Auditable Attribute-Based Credentials Scheme and Its Applications in Contact Tracing

Pengfei Wang[1], Xiangyu Su[2], Mario Larangeira[2,3], and Keisuke Tanaka[2]

[1] Rakuten
[2] Department of Mathematical and Computing Sciences, School of Computing, Tokyo Institute of Technology. Tokyo-to Meguro-ku Oookayama 2-12-1 W8-55.
wang.p.ae@m.titech.ac.jp, su.x.ab@m.titech.ac.jp, mario@c.titech.ac.jp, keisuke@is.titech.ac.jp.
[3] 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*. We show a concrete construction by adding a DDH-based "auditable public key" mechanism to the Connolly et al.'s ABC scheme (PKC '22). In this work we present three contributions regarding the auditable ABC: (1) we refine the environmental-adaptive contact tracing framework, (2) present a formal treatment which includes game-based security definition and a detailed protocol construction. Finally, (3) we implement our construction to showcase the practicality of our protocol.

Keywords: Contact Tracing, Attribute-Based Credentials, Auditable Public Keys

1 Introduction

1.1 Background and Motivation

Contact tracing, a method that prevents diseases from spreading, faces new challenges considering new founds in epidemiology research. Proposed in [18], the environmental-adaptive contact tracing (EACT) framework took virus distribution (*e.g.*, lifespan and region size, which depends on environmental factors) and different transmission modes (*i.e.*, droplet and airborne) into consideration. However, the framework failed to unify the tracing processes in droplet

and airborne modes, hence, burdening the implementation and weakening the practicality. Moreover, the security definitions are based on an informal threat model, leaving a gap between the theoretical proofs and implementations.

The similarity between their framework and a self-issuing decentralized credentials scheme [11] inspires us to turn our eyes to credentials schemes, typically the attribute-based ones (ABC), which to the best of our knowledge, have never been connected with contact tracing in previous works despite of being a natural approach. 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 a general ABC scheme: (1) users can record environmental factors and local information as attributes; (2) users can prove their records anonymously; (3) the security of ABC, *i.e.*, anonymity and unforgeability, can easily be adapted to contact tracing (as we will show in Section 4.2). Moreover, it is also convenient to bring the broad spectrum of functionalities in ABC to contact tracing, *e.g.*, selective showing [9], proof of disjoint attributes [4], issuer-hiding [2, 4], delegation [1], *traceability* [14], etc.

Note that the traceable ABC [14] shares a similar traceability with group signatures schemes, *i.e.*, to revoke the anonymity of regular users. However, it differs from the traceability of contact tracing systems, which requires issuers (or third parties delegated on behalf of the issuer) of a credential can "audit" shown credentials, *i.e.*, to verify if the credential is issued by the issuer. The following of this paper will explain our approach that adds the new functionality of auditability to ABC schemes. We emphasize that an auditable ABC scheme may have its interests, and the applications will not be limited to contact tracing.

Related work. In EACT [18], the authors showed a rather extensive list of existing contact tracing systems. Hence, we will not repeat their observation but argue that none of these works, including [18], are in the same scope as ours, in which bringing credentials schemes into contact tracing. Moreover, despite the broad functionalities of ABC schemes, no existing work considers the same traceability (revoking issuer's anonymity) as in contact tracing systems.

1.2 Our approach and Contributions

Now, we show a brief image of our approach.

In order to build an auditable ABC scheme, we first propose a cryptographic tool called "auditable public keys (APK)", which extends the updatable public key mechanism given in [6]. The APK embeds extra structure within secret and public keys as a new auditing key so that the structure preserves even after updating the public key. Hence, 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. Like the updatable public key, our APK can be

used as a plug-in in many different cryptographic primitives, hence, not being limited to credentials schemes.

Next, we adapt APK to existing ABC schemes [9] and define the formal syntax of our auditable ABC. We show a concrete construction for the APK mechanism based on the matrix diffie-hellman assumptions over matrix distributions [5, 15]. We prove that our APK construction can be inserted into the structure-preserving signatures on equivalence classes (SPS-EQ) scheme [4] 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 [9], and general zero-knowledge proof-of-knowledge protocols (with perfect zero-knowledge) [7], we present a construction for the auditable ABC scheme.

Finally, we refine the EACT framework [18] and provide a construction based on our auditable ABC scheme. Hence, we can unify the tracing process of the conventional 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 auditable EACT 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 for our refined EACT 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 our APK, and show concrete constructions for APK and the auditable ABC scheme; (3) we refine and construct the EACT framework [18] based on credentials schemes. We also provide cryptographic game-based security definitions and implement the construction. Additionally, we add algorithms to jPBC library [3] to support matrix-based bilinear pairing operations during implementation.

1.3 Organization

We organize the main contents of the paper 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

Throughout this paper, we use λ 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, $x \xleftarrow{\$} A$ denotes that x is

randomly and uniformly sampled from A; whereas, for an algorithm Alg, $x \leftarrow \mathsf{Alg}$ denotes that x is assigned the output of an algorithm 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 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 $\mathbb{Z}_p[X]$, $\mathsf{Ch}_A(X) \triangleq \Pi_{x \in A}(X - x) = \sum_{i=0}^{|A|} c_i \cdot X^i$ as A's characteristic polynomial.

We denote the asymmetric bilinear group generator as $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$ where $\mathsf{BG} \triangleq (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. 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 [4], 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. This work takes the black-box use of three cryptographic primitives: (1) a digital signature scheme $\mathsf{SIG} \triangleq (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ that satisfies correctness and existentially unforgeability under adaptive chosen-message attacks (EUF-CMA) [12]; (2) a set-commitment scheme $\mathsf{SC} \triangleq (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenSubset}, \mathsf{VerifySubset})$ that satisfies correctness, binding, subset soundness and hiding [9]; (3) a zero-knowledge proofs of knowledge (ZKPoK) protocol Π that satisfies completeness, perfect zero-knowledge and knowledge-soundness [7]. Due to the page limitation, the formal descriptions of these primitives are omitted in this paper. They can be found in the corresponding references.

Assumptions. We assume the following assumptions hold over matrix distribution: the matrix decision diffie-hellman (MDDH) assumption [5] and the kernel matrix diffie-hellman (KerMDH) assumption [15]. We also assume the q-co-discrete-logarithm (q-co-DL) assumption holds over bilinear groups. The definitions are as follows.

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 $\mathcal{D}_k \triangleq \mathcal{D}_{k+1,k}$.*

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

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

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

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

3 Auditable Attribute-Based Credentials Scheme

This section presents our first contribution: an auditable attribute-based credentials 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 [9]. On a high level, correctness guarantees that verifies 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 identifier and verifiable credentials [16, 17] refueled the interest of the community in researching ABC schemes. New functionalities, as shown in Section 1.1, 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 deanonymize particular *issuers*. In order to show our scheme, we first introduce the notion of an auditable public key (APK) mechanism that extends the updatable public key from [6]. Then, we employ APK and present our auditable ABC scheme in terms of definitions and constructions.

3.1 Auditable Public Keys

Proposed in [6], 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, the auditors, who hold the auditing key 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 [6].

Definition 4 (Auditable Public Key Mechanism). *An auditable public key (APK) mechanism involves a tuple of algorithms* $\mathsf{APK} \triangleq (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Update}, \mathsf{VerifyUpdate}, \mathsf{VerifyAK}, \mathsf{Audit})$ *that are performed as follows.*

- $\mathsf{Setup}(1^\lambda)$ *takes as input the security parameter* λ *and outputs the public parameter* pp *that includes secret, auditing and public key space* $\mathcal{SK}, \mathcal{AK}, \mathcal{PK}$*. These are given implicitly as input to all other algorithms;*
- $\mathsf{KGen}(\mathsf{pp})$ *takes as input the public parameter* 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* pp *in algorithm inputs;*
- $\mathsf{Update}(\mathsf{pk}; r)$ *takes as input a public key* pk *and a randomness* r*. It outputs a new public key* $\mathsf{pk}' \in \mathcal{PK}$*;*
- $\mathsf{VerifyUpdate}(\mathsf{sk}, r, \mathsf{pk}')$ *is deterministic.* $\mathsf{VerifyUpdate}$ *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.* $\mathsf{VerifyAK}$ *takes as input a secret key* $\mathsf{sk} \in \mathcal{SK}$ *and an auditing key* $\mathsf{ak} \in \mathcal{AK}$*. It outputs* 1 *if* ak *corresponds to* sk*, 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* $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{SK} \times \mathcal{PK}$*.* Audit *takes as input a public key* $\mathsf{pk}' \in \mathcal{PK}$*, the auditing key* ak *and the public key* pk*. It outputs* 1 *if* pk' *is updated from* pk*, or* 0 *otherwise.*

The APK mechanism should satisfy correctness, indistinguishability, and unforgeabilities.

Definition 5 (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.,* $\mathsf{VerifyUpdate}(\mathsf{Update}(\mathsf{pk}; r), \mathsf{sk}, r) = 1$*; (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 [6], *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 KGen and Update since these algorithms are publicly available.

Definition 6 (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 $(\mathsf{sk}^*, \mathsf{pk}^*, \mathsf{ak}^*) \leftarrow \mathsf{KGen}(\mathsf{pp})$.*

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

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 is updated from the secret key's corresponding public key with respect to the randomness; (2) the secret verifies the given auditing key; (3) the auditing key verifies the output and given public keys. 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 [6], 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. Note that in the unforgeability game, the challenge given to the adversary must not be queried before.

Definition 7 (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 $(\mathsf{sk}, \mathsf{pk}, \mathsf{ak}) \leftarrow \mathsf{KGen}(\mathsf{pp})$.*

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

Definition 8 (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 [6], our APK can be constructed from the DDH assumption and variants. We will show one concrete example based on the MDDH (and KerMDH, which can be implied by the MDDH [15]) assumption in our auditable ABC construction given in the following (Section 3.3).

3.2 Formal Definitions of Auditable ABC

The start point of our auditable ABC is the scheme given in [9], 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.

Definition 9 (Auditable ABC Scheme). *An auditable ABC scheme* 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* Audit*. The participants in* AABC *perform as follows.*

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

In addition to the auditing process, we make two modifications to the ABC scheme from [9]. 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 [9, 4] only mentioned them in security definitions.

Second, our 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 opk of the Show algorithm. Their purpose is to prevent credentials from

being issued by unidentified issuers. However, as shown in [4], the exposure of issuer identity affects the anonymity of users. Although some previous works [2, 4] proposed the property of issuer-hiding so that users can hide their credential issuers' identities within a list of identified issuers, achieving such a property incurs heavy mechanisms. In our case, we rely on the Audit algorithm to provide an extra layer of verification. That is, given an updated issuer-public key in a credential showing, the auditor who holds an auditing key that *corresponds to an identified public key* must prove whether the shown credential is issued the corresponding secret key.

Security properties. We formally define correctness, anonymity, and *unforgeabilities* 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 pp if the algorithm takes as input other variables.

Definition 10 (Correctness). *An* 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 [9], 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 opk. The sets HU, 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 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 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 HU to 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 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, 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 Issue. If $\mathsf{cred} = \perp$, the oracle returns $\perp$; Otherwise, it adds (i, 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 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 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 opk and 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 $D \not\subseteq (\mathsf{ATTR}[j_0] \cap \mathsf{ATTR}[j_1])$, the oracle returns $\perp$; Otherwise, it runs $(S_b, \cdot) \leftarrow \langle \mathsf{Show}(\mathsf{opk}, \mathsf{ATTR}[j_b], D, \mathsf{CRED}[j_b]), \cdot \rangle$ by interacting with the adversary $\mathcal{A}$ running Verify.

Then, the formal definitions are as follows.

Definition 11 (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 $\lambda, q > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$.*

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

Definition 12 (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 $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{OrgKGen}(\mathsf{pp})$.*

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

Similar to APK, we require another unforgeability regarding to auditing keys. Concretely, a user should not be able to 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 8.

3.3 Our Constructions and Analysis

The construction of our auditable ABC follows the same approach of [4] that takes as building blocks a structure-preserving signatures on equivalence classes (SPS-EQ) scheme and a set-commit scheme. 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 [4], the setup algorithm Setup runs $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$ and samples a matrix $\mathbf{A} \xleftarrow{\$} \mathcal{D}_1$. It outputs $\mathsf{pp} \xleftarrow{\Delta} (\mathsf{BG}, [\mathbf{A}]_2, \ell)$ where $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$, and ℓ 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 (Auditable Public Key APK) *The rest of the algorithms are as follows.*

- $\mathsf{KGen}(\mathsf{pp})$*: Sample matrices* $\mathbf{K}_0 \xleftarrow{\$} \mathcal{D}_{\ell,2}$ *and* $\mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ *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* $r \xleftarrow{\$} \mathbb{Z}_p$ *and compute* $[\mathbf{B}']_2 = r \cdot [\mathbf{B}]_2, [\mathbf{C}']_2 = r \cdot [\mathbf{C}]_2$*. Output* $\mathsf{pk}' = ([\mathbf{B}']_2, [\mathbf{C}']_2)$*;*
- $\mathsf{VerifyUpdate}(\mathsf{pk}', \mathsf{sk}, r)$*: Parse* $\mathsf{pk}' = (\mathsf{pk}'_0, \mathsf{pk}'_1)$ *and* $\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1)$*. Output* 1 *if* $\mathsf{pk}'_0 = r \cdot \mathsf{sk}_0 \cdot [\mathbf{A}]_2 \wedge \mathsf{pk}'_1 = r \cdot \mathsf{pk}_1 \cdot [\mathbf{A}]_2$*, 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* APK *given by Construction 1 satisfies correctness (Definition 5), indistinguishability (Definition 6), and secret key and auditing key unforgeability (Definition 7 and 8) if the MDDH and KerMDH assumption (Definition 2 and 3) 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 $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{X}]_2, [\mathbf{X}y]_2, [\mathbf{z}]_2)$ where $\mathbf{X} \xleftarrow{\$} \mathcal{D}_{l,1}, y \xleftarrow{\$} \mathbb{Z}_p, \mathbf{z} \xleftarrow{\$} \mathbb{Z}_p^l$. Here, 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 $\ell\times 1$, respectively. Note that the reduction

needs to prepare both components of the public key. That is, it samples $\mathbf{X}' \xleftarrow{\$} \mathbb{Z}_p^{l' \times l}$ such that $l' \in \{2, \ell\} \wedge l' \neq l$, and embeds the MDDH challenge chl by setting $\mathsf{pk}^* \triangleq ([\mathbf{X}]_2, \mathbf{X}'[\mathbf{X}]_2)$ and $\mathsf{pk}' \triangleq ([\mathbf{z}]_2, \mathbf{X}'[\mathbf{z}]_2)$. The indistinguishability adversary $\mathcal{A}$ takes as input $(\mathsf{pk}^*, \mathsf{pk}')$. If the challenge tuple satisfies $[\mathbf{z}]_2 = y[\mathbf{X}]_2$, then pk' is distributed identically to pk_0 ($\mathsf{pk}_0 \leftarrow \mathsf{Update}(\mathsf{pk}^*)$). Otherwise, pk' is distributed identically to pk_1 (a freshly generated public key). 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.

The proofs of two types of unforgeability are similar. For secret key unforgeability, the reduction receives a KerMDH challenge over $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{A}]_2, [\mathbf{X}]_2 = [\mathbf{K}_1\mathbf{A}]_2)$ where $\mathbf{A} \in \mathcal{D}_1$, and $\mathbf{K}_1 \in \mathbb{Z}_p^{2\times 2}$ of full rank 2. The reduction then prepares the inputs for the unforgeability adversary $\mathcal{A}$. That is, it samples $\mathbf{K}_0 \xleftarrow{\$} \mathcal{D}_{\ell,2}$ and embeds the challenge chl by setting $\mathsf{ak} \triangleq \mathbf{K}_0$ and $\mathsf{pk} \triangleq ([\mathbf{X}]_2, [\mathbf{K}_0\mathbf{X}]_2)$. Hence, the input to the adversary in the reduction is distributed identically as in the definition of unforgeability (note that $[\mathbf{A}]_2 \in \mathsf{pp}$). Suppose the adversary $\mathcal{A}$ breaks secret key unforgeability, which means that $\mathsf{VerifyUpdate}, \mathsf{VerifyAK}, \mathsf{Audit}$ 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 $\mathsf{sk}'_1[\mathbf{A}]_2 = [\mathbf{K}_0\mathbf{X}]_2$, *i.e.*, $(\mathsf{sk}'_0 - \mathbf{K}_1)[\mathbf{A}]_2 = [\mathbf{0}]_2$ (hence, yielding $\mathsf{sk}'_1 = \mathbf{K}_0 \cdot \mathsf{sk}'_0$). The equation is equivalent to solving $\mathcal{D}_1$-KerMDH problems. Therefore, the advantage of the reduction 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 $\mathbb{G}_2$, $\mathsf{chl} = (P_2, [\mathbf{X}]_2, [\mathbf{Y}]_2 = [\mathbf{K}_0\mathbf{X}]_2)$ where $\mathbf{X} \in \mathcal{D}_1$, and $\mathbf{K}_0 \in \mathbb{Z}_p^{\ell\times 2}$ of full rank 2. It directly relays the challenge to the adversary. Hence, the input of the adversary, $\mathsf{pk} = ([\mathbf{X}]_2, [\mathbf{K}_0\mathbf{X}]_2)$, distributes identically to the definition. Suppose the adversary $\mathcal{A}$ breaks auditing key unforgeability, which means it finds 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.*, $\mathsf{ak}' \cdot [\mathbf{X}]_2$ should equal to $[\mathbf{K}_0\mathbf{X}]_2$. 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 [4] (the construction and security definitions can be found in Appendix A). 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 [4]. We show the original key generation of the SPS-EQ in the following. Recall that the Setup algorithm outputs $\mathsf{pp} = (\mathsf{BG}, [\mathbf{A}]_2, \ell)$.

– $\mathsf{SPSEQ.KGen}(\mathsf{pp})$: Sample matrices $\mathbf{K}_1 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ and $\mathbf{K} \xleftarrow{\$} \mathcal{D}_{\ell,2}$ 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 $\mathbf{K}_0 \xleftarrow{\$} \mathcal{D}_{\ell,2}$ of full rank 2 and compute $\mathbf{K}$ by the multiplication of $\mathbf{K}_0$ and $\mathbf{K}_1$. In the following lemma, we prove 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 SPS-EQ scheme* SPSEQ *given by Construction 4 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 games: Game_0 is our modified scheme with $\mathsf{APK.KGen}$, and Game_1 is the original SPS-EQ scheme with $\mathsf{SPSEQ.KGen}$. We further denote the adversary $\mathcal{A}$'s advantage with Adv_i for each game 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, $|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \mathsf{Adv}^{\mathrm{MDDH}}_{\mathcal{D}_{\ell,2},\mathbb{G}_2}$. Therefore, we conclude the lemma.

Constructing the auditable ABC. Let BGGen be the bilinear group generation, $\mathsf{SC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenSubset}, \mathsf{VerifySubset})$ be the set-commitment scheme [9] that satisfies correctness, binding, subset soundness and hiding, and Π be a general ZKPoK protocol that satisfies completeness, perfect zero-knowledge and knowledge-soundness. With the necessary algorithms from our APK mechanism and the SPS-EQ [4], *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 AABC in the following.

Construction 2 (Auditable ABC AABC**)** *The algorithms are as follows.*

– $\mathsf{Setup}(1^\lambda, aq)$*: 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* $\xleftarrow{\$} \mathbb{Z}_p^*$ *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* crs *for the non-interactive zero knowledge argument in* SPSEQ *(which we take as a black-box). 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* 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 [4], 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; 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 [9];*

– $\mathsf{Audit}(\mathsf{ak}, S, \mathsf{opk})$*: Parse* $S{=}(\mathsf{opk}', \mathsf{cred}', W; D)$*. Return* $\mathsf{APK.Audit}(\mathsf{ak}, \mathsf{opk}', \mathsf{opk})$*.*

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

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

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

Therefore, we have the following result.

Theorem 2. *The auditable ABC scheme* AABC *given by Construction 2 satisfies correctness (Definition 10), anonymity (Definition 11), unforgeability (Definition 12), and auditing unforgeability (Definition 8).*

Proof. We show a brief proof here. Correctness follows directly from the correctness of building blocks. If the ZKPoK protocol has perfect zero-knowledge, anonymity and unforgeability can be derived from adapting these properties of the original SPS-EQ [4] with our APK mechanism, which has been proven in

Lemma 1. It further requires the indistinguishability and secret key unforgeability of APK, which has been proven in Theorem 1. Particularly, unforgeability requires the set-commitment scheme to be subset-sound. Since we use set-commitment as a black-box building block, the reduction here follows the original ABC paper [9]. Finally, as we explained before, the auditing unforgeability is equivalent to the auditing key unforgeability of APK (we even 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 [18]. 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 settings in the EACT framework [18]. 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.*, $\mathsf{Compare}_{\{\mathrm{BLE,DLT}\}}(\mathsf{envpp}, D, A)$ takes as input the environmental parameters 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 (which will be shown in Construction 3). Note that 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, hence, providing additional evidence for the user's location data. In the following section, we present 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 general signature scheme that satisfies correctness and EUF-CMA, and let AABC be our auditable ABC construction given in Construction 2.

Construction 3 (Refined EACT 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* 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 be 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 be 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* pp *in algorithm inputs;*
- $\mathsf{KReg}(\mathsf{pk}, \mathsf{misc}; \mathbb{B})$ *is a DID [16] black-box, which takes as input a public key* $\mathsf{pk} \in \{\mathsf{opk}, \mathsf{mpk}\}$*, auxiliary information* misc*, and a bulletin board* $\mathbb{B}$*.* KReg *registers* pk *with the corresponding* 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* cred_0 *and* 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.,* $S \triangleq (\mathsf{opk}', \mathsf{cred}', W; D)$*. Moreover, we enable this protocol to process in batches, i.e., it can takes 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* $b = 1$*,* 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* $b_j = 1$*,* Trace *outputs* 1*; Otherwise, it outputs* 0*.*

4.2 Security and Analysis

In contrast to previous works, which only provide informal threat models, we directly employ the cryptographic game-based security definitions from our au-

ditable ABC scheme given in Section 3.2, including correctness, anonymity, and unforgeability. Note that the refined EACT requires signatures from medical agencies in Merge and the bulletin board $\mathbb{B}$ (which should satisfy the robust ledger properties [10], *i.e.*, the capability of achieving consensus atomically), we define separately "traceability" (which can be regarded as the correctness of the tracing process) to capture such a change.

Definition 13 (Traceability). *Given the bulletin board* $\mathbb{B}$*, a* 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* $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{pp})$ *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*.*

It is easy to prove traceability for our REACT construction based on the correctness of the underlying AABC and SIG. Hence, we have the following theorem.

Theorem 3. *Let the bulletin board satisfy the robust ledger properties [10]. The refined EACT* REACT *given by construction 3 satisfies traceability if* AABC *and* SIG *satisfy correctness, and the* Compare *algorithm is well-defined.*

Next, we consider the soundness of tracing, *i.e.*, the situation in which an honest user's 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 12) with additional assumptions for the bulletin board, signature scheme, and comparing algorithm (like in Theorem 3).

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 of freshness in $\mathsf{AABC}.\langle \mathsf{Show}, \mathsf{Verify} \rangle$ to be non-interactive. As shown in Theorem 2, the anonymity and unforgeability of AABC (also for REACT in Theorem 4) requires perfect zero-knowledge of Π. Hence, we must rely on heavy mechanisms, *e.g.*, [13], 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 [8] 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 Trace) can verify the showing.

Compared to the unforgeability of auditable ABC in Definition 12, due to the malicious $\mathcal{M}$ 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 $\mathcal{M}$).

Definition 14 (Tracing Soundness). *Given the bulletin board $\mathbb{B}$, a* REACT *system satisfies tracing soundness (with respect to malicious $\mathcal{M}$), if for any PPT adversary $\mathcal{A}$ that has access to oracles $\mathcal{O} = \{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}\}$, the following probability holds for any $\lambda > 0, q > 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, q)$, and $(\mathsf{osk}, \mathsf{opk}, \mathsf{ak}) \leftarrow \mathsf{OrgKGen}(\mathsf{pp})$.*

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

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. The proofs can be derived from previous content.

Theorem 4. *Our refined EACT* REACT *satisfies correctness, traceability, anonymity, and tracing soundness (with respect to malicious $\mathcal{M}$).*

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 [3] cannot fully support matrix-based bilinear pairing operations. The library-level implementation, together with extended parts for jPBC [3] library, can also be found in our anonymous repository[4].

Overview. In the recording protocol of our EAHT system, users only have limited time to exchange and record credentials. Hence, algorithms in the recording protocol, *i.e.*, UsrKGen, Exchange, have great impacts on the overall system performance. In contrast, in the tracing protocol, users can interact with medical agencies to get treatment. Hence, the time consumption of the tracing protocol is *unlikely* to be a performance bottleneck. Therefore, we focus on implementing the *recording* phase of the EAHT system. Concretely, we implement the following algorithms of REACT: (Setup, OrgKGen, UsrKGen, Exchange). Moreover, in the Exchange, we need to measure the performance of algorithms and transmission separately. For simplicity, we write data transmission in the form of $\mathsf{Transmit}(\cdot)$, and divide Exchange into (Obtain-1, $\mathsf{Transmit}_1$, Issue, $\mathsf{Transmit}_2$, Obtain-2). We run our implementation on a physical device and simulate data transmissions using Bluetooth 5.0. The results are shown in Table 1.

[4] https://anonymous.4open.science/r/EAHT_MODULE_TEST

Table 1. Experiment Results

Algorithms	Time	Algorithms	Time	Algorithms	Time
Setup	168.99	Obtain-1	40.08	Transmit(σ, τ)	75.16
OrgKGen	54.18	Transmit(π, 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.

Evaluation. Setup and OrgKGen only need to be executed once. Hence, they are performance-insensitive. 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 exposing. 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(π, C, R) and Transmit(σ, τ). 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 $\longrightarrow$ $\mathcal{U}_0$.Transmit(π, C, R) $\longrightarrow$ $\mathcal{U}_1$.Issue $\longrightarrow$ $\mathcal{U}_1$.Transmit(σ, τ) $\longrightarrow$ $\mathcal{U}_0$.Obtain-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 new requirements in contact tracing, we adopt a novel perspective from attribute-based credential schemes due to their similarity. By abstracting "traceability" from 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. Hence, we emphasize that the APK mechanism may be an independent point of interest.

Next, we extend the ABC schemes in [9, 4] 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 brings an additional layer of verification to the schemes that can hide identities for issuers, which is usually an overpowerful anonymity property in real-life. The auditability for identifying issuers may also be helpful in revoking credentials, which has been another long-worried problem when deploying credentials in reality.

Finally, our refined EACT framework fixes the problems in the original work [18], *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. 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, https://doi.org/10.1007/978-3-642-03356-8_7
2. 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, https://doi.org/10.1007/978-3-030-92548-2_9
3. 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, https://doi.org/10.1109/ISCC.2011.5983948
4. 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, https://doi.org/10.1007/978-3-030-97121-2_15
5. 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, https://doi.org/10.1007/978-3-642-40084-1_8
6. 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, https://doi.org/10.1007/978-3-030-34578-5_23
7. 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, https://doi.org/10.1007/0-387-34805-0_46
8. 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, https://doi.org/10.1007/3-540-47721-7_12
9. 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, https://doi.org/10.1007/s00145-018-9281-4
10. 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, https://doi.org/10.1007/978-3-662-46803-6_10
11. 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
12. 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, https://doi.org/10.1137/0217017
13. 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, https://doi.org/10.1007/11761679_21
14. 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, https://doi.org/10.1007/978-3-031-14791-3_18
15. 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, https://doi.org/10.1007/978-3-662-53887-6_27
16. Sporny, M., Sabadello, M., Guy, A., Reed, D.: Decentralized identifiers (DIDs) v1.0. W3C recommendation, W3C (Jul 2022), https://www.w3.org/TR/2022/REC-did-core-20220719/
17. Sporny, M., Zundel, B., Noble, G., Burnett, D., Longley, D., Hartog, K.D.: Verifiable credentials data model v1.1. W3C recommendation, W3C (Mar 2022), https://www.w3.org/TR/2022/REC-vc-data-model-20220303/

18. 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, https://doi.org/10.1007/978-3-030-94029-4_10

A The SPS-EQ Scheme from [4]

Here, we show the construction and security definitions of the SPS-EQ scheme given by [4]. First, the construction is built atop a fully adaptive non-interactive zero-knowledge (NIZK) argument $\mathsf{NIZK} \stackrel{\Delta}{=} (\mathsf{PGen}, \mathsf{PPro}, \mathsf{PSim}, \mathsf{PRVer}, \mathsf{PVer}, \mathsf{ZKEval})$. We omit their details due to the page limitation.

Construction 4 (SPS-EQ Scheme SPSEQ) *The algorithms are performed as follows.*

- $\mathsf{Setup}(1^\lambda)$. *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 matrices* $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \stackrel{\$}{\leftarrow} \mathcal{D}_1$ *from matrix distribution. Generate a common reference string and trapdoor for the malleable NIZK argument with* $\mathsf{NIZK.PGen}(1^\lambda, \mathsf{BG}) \rightarrow (\mathsf{crs}, \mathsf{td})$. *Return* $\mathsf{pp} = (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs}, \ell)$;
- $\mathsf{KGen}(\mathsf{pp})$. *Sample* $\mathbf{K}_0 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{2\times 2}, \mathbf{K} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^{\ell\times 2}$. *Compute* $[\mathbf{B}]_2 = [\mathbf{K}_0]_2[\mathbf{A}]_2$ *and* $[\mathbf{C}]_2 = [\mathbf{K}]_2[\mathbf{A}]_2$. *Set* $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K}_1, \mathbf{K})$ *and* $\mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2)$. *Return* $(\mathsf{sk}, \mathsf{pk})$;
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, [\mathbf{m}]_1)$. *Sample* $r_1, r_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. *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* $\mathsf{NIZK.PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2) \rightarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$. *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* (σ, τ);
- $\mathsf{ChgRep}(\mathsf{pp}, [\mathbf{m}]_1, (\sigma, \tau), \mu, \rho, \mathsf{pk})$. *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* $\alpha, \beta \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. *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* Ω_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* Ω_2 *with* $\mathsf{NIZK.PRVer}(\mathsf{crs}, [\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$ *and check if* $e([\mathbf{u}_2]_1^\top, \mathbf{A}]_2) = e([\mathbf{w}]_1^\top, \mathbf{B}]_2)$.

An SPS-EQ scheme is secure if it satisfies correctness, EUF-CMA, and perfect adaption of signatures with respect to message space.

Definition 15 (Correctness). *An SPS-EQ scheme* SPSEQ *satisfies correctness, if the following properties hold for any* $\lambda > 0, \ell > 1, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*, all message* $\mathbf{m} \in (\mathbb{G}_i^*)^\ell$ *and all* $\mu, \rho \in \mathbb{Z}_p^*$*.*

$$\Pr[\mathsf{Verify}(\mathsf{pk}, \mathbf{m}, \mathsf{Sign}(\mathsf{sk}, \mathbf{m})) = 1] = 1 \wedge$$
$$\Pr[\mathsf{Verify}(\rho\mathsf{pk}, \mathsf{ChgRep}(\mathbf{m}, \mathsf{Sign}(\mathsf{sk}, \mathbf{m}), \mu, \rho, \mathsf{pk}) = 1] = 1.$$

Definition 16 (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 holds for any* $\lambda > 0, \ell > 1$ *and* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*.*

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

Definition 17 (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* σ^* *is a* random *element in the signature space such that* $\mathsf{Verify}([\rho \cdot \mathsf{pk}, \mu \cdot \mathbf{m}]_i, \sigma^*) = 1$*.*