Accountable and Fine-Grained Controllable Rewriting in Blockchains

Shengmin Xu, Xinyi Huang, Jiaming Yuan, Yingjiu Li, *Member, IEEE*, and Robert H. Deng, *Fellow, IEEE*

***Abstract*— Most blockchains are designed to be immutable such that an object, e.g., a block or a transaction, is persisted once it has been registered. However, blockchain immutability hinders blockchain development due to the increasing abuse of blockchain storage and legal obligations. To break immutability in a controlled way, Derler et al. (NDSS'19) proposed a redactable blockchain with fine-grained controllable rewriting by introducing the notion of policy-based chameleon hash (PCH). Given a PCH-based object associated with an access policy, a trapdoor holder whose rewriting privileges satisfy the access policy can alter the object. Although this work offers an elegant approach to blockchain rewriting, it lacks accountability. In practice, the trapdoor holders may abuse their rewriting privileges, and even use their chameleon trapdoor to build a device in a blackbox manner to gain illegal profits while avoiding being caught. In this paper, we introduce a new design of PCH with blackbox accountability (PCHA). Blackbox accountability offers not only linkability between any modified object and its modifier, but also traceability that enables a central authority to identify responsible trapdoor holders whose secret keys have contributed to the blackbox device. Besides modeling PCHAs, we present a generic construction of PCHAs with rigorous security proofs. We instantiate a concrete construction of PCHA by introducing a practical attribute-based traitor tracing (ABTT) with adaptive security on prime-order pairing groups. The experimental analysis demonstrates that our PCHA and ABTT schemes have modest overheads and superior functionality to the state-of-the-art solutions. In particular, the price of accountability in key generation, hash, and adaption is almost negligible compared to the state-of-the-art solution.**

***Index Terms*— Redactable blockchain, accountable blockchain rewriting.**

I. Introduction

BLOCKCHAIN technologies originating from Bitcoin [24] have attracted extensive attention in the last decade. Blockchain technologies have been widely used in large-scale applications, such as digital currency, supply chain, insurance, energy, and medical system. In a nutshell, blockchain is an immutable append-only ledger whose state is maintained by a decentralized consensus protocol run among peer nodes, where registered objects cannot be modified once they appear in the chain. The immutability of blockchain is essential to prevent anyone from manipulating the registered objects for illegitimate benefits. Unfortunately, this property limits the development of blockchain due to the increasing abuse of blockchain storage and legal obligations. Nowadays, many improper contents, e.g., child pornography, copyrighted material, and sensitive information, appear in the chain [23]. Users may be unwilling to participate in such a chain because of the fear of being prosecuted for possession of illegal information. Moreover, blockchain immutability is no longer legal since data regulations, such as "the right to be forgotten" [2] and the General Data Protection Regulation (GDPR) [31], grant individual users the right to delete their personal data, including those in blockchain [29].

To break blockchain immutability in a controlled way, Ateniese et al. [3] introduced the concept of the redactable blockchain that allows a particular party, called modifier, to rewrite a registered object without breaking the link between blocks. This seminal work offers block-level rewriting controlled at a coarse-grained level based on the technologies of chameleon hash [18] and public key infrastructure (PKI). To improve flexibility, Derler et al. [9] proposed a redactable blockchain with transaction-level rewriting controlled at a fine-grained level by introducing the notion of policy-based chameleon hash (PCH). PCH is built from chameleon hashes with ephemeral trapdoors (CHET) [8] and attribute-based encryption (ABE) [1]. Recently, based on [9], new construction of redactable blockchain [30] with accountability was proposed by introducing the notion of policy-based chameleon hash with blackbox accountability (PCHA). PCHA is built from CHET [8], ABE [1], hierarchical identity-based encryption (HIBE) [6] and one-time signature. It enables any user to trace malicious modifiers if their secret keys are revealed, and allows a central authority to link any modified object to its modifier. Since then, a variety of redactable blockchains are proposed with essential properties: accountability [30], self-management [17], revocability [27], [33], decentralization [22], [35] and functionality [16], [32]. This trend demonstrates that PCH is a desirable and valuable cryptographic tool for redactable blockchain. However, these solutions suffer from non-negligible shortcomings.

The PCH-based redactable blockchain [9] inherits the limitation of traditional ABE. The nature of ABE is anonymity,

Manuscript received 25 March 2022; revised 17 September 2022; accepted 20 October 2022. Date of publication 26 October 2022; date of current version 7 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62102090 and Grant 62032005 and in part by the Science Foundation of Fujian Provincial Science and Technology Agency under Grant 2020J02016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lejla Batina. *(Corresponding author: Xinyi Huang.)*

Shengmin Xu is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China (e-mail: smxu1989@gmail.com).

Xinyi Huang is with the Artificial Intelligence Thrust, Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: xinyi@ust.hk).

Jiaming Yuan and Yingjiu Li are with the Computer and Information Science School, University of Oregon, Eugene, OR 97403 USA (e-mail: jiamingy@uoregon.edu; yingjiul@uoregon.edu).

Robert H. Deng is with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: robertdeng@smu.edu.sg).

Digital Object Identifier 10.1109/TIFS.2022.3217742

where an attribute could be shared with multiple users. Given a secret key of ABE, no one including the central authority can identify its owner. A malicious modifier may reveal his/her secret key without being identified, for example, selling it on eBay to gain illegal profits. To solve this problem, a trivial solution is encoding the identity of the secret key owner to his/her secret key or keeping a list to record each secret key and its owner during the key generation phase. Given a revealed secret key, the central authority can easily identify the corresponding malicious modifier. To avoid being caught, multiple malicious users may collusively encode their secret keys into a device in a blackbox manner, denoted *key-like decryption box*, where these secret keys and even the decryption algorithm could be hidden. Given a mutable object in blockchain, the key-like decryption box can be used by anyone to modify the mutable object if the attributes of the keys in the key-like decryption box satisfy the access policy associated with the object. The trivial solution cannot be used to identify these malicious users because their secret keys are not revealed. Therefore, accountability is a desirable property for deploying redactable blockchain in practice.

To mitigate this problem, one possible approach is to combine CHET and *attribute-based traitor tracing* (ABTT). ABTT is a cryptographic tool that enables fine-grained access control and traitor tracing in a blackbox manner. ABTT was first envisioned by Liu et al. [19] and several efficient and secure instantiations of it were later introduced [20], [21], [25]. In ABTT, one can observe the I/O streams of a key-like decryption box to identify the malicious users whose secret keys have contributed to the key-like decryption box. However, one cannot naively employ the existing ABTT and CHET to extend PCH with accountability without sacrificing the performance or security of PCH. Specifically, most PCH-based redactable blockchain solutions [9], [30], [33] are instantiated using fast attribute-based message encryption (FAME) [1], where FAME is adaptively secure on prime-order pairing groups. Current ABTT solutions are either based on composite-order pairing groups or in the weak security model, i.e., selective security. The performance of composite-order pairing groups is much worse than prime-order pairing groups [12]. Though dual pairing vector space [26] can encode composite-order groups to prime-order groups, it incurs a significant overhead due to enormous encoding schemes [4].

Tian et al. [30] made the first effort to address this challenge by introducing a new design of ABTT that applies the HIBE and one-time signature. However, their solution only offers weak accountability, where semi-trusted modifiers can be caught if they leak their secret keys in blackbox manner. An untrusted modifier can launch the impersonation attack and signature-reuse attack without being identified. We summarize its fatal security issues as follows.

- *Impersonation Attack*. Key delegation and key indistinguishability are two important properties in HIBE. HIBE allows a central authority to issue a secret key associated with an identity, and the secret key can be used to derive multiple sub-keys associated with different identities (with the same prefix). Given a secret key, it is indistinguishable that this secret key is derived from the central authority or a secret key holder. Hence, a malicious key holder may derive a sub-key and use it in forming a key-like decryption box to avoid being identified.
- *Signature-reuse Attack*. One-time signatures do not offer user authenticity. Based on the generic construction of PCHA proposed in [30], the verification algorithm only takes into account the validation of the chameleon hash and the one-time signature. It does not check the freshness of the one-time signature and the identity of the corresponding signer. A malicious modifier whose attributes satisfy the access policy of PCHA can modify the signed object and reuse the one-time signature associated with the original object to pass verification.

Besides [30], Panwar et al. [27] proposed a solution to trace malicious modifiers in blockchain rewriting. However, their solution only traces the malicious users who perform malicious rewriting by themselves; the leakage of secret keys has not been taken into account.

To sum up, current redactable blockchain solutions only offer weak accountability. The modifiers who perform malicious rewriting by themselves can be identified but the modifiers who leak their secret keys in blackbox manner cannot be traced properly. In other words, malicious modifiers may collusively derive key-like decryption boxes and use them to compromise the rewriting privileges in redactable blockchain without being identified. Therefore, the following problem arises:

"Can we build a practical and secure redactable blockchain with fine-grained control and blackbox accountability, including linkability and traceability?"

A. Contribution

We give an affirmative answer to the above problem by introducing the concept of accountability toward redactable blockchain, where modifiers are untrusted. Malicious modifiers can always be identified no matter they make malicious modifications by themselves in redactable blockchain or they leak their secret keys in blackbox manner. The main contributions are described as follows.

- *A new construction of PCHA*. We introduce a new generic construction of *policy-based chameleon hash with blackbox accountability* (PCHA). We give a formal definition of PCHA and define the notions of *full indistinguishability*, *collision-resistance*, *uniqueness*, and *accountability* in presence of malicious behaviors. Compared to PCH [9], the price of accountability in terms of hash and adaption is almost negligible.
- *A new design of ABTT*. To optimize the performance of PCHA, we present a new design of *attribute-based traitor tracing* (ABTT), which is adaptively secure on prime-order pairing groups. Our ABTT enjoys the constant secret key and ciphertext overheads. Specifically, compared to FAME [1], our ABTT requires 2 additional group elements in a secret key and 6 additional group elements in a ciphertext.

- *A redactable blockchain with accountability*. Although ABTT offers traitor tracing in the key-like decryption box, it cannot provide the accountability that links the modified object to its modifier. To realize this property, we apply the existentially unforgeable identity-based signature (IBS) [15] to link any modified object to its modifier publicly, where the identity associated with each modifier is the same as the identity used in ABTT.

To sum up, we pursue to carefully integrate ABTT, IBS [15], CHET [8] with the concept of PCHA and present a generic construction of PCHAs. Compared to the previous PCH [9], our PCHA achieves accountability with modest overhead. We demonstrate that our PCHA is practical by integrating it into redactable blockchain for transaction-level blockchain rewriting in Section V-E. We support this claim via an experimental implementation.

B. Roadmap

The rest of the paper is structured as follows. We formulate the problem and system architecture in Section II. Then, we review several important cryptographic building blocks used in our proposed solution in Section III. Next, we formalize our proposed approach by presenting a new design of ABTT in Section IV and PCHA with a blockchain-based application in Section V. We report the experimental evaluation of our proposed ABTT and PCHA in Section VI. Finally, we present the related work in Section VII before concluding our work in Section VIII.

II. System Overview

In this section, we formulate the problem of current redactable blockchains and discuss its challenges. Before presenting the formulation, we give a running example to illustrate the problem and its relevant concepts.

A. System Model

As shown in Fig. 1, the redactable blockchain allows a data *owner* to compute a chameleon hash associated with an access policy and a signature. A *modifier*, who possesses privileges that satisfy the access policy in a given hash can then find arbitrary collisions to modify the hashed object in the blockchain without changing its hash value. The modifier also needs to produce a valid signature, which links the chameleon hash value to its producer. In the case that a malicious modifier does not make malicious rewriting themselves but leaks his/her secret key in a key-like decryption box, the central authority can trace the malicious user whose secret key has contributed to the key-like decryption box.[1] Thus, the redactable blockchain supports modifiability at a fine-grained level and accountability.

[1]The key-like decryption box offers the same functions as the secret keys embedded in the key-like decryption box except for the secret keys and even the decryption algorithm remains hidden. The reader may refer to Section V-A for the detailed definition.

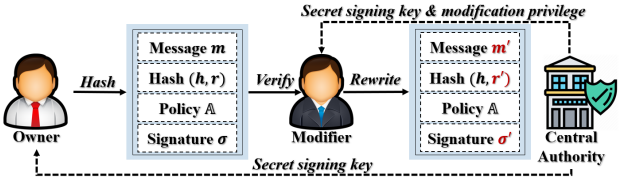


Fig. 1. System model of PCHA.

B. Threat Model

We assume that the central authority and transaction owners are fully trusted, while transaction modifiers and outsiders (i.e., blockchain users who cannot modify blockchain objects) are untrusted. In the following, we focus on the attacks performed by malicious modifiers and the security requirements that inherit from PCH. For simplicity, we do not specify the attacks performed by outsiders since they are covered by the attacks which malicious modifiers can perform. In practice, a transaction modifier may abuse his/her rewriting privilege, and launch the following attacks:

- *Collision-Resistance*: Given a registered object associated with an access policy, the malicious modifier whose attributes do not match the access policy may attempt to modify the registered object.
- *Accountability*: The malicious modifiers may attempt to break the accountability of redactable blockchain by impersonating other modifiers in rewriting the registered object, and/or encapsulating their secrets key in a black-box manner to derive a key-like decryption box so that other users may use the decryption box to modify the registered object anonymously.

We consider the following security requirements that inherit from PCH:

- *Full Indistinguishability*: The outputs of hashing and collision finding are indistinguishable for adversarially chosen messages and the public key. In PCH, this security requirement focuses on the hash value only. In our PCHA, we require that the hash value and signature pair cannot be distinguished from hashing and collision finding.
- *Uniqueness*: Uniqueness requires that it is hard to find different randomnesses yielding the same hash value for adversarially chosen messages and the public key.

To formalize the above attacks and security requirements in the threat model, we present four security models in Section V-B.

III. Cryptographic Building Blocks

In this section, we present some building blocks used in our solution. In particular, we introduce the fingerprint code, which is used as an ingredient in our ABTT construction. Then, we recall the formal definition of ABTT. Finally, we review the formal definition of CHET and IBS, which are two important building blocks in our PCHA construction.

A. Fingerprint Code

Please refer to the detailed description of fingerprint code and collision-resistant (CR) security of fingerprint code in [6]. The definition of fingerprint code is presented below.

Definition 1 (Fingerprint Code): A fingerprint code scheme $\mathcal{FC}$ is a pair of algorithms defined as follows:

$\mathcal{FC}.\mathsf{Gen}(1^\kappa, n, t) \to (\Gamma, tk)$*: The probabilistic code generator algorithm takes a security parameter $\kappa \in \mathbb{N}$, the number of words n and the collusion bound t as input, and outputs a code Γ and a tracing key tk, where Γ contains n words $\{\bar{w}_1, \ldots, \bar{w}_n\}$ in $\{0,1\}^\ell$ for some $\ell > 0$.*

$\mathcal{FC}.\mathsf{Trace}(tk, \bar{w}^*) \to S$*: The deterministic tracing algorithm takes a tracing key tk and a word $\bar{w}^* \in \{0,1\}^\ell$, and outputs a subset S of $\{1, \ldots, n\}$. Informally, elements in S are "accused" of creating the word $\bar{w}$.*

B. Attribute-Based Traitor Tracing

Please refer to the detailed description of ABTT and IND-CPA security and collision-resistant traceability (CRT) of ABTT in [19]. The definition of ABTT is presented below.

Definition 2 (ABTT): An ABTT scheme $\mathcal{ABTT}$ with an attribute universe Ω, a message space $\mathcal{M}$ and an identity space $\mathcal{I}$ consists of the following six algorithms:

$\mathcal{ABTT}.\mathsf{ParGen}(1^\kappa) \to pp$*: The probabilistic parameter generation algorithm takes a security parameter $\kappa \in \mathbb{N}$ as input, and outputs a public parameter pp, where pp is implicit input to all other algorithms.*

$\mathcal{ABTT}.\mathsf{Setup}(n, t) \to (mpk, msk)$*: The probabilistic setup algorithm takes the number of users n and the collusion bound t as input, and outputs a master public key mpk and a master secret key msk.*

$\mathcal{ABTT}.\mathsf{KeyGen}(msk, \mathcal{S}, \bar{w}) \to sk_{\mathcal{S},\bar{w}}$*: The probabilistic key generation algorithm takes a master secret key msk, a set of attributes $\mathcal{S} \subseteq \Omega$ and an identity $\bar{w} \in \mathcal{I}$ as input, and outputs a secret key $sk_{\mathcal{S},\bar{w}}$.*

$\mathcal{ABTT}.\mathsf{Enc}(mpk, m, \mathbb{A}) \to c$*: The probabilistic encryption algorithm takes a master public key mpk, a message $m \in \mathcal{M}$ and an access policy $\mathbb{A}$ as input, and outputs a ciphertext c.*

$\mathcal{ABTT}.\mathsf{Dec}(sk_{\mathcal{S},\bar{w}}, c) \to m$*: The deterministic decryption algorithm takes a secret key $sk_{\mathcal{S},\bar{w}}$ and a ciphertext c as input, and outputs a message $m \in \mathcal{M}$.*

$\mathcal{ABTT}.\mathsf{Trace}(msk, \mathcal{D}) \to S$*: The deterministic tracing algorithm takes a master secret key msk and a key-like decryption box $\mathcal{D}$ (associated with a set of attributes $\mathcal{D}_\mathcal{S}$) as input, and outputs a subset S of $\{1, \ldots, n\}$, where $\mathcal{D}$ can output m by giving c with $c \leftarrow \mathcal{ABTT}.\mathsf{Enc}(mpk, m, \mathbb{A})$ and $\mathcal{D}_\mathcal{S} \in \mathbb{A}$.*

C. Chameleon Hash With Ephemeral Trapdoor

CHET [8] is a variant of a chameleon-hash used to design PCH [9]. The security of CHET consists of *full indistinguishability*, *public collision-resistance*, *strong private collision-resistance* and *uniqueness*. We refer the reader to [28] for a detailed and well-written definition of the security model. The definition of CHET is presented below.

Definition 3 (CHET): A CHET scheme $\mathcal{CHET}$ with a message space $\mathcal{M}$ consists of the following five algorithms:

$\mathcal{CHET}.\mathsf{ParGen}(1^\kappa) \to pp$*: The probabilistic parameter generation algorithm takes a security parameter $\kappa \in \mathbb{N}$ as input, and outputs a public parameter pp.*

$\mathcal{CHET}.\mathsf{KeyGen}(pp) \to (sk, pk)$*: The probabilistic key generation algorithm takes a public parameter pp as input, and outputs a secret key sk and a public key pk.*

$\mathcal{CHET}.\mathsf{Hash}(pk, m) \to (h, r, \mathsf{etd})$*: The probabilistic hash algorithm takes a public key pk and a message $m \in \mathcal{M}$ as input, and outputs a hash h, a randomness r (sometimes referred to as "check value") and an ephemeral trapdoor* etd.

$\mathcal{CHET}.\mathsf{Verify}(pk, m, h, r) \to \{0,1\}$*: The deterministic verification algorithm takes a public key pk, a message $m \in \mathcal{M}$, a hash h and a randomness r as input, and outputs a bit indicating if (m, h, r) is valid.*

$\mathcal{CHET}.\mathsf{Adapt}(sk, \mathsf{etd}, m, m', h, r) \to r'$*: The deterministic adaption algorithm takes a secret key sk, an ephemeral trapdoor* etd*, a message $m \in \mathcal{M}$, a message $m' \in \mathcal{M}$, a hash h and a randomness r as input, and outputs a randomness r'.*

D. Identity-Based Signature

IBS allows users to verify digital signatures via the user's identifier rather than the public key to mitigate the need for the certificate authority. We refer the reader to [15] for a detailed and well-written security model, *existential unforgeability*, for IBS. The definition of IBS is presented below.

Definition 4 (IBS): An IBS scheme $\mathcal{IBS}$ with a message space $\mathcal{M}$ and an identity space $\mathcal{I}$ consists of the following five algorithms:

$\mathcal{IBS}.\mathsf{ParGen}(1^\kappa) \to pp$*: The probabilistic parameter generation algorithm takes a security parameter $\kappa \in \mathbb{N}$ as input, and outputs a public parameter pp.*

$\mathcal{IBS}.\mathsf{Setup}(pp) \to (mpk, msk)$*: The probabilistic setup algorithm takes a public parameter pp as input, and outputs a master public key mpk and a master secret key msk.*

$\mathcal{IBS}.\mathsf{KeyGen}(msk, \bar{w}) \to sk_{\bar{w}}$*: The probabilistic key generation takes a master secret key msk and an identity $\bar{w} \in \mathcal{I}$ as input, and outputs a secret key $sk_{\bar{w}}$.*

$\mathcal{IBS}.\mathsf{Sign}(sk_{\bar{w}}, m) \to \sigma_{\bar{w}}$*: The probabilistic signing algorithm takes a secret key $sk_{\bar{w}}$ and a message $m \in \mathcal{M}$ as input, and outputs a signature $\sigma_{\bar{w}}$.*

$\mathcal{IBS}.\mathsf{Verify}(mpk, \bar{w}, m, \sigma_{\bar{w}}) \to \{0,1\}$*: The deterministic verification algorithm takes a master public key mpk, an identity $\bar{w} \in \mathcal{I}$, a message m and a signature $\sigma_{\bar{w}}$ as input, and outputs a bit indicating if $\sigma_{\bar{w}}$ is valid.*

IV. Attribute-Based Traitor Tracing

In this section, we show a new design of ABTT and rigorously prove the security of our proposed construction. The efficiency of this concrete construction is confirmed with practical implementation in Section VI.

A. Concrete Construction of ABTT

Our ABTT is based on an adaptively secure ABE [1], a fully collision-resistant fingerprint code [7], and a selectively secure ABTT [34]. Before presenting the concrete construction, we introduce the overall ideas of our design.

1) Sketch of ABTT: We give a sketch of our ABTT about its construction and the realization of blackbox traceability.

- Our concrete construction starts from the adaptively secure ABE [1]. However, it lacks accountability. An intuitive idea to add accountability is to apply the fully collision-resistant fingerprint code [7] to bind each secret key of ABE to its owner. Unfortunately, such fingerprint

Let $\Omega = \{0,1\}^*$ denote the attribute universe, $\mathcal{M} = \{0,1\}^{\ell_m}$ be the message space, $\mathcal{I} = \{0,1\}^{\ell}$ denote the identity space, and $\mathcal{FC} = \{\mathsf{Gen}, \mathsf{Trace}\}$ be a collision-resistant fingerprint code. The concrete construction of ABTT is presented below:

$\mathcal{ABTT}.\mathsf{ParGen}(1^\kappa) \to pp$: Run group generator $\mathcal{G}(1^\kappa)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Return $pp = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$.

$\mathcal{ABTT}.\mathsf{Setup}(n, t) \to (mpk, msk)$:

1) Run $\mathcal{FC}.\mathsf{Gen}(1^\kappa, n, t)$ to obtain (Γ, tk), where Γ contains n words in $\{0,1\}^\ell$ for some $\ell > 0$.
2) Pick $a_1, a_2, b_1, b_2 \in \mathbb{Z}_p^*$. Set $H_1 = h^{a_1}$ and $H_2 = h^{a_2}$.
3) Pick $d_1, d_2, d_3 \in \mathbb{Z}_p$. Set $T_1 = e(g,h)^{d_1 a_1 + d_3}$ and $T_2 = e(g,h)^{d_2 a_2 + d_3}$.
4) Pick $\alpha \in \mathbb{Z}_p^*$. Set $g_1 = g^\alpha$ and $h_i = h^{(\alpha^i)}$ for all $i \in \{1, \ldots, \ell\}$.
5) Pick hash functions $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{H}_3 : \mathbb{G}_T \to \{0,1\}^{\ell_m}$.
6) Return $mpk = (g_1, h_1, \ldots, h_\ell, H_1, H_2, T_1, T_2, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$ and $msk = (\Gamma, tk, \alpha, a_1, a_2, b_1, b_2, g^{d_1}, g^{d_2}, g^{d_3})$.

$\mathcal{ABTT}.\mathsf{KeyGen}(msk, \mathcal{S}, \bar{w}) \to sk_{\mathcal{S},\bar{w}}$: Parse $\bar{w} = w_1 \ldots w_\ell$ and $\bar{w} \in \Gamma$.

1) Pick $r_1, r_2 \in \mathbb{Z}_p$. Set $sk_0 = (sk_{0,1}, sk_{0,2}, sk_{0,3}) = (h^{b_1 r_1}, h^{b_2 r_2}, h^{r_1 + r_2})$
2) For all $y \in \mathcal{S}$ and $z = 1, 2$:
   Pick $\sigma_y \in \mathbb{Z}_p$. Compute $sk_{y,z} = \mathcal{H}_1(y1z)^{\frac{b_1 r_1}{a_z}} \cdot \mathcal{H}_1(y2z)^{\frac{b_2 r_2}{a_z}} \cdot \mathcal{H}_1(y3z)^{\frac{r_1 + r_2}{a_z}} \cdot g^{\frac{\sigma_y}{a_z}}$. Set $sk_y = (sk_{y,1}, sk_{y,2}, g^{-\sigma_y})$.
3) For $z = 1, 2$:
   Pick $\sigma' \in \mathbb{Z}_p$. Compute $sk'_z = g^{d_z} \cdot \mathcal{H}_1(011z)^{\frac{b_1 r_1}{a_z}} \cdot \mathcal{H}_1(012z)^{\frac{b_2 r_2}{a_z}} \cdot \mathcal{H}_1(013z)^{\frac{r_1 + r_2}{a_z}} \cdot g^{\frac{\sigma'}{a_z}}$. Set $sk' = (sk'_1, sk'_2, g^{d_3} \cdot g^{-\sigma'})$.
4) For $b = 0, 1$:
   Let $\mathcal{W}_b \subseteq \{1, \ldots, \ell\}$ be the set of all i for which $w_i = b$. Compute $sk_{\mathcal{W}_b} = g^{\sum_{i \in \mathcal{W}_b} \frac{1}{\alpha - \mathcal{H}_2(ib)}}$. Set $sk_{\bar{w}} = (sk_{\mathcal{W}_0}, sk_{\mathcal{W}_1})$.
5) Return $sk_{\mathcal{S},\bar{w}} = (sk_0, \{sk_y\}_{y \in \mathcal{S}}, sk', sk_{\bar{w}})$.

$\mathcal{ABTT}.\mathsf{Enc}(mpk, m, \mathbb{A}) \to c$: Parse $\mathbb{A} = (\mathbb{M}, \pi)$, where $\mathbb{M}$ has n_1 rows and n_2 columns.

1) Pick $m_0 \in \mathcal{M}$. Set $m_1 = m \oplus m_0$.
2) Pick $s_1, s_2 \in \mathbb{Z}_p$. Set $c_0 = (c_{0,1}, c_{0,2}, c_{0,3}) = (H_1^{s_1}, H_2^{s_2}, h^{s_1 + s_2})$ and $c' = m_0 \oplus \mathcal{H}_3(T_1^{s_1} \cdot T_2^{s_2})$.
3) For $i = 1, \ldots, n_1$ and $u = 1, 2, 3$:
   Compute $c_{i,u} = \mathcal{H}_1(\pi(i)u1)^{s_1} \cdot \mathcal{H}_1(\pi(i)u2)^{s_2} \cdot \prod_{j=1}^{n_2} [\mathcal{H}_1(0ju1)^{s_1} \cdot \mathcal{H}_1(0ju2)^{s_2}]^{\mathbb{M}_{i,j}}$, where $\mathbb{M}_{i,j}$ denotes the (i,j)-th elemnet of $\mathbb{M}$. Set $c_i = (c_{i,1}, c_{i,2}, c_{i,3})$.
4) Pick $\psi \in \{1, \ldots, n\}$. For $b = 0, 1$:
   Pick $r \in \mathbb{Z}_p$. Compute $c_{\psi,b,1} = (h^{\sum_{i=1}^{\ell}(\alpha - \mathcal{H}_2(ib))})^r$, $c_{\psi,b,2} = (g^{\alpha - \mathcal{H}_2(\psi b)})^r$ and $c_{\psi,b,3} = m_1 \oplus \mathcal{H}_3(e(g, h^{\sum_{i=1, i \neq \psi}^{\ell}(\alpha - \mathcal{H}_2(ib))})^r)$. Set $c_{w_\psi,b} = (c_{\psi,b,1}, c_{\psi,b,2}, c_{\psi,b,3})$ and $c_{w_\psi} = (c_{w_\psi,0}, c_{w_\psi,1})$.
5) Return $c = (\psi, c_0, c_1, \ldots, c_{n_1}, c', c_{w_\psi})$.

$\mathcal{ABTT}.\mathsf{Dec}(sk_{\mathcal{S},\bar{w}}, c) \to m$: If the attribute set $\mathcal{S}$ in $sk_{\mathcal{S},\bar{w}}$ satisfies the access policy $\mathbb{A} = (\mathbb{M}, \pi)$ in c, then there exists a constant set $\{\gamma_i\}_{i \in I}$ that satisfies $\sum_{i \in I} \gamma_i \mathbb{M}_i = (1, 0, \ldots, 0)$.

1) Compute $num = e(\prod_{I \in i} c_{i,1}^{\gamma_i}, sk_{0,1}) \cdot e(\prod_{I \in i} c_{i,2}^{\gamma_i}, sk_{0,2}) \cdot e(\prod_{I \in i} c_{i,3}^{\gamma_i}, sk_{0,3})$ and $dec = e(sk'_1 \cdot \prod_{i \in I} sk_{\pi(i),1}^{\gamma_i}, c_{0,1}) \cdot e(sk'_2 \cdot \prod_{i \in I} sk_{\pi(i),2}^{\gamma_i}, c_{0,2}) \cdot e(sk'_3 \cdot \prod_{i \in I} sk_{\pi(i),3}^{\gamma_i}, c_{0,3})$. Set $m_0 = c' \oplus \mathcal{H}_3(num/dec)$.
2) Set $b = w_\psi$. Let $f(x) = \sum_{i=1}^{\ell}(x - \mathcal{H}_2(ib)) \cdot \sum_{j \in \mathcal{W}_b} \frac{1}{x - \mathcal{H}_2(jb)}$. Because of $\ell \geq |\mathcal{W}_b|$ *and* $\psi \in \{1, \ldots, \ell\} \cap \mathcal{W}_b$, we can write $f(x)$ as $f(x) = \frac{\sum_{i=1}^{\ell}(x - \mathcal{H}_2(ib))}{x - \mathcal{H}_2(\psi b)} + (x - \mathcal{H}_2(\psi b)) \cdot \sum_{i=0}^{|\mathcal{W}_b|-2} f_i x^i$, where f_i is the coefficient of x^i. Set $m_1 = c_{\psi,b,3} \oplus \mathcal{H}_3(e(sk_{\mathcal{W}_b}, c_{\psi,b,1}) \cdot e(c_{\psi,b,2}, \prod_{i=1}^{|\mathcal{W}_b|-2} h_i^{f_i} \cdot h^{f_0})^{-1})$.
3) Return $m = m_0 \oplus m_1$.

$\mathcal{ABTT}.\mathsf{Trace}(msk, \mathcal{D}) \to S$: The key-like decryption box $\mathcal{D}$ is associated with a set of attributes $\mathcal{D}_{\mathcal{S}}$.

1) Let $\bar{w}^* = w_1^* \ldots w_\ell^*$ in $\{0,1\}^\ell$. Modify $\mathcal{ABTT}.\mathsf{Enc}(m, \mathbb{A})$ *to* $\mathcal{ABTT}.\mathsf{mEnc}(m, \mathbb{A})$ with the different c_{w_ψ} component:
   - For $b = 0$, keep the original encryption algorithm with the message $m_1 \neq 0$.
   - For $b = 1$, change the original encryption algorithm by setting the message $m_1 = 0$.
2) For $\psi = 1, \ldots, \ell$:
   Pick $m \in \mathcal{M}$. Run $\mathcal{ABTT}.\mathsf{mEnc}(m, \mathbb{A})$ (for some $\mathbb{A}$ s.t. $\mathcal{D}_{\mathcal{S}} \in \mathbb{A}$) to obtain c. Execute $\mathcal{D}(c)$ to obtain m'. Set $w_\psi^* = 0$ if $m = m'$; otherwise, set $w_\psi^* = 1$.
3) Return $S \leftarrow \mathcal{FC}.\mathsf{Trace}(tk, \bar{w}^*)$.

Fig. 2. Concrete construction of ABTT.

code increases the sizes of the secret key and the ciphertext linearly to the number of users. To address this issue, we further apply the cryptographic accumulator used in the selectively secure ABTT [34] to reduce the linear complexity to the constant size. We detail the realization of blackbox traceability and efficiency of our ABTT in the rest of this subsection.

- The parameter generation algorithm outputs a group description as the public parameter. The setup algorithm runs the setup algorithm of ABE and the code generator algorithm of fingerprint code to derive the master public key and master secret key. Every participant is assigned a unique identity $\bar{w}$ and is issued a secret key $sk_{\mathcal{S},\bar{w}}$ associated with a set of attributes $\mathcal{S}$ by the key generation algorithm. Given a message m and an access policy $\mathbb{A}$, every participant can run the encryption algorithm to derive a ciphertext c. Only the participants with the set of attributes $\mathcal{S}$ such that $\mathcal{S} \in \mathbb{A}$ can decrypt the ciphertext c by running the decryption algorithm. Once a key-like decryption box $\mathcal{D}$ is found, the tracing key holder can run the tracing algorithm to obtain a set of identities S. Informally, the identities in S are "accused" of creating the key-like decryption box $\mathcal{D}$. In other words, our ABTT supports blackbox traceability.

2) Sketch of Technical Overview: We give the sketch of a technical overview from the secret key, ciphertext, and traitor tracing.

- *Secret Key*: The secret key can be divided into two components. One is the attribute-based component that provides fine-grained access control. The other is an identity-based component based on the fingerprint code.

In particular, each data user has a unique identity $\bar{w}$ in $\{0,1\}^\ell$ and the malicious user can be traced based on the identity-based component. To improve the efficiency, we compress the ℓ-bit identity into constant size. Specifically, two collections that recall all i for $\bar{w}_i = 0$ and $\bar{w}_i = 1$ are used.

- *Ciphertext*: To encrypt a message $m = m_0 \oplus m_1$, we first encrypt m_0 based on the access policy $\mathbb{A}$ and then encrypt m_1 depending on a random challenge bit at the position $\psi \in \{1, \ldots, \ell\}$. Hence, the ciphertext has two components: m_0 under $\mathbb{A}$ that can be decrypted with $\mathcal{S} \in \mathbb{A}$ and m_1 that tests the ψ-th bit to be either 0 or 1, where testing the ψ-th bit to be either 0 or 1 is not important for the data confidentiality but for traitor tracing.
- *Traitor Tracing*: As we mentioned before, the ciphertext has two components. In the tracing algorithm, the ciphertext is generated based on two different messages. For all $\psi \in \{1, \ldots, \ell\}$, the tracing algorithm 1) chooses two messages m_0 and $m_1 \neq 0$, then encrypts $m_0 \oplus m_1$ for $w_\psi = 0$ and $m_0 \oplus 0$ for $w_\psi = 1$; 2) inputs each ciphertext into the key-like decryption box and observes the output m' to derive a word $\bar{w}^* = w_1^* \ldots w_\ell^*$ as: $w_\psi^* = 0$ if $m_0 \oplus m_1 = m'$, and $w_\psi^* = 1$ otherwise. Although anyone can derive a word $\bar{w}^*$ from a decryption-key like box, the secret tracing key held by the central authority is required to reveal the modifier(s) from the word $\bar{w}^*$ according to the definition of the tracing algorithm in the fingerprint code. Finally, the central authority runs the tracing algorithm of the fingerprint code to trace the malicious users.

3) Selection of Suitable Parameters: Subsequently, we discuss the parameters of bilinear groups and fingerprint codes.

- *Bilinear Groups*. We assume that the reader is familiar with bilinear maps. We use the group description $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$ for a Type-III pairing and MNT6-992 curve with a 156-bit security level as an instantiation [13].
- *Fingerprint Codes*. We apply fully collision-resistant codes. For n users, the fingerprint code is built from the following set of n words Γ_0.

	Block 0	*Block* 1		⋯		*Block* n
Word 1:	0000	1111	1111		1111	1111
Word 2:	0000	0000	1111	⋯	1111	1111
Word 3:	0000	0000	0000		1111	1111
⋮	⋮		⋮		⋮	
Word n:	0000	0000	0000	⋯	0000	1111

The code generator $\mathcal{FC}.\mathsf{Gen}$ chooses a random permutation π on $(1, \ldots, \ell)$ and permutes the columns of Γ_0 according to π to output codewords π with tracing key $tk = \pi$. The tracing algorithm $\mathcal{FC}.\mathsf{Trace}$ computes the quantities $a_0, \ldots, a_n$ and uses the tracing key π to undo the random permutation to infer the adversary who has codeword number i with overwhelming probability. We refer the reader to [7] for a detailed and well-written description of the fingerprint code.

Our ABTT uses three hash functions $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$ and $\mathcal{H}_3 : \mathbb{G}_T \to \{0,1\}^{\ell_m}$, where ℓ_m is the bit-length of each message and these hash functions are modeled as random oracles in the security proof. The concrete construction of ABTT is given in Fig. 2.

B. Security Analysis of ABTT

Theorem 1: The proposed ABTT scheme is IND-CPA *secure if the DLIN assumption holds in the random oracle model.*

To prove the IND-CPA security of our ABTT, we apply the security reduction. In particular, if there exists a probabilistic polynomial-time adversary who can break the IND-CPA security of our ABTT with non-negligible advantage, we can construct a probabilistic polynomial-time simulator to break the IND-CPA security of FAME [1], where FAME has been proved IND-CPA secure if the DLIN assumption holds. There is no security loss during our security reduction. Hence, our ABTT scheme is IND-CPA secure if the DLIN assumption holds. We provide a proof of this theorem in Appendix VIII-B.

Theorem 2: The proposed ABTT scheme has collusion-resistant traceability in the random oracle model if the modified q-BDHE assumption holds and the underlying fingerprint code is collision-resistant.

To prove the collusion-resistant traceability of our ABTT, we apply the security reduction approach. In particular, if there exists a probabilistic polynomial-time adversary who can break the collusion-resistant traceability of our ABTT with a non-negligible advantage, we can construct a probabilistic polynomial-time simulator to break the modified q-BDHE assumption and collision-resistance of fingerprint code. Specifically, the simulator uses the given identity set from fingerprint code and elements of the modified q-BDHE problem to train the adversary. After receiving the key-like decryption box, the simulator can reveal the identities whose secret keys are used in constructing this key-like decryption box to break the collision-resistance of fingerprint code, and query the key-like decryption box to find the target element in the modified q-BDHE assumption. Please refer to Appendix VIII-A for the formal definition of the modified q-BDHE assumption, and Appendix VIII-C for the detailed proofs.

V. Policy-Based Chameleon Hash With Blackbox Accountability

In this section, we show a formal definition and a security model of PCHA, and give a generic construction of PCHA with a rigorous security proof. We then present a concrete construction of PCHA based on the proposed generic construction and Fujisaki-Okamoto transformation [11].

A. Definition of PCHA

Definition 5 (PCHA): A PCHA scheme $\mathcal{PCHA}$ with an attribute universe Ω, a message space $\mathcal{M}$ and an identity space $\mathcal{I}$, involves three types of parties: a central authority, transaction owners and transaction modifiers, and consists of the following eight algorithms:

$\mathcal{PCHA}.\mathsf{ParGen}(1^\kappa) \rightarrow pp$*: The probabilistic setup algorithm is run by the central authority. It takes a security parameter* $\kappa \in \mathbb{N}$ *as input, and outputs a public parameter* pp*, where* pp *is implicit input to all other algorithms.*

$\mathcal{PCHA}.\mathsf{Setup}(n, t) \rightarrow (mpk, msk)$*: The probabilistic setup algorithm is run by the central authority. It takes the number of users* n *and the collusion bound* t *as input, and outputs a master public key* mpk *and a master secret key* msk*.*

$\mathcal{PCHA}.\mathsf{UKeyGen}(msk, \bar{w}) \rightarrow sk_{\bar{w}}$*: The probabilistic user key algorithm is run by the central authority. It takes a master secret key* msk *and an identity* $\bar{w} \in \mathcal{I}$ *as input, and outputs a secret key* $sk_{\bar{w}}$*.*

$\mathcal{PCHA}.\mathsf{MKeyGen}(msk, \mathcal{S}, \bar{w}') \rightarrow (sk_{\bar{w}'}, sk_{\mathcal{S},\bar{w}'})$*: The probabilistic modifier key generation algorithm is run by the central authority. It takes a master secret key* msk*, a set of attributes* $\mathcal{S} \subseteq \Omega$ *and an identity* $\bar{w}' \in \mathcal{I}$*, and outputs two secret keys* $sk_{\bar{w}'}$ *and* $sk_{\mathcal{S},\bar{w}'}$*.*

$\mathcal{PCHA}.\mathsf{Hash}(mpk, sk_{\bar{w}}, m, \mathbb{A}) \rightarrow (h, r, \sigma_{\bar{w}})$*: The probabilistic hash algorithm is run by the transaction owner* $\bar{w}$*. It takes a master public key* mpk*, a secret key* $sk_{\bar{w}}$*, a message* $m \in \mathcal{M}$ *and an access policy* $\mathbb{A}$ *as input, and outputs a hash* h*, a randomness (sometimes referred to as "check value")* r *and a signature* $\sigma_{\bar{w}}$*.*

$\mathcal{PCHA}.\mathsf{Verify}(mpk, \bar{w}, m, h, r, \sigma_{\bar{w}}) \rightarrow \{0, 1\}$*: The deterministic verification algorithm is run by any party. It takes a master public key* mpk*, an identity* $\bar{w} \in \mathcal{I}$*, a message* $m \in \mathcal{M}$*, a hash* h*, a randomness* r *and a signature* $\sigma_{\bar{w}}$*, and outputs a bit indicating if* (h, r) *is valid hash and* $(\bar{w}, \sigma_{\bar{w}})$ *is valid signature.*

$\mathcal{PCHA}.\mathsf{Adapt}(mpk, \bar{w}, sk_{\bar{w}'}, sk_{\mathcal{S},\bar{w}'}, m, m', h, r, \sigma_{\bar{w}}) \rightarrow (r', \sigma_{\bar{w}'})$*: The probabilistic adaption algorithm is run by the modifier* $\bar{w}'$*. It takes a master public key* mpk*, an identity* $\bar{w} \in \mathcal{I}$*, a secret key* $sk_{\bar{w}'}$*, a secret key* $sk_{\mathcal{S},\bar{w}'}$*, a message* $m \in \mathcal{M}$*, a message* $m' \in \mathcal{M}$*, a hash* h*, a randomness* r *and a signature* $\sigma_{\bar{w}}$*, and outputs a randomness* r' *and a signature* $\sigma_{\bar{w}'}$*.*

$\mathcal{PCHA}.\mathsf{Trace}(msk, \mathcal{D}) \rightarrow S$*: The deterministic tracing algorithm is run by the central authority. It takes a master secret key* msk*, a key-like decryption box* $\mathcal{D}$ *(associated with a set of attributes* $\mathcal{D}_\mathcal{S}$*) as input, and outputs a subset* $S \subseteq \{1, \ldots, n\}$*. Given* $(m, m', \mathcal{PCHA}.\mathsf{Hash}(mpk, sk_{\bar{w}}, m, \mathbb{A}))$ *with* $\mathcal{D}_\mathcal{S} \in \mathbb{A}$*,* $\mathcal{D}$ *can output* $(r', \sigma_{\bar{w}'})$ *with* $\mathcal{PCHA}.\mathsf{Verify}(mpk, \bar{w}, m', h, r', \sigma_{\bar{w}'}) = 1$*.*

B. Security Model of PCHA

1) Full Indistinguishability: Indistinguishability requires that it should be infeasible to decide whether a hash digest and the corresponding signature are fresh or were from the adaption algorithm. In the previously indistinguishable model [9], the adversary must follow the keys chosen by the challenger to launch the attacks, which is a relatively weak model compared to the fully indistinguishable model. Our full indistinguishability allows the adversary to choose the signing key and the secret key used in the HashOrAdapt oracle, where the signing key and the secret key should belong to a single user to prevent trivial attacks. The security experiment grants the adversary access to a HashOrAdapt oracle and requires that

$\mathbf{Exp}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t)$
  $pp \leftarrow \mathcal{PCHA}.\mathsf{ParGen}(1^\kappa), b \in \{0, 1\}$
  $(mpk, b') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{HashOrAdapt}}(\cdot)}(pp, n, t)$
    where $\mathcal{O}_{\mathsf{HashOrAdapt}}(\cdot, \cdot, \cdot, \cdot, \cdot)$ *on input* $sk_{\bar{w}}, sk_{\mathcal{S},\bar{w}}, m, m', \mathbb{A}$:
      $(h_0, r_0, \sigma_0) \leftarrow \mathcal{PCHA}.\mathsf{Hash}(mpk, sk_{\bar{w}}, m', \mathbb{A})$
      $(h_1, r_1, \sigma_1) \leftarrow \mathcal{PCHA}.\mathsf{Hash}(mpk, sk_{\bar{w}}, m, \mathbb{A})$
      $(r_1, \sigma_1) \leftarrow \mathcal{PCHA}.\mathsf{Adapt}(mpk, \bar{w}, sk_{\bar{w}}, sk_{\mathcal{S},\bar{w}}, m, m', h_1, r_1, \sigma_1)$
      return (h_b, r_b, σ_b)
  return 1 *if* $b = b'$*, else return* 0

Fig. 3. FIND-security of PCH.

the randomness r and the signature σ do not reveal whether they were obtained through $\mathcal{PCHA}.\mathsf{Hash}$ or $\mathcal{PCHA}.\mathsf{Adapt}$. The messages, signing keys and secret keys are adaptively chosen by the adversary.

*Definition 6 (*FIND*-Security of PCHA): Let the advantage of an adversary* $\mathcal{A}$ *in the* FIND *experiment be*:

$$\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = |\Pr[\mathbf{Exp}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = 1] - 1/2|.$$

We say that $\mathcal{PCHA}$ *is* FIND *secure if* $\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t)$ *is a negligible function in* κ *for all probabilistic polynomial-time adversaries* $\mathcal{A}$*.*

2) Collision-Resistance: Collision-resistance requires that it should be infeasible to find collisions for hashes which were computed concerning policies which are not satisfied by the secret keys. The seminal work of PCH [9] introduces insider collision-resistance and outsider collision-resistance, where insider collision-resistance implies outsider collision-resistance. For simplicity, we propose a security model of collision-resistance to mimic insider collision-resistance. More specifically, we allow the adversary to corrupt transaction owners, partial transaction modifiers, and access to collisions for any mutable transaction. The adversary cannot find any collision for the uncorrupted mutable transactions whose access policies do not match the attributes of the corrupted modifiers.

*Definition 7 (*CR*-Security of PCHA): Let the advantage of an adversary* $\mathcal{A}$ *in the* CR *experiment be*:

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = \Pr[\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = 1].$$

We say that $\mathcal{PCHA}$ *is* CR *secure if* $\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t)$ *is a negligible function in* κ *for all probabilistic polynomial-time adversaries* $\mathcal{A}$*.*

3) Uniqueness: Uniqueness requires that it is hard to find different randomnesses yielding the same hash value for the adversarial-chosen message and public key.

*Definition 8 (*UNI*-Security of PCHA): Let the advantage of an adversary* $\mathcal{A}$ *in the* UNI *experiment be*:

$$\mathbf{Adv}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = \Pr[\mathbf{Exp}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) = 1].$$

We say that $\mathcal{PCHA}$ *is* UNI *secure if* $\mathbf{Adv}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t)$ *is a negligible function in* κ *for all probabilistic polynomial-time adversaries* $\mathcal{A}$*.*

4) Accountability: Accountability requires that a key-like decryption box cannot accuse any identities whose secret keys do not contribute to the key-like decryption box. We allow the adversary to corrupt a part of transaction owners (oracle $\mathcal{O}_{\mathsf{UKeyGen}}$) and transaction modifiers (oracle $\mathcal{O}_{\mathsf{MKeyGen}}$) to gain the corresponding secret keys.

$\mathbf{Exp}^{\mathsf{CR}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)$
$\mathcal{L}_{\mathcal{U}},\mathcal{L}_{\mathcal{S}},\mathcal{L}_{\mathcal{H}}\leftarrow\emptyset, pp\leftarrow\mathcal{PCHA}.\mathsf{ParGen}(1^{\kappa})$
$(mpk,msk)\leftarrow\mathcal{PCHA}.\mathsf{Setup}(n,t)$
$(\bar{w}^*,m^*,r^*,\sigma^*_{\bar{w}},\bar{w}'^*,m'^*,r'^*,\sigma^*_{\bar{w}'},h^*)\leftarrow\mathcal{A}^{\mathcal{O}}(pp,mpk)$
where $\mathcal{O}\leftarrow\{\mathcal{O}_{\mathsf{UKeyGen}}(\cdot),\mathcal{O}_{\mathsf{MKeyGen}}(\cdot,\cdot),\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot),$
$\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)\}$
and $\mathcal{O}_{\mathsf{UKeyGen}}(\cdot)$ *on input* $\bar{w}\in\mathcal{I}$:
$\mathcal{L}_{\mathcal{U}}\leftarrow\mathcal{L}_{\mathcal{U}}\cup\bar{w}$
return $\mathcal{PCHA}.\mathsf{UKeyGen}(msk,\bar{w})$
and $\mathcal{O}_{\mathsf{MKeyGen}}(\cdot,\cdot)$ *on input* $\mathcal{S}\subseteq\Omega$ *and* $\bar{w}\in\mathcal{I}$:
$\mathcal{L}_{\mathcal{S}}\leftarrow\mathcal{L}_{\mathcal{S}}\cup\{\mathcal{S}\}$
return $\mathcal{PCHA}.\mathsf{MKeyGen}(msk,\mathcal{S},\bar{w})$
and $\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot)$ *on input* $\bar{w}\in\mathcal{I}$, $m\in\mathcal{M}$ *and* $\mathbb{A}$:
$sk_{\bar{w}}\leftarrow\mathcal{PCHA}.\mathsf{UKeyGen}(msk,\bar{w})$ if $\bar{w}\notin\mathcal{L}_{\mathcal{U}}$
$(h,r,\sigma_{\bar{w}})\leftarrow\mathcal{PCHA}.\mathsf{Hash}(mpk,sk_{\bar{w}},m,\mathbb{A})$
return $\perp$ *if* $r=\perp$
$\mathcal{L}_{\mathcal{H}}\leftarrow\mathcal{L}_{\mathcal{H}}\cup\{(h,\mathbb{A},m)\}$
return $(h,r,\sigma_{\bar{w}})$
and $\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)\}$ *on input* $\bar{w},sk_{\bar{w}'},sk_{\mathcal{S},\bar{w}'},m,m',h,r,\sigma_{\bar{w}}$
$(r',\sigma_{\bar{w}'})\leftarrow\mathcal{PCHA}.\mathsf{Adapt}(mpk,\bar{w},sk_{\bar{w}'},sk_{\mathcal{S},\bar{w}'},m,m',h,r,\sigma_{\bar{w}})$
return $\perp$ *if* $r'=\perp$
$\mathcal{L}_{\mathcal{H}}\leftarrow\mathcal{L}_{\mathcal{H}}\cup\{(h,\mathbb{A},m')\}$ *if* $(h,\cdot,m)\in\mathcal{L}_{\mathcal{H}}$
return $r',\sigma_{\bar{w}'}$
return 1 *if* $\mathcal{PCHA}.\mathsf{Verify}(mpk,\bar{w}^*,m^*,h^*,r^*,\sigma^*_{\bar{w}})=1\wedge$
$\mathcal{PCHA}.\mathsf{Verify}(mpk,\bar{w}'^*,m'^*,h^*,r'^*,\sigma^*_{\bar{w}'})=1\wedge$
$(h^*,\mathbb{A},\cdot)\in\mathcal{L}_{\mathcal{H}}\wedge m^*\neq m'^*\wedge\mathbb{A}\cap\mathcal{L}_{\mathcal{S}}=\emptyset\wedge(h^*,\cdot,m^*)\notin\mathcal{H}$
return 0

Fig. 4. CR-security of PCH.

$\mathbf{Exp}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)$
$pp\leftarrow\mathcal{PCHA}.\mathsf{ParGen}(1^{\kappa})$
$(mpk,\bar{w}^*,m^*,h^*,r^*,r'^*,\sigma^*_{\bar{w}})\leftarrow\mathcal{A}(pp,n,t)$
return 1 *if* $\mathcal{PCHA}.\mathsf{Verify}(mpk,\bar{w}^*,m^*,h^*,r^*,\sigma^*_{\bar{w}})=1\wedge$
$\mathcal{PCHA}.\mathsf{Verify}(mpk,\bar{w}^*,m^*,h^*,r'^*,\sigma^*_{\bar{w}})=1\wedge r^*\neq r'^*$
return 0

Fig. 5. UNI-security of PCH.

$\mathbf{Exp}^{\mathsf{ACT}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)$
$\mathcal{L}_{\mathcal{U}},\mathcal{L}_{\mathcal{M}}\leftarrow\emptyset$
$pp\leftarrow\mathcal{PCHA}.\mathsf{ParGen}(1^{\kappa})$
$(mpk,msk)\leftarrow\mathcal{PCHA}.\mathsf{Setup}(n,t)$
$\mathcal{D}\leftarrow\mathcal{A}^{\mathcal{O}_{\mathsf{UKeyGen}}(\cdot),\mathcal{O}_{\mathsf{MKeyGen}}(\cdot,\cdot)}(pp,mpk)$
and $\mathcal{O}_{\mathsf{UKeyGen}}(\cdot)$ *on input* $\bar{w}\in\mathcal{I}$:
$\mathcal{L}_{\mathcal{U}}\leftarrow\mathcal{L}_{\mathcal{U}}\cup\{\bar{w}\}$
return $\mathcal{PCHA}.\mathsf{UKeyGen}(msk,\bar{w})$
and $\mathcal{O}_{\mathsf{MKeyGen}}(\cdot,\cdot)$ *on input* $\mathcal{S}\subseteq\Omega$ *and* $\bar{w}\in\mathcal{I}$:
$\mathcal{L}_{\mathcal{M}}\leftarrow\mathcal{L}_{\mathcal{M}}\cup\{\bar{w}\}$
return $\mathcal{PCHA}.\mathsf{MKeyGen}(msk,\mathcal{S},\bar{w})$
return 1 *if* $(S=\emptyset\vee S\not\subseteq\mathcal{L}_{\mathcal{U}}\cup\mathcal{L}_{\mathcal{M}})\wedge|\mathcal{L}_{\mathcal{M}}|\leq t$
where $S\leftarrow\mathcal{PCHA}.\mathsf{Trace}(msk,\mathcal{D})$
return 0

Fig. 6. ACT-security of PCH.

Definition 9 (ACT*-Security of PCHA): Let the advantage of an adversary $\mathcal{A}$ in the* ACT *experiment be*:

$$\mathbf{Adv}^{\mathsf{ACT}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)=\Pr[\mathbf{Exp}^{\mathsf{ACT}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)=1].$$

We say that $\mathcal{PCHA}$ is ACT *secure if $\mathbf{Adv}^{\mathsf{ACT}}_{\mathcal{A},\mathcal{PCHA}}(\kappa,n,t)$ is a negligible function in κ for all probabilistic polynomial-time adversaries $\mathcal{A}$.*

Ingredients:
- $\mathcal{ABTT}=\{\mathsf{ParGen},\mathsf{Setup},\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec},\mathsf{Trace}\}$.
- $\mathcal{CHET}=\{\mathsf{ParGen},\mathsf{KeyGen},\mathsf{Hash},\mathsf{Verify},\mathsf{Adapt}\}$.
- $\mathcal{IBS}=\{\mathsf{ParGen},\mathsf{Setup},\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify}\}$.

$\mathcal{PCHA}.\mathsf{ParGen}(1^{\kappa})\rightarrow pp$: Return $pp\leftarrow(pp_{\mathcal{ABTT}},pp_{\mathcal{CHET}},pp_{\mathcal{IBS}})$, where
$pp_{\mathcal{ABTT}}\leftarrow\mathcal{ABTT}.\mathsf{ParGen}(1^{\kappa})$,
$pp_{\mathcal{CHET}}\leftarrow\mathcal{CHET}.\mathsf{ParGen}(1^{\kappa})$,
$pp_{\mathcal{IBS}}\leftarrow\mathcal{IBS}.\mathsf{ParGen}(1^{\kappa})$.
$\mathcal{PCHA}.\mathsf{Setup}(n,t)\rightarrow(mpk,msk)$: Return $mpk\leftarrow(mpk_{\mathcal{ABTT}},pk_{\mathcal{CHET}},mpk_{\mathcal{IBS}})$ and $msk\leftarrow(msk_{\mathcal{ABTT}},sk_{\mathcal{CHET}},msk_{\mathcal{IBS}})$, where
$(mpk_{\mathcal{ABTT}},msk_{\mathcal{ABTT}})\leftarrow\mathcal{ABTT}.\mathsf{Setup}(n,t)$,
$(sk_{\mathcal{CHET}},pk_{\mathcal{CHET}})\leftarrow\mathcal{CHET}.\mathsf{KeyGen}(pp_{\mathcal{CHET}})$,
$(mpk_{\mathcal{IBS}},msk_{\mathcal{IBS}})\leftarrow\mathcal{IBS}.\mathsf{Setup}(pp_{\mathcal{IBS}})$.
$\mathcal{PCHA}.\mathsf{UKeyGen}(msk,\bar{w})\rightarrow sk_{\bar{w}}$: Return $sk_{\bar{w}}\leftarrow sk^{(\bar{w})}_{\mathcal{IBS}}$, where $sk^{(\bar{w})}_{\mathcal{IBS}}\leftarrow\mathcal{IBS}.\mathsf{KeyGen}(msk,\bar{w})$.
$\mathcal{PCHA}.\mathsf{MKeyGen}(msk,\mathcal{S},\bar{w}')\rightarrow(sk_{\bar{w}'},sk_{\mathcal{S},\bar{w}'})$: Return $sk_{\bar{w}'}\leftarrow sk^{(\bar{w}')}_{\mathcal{IBS}}$ and $sk_{\mathcal{S},\bar{w}'}\leftarrow(sk^{(\mathcal{S},\bar{w}')}_{\mathcal{ABTT}},sk_{\mathcal{CHET}})$, where
$sk^{(\bar{w}')}_{\mathcal{IBS}}\leftarrow\mathcal{IBS}.\mathsf{KeyGen}(msk_{\mathcal{IBS}},\bar{w}')$,
$sk^{(\mathcal{S},\bar{w}')}_{\mathcal{ABTT}}\leftarrow\mathcal{ABTT}.\mathsf{KeyGen}(msk_{\mathcal{ABTT}},\mathcal{S},\bar{w}')$.
$\mathcal{PCHA}.\mathsf{Hash}(mpk,sk_{\bar{w}},m,\mathbb{A})\rightarrow(h,r,\sigma_{\bar{w}})$: Return $h\leftarrow(h_{\mathcal{CHET}},c_{\mathcal{ABTT}})$, $r\leftarrow r_{\mathcal{CHET}}$ and $\sigma_{\bar{w}}\leftarrow\sigma^{(\bar{w})}_{\mathcal{IBS}}$, where
$(h_{\mathcal{CHET}},r_{\mathcal{CHET}},\mathsf{etd}_{\mathcal{CHET}})\leftarrow\mathcal{CHET}.\mathsf{Hash}(pk_{\mathcal{CHET}},m)$,
$c_{\mathcal{ABTT}}\leftarrow\mathcal{ABTT}.\mathsf{Enc}(\mathsf{etd}_{\mathcal{CHET}},\mathbb{A})$,
$\sigma^{(\bar{w})}_{\mathcal{IBS}}\leftarrow\mathcal{IBS}.\mathsf{Sign}(sk^{(\bar{w})}_{\mathcal{IBS}},(m,h_{\mathcal{CHET}},r_{\mathcal{CHET}},c_{\mathcal{ABTT}}))$.
$\mathcal{PCHA}.\mathsf{Verify}(mpk,\bar{w},m,h,r,\sigma_{\bar{w}})\rightarrow\{0,1\}$: Return
$\mathcal{IBS}.\mathsf{Verify}(mpk_{\mathcal{IBS}},\bar{w},(m,h_{\mathcal{CHET}},r_{\mathcal{CHET}},c_{\mathcal{ABTT}}),\sigma^{(\bar{w})}_{\mathcal{IBS}})\wedge$
$\mathcal{CHET}.\mathsf{Verify}(pk_{\mathcal{CHET}},m,h_{\mathcal{CHET}},r_{\mathcal{CHET}})$.
$\mathcal{PCHA}.\mathsf{Adapt}(mpk,\bar{w},sk_{\bar{w}'},sk_{\mathcal{S},\bar{w}'},m,m',h,r,\sigma_{\bar{w}})\rightarrow(r',\sigma_{\bar{w}'})$: Return $\perp$ if $\mathcal{PCHA}.\mathsf{Verify}(mpk_{\mathcal{PCHA}},\bar{w},m,h,r,\sigma_{\bar{w}})=0$; otherwise, return $r'\leftarrow r'_{\mathcal{CHET}}$ and $\sigma_{\bar{w}'}\leftarrow\sigma^{(\bar{w}')}_{\mathcal{IBS}}$, where
$\mathsf{etd}_{\mathcal{CHET}}\leftarrow\mathcal{ABTT}.\mathsf{Dec}(sk^{(\mathcal{S},\bar{w}')}_{\mathcal{ABTT}},c_{\mathcal{ABTT}})$,
$r'_{\mathcal{CHET}}\leftarrow\mathcal{CHET}.\mathsf{Adapt}(sk_{\mathcal{CHET}},\mathsf{etd}_{\mathcal{CHET}},m,m',h_{\mathcal{CHET}},r_{\mathcal{CHET}})$,
$\sigma^{(\bar{w}')}_{\mathcal{IBS}}\leftarrow\mathcal{IBS}.\mathsf{Sign}(sk^{(\bar{w}')}_{\mathcal{IBS}},(m',h_{\mathcal{CHET}},r'_{\mathcal{CHET}},c_{\mathcal{ABTT}}))$.
$\mathcal{PCHA}.\mathsf{Trace}(msk,\mathcal{D})\rightarrow S$: Parse $\mathcal{D}=(\mathcal{D},\mathcal{D}_{\mathcal{S}})$ and $(r',\sigma_{\bar{w}'})\leftarrow\mathcal{D}(m,m',\mathcal{PCHA}.\mathsf{Hash}(mpk,sk_{\bar{w}},m,\mathbb{A}))$ for all $\mathbb{A}$ s.t. $\mathcal{D}_{\mathcal{S}}\in\mathbb{A}$. Derive $\mathcal{D}'$ from $\mathcal{D}$ by changing the I/O stream of $\mathcal{D}$. In particular,
Input: $\mathcal{D}$ *takes* $(m,m',h,r,\sigma_{\bar{w}})$*, where* $h=(h_{\mathcal{CHET}},c_{\mathcal{ABTT}})$,
$\mathcal{D}'$ *takes* $c_{\mathcal{ABTT}}$*, where* $c_{\mathcal{ABTT}}\leftarrow\mathcal{ABTT}.\mathsf{Enc}(r,\mathcal{D}_{\mathbb{A}})$.
Output: $\mathcal{D}$ *outputs* $(r',\sigma_{\bar{w}'})$*, where* (m',h,r') *and* $\sigma_{\bar{w}'}$ *are valid*,
$\mathcal{D}'$ *outputs* r *if* (m',h,r') *and* $\sigma_{\bar{w}'}$ *are valid.*
Then we have $r\leftarrow\mathcal{D}'(\mathcal{ABTT}.\mathsf{Enc}(r,\mathcal{D}_{\mathbb{A}}))$ which is identical to $\mathcal{D}_{\mathcal{ABTT}}$. Return $S\leftarrow S_{\mathcal{ABTT}}\cup S_{\mathcal{IBS}}$, where $S_{\mathcal{ABTT}}\leftarrow\mathcal{ABTT}.\mathsf{Trace}(msk_{\mathcal{ABTT}},\mathcal{D}')$ and $S_{\mathcal{IBS}}$ records the signer of all $\sigma_{\bar{w}'}$ generated during the tracing algorithm.

Fig. 7. Generic construction of PCHA.

C. Generic Construction of PCHA

Our PCHA construction is shown in Fig. 7, which is based on an adaptively secure and fully collision-resistant ABTT, a fully indistinguishable, collision-resistant and uniqueness-driven CHET and an existentially unforgeable IBS. Based on the operations in redactable blockchain, we give a sketch of our generic construction as follows.

1) Initialization: The central authority runs the parameter generation algorithm and setup algorithm to initialize the system parameters. The parameter generation algorithm runs parameter generation algorithms of ABTT, CHET and IBS to generate the public parameters. The setup algorithm runs the setup algorithm of ABTT and IBS as well as the key

Let $\Omega = \{0,1\}^*$ denote the attribute universe, $\mathcal{M} = \{0,1\}^{\ell_m}$ be the message space, and $\mathcal{I} = \{0,1\}^{\ell}$ denote the identity space. The concrete construction of PCHA is presented below:

$\mathcal{PCHA}.\mathsf{ParGen}(1^\kappa) \rightarrow pp$: Run group generator $\mathcal{G}(1^\kappa)$ to obtain $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$. Return $pp = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$.

$\mathcal{PCHA}.\mathsf{Setup}(n,t) \rightarrow (mpk, msk)$:

1) Run $\mathcal{ABTT}.\mathsf{Setup}(n,t)$ to obtain $(mpk_{\mathcal{ABTT}}, msk_{\mathcal{ABTT}})$, where $mpk_{\mathcal{ABTT}}$ includes $\mathcal{H}_1$ and $\mathcal{H}_2$.
2) Pick a symmertic-key encryption scheme $\mathcal{SE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$.
3) Pick prime $e > N'$ with $N' = max_r\{(N, \cdot, \cdot, \cdot, \cdot) \leftarrow \mathsf{RSAKeyGen}(1^\kappa; r)\}$.
4) Run $(N_1, p_1, q_1, \cdot, \cdot) \leftarrow \mathsf{RSAKeyGen}(1^\kappa)$, pick a hash function $\mathcal{H}_4 : \{0,1\}^* \rightarrow \mathbb{Z}^*_{N_1}$.
5) Compute d_1 s.t. $ed_1 \equiv 1 \bmod (p_1 - 1)(q_1 - 1)$. Pick a hash function $\mathcal{H}_6 : \{0,1\}^* \rightarrow \mathbb{Z}_p^2$ and $x \in \mathbb{Z}_p^*$.
6) Return $mpk = (mpk_{\mathcal{ABTT}}, \mathcal{SE}, N_1, e, \mathcal{H}_4, \mathcal{H}_6, h^x)$ and $msk = (msk_{\mathcal{ABTT}}, d_1, x)$.

$\mathcal{PCHA}.\mathsf{UKeyGen}(msk, \bar{w}) \rightarrow sk_{\bar{w}}$: Compute $sk_{\bar{w}} = \mathcal{H}_1(\bar{w})^x$. Return $sk_{\bar{w}}$.

$\mathcal{PCHA}.\mathsf{MKeyGen}(msk, \mathcal{S}, \bar{w}') \rightarrow (sk_{\bar{w}'}, sk_{\mathcal{S},\bar{w}'})$: Run $\mathcal{ABTT}.\mathsf{KeyGen}(msk_{\mathcal{ABTT}}, \mathcal{S}, \bar{w}')$ to obtain $sk^{(\mathcal{S},\bar{w}')}_{\mathcal{ABTT}}$. Compute $sk_{\bar{w}'} = \mathcal{H}_1(\bar{w}')^x$. Return $sk_{\bar{w}'} = \mathcal{H}_1(\bar{w}')^x$ and $sk_{\mathcal{S},\bar{w}'} = sk^{(\mathcal{S},\bar{w}')}_{\mathcal{ABTT}}$.

$\mathcal{PCHA}.\mathsf{Hash}(mpk, sk_{\bar{w}}, m, \mathbb{A}) \rightarrow (h, r, \sigma_{\bar{w}})$:

1) Run $(N_2, p_2, q_2, \cdot, \cdot) \leftarrow \mathsf{RSAKeyGen}(1^\kappa)$. Pick a hash function $\mathcal{H}_5 : \{0,1\}^* \rightarrow \mathbb{Z}^*_{N_2}$.
2) Compute d_2 s.t. $ed_2 \equiv 1 \bmod (p_2 - 1)(q_2 - 1)$.
3) Pick $r_1 \in \mathbb{Z}^*_{N_1}$ and $r_2 \in \mathbb{Z}^*_{N_2}$. Compute $h_1 = \mathcal{H}_4(m, N_1, N_2) r_1^e \bmod N_1$ and $h_2 = \mathcal{H}_5(m, N_1, N_2) r_2^e \bmod N_2$.
4) Pick $\mu \in \{0,1\}^\kappa$. Run $\mathcal{SE}.\mathsf{KeyGen}(1^\kappa)$ to obtain $k_{\mathcal{SE}}$. Run $\mathcal{SE}.\mathsf{Enc}(k_{\mathcal{SE}}, d_2)$ to obtain $c_{\mathcal{SE}}$.
5) Run $\mathsf{encode}(k_{\mathcal{SE}}, \mu)$ to obtain K, where encode is an injectively function that maps $(k_{\mathcal{SE}}, \mu)$ to $\mathcal{M}$.
6) Run $\mathcal{ABTT}.\mathsf{Enc}(K, \mathbb{A})$ with the randomness $(s_1, s_2) \leftarrow \mathcal{H}_6(\mu, \mathbb{A})$ to obtain $c_{\mathcal{ABTT}}$.
7) Pick $g' \in \mathbb{G}$ and $k \in \mathbb{Z}_p$. Compute $R = e(g', h)^k$. Set $v = \mathcal{H}_2(m, h_1, h_2, r_1, r_2, c_{\mathcal{ABTT}}, c_{\mathcal{SE}}, R)$ and $u = (\mathcal{H}_1(\bar{w})^x)^v \cdot (g')^k$.
8) Return $h = (h_1, h_2, N_2, \mathcal{H}_6, c_{\mathcal{ABTT}}, c_{\mathcal{SE}})$, $r = (r_1, r_2)$ and $\sigma_{\bar{w}} = (u, v)$.

$\mathcal{PCHA}.\mathsf{Verify}(mpk, \bar{w}, m, h, r, \sigma_{\bar{w}}) \rightarrow \{0,1\}$: Verify $r_1 \in \mathbb{Z}^*_{N_1}$ and $r_2 \in \mathbb{Z}^*_{N_2}$, $h_1 = \mathcal{H}_4(m, N_1, N_2) r_1^e \bmod N_1$ and $h_2 = \mathcal{H}_5(m, N_1, N_2) r_2^e \bmod N_2$. Verify $v = \mathcal{H}_2(m, h_1, h_2, r_1, r_2, c_{\mathcal{ABTT}}, c_{\mathcal{SE}}, R')$, where $R' = e(u, h) \cdot e(\mathcal{H}_1(\bar{w}), (h^x)^{-1})^v$. Return 1 if all checks hold; otherwise return 0.

$\mathcal{PCHA}.\mathsf{Adapt}(mpk, \bar{w}, sk_{\bar{w}'}, sk_{\mathcal{S},\bar{w}'}, m, m', h, r, \sigma_{\bar{w}}) \rightarrow (r', \sigma_{\bar{w}'})$:

1) Return $\perp$ if $\mathcal{PCHA}.\mathsf{Verify}(mpk, \bar{w}, m, h, r, \sigma_{\bar{w}}) = 0$.
2) Run $\mathcal{ABTT}.\mathsf{Dec}(sk_{\mathcal{S},\bar{w}'}, c_{\mathcal{ABTT}})$ to obtain K', and $\mathsf{encode}^{-1}(K')$ to obtain $(k'_{\mathcal{SE}}, \mu')$.
3) Run $\mathcal{ABTT}.\mathsf{Enc}(K', \mathbb{A})$ with the randomness $(s'_1, s'_2) \leftarrow \mathcal{H}_6(\mu', \mathbb{A})$ to obtain $c'_{\mathcal{ABTT}}$. Return $\perp$ if $c_{\mathcal{ABTT}} \neq c'_{\mathcal{ABTT}}$.
4) Run $\mathcal{SE}.\mathsf{Dec}(k'_{\mathcal{SE}}, c_{\mathcal{SE}})$ to obtain d_2. Return $\perp$ if $d_2 = \perp$.
5) Let $x_1 = \mathcal{H}_4(m, N_1, N_2)$, $x'_1 = \mathcal{H}_4(m', N_1, N_2)$ and $y_1 = x_1 r_1^e \bmod N_1$. Let $x_2 = \mathcal{H}_5(m, N_1, N_2)$, $x'_2 = \mathcal{H}_5(m', N_1, N_2)$ and $y_2 = x_2 r_2^e \bmod N_2$. Compute $r'_1 = (y_1 (x'_1)^{-1})^{d_1} \bmod N_1$ and $r'_2 = (y_2 (x'_2)^{-1})^{d_2} \bmod N_2$.
6) Return $\perp$ if $h_1 \neq \mathcal{H}_4(m', N_1, N_2) r'^e_1 \bmod N_1$ and $h_2 \neq \mathcal{H}_5(m', N_1, N_2) r'^e_2 \bmod N_2$.
7) Pick $g' \in \mathbb{G}$ and $k \in \mathbb{Z}_p$. Compute $R = e(g', h)^k$. Set $v = \mathcal{H}_2(m, h_1, h_2, r_1, r_2, c_{\mathcal{ABTT}}, c_{\mathcal{SE}}, R)$ and $u = (\mathcal{H}_1(\bar{w})^x)^v \cdot (g')^k$.
8) Return $r' = (r'_1, r'_2)$ and $\sigma_{\bar{w}'} = (u, v)$.

$\mathcal{PCHA}.\mathsf{Trace}(msk, \mathcal{D}) \rightarrow S$:

1) Let $\bar{w}^* = w^*_1 \ldots w^*_\ell$ in $\{0,1\}^\ell$. Set $S \leftarrow \emptyset$. Modify $\mathcal{PCHA}.\mathsf{Hash}(sk_{\bar{w}}, m, \mathbb{A})$ to $\mathcal{PCHA}.\mathsf{mHash}(sk_{\bar{w}}, m, \mathbb{A})$. In particular, the nested algorithm $\mathcal{ABTT}.\mathsf{Enc}(m, \mathbb{A})$ is modified to $\mathcal{ABTT}.\mathsf{mEnc}(m, \mathcal{D}_{\mathbb{A}})$ with the different c_{w_ψ} component:
   - For $b = 0$, we keep the original encryption algorithm with the message $K_1 \neq 0$.
   - For $b = 1$, we change the original encryption algorithm by setting the message $K_1 = 0$.
2) For $\psi = 1, \ldots, \ell$: Pick $m, m' \in \mathcal{M}$. Run $\mathcal{PCHA}.\mathsf{UKeyGen}(msk, \bar{w})$ to obtain $sk_{\bar{w}}$ for any $\bar{w} \in \mathcal{I}$. Run $\mathcal{PCHA}.\mathsf{mHash}(sk_{\bar{w}}, m, \mathbb{A})$ (for some $\mathbb{A}$ s.t. $\mathcal{D} \in \mathbb{A}$) to obtain $(h, r, \sigma_{\bar{w}})$. Run $\mathcal{D}(mpk, m, m', h, r, \sigma_{\bar{w}})$ to obtain $(r', \sigma_{\bar{w}'})$. Set $w^*_\psi = 0$ if $\mathcal{PCHA}.\mathsf{Verify}(\bar{w}, m', h, r', \sigma_{\bar{w}'}) = 1$; otherwise $w^*_\psi = 1$. Set $S \leftarrow S \cup \bar{w}$ if $\sigma_{\bar{w}'}$ is valid.
3) Run $\mathcal{FC}.\mathsf{Trace}(tk, \bar{w}^*)$ to obtain $S_{\mathcal{FC}}$. Return $S \leftarrow S \cup S_{\mathcal{FC}}$.

Fig. 8. Concrete construction of PCHA.

generation algorithm of CHET to generate the master public key and the master secret key. To initialize a transaction owner, the central authority runs the user key generation algorithm which runs the key generation of IBS associated with an identity $\bar{w}$ to derive a secret signing key. To initialize a transaction modifier, the central authority runs the modifier key generation algorithm which runs the key generation of IBS associated with an identity $\bar{w}'$ and ABTT associated with a set of attributes $\mathcal{S}$. The transaction modifier will gain a secret signing key of IBS, a secret key of ABTT and a secret key of CHET.

2) Hashing: Given a message m and an access policy $\mathbb{A}$, the data owner computes a CHET according to the message m and encrypts the ephemeral trapdoor under $\mathbb{A}$ via ABTT as well as signing the hash result under the identity of transaction owner using the signing algorithm of IBS to preserve accountability.

3) Collision-Finding: Given a secret key of ABTT whose $\mathcal{S}$ satisfies $\mathbb{A}$, the modifier can reconstruct the ephemeral trapdoor to compute the arbitrary collision.

4) Traitor Tracing: Given a key-like decryption box $\mathcal{D}$, the central authority can trace the malicious modifiers whose keys have contributed to $\mathcal{D}$. In particular, the central authority can observe the outputting signature and derive $\mathcal{D}'$ from $\mathcal{D}$ adapted to trace malicious users.

D. Security Analysis of PCHA

Theorem 3: The proposed PCHA has full indistinguishability if the full indistinguishability of the underlying CHET holds.

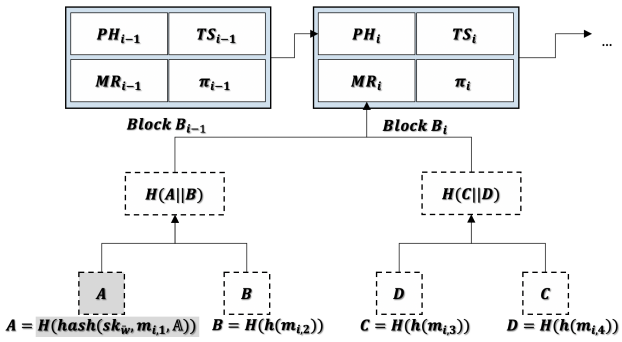


Fig. 9. PCHA-based redactable blockchain.

Theorem 4: The proposed PCHA has collision-resistance if the strongly private collision-resistance of CHET and the IND-CCA *security of ABTT hold.*[2]

Theorem 5: The proposed PCHA has uniqueness if the uniqueness of the underlying CHET holds.

Theorem 6: The proposed PCHA has accountability if the extensional unforgeability of the underlying IBS and collusion-resistant traceability of the underlying ABTT hold.

Please refer to Appendix VIII-D to VIII-G for the detailed proofs.

E. An Instantiation of PCHA

For the sake of readability, we omit the details of ABTT in the instantiation of PCHA, where the instantiation of ABTT is given in Fig. 2. In our concrete construction, we use RSA-based chameleon hash [9] to instantiate CHET and efficient IBS construction [15]. The concrete construction of PCHA is depicted in Fig. 8.

1) PCHA-Based Redactable Blockchain: In Fig. 9, we illustrate a redactable blockchain based on PCHA for transaction-level blockchain rewriting. The blockchain remains intact even if a certain policy-based redactable transaction has been rewritten. In the following, we illustrate the general structure of the PCHA-based redactable blockchain with four transactions. In reality, the number of transactions in each block could be numerous.

The structure of the PCHA-based redactable blockchain is similar to the traditional immutable blockchain except for the hash function used in the transaction. For the i-th block B_i, the block header includes the hash digest of the previous block PH_i, a timestamp TS_i that records the time of the block appended, a Merkle root MR_i that accumulates four transactions in the i-th block and a consensus proof π_i, where π_i could be a nonce at certain blockchain difficulty in the PoW-based blockchain or a signature of qualified stockholders in the PoS-based blockchain.

The PCHA-based redactable blockchain includes two types of transactions: mutable transactions and immutable transactions. For the i-th block B_i, the Merkle root MR_i accumulates four transactions. The transactions $m_{i,2}$, $m_{i,3}$ and $m_{i,4}$ are immutable transactions, where the traditional collision-resistant hash function h is applied. The transaction $m_{i,1}$ is a mutable transaction, where the PCHA is applied associated with an access policy $\mathbb{A}$. When $m_{i,1}$ needs to be rewritten to $m'_{i,1}$, a modifier with a PCHA-based secret key whose attributes $\mathcal{S}$ satisfy $\mathcal{S} \models \mathbb{A}$ can compute the arbitrary collision r' without affecting its hash digest. The transaction modifier then broadcasts $(m'_{i,1}, r')$, and all participants verify the correctness of the new randomness and update their local copy of the blockchain with the new message and randomness pair $(m'_{i,1}, r')$ if they are valid.

[2]ABTT with the IND-CCA security is important since the decryption oracle of ABTT is required to simulate the adapt oracle. In our concrete scheme, as shown in Fig. 8, we follow the construction of the seminal work of PCH [9] via applying Fujisaki-Okamoto transformation [11] to realize ABTT with the IND-CCA security.

VI. Performance Evaluation

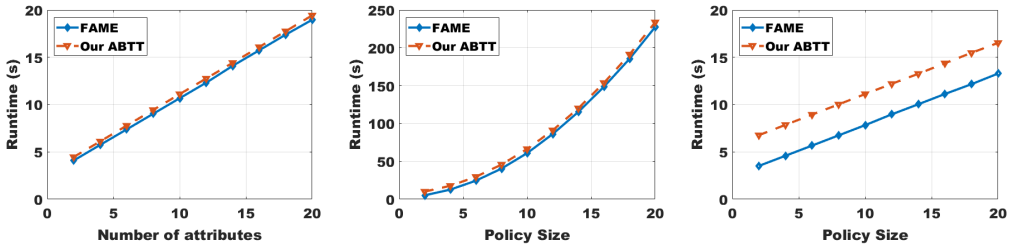
Our implementation was on a PC with Windows 11 x64, 12th Gen Intel(R) Core(TM) i7-12700K 3.61 GHz, RAM 32GB 3200MHz, and performed in Java 8 using the JPBC library. We implemented our ABTT and PCHA to compare FAME [1] and PCH [9], respectively, to quantify the price of adding accountability. Note that our PCHA is the first scheme that offers transaction-level blockchain rewriting with accountability; there is no equivalent scheme to compare with. The previous solutions offer either no accountability [3], [9], [16], [17], [22], [32], [33], [35] or weak accountability [10], [27], [30].

A. Results of FAME and our ABTT

We used the MNT6-992 curve for pairing since it is the best Type-III curve in PBC and offers a 156-bit security level [13]. We set the size of the fingerprint code ℓ to be 16-bit, the number of attributes and the size of the policy from $\{2, 4, \ldots, 20\}$, where the fingerprint code is instantiated by the fully collusion resistant code [7]. Fig. 10 demonstrates that our ABTT only has a modest overhead compared to FAME. As shown in Fig. 10a, our key generation algorithm takes additional $\mathcal{O}(\ell)$ operations to generate 2 additional elements in $\mathbb{G}^2$ to derive a secret key. As shown in Fig. 10b, our encryption algorithm takes additional $\mathcal{O}(\ell)$ operations to generate 6 additional elements in $\mathbb{G}^2 \times \mathbb{H}^2 \times \mathcal{M}^2$ to derive a ciphertext. As shown in Fig. 10c, our decryption algorithm takes the constant overhead to reveal a ciphertext.

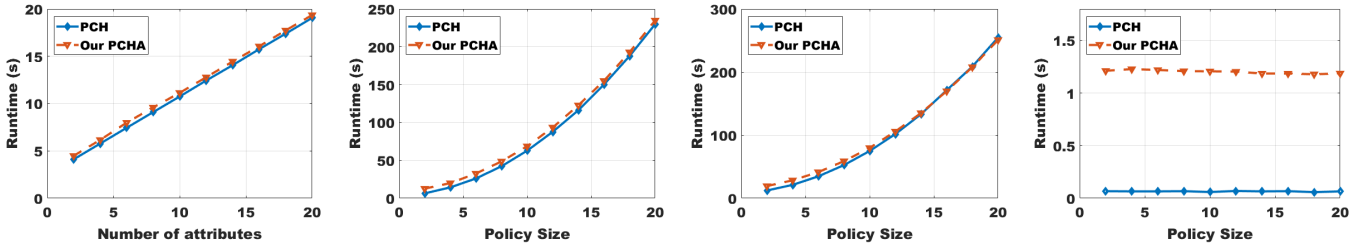
B. Results of PCH and our PCHA

We used the MNT6-992 curve for pairing and the 2048-bit RSA group for the chameleon hash, where the 2048-bit RSA group provides a 112-bit security level [5]. We set the size of the fingerprint code ℓ to be 16-bit, the number of attributes and the size of the policy from $\{2, 4, \ldots, 20\}$. Fig. 11 demonstrates that our PCHA only has a modest overhead compared to non-accountability PCH. As in Fig. 11a, 11b and 11c, the trends of modifier key generation, hash and adaption are very similar to key generation, encryption, and decryption in Fig. 10a, 10b and 10c since PCH and PCHA are built from FAME and ABTT, respectively. Besides, PCH and PCHA take more running time than FAME and ABTT since the RSA-based chameleon hash is used. Fig. 11d illustrates that PCHA takes



(a) Key Generation (b) Data Encryption (c) Data Decryption

Fig. 10. Performances of FAME and our ABTT.



(a) Key Generation (b) Hash (c) Adaption (d) Verification

Fig. 11. Performances of PCH and our PCHA.

around 120ms to process verification compared to 70ms in PCH since PCHA needs to run the IBS to verify the linkability of the modified transaction and its modifier.

VII. Related Work

Blockchain rewriting yields many interesting works [3], [9], [10], [16], [17], [22], [27], [30], [32], [33], [35].

The seminal work of redactable blockchain was introduced by Ateniese et al. [3]. They proposed a redactable blockchain in a permissioned setting that requires a central authority to grant rewriting privileges via secret keys to a particular party, called modifier. The select modifier can come together and alert contents from the blockchain using a large scale MPC protocol. It offers a block-level rewriting controlled at a coarse-grained level via an enhanced collision-resistance chameleon hash [18] and public key infrastructure (PKI). The enhanced collision-resistance chameleon is used to change the hash link between the block header and PKI seals the chameleon hash trapdoor.

To manage rewriting privileges flexibly, Derler et al. [9] proposed a redactable blockchain in a permissioned setting with transaction-level rewriting controlled at a fine-grained level. Compared to [3], here chameleon hashes are used to hash the transaction in computing the Merkle root. They introduced a cryptographic notion, called policy-based chameleon hash (PCH), to formalize their solution. PCH is derived from chameleon hashes with ephemeral trapdoors (CHET) [8] and attribute-based encryption (ABE) [1]. CHET includes a long-term trapdoor and an ephemeral trapdoor. The long-term trapdoor is generated by the central authority and will be issued to the modifier and the ephemeral trapdoor is derived from the transaction owner for each transaction. The ephemeral trapdoor is sealed by the ABE scheme. The modifier has both trapdoors that can alert the transaction and the transaction modifier cannot process rewriting since unknown to the long-term trapdoor.

To eliminate the trusted central authority, Deuber et al. [10] proposed a block-level redactable blockchain in a permissionless setting via consensus-based voting. The block header is modified to have two hash links. Once a modification proposed aggregates enough votes, the modification will be excited by breaking one of the links while the other holds. This gives weak accountability since the identifier of the modifier can be traced from the proposed pool. However, this redactable blockchain inherits the vulnerability of consensus-based voting that suffers from bribing and selfish mining attacks.

After that, a variety of redactable blockchains are proposed with essential properties: accountability [30], self-management [17], revocability [27], [33], decentralization [22], [35] and functionality [16], [32].

Tian et al. [30] first considered accountability in PCH-based redactable blockchain. However, their solution only supports weak accountability that can link the modified transaction to its modifier but the key leakage.

Jia et al. [17] introduced the concept of self-management in redactable blockchain, where the transaction owner can modifier his/her transaction and this ability can be revoked by the transaction modifier. However, they assume the modifier is a semi-trusted party and the accountability has not been taken into consideration.

Panwar et al. [27] introduced a permissioned redactable blockchain with traceability and revocability by applying dynamic group signature schemes (DGSS) and revocable FAME (RFAME). The traceability links the modified transaction to its modifier and the revocability can be used to revoke the rewriting privilege of the malicious modifier. However, the traceability only offers weak accountability and the revocation mechanism takes linear complexity with the size

of non-revoked modifiers and the updated secret key must be granted via secure channels.

To manage modifier revocation efficiently, Xu et al. [33] introduced a novel cryptographic notion, called revocable policy-based chameleon hash (RPCH), as an extension of PCH. RPCH is built from a new RFAME and CHET. Unlike [27], here RFAME only carries out logarithmic complicity and the updated secret key is transferred via public channels. Compared to the FAME [1], the price of adding revocability is almost negligible.

To realize the rewriting authorization in the decentralizing setting, Zhang et al. [35] introduced a multi-authority policy-based chameleon hash (MPCH), and Ma et al. [22] presented a decentralized policy-based chameleon hash (DPCH). The idea behind MPCH and DPCH is similar, where CHET is used to manage the data rewriting and multi-authority attribute-based encryption (MA-ABE) is leveraged to handle the rewriting privilege.

To control the rewriting privilege, Xu et al. [32] introduced a novel redactable blockchain, called k-time modifiable and epoch-based redactable blockchain (KERB), inspired by the double-authentication preventing signature (DAPS). In KERB, the timeline of the system is partitioned into epochs and each transaction modifier must make a time-locked deposit, where the deposit can be drawn after the rewriting privilege is invalid. In each epoch, the modifier can operate the transaction modification k times at most. If the modifier processes modification operations more than k times, the secret key of the modifier can be extracted and the time-locked deposit is lost.

To support the redaction of additional data and unexpended transaction output (UTXO) simultaneously, Hou et al. [16] introduced a new design of redactable blockchain. The proposed solution is based on the PCH and sanitizable signature to support blockchain rewriting, and designs a multi-round protocol to ensure that the UTXO can be safely modified.

VIII. Conclusion

In this paper, we revisited redactable blockchains and discovered that current solutions cannot realize accountability properly. To address this problem, we introduced a useful primitive, called policy-based chameleon hash with black-box accountability (PCHA), for redactable blockchain with accountability. We provided generic construction with formal security proofs and gave a practical instantiation. The experiment demonstrates that the price of accountability is modest compared to the non-accountable solutions.

Appendix

A. Modified q-BDHE Assumption

The modified q-Bilinear Diffie-Hellman Exponent (q-BDHE) assumption was introduced in [14]. The challenger chooses a group $\mathcal{G}$ based on the security parameter λ, randomly picks $a \in \mathbb{Z}_p$ and returns the following terms:

$$g, g^{(a)}, g^{(a^2)}, \ldots, g^{a^q}, g^{(a^{2q+2})}, g^{(a^{2q+3})}, \ldots, g^{(a^{3q+1})} \in \mathbb{G}.$$

Any probabilistic polynomial-time algorithm cannot output $e(g,g)^{a^{2q+1}}$ with more than negligible advantage.

B. Security Proof of Theorem 1

Proof: Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ who can break the IND-CPA security of our ABTT with non-negligible advantage. We can construct a probabilistic polynomial-time simulator $\mathcal{B}$ to break the IND-CPA security of FAME [1] executed by $\mathcal{C}$.

1) Setup: $\mathcal{B}$ runs $\mathcal{FC}.\mathsf{Gen}(1^\kappa, n)$ to obtain (Γ, tk), where Γ contains n words in $\{0,1\}^\ell$ for some $\ell > 0$. $\mathcal{B}$ sends κ to $\mathcal{C}$ and receives $pp_{\mathcal{C}} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h, H_1, H_2, T_1, T_2, \mathcal{H}_1, \mathcal{H}_3)$. $\mathcal{B}$ picks $\alpha \in \mathbb{Z}_p^*$ and a hash function $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_p$. $\mathcal{B}$ the computes $g_1 = g^\alpha$ and $h_i = h^{(\alpha^i)}$ for all $i \in \{1, \ldots, \ell\}$. $\mathcal{B}$ returns $pp = (\Gamma, p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h, g_1, h_1, \ldots, h_\ell, H_1, H_2, T_1, T_2, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$ to $\mathcal{A}$.

2) Phase 1 & 2: $\mathcal{A}$ queries the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$ on a set of attributes $\mathcal{S} \subseteq \Omega$ and an identity $\bar{w} \in \mathcal{I}$. $\mathcal{B}$ forwards $\mathcal{S} \subseteq \Omega$ to $\mathcal{C}$ and receives $sk_{\mathcal{C}} = (sk_0, \{sk_y\}_{y\in\mathcal{S}}, sk')$. Let $\mathcal{W}_b \subseteq \{1, \ldots, \ell\}$ be the set of all i for which $w_i = b$. For $b = 0, 1$, $\mathcal{B}$ computes $sk_{\mathcal{W}_b} = g^{\sum_{i\in\mathcal{W}_b} \frac{1}{\alpha - \mathcal{H}_2(ib)}}$. $\mathcal{B}$ returns $sk_{\mathcal{S},\bar{w}} = (sk_0, \{sk_y\}_{y\in\mathcal{S}}, sk', sk_{\bar{w}})$ to $\mathcal{A}$, where $sk_{\bar{w}} = (sk_{\mathcal{W}_0}, sk_{\mathcal{W}_1})$.

3) Challenge: $\mathcal{A}$ submits two messages (m_0, m_1) of the same size, and an access policy $\mathbb{A}^*$. $\mathcal{B}$ picks $m' \in \mathcal{M}$. $\mathcal{B}$ sends $(m_0 \oplus m', m_1 \oplus m')$ and $\mathbb{A}^*$ to $\mathcal{C}$, and receives $c^*_{\mathcal{C}} = (c_0, c_1, \ldots, c_{n_1}, c')$. $\mathcal{B}$ picks $\psi \in \{1, \ldots, \ell\}$. For $b = \{0, 1\}$, $\mathcal{B}$ picks $r \in \mathbb{Z}_p$ and computes

$$c_{w_\psi, b} = ((h^{\sum_{i=1}^{\ell}(\alpha - \mathcal{H}_2(ib))})^r, (g^{\alpha - \mathcal{H}_2(\psi b)})^r, \times m' \oplus \mathcal{H}_3(e(g, h^{\sum_{i=1, i\neq\psi}^{\ell}(\alpha - \mathcal{H}_2(ib))})^r).$$

$\mathcal{B}$ returns $c^* = (\psi, c_0, c_1, \ldots, c_{n_1}, c', c_{w_\psi})$ to $\mathcal{A}$, where $c_{w_\psi} = (c_{w_\psi,0}, c_{w_\psi,1})$.

4) Guess: $\mathcal{B}$ receives a guess b' from $\mathcal{A}$ and sends b' to $\mathcal{C}$.

The challenge ciphertext c^* is derived from $c^*_{\mathcal{C}}$ and c_{w_ψ}. $c^*_{\mathcal{C}}$ is generated by $\mathcal{C}$ who encrypts one of $m_0 \oplus m'$ and $m_1 \oplus m'$ under the access policy $\mathbb{A}^*$. c_{w_ψ} is computed by $\mathcal{B}$ and seals the message m'. Because $\mathcal{A}$ can distinguish the message used in c^*, $\mathcal{B}$ then uses the returning value of $\mathcal{A}$ to break $\mathcal{C}$. There is no abort during the security reduction. Hence, we have $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathcal{ABTT}}(1^\kappa) = \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{C},\mathcal{FAME}}(1^\kappa)$. Based on the Theorem 4.1 in [1], the advantage of our ABTT scheme to break the DLIN assumption is

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathcal{ABTT}}(1^\kappa, n) \leq (8Q+2)\mathbf{Adv}^{\mathsf{DLIN}}_{\mathcal{C}}(1^\kappa) + (16Q+6)/p,$$

where Q is the number of key generation query. □

C. Security Proof of Theorem 2

Proof: Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ who can break the collusion-resistant traceability of our ABTT with a non-negligible advantage. We can construct a probabilistic polynomial-time simulator $\mathcal{B}$ to break the modified q-BDHE assumption and the collision-resistance of the fingerprint code executed by $\mathcal{C}$.

1) Setup: $\mathcal{B}$ runs $\mathcal{C}$ and receives a set of corrupted $W = \{\bar{w}^{(i)}\}_{i\in C}$ in the code Γ, where $C \subseteq \{1,\ldots,n\}$. $\mathcal{B}$ has the given elements of the modified q-BDHE assumption as

$$(g, h, g^{(\alpha^1)} \ldots, g^{(\alpha^q)}, g^{(\alpha^{q+2})}, \ldots g^{(\alpha^{3q+1})}, \times h^{(\alpha^1)} \ldots, h^{(\alpha^q)}, h^{(\alpha^{q+2})}, \ldots h^{(\alpha^{3q+1})})$$

for $q = \{q_{\mathcal{H}_2}, n\}_{max}$ with the group description $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h)$, where $q_{\mathcal{H}_2}$ is the number of hash queries to $\mathcal{H}_2$. $\mathcal{B}$ picks $I_1,\ldots,I_{q_{\mathcal{H}_2}}, a \in \mathbb{Z}_p$ and $i \in \{1,\ldots,q_{\mathcal{H}_2}\}$. Let $f(x) \in \mathbb{Z}_p[x]$ be a $(q_{\mathcal{H}_2}-1)$-degree polynomial function as

$$F(x) = a\prod_{i=1,i\neq i^*}^{q_{\mathcal{H}_2}}(x - I_i) = F_{q_{\mathcal{H}_2}-1}x^{q_{\mathcal{H}_2}-1} + \cdots + F_1x^1 + F_0.$$

$\mathcal{B}$ sets $h_i = h^{(\alpha^i)}$ for all $i = 1,\ldots,n$, and $g = g^{F(\alpha)}$, $g_1 = g^{\alpha F(\alpha)}$ from $F(x)$. $\mathcal{B}$ picks $a_1, a_2, b_1, b_2 \in \mathbb{Z}_p^*$ and $d_1, d_2, d_3 \in \mathbb{Z}_p$, and sets $H_1 = h^{a_1}, H_2 = g^{a_2}$ and $T_1 = e(g,h)^{d_1a_1+d_3}, T_2 = e(g,h)^{d_2a_2+d_3}$. $\mathcal{B}$ returns $pp = (\Gamma, p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h, g_1, h_1, \ldots, h_\ell, H_1, H_2, T_1, T_2)$ and allows $\mathcal{A}$ to access the following three hash oracles at any time.

- $\mathcal{H}_1(\cdot)$: For a querying message $x_{\mathcal{H}_1} \in \{0,1\}^*$, $\mathcal{B}$ maintains a list $\mathcal{L}_{\mathcal{H}_1}$ and responds as follows. If there has been already a tuple $(x_{\mathcal{H}_1}, V)$ in the list $\mathcal{L}_{\mathcal{H}_1}$, $\mathcal{B}$ responds with V; otherwise, let $x_{\mathcal{H}_1}$ be the i-th distinct query, $\mathcal{B}$ returns a different $V \in \mathbb{G}$ to $\mathcal{A}$, and updates $\mathcal{L}_{\mathcal{H}_1} \leftarrow \mathcal{L}_{\mathcal{H}_1} \cup (x_{\mathcal{H}_1}, V)$.
- $\mathcal{H}_2(\cdot)$: For a querying message $x_{\mathcal{H}_2} \in \{0,1\}^*$, $\mathcal{B}$ maintains a list $\mathcal{L}_{\mathcal{H}_2}$ and responds as follows. If there has been already a tuple $(x_{\mathcal{H}_2}, I)$ in the list $\mathcal{L}_{\mathcal{H}_2}$, $\mathcal{B}$ responds with I; otherwise, let $x_{\mathcal{H}_2}$ be the i-th distinct query, $\mathcal{B}$ returns I_i to $\mathcal{A}$, and updates $\mathcal{L}_{\mathcal{H}_2} \leftarrow \mathcal{L}_{\mathcal{H}_2} \cup (x_{\mathcal{H}_2}, I_i)$.
- $\mathcal{H}_3(\cdot)$: For a querying message $x_{\mathcal{H}_3} \in \mathbb{G}_T$, $\mathcal{B}$ maintains a list $\mathcal{L}_{\mathcal{H}_3}$ and responds as follows. If there has been already a tuple $(x_{\mathcal{H}_3}, Y)$ in the list $\mathcal{L}_{\mathcal{H}_3}$, $\mathcal{B}$ responds with Y; otherwise, let $x_{\mathcal{H}_3}$ be the i-th distinct query, $\mathcal{B}$ returns a different Y in the domain $\{0,1\}^{\ell_m}$ to $\mathcal{A}$, and updates $\mathcal{L}_{\mathcal{H}_3} \leftarrow \mathcal{L}_{\mathcal{H}_3} \cup (x_{\mathcal{H}_3}, Y)$.

2) Query: $\mathcal{A}$ queries the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$ on a set of attributes $\mathcal{S} \subseteq \Omega$ and an identity $\bar{w} \in \mathcal{I}$. Let $\bar{w}^{(i)}$ be the i-th element in W and $\bar{w}_j^{(i)}$ be the j-th bit in $\bar{w}^{(i)}$. For the i-th distinct query and all $j = 1,\ldots,\ell$, $\mathcal{B}$ runs $\mathcal{H}_2(j\bar{w}_j^{(i)})$ to obtain I_j. If $I_j = I_{i^*}$ is returned by $\mathcal{H}_2$, $\mathcal{B}$ aborts; otherwise, $\mathcal{B}$ responds as follows. For $b = 0, 1$, let $\mathcal{W}_b \subseteq \{1,\ldots,\ell\}$ be the set of all j for which $\bar{w}_j^{(i)} = b$, let $F_{\mathcal{W}_b}(x) = F(x)\cdot\sum_{j\in\mathcal{W}_b}\frac{1}{x-\mathcal{H}_2(jb)}$ be a $(q_{\mathcal{H}_2}-2)$-degree at most polynomial function. $\mathcal{B}$ computes $sk_{\mathcal{W}_b} = g^{\sum_{j\in\mathcal{W}_b}\frac{1}{\alpha-\mathcal{H}_2(jb)}} = g^{F_{\mathcal{W}_b}(\alpha)}$ and sets $sk_{\bar{w}} = (sk_{\mathcal{W}_0}, sk_{\mathcal{W}_1})$. $\mathcal{B}$ picks $r_1, r_2 \in \mathbb{Z}_p$ and sets $sk_0 = (h^{b_1r_1}, h^{b_2r_2}, h^{r_1+r_2})$. For all $y \in \mathcal{S}$ and $z = 1, 2$, $\mathcal{B}$ picks $\sigma_b \in \mathbb{Z}_p$, computes

$$sk_{y,z} = \mathcal{H}_1(y1z)^{\frac{b_1r_1}{a_z}}\cdot\mathcal{H}_1(y2z)^{\frac{b_2r_2}{a_z}} \cdot\mathcal{H}_1(y3z)^{\frac{r_1+r_2}{a_z}}\cdot g^{\frac{\sigma_y}{a_z}}$$

and sets $sk_y = (sk_{y,1}, sk_{y,2}, g^{-\sigma_y})$. For $z = 1, 2$, $\mathcal{B}$ picks $\sigma' \in \mathbb{Z}_p$, computes

$$sk'_z = g^{d_z}\cdot\mathcal{H}_1(011z)^{\frac{b_1r_1}{a_z}}\cdot\mathcal{H}_1(012z)^{\frac{b_2r_2}{a_z}}\cdot\mathcal{H}_1(013z)^{\frac{r_1+r_2}{a_z}}\cdot g^{\frac{\sigma'}{a_z}}$$

and sets $sk' = (sk'_1, sk'_2, g^{d_3}\cdot g^{-\sigma'})$. $\mathcal{B}$ returns $sk_{\mathcal{S},\bar{w}} = (sk_0, \{sk_y\}_{y\in\mathcal{S}}, sk', sk_{\bar{w}})$ to $\mathcal{A}$.

3) Output: $\mathcal{A}$ outputs a key-like decryption box $\mathcal{D}$ and an access policy $\mathcal{D}_{\mathbb{A}}$.

$\mathcal{B}$ finds the element $e(g,h)^{\alpha^{q+1}}$ for responding the modified q-BDHE problem. Let

$$F'(x,b) = \frac{\prod_{i=1}^{n}(x - \mathcal{H}_2(ib))}{x - I^*}$$

be a $(n-1)$-degree polynomial function. $\mathcal{B}$ picks $r' \in \mathbb{Z}_p$ and $\psi \in \{1,\ldots,n\}$, and computes

$$c_{w_\psi,b} = (h^{r'(\alpha^{2q+2}-I^{*2q+2})F'(\alpha)}, g^{r'(\alpha^{2q+2}-I^{*2q+2})F(\alpha)}, c_{\psi,b,3})$$

for $c_{\psi,b,3} \in \{0,1\}^{\ell_m}$. Let $c_{w_\psi} = (c_{w_\psi,0}, c_{w_\psi,1})$. $\mathcal{B}$ picks $m \in \{0,1\}^{\ell_m}$ and $s_1, s_2 \in \mathbb{Z}_p$, and computes $c_0 = (H_1^{s_1}, H_2^{s_2}, h^{s_1+s_2})$. For $i = 1,\ldots,n_1$ and $u = 1, 2, 3$, $\mathcal{B}$ computes

$$c_{i,u} = \mathcal{H}_1(\pi(i)u1)^{s_1}\cdot\mathcal{H}_1(\pi(i)u2)^{s_2} \cdot\prod_{j=1}^{n_2}\left[\mathcal{H}_1(0ju1)^{s_1}\cdot\mathcal{H}_1(0ju2)^{s_2}\right]^{\mathbb{M}_{i,j}}$$

and sets $c_i = (c_{i,1}, c_{i,2}, c_{i,3})$, where $\mathcal{D}_{\mathbb{A}} = (\mathbb{M}, \pi)$. $\mathcal{B}$ computes $c' = m_1^* \oplus \mathcal{H}_3(T_1^{s_1}\cdot T_2^{s_2})$ for a random message $m_1^* \in \{0,1\}^{\ell_m}$ and inputs $c = (\psi, c_0, c_1, \ldots, c_{n_1}, c', c_{w_\psi})$ to $\mathcal{D}$. $\mathcal{B}$ receives a message m^* from $\mathcal{D}$. Let $F''(x)$ be a $(2q + n + q_{\mathcal{H}_1} - 1)$-degree polynomial function

$$F''(x) = r'\cdot\frac{x^{2q+2} - I^{*2q+2}}{x - I^*}\cdot F'(x)\cdot F(x)$$

and F''_i be the coefficient of x^i in $F''(x)$ such that

$$e(g, h^{\frac{\sum_{i=1}^{n}(\alpha-\mathcal{H}_2(ib))}{\alpha-\mathcal{H}_2(\psi b)}})^r = e(g,h)^{F''_i\cdot\alpha^i}.$$

It is easy to verify that F''_{2q+1} is equal to $r'F'(I^*)F(I^*)$ which is nonzero, and that $e(g,h)^{F''_i\cdot\alpha^i}$ for all $i \neq 2q+1$ are commutable from the input of modified q-BDHE problem. $\mathcal{B}$ picks a random tuple $(x_{\mathcal{H}_3}, Y)$ from the list $\mathcal{L}_{\mathcal{H}_3}$ and computes

$$(x_{\mathcal{H}_3}\cdot\prod_{i=1,i\neq 2q+1}^{2q+n+q_{\mathcal{H}_2}-1} e(g,h)^{-F''_i\cdot\alpha^i})^{\frac{1}{r'F'(I^*)F(I^*)}} = e(g,h)^{2q+1}$$

as the solution to the modified q-BDHE assumption.

$\mathcal{B}$ breaks the collision-resistance of the fingerprint code by deriving a word $\bar{w}^* = w_1^*\ldots w_\ell^*$ in $\{0,1\}^\ell$. For all $\psi \in \{1,\ldots,n\}$, $\mathcal{B}$ modifies the encryption algorithm with the different c_{w_ψ} component:

- If $\mathcal{H}_2(\psi b) = I^*$ for $b \in \{0,1\}$, $\mathcal{B}$ generates a ciphertext based on m^* and m_1^* in our previous simulation for breaking the modified q-BDHE assumption. For b such as $\mathcal{H}(\psi b) = I^*$, it keeps the original encryption algorithm with the message m_1^*. For $(1-b)$ such as $\mathcal{H}(\psi(1-b)) \neq I^*$, it changes the original encryption algorithm by setting

the message $m_1 = 0$. $\mathcal{B}$ encrypts a random message $m \in \{0,1\}^{\ell_m}$ and inputs m to $\mathcal{D}$. After receiving m' from $\mathcal{D}$, $\mathcal{B}$ sets $w^*_\psi = b$ if $m^* = m'$; otherwise, $w^*_\psi = 1 - b$.

- Otherwise, for $b = 0$, it keeps the original encryption algorithm with the message $m_1 \neq 0$, and for $b = 1$, it changes the original encryption algorithm by setting the message $m_1 = 0$. $\mathcal{B}$ encrypts a random message $m \in \{0,1\}^{\ell_m}$ and inputs m to $\mathcal{D}$. After receiving m' from $\mathcal{D}$, $\mathcal{B}$ sets $w^*_\psi = 0$ if $m = m'$; otherwise, $w^*_\psi = 1$.

At the end, $\mathcal{B}$ returns $\bar{w}^*$ to $\mathcal{C}$ as the word. $\mathcal{C}$ cannot find the correct identities whose secret keys contribute to $\mathcal{D}$.

In the above security simulation, we can construct $\mathcal{B}$ to break the modified q-BDHE assumption and the collision-resistance of the fingerprint code. There is an abort during the simulation. If $\mathcal{A}$ queries the secret key related to the i^*-th $\mathcal{H}_2(\cdot)$ in the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}(\cdot,\cdot)$, $\mathcal{B}$ aborts. In other words, the abort happens if $\mathcal{A}$ guesses i^* incorrectly with $1/q_{\mathcal{H}_2}q_{\mathcal{H}_3}$. Hence, the advantage of our ABTT scheme to break the modified q-BDHE assumption and the collision-resistance of the fingerprint code is

$$\mathbf{Adv}^{\mathsf{CRT}}_{\mathcal{A},\mathcal{ABTT}}(1^\kappa, n) \leq q_{\mathcal{H}_2}q_{\mathcal{H}_3}(\mathbf{Adv}^{\mathsf{q\text{-}BDHE}}_{\mathcal{B}}(1^\kappa) + \mathbf{Adv}^{\mathsf{CR}}_{\mathcal{C},\mathcal{FC}}(1^\kappa, n)).$$

□

D. Security Proof of Theorem 3

Proof: Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ who can break the FIND security of our PCHA with non-negligible advantage. We can construct a probabilistic polynomial-time simulator $\mathcal{B}$ to break the FIND security of the underlying CHET executed by $\mathcal{C}$.

1) Setup: $\mathcal{B}$ sends κ to $\mathcal{C}$ and receives the public parameter $pp_{\mathcal{CHET}}$. $\mathcal{C}$ then runs $\mathcal{ABTT}.\mathsf{ParGen}(1^\kappa)$ and $\mathcal{IBS}.\mathsf{ParGen}(1^\kappa)$ to obtain $pp_{\mathcal{ABTT}}$ and $pp_{\mathcal{IBS}}$. $\mathcal{B}$ sends $pp = (pp_{\mathcal{ABTT}}, pp_{\mathcal{CHET}}, pp_{\mathcal{IBS}})$ to $\mathcal{A}$.

2) Query: $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{HashOrAdapt}}(\cdot,\cdot,\cdot,\cdot,\cdot)$ on a secret key $sk_{\bar{w}}$, a secret key $sk_{\mathcal{S},\bar{w}}$, a message m, a message m' and an access policy $\mathbb{A}$, where $sk_{\mathcal{S},\bar{w}} = (sk^{\mathcal{S},\bar{w}}_{\mathcal{ABTT}}, sk_{\mathcal{CHET}})$. $\mathcal{B}$ forwards $(sk_{\mathcal{CHET}}, m, m')$ to $\mathcal{C}$ and receives $(h_{\mathcal{CHET}}, r_{\mathcal{CHET}}, \mathsf{etd}_{\mathcal{CHET}})$. $\mathcal{B}$ then runs $\mathcal{ABTT}.\mathsf{Enc}(\mathsf{etd}_{\mathcal{CHET}}, \mathbb{A})$ to obtain $c_{\mathcal{ABTT}}$ and $\mathcal{IBS}.\mathsf{Sign}(sk^{(\bar{w})}_{\mathcal{IBS}}, (m, h_{\mathcal{CHET}}, r_{\mathcal{CHET}}, c_{\mathcal{ABTT}}))$ to obtain $\sigma^{(\bar{w})}_{\mathcal{IBS}}$. $\mathcal{B}$ returns $h = (h_{\mathcal{CHET}}, c_{\mathcal{ABTT}}), r = r_{\mathcal{CHET}}$, and $\sigma = \sigma^{(\bar{w})}_{\mathcal{IBS}}$ to $\mathcal{A}$.

3) Output: $\mathcal{A}$ outputs a bit b' and $\mathcal{B}$ forwards b' to $\mathcal{C}$.

The simulation is perfect from $\mathcal{A}$'s point of view. The advantage of $\mathcal{A}$ breaks FIND security is

$$\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) \leq \mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{CHET}}(\kappa).$$

where $\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{CHET}}(\kappa)$ is the advantage of breaking the full indistinguishability of CHET [28].

□

E. Security Proof of Theorem 4

Proof: The security proof of Theorem 4 is based on a sequence of games.

$Game_0$: The original collision-resistance game.

$Game_1$: As $Game_0$, but we abort, if $\mathcal{A}$ makes a query $(\bar{w}, sk_{\bar{w}'}, sk_{\mathcal{S},\bar{w}'}, m, m', h, r, \sigma_{\bar{w}})$, for which h verifies, to the adaptive oracle, for a h returned by the hashing oracle, but m has never been input to the hashing oracle or the adaption oracle, and $\mathcal{A}$ does not have enough attributes to find a collision all by itself. Let this event be E_1.

Assume that event E_1 happens with non-negligible probability. We can construct a simulator $\mathcal{B}$ which breaks the strongly private collision-resistance of the underlying CHET. $\mathcal{B}$ works as follows. Let Q be an upper bound on the queries to the hashing oracle. $\mathcal{B}$ then makes a guess $i \in \{1, 2, \ldots, Q\}$. All queries but the i-th one, are answered as in the prior game. On the i-th query, however, $\mathcal{B}$ encrypts 0 instead of the real etd. Thus, we have at most lost a factor of Q. If $\mathcal{B}$ guessed right, but $\mathcal{A}$ behaves noticeably different now. $\mathcal{B}$ can use $\mathcal{A}$ to break the IND-CCA of the underlying ABTT.

In the case that etd is no longer given to $\mathcal{A}$, which also means that $\mathcal{A}$ can find a collision, without etd. Thus, $\mathcal{B}$ can use $\mathcal{A}$ to break the strongly private collision-resistance of CHET.

Hence, if $\mathcal{B}$ guessed i right, $\mathcal{B}$ can use $\mathcal{A}$ to break IND-CCA of the underlying ABTT and the strongly private collision-resistance of CHET.

$$\left|\Pr[S_0] - \Pr[S_1]\right| \leq qQ(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t) + \mathbf{Adv}^{\mathsf{SPrivColl}}_{\mathcal{A},\mathcal{CHET}}(\kappa)),$$

where $\mathbf{Adv}^{\mathsf{SPrivColl}}_{\mathcal{A},\mathcal{CHET}}(\kappa)$ is the advantage of breaking strongly private collision-resistance of CHET [28].

$Game_2$: As $Game_1$, but we abort, if the adversary outputs $(\bar{w}^*, m^*, r^*, \sigma^*_{\bar{w}}, \bar{w}'^*, m'^*, r'^*, \sigma^*_{\bar{w}'}, h^*)$, such that the winning conditions are fulfilled. Let this event be E_2.

Assume that event E_2 happens with non-negligible probability. We can construct $\mathcal{B}$ to break the strongly private collision-resistance of CHET.

Let Q be an upper bound on the queries to hashing oracle. $\mathcal{B}$ then makes a guess $i \in \{1, 2, \ldots, Q\}$. All queries but the i-th one, are answered as in the prior game. On the i-th query, however, $\mathcal{B}$ encrypts $0^{|\mathsf{etd}|}$ instead of the real etd. Thus, we have at most lost a factor of Q. If $\mathcal{B}$ guessed right, but $\mathcal{A}$ behaves noticeably different now. $\mathcal{B}$ can use $\mathcal{A}$ to break the IND-CCA of the underlying ABTT.

In the case that etd is no longer given to $\mathcal{A}$, which also means that $\mathcal{A}$ can find a collision, without etd. Thus, $\mathcal{B}$ can use $\mathcal{A}$ to break the strongly private collision-resistance of CHET.

Hence, if $\mathcal{B}$ guessed i right, $\mathcal{B}$ can use $\mathcal{A}$ to break the IND-CPA of the underlying ABTT and the strongly private collision-resistance of CHET.

$$\left|\Pr[S_1] - \Pr[S_2]\right| \leq Q(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t) + \mathbf{Adv}^{\mathsf{SPrivColl}}_{\mathcal{A},\mathcal{CHET}}(\kappa)).$$

As now $\mathcal{A}$ has no additional way to win this game, our statement is proven. Hence, the advantage of $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{CR}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) \leq 2Q(\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t) + \mathbf{Adv}^{\mathsf{SPrivColl}}_{\mathcal{A},\mathcal{CHET}}(\kappa)).$$

□

F. Security Proof of Theorem 5

Proof: Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ who can break the UNI security of our PCHA with non-negligible advantage. We can construct a probabilistic polynomial-time simulator $\mathcal{B}$ to break the UNI security of the underlying CHET executed by $\mathcal{C}$.

1) Setup: $\mathcal{B}$ sends κ to $\mathcal{C}$ and receives the public parameter $pp_{\mathcal{CHET}}$. $\mathcal{C}$ then runs $\mathcal{ABTT}.\mathsf{ParGen}(1^\kappa)$ and $\mathcal{IBS}.\mathsf{ParGen}(1^\kappa)$ to obtain $pp_{\mathcal{ABTT}}$ and $pp_{\mathcal{IBS}}$. $\mathcal{B}$ sends $pp = (pp_{\mathcal{ABTT}}, pp_{\mathcal{CHET}}, pp_{\mathcal{IBS}})$ to $\mathcal{A}$.

2) Output: $\mathcal{A}$ outputs $(mpk, \bar{w}^*, m^*, h^*, r^*, r'^*, \sigma^*_{\bar{w}})$, where $h^* = (h^*_{\mathcal{CHET}}, c_{\mathcal{ABTT}})$. $\mathcal{B}$ forwards $(mpk, m^*, h^*_{\mathcal{CHET}}, r^*, r'^*)$ to $\mathcal{C}$.

The simulation is perfect from $\mathcal{A}$'s point of view. The advantage of $\mathcal{A}$ breaks UNI security is

$$\mathbf{Adv}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) \leq \mathbf{Adv}^{\mathsf{UNI}}_{\mathcal{A},\mathcal{CHET}}(\kappa).$$

where $\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{A},\mathcal{CHET}}(\kappa)$ is the advantage of breaking the uniqueness of CHET [28].

□

G. Security Proof of Theorem 6

Proof: Suppose there exists a probabilistic polynomial-time adversary $\mathcal{A}$ who can break the ACT security of our PCHA with non-negligible advantage. We can construct a probabilistic polynomial-time simulator $\mathcal{B}$ to break the CRT security of the underlying ABTT executed by $\mathcal{C}_{\mathcal{ABTT}}$ and the existential unforgeability of the underlying IBS executed by $\mathcal{C}_{\mathcal{IBS}}$.

1) Setup: $\mathcal{B}$ sends (κ, n, t) to $\mathcal{C}_{\mathcal{ABTT}}$ and receives the public parameter $pp_{\mathcal{ABTT}}$ and the master public key $mpk_{\mathcal{ABTT}}$. $\mathcal{B}$ sends (κ, n, t) to $\mathcal{C}_{\mathcal{IBS}}$ and receives the public parameter $pp_{\mathcal{IBS}}$ and the master public key $mpk_{\mathcal{IBS}}$. $\mathcal{C}$ then runs $\mathcal{CHET}.\mathsf{ParGen}(1^\kappa)$ to obtain $pp_{\mathcal{CHET}}$ and $\mathcal{CHET}.\mathsf{KeyGen}(1^\kappa)$ to obtain $sk_{\mathcal{CHET}}$ and $sk_{\mathcal{CHET}}$. $\mathcal{C}$ returns $pp = (pp_{\mathcal{ABTT}}, pp_{\mathcal{CHET}}, pp_{\mathcal{IBS}})$ and $mpk = (mpk_{\mathcal{ABTT}}, pk_{\mathcal{CHET}}, mpk_{\mathcal{IBS}})$ to $\mathcal{A}$.

2) Query: $\mathcal{A}$ can adaptively query the following oracles.

- $\mathcal{O}_{\mathsf{UKeyGen}}(\cdot)$: $\mathcal{A}$ queries on an identity $\bar{w}$. $\mathcal{B}$ forwards $\bar{w}$ to $\mathcal{C}_{\mathcal{IBS}}$ and receives the secret key $sk^{(\bar{w})}_{\mathcal{IBS}}$. $\mathcal{B}$ returns $sk_{\bar{w}} \leftarrow sk^{(\bar{w})}_{\mathcal{IBS}}$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{UKeyGen}}(\cdot,\cdot)$: $\mathcal{A}$ queries on a set of attributes $\mathcal{S}$ and an identity $\bar{w}$. $\mathcal{B}$ forwards $\bar{w}$ to $\mathcal{C}_{\mathcal{ABTT}}$ and receives the secret key $sk^{(\bar{w})}_{\mathcal{IBS}}$. $\mathcal{B}$ forwards $\bar{w}$ to $\mathcal{C}_{\mathcal{IBS}}$ and receives the secret key $sk^{(\mathcal{S},\bar{w})}_{\mathcal{ABTT}}$. $\mathcal{B}$ returns $sk_{\bar{w}} \leftarrow sk^{(\bar{w})}_{\mathcal{IBS}}$ and $sk_{\mathcal{S},\bar{w}} \leftarrow sk^{(\mathcal{S},\bar{w})}_{\mathcal{ABTT}}$ to $\mathcal{A}$.

3) Output: $\mathcal{A}$ outputs a key-like decryption box $\mathcal{D}$. $\mathcal{B}$ runs $\mathcal{PCHA}.\mathsf{Judge}$. If $\mathcal{D}$ can derive a valid signature $\sigma_{\bar{w}'}$, $\mathcal{B}$ forwards $(\bar{w}', \sigma_{\bar{w}'})$ to $\mathcal{C}_{\mathcal{IBS}}$ as valid forgery. If $\mathcal{D}$ can trace to the users whose secret keys have not been queried before, $\mathcal{B}$ forwards the modified $\mathcal{D}'$ generated in $\mathcal{PCHA}.\mathsf{Trace}$ to $\mathcal{C}_{\mathcal{ABTT}}$.

The simulation is perfect from $\mathcal{A}$'s point of view. The advantage of $\mathcal{A}$ breaks the ACT security is

$$\mathbf{Adv}^{\mathsf{ACT}}_{\mathcal{A},\mathcal{PCHA}}(\kappa, n, t) \leq \mathbf{Adv}^{\mathsf{CRT}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t) + \mathbf{Adv}^{\mathsf{EU}}_{\mathcal{A},\mathcal{IBS}}(\kappa).$$

where $\mathbf{Adv}^{\mathsf{CRT}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t)$ is the advantage of breaking the collision-resistant traceability of ABTT and $\mathbf{Adv}^{\mathsf{CRT}}_{\mathcal{A},\mathcal{ABTT}}(\kappa, n, t)$ is the advantage of breaking the existential unforgeability of IBS. □

REFERENCES

[1] S. Agrawal and M. Chase, "FAME: Fast attribute-based message encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 665–682.

[2] J. M. L. Alfonsín, "Argentina: The right to be forgotten," in *The Right To Be Forgotten*. Springer, 2020, pp. 239–248. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-33512-0_12

[3] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain–or–rewriting history in bitcoin and friends," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2017, pp. 111–126.

[4] N. Attrapadung, "Dual system encryption framework in prime-order groups via computational pair encodings," in *Advances in Cryptology* (Lecture Notes in Computer Science). Hanoi, Vietnam: Springer, 2016, pp. 591–623. [Online]. Available: https://dblp.uni-trier.de/rec/conf/asiacrypt/Attrapadung16.html?view=bibtex

[5] E. Barker et al., "Recommendation for key management: Part 1: General," Nat. Inst. Standards Technol., Technol. Admin., Gaithersburg, MD, USA, Tech. Rep., 2006. [Online]. Available: https://blkcipher.pl/assets/pdfs/NIST.SP.800-57pt1r5.pdf

[6] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Advances in Cryptology*, vol. 3494, R. Cramer, Ed. Aarhus, Denmark: Springer, 2005, pp. 440–456. [Online]. Available: https://dblp.uni-trier.de/rec/conf/eurocrypt/BonehBG05.html?view=bibtex

[7] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data (extended bstract)," in *Advances in Cryptology*. Santa Barbara, CA, USA: Springer, 1995, pp. 452–465. [Online]. Available: https://dblp.uni-trier.de/rec/conf/crypto/BonehS95.html?view=bibtex

[8] J. Camenisch, D. Derler, S. Krenn, C. H. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors- and applications to invisible sanitizable signatures," in *Proc. PKC*, 2017, pp. 152–182.

[9] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–51.

[10] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," in *Proc. IEEE Symp. Secur. Privacy*, May 2019, pp. 124–138.

[11] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology*. Santa Barbara, CA, USA: Springer, 1999, pp. 537–554. [Online]. Available: https://dblp.uni-trier.de/rec/conf/crypto/FujisakiO99.html?view=bibtex

[12] A. Guillevic, "Comparing the pairing efficiency over composite-order and prime-order elliptic curves," in *Applied Cryptography and Network Security*. Banff, AB, Canada: Springer, 2013, pp. 357–372. [Online]. Available: https://dblp.uni-trier.de/rec/conf/acns/Guillevic13.html?view=bibtex

[13] A. Guillevic. (2021). *Pairing-Friendly Curves*. [Online]. Available: https://members.loria.fr/AGuillevic/pairing-friendly-curves

[14] F. Guo, Y. Mu, and W. Susilo, "Identity-based traitor tracing with short private key and short ciphertext," in *Computer Security*. 2012, pp. 609–626.

[15] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc. SAC*, 2002, pp. 310–324.

[16] H. Hou, S. Hao, J. Yuan, S. Xu, and Y. Zhao, "Fine-grained and controllably redactable blockchain with harmful data forced removal," *Secur. Commun. Netw.*, vol. 2021, pp. 1–20, May 2021.

[17] Y. Jia, S.-F. Sun, Y. Zhang, Z. Liu, and D. Gu, "Redactable blockchain supporting supervision and self-management," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, May 2021, pp. 844–858.

[18] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures," *IACR Cryptol. ePrint Arch.*, vol. 1998, p. 10, Mar. 1998.

[19] Z. Liu, Z. Cao, and D. S. Wong, "Blackbox traceable CP-ABE: How to catch people leaking their keys by selling decryption devices on ebay," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 475–486.

[20] Z. Liu, Z. Cao, and D. S. Wong, "Traceable CP-ABE: How to trace decryption devices found in the wild," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 55–68, Jan. 2015.

[21] Z. Liu and D. S. Wong, "Practical attribute-based encryption: Traitor tracing, revocation and large universe," *Comput. J.*, vol. 59, no. 7, pp. 983–1004, Jul. 2016.
[22] J. Ma, S. Xu, J. Ning, X. Huang, and R. H. Deng, "Redactable blockchain in decentralized setting," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1227–1242, 2022.
[23] R. Matzutt et al., "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *Proc. FC*, 2018, pp. 420–438.
[24] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[25] J. Ning, Z. Cao, X. Dong, J. Gong, and J. Chen, "Traceable CP-ABE with short ciphertexts: How to catch people selling decryption devices on ebay efficiently," in *Computer Security*. Heraklion, Greece: Springer, 2016, pp. 551–569. [Online]. Available: https://dblp.uni-trier.de/rec/conf/esorics/NingCDG016.html?view=bibtex
[26] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," in *Advances in Cryptology*. Beijing, China: Springer, 2012, pp. 349–366. [Online]. Available: https://dblp.uni-trier.de/rec/conf/asiacrypt/OkamotoT12.html?view=bibtex
[27] G. Panwar, R. Vishwanathan, and S. Misra, "ReTRACe: Revocable and traceable blockchain rewrites using attribute-based cryptosystems," in *Proc. 26th ACM Symp. Access Control Models Technol.*, Jun. 2021, pp. 103–114.
[28] K. Samelin and D. Slamanig, "Policy-based sanitizable signatures," in *Topics in Cryptology*. San Francisco, CA, USA: Springer, 2020, pp. 538–563. [Online]. Available: https://dblp.uni-trier.de/rec/conf/ctrsa/SamelinS20.html?view=bibtex
[29] S. Schwerin, "Blockchain and privacy protection in the case of the European general data protection regulation (GDPR): A delphi study," *J. Brit. Blockchain Assoc.*, vol. 1, no. 1, p. 3554, 2018.
[30] Y. Tian, N. Li, Y. Li, P. Szalachowski, and J. Zhou, "Policy-based chameleon hash for blockchain rewriting with black-box accountability," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2020, pp. 813–828.
[31] P. Voigt and A. V. D. Bussche, "The EU general data protection regulation (GDPR)," *A Practical Guide*, 1st, ed. Cham, Switzerland: Springer, 2017.
[32] S. Xu, J. Ning, J. Ma, X. Huang, and R. H. Deng, "K-time modifiable and epoch-based redactable blockchain," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4507–4520, 2021.
[33] S. Xu, J. Ning, J. Ma, G. Xu, J. Yuan, and H. R. Deng, "Revocable policy-based chameleon hash," in *Computer Security*. Darmstadt, Germany: Springer, 2021, pp. 327–347. [Online]. Available: https://dblp.uni-trier.de/rec/conf/esorics/XuNMXYD21.html?view=bibtex
[34] S. Xu et al., "Efficient ciphertext-policy attribute-based encryption with blackbox traceability," *Inf. Sci.*, vol. 538, pp. 19–38, Oct. 2020.
[35] Z. Zhang, T. Li, Z. Wang, and J. Liu, "Redactable transactions in consortium blockchain: Controlled by multi-authority CP-ABE," in *Information Security and Privacy*, vol. 13083. Springer, 2021, pp. 408–429. [Online]. Available: https://dblp.uni-trier.de/rec/conf/acisp/ZhangLWL21.html?view=bibtex

Shengmin Xu received the Ph.D. degree from the School of Computing and Information Technology, University of Wollongong, Australia, in 2018. He is currently an Associate Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou, China. Previously, he was a Senior Research Engineer with the School of Computing and Information Systems, Singapore Management University, and a Research Fellow with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design. He has published over 40 research papers in refereed international conferences and journals, such as ESORICS, ACSAC, IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSICS, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. His research interests include cryptography and information security.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently an Associate Professor at the Thrust of Artificial Intelligence, Information Hub, The Hong Kong University of Science and Technology (Guangzhou), China. He has published over 160 research papers in refereed international conferences and journals, such as ACM CCS, Crypto, Asiacrypt, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON INFORMATION SECURITY AND FORENSICS. His work has been cited more than 10000 times at Google scholar. His research interests include cryptography and information security. He has served as the program/general chair or a program committee member for over 120 international conferences. He is in the Editorial Board of *International Journal of Information Security* and *Science China Information Sciences*.

Jiaming Yuan is currently pursuing the Ph.D. degree with the Computer and Information Science Department, University of Oregon, USA. Previously, he was a Research Engineer at the School of Computer and Information Systems, Singapore Management University, Singapore. He has published seven research papers in refereed international conferences and journals, such as ACM CCS, ESORICS, *ACM DLT*, IEEE DSC, and *Information Sciences*. His research interests include information security, applied cryptography, and cloud security.

Yingjiu Li (Member, IEEE) is currently a Ripple Professor with the Computer and Information Science Department, University of Oregon. He has published over 140 technical papers in international conferences and journals, and served in the program committees for over 80 international conferences and workshops, including top-tier cybersecurity conferences and journals. His research interests include the IoT security and privacy, mobile and system security, applied cryptography and cloud security, and data application security and privacy.

Robert H. Deng (Fellow, IEEE) has been an AXA Chair Professor of cybersecurity and a Professor of information systems with the School of Information Systems, Singapore Management University, Singapore, since 2004. His current research interests include data security and privacy, multimedia security, and network and system security. He was a recipient of the Distinguished Paper Award from NDSS in 2012, the Best Paper Award from CMS in 2012, and the Best Journal Paper Award from IEEE Communications Society in 2017. He has served on the Editorial Boards for many international journals, including the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.