk}, \mathsf{ak}); & \mathsf{Trace}(\mathsf{ak}, \cdot, (S, \pi)) = 1 \wedge \\ ((S, \pi), b) \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathsf{Verify}(D)\rangle : & \mathit{If}\ \mathsf{OWNER}[j] {\in} \mathsf{CU}, D {\notin} \mathsf{ATTR}[j]\end{array}\right] \leq \mathsf{negl}(\lambda),$$

*where $\pi = (\pi_1, \pi_2, \pi_3) \leftarrow \Pi^{(C, rC, P_1)}(C_1, C_2, C_3)$, and the variables are given in Figure 1 of auditable ABC construction.*

Finally, we have the following theorem.

**Theorem 3.** *Our refined EACT $\mathsf{REACT}$ satisfies correctness, anonymity, unforgeability, traceability, and tracing-soundness (with respect to malicious $\mathcal{A}_{\mathcal{M}}$).*

*Proof.* The proofs of correctness, anonymity, and unforgeability follow directly from the underlying auditable ABC scheme, which has been given in Theorem 2.

Traceability has been captured by Lemma 2. Hence, we only show the proof of tracing-soundness with respect to malicious $\mathcal{A}_{\mathcal{M}}$.

Let $(\mathsf{mpk}, (S, \pi)\sigma) \in \mathbb{B}$ be an entry stored on the bulletin board that triggers an honest user's tracing algorithm, *i.e.*, $\mathsf{Trace}(\mathsf{ak}, A, (\mathsf{mpk}, (S, \pi), \sigma)) = 1$. Here, $\mathsf{ak}$ and $A$ are the user's auditing key and attribute set when issuing the shown credential; whereas, $\mathsf{mpk}$ and $\sigma$ are the public key and signature of the malicious $\mathcal{A}_{\mathcal{M}}$. Now, we consider the adversary's outputs, $((S, \pi), b) \leftarrow \langle \mathcal{A}(\mathsf{st}), \mathsf{Verify}(D) \rangle$. By traceability (Lemma 2), if $b = 1$, then the $\mathsf{Trace}$ algorithm will output 1. However, by the unforgeability of the underlying auditable ABC, the probability of the adversary outputting $b = 1$ is negligible of $\lambda$. In contrast, since we assume the medical agency $\mathcal{A}_{\mathcal{M}}$ to be malicious, it may approve invalid showing transcripts, *i.e.*, $b = 0$, to be uploaded to the bulletin board. However, as discussed above, the non-interactive proof $\pi$ enhanced the showing transcript $S$ to be publicly verifiable (freshness and validity). Hence, the user who runs $\mathsf{Trace}$ can reproduce $b' \leftarrow \langle \mathsf{Show}, \mathsf{Verify} \rangle (S, \pi)$ on herself. Considering the situation where $b = 0$, we should have $b' = 0$. Therefore, $\mathsf{Trace}$ will also output 0, *i.e.*, the adversary fails to break tracing-soundness even if it colludes with a malicious medical agency.

**Comparison with existing game-based security definitions in [19].** The authors consider unlinkability and integrity. The former requires the adversary to distinguish two given pseudonyms, and the latter focuses on the false positive attack, *i.e.*, the adversary tries to trigger an honest user's tracing algorithm to output 1. Given the differences in syntax (their model [19] only considers pseudonyms; whereas, our system considers showing of credentials), we compare our definitions with theirs as follows.

Concretely, the unlinkability in [19] is further separated into pseudonym unlinkability (during the recording phase in which other users' pseudonyms are stored locally) and trace unlinkability (during the tracing phase in which recorded pseudonyms become publicly available). In comparison, the anonymity of our auditable ABC schemes (Definition 13) guarantees that no adversary can distinguish any two credential showings (similar to the revealed pseudonyms in trace unlinkability), even when the adversary has full control over issuing the challenge credentials. Note that the adversary can control issuer-public keys but not the corresponding auditing keys (this is guaranteed by the auditing key unforgeability of the underlying APK mechanism). Otherwise, it can trivially win the anonymity game by auditing the updated public keys in credential showings (as we explained in the proof of Theorem 2). Moreover, user-public keys (similar to the pseudonyms in [19]) are generated with $\mathsf{AABC.UsrKGen}$, which are distributed uniformly on $\mathbb{G}_1$ (since $\mathsf{usk}$ is sampled uniformly at random from $\mathbb{Z}_p^*$). Hence, no adversary, as in the pseudonym unlinkability game, can distinguish any two distinct user-public keys in our system. Therefore, our anonymity captures both of the unlinkability definitions.

Next, the integrity in [19] provides its adversary with oracles to: (1) create new users; (2) generate, record, and send pseudonyms (during the recording phase, similar to credentials in our system); (3) generate and upload pseudonyms for tracing (during tracing phase, similar to our credential showings); (4) set time periods. In comparison, our tracing-soundness (Definition 16) provides similar oracle accessibility to the adversary, *i.e.*, (1) create new users; (2) issue and obtain credentials (contact records); (3) show credentials and upload the showing. Hence, it resembles the integrity in [19] except for the oracle that sets time for the adversary (this is because our system model is not period-specific like the one in [19]).

### 4.3 Implementation

We provide a proof-of-concept implementation for the refined EACT construction to prove its practicality on mobile devices with comparatively limited performance. The implementation uses Java/Kotlin for the raw Android environment. However, we also implement necessary functions since the Java Pairing-Based Cryptography (jPBC) library [15] cannot fully support matrix-based bilinear pairing operations. The library-level implementation, together with extended parts for jPBC [15] library, can also be found in our anonymous repository (`https://anonymous.4open.science/r/EAHT_MODULE_TEST`).

Overall, we implement the following algorithms (Setup, OrgKGen, UsrKGen, Exchange) $\in$ REACT. Moreover, in Exchange, we need to measure the performance of algorithms and transmission (which is written in the form of Transmit$(\cdot)$ for simplicity), separately. Hence, we further divide Exchange into (Obtain-1, Transmit$_1$, AABC.Issue, Transmit$_2$, Obtain-2). The results are shown in Table 1.

**Table 1.** Experiment Results (Time in milliseconds)

| Algorithms | Time | Algorithms | Time | Algorithms | Time |
|---|---|---|---|---|---|
| Setup | 168.99 | Obtain-1 | 40.08 | Transmit$(\sigma, \tau)$ | 75.16 |
| OrgKGen | 54.18 | Transmit$(\pi, C, R)$ | 38.32 | Obtain-2 | 164.81 |
| UsrKGen | 9.05 | AABC.Issue | 257.50 | GenProof | 0.26 |

Experiment device: Samsung SM-S9080 Android 12, Bluetooth 5.0 (Bluetooth Low Energy); Time consumption is presented in milliseconds and calculated with the average of 100 attempts.

Setup and OrgKGen are performance-insensitive because they only need to be executed once. We implement them merely to support other algorithms. Although we do not require user key pairs to be renewed once per contact, UsrKGen should be run periodically (*e.g.*, once per hour) to prevent a user's complete track under its public key from being exposed. We leave the setting of the renewal interval for real-life users to decide. Finally, for Exchange, we consider the performance of AABC.Obtain = (Obtain-1, Obtain-2), AABC.Issue and the time cost of data transmission, *i.e.*, Transmit$(\pi, C, R)$ and Transmit$(\sigma, \tau)$. A one-sided round trip, *e.g.*, $\mathcal{U}_0$ issuing a credential to $\mathcal{U}_1$ is performed with ($\mathcal{U}_0$.Obtain-1 $\longrightarrow$

$\mathcal{U}_0.\mathsf{Transmit}(\pi, C, R) \longrightarrow \mathcal{U}_1.\mathsf{Issue} \longrightarrow \mathcal{U}_1.\mathsf{Transmit}(\sigma, \tau) \longrightarrow \mathcal{U}_0.\mathsf{Obtain\text{-}2})$ takes approximately 575.87 milliseconds in total. Consider the worst case, *e.g.*, when a crowded train is filled with 101 users. Each of them needs to Exchange with the other 100, hence, taking approximately 57.6 seconds to finish the execution and transmission. We consider this result to be reasonable and plausible.

## 5 Conclusion

Motivated by the contact tracing new requirements, we adopt a novel approach from ABC schemes due to their similarity. By abstracting "traceability" in contact tracing systems, we propose an auditable public key (APK) mechanism that, like its predecessor, the updatable public keys can be applied in many cryptographic primitives, making it of independent interest.

Next, we extend the ABC schemes in [26, 18] with our APK mechanism to port the auditability to the world of ABC. Such property enables auditors, delegated by an issuer, to audit if a shown credential is issued by the issuer. We argue that it adds an additional layer of accountability to the schemes in which credential showings can hide issuer identities. The capability of hiding issuer identities is usually considered an overpowerful anonymity property in real- ife. The auditability for identifying issuers may also help credential revocation, which has been another long-worried problem of credentials schemes.

Finally, our refined EACT framework fixes the problems in the original work [38], *i.e.*, (1) distinct tracing approaches for different settings; (2) weak security guarantee from informal threat models. We achieve so by constructing it from our auditable ABC and adapting security properties accordingly. Moreover, we clarify that EACT is only one example application for our auditable primitives (public keys and ABC).

## References

1. AISEC, F.: Pandemic contact tracing apps: Dp-3t, PEPP-PT ntk, and ROBERT from a privacy perspective. IACR Cryptol. ePrint Arch. p. 489 (2020), https://eprint.iacr.org/2020/489
2. Apple Inc. and Google LLC: Privacy-preserving contact tracing. https://covid19.apple.com/contacttracing (2020 [Online])
3. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. IACR Cryptol. ePrint Arch. p. 385 (2005), http://eprint.iacr.org/2005/385
4. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 108–125. Springer (2009), https://doi.org/10.1007/978-3-642-03356-8_7

5. Beskorovajnov, W., Dörre, F., Hartung, G., Koch, A., Müller-Quade, J., Strufe, T.: Contra corona: Contact tracing against the coronavirus by bridging the centralized-decentralized divide for stronger privacy. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13091, pp. 665–695. Springer (2021). https://doi.org/10.1007/978-3-030-92075-3“˙23, https://doi.org/10.1007/978-3-030-92075-3_23
6. Bobolz, J., Eidens, F., Krenn, S., Ramacher, S., Samelin, K.: Issuer-hiding attribute-based credentials. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security - 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13099, pp. 158–178. Springer (2021), https://doi.org/10.1007/978-3-030-92548-2_9
7. Bogatov, D., Caro, A.D., Elkhiyaoui, K., Tackmann, B.: Anonymous transactions with revocation and auditing in hyperledger fabric. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security - 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13099, pp. 435–459. Springer (2021), https://doi.org/10.1007/978-3-030-92548-2_23
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 416–432. Springer (2003). https://doi.org/10.1007/3-540-39200-9“˙26, https://doi.org/10.1007/3-540-39200-9_26
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Advances in Cryptology - ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 514–532. Springer (2001). https://doi.org/10.1007/3-540-45682-1“˙30, https://doi.org/10.1007/3-540-45682-1_30
10. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pedersen, M.Ø.: Formal treatment of privacy-enhancing credential systems. In: Dunkelman, O., Keliher, L. (eds.) Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9566, pp. 3–24. Springer (2015). https://doi.org/10.1007/978-3-319-31301-6“˙1, https://doi.org/10.1007/978-3-319-31301-6_1
11. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 93–118. Springer (2001), https://doi.org/10.1007/3-540-44987-6_7
12. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers. Lecture Notes in Computer Science, vol. 2576, pp. 268–289. Springer (2002), https://doi.org/10.1007/3-540-36413-7_20

13. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M.K. (ed.) Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3152, pp. 56–72. Springer (2004), https://doi.org/10.1007/978-3-540-28628-8_4
14. Canetti, R., Kalai, Y.T., Lysyanskaya, A., Rivest, R.L., Shamir, A., Shen, E., Trachtenberg, A., Varia, M., Weitzner, D.J.: Privacy-preserving automated exposure notification. IACR Cryptol. ePrint Arch. p. 863 (2020), https://eprint.iacr.org/2020/863
15. Caro, A.D., Iovino, V.: jpbc: Java pairing based cryptography. In: Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28 - July 1, 2011. pp. 850–855. IEEE Computer Society (2011), https://doi.org/10.1109/ISCC.2011.5983948
16. Chen, L.D.: Effects of ambient temperature and humidity on droplet lifetime – a perspective of exhalation sneeze droplets with COVID-19 virus transmission. International Journal of Hygiene and Environmental Health (2020). https://doi.org/10.1016/j.ijheh.2020.113568
17. Connolly, A., Deschamps, J., Lafourcade, P., Perez-Kempner, O.: Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In: Isobe, T., Sarkar, S. (eds.) Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13774, pp. 249–271. Springer (2022), https://doi.org/10.1007/978-3-031-22912-1_11
18. Connolly, A., Lafourcade, P., Perez-Kempner, O.: Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13177, pp. 409–438. Springer (2022), https://doi.org/10.1007/978-3-030-97121-2_15
19. Danz, N., Derwisch, O., Lehmann, A., Pünter, W., Stolle, M., Ziemann, J.: Security and privacy of decentralized cryptographic contact tracing. IACR Cryptol. ePrint Arch. p. 1309 (2020), https://eprint.iacr.org/2020/1309
20. Das, S.K., Alam, J.E., Plumari, S., Greco, V.: Transmission of airborne virus through sneezed and coughed droplets (Sep 2020), https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7513825/
21. for Disease Control, C., Prevention: Transmission-based precautions (2016), https://www.cdc.gov/infectioncontrol/basics/transmission-based-precautions.html#anchor_1564058235
22. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.L.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 129–147. Springer (2013), https://doi.org/10.1007/978-3-642-40084-1_8
23. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: A new design for anonymous cryptocurrencies. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory

and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 649–678. Springer (2019), `https://doi.org/10.1007/978-3-030-34578-5\_23`

24. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 526–544. Springer (1989), `https://doi.org/10.1007/0-387-34805-0\_46`
25. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986), `https://doi.org/10.1007/3-540-47721-7\_12`
26. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. J. Cryptol. **32**(2), 498–546 (2019), `https://doi.org/10.1007/s00145-018-9281-4`
27. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 281–310. Springer (2015), `https://doi.org/10.1007/978-3-662-46803-6\_10`
28. Garman, C., Green, M., Miers, I.: Decentralized anonymous credentials. In: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014. The Internet Society (2014), `https://www.ndss-symposium.org/ndss2014/decentralized-anonymous-credentials`
29. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988), `https://doi.org/10.1137/0217017`
30. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4004, pp. 339–358. Springer (2006), `https://doi.org/10.1007/11761679\_21`
31. Han, Z., Weng, W., Huang, Q.: Characterizations of particle size distribution of the droplets exhaled by sneeze. Journal of the Royal Society, Interface / the Royal Society **10**, 20130560 (11 2013). https://doi.org/10.1098/rsif.2013.0560
32. Hébant, C., Pointcheval, D.: Traceable constant-size multi-authority credentials. In: Galdi, C., Jarecki, S. (eds.) Security and Cryptography for Networks - 13th International Conference, SCN 2022, Amalfi, Italy, September 12-14, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13409, pp. 411–434. Springer (2022), `https://doi.org/10.1007/978-3-031-14791-3\_18`
33. Jones, M., Sporny, M., Terbu, O., Cohen, G., Steele, O.: Verifiable credentials data model v2.0. W3C working draft, W3C (Jul 2023), https://www.w3.org/TR/2023/WD-vc-data-model-2.0-20230718/
34. Liu, J.K., Au, M.H., Yuen, T.H., Zuo, C., Wang, J., Sakzad, A., Luo, X., Li, L.: Privacy-preserving COVID-19 contact tracing app: A zero-knowledge proof approach. IACR Cryptol. ePrint Arch. p. 528 (2020), `https://eprint.iacr.org/2020/528`

35. Morillo, P., Ràfols, C., Villar, J.L.: The kernel matrix diffie-hellman assumption. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 729–758 (2016), https://doi.org/10.1007/978-3-662-53887-6_27
36. Reed, D., Sporny, M., Sabadello, M., Guy, A.: Decentralized identifiers (DIDs) v1.0. W3C recommendation, W3C (Jul 2022), https://www.w3.org/TR/2022/REC-did-core-20220719/
37. Silde, T., Strand, M.: Anonymous tokens with public metadata and applications to private contact tracing. In: Eyal, I., Garay, J.A. (eds.) Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13411, pp. 179–199. Springer (2022), https://doi.org/10.1007/978-3-031-18283-9_9
38. Wang, P., Su, X., Jourenko, M., Jiang, Z., Larangeira, M., Tanaka, K.: Environmental adaptive privacy preserving contact tracing system for respiratory infectious diseases. In: Meng, W., Conti, M. (eds.) Cyberspace Safety and Security - 13th International Symposium, CSS 2021, Virtual Event, November 9-11, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13172, pp. 131–144. Springer (2021), https://doi.org/10.1007/978-3-030-94029-4_10

## A The Necessity of Enhancing Contact Tracing Systems

This work considers the enhancement of contact tracing systems due to the following two epidemiology findings: (1) modes of transmission (droplet and airborne); (2) environmental factors (temperature, humidity, air velocity, etc.).

Droplet transmission refers to the infections caused by viruses ejected with droplets by sneezes or coughs; whereas, airborne transmission means that infections are caused by floating liquid drops carrying viruses suspended in the air [21]. Hence, "close contact" should be defined differently in the two transmission modes, *i.e.*, droplet transmissible viruses require face-to-face contact; whereas, airborne transmissible viruses only require people to come into the region of the floating virus during their lifespan. Conventional contact tracing systems utilize Bluetooth Low Energy (BLE) technology to trace face-to-face contact. In contrast, to the best of our knowledge, only Wang et al. [38] considered the airborne transmission with their discrete-location-tracing setting (DLT). Intuitively, the DLT setting records users' relative and absolute positions to decide close contact[4].

As mentioned above, virus distribution, *e.g.*, lifespan and region size, affects the infectiousness of viruses. Epidemiology research [16, 20, 31] concludes that virus distribution depends on environmental factors, including temperature, humidity, air velocity, etc. However, in conventional contact tracing systems, BLE usually scans according to predetermined parameters, *e.g.*, interval and radius. The mismatch between virus distribution and scan parameters may cause overwhelmingly false-positive records, burdening the medical system in real-life. Therefore, we consider that it is necessary to filter records according to environmental factors for practical contact tracing systems.

A problem caused by including more data (*i.e.*, position and environmental factors) is that revealing these data may result in the identification of users. Therefore, this work embeds them into the attributes of an attribute-based credentials (ABC) scheme. Anonymity and selective showing capability of ABC schemes (from [26, 18] and our Construction 2) empower users to reveal only necessary attributes to verifiers while keeping other attributes secret, *i.e.*, medical agencies and other users (including the issuer) cannot learn more than what is revealed during *the showing of credentials*[5]. Potentially, we can further tweak the set-commitment scheme [26] (mentioned in Section 2) to enable the proof of knowledge of the commitment content, hence, achieving blind issuance as shown in [10]. The blind issuance capability can prevent issuers from learning users' attributes during *the issuance of credentials.*

[4] However, due to the inherent decentralization of their system (*i.e.*, users are designed to issue their own contact records), they failed to achieve meaningful integrity guarantees for the DLT setting.

[5] Users are required to report necessary data to decide if they are closed enough to be considered involved in a contact. The reveal of such data may also have privacy impacts. However, we found it hard to quantify such impacts and left this problem for further consideration.

## B Auxiliary Definitions

This section recalls the formal definitions for our black-box building blocks.

### B.1 Digital Signature Schemes

The tuple of algorithm in a digital signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ works as follows.

- $\mathsf{KGen}(1^\lambda)$ takes as input the security parameter $\lambda$ and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$;
- $\mathsf{Sign}(\mathsf{sk}, \mathrm{m})$ takes as input the secret key $\mathsf{sk}$ and a message m. It outputs a signature $\sigma$ on m under $\mathsf{sk}$;
- $\mathsf{Verify}(\mathsf{pk}, \mathrm{m}, \sigma)$ takes as input the public key $\mathsf{pk}$, the message m and the signature $\sigma$. It outputs 1 if the signature is valid and 0 otherwise.

This work employs a signature scheme that satisfies correctness and the existential unforgeability under adaptive chosen message attacks (EUF-CMA) [29].

**Definition 17 (Correctness).** *A signature scheme satisfies correctness, if for any $\lambda > 0$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$:*

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1\right] = 1.$$

**Definition 18 (EUF-CMA).** *A signature scheme satisfies EUF-CMA, if for any adversary that has access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}(\mathsf{sk}, \cdot)$ with queries $m \in \mathsf{Q}$, the following probability is negligible of $\lambda$ for any $\lambda > 0$ and $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$:*

$$\Pr\left[(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk}) : m^* \notin \mathsf{Q} \wedge \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}) = 1\right].$$

### B.2 Set-Commitment Scheme

We consider a set-commitment scheme $\mathsf{SC} \overset{\Delta}{=} (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenSubset}, \mathsf{VerifySubset})$ that satisfies correctness, binding, subset-soundness and hiding [26]. Its algorithms work as follows [26].

- $\mathsf{Setup}(1^\lambda, q)$ takes as input the security parameter $\lambda$ and the size upper bound $q$ of committed sets. It outputs the public parameter $\mathsf{pp}$;
- $\mathsf{Commit}(\mathsf{pp}, A)$ takes as input $\mathsf{pp}$ and a non-empty set $A$. It outputs a commitment $C$ and opening information $O$;
- $\mathsf{Open}(\mathsf{pp}, C, A, O)$ takes as input $\mathsf{pp}$, a commitment $C$, a set $A$, and opening information $O$. It outputs 1 if $O$ is a valid opening of $C$ to $A$; and 0 otherwise;
- $\mathsf{OpenSubset}(\mathsf{pp}, C, A, O, D)$ takes as input $\mathsf{pp}$, a commitment $C$, a set $A$, opening information $O$, and a non-empty set $D$. If $D \subseteq A$, it outputs a witness $W$ that shows $D$ being a subset of the set $A$ committed in $C$;
- $\mathsf{VerifySubset}(\mathsf{pp}, C, D, W)$ takes as input $\mathsf{pp}$, a commitment $C$, a non-empty set $D$, and a witness $W$. It outputs 1 if $W$ is a witness that shows $D$ being a subset of the set $A$ committed in $C$; and 0 otherwise.

**Definition 19 (Correctness).** *A set-commitment scheme satisfies correctness, if for any $\lambda, q \geq 0$, any $A$ and non-empty $D \subseteq A$, and any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$:*

$$\Pr\left[(C,O) \leftarrow \mathsf{Commit}(\mathsf{pp},A) : \mathsf{Open}(\mathsf{pp},C,A,O) = 1\right] = 1 \wedge$$

$$\Pr\begin{bmatrix} (C,O) \leftarrow \mathsf{Commit}(\mathsf{pp},A) & : \mathsf{VerifySubset}(\mathsf{pp},C,D,W) = 1 \\ W \leftarrow \mathsf{OpenSubset}(\mathsf{pp},C,A,O,D) & \end{bmatrix} = 1.$$

**Definition 20 (Binding).** *A set-commitment scheme is binding, if for any $\lambda, q \geq 0$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and all PPT adversary $\mathcal{A}$, the following probability is negligible of $\lambda$:*

$$\Pr\begin{bmatrix} & \mathsf{Open}(\mathsf{pp},C,A,O) = 1 \wedge \\ (C,A,O,A',O') \leftarrow \mathcal{A}(\mathsf{pp}) : & \mathsf{Open}(\mathsf{pp},C,A',O') = 1 \wedge \\ & A \neq A' \end{bmatrix}.$$

**Definition 21 (Subset-Soundness).** *A set-commitment scheme satisfies subset-soundness, if for any $\lambda, q \geq 0$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and all PPT adversary $\mathcal{A}$, the following probability is negligible of $\lambda$:*

$$\Pr\begin{bmatrix} & \mathsf{Open}(\mathsf{pp},C,A,O) = 1 \wedge \\ (C,A,O,D,W) \leftarrow \mathcal{A}(\mathsf{pp}) : & \mathsf{VerifySubset}(\mathsf{pp},C,D,W) = 1 \wedge \\ & D \nsubseteq A \end{bmatrix}.$$

**Definition 22 (Hiding).** *A set-commitment scheme is hiding, if for any $\lambda, q \geq 0$, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and all PPT adversary $\mathcal{A}$ with accessibility to an $\mathcal{O}_{\mathsf{OpenSubset}}$ oracle that opens the challenge commitment to any subset in the* intersection *of the two committed sets, the following probability holds:*

$$\left|\Pr\begin{bmatrix} b \xleftarrow{\$} \{0,1\}; (A_0, A_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}) & \\ (C,O) \leftarrow \mathsf{Commit}(\mathsf{pp},A_b); & : b^* = b \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{OpenSubset}}(\mathsf{pp},C,A_b,O,\cdot\subseteq A_0 \cap A_1)}(\mathsf{st},C) & \end{bmatrix} - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

### B.3 Zero-Knowledge Proofs of Knowledge (ZKPoK) Protocol

We follow the generic definition of ZKPoK given in [26]. Let $\mathcal{L}_\mathcal{R} = \{x : \exists w$, s.t., $(x,w) \in \mathcal{R}\} \subseteq \{0,1\}^*$ be a formal language with respect to a binary, polynomial-time (witness) relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, *i.e.*, when given a polynomial-size (of $|x|$) witness $w$ that certifies $(x,w) \in \mathcal{R}$, $x \in \mathcal{L}_\mathcal{R}$ can be decided in the polynomial time of $|x|$. Denote the interactive protocol between a (potentially unbounded) prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ with $\langle \cdot, b\rangle \leftarrow \langle \mathcal{P}(\cdot,\cdot), \mathcal{V}(\cdot)\rangle$ where $b \in \{0,1\}$. Here, $b = 1$ indicates that $\mathcal{V}$ accepts the conversation with $\mathcal{P}$; and $b = 0$ indicates $\mathcal{V}$ rejects. If an interactive protocol satisfies completeness, perfect zero-knowledge, and knowledge-soundness [24], we call it a ZKPoK.

**Definition 23 (Completeness).** *An interactive protocol $\langle \mathcal{P}, \mathcal{V}\rangle$ for a relation $\mathcal{R}$ satisfies completeness, if for any $x \in \mathcal{L}_\mathcal{R}$ and $w$ s.t., $(x,w) \in \mathcal{R}$:*

$$\Pr[\langle \cdot, 1\rangle \leftarrow \langle \mathcal{P}(x,w), \mathcal{V}(x)\rangle] = 1.$$

**Definition 24 ((Perfect) Zero-Knowledge).** *An interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ for a relation $\mathcal{R}$ is (perfect) zero-knowledge, if for any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that $\{\mathcal{S}^{\mathcal{A}}(x)\}_{x \in \mathcal{L}_{\mathcal{R}}} \approx \{\langle \mathcal{P}(x, w), \mathcal{A}(x) \rangle\}_{(x,w) \in \mathcal{R}}$ where $\langle \mathcal{P}(\cdot, \cdot), \mathcal{A}(\cdot) \rangle$ denotes the transcript of the interaction between $\mathcal{P}$ and $\mathcal{A}$, and "$\approx$" denotes (perfect) indistinguishability.*

**Definition 25 (Knowledge-Soundness).** *An interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ is proofs of knowledge (PoK) relative to an NP relation $\mathcal{R}$, if for any potentially unbounded adversarial prover $\mathcal{A}$ accepted by $\mathcal{V}$ on $x$, i.e., $\langle \cdot, 1 \rangle \leftarrow \langle \mathcal{A}(x), \mathcal{V}(x) \rangle$, with probability greater than $\epsilon$, then there exists a PPT knowledge extractor $\mathcal{K}^{\mathcal{A}}(x)$ (denoting that the extractor has rewinding black-box access to $\mathcal{A}$) that can output a value $w$ satisfying $(x, w) \in \mathcal{R}$ with probability polynomial of $\epsilon$.*

Since we assume the symmetric external Diffie-Hellman (SXDH) assumption [3] holds over bilinear groups for the DDH-based APK construction (Construction 5), we also show its definition in the following.

**Definition 26 (SXDH assumption).** *The SXDH assumption holds for* $\mathsf{BGGen}$*, if the DDH assumption (Definition 4) holds in* $\mathbb{G}_1$ and $\mathbb{G}_2$.

## C The SPS-EQ Scheme from [18]

We show the SPS-EQ scheme given by [18] with respect to a fully adaptive NIZK argument $\mathsf{NIZK} \overset{\Delta}{=} (\mathsf{PGen}, \mathsf{PPro}, \mathsf{PSim}, \mathsf{PRVer}, \mathsf{PVer}, \mathsf{ZKEval})$.

**Construction 4 (SPS-EQ Scheme $\mathsf{SPSEQ}$)** *The algorithms are as follows.*

- $\mathsf{Setup}(1^\lambda)$. *Run* $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$ *and sample matrices* $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \overset{\$}{\leftarrow} \mathcal{D}_1$ *from matrix distribution. Generate a common reference string and trapdoor for the malleable NIZK argument with* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.PGen}(1^\lambda, \mathsf{BG})$. *Return* $\mathsf{pp} = (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs}, \ell)$*;*
- $\mathsf{KGen}(\mathsf{pp})$. *Sample* $\mathbf{K}_0 \overset{\$}{\leftarrow} \mathbb{Z}_p^{2 \times 2}, \mathbf{K} \overset{\$}{\leftarrow} \mathbb{Z}_p^{\ell \times 2}$. *Compute* $[\mathbf{B}]_2 = [\mathbf{K}_0]_2[\mathbf{A}]_2$ *and* $[\mathbf{C}]_2 = [\mathbf{K}]_2[\mathbf{A}]_2$. *Set* $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K})$ *and* $\mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2)$. *Return* $(\mathsf{sk}, \mathsf{pk})$*;*
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, [\mathbf{m}]_1)$. *Sample* $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$. *Compute* $[\mathbf{t}]_1 = [\mathbf{A}_0]_1 r_1$ *and* $[\mathbf{w}]_1 = [\mathbf{A}_0]_1 r_2$. *Compute* $\mathbf{u}_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top [\mathbf{m}]_1$ *and* $\mathbf{u}_2 = \mathbf{K}_0^\top [\mathbf{w}]_1$. *Generate proof with* $(\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1) \leftarrow \mathsf{NIZK.PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$. *Set* $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ *and* $\tau = ([\mathbf{u}_2]_1, [\mathbf{u_2}]_1, [\mathbf{w}]_1, \Omega_2)$. *Return* $(\sigma, \tau)$*;*
- $\mathsf{ChgRep}(\mathsf{pp}, [\mathbf{m}]_1, (\sigma, \tau), \mu, \rho, \mathsf{pk})$. *Parse* $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ *and* $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2), \perp\}$. *Let* $\Omega = (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$. *Check proof with* $\mathsf{NIZK.PVer}(\mathsf{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$. *Check if* $e([\mathbf{u}_2]_1^\top, \mathbf{A}]_2) = e([\mathbf{w}]_1^\top, \mathbf{B}]_2)$ *and* $e([\mathbf{u}_1]_1^\top, \mathbf{A}]_2) = e([\mathbf{t}]_1^\top, \mathbf{B}]_2) + e([\mathbf{m}]_1^\top, \mathbf{C}]_2)$. *Sample* $\alpha, \beta \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. *Compute* $[\mathbf{u'}_1]_1 = \rho(\mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1)$ *and* $[\mathbf{t}']_1 = \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$. *And for* $i \in \{0, 1\}$, *compute* $[z_i']_2 = \alpha[z_i]_2, [\mathbf{a'}_i]_1 = \alpha\mu[\mathbf{a}_i^1]_1 + \alpha\beta[\mathbf{a}_i^2]_1, [d_i']_2 = \alpha\mu[d_i^1]_2 + \alpha\beta[d_i^2]_2$. *Set* $\Omega' = (([\mathbf{a'}_i]_1, [d_i']_2, [z_i']_2)_{i \in \{0,1\}}, \alpha Z_1)$. *Set* $\sigma' = ([\mathbf{u'}_1]_1, [\mathbf{t}']_1, \Omega')$. *Return* $(\mu[\mathbf{m}]_1, \sigma')$*;*

- $\mathsf{Verify}(\mathsf{pp}, (\rho, \mathsf{pk}), [\mathbf{m}]_1, (\sigma, \tau))$. *Parse* $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ *and* $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2), \perp\}$. *Check proof* $\Omega_1$ *with* $\mathsf{NIZK.PRVer}(\mathsf{crs}, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ *and check if* $e([\mathbf{u}_1]_1^\top, \mathbf{A}]_2) = e([\mathbf{t}]_1^\top, \mathbf{B}]_2) + e([\mathbf{m}]_1^\top, \mathbf{C}]_2)$. *If* $\tau \neq \perp$, *then check proof* $\Omega_2$ *with* $\mathsf{NIZK.PRVer}(\mathsf{crs}, [\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$ *and check if* $e([\mathbf{u}_2]_1^\top, \mathbf{A}]_2) = e([\mathbf{w}]_1^\top, \mathbf{B}]_2)$.

The given SPS-EQ construction (Construction 4) satisfies correctness, the EUF-CMA, and the property of Perfect Adaption of Signatures with respect to Message Space. The formal definitions are as follows.

**Definition 27 (Correctness).** *An SPS-EQ scheme satisfies correctness, if for any* $\lambda > 0, \ell > 1$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*:*

$$\begin{aligned}&\Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, [\mathbf{m}]_1)) = 1 \wedge\right.\\ &\Pr\left[\mathsf{Verify}(\rho \cdot \mathsf{pk}, \mathsf{ChgRep}([\mathbf{m}]_1, \mathsf{Sign}(\mathsf{sk}, [\mathbf{m}]_1), \mu, \rho, \mathsf{pk}))\right] = 1.\end{aligned}$$

**Definition 28 (EUF-CMA).** *An SPS-EQ scheme satisfies EUF-CMA, if for any adversary that has access to a signing oracle* $\mathcal{O}_{\mathsf{Sign}}(\mathsf{sk}, \cdot)$ *with queries* $[\mathbf{m}]_i \in \mathsf{Q}$*, the following probability is negligible of* $\lambda$ *for any* $\lambda > 0, \ell > 1$ *and* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*:*

$$\Pr\left[\begin{array}{ll}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp}); : & \forall [\mathbf{m}]_i \in \mathsf{Q}, [\mathbf{m}^*]_{\mathcal{R}} \neq [\mathbf{m}]_{\mathcal{R}} \wedge \\ ([\mathbf{m}]_i^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk}) & \mathsf{Verify}([\mathbf{m}]_i^*, \sigma^*, \mathsf{pk}) = 1\end{array}\right].$$

**Definition 29 (Perfect Adaption of Signatures with respect to Message Space (under Malicious Keys in the Honest Parameters Model)).** *An SPS-EQ scheme over a message space* $\mathcal{S}_{\mathbf{m}} \subseteq (\mathbb{G}_i^*)^\ell$ *perfectly adapts signatures with respect to the message space, if for all tuples* $(\mathsf{pp}, [\mathsf{pk}]_j, [\mathbf{m}]_i, (\sigma, \tau), \mu, \rho)$ *such that* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), [\mathbf{m}]_i \in \mathcal{S}_{\mathbf{m}}$*,* $\mu, \rho \in \mathbb{Z}_p^*$*, and* $\mathsf{Verify}(\mathsf{pk}, [\mathbf{m}]_i, (\sigma, \tau)) = 1$*, we have the output* $([\mu \cdot \mathbf{m}]_i, \sigma^*) \leftarrow \mathsf{ChgRep}([\mathbf{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{pk}]_j)$ *where* $\sigma^*$ *is a* random *element in the signature space such that* $\mathsf{Verify}([\rho \cdot \mathsf{pk}, \mu \cdot \mathbf{m}]_i, \sigma^*) = 1$.

## D Extending the BLS Signature [9] with APK

This section demonstrates the capability of APK being used as a plug-in tool for cryptographic primitives. Concretely, we give a DDH-based APK construction and integrate it into the BLS signature scheme [9]. We modify the update algorithm in APK to enable signers in the BLS scheme to re-randomize their public keys and signatures. Whereas, the audit algorithm helps to link the updated public keys to their original signers.

First, recall the BLS construction given in [9]. Let $\mathsf{BGGen}(1^\lambda)$ be the bilinear group generator that outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ as shown in Section 2. Let $\mathsf{H} : \mathcal{S}_{\mathbf{m}} \rightarrow \mathbb{G}_1$ be a cryptographic hash function where $\mathcal{S}_{\mathbf{m}}$ denotes the message space. For simplicity, we omit the algorithms for aggregation to focus on auditability. Denote the algorithms with $\mathsf{BLS} \stackrel{\Delta}{=} (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$.

- $\mathsf{KGen}(\mathsf{BG})$: Sample $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathsf{pk} = \mathsf{sk} \cdot P_2 \in \mathbb{G}_2$. Output $(\mathsf{sk}, \mathsf{pk})$;

- $\mathsf{Sign}(\mathsf{sk}, m)$: Output $\sigma = \mathsf{sk} \cdot \mathsf{H}(m) \in \mathbb{G}_1$;
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Output 1 if $e(\sigma, P_2) = e(\mathsf{H}(m), \mathsf{pk})$, or 0 otherwise;

Next, we show the DDH problem (in group $\mathbb{G}_2$)-based APK construction.

**Construction 5 (DDH-Based APK $\mathsf{APK}_{\mathrm{DDH}}$)** *Let* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ *be the output of the bilinear group generator* $\mathsf{BGGen}(1^\lambda)$*. The algorithms of APK are as follows.*

- $\mathsf{KGen}(\mathsf{BG})$*: Sample* $\mathsf{ak}, \mathsf{sk}_0 \xleftarrow{\$} \mathbb{Z}_p$*. Set* $\mathsf{sk}_1 = \mathsf{ak} \cdot \mathsf{sk}_0$*. Then, compute* $\mathsf{pk}_0 = \mathsf{sk}_0 \cdot P_2$ *and* $\mathsf{pk}_1 = \mathsf{sk}_1 \cdot P_2$*. Finally, set* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1), \mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ *and output* $(\mathsf{sk}, \mathsf{ak}, \mathsf{pk})$*;*
- $\mathsf{Update}(\mathsf{pk}; r)$*: Parse* $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$*. Sample* $r \xleftarrow{\$} \mathbb{Z}_p$ *and compute* $\mathsf{pk}'_0 = r \cdot \mathsf{pk}_0, \mathsf{pk}'_1 = r \cdot \mathsf{pk}_1$*. Output* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1)$*;*
- $\mathsf{VerifyKP}(\mathsf{sk}, \mathsf{pk}', r)$*: Parse* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1)$*. Output* 1 *if* $\mathsf{pk}'_0 = r \cdot \mathsf{sk}_0 \cdot P_2 \wedge \mathsf{pk}'_1 = r \cdot \mathsf{sk}_1 \cdot P_2$*, or* 0 *otherwise;*
- $\mathsf{VerifyAK}(\mathsf{sk}, \mathsf{ak})$*: Parse* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$*. Output* 1 *if* $\mathsf{sk}_1 = \mathsf{ak} \cdot \mathsf{sk}_0$*, or* 0 *otherwise;*
- $\mathsf{Audit}(\mathsf{ak}, \mathsf{pk}', \mathsf{pk})$*: Parse* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1), \mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$*. Output* 1 *if* $\mathsf{pk}_1 = \mathsf{ak} \cdot \mathsf{pk}_0 \wedge \mathsf{pk}'_1 = \mathsf{ak} \cdot \mathsf{pk}'_0$*, or* 0 *otherwise.*

Therefore, the integration works as follows.

**Construction 6 (BLS with APK)** *Let* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ *be the output of the bilinear group generator* $\mathsf{BGGen}(1^\lambda)$*. Let* $\mathsf{H} : \mathcal{S}_{\mathbf{m}} \rightarrow \mathbb{G}_1$ *be a cryptographic hash function where* $\mathcal{S}_{\mathbf{m}}$ *denotes the message space. Let* $\mathsf{APK}_{\mathrm{DDH}} = (\mathsf{KGen}, \mathsf{Update}, \mathsf{VerifyKP}, \mathsf{VerifyAK}, \mathsf{Audit})$ *be a DDH-based APK mechanism. The algorithms of BLS with APK are as follows.*

- $\mathsf{KGen}, \mathsf{VerifyKP}, \mathsf{VerifyAK}, \mathsf{Audit}$ *are the same as in* $\mathsf{APK}_{\mathrm{DDH}}$*;*
- $\mathsf{Sign}(\mathsf{sk}, m)$*: Parse* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$ *and output* $\sigma = \mathsf{sk}_1 \cdot \mathsf{H}(m) \in \mathbb{G}_1$*;*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$*: Parse* $\mathsf{pk} = (\mathsf{pk}_0, \mathsf{pk}_1)$ *and output* 1 *if* $e(\sigma, P_2) = e(\mathsf{H}(m), \mathsf{pk}_1)$*, or* 0 *otherwise;*
- $\mathsf{Update}(\mathsf{pk}, \sigma; r)$*: Run* $\mathsf{APK}_{\mathrm{DDH}}.\mathsf{Update}(\mathsf{pk}; r) \rightarrow \mathsf{pk}'$ *and compute* $\sigma' = r \cdot \sigma$*. Output* $(\mathsf{pk}', \sigma')$*.*

Since the EUF-CMA security of the (type-3) BLS signature is proven under the co-CDH assumption [8], and the APK given in Construction 5 considers the DDH problem in $\mathbb{G}_2$, it is convenient to assume the SXDH assumption to hold for $\mathsf{BGGen}$ to prove the EUF-CMA security of our extended BLS construction (Construction 6).