10-2021

Revocable policy-based Chameleon hash

Shengmin XU

Jianting NING

Jinhua MA

Guowen XU

Jiaming YUAN

See next page for additional authors

Citation

XU, Shengmin; NING, Jianting; MA, Jinhua; XU, Guowen; YUAN, Jiaming; and DENG, Robert H.. Revocable policy-based Chameleon hash. (2021). *Proceedings of the 26th European Symposium on Research in Computer Security (ESORICS 2021), Darmstadt, Germany, October 4–8*. 327-347. Research Collection School Of Computing and Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/6741

Revocable Policy-Based Chameleon Hash

Shengmin Xu[1,2], Jianting Ning[1,3(✉)], Jinhua Ma[1,2], Guowen Xu[4], Jiaming Yuan[5], and Robert H. Deng[2]

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
jtning@fjnu.edu.cn

[2] School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore
{smxu,robertdeng}@smu.edu.sg

[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[4] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore
guowen.xu@ntu.edu.sg

[5] Computer and Information Science Department, University of Oregon, Eugene, OR 97403, USA
jiamingy@uoregon.edu

Abstract. Policy-based chameleon hash (PCH) is a cryptographic building block which finds increasing practical applications. Given a message and an access policy, for any chameleon hash generated by a PCH scheme, a chameleon trapdoor holder whose rewriting privileges satisfy the access policy can amend the underlying message without affecting the hash value. In practice, it is necessary to revoke the rewriting privileges of a trapdoor holder due to various reasons, such as change of positions, compromise of credentials, or malicious behaviours. In this paper, we introduce the notion of revocable PCH (RPCH) and formally define its security. We instantiate a concrete RPCH construction by putting forward a practical revocable attribute-based encryption (RABE) scheme which is adaptively secure under a standard assumption on prime-order pairing groups. As application examples, we show how to effectively integrate RPCH into mutable blockchain and sanitizable signature for revoking the rewriting privileges of any chameleon trapdoor holders. We implement our RPCH scheme and evaluate its performance to demonstrate its efficiency.

Keywords: Policy-based chameleon hash · Revocable attribute-based encryption · Mutable blockchain · Sanitizable signature

1 Introduction

Policy-based Chameleon Hash (PCH) generalizes the notion of chameleon hash by giving one the ability to compute a chameleon hash and associate an access

E. Bertino et al. (Eds.): ESORICS 2021, LNCS 12972, pp. 327–347, 2021.
https://doi.org/10.1007/978-3-030-88418-5_16

policy to the hash. Chameleon trapdoor holders are issued rewriting privileges based on their attributes. A chameleon trapdoor holder whose rewriting privileges satisfy the access policy of a chameleon hash can find arbitrary collisions of the hash. Since PCH was proposed, it has found increasing applications in mutable blockchain and sanitizable signature to support fine-grained and controlled modifiability.

Mutable Blockchain. Most existing blockchains are designed to be immutable such that transactions in a block cannot be altered once they are confirmed. However, in many practical application scenarios, blockchain rewriting is necessary for reasons such as removing inappropriate contents [27,28] and complying legal obligations [1]. A mutable blockchain allows a certain party, called transaction modifier [4] who holds a chameleon trapdoor to process blockchain rewriting in a controlled way. By applying traditional asymmetric-key encryption and chameleon hash [24], Ateniese et al. [4] introduced the notion of mutable blockchain and proposed a construction that realizes block-level rewriting. Derler et al. [16] introduced the first construction of PCH, which supports transaction-level rewriting in a fine-grained way. Deuber et al. [17] later proposed a permissionless transaction rewriting mechanism based on consensus-based e-voting [23]. Recently, Tian et al. [36] considered the accountability in PCH-based mutable blockchain.

Sanitizable Signature. As a variant of digital signatures, sanitizable signatures allow a signer and a signer designated party, called sanitizer [3], to hold a chameleon trapdoor for rewriting signed messages in a controlled way. By replacing the traditional hash with chameleon hash [24], Ateniese et al. [3] introduced the notion of sanitizable signature and proposed a construction that realizes *unforgeability*, *privacy*, *transparency*, *immutability* and *accountability*. Sanitizable signature with *unlinkability* was proposed by Fleischhacker et al. [19]. Camenisch et al. [14] introduced chameleon hash with ephemeral trapdoor (CHET) to realize *invisibility*. However, the aforementioned sanitizable signatures and some following works [8,13,18] mainly concentrated on investigating additional security requirements, the sanitizer cannot be managed efficiently and flexibly. In particular, the signer needs to know the sanitizer before the signature generation phase, and sanitizing capabilities are controlled in a coarse-grained way. To address this problem, a PCH-based sanitizable signature [34] is proposed to allow the signer to define multiple sanitizers for a signed message in a fine-grained way.

Motivation. In practice, a chameleon trapdoor holder may abuse her/his rewriting privileges, maliciously rewrite the hashed objects (e.g., transaction contents in mutable blockchain and messages in sanitizable signature) to spread inappropriate/incorrect contents, or even sell the rewriting privileges to gain illegal profits. Such abused privileges must be revoked as the actions of trapdoor holders impact the security, reputation, and robustness of the entire system. However, to the best of our knowledge, current PCH proposals [16,34,36] cannot provide revocability toward rewriting privileges.

To control rewriting privileges in a versatile way, fast attribute-based message encryption (FAME) [2], as a variant of attribute-based encryption (ABE), has been used as the underlying building block of PCH [16,34,36]. However, a PCH built on standard FAME does not support user revocation. To address this issue, revocable ABE (RABE) can be used to replace the underlying FAME and realize PCH with revocability. However, one cannot naively employ the existing direct/indirect RABE to replace FAME without sacrificing the security or performance of PCH. Specifically, direct RABE incurs large overhead and is impractical in real-world scenarios while indirect RABE suffers from weak security or poor performance compared to FAME.

Direct RABE [6,7,12,26]. There are two strategies in direct revocation. The first strategy is mainly built on the key generation center (KGC) who broadcasts an updated secret key for each non-revoked user via a secure channel periodically. The secret key is bound to attributes appending a timestamp, i.e., $att\|t$, where att is an attribute and t is timestamp updated in each time epoch. This approach is impractical since a ciphertext would contain a long policy to cover all possible t values and the secure channels for a periodic secret key update to all non-revoked users are expensive. The other approach is embedding identities of all the revoked users in the ciphertexts, which is also impractical since data owners must keep the revocation list update to date and the revocation list will grow longer as time goes by.

Indirect RABE [15,31,33,37–41]. To address the problem in direct revocation, indirect revocation divides a user's decryption privilege into a secret key and a public key-updating material. A KGC issues a secret key to a user when she/he joins the system and broadcasts the key-updating material to all users periodically over a public channel. Only non-revoked users can combine their secret keys and the key-updating material to obtain decryption privileges at each time period.

Although RABE[1] offers a potential solution to realize PCH with revocation, how to realize RABE based on FAME is an open problem. FAME [2], as a basic building block for PCH [16,34,36], is an adaptively secure ABE on prime-order groups. Previous RABE solutions generally rely on the property of linear master secret sharing, named ElGamal-type ABE [40], such as adaptively secure ABE (on composite-order groups) [25] used in [33] and selectively secure ABE (on prime-order groups) [32] used in [15,31,37,39,40]. In ElGamal-type ABE, the master secret key and ciphertext are in the form of α and $m \cdot e(g,g)^{\alpha s}$, respectively. By dividing α into $\alpha - \beta$ and β, the KGC applies $\alpha - \beta$ to issue secret keys to users and uses β to generate the public key-updating material. A revoked user cannot erase β in her/his secret key to process data decryption, and a non-revoked user can easily combine the secret key and the key-updating material to erase β to reveal the sealed message. However, FAME [2] is not an ElGamal-type ABE and cannot follow this strategy. The following problem arises naturally:

[1] In the rest of the paper, unless otherwise specified, RABE represents indirect RABE.

"Can RABE be built from FAME and further be integrated into PCH to revoke rewriting privileges?"

Our Contributions. In this paper, we give an affirmative answer to the above problem by introducing the first formal treatment for revocability to PCH, dubbed *revocable policy-based chameleon hash* (RPCH). We present a new *revocable attribute-based encryption*, which is adaptively secure under a standard assumption on prime-order pairing groups, and then based on it we realize a practical construction of RPCH. The major contributions of the paper are threefold.

- *Formal definition of RPCH.* RPCH extends PCH with revocability. We give a formal definition of RPCH and propose the notions of *fully indistinguishability*, *collision-resistance* and *uniqueness* in presence of attackers.
- *Adaptively secure RABE.* RABE serves as the fundamental building block to offer the properties of policy-based access control and revocability simultaneously. To instantiate an efficient RPCH construction, we present an adaptively secure RABE under a standard assumption on prime-order pairing groups.[2]
- *Practical RPCH construction.* We provide a concrete construction of RPCH with performance evaluation. Compared to PCH [16], our RPCH achieves revocability with negligible overhead. Specifically, our RPCH takes one extra exponentiation (around 3.83 ms) for hashing, and one more multiplication and pairing (around 21.19 ms) for collision finding. We also show that our RPCH is practical when integrating it into mutable blockchain and sanitizable signature.

2 Overview

In this section, we give an overview of the proposed RABE and RPCH, and the design intuitions behind them.

Overview of RABE Technique. We resort to the techniques in indistinguishability under chosen plaintext attacks (IND-CPA) secure FAME [2] and IND-CPA secure revocable identity-based encryption (RIBE) with decryption key[3] exposure resistance [35], where key exposure resistance guarantees that the exposed short-term decryption key does not affect the security of any other time periods. In our solution, the secret key follows the structure of FAME and imitates the form of a second-level secret key of the hierarchical identity-based encryption

[2] As explained above, previous RABE solutions are either selectively secure [15,37–41] or adaptively secure under non-standard assumptions or composite-order groups [33]. Guillevic [22] reported that bilinear pairings are 254 times slower in composite-order than in prime-order groups for the same 128-bit security. Despite dual pairing vector space [30] can transfer composite-order groups to prime-order groups, it could be paramount for enormous encoding schemes [5].

[3] In RABE, the decryption privilege is based on the decryption key, which is derived from the long-term secret key and public key-updating material.

(HIBE) [10] to achieve fine-grained access control and user revocation simultaneously. In the following, we describe the intuition behind our RABE construction from the perspectives of *secret key structure* and *key period management*.

- *Secret key structure.* We modify the structure of the FAME secret key by removing a random element g_θ, where θ is used to manage user revocation. At the beginning of each period, the key-updating material is published, and it contains g_θ with an additional time-based restriction. A non-revoked user can update her/his long-term secret key to obtain a FAME secret key with the time-based restriction, and this secret key can be used for decryption if the time-based restriction matches the timestamp associated with the ciphertext. To resist decryption key exposure attacks, we design a decryption key generation algorithm that derives a probabilistic short-term decryption key. Thus, by knowing the short-term decryption key and the public key-updating material, no one can derive the corresponding long-term secret key.
- *Key periodical management.* We rely on the key-update-nodes $\mathsf{KUNodes}$ algorithm [29] to reduce the size of the key-updating material from linear to the number of system users to logarithmic. Each chameleon trapdoor holder with an identifier id has $\log n$ secret keys that relate to the path of their positions in the tree-based state st to the tree root node, denoted as $\mathsf{Path}(id)$. The key update generation algorithm outputs a key-updating material based on $\mathsf{KUNodes}$ by inputting a state st, a revocation list rl, and a timestamp t. Each non-revoked user can find only one node $\theta \in \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$ to generate a decryption key, while revoked users cannot find it $\emptyset \in \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$, hence, they are revoked implicitly. Although the size of the secret key is increased from constant to logarithmic, the effect is minimal since the secret key is distributed once and the key-updating material is broadcast each period.

Overview of RPCH System Model. As shown in Fig. 1, RPCH allows a party, e.g., a data *owner*, to compute a chameleon hash associated with an access policy and a timestamp. Another party, called chameleon trapdoor holder or *modifier*, who possesses privileges and valid key-updating materials that satisfy the access policy and the timestamp in a given hash can then find arbitrary collisions. RPCH thus supports the modifiability at a fine-grained level and the revocability of the modifier rewriting privileges.

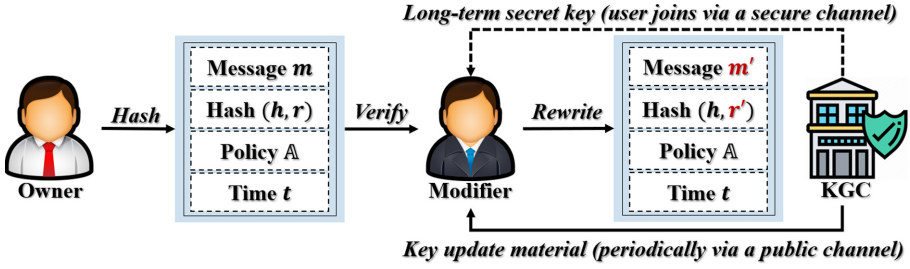

Fig. 1. System model of RPCH

Overview of RPCH Technique. We resort to the techniques in previous PCH [16], sanitizable signature [34] and our proposed RABE. Our PRCH follows the previous PCH solutions [16,34] by replacing the underlying FAME scheme with our proposed RABE, and combining with CHET [14]. In CHET, two trapdoors are used to guarantee the security of controlled rewriting: a long-term trapdoor and an ephemeral trapdoor. In RABE, we have an attribute-based long-term secret key issued when the modifier joins and a key-updating material publicly distributed each period. In the following, we describe the intuition behind our RPCH construction from the perspectives of *long-term secret key* and *ephemeral trapdoor*.

- *Long-term secret*. The long-term secret key consists of an attribute-based long-term secret key and a CHET long-term trapdoor, and both of them are issued when the modifier joins the system. The attribute-based long-term secret key is used to combine the key-updating material to enable the non-revoked modifier to derive a short-term decryption key and revoke the invalid modifier implicitly. The CHET long-term trapdoor cannot work for rewriting individually and must cooperate with the CHET ephemeral trapdoor.
- *Ephemeral trapdoor*. Each data owner picks a CHET ephemeral trapdoor during data hash and encrypts this trapdoor via hybrid encryption, where symmetric-key encryption to seal this trapdoor and RABE to seal the symmetric key. Hence, to operate the rewriting procedure, the modifier must have an attributed-based decryption key (that satisfies the access policy and the timestamp associated with the sealed ephemeral trapdoor) and the CHET long-term trapdoor. In other words, the rewriting privilege is authorized by the KGC and the data owner simultaneously. Note that, the data owner cannot process the rewriting procedure alone since the CHET long-term trapdoor is unknown to him/her.

We propose notions of *fully indistinguishability*, *collision-resistance* and *uniqueness* as the security requirements for RPCH. Derler et al. [16] introduced the first PCH, and its security in terms of indistinguishability and collision-resistance. Samelin and Slamanig [34] improved the security by introducing the notions of full indistinguishability, collision-resistance, and uniqueness. We refine security notations in the above works by additionally considering several different types of adversaries. In particular, in our security model, an adversary could be the combination of outsiders, insiders without valid secret keys, and insiders with valid secret keys but being revoked.

3 Preliminaries

In this section, we present the hard assumption, access structure and some building blocks, which are used in our proposed RABE and PCH schemes.

3.1 Bilinear Map

Let $\mathcal{G}$ be an asymmetric prime-order (Type-III) pairing group generator that on input 1^κ, outputs description of three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with a bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, and generators g and h for $\mathbb{G}$ and $\mathbb{H}$ with following properties: 1) bilinearity: for all $u \in \mathbb{G}, v \in \mathbb{H}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$; 2) non-degeneration: $e(g, h) \neq 1$; and 3) computability: it is efficient to compute $e(u, v)$ for any $u \in \mathbb{G}$ and $v \in \mathbb{H}$.

3.2 Hard Assumption

Definition 1 (DLIN). *A bilinear pairing group generator $\mathcal{G}$ satisfies the decisional linear assumption (DLIN) if for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Adv}^{\mathsf{DLIN}}_{\mathcal{A}}(\kappa) = \left|\Pr[\mathcal{A}(1^\kappa, pp, D, T_0)] - \Pr[\mathcal{A}(1^\kappa, pp, D, T_1)]\right|$ is negligible in the security parameter κ, where $pp = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^\kappa)$; $a_1, a_2 \in \mathbb{Z}_p^*$; $s_1, s_2, s \in \mathbb{Z}_p$; $D = (g^{a_1}, g^{a_2}, h^{a_1}, h^{a_2}, g^{a_1 s_1}, g^{a_2 s_2}, h^{a_1 s_1}, h^{a_2 s_2})$; $T_0 = (g^{s_1+s_2}, h^{s_1+s_2})$; and $T_1 = (g^s, h^s)$.*

3.3 Access Structure

Definition 2 (Access Structure). *Let $\mathbb{U}$ denote the universe of attributes. A collection $\mathbb{A} \in 2^{\mathbb{U}} \setminus \{\emptyset\}$ of non-empty sets is an access structure on $\mathbb{U}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets. A collection $\mathbb{A} \in 2^{\mathbb{U}} \setminus \{\emptyset\}$ is called monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

3.4 Revocable Attribute-Based Encryption

Definition 3 (RABE). *Let $\mathcal{I}$ be an identifier space, $\mathcal{M}$ denote a message space and $\mathcal{T}$ be a time space. A revocable attribute-based encryption with decryption key exposure resistance $\mathcal{RABE}$ consists of the algorithms* $\{\mathsf{Setup}, \mathsf{KGen}, \mathsf{KUpt}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev}\}$ *such that:*

- $\mathsf{Setup}(1^\kappa, n) \to (mpk, msk, st, rl)$: *The probabilistic setup algorithm takes a security parameter $\kappa \in \mathbb{N}$ and the number of system users $n \in \mathbb{N}$ as input, and outputs a master public key mpk, a master secret key sk, a state st and a revocation list rl, where 1^κ and mpk are implicit input to all other algorithms.*
- $\mathsf{KGen}(msk, st, id, \mathcal{S}) \to (sk_{id}, st)$: *The probabilistic key generation algorithm takes a master secret key msk, a state st, an identifier $id \in \mathcal{I}$ and an attribute set $\mathcal{S} \subseteq \mathbb{U}$ as input, and outputs a secret key sk_{id} and an updated state st.*
- $\mathsf{KUpt}(st, rl, t) \to ku_t$: *The probabilistic key update generation algorithm takes a state st, a revocation list rl and a timestamp $t \in \mathcal{T}$ as input, and outputs a key-updating material ku_t. Note that st is kept secret by the KGC.*
- $\mathsf{DKGen}(sk_{id}, ku_t) \to dk_{id,t}$: *The probabilistic decryption key generation algorithm takes a secret key sk_{id} and a key-updating material ku_t as input, and outputs a decryption key $dk_{id,t}$.*

- $\mathsf{Enc}(m, \mathbb{A}, t) \rightarrow c$: *The probabilistic encryption algorithm takes a message* $m \in \mathcal{M}$*, an access policy* $\mathbb{A} \in 2^{\mathbb{U}}$ *and a timestamp* $t \in \mathcal{T}$ *as input, and outputs a ciphertext* c*.*
- $\mathsf{Dec}(dk_{id,t}, c) \rightarrow m$: *The deterministic decryption algorithm takes a decryption key* $dk_{id,t}$ *and a ciphertext* c *as input, and outputs a message* $m \in \mathcal{M}$*.*
- $\mathsf{Rev}(rl, id, t) \rightarrow rl$: *The deterministic revocation algorithm takes a revocation list* rl*, an identifier* $id \in \mathcal{I}$ *and a timestamp* $t \in \mathcal{T}$ *as input, and outputs an updated revocation list* rl*.*

Definition 4 (IND-CPA)**.** *The security definition of* IND-CPA *for* $\mathcal{RABE}$ *between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*.*

Setup. $\mathcal{C}$ *runs* $\mathsf{Setup}(1^{\kappa}, n)$ *and gives* mpk *to* $\mathcal{A}$*.* $\mathcal{C}$ *keeps* msk, st *and* rl *secret.*

Phase 1. $\mathcal{A}$ *adaptively issues a sequence of following queries to* $\mathcal{C}$*.*

- $\mathcal{O}_{\mathsf{KGen}}(\cdot,\cdot)$*:* $\mathcal{A}$ *issues key generation query on an identifier* $id \in \mathcal{I}$ *and a set of attributes* $\mathcal{S} \subseteq \mathbb{U}$*.* $\mathcal{C}$ *returns a secret key* sk_{id} *by running* $\mathsf{KGen}(msk, st, id, \mathcal{S})$*.*
- $\mathcal{O}_{\mathsf{KUpt}}(\cdot)$*:* $\mathcal{A}$ *issues key update query on a timestamp* $t \in \mathcal{T}$*.* $\mathcal{C}$ *returns a key-updating material* ku_t *by running* $\mathsf{KUpt}(st, rl, t)$*.*
- $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$*:* $\mathcal{A}$ *issues decryption key generation query on an identifier* $id \in \mathcal{I}$*, a set of attributes* $\mathcal{S} \subseteq \mathbb{U}$ *and a timestamp* $t \in \mathcal{T}$*.* $\mathcal{C}$ *returns a decryption key* $dk_{id,t}$ *by running* $\mathsf{DKGen}(sk_{id}, ku_t)$ *if the secret key* sk_{id} *and the key-updating material* ku_t *are available. Otherwise,* $\mathcal{C}$ *generates* sk_{id} *and* ku_t *first.*
- $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$*:* $\mathcal{A}$ *issues revocation query on an identifier* $id \in \mathcal{I}$ *and a timestamp* $t \in \mathcal{T}$*.* $\mathcal{C}$ *updates the revocation list* rl *by running* $\mathsf{Rev}(rl, id, t)$*.*

$\mathcal{A}$ *is allowed to query above oracles with the following restrictions:*

1. $\mathcal{O}_{\mathsf{KUpt}}(\cdot)$ *and* $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ *can be queried at the time* $t \in \mathcal{T}$ *which is greater than or equal to that of all previous queries.*
2. $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ *cannot be queried at the time* $t \in \mathcal{T}$ *if* $\mathcal{O}_{\mathsf{KUpt}}(\cdot)$ *was queried at the time* $t \in \mathcal{T}$*.*

Challenge. $\mathcal{A}$ *outputs two messages* m_0 *and* m_1 *of the same size, an access structure* $\mathbb{A}^*$ *and a timestamp* $t^* \in \mathcal{T}$*.* $\mathcal{C}$ *terminates if the previous queries against the following restrictions:*

3. $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ *cannot be queried any identifier* $id \in \mathcal{I}$ *with any set of attributes* $\mathcal{S} \models \mathbb{A}^*$ *at the challenge time* $t^* \in \mathcal{T}$*.*
4. $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ *must be queried the identifier* $id \in \mathcal{I}$ *associated attributes* $\mathcal{S} \models \mathbb{A}^*$ *and the timestamp* $t \leq t^*$ *if* $\mathcal{O}_{\mathsf{KGen}}(\cdot,\cdot)$ *was queried any identifier* $id \in \mathcal{I}$ *and any set of attributes* $\mathcal{S} \models \mathbb{A}^*$*.*

$\mathcal{C}$ *randomly chooses* $b \in \{0,1\}$*, and returns* c^* *to* $\mathcal{A}$ *by running* $\mathsf{Enc}(m_b, \mathbb{A}^*, t^*)$*.*

Phase 2. $\mathcal{A}$ *continues issuing queries to* $\mathcal{C}$ *with the previous restrictions.*

Guess. $\mathcal{A}$ *makes a guess* b' *for* b*, and it wins the game if* $b' = b$*.*

The advantage of $\mathcal{A}$ *in this game is defined as* $\mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{RABE},\mathcal{A}}(\kappa, n) = \left|\Pr[b = b'] - 1/2\right|$*. An* $\mathcal{RABE}$ *is* IND-CPA *secure if any probabilistic polynomial time adversary has at most a negligible advantage in* κ*.*

3.5 Tree-Based Structure for User Revocation

The tree-based revocation list has been widely used to reduce the computation and communication costs of key-updating materials from linear to logarithmic. The basic idea is to find a minimum of sub-tree roots to cover all non-revoked users. Specifically, as shown in Fig. 2, each data user is assigned to an individual leaf node in a binary tree and is issued secret keys from the corresponding leaf node to the root (see Step 1). To revoke users (see Step 2), we only need to find sub-tree roots that are exclusive of revoked users to generate key updates (see Step 3).

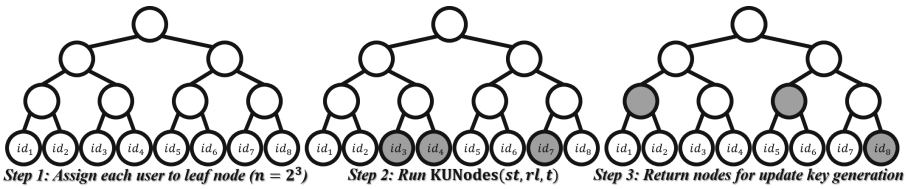

Fig. 2. Tree-based revocation structure

The key-update-nodes algorithm $\mathsf{KUNodes}(st, rl, t)$ [29] is proposed to process the above mechanism. It takes a state st denoting the state of the binary tree, a revocation list rl, and a timestamp t, and outputs a set of nodes, where rl records a set of identifier and timestamp pairs. When a user joins the system, who will be labeled a random identifier id and assigned to an undefined leaf node. The user id will be issued a set of keys related to $\mathsf{Path}(id)$, where $\mathsf{Path}(id)$ denotes all nodes in the path from the root node to the leaf node id. The formal definition for $\mathsf{KUNodes}$ algorithm is referred to [9], which first introduced $\mathsf{KUNodes}$ algorithm in the revocable identity-based cryptosystem.

4 Revocable Policy-Based Chameleon Hash

In this section, we present the system model. Then, we present the formal definition and the security model.

4.1 System Model

PCH is a chameleon hash system with controlled rewriting privilege at a fine-grained level and involves three types of users: KGC, data owner, and data modifier. In terms of a decentralized setting, every user can play the role of a KGC and tag other users with attributes. We use data "owner" and "modifier" in presenting RPCH. In particular, we assume the number of the modifier is a small amount since the rewriting privilege should be controlled carefully and cannot be performed by the majority of users.

As shown in Fig. 1, the KGC issues the long-term secret key when the modifier joins the system via a secure channel and distributes the key-updating material periodically via a public channel. Each owner is allowed to generate chameleon hash (h, r) by setting an access policy $\mathbb{A}$ and a timestamp t for rewriting the hashed object. The modifier whose long-term secret key satisfies the policy associated with the hashed object $\mathcal{S} \models \mathbb{A}$ and the valid key-updating material in period t can operate the rewriting produce by outputting a chameleon collision.

4.2 Formal Definition

Definition 5 (RPCH). *A revocable policy-based chameleon hash $\mathcal{RPCH}$ consists of the algorithms* {Setup, KGen, KUpt, DKGen, Rev, Hash, Verify, Adapt}*, where the algorithms* Setup, KGen, KUpt, DKGen*, and* Rev *are described in $\mathcal{RABE}$. The definition for the rest of algorithms* Hash, Verify*, and* Adapt *is:*

- $\mathsf{Hash}(m, \mathbb{A}, t) \rightarrow (h, r)$*: The probabilistic hash algorithm takes a message $m \in \mathcal{M}$, an access policy $\mathbb{A} \in 2^{\mathbb{U}}$ and a timestamp $t \in \mathcal{T}$ as input, and outputs a chameleon hash h and a randomness (sometimes referred to as "check value") r.*
- $\mathsf{Verify}(m, h, r) \rightarrow b$*: The deterministic verification algorithm takes a message $m \in \mathcal{M}$, a chameleon hash h and a randomness r as input, and outputs a bit $b \in \{0, 1\}$.*
- $\mathsf{Adapt}(dk_{id,t}, m, m', h, r) \rightarrow r'$*: The deterministic adaptation algorithm takes a decryption key $dk_{id,t}$, a message $m \in \mathcal{M}$, a message $m' \in \mathcal{M}$, a chameleon hash h and a randomness r as input, and outputs a randomness r'.*

For each $\mathcal{RPCH}$, it is required that the correctness property holds. In particular, it is required that for all $\kappa \in \mathbb{N}$, for all $(mpk, msk, st, rl) \leftarrow \mathsf{Setup}(1^\kappa, n)$, for all $\mathcal{S} \subseteq \mathbb{U}$, for all $(sk_{id}, st) \leftarrow \mathsf{KGen}(msk, st, id, \mathcal{S})$, for all $t \in \mathcal{T}$, for all $ku_t \leftarrow \mathsf{KUpt}(st, rl, t)$, for all $dk_{id,t} \leftarrow \mathsf{DKGen}(sk_{id}, ku_t)$, for all $m \in \mathcal{M}$, for all $(h, r) \leftarrow \mathsf{Hash}(m, \mathbb{A}, t)$, for all $m' \in \mathcal{M}$, for all $r' \leftarrow \mathsf{Adapt}(dk_{id,t}, m, m', h, r)$, we have $1 = \mathsf{Verify}(m, h, r) = \mathsf{Verify}(m', h, r')$.

4.3 Security Model

Definition 6 (FIND). *The security definition of full indistinguishability (*FIND*) for $\mathcal{RPCH}$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

Setup. *$\mathcal{C}$ runs* $\mathsf{Setup}(1^\kappa, n)$ *and gives mpk and msk to $\mathcal{A}$.*

Query Phase. *$\mathcal{C}$ randomly picks $b \in \{0, 1\}$. $\mathcal{A}$ issues the following queries to $\mathcal{C}$.*

- $\mathcal{O}_{\mathsf{HashOrAdapt}}(\cdot, \cdot, \cdot, \cdot, \cdot)$*: $\mathcal{A}$ issues hash or adapt query on two messages $m, m' \in \mathcal{M}$, a decryption key $dk_{id,t}$, an access policy $\mathbb{A} \in 2^{\mathbb{U}}$ and a timestamp $t \in \mathcal{T}$. $\mathcal{C}$ generates (h_0, r_0) by running* $\mathsf{Hash}(m', \mathbb{A}, t)$*, (h_1, r_1) by running* $\mathsf{Hash}(m, \mathbb{A}, t)$ *and r_1 by running* $\mathsf{Adapt}(dk_{id,t}, m, m', h_1, r_1)$*. $\mathcal{C}$ returns $\perp$ if $r_0 = \perp$ or $r_1 = \perp$. Otherwise, $\mathcal{C}$ returns (h_b, r_b).*

Guess. *$\mathcal{A}$ makes a guess b' for b.*

The advantage of $\mathcal{A}$ in this game is defined as $\mathbf{Adv}^{\mathsf{FIND}}_{\mathcal{RPCH},\mathcal{A}}(\kappa, n) = \left|\Pr[b = b'] - 1/2\right|$. *An $\mathcal{RPCH}$ is* FIND *if any probabilistic polynomial time adversary has at most a negligible advantage in κ.*

Definition 7 (ICR). *The security definition of insider collision-resistance (*ICR*) for $\mathcal{RPCH}$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*[4]

Setup. *$\mathcal{C}$ runs* $\mathsf{Setup}(1^\kappa, n)$ *and gives mpk to $\mathcal{A}$. $\mathcal{C}$ keeps msk, st and rl secret.*

Query Phase. *$\mathcal{A}$ adaptively issues a sequence of following queries to $\mathcal{C}$.*

- $\mathcal{O}_{\mathsf{KGen}}(\cdot,\cdot)$, $\mathcal{O}_{\mathsf{KUpt}}(\cdot)$, $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$ *and* $\mathcal{O}_{\mathsf{Rev}}(\cdot,\cdot)$ *are the same oracles defined in the* IND-CPA *security model for $\mathcal{RABE}$ with the same restrictions.*
- $\mathcal{O}'_{\mathsf{KGen}}(\cdot,\cdot)$*: $\mathcal{A}$ issues key generation query on an identifier $id \in \mathcal{I}$ and a set of attributes $\mathcal{S} \subseteq \mathbb{U}$. $\mathcal{C}$ runs* $\mathsf{KGen}(msk, st, id, \mathcal{S})$ *and keeps the secret key sk_{id}.*
- $\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot)$*: $\mathcal{A}$ issues hash query on a message $m \in \mathcal{M}$, an access policy $\mathbb{A} \in 2^{\mathbb{U}}$ and a timestamp $t \in \mathcal{T}$. $\mathcal{C}$ returns (h, r) by running* $\mathsf{Hash}(m, \mathbb{A}, t)$.
- $\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot)$*: $\mathcal{A}$ issues adaption query on two messages $m, m' \in \mathcal{M}$, an access policy $\mathbb{A} \in 2^{\mathbb{U}}$, a timestamp $t \in \mathcal{T}$ and an identifier $id \in \mathcal{I}$. $\mathcal{C}$ returns $\perp$ if id was not queried to $\mathcal{O}_{\mathsf{KGen}}(\cdot,\cdot)$ before and $\mathcal{O}'_{\mathsf{KGen}}(\cdot,\cdot)$, or id's corresponding attributes $\mathcal{S} \not\models \mathbb{A}$. Otherwise, $\mathcal{C}$ returns r' by running* $\mathsf{Adapt}(dk_{id,t}, m, m', h, r)$.

Output. *$\mathcal{A}$ outputs two different messages $m^*, m'^* \in \mathcal{M}$, two randomnesses r^*, r'^*, a chameleon hash h^* and a timestamp $t^* \in \mathcal{T}$, and it wins the game if*

- $1 = \mathsf{Verify}(m^*, h^*, r^*) = \mathsf{Verify}(m'^*, h^*, r'^*)$,
- *h^* appears in $\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot)$ or $\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot)$ with some $\mathbb{A}^* \in 2^{\mathbb{U}}$ and $t^* \in \mathcal{T}$,*
- *(h^*, m^*) does not appear in $\mathcal{O}_{\mathsf{Hash}}(\cdot,\cdot,\cdot)$ or $\mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot,\cdot,\cdot,\cdot)$,*
- *any $\mathcal{S} \models \mathbb{A}^*$ and $t^* \in \mathcal{T}$ have not been queried to $\mathcal{O}_{\mathsf{DKGen}}(\cdot,\cdot,\cdot)$, and*
- *any $id \in \mathcal{I}$ with $\mathcal{S} \models \mathbb{A}^*$ was revoked in $t \leq t^*$ or has never been queried to $\mathcal{O}_{\mathsf{KGen}}(\cdot,\cdot)$.*

The advantage of $\mathcal{A}$ in this game is defined as $\mathbf{Adv}^{\mathsf{ICR}}_{\mathcal{RPCH},\mathcal{A}}(\kappa, n) = \Pr[\mathcal{A}\ wins]$. *An $\mathcal{RPCH}$ is* ICR *if any probabilistic polynomial time adversary has at most a negligible advantage in κ.*

Definition 8 (UNI). *The security definition of uniqueness (*UNI*) for $\mathcal{RPCH}$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

Setup. *$\mathcal{C}$ runs* $\mathsf{Setup}(1^\kappa, n)$ *and gives mpk and msk to $\mathcal{A}$.*

Output. *$\mathcal{A}$ outputs a message $m^* \in \mathcal{M}$, two randomness r^*, r'^* and a chameleon hash h^*. It wins the game if* $1 = \mathsf{Verify}(m^*, h^*, r^*) = \mathsf{Verify}(m^*, h^*, r'^*)$ *and $r^* \neq r'^*$.*

The advantage of $\mathcal{A}$ in this game is defined as $\mathbf{Adv}^{\mathsf{UNI}}_{\mathcal{RPCH},\mathcal{A}}(\kappa, n) = \Pr[\mathcal{A}\ wins]$. *An $\mathcal{RPCH}$ is* UNI *if any probabilistic polynomial time adversary has at most a negligible advantage in κ.*

[4] To simplicity, the weak model, outsider collision-resistance [16] has not taken into consideration since ICR covers this weak model as in [34].

5 Revocable Policy-Based Chameleon Hash

In this section, we present the concrete constructions of RABE and RPCH, respectively. Then, we show that RPCH can be effectively integrated into mutable blockchain and sanitizable signature.

5.1 Proposed RABE

Before presenting the construction of RPCH, we introduce the construction of RABE first, which serves as an important building block to RPCH. Our RABE uses a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$, and it will be modeled as a random oracle in the security proof. In particular, three types of inputs will be given to $\mathcal{H}$: inputs of the form (y, ℓ, z), (j, ℓ, z) or t, where $y \in \mathcal{S}$, $j \in \mathbb{N}$, $\ell \in \{1,2,3\}$, $z \in \{1,2\}$ and $t \in \mathcal{T}$. For the seek of readability, we represent these three inputs as $y\ell z$, $0j\ell z$ and $1t$, respectively, appending 0 at the beginning of the second one and 1 at the beginning of the third one so that it is not confused each other. We assume that the inputs are appropriately encoded so that no three different tuples collide. In the following, we present the concrete construction of RABE and the sketch of the security proof.

$\mathsf{Setup}(1^\kappa, n)$**:** Run $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e, g, h) \leftarrow \mathcal{G}(1^\kappa)$. Pick $a_1, a_2, b_1, b_2 \in \mathbb{Z}_p^*$ and $d_1, d_2, d_3 \in \mathbb{Z}_p$. Output $mpk = (h, H_1 = h^{a_1}, H_2 = h^{a_2}, T_1 = e(g,h)^{d_1 a_1 + d_3}, T_2 = e(g,h)^{d_2 a_2 + d_3}, \mathcal{H}), msk = (g, h, a_1, a_2, b_1, b_2, g^{d_1}, g^{d_2}, g^{d_3}), st \leftarrow \mathsf{BT}$ and $rl \leftarrow \emptyset$, where BT denotes a binary tree with n leaf nodes.

$\mathsf{KGen}(msk, st, id, \mathcal{S})$**:** Pick $r_1, r_2 \in \mathbb{Z}_p$ and compute $sk_0 = (h^{b_1 r_1}, h^{b_2 r_2}, h^{r_1 + r_2})$ using h, b_1, b_2 from msk. For all $y \in \mathcal{S}$ and $z = 1, 2$:

1. Pick $\sigma_y, \sigma' \in \mathbb{Z}_p$.
2. Compute $sk_{y,z} = \mathcal{H}(y1z)^{\frac{b_1 r_1}{a_z}} \cdot \mathcal{H}(y2z)^{\frac{b_2 r_2}{a_z}} \cdot \mathcal{H}(y3z)^{\frac{r_1 + r_2}{a_z}} \cdot g^{\frac{\sigma_y}{a_z}}$.
3. Compute $sk'_z = g^{d_z} \cdot \mathcal{H}(011z)^{\frac{b_1 r_1}{a_z}} \cdot \mathcal{H}(012z)^{\frac{b_2 r_2}{a_z}} \cdot \mathcal{H}(013z)^{\frac{r_1 + r_2}{a_z}} \cdot g^{\frac{\sigma'}{a_z}}$.

Set $sk_y = (sk_{y,1}, sk_{y,2}, g^{-\sigma_y})$ and $sk' = (sk'_1, sk'_2)$. Pick an unassigned left node in st and label id to it. For all $\theta \in \mathsf{Path}(id)$:

1. Fetch g_θ if available; else pick $g_\theta \in \mathbb{G}$ and store g_θ in θ to update st.
2. Compute $sk_\theta = g^{d_3} \cdot g^{-\sigma'} / g_\theta$.

Output $sk_{id} = (\mathcal{S}, sk_0, \{sk_y\}_{y \in \mathcal{S}}, sk', \{\theta, sk_\theta\}_{\theta \in \mathsf{Path}(id)})$ and st.

$\mathsf{KUpt}(st, rl, t)$**:** For all $\theta \in \mathsf{KUNodes}(st, rl, t)$: Fetch g_θ, pick $r_\theta \in \mathbb{Z}_p$, and compute $ku_\theta = (g_\theta \cdot \mathcal{H}(1t)^{r_\theta}, h^{r_\theta})$. Output $ku_t = (t, \{\theta, ku_\theta\}_{\theta \in \mathsf{KUNodes}(st, rl, t)})$.

$\mathsf{DKGen}(sk_{id}, ku_t)$**:** Find $\theta \in \mathsf{Path}(id) \cap \mathsf{KUNodes}(st, rl, t)$. Output $\perp$ if $\theta = \emptyset$, else pick $r'_\theta \in \mathbb{Z}_p$ and compute $sk'_3 = sk_\theta \cdot ku_{\theta,1} \cdot \mathcal{H}(1t)^{r'_\theta} = g^{d_3} \cdot g^{-\sigma'} \cdot \mathcal{H}(1t)^{r_\theta + r'_\theta}$ and $sk_{0,4} = ku_{\theta,2} \cdot h^{r'_\theta} = h^{r_\theta + r'_\theta}$, where $ku_{\theta,1}$ and $ku_{\theta,2}$ denote the first and the second elements of ku_θ. Set $sk'' = (sk'_1, sk'_2, sk'_3)$ and $sk'_0 = (sk_{0,1}, sk_{0,2}, sk_{0,3}, sk_{0,4})$. Here, $sk_{0,1}$, $sk_{0,2}$ and $sk_{0,3}$ denote the first, second and third elements of sk_0. Output $dk_{id,t} = (\mathcal{S}, t, sk'_0, \{sk_y\}_{y \in \mathcal{S}}, sk'')$.

$\mathsf{Enc}(m, \mathbb{A} = (\mathbb{M}, \pi), t)$**:**

Pick $s_1, s_2 \in \mathbb{Z}_p$. Compute $c_0 = (H_1^{s_1}, H_2^{s_2}, h^{s_1 + s_2}, \mathcal{H}(1t)^{s_1 + s_2})$. Suppose $\mathbb{M}$ has n_1 rows and n_2 columns. For $i = 1, ..., n_1$ and $\ell = 1, 2, 3$, compute

$c_{i,\ell} = \mathcal{H}(\pi(i)\ell 1)^{s_1} \cdot \mathcal{H}(\pi(i)\ell 2)^{s_2} \cdot \prod_{j=1}^{n_2} [\mathcal{H}(0j\ell 1)^{s_1} \cdot \mathcal{H}(0j\ell 2)^{s_2}]^{\mathbb{M}_{i,j}}$. Set $c_i = (c_{i,1}, c_{i,2}, c_{i,3})$. Compute $c' = T_1^{s_1} \cdot T_2^{s_2} \cdot m$. Output $c = (\mathbb{A}, t, c_0, c_1, \ldots, c_{n_1}, c')$.

$\mathsf{Dec}(dk_{id,t}, c)$**:** If $\mathcal{S}$ in $dk_{id,t}$ satisfies $\mathbb{A} = (\mathbb{M}, \pi)$ in c, then there exist constants $\{\gamma_i\}_{i \in I}$ that satisfy $\sum_{i \in I} \gamma_i \mathbb{M}_i = (1, 0, \ldots, 0)$. Compute

$$num = c' \cdot e\left(\prod_{i \in I} c_{i,1}^{\gamma_i}, sk_{0,1}\right) \cdot e\left(\prod_{i \in I} c_{i,2}^{\gamma_i}, sk_{0,2}\right) \cdot e\left(\prod_{i \in I} c_{i,3}^{\gamma_i}, sk_{0,3}\right) \cdot e\left(c_{0,4}, sk_{0,4}\right),$$

$$den = e\left(sk_1' \cdot \prod_{i \in I} sk_{\pi(i),1}^{\gamma_i}, c_{0,1}\right) \cdot e\left(sk_2' \cdot \prod_{i \in I} sk_{\pi(i),2}^{\gamma_i}, c_{0,2}\right) \cdot e\left(sk_3' \cdot \prod_{i \in I} sk_{\pi(i),3}^{\gamma_i}, c_{0,3}\right)$$

and output num/den. Here, $c_{0,1}$, $c_{0,2}$, $c_{0,3}$ and $c_{0,4}$ denote the first, second, third and forth elements of c_0.

$\mathsf{Rev}(rl, id, t)$**:** Output $rl \leftarrow rl \cup \{(id, t)\}$.

Theorem 1. *The proposed RABE is* IND-CPA *secure if the* DLIN *assumption is held in the random oracle model.*

We give the sketch of our security proof and details are omitted to conserve space. Our security proof is based on the proofs in RIBE [35] and FAME [2]. We can construct a simulation $\mathcal{B}$ to simulate the security game. $\mathcal{B}$ simulates the public parameters and oracles depending on FAME with two major differences. One is that $\mathcal{B}$ needs to guess the type of $\mathcal{A}$, where $\mathcal{A}$ can play non-revoked users who can get key-updating material in the challenge time, or revoked users who must be revoked before the challenge time. The other one is that $\mathcal{B}$ needs to guess the challenge time $t^* \in \mathcal{T}$. $\mathcal{B}$ can then simulate the IND-CPA game perfectly if guess correctly. Finally, $\mathcal{B}$ can forward the guess of $\mathcal{A}$ as the result to break DLIN assumption, which is also the hard assumption of FAME.

5.2 Proposed RPCH

We present the concrete construction of RPCH directly and the sketch of the security proof. The generic construction is similar to the previous RPCH [16,34] except that the underlying ABE is replaced to RABE, and additional algorithms Kupt, DKGen and Rev are used to manage user revocation. In our concrete construction, hybrid encryption is considered for the large-size chameleon trapdoor. In particular, an encoding method encodes a symmetric key to an RABE message, and a decoding method decodes the RABE message to a symmetric key. To simplicity, we represent the encoding and decoding method as $\mathsf{encode} : \{0,1\}^* \rightarrow \mathbb{G}_T$ and $\mathsf{encode}^{-1} : \mathbb{G}_T \rightarrow \{0,1\}^*$, respectively. Let $\mathcal{RABE} = \{\mathsf{Setup}, \mathsf{KGen}, \mathsf{KUpt}, \mathsf{DKGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rev}\}$ be an IND-CPA secure RABE. The constriction of RPCH is described as follows:

$\mathsf{Setup}(1^\kappa, n)$**:** It includes *parameter initialization*, *trapdoor selection*, *symmetric-key encryption initialization*:

1. Run $(mpk_{\mathcal{RABE}}, msk_{\mathcal{RABE}}, st, rl) \leftarrow \mathcal{RABE}.\mathsf{Setup}(1^\kappa, n)$. Pick $e_1 \geq N'$ with $N' = max_r\{N \in \mathbb{N} : (N, \cdot, \cdot, \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa; r)\}$.

2. Run $(N_1, p_1, q_1, \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa)$, choose hash functions $\mathcal{H}_1 : \{0,1\} \rightarrow \mathbb{Z}^*_{N_1}$ and $\mathcal{H}_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p \times \mathbb{Z}_p$. Compute d_1 s.t. $ed_1 \equiv 1 \bmod (p_1 - 1)(q_1 - 1)$.
3. Choose a symmetric-key encryption scheme $\mathcal{SE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$.

Output $mpk = (mpk_{\mathcal{RABE}}, N_1, e, \mathcal{H}_1, \mathcal{H}_3, \mathcal{SE}), msk = (d_1, msk_{\mathcal{RABE}}), st, rl$.

$\mathsf{KGen}(msk, st, id, \mathcal{S})$**:** Run $(sk_{\mathcal{RABE},id}, st) \leftarrow \mathcal{RABE}.\mathsf{KGen}(msk_{\mathcal{RABE}}, st, id, \mathcal{S})$. Output $sk_{id} = (d_1, sk_{\mathcal{RABE},id})$ and st.

$\mathsf{KUpt}(st, rl, t)$**:** Output ku_t by running $ku_t \leftarrow \mathcal{RABE}.\mathsf{KUpt}(st, rl, t)$.

$\mathsf{DKGen}(sk_{id}, ku_t)$**:** Run $dk_{\mathcal{RABE},id,t} \leftarrow \mathcal{RABE}.\mathsf{DKGen}(sk_{\mathcal{RABE},id}, ku_t)$. Output $dk_{id,t} = (d_1, dk_{\mathcal{RABE},id,t})$

$\mathsf{Rev}(rl, id, t)$**:** Output rl by running $rl \leftarrow \mathcal{RABE}.\mathsf{Rev}(rl, id, t)$.

$\mathsf{Hash}(m, \mathbb{A}, t)$**:** It includes *chameleon hash parameter initialization*, *trapdoor selection*, *trapdoor encapsulation*:

1. Run $(N_2, p_2, q_2, \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa)$, choose a hash function $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}^*_{N_2}$. Compute d_2, s.t. $ed_2 \equiv 1 \bmod (p_2 - 1)(q_2 - 1)$.
2. Choose $r_1 \in \mathbb{Z}^*_{N_1}$, $r_2 \in \mathbb{Z}^*_{N_2}$, compute $h_1 = \mathcal{H}_1(m, N_1, N_2) r_1^e \bmod N_1$ and $h_2 = \mathcal{H}_2(m, N_1, N_2) r_2^e \bmod N_2$. Set $h' = (h_1, h_2)$ and $r' = (r_1, r_2)$.
3. Choose $r \in \{0,1\}^\kappa$, $k \leftarrow \mathcal{SE}.\mathsf{KGen}(1^\kappa)$, $c_{\mathcal{SE}} \leftarrow \mathcal{SE}.\mathsf{Enc}(k, d_2)$, $K \leftarrow \mathsf{encode}(k, r)$. Run $c_{\mathcal{RABE}} \leftarrow \mathcal{RABE}.\mathsf{Enc}(K, \mathbb{A}, t)$ with the randomnesses $(s_1, s_2) \leftarrow \mathcal{H}_3(r, \mathbb{A}, t)$.

Output $h = (h', N_2, \mathcal{H}_2, c_{\mathcal{RABE}}, c_{\mathcal{SE}})$ and $r = r'$.

$\mathsf{Verify}(m, h, r)$**:** Verify $r_1 \in \mathbb{Z}^*_{N_1}$, $r_2 \in \mathbb{Z}^*_{N_2}$ and whether $h_1 = \mathcal{H}_1(m, N_1, N_2) r_1^e \bmod N_1$ and $h_2 = \mathcal{H}_2(m, N_1, N_2) r_2^e \bmod N_2$. If all checks hold, return 1 and 0 otherwise.

$\mathsf{Adapt}(dk_{id,t}, m, m', h, r)$**:** It includes *chameleon hash verification*, *symmetric key revelation*, *ciphertext verification and trapdoor revelation*, *message adaptation*, *adapted chameleon hash verification*:

1. Run $b \leftarrow \mathsf{Verify}(m, h, r)$ and return $\perp$ if $b = 0$.
2. Run $K' \leftarrow \mathcal{RABE}.\mathsf{Dec}(dk_{\mathcal{RABE},id,t}, c_{\mathcal{RABE}})$ and set $(k', r') \leftarrow \mathsf{encode}^{-1}(K')$.
3. Run $c'_{\mathcal{RABE}} \leftarrow \mathcal{RABE}.\mathsf{Enc}(K', \mathbb{A}, t)$ with the randomnesses $(s'_1, s'_2) \leftarrow \mathcal{H}_3(r', \mathbb{A}, t)$. Output $\perp$ if $c_{\mathcal{RABE}} \neq c'_{\mathcal{RABE}}$. Otherwise, compute $d'_2 \leftarrow \mathcal{SE}.\mathsf{Dec}(k', \mathcal{C}_{\mathcal{SE}})$ and return $\perp$ if $d_2 = \perp$.
4. Let $x_1 = \mathcal{H}_1(m, N_1, N_2)$, $x'_1 = \mathcal{H}_1(m', N_1, N_2)$, $y_1 = x_1 r_1^e \bmod N_1$ as well as $x_2 = \mathcal{H}_2(m, N_1, N_2)$, $x'_2 = \mathcal{H}_1(m', N_1, N_2)$, $y_2 = x_2 r_2^e \bmod N_2$. Compute $r'_1 \leftarrow (y_1 (x'^{-1}_1))^{d_1} \bmod N_1$ and $r'_2 \leftarrow (y_2 (x'^{-1}_2))^{d_2} \bmod N_2$.
5. Return $\perp$ if $h_1 \neq \mathcal{H}_1(m', N_1, N_2) r'^e_1 \bmod N_1$ or $h_2 \neq \mathcal{H}_2(m', N_1, N_2) r'^e_2 \bmod N_2$.

Output $r' = (r'_1, r'_2)$.

Remark. In the above constriction, we apply the well-known Fujisaki-Okamoto transform [20] to our proposed RABE. Basically, the hash algorithm takes the randomness $(s_1, s_2) \leftarrow \mathcal{H}_3(r, \mathbb{A}, t)$ based on a sufficiently large randomly sampled bitstring r to encrypt the ephemeral trapdoor and r. The adaptation algorithm applies the original decryption algorithm to receive the ephemeral trapdoor and r'. Then, it re-encrypts the ephemeral trapdoor and r' based on the randomness $(s'_1, s'_2) \leftarrow \mathcal{H}_3(r', \mathbb{A}, t)$ to validate the ciphertext. If the re-encrypted result and the original ciphertext are different, it outputs $\perp$.

Theorem 2. *The proposed RPCH is fully indistinguishable, insider collision-resistant, unique, and correct if the underlying RABE is* IND-CPA *and correct, and the underlying CHET is fully indistinguishable, strongly private collision-resistant, unique, and correct.*

All properties, but insider collision-resistance, can be proved by following the methods used in [34]. In particular, the security of fully indistinguishability and uniqueness can be reduced to the security of CHET. Only insider collision-resistance is based on strongly private collision-resistance of the underlying CHET and the security of the underlying ABE (which is RABE in our proposed RPCH). Our security proof of insider collision-resistance is a sequence of games that is similar to [34], and the security of insider collision-resistance can be reduced to the security of IND-CPA RABE and strongly private collision-resistant CHET.

5.3 Applications

We show two applications of RPCH for mutable blockchain and sanitizable signature. On a high-level, the blockchain remains intact even if a certain policy-based mutable transaction has been rewritten and a signature holds integrity when the admissible blocks of the signed message have been altered.

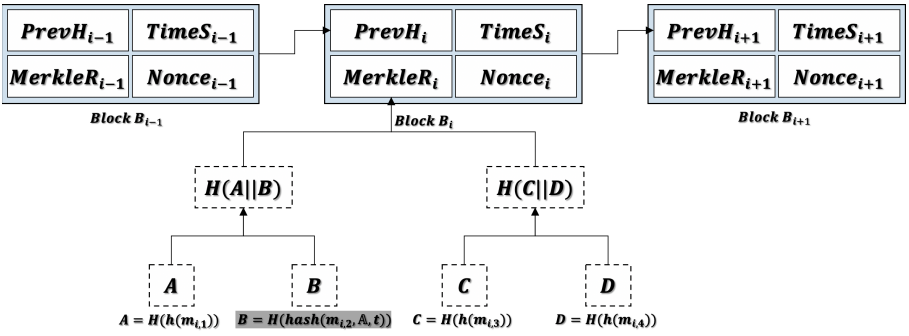

Fig. 3. RPCH for mutable blockchain

Figure 3 presents the application of RPCH for transaction-level blockchain rewriting. A block b_i has a Merkle root that accumulates four transactions: $m_{i,1}$, $m_{i,2}$, $m_{i,3}$ and $m_{i,4}$. $m_{i,1}$, $m_{i,3}$ and $m_{i,4}$ are immutable transactions hashed by the traditional collision-resistant hash function h. $m_{i,2}$ is a mutable transaction associated with an access structure $\mathbb{A}$ and a timestamp t. When $m_{i,2}$ needs to be altered to $m'_{i,2}$, a transaction modifier with PCH-based decryption key associated with the attributes $\mathcal{S}$ and timestamp t' satisfying $\mathcal{S} \models \mathbb{A}$ and $t' = t$ can compute a valid chameleon randomness r without modifying its hash value, hence, Merkle root is never modified. The transaction modifier then broadcasts

$(m'_{i,2}, r')$ to the blockchain network. All participants verify the correctness of the new randomness and update their local copy of the blockchain with the new message and randomness pair $(m'_{i,2}, r')$.

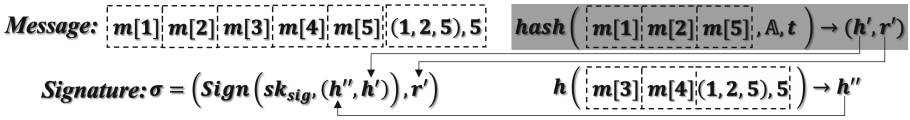

Fig. 4. RPCH for sanitizable signature

Figure 4 is an application of RPCH for sanitizable signatures. A message m has five blocks $m[1]$, $m[2]$, $m[3]$, $m[4]$ and $m[5]$, where $m[1]$, $m[2]$ and $m[5]$ are admissible blocks that can be altered. The admissible blocks are hashed via RPCH hash function $hash$ associated with an access structure $\mathbb{A}$ and a timestamp t. The non-admissible blocks and the information of admissible blocks $((1, 2, 5), 5)$ are hashed by the traditional collision-resistant hash function h, where information of admissible blocks $((1, 2, 5), 5)$ indicates that the 1st, 2nd and 5th blocks can be altered and the total number of blocks in m is 5. When the admissible blocks need to be altered, a sanitizer with PCH-based decryption key associated with attributes $\mathcal{S}$ and timestamp t' satisfying $\mathcal{S} \models \mathbb{A}$ and $t' = t$ can compute a valid chameleon randomness without rewriting its signature. The sanitizer then broadcasts the randomness and updates admissible blocks. The verifier then can process the verification of the updated signature.

Remark. We only consider backward secrecy in the term of revocability, where the modifier cannot alter any message after being revoked. To achieve forward secrecy, ciphertext delegation [33] can be used, where a (semi-)trusted third party updates the ciphertext periodically. However, ciphertext delegation is impractical in the application scenarios of mutable blockchain and sanitizable signature. It is hard to find a (semi-)trusted third party in the untrusted environment, and even enable them to process ciphertext update periodically.[5] The signature is hard to trace back and update once it has been released. To remedy this shortcoming, we consider processing user revocation periodically and require the mutable transaction owner and the signer of the sanitizable signature to update the timestamp in each RPCH hash in the same frequency. Hence, the compromised long-term secret key only affects several periods, and the chameleon hash under those periods may be updated before key compromization. We leave the PRCH with forward secrecy, a modifier cannot process rewriting to the hashed object generated before being revoked, without ciphertext delegation or a (semi-)trusted third party as interesting future work.

[5] Outsourced decryption has also not taken into consideration due to a (semi-)trusted third party is needed and processes outsourced decryption.

6 Performance Analysis

In this section, we give detailed comparisons of indirect RABE and PCH. Then, we implement PCH [16] and ours, where PCH [36] only considers accountability, and no concrete PCH construction is provided in [34].

Table 1. Comparison of RABE schemes

	Scheme	Group-order	Assumption	DKE-resistant	Security
SSW12 [33]	ABE [25]	*Composite*	SDP	✗	*adaptive*
CDLQ16 [15]	ABE [32]	*Prime*	*q-type*	✗	*selective*
QZZC17 [31]	ABE [32]	*Prime*	*q-type*	✓	*selective*
XYMD18 [38]	ABE [21]	*Prime*	DBDH	✗	*selective*
XYM19 [37]	ABE [32]	*Prime*	*q-type*	✓	*selective*
XYML19 [39]	ABE [32]	*Prime*	*q-type*	✓	*selective*
Ours	FAME [2]	*Prime*	DLIN	✓	*adaptive*

"***Scheme***" *means that on which ABE the RABE is based.* "***Assumption***" *means that on which assumption the RABE is based.* "***DKE-resistance***" *means that whether the RABE is decryption key exposure resistant or not.* "DBDH" *means Decisional Bilinear Diffie-Hellman assumption.*

Table 2. Comparison of PCH schemes

	Scheme	Revocability	Application
DSSS19 [16]	FAME [2] + RSA-Based CH [14]	✗	*Blockchain rewriting*
TLL+20 [36]	ABET [36] + DL-Based CH [14]	✗	*Blockchain rewriting*
SS20 [34]	FAME [2] + RSA-Based CH [14]	✗	*Sanitizable signature*
Ours	RABE + RSA-based CH [14]	✓	*Both*

"***Scheme***" *means that on which cryptographic primitive the PCH is based.* "***Revocability***" *means that whether the PCH supports rewriting privilege revocation. "CH" is short for chameleon hash.*

Detailed Comparisons. Table 1 compares the classical and recent indirect RABE schemes [15,31,33,37–39] and ours. The previous RABE schemes, as in Table 1, are selectively secure except SSW12 [33], where SSW12 is adaptively secure under Subgroup Decision Problem (SDP) on composite-order groups. Our RABE is adaptively secure under a standard assumption on prime-order groups.

Table 2 compares the recently PCH schemes [16,34,36] and ours. All of them are derived from FAME [2], where TLL+20 [36] introduced attribute-based encryption scheme with traceability (ABET) based on FAME and HIBE [11], and our RABE based on FAME and RIBE. To the best of our knowledge, our RPCH is the first PCH with revocability and can be integrated into blockchain and sanitizable signatures.

Implementation and Evaluation. We implement PCH [16] and our RPCH in Java 8 using the PBC Library. We use MNT224 curve for pairings because it is the best Type-III curve in PBC and provides 96-bit security level [2]. Our experimental simulation was measured on a personal laptop with a 3.0 GHz AMD Ryzen R5 4600 processor and 16 GB RAM. To simplicity, we define the number of system users $n = 2^{10}$ in our implementation.

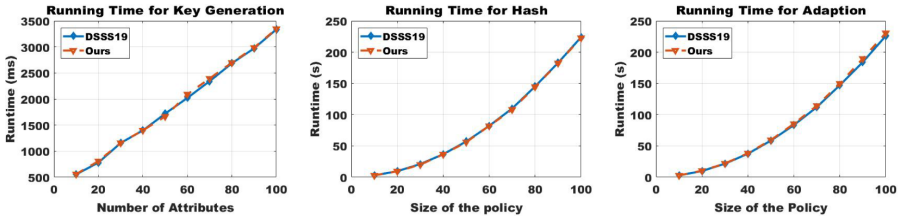

Fig. 5. Performance comparison of PCH schemes

Figure 5 compares the running time of key generation, hash and adaption for PCH schemes we consider. Our solution with a negligible performance overhead compared to DSSS19. Compared to DSSS19, our RPCH takes additional $\log_2 n - 1$ multiplications (around 0.11 ms) for key generation, additional one exponentiation (about 3.83 ms) for hash, and additional one multiplication and pairing (around 21.19 ms) for adaption.

7 Conclusion

In this work, we proposed the notion of revocable policy-based chameleon hash (RPCH) and illustrated its applications in mutable blockchain and sanitizable signature. Our RPCH allows a trusted party to effectively revoke the rewriting privileges of a chameleon trapdoor holder. We gave a practical instantiation by introducing adaptively secure revocable attribute-based encryption under a standard assumption on prime-order groups. The future work could be investigating the usability and security of RPCH, e.g., outsourced decryption and forward secrecy without any (semi-)trusted third party.

Acknowledgments. This work is supported in part by AXA Research Fund, the National Natural Science Foundation of China (Grant Nos. 62102090, 62032005, 61972094), the young talent promotion project of Fujian Science and Technology Association, and Science Foundation of Fujian Provincial Science and Technology Agency (2020J02016).

References

1. General data protection regulation. https://gdpr-info.eu/

2. Agrawal, S., Chase, M.: FAME: fast attribute-based message encryption. In: CCS, pp. 665–682 (2017)
3. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: ESORICS, vol. 3679, pp. 159–177 (2005)
4. Ateniese, G., Magri, B., Venturi, D., Andrade, E.R.: Redactable blockchain - or - rewriting history in bitcoin and friends. In: EuroS&P, pp. 111–126 (2017)
5. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: ASIACRYPT, pp. 591–623 (2016)
6. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In: IMA, pp. 278–300 (2009)
7. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE S&P, pp. 321–334 (2007)
8. Bilzhause, A., Pöhls, H.C., Samelin, K.: Position paper: the past, present, and future of sanitizable and redactable signatures. In: ARES, pp. 87:1–87:9 (2017)
9. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: CCS, pp. 417–426 (2008)
10. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: EUROCRYPT, vol. 3027, pp. 223–238 (2004)
11. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: EUROCRYPT, vol. 3494, pp. 440–456 (2005)
12. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
13. Bultel, X., Lafourcade, P., Lai, R.W.F., Malavolta, G., Schröder, D., Thyagarajan, S.A.K.: Efficient invisible and unlinkable sanitizable signatures. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 159–189. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_6
14. Camenisch, J., Derler, D., Krenn, S., Pöhls, H.C., Samelin, K., Slamanig, D.: Chameleon-hashes with ephemeral trapdoors. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 152–182. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54388-7_6
15. Cui, H., Deng, R.H., Li, Y., Qin, B.: Server-aided revocable attribute-based encryption. In: ESORICS, vol. 9879, pp. 570–587 (2016)
16. Derler, D., Samelin, K., Slamanig, D., Striecks, C.: Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based. In: NDSS (2019)
17. Deuber, D., Magri, B., Thyagarajan, S.A.K.: Redactable blockchain in the permissionless setting. In: IEEE SP, pp. 124–138 (2019)
18. Fischlin, M., Harasser, P.: Invisible sanitizable signatures and public-key encryption are equivalent. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 202–220. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_11
19. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 301–330. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_12
20. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34

21. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS, pp. 89–98 (2006)
22. Guillevic, A.: Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 357–372. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_22
23. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: IEEE S&P, p. 27 (2004)
24. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS (2000)
25. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT, vol. 6110, pp. 62–91 (2010)
26. Liu, J.K., Yuen, T.H., Zhang, P., Liang, K.: Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 516–534. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_27
27. Matzutt, R., et al.: A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 420–438. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_23
28. Matzutt, R., Hohlfeld, O., Henze, M., Rawiel, R., Ziegeldorf, J.H., Wehrle, K.: POSTER: i don't want that content! on the risks of exploiting bitcoin's blockchain as a content store. In: CCS, pp. 1769–1771 (2016)
29. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_3
30. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_22
31. Qin, B., Zhao, Q., Zheng, D., Cui, H.: Server-aided revocable attribute-based encryption resilient to decryption key exposure. In: Capkun, S., Chow, S.S.M. (eds.) CANS 2017. LNCS, vol. 11261, pp. 504–514. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02641-7_25
32. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS, pp. 463–474 (2013)
33. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_13
34. Samelin, K., Slamanig, D.: Policy-based sanitizable signatures. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 538–563. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-40186-3_23
35. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_14
36. Tian, Y., Li, N., Li, Y., Szalachowski, P., Zhou, J.: Policy-based chameleon hash for blockchain rewriting with black-box accountability. In: ACSAC, pp. 813–828 (2020)

37. Xu, S., Yang, G., Mu, Y.: Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation. Inf. Sci. **479**, 116–134 (2019)
38. Xu, S., Yang, G., Mu, Y., Deng, R.H.: Secure fine-grained access control and data sharing for dynamic groups in the cloud. IEEE Trans. Inf. Forensics Secur. **13**(8), 2101–2113 (2018)
39. Xu, S., Yang, G., Mu, Y., Liu, X.: A secure IoT cloud storage system with fine-grained access control and decryption key exposure resistance. Future Gener. Comput. Syst. **97**, 284–294 (2019)
40. Xu, S., Zhang, Y., Li, Y., Liu, X., Yang, G.: Generic construction of ElGamal-type attribute-based encryption schemes with revocability and dual-policy. In: SecureComm, vol. 305, pp. 184–204 (2019)
41. Yang, Y., Liu, J.K., Liang, K., Choo, K.-K.R., Zhou, J.: Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 146–166. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_8