Secure Redactable Blockchain with Dynamic Support

Di Zhang, Junqing Le, Xinyu Lei, Tao Xiang, *Senior Member, IEEE,* and Xiaofeng Liao, *Fellow, IEEE*

Abstract—Blockchain is extensively applied to many fields as an immutable distributed ledger. However, the immutability contradicts regulations such as the GDPR ruling "the right to be forgotten" of data. Besides, numerous emerging blockchain-based applications call for elastic data management. To erase some data, redactable blockchains are proposed for breaking the immutability in a controlled way. Unfortunately, the prior solutions may suffer from poor security and centralized control of the redaction privilege. They cannot support dynamic nodes, where the departure of participators will result in a single point of failure. This paper proposes a novel dynamic and decentralized attribute-based chameleon hash (DACH) to make blockchain history mutable, achieving a securely and dynamically redactable blockchain (SDR-chain) in a decentralized setting. We first propose the formal definition, security models, and concrete construction of our DACH. Meanwhile, we design a delegation algorithm of DACH to support a dynamically changing committee, where participators can freely and securely leave and join the network. Then, the transactions of the SDR-chain are redacted by computing DACH collisions. The security is analyzed in the random oracle model. Finally, theoretical analysis and experimental evaluation demonstrate that our SDR-chain is superior to the prior solutions in terms of security and functionality.

Index Terms—Blockchain, GDPR, Chameleon hash, Attribute-based encryption, Delegation.

1 Introduction

Blockchain [1], regarded as an ingenious and revolutionary technique, has captured the attention of academia, government, and industry. It has been widely applied to various applications, such as supply chain [2] healthcare [3], and smart grid [4]. A recent study shows that the worldwide spending on blockchain solutions is estimated to reach 4.1 billion dollars by 2020 and will continue to grow to an estimated 19 billion dollars by 2024 [5].

A blockchain is an increasing list of blocks that links together through a hash chain. Each block contains a previous block hash as a reference to link the previous block. The block records a set of transaction data (such as cryptocurrency [6] and smart contract [7]) that are usually hashed and organized in a Merkle tree (MT). Blocks and transactions are broadcast and verified in a peer-to-peer (P2P) communication network. Nodes of the P2P network eventually agree on a unified view of the blockchain (i.e., consistency). In addition, the traditional blockchain is resistant to modification of its data (i.e., immutability), which facilitates the data auditing process.

Recently, numerous researchers argue that it is significant to redact blockchain in strictly specific circumstances [8–18]. The two main reasons are explained as follows. *(i) The requirement for redacting data in blockchains has been stimulated by regulations.* The European General Data Protection Regulation (GDPR) [19] has ruled that data should have a "right to be forgotten". Users should be able to erase their personal data and copies anytime in the network. At present, blockchians have been used to store the personal data for malicious purposes immutably. For instance, Bitcoin blockchain has stored cross-site scripting attack code [20], and data referred to child pornography [21]. Therefore, erasing data from blockchains evolves an important and imperative requirement. *(ii) An immutable ledger is not appropriate for all emerging blockchain-based applications, and a certain degree of flexibility for read and write operations is called for.* For example, erasing stale data from blockchain to save the storage space of edge nodes in the IoT [22, 23]. Becasuse the ever-joining devices and the ever-generating data cause edge nodes to face the risk of running out of storage space with the growing size of the blockchain. Besides, some blockchain-based applications such as storage [24, 25], sharing [26], and authentication [27] may need to correct objects (e.g. updating/amending contract) in time to provide superior quality services. To sum up, it is imperative to develop techniques to redact data in blockchain under some strict constraints.

The previous solutions roughly fall into non-cryptography-based [8–11] and cryptography-based [12–17]. The main idea of non-cryptography-based solutions is that a new version block/transaction is generated to replace the original one by voting. These solutions [8–11] have three major drawbacks. (1) They have a huge cost for every redaction because of the generation and broadcast of the new version block/transaction. (2) They cannot handle high frequent redaction requests and have low scalability because they have to spend a long time agreeing on each new version to ensure system security. (3) They do not satisfy the historic security because the historic versions can be reconstructed easily. Besides, the solutions (e.g., [12–17]) are based

- *Di Zhang, Junqing Le, Tao Xiang, and Xiaofeng Liao are with Key Laboratory of Dependable Services Computing in Cyber Physical Society-Ministry of Education and College of Computer Science, Chongqing University, Chongqing 400044, China.*
  E-mail: {zhangdiii, lejunqing}@163.com, {txiang, xfliao}@cqu.edu.cn
- *Xinyu Lei is with the Department of Computer Science, Michigan Technological University, Houghton, MI 49931, USA.*
  E-mail: xinyulei@mtu.edu
  (Corresponding authors: Junqing Le and Xiaofeng Liao)

TABLE 1
Advantages of SDR-chain over the Prior Solutions

Performance \ Schemes	[8–11]	[12]	[13–16]	[17]	SDR-chain
Fine-Grained Control	○	○	●	●	●
Decentralized Redaction	◐	●	○	●	●
Dynamic Support	●	●	○	○	●
Scalability	◐	●	●	●	●
Historic Security	○	●	●	●	●
Backward/Forward	−	○	○	○	●

* ●: Yes. ○: No. ◐: Partially. −: not applicable.

on cryptographic primitives, where the block/transaction is modified by computing chameleon hash (CH) [28] collisions. These solutions also suffer from some limitations. Specifically, literature [12] achieves redactions in a coarse-grained way. In [13–16], redactions are controlled by a fully trusted central authority. Literatures [12–17] cannot satisfy the backward/forward security. Literatures [13–17] cannot satisfy the dynamic support. They will suffer from the single point of failure when any one of authorities controlling redactions is offline.

We aim to propose a novel hash function to make blockchain redactable, and the redactable blockchain should satisfy the following design goals (as shown in TABLE 1). (1) **Fine-Grained Control.** It should control what data can be redacted and who can perform redaction in a fine-grained way. (2) **Decentralized Redaction.** It should have high degree of decentralization, i.e., redaction is controlled by multiple semi-trusted nodes instead of a single full trusted node. (3) **Dynamic support.** It should support that nodes participating in redaction can free join and leave the blockchain P2P network. (4) **Scalability.** It should support large scale redaction requests. (5) **Security.** It should satisfy the following two security requirements. (a) *Historic security.* After redaction, the historic original data cannot be reconstructed from the redacted data. (b) *Backward/forward security.* After any participator leaves, its keys cannot be used for future redaction. For any newly joining participator, its keys should be able to redact the previous transactions.

We start by developing a customized chameleon hash (i.e., dynamic and decentralized attribute-based chameleon hash, denoted as DACH). Then, we apply DACH to make blockchain redactable, achieving a securely and dynamically redactable blockchain (SDR-chain). There are two major technical challenges. *(i) How to achieve a decentralized control of the redaction operation*. To address this challenge, we design a decentralized method to manage the secret key of redactions by taking advantage of a decentralized attribute-based encryption (DCP-ABE) [29] scheme. In DCP-ABE, the policy encoded over attributes determines who can redact the transaction, which controls the redaction privilege and prevents abuse of redaction operations. Illegitimate users are hard to perform malicious redactions to break the security and stability of blockchain ecosystems. Besides, the trapdoor key is managed by a group of attribute authorities (AAs) based on DCP-ABE. The redaction operation is not controlled by a single full trusted center, which achieves a high degree of decentralization. *(ii) How to achieve dynamic support*. Once any one of participators (i.e., AAs) leaves the network, redaction process will terminate i.e., the single point of failure. To address this challenge, we employ the concept of re-encryption to design a delegation algorithm of DACH. Based on the delegation algorithm, the offline AA can securely delegate its role to a new AA. Furthermore, the new AA can issue valid keys to help redactors modify transactions and make the keys of the offline AA invalid for future redaction. Thus, DACH can support a dynamically changing redaction committee.

TABLE 1 shows the advantages of our SDR-chain over the prior solutions. SDR-chain considers more adversary models than other solutions. In summary, our contributions are three-fold.

- We propose a novel chameleon hash primitive DACH for achieving controlled and decentralized redaction in blockchain. Based on DACH, the redaction operation can be performed in a secure and high degree decentralized way.
- We design a delegation algorithm of DACH to support a dynamically changing redaction committee. The delegation algorithm satisfies the backward/forward security.
- We theoretically prove the security of DACH in the random oracle model, and then we apply the DACH to make blockchain such as Bitcoin redactable. From the analyses, we achieve a redactable blockchain with security, dynamic support, scalability, and fine-grained control in the decentralized setting, achieving the design goals.

The remaining part of this paper is organized as follows. The following Section 2 defines the preliminary and Section 3 provides an overview of our SDR-chain. In Section 4, we design our DACH function including formal definition, security model, and concrete construction. Then, we present our DACH-based SDR-chain in Section 5 and analyze the correctness, security, and design goals in Section 6. After that, Section 7 evaluates performance theoretically and experimentally. Section 8 reviews some related work. Finally, Section 9 gives a conclusion for this paper.

2 PRELIMINARY

In this section, we review some notations and cryptographic building blocks, which are the ingredients to construct our DACH function.

2.1 Notations

Let $\kappa \in \mathbb{N}$ be the security parameter and it is assumed to be an implicit input of algorithms. Let $\mathcal{M}$ be a message space and message $m \in \mathcal{M}$. For a set X, $x \xleftarrow{R} X$ denotes the operation of picking an element x from X uniformly at random, and $|x|$ represents the length of x. Let $w \leftarrow \mathcal{A}(x)$ denote that a algorithm $\mathcal{A}$ is run with the input x and then it outputs w. By setting $w \leftarrow \mathcal{A}^{\mathcal{O}}(x)$, it is indicated that $\mathcal{A}$ has black-box access to oracles $\mathcal{O}$. All algorithms in this paper are assumed to run in probabilistic polynomial time (PPT) unless otherwise specified, i.e., their running time can be bounded by a polynomial $poly(|x|)$. Let $\mathsf{Pr}[\mathbf{E}]$ be the probability that an event $\mathbf{E}$ occurs. A function f is negligible if $\forall c\ \exists N\ \forall \kappa > N : f(\kappa) < \kappa^{-c}$. Let $(N, p, q, e, d) \leftarrow \mathsf{RSAKGen}(1^{\kappa})$ be

a key generation algorithm of RSA. On input parameter κ, the algorithm RSAKGen outputs two distinct large primes p and q, a modulus $N = pq$, a constant e, and a modular multiplicative inverse d of e, i.e., $ed \equiv 1 \bmod \varphi(N)$, where $\varphi(\cdot)$ is Euler's totient function. Let $(p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g) \leftarrow \mathcal{G}(1^\kappa)$ be a pairing group generation algorithm. On input parameter κ, the algorithm $\mathcal{G}$ outputs two multiplicative cyclic groups $\mathbb{G}_T$ and $\mathbb{G} =< g >$ of prime order p with a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

2.2 Cryptographic Building Blocks

Definition 1 (LSSS). *A linear secret sharing scheme (LSSS) [30] over a finite field $\mathbb{Z}_p$ satisfies that:*

1) *The shares for each party form a vector over $\mathbb{Z}_p$.*
2) *There exists a share-generating matrix $A_{k\times l}$ encoding a monotone access structure $\mathbb{A}$ over attributes subset $\mathbb{U}$. For all $x \in [k]$, the xth row of A is labeled by $\rho(x)$, where the function ρ maps row numbers to attributes, i.e., $\rho : [k] \rightarrow \mathbb{U}$. During the generation of shares, a column vector $v = (s, a_2, ..., a_l)$ is generated, where $s \in \mathbb{Z}_p$ is the secret to be shared and $a_2, ..., a_l \xleftarrow{R} \mathbb{Z}_p$. Then the vector of k shares of the secret s is computed by $\lambda = A \cdot v \in \mathbb{Z}_p^k$. The share λ_x belongs to the attribute $\rho(x)$.*

Assume that the vector $(1, 0, .., 0)$ is in the span of rows of A labeled by $\rho(x)$, denoted as $(1, 0, .., 0) \in span(A_x)$. There exist constants $\{C_x \in \mathbb{Z}_p\}$ such that $\sum_x C_x \lambda_x = s$. The pair (A, ρ) describes the policy of the access structure $\mathbb{A}$. Lewko and Waters [31] have described a general algorithm to convert the Boolean formula including AND and OR operators into the LSSS matrix.

Definition 2 (DCP-ABE). *A decentralized ciphertext-policy attribute-based encryption (DCP-ABE) scheme [29] is a tuple of five PPT algorithms (*GSetup, AASetup, Enc, KGen, Dec*):*

- GSetup(1^κ). *On input a security parameter κ, the global setup algorithm* GSetup *outputs the public parameter pp.*
- AASetup(pp). *On input the public parameter pp, the authority setup algorithm* AASetup *outputs a key pair (pk, sk) for an attribute i.*
- Enc$(m, (A, \rho), \{pk\})$. *On input a message m, an access matrix (A, ρ), and a set of public keys $\{pk\}$, the encryption algorithm* Enc *outputs a ciphertext ct.*
- KGen(GID, i, sk). *On input a global identity GID, the attribute i, and the secret key sk, the key generation algorithm* KGen *outputs a key $SK_{i,GID}$ for this attribute.*
- Dec$(\{SK_{i,GID}\}, ct)$. *On input a collection of keys $\{SK_{i,GID}\}$ corresponding to the attributes of the same identity and the ciphertext ct, the decryption algorithm* Dec *outputs a message m if the collection of attributes satisfies the access matrix. Otherwise, this algorithm returns $\perp$.*

Lewko and Waters have proved that the DCP-ABE scheme with prime order groups in the full version of [31] is IND-CCA2 secure for any PPT algorithm $\mathcal{A}$. The advantage of $\mathcal{A}$ winning the IND-CCA2 experiment is negligible, i.e., $\mathbf{Adv}^{\mathsf{IND-CCA2}}_{\mathcal{A},\mathcal{DCP}-\mathcal{ABE}} = |\Pr[b^* = b] - \frac{1}{2}| \leq f(\kappa)$.

Definition 3 (CHET). *A chameleon hash with ephemeral trapdoors (CHET) scheme [32] consists of five PPT algorithms (*PPGen, CHKey, CHash, CHVer, CHAdapt*) such that:*

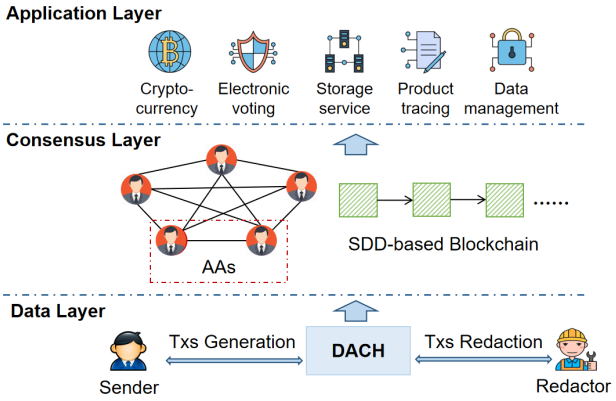

Fig. 1: The system model. Based on a critical component DACH, SDR-chain supports transactions (Txs) redaction and provides elastic data management for various emerging applications.

- PPGen(1^κ). *On input the security parameter κ, the algorithm* PPGen *outputs the public parameter pp.*
- CHKey(pp). *On input the public parameter pp, the algorithm* CHKey *outputs a long-term key pairs (pk, sk).*
- CHash(pk, m). *On input a message m and the public key pk, the algorithm* CHash *outputs a hash h, randomness r, and ephemeral trapdoor etd.*
- CHVer(pk, m, h, r). *On input a public key pk, message m, hash h and randomness r, the algorithm* CHVer *outputs a bit $b \in \{0, 1\}$ indicating the validity of hash h.*
- CHAdapt(sk, etd, m, h, r, m'). *On input the secret key sk, ephemeral trapdoor etd, message m, hash h, randomness r and a new message $m' \in \mathcal{M}$, the algorithm* CHAdapt *outputs a new randomness r'.*

It has been proved that CHET in [32] is secure under the RSA-like assumption, i.e., CHET is indistinguishable, publicly collision-resistant, and privately collision-resistant. The detailed analysis has been presented in Appendix E.1 and F.2 of the full version.

3 Overview

In this section, we present an overview of our DACH-based SDR-chain, including system model, thread model, and main phases.

3.1 System Model

The system model, as shown in Fig. 1, is constructed by attribute authorities (AAs), sender, and redactor. Senders/redactors generate/redact transactions based on our DACH function with the help of a group of AAs in the data layer. Then, the generated/redacted transactions are broadcast and verified in the consensus layer. Finally, DACH-based SDR-chain provides elastic data management for various applications.

1) *Attribute Authorities (AAs).* AAs are selected from nodes in the P2P network. They are semi-honest entities that honestly generate parameters to help sender/redactor generate/ redact transactions but they may try to recover the keys to get redaction privilege.
2) *Sender.* It is an honest entity that generates transactions with a valid access policy in the network.

3) *Redactor.* It is an entity that redacts transactions in the network. It may try to make a profit by requesting malicious redactions.

3.2 Threat Models

We consider the PPT adversaries who may try to reconstruct the keys of DACH function for redactions. Their goals are to abuse the redaction such as faking redactions by launching attacks. Assume that the point-to-point communication channel is secure and the PPT adversaries can not break the standard cryptographic primitives and assumptions. According to the ability of adversaries, we consider three types of threat models as follows.

1) *Inside adversary*. It can be a semi-trusted AA owning one of the secret keys corresponding to attributes of the redaction policy. It honestly executes processes but it would take advantage of its abilities to recover the trapdoor key used to redact transactions. A majority of AAs may collude with each other. Besides, the goal of the colluding AAs is to modify transactions while they do not know the trapdoor key. The insider adversary can access oracles DKGen and DAdapt of the DACH function.
2) *Outside adversary*. It can be a redactor owning some attributes, which cannot satisfy the redaction policy. It can access the oracle DAdapt and tries to recover the trapdoor key to modify transactions. Besides, it may request a malicious redaction to launch attacks such as the collusion attack and the double-spending attack.
3) *Adaptive adversary*. It can be both an insider and outsider adversary adaptively. It can adaptively access oracles DKGen and DAdapt and has some attributes which do not satisfy the policy.

3.3 Main Phases

The proposed SDR-chain includes four phases as follows.

1) *AAs selection*. AAs are selected from nodes for the parameters generation and keys distribution.
2) *Redactable transaction generation*. Senders generate the redactable transactions based on DACH. The redactable transactions are recorded in blocks and then blocks build the SDR-chain after broadcast and verification.
3) *Transaction redaction*. A redactor is allowed to modify the transaction if the redactor can efficiently compute a DACH collision for the modified transaction.
4) *Delegation*. Any leaving AA can securely delegate its role to other nodes to ensure redaction execution.

4 Designed DACH Function

The DACH function is the critical component of the proposed SDR-chain. In this section, we introduce the formal definition, security model, and concrete construction of our DACH function.

4.1 Formal Definition

Definition 4 (DACH). *A dynamic and decentralized attribute-based chameleon hash (DACH) function with message space $\mathcal{M}$*

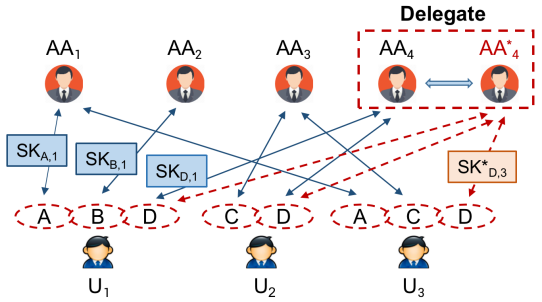

Fig. 2: The process of recovering the trapdoor key. Users obtain the secret keys corresponding to attributes from a group of AAs. Any AA can delegate its role to other nodes, as an example of AA_4 and AA_4^* shows.

consists of seven PPT algorithms $\Pi_{\mathsf{DACH}} = (\mathsf{PPGen}, \mathsf{AASetup}, \mathsf{DKGen}, \mathsf{DHash}, \mathsf{DVer}, \mathsf{DAdapt}, \mathsf{Delegate})$ *such that:*

$\mathsf{PPGen}(1^\kappa)$. *On input the security parameter* κ*, the algorithm* PPGen *outputs the public parameter* pp.

$\mathsf{AASetup}(pp)$. *On input the parameter* pp*, the algorithm* AASetup *outputs keys* $pk \leftarrow (mpk, spk)$ *and* $sk \leftarrow (msk, ssk)$.

$\mathsf{DKGen}(GID, sk, \mathbb{S})$. *On input a global identity* GID*, secret key* sk*, and attributes* $\mathbb{S}$*, the algorithm* DKGen *outputs a secret key* SK_i *for the attribute* $i \in \mathbb{S}$ *and* $SK_{\mathbb{S}} = \{SK_i | i \in \mathbb{S}\}$.

$\mathsf{DHash}(pk, m, \mathbb{A})$. *On input a message* m*, the key* pk*, and access structure* $\mathbb{A} \subseteq 2^{\mathbb{U}}$*, the algorithm* DHash *outputs a hash* h*, randomness* r *and auxiliary information* ct.

$\mathsf{DVer}(pk, m, h, r)$. *On input a public key* pk*, message* m*, hash* h *and randomness* r*, the algorithm* DVer *outputs a bit* $b \in \{0, 1\}$ *to indicate whether the hash* h *is valid.*

$\mathsf{DAdapt}(SK_{\mathbb{S}}, m, h, r, ct, m')$. *On input the key* $SK_{\mathbb{S}}$*, message* m*, hash* h*, randomness* r *and a new message* $m' \in \mathcal{M}$*, the algorithm* DAdapt *outputs a new randomness* r'.

$\mathsf{Delegate}(pk, SK_i, i, ct)$. *On input the public key* pk*, key* SK_i*, attribute* $i \in \mathbb{S}$*, and auxiliary information* ct*, the algorithm* Delegate *outputs a new secret key* $\tilde{SK}_i$ *and a new auxiliary information* $\tilde{ct}$ *if* SK_i *is valid. Otherwise,* Delegate *outputs* $\perp$.

Correctness. The correctness of DACH requires that for all $\kappa \in \mathbb{N}$, for all $\mathbb{A} \subseteq 2^{\mathbb{U}}$, for all $\mathbb{S} \in \mathbb{A}$, for all $i \in \mathbb{S}$, for all $pp \leftarrow \mathsf{PPGen}(1^\kappa)$, for all $(pk, sk) \leftarrow \mathsf{AASetup}(pp)$, for all $GID \in \{0,1\}^*$, for all $SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}(GID, sk, \mathbb{S})$, for all $m, m' \in \mathcal{M}$, for all $(h, r, ct) \leftarrow \mathsf{DHash}(pk, m, \mathbb{A})$, for all $(\tilde{SK}_i, \tilde{ct}) \leftarrow \mathsf{Delegate}(pk, SK_i, i, ct)$, there is $r' \leftarrow \mathsf{DAdapt}(\{SK_{\mathbb{S}}, \tilde{SK}_i\}, m, h, r, \{ct, \tilde{ct}\}, m')$ such that $1 = \mathsf{DVer}(pk, m, h, r) = \mathsf{DVer}(pk, m', h, r')$.

4.2 Security Model

The security models of our DACH by following the games between a challenger and an attacker are presented as follows.

Indistinguishability. It is hard for any PPT adversary $\mathcal{A}$ to distinguish whether the randomness was obtained via DHash or DAdapt, i.e., the adapted hashes are indistinguishable from the fresh one. Thus, DACH is indistinguishable if the advantage of $\mathcal{A}$ winning the following experiment is negligibly, i.e., $|\Pr[\mathbf{Exp}^{\mathsf{Ind}}_{\mathcal{A},\mathsf{DACH}}(\kappa) = 1] - \frac{1}{2}| \leq f(\kappa)$.

Experiment $\mathbf{Exp}^{\mathsf{Ind}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$:
$pp \leftarrow \mathsf{PPGen}(1^\kappa)$, $(pk, sk) \leftarrow \mathsf{AASetup}(pp)$
$b \leftarrow \{0,1\}$, $b^* \leftarrow \mathcal{A}^{\mathcal{O}}(pk, sk)$
where $\mathcal{O} \leftarrow \{\mathsf{DHash}, \mathsf{DAdapt}\}$ and DHash, DAdapt on input $m, m', \mathbb{S}, \mathbb{A}$:
$(h', r', ct') \leftarrow \mathsf{DHash}(pk, m', \mathbb{A})$
$(h, r, ct) \leftarrow \mathsf{DHash}(pk, m, \mathbb{A})$
$SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}(GID, sk, \mathbb{S})$
$r'' \leftarrow \mathsf{DAdapt}(SK_{\mathbb{S}}, m, h, r, ct, m')$
if $r' = \perp \vee r'' = \perp \vee m' \notin \mathcal{M}$, return $\perp$
if $b = 0$, return (h', r', ct')
if $b = 1$, return (h, r'', ct)
return 1, if $b^* = b$; else, return 0.

Outsider Collision Resistance. It requires that it is hard for any PPT adversary $\mathcal{A}$ adaptively accessing to the oracle DAdapt to find collisions for messages which is not queried to DAdapt. Thus, DACH is outsider collision-resistant if for $\mathcal{A}$ the probability of the following experiment returning 1 is negligibly, i.e., $\mathsf{Pr}[\mathbf{Exp}^{\mathsf{OCR}}_{\mathcal{A},\mathsf{DACH}}(\kappa) = 1] \leq f(\kappa)$.

Experiment $\mathbf{Exp}^{\mathsf{OCR}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$:
$pp \leftarrow \mathsf{PPGen}(1^\kappa)$, $(pk, sk) \leftarrow \mathsf{AASetup}(pp)$
$\mathbb{Q}, \mathbb{M} \leftarrow \emptyset$, $k \leftarrow 0$
$(m_0, h_0, r_0, ct_0, m_0', r_0') \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
where $\mathcal{O} \leftarrow \{\mathsf{DKGen}, \mathsf{DKGen}', \mathsf{DAdapt}, \mathsf{DAdapt}'\}$
and DKGen on input $sk, \mathbb{S}$:
$SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}'(\cdot, sk, \mathbb{S})$
$\mathbb{Q} \leftarrow \mathbb{Q} \cup \{(k, SK_{\mathbb{S}})\}$, $k \leftarrow k + 1$
and DAdapt on input $m, h, m', (j, SK_{\mathbb{S}}), ct$:
return $\perp$, if $\mathsf{DVer}(pk, m, h, r) \neq 1 \vee (j, SK_{\mathbb{S}}) \notin \mathbb{Q}$
return $r' \leftarrow \mathsf{DAdapt}'(SK_{\mathbb{S}}, m, h, r, ct, m')$
$\mathbb{M} \leftarrow \mathbb{M} \cup \{m, m'\}$
return 1, if $\mathsf{DVer}(pk, m_0, h_0, r_0) = 1 \wedge \mathsf{DVer}(pk, m_0', h_0, r_0') = 1 \wedge m_0 \notin \mathbb{M} \wedge m_0 \neq m_0'$
else, return 0.

Insider Collision Resistance. Assume that any PPT adversary $\mathcal{A}$ can adaptively access to oracles DKGen and DAdapt, and $\mathcal{A}$ holds some attributes which can find collisions for some hashes. It is hard for the adversary $\mathcal{A}$ whose attributes cannot satisfy the policy encoded in the attacked hash to find collisions. Thus, DACH is insider collision-resistant if for $\mathcal{A}$ the probability that the following experiment returns 1 is negligibly, i.e., $\mathsf{Pr}[\mathbf{Exp}^{\mathsf{ICR}}_{\mathcal{A},\mathsf{DACH}}(\kappa) = 1] \leq f(\kappa)$.

Experiment $\mathbf{Exp}^{\mathsf{ICR}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$:
$pp \leftarrow \mathsf{PPGen}(1^\kappa)$, $(pk, sk) \leftarrow \mathsf{AASetup}(pp)$
$\mathbb{Q}, \mathbb{V}, \mathbb{H} \leftarrow \emptyset$, $k \leftarrow 0$
$(m_0, h_0, r_0, ct_0, m_0', r_0') \leftarrow \mathcal{A}^{\mathcal{O}}(pk)$
where $\mathcal{O} \leftarrow \{\mathsf{DKGen}, \mathsf{DKGen}', \mathsf{DHash}, \mathsf{DAdapt}\}$
and DKGen on input $sk, \mathbb{S}$:
$SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}(\cdot, sk, \mathbb{S})$
$\mathbb{V} \leftarrow \mathbb{V} \cup \{\mathbb{S}\}$
and DKGen′ on input $sk, \mathbb{S}$:
$SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}'(\cdot, sk, \mathbb{S})$
$\mathbb{Q} \leftarrow \mathbb{Q} \cup \{(k, SK_{\mathbb{S}})\}$, $k \leftarrow k + 1$
and DHash on input $m, \mathbb{A}$:
$(h, r, ct) \leftarrow \mathsf{DHash}(pk, m, \mathbb{A})$
$\mathbb{H} \leftarrow \mathbb{H} \cup \{(h, m, \mathbb{A})\}$
and DAdapt on input $m, h, m', (j, SK_{\mathbb{S}}), ct$:
return $\perp$, if $(j, SK_{\mathbb{S}}) \notin \mathbb{Q}$
return $r' \leftarrow \mathsf{CHAdapt}(SK_{\mathbb{S}}, m, h, r, ct, m')$
if $(h, m, \mathbb{A}) \in \mathbb{H}$, let $\mathbb{H} \leftarrow \mathbb{H} \cup \{(h, m', \mathbb{A})\}$
return 1, if $\mathsf{DVer}(pk, m_0, h_0, r_0) = 1 \wedge \mathsf{DVer}(pk, m_0', h_0, r_0') = 1 \wedge (h_0, \cdot, \mathbb{A}) \in \mathbb{H}$, where $m_0 \neq m_0' \wedge \mathbb{A} \cap \mathbb{S} = \emptyset \wedge (h_0, m_0, \cdot) \notin \mathbb{H}$
else, return 0.

Backward/Forward Security. A delegation algorithm is secure if the algorithm satisfies with: (1) Backward Security. It is hard for any PPT adversary to find collisions for messages which are not queried to oracle DAdapt with the keys of AA left from the network. (2) Forward Security. After the previous AA delegates its role to a new AA, the new AA can issue new keys to help efficiently compute collisions of previously published hashes. Meanwhile, it is hard for any PPT adversaries who are delegated and collude with each other to find collisions for messages which are not queried to oracle DAdapt.

4.3 Concrete Construction of DACH

A scalable and secure instantiation of DACH is presented in Fig. 3. In particular, the notations are the same as those introduced in Section 2.

The construction of our DACH consists of the LSSS, DCP-ABE, and RSA-based CHET. As shown in Fig. 3, the system is initialized according to the algorithm $pp \leftarrow \mathsf{PPGen}(1^\kappa)$ and AA joins the system by running the algorithm $(pk, sk) \leftarrow \mathsf{AASetup}(pp)$. Data m is hashed with an access policy $\mathbb{A}$ via the algorithm DHash. Based on the LSSS scheme, policy $\mathbb{A}$ encoded by a set $\mathbb{S}$ of attributes is represented as an access matrix (A, ρ). After running the algorithm DHash, the hash tuple (h, r, ct) of data m is obtained, where h is the hash value, r is the randomness, and ct is the auxiliary information. Then, any user can obtain a set of keys corresponding to its attributes via algorithm $SK_{\mathbb{S}} \leftarrow \mathsf{DKGen}(GID, SK, \mathbb{S})$ from different AAs, where GID is the global identity of the user. Next, if user's attributes can satisfy the policy $\mathbb{A}$, the user's keys $SK_{\mathbb{S}}$ can recover the correct key etd and effectively compute a collision for new data m' via algorithm $r' \leftarrow \mathsf{DAdapt}(SK_{\mathbb{S}}, m, h, r, ct, m')$. The new randomness r' can map m and m' ($m \neq m'$) into the same hash value h. Finally, any AA can delegate its role to a new AA via the algorithm Delegate. The new AA distributes the new auxiliary information $\tilde{ct}$ and key $\tilde{SK}_{i,GID}$ corresponding to the attribute i to help the users compute collisions.

5 DACH-Based SDR-Chain

In this section, we apply our DACH to make blockchain mutable, called SDR-chain. We first present the main phases and then take the Bitcoin blockchain as an example to present how DACH integrates with blockchain.

5.1 Main Phases

We present the main phases of applying DACH to SDR-chain, including AAs selection, redactable transactions generation, transaction redaction, and delegation. Notice that the core components of SDR-chain including consensus protocol and block propagation follow the original immutable-blockchain except the block proposal/validation.

AAs selection. AAs are selected from nodes and in charge of keys distribution. How to select AAs is important. In order to ensure the redaction controlled by a majority of honest nodes, AAs are selected on the basis of their contributions in the blockchain system. The more contributions the node makes, the higher probability the node is selected with. The contributions are estimated by tow components. The first component is measured by the ratio of the number of blocks that the node generates in the main chain, denoted as N_B. The second component is measured by the ratio of the number of valid redactions that the node achieves, denoted as N_R. Then, the probability is denoted as the weighted sum of two contributions, i.e., $\mathsf{Pr}_c = \zeta N_B + \eta N_R$, where $0 \leq \zeta, \eta \leq 1$. The parameters ζ and η may have an impact

$\mathsf{PPGen}(1^\kappa)$. On input security parameter κ:

1) Generate parameters of bilinear group $(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g) \leftarrow \mathcal{G}(1^\kappa)$.
2) Choose a large prime $e_0 > N_1$, where $N_1 = max\{N' | (N', \cdot, \cdot, \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa)\}$.
3) Choose two hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathsf{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}^*_{nn'}$, an encoding algorithm $\mathsf{encode} : \{0,1\}^* \rightarrow \mathbb{G}_T$, a decoding algorithm $\mathsf{decode} : \mathbb{G}_T \rightarrow \{0,1\}^*$.

Return public parameters $pp \leftarrow \{N, \mathbb{G}, \mathbb{G}_T, g, e_0\}$.

$\mathsf{AASetup}(pp)$. On input the public parameters pp:

1) Each authority AA_i controls an attribute $i \in \mathbb{S}$. For the attribute $i \in \mathbb{S}$, the authority chooses $\alpha_i, \beta_i \xleftarrow{R} \mathbb{Z}_N$ and computes $spk \leftarrow \{\hat{e}(g,g)^{\alpha_i}, g^{\beta_i}\}$ where $ssk \leftarrow \{\alpha_i, \beta_i\}$.
2) Run $(n, p, q, \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa)$ and let $msk \leftarrow (p, q)$.

Return $pk \leftarrow (n, spk)$ and $sk \leftarrow (msk, ssk)$.

$\mathsf{DKGen}(GID, sk, \mathbb{S})$. On input GID, secret key sk, and attribute $\mathbb{S}$:

Create a secret key $SK_{i,GID} = g^{\alpha_i} y^{\beta_i}$ corresponding for each attribute $i \in \mathbb{S}$ of GID, where $y = \mathsf{H}_1(GID)$.

Return $SK_{\mathbb{S}} \leftarrow (\{SK_{i,GID} | i \in \mathbb{S}\}, msk)$.

$\mathsf{DHash}(pk, m, (A, \rho))$. On input public key pk, message m, monotone span program $(A_{k\times l}, \rho)$ of access structure $\mathbb{A}$:

1) Run $(n', p', q', \cdot, \cdot) \leftarrow \mathsf{RSAKGen}(1^\kappa)$, where $gcd(n, n') = 1$ and $etd \leftarrow \mathsf{encode}(p', q')$. Let etd keep secret and publish n'.
2) Choose $r \xleftarrow{R} \mathbb{Z}^*_{nn'}$ and compute $h = \mathsf{H}_2(m) r^{e_0} \bmod (nn')$.
3) Choose $s, (a_2, ..., a_l), (b_2, ..., b_l) \xleftarrow{R} \mathbb{Z}_N$ to generate vectors $v = (s, a_2, ..., a_l)$ and $w = (0, b_2, ..., b_l)$. Compute $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot w$. For the xth row A_x of A, choose $r_x \xleftarrow{R} \mathbb{Z}_N$, then compute

$$ct \leftarrow \{c_0 = etd \cdot \hat{e}(g,g)^s, c_{1,x} = \hat{e}(g,g)^{\lambda_x} \hat{e}(g,g)^{\alpha_{\rho(x)} r_x}, c_{2,x} = g^{r_x}, c_{3,x} = g^{r_x \beta_{\rho(x)}} g^{\omega_x}\}.$$

Return (h, r, ct).

$\mathsf{DVer}(pk, m, h, r)$. On input public key pk, message m, hash value h, and randomness r:

Verify whether tuple $(h, r \in \mathbb{Z}_{nn'})$ satifies $h = \mathsf{H}_2(\mathsf{m}) r^{e_0}$ or not.

Return $b \in \{1, 0\}$.

$\mathsf{DAdapt}(SK_{\mathbb{S}}, m, h, r, ct, m')$. On input keys $SK_{\mathbb{S}} = (\{SK_{i,GID} | i \in \mathbb{S}\}, msk)$, message m, tuple (h, r, ct), and new message m':

If the attribute set $\mathbb{S}$ of GID satisfies the access structure $\mathbb{A}$:

1) Recover $etd = c_0 / \prod_x (D_x)^{C_x}$, where coefficients $\{C_x \in \mathbb{Z}_N\}$ satisfy $\sum_x C_x A_x = (1, 0, ..., 0)$, and

$$D_x = c_{1,x} \cdot \hat{e}(\mathsf{H}_1(GID), c_{3,x}) / \hat{e}(SK_{\rho(x),GID}, c_{2,x}) = \hat{e}(g,g)^{\lambda_x} \hat{e}(y,g)^{\omega_x}.$$

2) Compute d_0 s.t. $e_0 d_0 \equiv 1 \bmod \varphi(nn')$ with msk and $\mathsf{decode}(etd)$.
3) Compute a new randomness r' for the new message m', i.e., $r' = (h \mathsf{H}_2(m')^{-1})^{d_0} \bmod (nn')$.
4) Check the verification by invoking the algorithm DVer. If $1 \leftarrow \mathsf{DVer}(pk, m', h, r')$, return r'.

Return r' and $\perp$ otherwise.

$\mathsf{Delegate}(pk, SK_{i,\cdot}, i, ct)$. On input public key pk, secret key $SK_{i,\cdot} = g^{\alpha_i} y_1^{\beta_i}$ for $\rho(x) = i$, auxiliary information ct:

1) Compute $K_i \leftarrow (u_1, u_2)$, where $u_1 = g^{\alpha_i}$, $u_2 = g^{r_x \beta_i}$, and $y_1 = \mathsf{H}_1(\cdot)$.
2) Verify K_i by

$$\hat{e}(SK_{\rho(x),\cdot}, c_{2,x}) \stackrel{?}{=} \hat{e}(u_1, c_{2,x}) \cdot \hat{e}(y_1, u_2).$$

3) If holds, choose $\alpha'_i, \beta'_i \xleftarrow{R} \mathbb{Z}_N$ and $\alpha'_i \neq \alpha_i$, $\beta'_i \neq \beta_i$ to create a new auxiliary information $\tilde{ct}$, i.e.,

$$\tilde{ct} \leftarrow \{\tilde{c_0} = c_0, \tilde{c}_{1,x} = c_{1,x} \cdot \hat{e}(c_{2,x}, g^{\alpha'_i}) / \hat{e}(c_{2,x}, u_1), \tilde{c}_{2,x} = c_{2,x}, \tilde{c}_{3,x} = c_{3,x} \cdot (c_{2,x})^{\beta'_i} / u_2\}.$$

4) Invoke $\tilde{SK}_{i,GID} \leftarrow \mathsf{DKGen}(GID, (\alpha'_i, \beta'_i), i)$ for GID.

Return $(\hat{ct}, \tilde{SK}_{i,GID})$ and $\perp$ otherwise.

Fig. 3: Concrete construction of DACH.

on the performance of the blockchain system, which is an interesting topic and will be researched in our future work.

Redactable Transaction Generation. As shown in steps $1-3$ of Fig. 4, the sender generates a redactable transaction Tx based on DACH, and the transaction Tx is broadcast and verified to generate a new block in the network. The detailed algorithms are invoked as follows. First, the system initializes to generate parameters by invoking algorithms $\Pi_{\mathsf{DACH}}.\mathsf{PPGen}$ and $\Pi_{\mathsf{DACH}}.\mathsf{AASetup}$. Next, the sender hashes transaction Tx with a policy $\mathbb{A}$ by invoking the algorithm $\Pi_{\mathsf{DACH}}.\mathsf{DHash}$, i.e., $(h, r, ct) \leftarrow \mathsf{DHash}(pk, \mathsf{Tx}, \mathbb{A})$. The hash

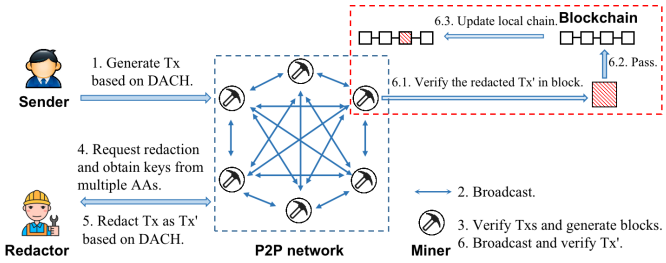

Fig. 4: The process of redaction.

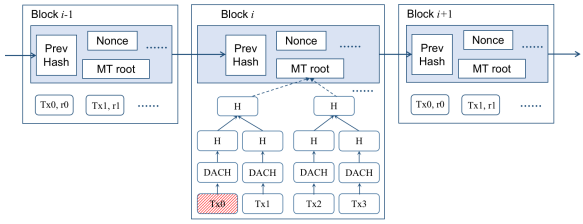

Fig. 5: The structure of DACH-based blockchain, where H is hash function such as SHA256. If Tx0 is redacted, DACH function ensures that the inputs of H is invariant.

value h is signed by using the algorithm ECDSA. Then, the transaction Tx is broadcast and verified in the P2P network. nodes verify the hash value h by invoking the algorithm Π_{DACH}.DVer besides the regular validation process of the transaction. If the algorithm outputs 1, Tx is valid and its h is integrated into a Merkle tree to create a new block, as shown in Fig. 5.

Transaction Redaction. As shown in steps $4-6$ of Fig. 4, the redactor gets a set of key $SK_{i,GID}$ corresponding to its attributes by interacting with a group of AAs who invoke the algorithm Π_{DACH}.DKGen. Then, the redactor recovers the trapdoor key to find a collision of the redacted transaction Tx' by invoking the algorithm Π_{DACH}.DAdapt. Specifically, a new randomness r can be computed such that the original transaction Tx and the redacted transaction Tx' are mapped to the same hash value h. Notice that the data required to redact should have no impact on related transactions/blocks in the past and future, such as the data stored in the field OP_RETURN of the transaction. After that, the redacted transaction Tx' is braodcast and verified in the work.

An example is given to present the above process. Assume that the transaction Tx is recorded in block $\#i$ of the chain $\mathcal{C}$. Let the redacted transaction be Tx' and let the redaction policy be $\mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C}) \wedge \mathsf{D}$. As shown in Fig. 2, the redactor U_1, holding a set of attributes $\mathbb{S} = \{\mathsf{A}, \mathsf{B}, \mathsf{D}\}$, interacts with a group of AAs. Then, the redactor gets keys $SK_{\mathbb{S}} = \{SK_{\mathsf{A},1}, SK_{\mathsf{B},1}, SK_{\mathsf{D},1}\}$ from $\{AA_1, AA_2, AA_4\}$, respectively. These keys are used for computing the collision for Tx' by invoking algorithm $r' \leftarrow \mathsf{DAdapt}(SK_{\mathbb{S}}, \mathsf{Tx}, h, r, ct, \mathsf{Tx}',)$. Notice that the redactor can compute a valid randomness r because the attributes of the redactor can satisfy the policy. After that, Tx' is broadcast and verified in the work. Nodes check r' by invoking the algorithm $\mathsf{DVer}(pk, \mathsf{Tx}', h, r')$ besides the regular transaction verification. If both of them are valid, nodes update their local chain by replacing Tx with transaction Tx'.

Delegation. Any AA can delegate its role to other nodes by running the algorithm Π_{DACH}.Delegate in our SDR-chain. Delegation is bootstrapped by two cases. (1) A node requests to be the AA. The probability of the node being successfully delegated is also estimated by Pr_c. (2) The AA who wants to leave the system requests delegation by announcing a delegation message in the network. The nodes who want to become authorities interact with the AA and they are selected according to their probability Pr_c. Note that each AA is allowed to control an attribute per policy, and each attribute is allowed to be controlled by multiple AAs. For the attribute that is used multiple times, it should be labeled with a fresh value j, denoted as $\mathsf{Att} : j$.

An example is shown in Fig. 2, where the policy is defined as $\mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C}) \wedge \mathsf{D}$. Assume that AA_4 delegates its role (corresponding to the attribute D) to AA_4^*. AA_4^* runs the algorithm Delegate to generate new auxiliary information $\tilde{ct}$ and new secret keys $SK^*_{\mathsf{D},3}$. The redactor holding attributes $\{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$ gets keys $\{SK_{\mathsf{A},3}, SK_{\mathsf{C},3}, SK^*_{\mathsf{D},3}\}$ from AA_1, AA_3, and AA_4^*, respectively. Then the redactor can recover the correct trapdoor key from $\{c_0, c_{1,\mathsf{A}}, c_{2,\mathsf{A}}, c_{3,\mathsf{A}}, c_{1,\mathsf{C}}, c_{2,\mathsf{C}}, c_{3,\mathsf{C}}, \tilde{c}_{1,\mathsf{D}}, \tilde{c}_{2,\mathsf{D}}, \tilde{c}_{3,\mathsf{D}}\}$. That is, the new AA_4^* can issue valid keys to execute redactions, which improves the functionality of the SDR-chain.

5.2 Integrating with Blockchain

This section presents how DACH integrates with Bitcoin blockchain. The block propagation and consensus protocol follow the Bitcoin blockchain. Blocks are broadcast in the P2P network and the longest proof of work chain is always chosen. In the phase of block proposal/validation, there are some differences from the Bitcoin.

block proposal/validation. After transactions are generated and hashed by invoking Π_{DACH}.DHash, they are broadcast in the P2P network. As shown in Fig. 5, the additional data r is recorded in the non-signed part of transactions. Miners collect valid transactions to propose new blocks by verifying the DACH values of transactions besides the regular processes, where the verification algorithm invokes Π_{DACH}.DVer.

In summary, our DACH acts as a data preprocessing layer on the Bitcoin blockchain. There are two additional steps required. (1) The DACH value (h, r) of the transaction needs to be checked in transaction verification. (2) The transaction needs to record the additional data r besides the regular fields, which may increase the length of the transaction. However, it has a tiny impact on the throughput of the blockchain system because the length of r is short and the throughput is mainly determined by the consensus mechanism.

6 Analysis

The SDR-chain is built on the DACH function to perform redactions. Thus, we first analyze the correctness and security of the DACH and then analyze the design goals of the SDR-chain in this section.

6.1 Correctness

In this section, we analyze the correctness of the DACH which determines the correctness of redactions in the SDR-chain.

Correctness of the DACH. It requires that (1) the DAdapt algorithm allows the one whose attributes satisfy the access policy to find a valid collision correctly and (2) the algorithm Delegate allows the one who leaves the network to delegate the role to another correctly. The processes are described as follows in detail.

(1) Given $\mathbb{S}$ satisfying the policy (A, ρ), $h \leftarrow \mathsf{DHsh}(\cdot, m, \cdot)$, for $m' \neq m$, if there is $h' \leftarrow \mathsf{DHsh}(\cdot, m', \cdot)$ such that $\mathsf{DVer}(\cdot, m', h', r') = 1$ and $h = h'$. Run the DKGen and DKGen algorithms for the attributes in $\mathbb{S}$ and output the new randomness $r' = (h\mathsf{H}_2(m')^{-1})^d \bmod (nn')$ for the new message m', where $ed \equiv 1 \bmod \varphi(nn')$. Then, a hash value h' is easily computed, i.e., $h' = \mathsf{H}_2(m')r'^e$ and $h' = \mathsf{H}_2(m')\big((h\mathsf{H}_2(m')^{-1})^d\big)^e = h$. That is, the value r' can make $\mathsf{DVer}(\cdot, m', h', r') = 1$ and $h = h'$ hold. Hence, the one whose attributes satisfy the access policy can correctly find a valid collision.

(2) AA_j runs the algorithm Delegate for an attribute $k \in \mathbb{S}$. Given the new auxiliary information $\tilde{ct} \leftarrow \{\tilde{c_0}, \tilde{c_{1,k}}, \tilde{c_{2,k}}, \tilde{c_{3,k}}\}$, the new key of AA_j should be valid to recover the trapdoor key etd with the cooperation of other AAs who control other attributes in $\mathbb{S}$. Assume that the key $\{SK_{i,\cdot} : i \in \mathbb{S}/k\}$ for other attributes $i \in \mathbb{S}/k$ and the values $\{D_{\rho^{-1}(i)}\}$ are also given. The key for the attribute k of AA_j is denoted as $SK_{k,\cdot} = g^{\alpha'_k} y^{\beta'_k}$. Let the attribute $\rho(x) = k$, then the new auxiliary information $\tilde{ct}$ associated with k can be denoted as:

$$\begin{aligned} \tilde{c_0} &= c_0 = etd \cdot \hat{e}(g,g)^s \\ \tilde{c_{1,x}} &= c_{1,x}\hat{e}(c_{2,x}, g^{\alpha'_i})/\hat{e}(c_{2,x}, u_1) = \hat{e}(g,g)^{\lambda_x}\hat{e}(g,g)^{\alpha'_k r_x} \\ \hat{c_{2,x}} &= g^{r_x} \\ \hat{c_{3,x}} &= c_{3,x}(c_{2,x})^{\beta'_k}/u_2 = g^{r_x\beta'_k} g^{\omega_x}. \end{aligned} \tag{1}$$

Thus, we can get

$$\begin{aligned} D_{\rho^{-1}(k)} &= \tilde{c_{1,x}} \cdot \hat{e}(\mathsf{H}_1(\cdot), \tilde{c_{3,x}})/\hat{e}(SK_{k,\cdot}, \tilde{c_{2,x}}) \\ &= \hat{e}(g,g)^{\lambda_x}\hat{e}(y,g)^{\omega_x}. \end{aligned}$$

With the keys $\{D_{\rho^{-1}(k)}\} \cup \{D_{\rho^{-1}(i)} : i \in \mathbb{S}/k\}$, we can compute the correct coefficients $\{C_x\}$ such that $\sum_x C_x A_x = (1, 0, 0, ...)$ to compute

$$\prod_{x \in \rho^{-1}(\mathbb{S})} (D_x)^{C_x} = \hat{e}(g,g)^s.$$

Next, we can recover the valid etd from $c_0 = etd \cdot \hat{e}(g,g)^s$. Given $(p', q') \leftarrow \mathsf{decode}(etd)$ and msk, we can easily compute a constant d such as $ed \equiv 1 \bmod \varphi(nn')$. It is easy to find a collision as the description of (a). Thus, based on the Delegate algorithm, AA_j can help the redactor recover the trapdoor key correctly. In summary, our DACH is correct.

6.2 Security Analysis

In this section, we analyze the security of the DACH and the SDR-chain in detail.

6.2.1 Security of the DACH

We analyze the security of the DACH constructed as Fig. 3 in the random oracle model.

Theorem 1. *For all PPT adversaries, the DACH scheme is strongly indistinguishable if it is based on a strongly indistinguishable CHET scheme.*

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be the adversary against the schemes DACH and CHET, respectively. $\mathcal{A}$ runs the experiment $\mathbf{Exp}^{\mathsf{Ind}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$. For the oracle DHash and DAdapt of $\mathcal{A}$, $\mathcal{B}$ internally uses the oracle CHash and CHAdapt of scheme CHET. That is, $\mathcal{B}$ queries challenger to get (h, r, etd) by access to oracle CHash and returns (h, r, ct) to $\mathcal{A}$, where $ct \leftarrow \Pi_{\mathsf{DCP-ABE}}.\mathsf{Enc}(etd, \cdot, \cdot)$. Then, $\mathcal{B}$ recovers $etd \leftarrow \Pi_{\mathsf{DCP-ABE}}.\mathsf{Dec}(\cdot, ct)$ and gets the randomness r'' by querying challenger access to oracle CHAdapt. After that, $\mathcal{B}$ returns r'' to $\mathcal{A}$. As soon as $\mathcal{A}$ outputs its guess b, $\mathcal{B}$ returns b to the challenger. Obviously, the probability of $\mathcal{A}$ winning is the same as $\mathcal{B}$ winning. Besides, the CHET scheme is indistinguishable based on the one-more RSA-inversion assumption [33]. Thus, Theorem 1 holds.

Theorem 2. *For all PPT adversaries, the DACH scheme is outsider collision-resistant if it is based on a publicly collision-resistant CHET scheme.*

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be the adversary against the schemes DACH and CHET, respectively. $\mathcal{A}$ runs the experiment $\mathbf{Exp}^{\mathsf{OCR}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$. For the oracle DAdapt of $\mathcal{A}$, $\mathcal{B}$ internally uses the oracle CHAdapt of scheme CHET. $\mathcal{B}$ recovers etd from ct and queries challenger access to oracle CHAdapt. $\mathcal{B}$ returns valid r' for m' to $\mathcal{A}$. For any messages $\{m_0, m'_0\}$ that was not queried, the possibility of $\mathcal{B}$ finding a valid collision tuple $(h_0, m_0, r_0, m'_0, r'_0, ct_0)$ is the same with that of $\mathcal{A}$. Moreover, according to the RSA assumption, $\mathcal{B}$ cannot effectively compute r'_0 without the valid etd_0. Thus, Theorem 2 holds.

Theorem 3. *For all PPT adversaries, the DACH scheme is insider collision-resistant if it is based on a privately collision-resistant CHET scheme and an IND-CCA2 secure DCP-ABE scheme.*

Proof. Let $\mathcal{B}$ be the adversary against the private collision-resistance of scheme CHET and let $\mathcal{C}$ be the adversary against IND-CCA2 security of scheme DCP-ABE. Let $\mathsf{Pr}[\mathbf{G}_i]$ be the success probability of the adversary in Game i and the number of queries to DAdapt be denoted as q. Consider the following sequence of games.

Game 0. It is the original insider collision resistance experiment $\mathbf{Exp}^{\mathsf{ICR}}_{\mathcal{A},\mathsf{DACH}}(\kappa)$.

Game 1. As Game 0, but adversary $\mathcal{A}$ who does not have enough attributes makes kth query $(h_0, r_0, ct_0, m_0, m'_0)$, where m_0 and m'_0 are fresh and the oracle DAdapt is internally queried to CHAdapt. What $\mathcal{A}$ outputs is the output of $\mathcal{B}$. If $\mathcal{A}$ finds a valid collision, the game aborts. Therefore, the winning probability is the same as that of Game 0, unless an abort happens, i.e., $\mathsf{Pr}[\mathbf{G}_1] = \mathsf{Pr}[\mathbf{G}_0] \cdot \frac{1}{q}$.

Game 2. As Game 1, but etd_0 is directly used to compute collision instead of decrypting ct_0. The winning probability is the same as that of Game 1, i.e., $\mathsf{Pr}[\mathbf{G}_2] = \mathsf{Pr}[\mathbf{G}_1]$.

Game 3. As Game 2, but the oracle DHash is internally queried to CHash, where $\mathcal{C}$ encrypts 0 instead of the real etd_0, i.e., $ct_0 \leftarrow \Pi_{\mathsf{DCP-ABE}}.\mathsf{Enc}(0, \cdot, \cdot)$, and the oracle DKGen is internally queried to KGen. Game 2 and Game 3 are

indistinguishable under the IND-CCA2 security of scheme DCP-ABE, i.e., $|\mathsf{Pr}[\mathbf{G}_3] - \mathsf{Pr}[\mathbf{G}_2]| \leq \mathbf{Adv}^{\mathsf{IND-CCA2}}_{\mathcal{C},\mathsf{DCP-ABE}}(\kappa)$.

Game 4. As Game 3, but the game aborts, if $\mathcal{A}$ finds a valid collision and returns $(h_0, r_0, ct_0, m_0, m_0', r_0')$. Then, $\mathcal{B}$ uses this attack to break the strong private collision-resistance of CHET. Finally, $\mathcal{B}$ also finds the collision. The winning probability of Game 4 is the same as that of Game 3. Moreover, there is $\mathsf{Pr}[\mathbf{G}_4] \leq \mathbf{Adv}^{\mathsf{PrivateCR}}_{\mathcal{B},\mathsf{CHET}}(\kappa)$.

As the above games show, the advantage of $\mathcal{A}$ in the original game is bounded by $\mathsf{Pr}[\mathbf{G}_0] \leq q(\mathbf{Adv}^{\mathsf{PrivateCR}}_{\mathcal{B},\mathsf{CHET}}(\kappa) + \mathbf{Adv}^{\mathsf{IND-CCA2}}_{\mathcal{C},\mathsf{DCP-ABE}}(\kappa))$, which is negligible. Thus, the DACH scheme is insider collision-resistant, i.e., Theorem 3 holds.

Note that DACH can resist the outsider and insider collisions so that it also satisfies key-exposure freeness, meaning that given a collision for a message, no PPT adversary can efficiently extract the secret key to find collisions [34]. Because key-exposure freeness is weaker than the outsider collision resistance.

Theorem 4. *The DACH scheme is collusion-resistant.*

Proof. The DACH scheme is collusion resistant if it can resist two kinds of collusion attacks. Assume that the redaction policy is coded over attributes $\mathbb{S} = \{S_1, S_2, ..., S_i\}$. (1) Consider the collusion between redactors whose attributes cannot pass the policy alone. After collusion, they can get a set of attributes which can pass the policy. Let the colluded attributes be $\mathbb{S}$. Then, the colluding redactors, owning the global identities $\{GID_1, ..., GID_j\}$, aims at recovering the valid trapdoor key etd to compute valid collisions. Redactors get the secret keys $\{SK_{S_i,GID_j}\}$ from AAs and compute values $\{D_x = \hat{e}(g,g)^{\lambda_x}\hat{e}(y_j,g)^{\omega_x} | x \in \rho^{-1}(\mathbb{S})\}$. It is obvious that the component $y_j = \mathsf{H}_1(GID_j)$ of each $SK_{i,GID}$ is different, so the values $\{D_x\}$ is blind. After combining $\{D_x\}$, the shares ω_x of 0 in the exponent with different base $\hat{e}(y_j, g)$ cannot be canceled out. Thus, the trapdoor key computed by collusion attacks is still blind and invalid.

(2) Some AAs, who can issue the secret keys $SK_{\mathbb{S}'}$ corresponding to the attributes $\mathbb{S}'$ which cannot satisfying the policy, may collude to recover the trapdoor key. Intuitively, it is hard to compute a correct D_x because the components y_1 in the key $SK_{i,\cdot}$ of each AA is different. That is, the collusion of AAs cannot recover the trapdoor key unless all AAs collude. However, it contradicts the threat model. In summary, Theorem 4 holds.

Theorem 5. *For all PPT adversaries, the algorithm* Delegate *of the DACH is secure.*

Proof. As the previous definition, the algorithm Delegate is secure if it satisfies backward security and forward security. Assume that AA_j who is in charge of attribute i wants to leave the network and delegate the information about attribute i to AA_k. Then, AA_k runs Delegate algorithm on input of a secret key $K_i = (g^{\alpha_i}, g^{r_x\beta_i})$ of AA_j. The algorithm Delegate returns new $\tilde{ct}$ as shown in formula (1).

Backward Security: Given $\tilde{ct}, spk', ssk, SK_{i,GID}$, if an adversary holding the attribute i cannot compute the correct $D_{\rho^{-1}(i)}$ then the algorithm Delegate satisfies backward security. Obviously, $\tilde{ct}$ is associated with the secret key $ssk' = (\alpha', \beta')$ of AA_k. Due to $\alpha' \neq \alpha$ and $\beta' \neq \beta$, the target value $D_{\rho^{-1}(i)}$ is blind by a value $\hat{e}(g,g)^{(\alpha_i - \alpha_i')r_{\rho^{-1}(i)}}\hat{e}(y,g)^{(\beta i - \beta_i')r_{\rho^{-1}(i)}}$. That is, the adversary cannot compute the correct $D_{\rho^{-1}(i)}$.

Forward Security: Given $\tilde{ct}, spk', \tilde{SK}_{i,GID}$, if a redactor can correctly recover $D_{\rho^{-1}(x)}$ with the new key of AA_k, then the algorithm Delegate satisfies forward security. According to the correctness analysis of algorithm Delegate described in Section 6.1, it is effective for the redactor to recover $D_{\rho^{-1}(x)}$.

In summary, the algorithm Delegate of the DACH scheme is secure, i.e., Theorem 5 holds.

Theorem 6. *For all PPT adversaries, the DACH scheme is secure if it is correct, indistinguishable, outsider collision-resistant, insider collision-resistant, and attribute authority delegation secure.*

6.2.2 Security of SDR-chain

The generic security properties and common attacks of SDR-chain are analyzed as follows.

Consistency. In the traditional blockchain, redacting transactions will result in breaking links between transactions (blocks) and making blockchain inconsistent. However, our scheme ensures that the hash of the redacted transaction is the same as that of the original transaction, meaning that the links between transactions (blocks) are maintained. Honest nodes can fast agree on a unified view of blockchain history based on our scheme, i.e., the consistency is satisfied.

Consensus Delay. The consensus is affected by the network connectivity, forks, and mining strategy [35], and a consensus delay will undermine the security. The delay occurs when miners cannot agree on a final state of the chain. In the SDR-chain, the redacted transaction is broadcast to the network and verified by nodes. If it is valid, the honest nodes approve it and update their local chain. Finally, an honest redaction will efficiently be approved since the majority of nodes are honest in the network. Therefore, our redactable blockchain does not suffer from consensus delay.

Collusion Attacks. According to Theorem 4, our DACH is collusion resistant. Therefore, we mainly discuss if a single malicious AA or multiple colluding AAs abuse the redaction by manipulating the delegation function. We consider two cases. (1) *A single malicious AA manipulates the delegation function and may delegate its secret key to multiple nodes*. In this case, the multiple delegated nodes may also collude. However, their collusion can only get information about the same secret key. Because an AA only owns a secret key corresponding to one of the attributes of the redaction policy. Their collusion cannot recover the trapdoor key for redacting. Thus, this collusion cannot abuse the redaction. (2) *Multiple colluding AAs manipulate the delegation function and may collude with each other*. In our threat model, the collusion of all AAs is not allowed. Besides, if the multiple colluding AAs cannot pass the policy, they cannot get any information about the trapdoor key intuitively. If the colluding AAs can pass the policy, they still cannot recover the trapdoor key. Because their each keys corresponding to each attribute is blind by the different global identity GID. The colluded keys cannot cancel out the component ω_x of D_x, as described in Theorem 4. Hence, it is hard for multiple colluding AAs to abuse the redaction. In summary, our Delegation algorithm is collusion resistant.

Double Spending Attacks. Let us track some transactions Tx1 $\rightarrow$ Tx2 $\rightarrow$ Tx3 appended on blockchain, where the funds of Tx1 flow to Tx3. Let $Add_i (i = 0, 1, 2, 3)$ be users' addresses. Let $Add_1 \rightarrow Add_2$ denote that Add_1 pays Add_2 in Tx2. A malicious user may launch the double-spending attack by requesting to modify transactions' outputs. (1) Consider the first case where the transaction has already been spent. Let the spent transaction Tx2 ($Add_1 \rightarrow Add_2$) be maliciously modified as Tx2* ($Add_1 \rightarrow Add_2^*$). If Tx2* is confirmed by honest nodes, the attack succeeds. Obviously, Tx2* makes Tx3 invalid and conflicts with Tx2, so it cannot be approved by honest nodes. Therefore, the attack fails. (2) Consider the second case where the transaction is not spent. Let the unspent transaction Tx3 ($Add_2 \rightarrow Add_3$) be modified as Tx3* ($Add_2 \rightarrow Add_3^*$). Meanwhile, an transaction Tx4*($Add_3^* \rightarrow Add_4^*$) is issued. If the attacker can double spend the funds of Add_2, the attack succeeds. Obviously, the conflicting transactions Tx3 and Tx3* cannot be accepted by miners in the same block. If Tx3* is approved by honest nodes, Tx3* will replace Tx3 and Tx4* will be confirmed, meaning that Tx3 is erased from the blockchain. Nodes agree with the view of funds flow on $Add_0 \rightarrow Add_1 \rightarrow Add_2 \rightarrow Add_3^* \rightarrow Add_4^*$. If Tx3* is not approved by honest nodes, Tx3 remains intact and another transaction Tx4 ($Add_3 \rightarrow Add_4$) will be confirmed. The view of funds flow is $Add_0 \rightarrow Add_1 \rightarrow Add_2 \rightarrow Add_3 \rightarrow Add_4$. In either case, the funds of Add_2 are spent once, i.e., the attack fails. In summary, the malicious user cannot double-spend transactions by requesting redaction operations.

Security Properties. The proposed SDR-chain is proposed by applying our DACH to an immutable blockchain and its behavior is exactly like the immutable blockchain before the transaction redaction happens. The immutable blockchain should satisfy three security properties defined in [36], including *chain growth, chain quality*, and *common prefix*. We discuss whether redacting on SDR-chain damages the properties or not. If redacting does not damage the security properties, the SDR-chain satisfies the security properties. (1) *Chain growth*. The SDR-chain redacts data at the transaction level and does not allow deleting blocks from the chain, so the redaction operations do not reduce the length of the chain. That is, the redaction operations do not damage the chain growth. (2) *Chain quality*. According to the security analyses, DACH is resistant to inside and outside attacks. It is infeasible for adversaries to perform a maliciously redacted transaction successfully. Thus, it is hard to corrupt an honest block by redaction operations. The proportion of malicious blocks in the chain is not increased, meaning that the chain quality is not broken. (3) *Common prefix*. The SDR-chain does not suffer from the consensus delay, and honest miners can efficiently agree on an honest redaction. That is, honest nodes do not generate alternative chains as a result of the redaction operations. The common prefix of the chains of any two honest nodes is maintained. In summary, the redaction operations do not damage the chain growth, chain quality, and common prefix, so the SDR-chain satisfies the security properties.

6.3 Analysis of Design Goals

We discuss whether the SDR-chain achieves the design goals in this section.

Fine-Grained Control. In the SDR-chain, redaction is performed in trsaction level. Each transaction is hashed with an access policy encoding over attributes. According to the correctness of DACH, anyone who cannot pass the access policy cannot effectively find a valid collision for the redacted transaction. That is, redaction is controlled by the policy in a fine-grained way. Beside, the ability to perform redaction is held by a group of users whose attributes can satisfy the policy. Even though one of them may perform an invalid or even incorrect redaction, others can redress the redaction. This way avoids abusing the redacting power in some degree.

Decentralized Redaction. SDR-chain controls redactions by managing the trapdoor key based on the DCP-ABE. Thus, we discuss the potential centralities of using DACH to indicate the decentralization of the redaction.

On the one hand, the trapdoor key used to redact transactions is encrypted and shared among a group of AAs, which are selected from nodes and dynamically changing. Transaction redaction and validation need the help of AAs and miners, and there are no requirements for any centralized decision-making in the phase of redacting. Therefore, it is hard for any AA to control redaction in the centralized way unless it can collude the rest of AAs.

On the other hand, the redaction policy is determined by the data owner. That is, the policy of each transaction is independently set. The policy structure and attributes are predefined in the phase of initialization, which may require some centralized decision-making. In order to enhance the decentralization and security, DCP-ABE requires that each attribute of the policy structure is controlled only once by the AA. If the attribute is used multiple times, it is labeled with a fresh value j, i.e., Att : j. Thus, the attributes of the policy structure are controlled in a decentralized way, differing from the traditional way where all attributes are controlled by a single trust AA. Besides, the one-use rule also preserves the AAs from tampering with data unless all AAs collude. In summary, SDR-chain achieves a high degree of decentralized control of the redaction.

Dynamic Support. Based on the algorithm Delegate, no matter what role the node plays in the network, (s)he can delegate the role to others when (s)he leaves. Moreover, any node running the blockchain protocol can play the delegated role. Due to the algorithm Delegate satisfying the backward/forward security, the generation and redaction are not affected by nodes leaving in the SDR-chain. Thus, the SDR-chain can support dynamic nodes.

Scalability. The SDR-chain neither requires to re-compute the PoW puzzle of the redacted block nor waits for a long voting period. A valid redacted transaction is required to be broadcast as the regular transaction. SDR-chain can support a large scale redaction requests, so it is scalable.

Historic Security. In the proposed SDR-chain, DACH is applied to hash the transactions. The historic security of transactions is affected by the indistinguishability of DACH directly. As the description in Theorem 1, the DACH satisfies the indistinguishability. It is hard for adversaries to distinguish the adapted hash from the fresh hash. That is, the original transaction and the redacted transaction are indistinguishable in SDR-chain which is based on DACH. Besides, DACH is collision-resistant as the descriptions of

TABLE 2
Complexity Comparison of Algorithms

Schemes	KeyGen				Hash					Verify			Adapt				
	GExp	GMul	BP	H	GExp	GMul	BP	H	SE	GExp	GMul	H	GExp	GMul	BP	H	SE
PCH [13]	24	16	-	12	$7+6l+9k$	$4+3l+6kl$	-	$3+6l+6k$	1	2	2	2	$7+6i+6l+9kl$	$10+6i+6l+6kl$	6	$3+6l+6k$	1
DPCH [17]	4	2	2	3	$3+6l$	$3+2l$	$1+2l$	$l+2k$	1	2	2	2	$3+i+6l$	$3+4i+2l$	$1+3i+2l$	$4+2l+2k$	1
DACH	2	1	-	1	$2+5l$	$2+2l$	$1+2l$	1	-	1	1	1	$1+i$	$1+2i$	$2i$	1	-

* The complexity is evaluated by the number of various operations in group. GExp: group exponentiation; GMul: group multiplication; BP: bilinear pairing operation; SE: symmetric encryption; H: hash; l: the rows of the access matrix; k: the columns of the access matrix; i: the number of attributes satisfying the access policy and $i \leq l$.

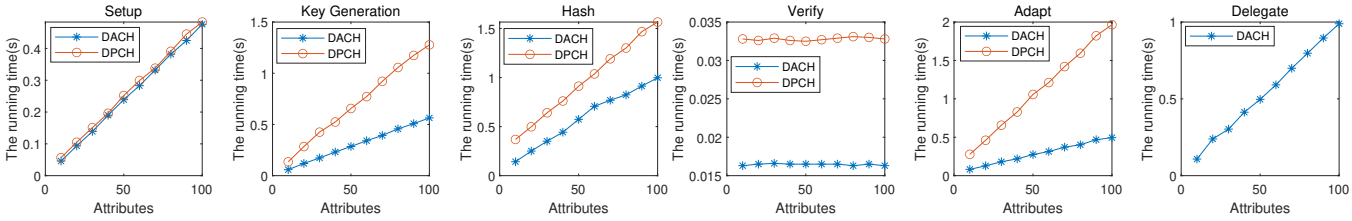

Fig. 6: The running time of our DACH and the compared DPCH [17]. Both of them are decentralized solutions.

TABLE 3
Size Comparison with Prior Schemes

Schemes	Key Size		Hash Size	Ciphertext Size			
	$\mathbb{G}$	$\mathbb{G}_1$	$\mathbb{Z}_N^*$	$\mathbb{G}$	$\mathbb{G}_1$	$\mathbb{G}_T$	$\mathbb{Z}_N^*$
PCH [13]	$3(i+1)$	3	4	$3l$	3	2	-
DPCH [17]	$2i$	-	4	$3l$	-	$1+l$	1
DACH	i	-	2	$2l$	-	$1+l$	-

* The size of keys, hashes, and ciphertexts is evaluated by the number of elements in groups $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_T$, and $\mathbb{Z}_N^*$, where the size of an element of $\mathbb{G}_1$ is 3 times that of $\mathbb{G}$. l: the rows of the access matrix; i: the number of attributes satisfying the access policy and $i \leq l$.

TABLE 4
Increment Size of the Redacted Transactions

Name	Block ID	Transaction ID	Deletion L Bytes	Increment Δ Bytes
Tx1	268060	$d29c9c0e8e4d2a9790922af73f0b8d51$ $f0bd4bb19940d9cf910ead8fbe85bc9b$	983	-855
Tx2	251768	$59bd7b2cff5da929581fc9fef31a2fba$ $14508f1477e366befb1eb42a8810a000$	95	$+33$
Tx3	475199	$be0f09e9a6671520808 2afdad71c4c94$ $45328788b49a6475e1301bab5345126b$	0	$+128$

* The data required to delete is stored by OP_RETURN of Bitcoin transactions.

Theorem 2 and Theorem 3, so it is hard for adversaries to reconstruct any information about the history of the redacted transaction. Moreover, the adversaries cannot deduce other collisions to tamper with transactions even though the collision is public. In summary, these properties ensure the historic security of SDR-chain.

Backward/Forward Security. In the terms of backward/forward security, SDR-chain is inherited from DACH. As the description of Theorem 5, the algorithm Delegate satisfies the backward/forward security. After nodes leave from the network, redaction can be securely performed in SDR-chain.

To sum up, SDR-chain satisfies the defined design goals. It can achieve controlled, scalable, decentralized, dynamic support, and secure redactions.

7 Evaluation

In this section, we implement our scheme and evaluate the performance of our scheme by comparing with the previous work.

7.1 Performance Comparison

As shown in TABLE 2, we compare our DACH with the previous PCH [13] and DPCH [17], which are policy-based chameleon hash functions. We mainly compare the complexity of their group operations because operations on group are significantly expensive. The comparison results show that our DACH has lower complexity than PCH and DPCH, where the complexities of algorithm KeyGen are evaluated for each attribute. Besides, the complexity of algorithm Delegate of our DACH is $4\mathsf{GExp} + 3\mathsf{GMul} + 4\mathsf{BP} + \mathsf{H}$.

TABLE 3 shows the size comparison results of keys, hashes, and ciphertexts, where hashes include (h, r) and ciphertexts ct are the result of encrypting the trapdoor key. From the comparison results, the sizes of our key, hash, and ciphertext are much smaller than all the schemes. That is, our DACH has lower overhead in communication and storage than DCH [13] and DPCH [17].

The communication complexity is measured in bits sent by honest redactors. In our protocol, the redactor needs to communicate with the AAs for a κ bits key $SK_{i,GID}$, so the network/communication complexity is $O(m\kappa)$ bits, where m is the number of AAs. Besides, the redacted transaction will cost the communication overhead consisting of the communications to broadcast the γ bits redacted transaction, i.e, $O(n^2\gamma)$ bits, where n $(n > m)$ is the number of nodes in the Bitcoin network. Thus, the total communication complexity is $O(n^2\gamma + m\kappa)$ bits, i.e, $O(n^2)$.

The practical cost of applying the DACH to Bitcoin to erase history is analyzed as follows. The cost is mainly brought by broadcasting the redactable transaction and the redacted transaction. In the phase of broadcasting the redactable transaction, the redactable transaction needs to store the randomness r, where r is 128 bytes in our experiments. Let the size of the original Bitcoin transaction Tx be $|\mathrm{Tx}|$ bytes. Then, the size of the redactable transaction is $|\mathrm{Tx}| + 128$ bytes. Therefore, the additional communication cost required by a node with respect to the immutable Bitcoin consists of the communication cost required to broadcast the randomness r, i.e., 128 bytes. In the phase of broadcasting the redacted transaction, the data stored in the field OP_RETURN may be deleted as shown in TABLE 4. The size of the redacted transaction is $|\mathrm{Tx}| + 128 - L$ bytes, where L is the size of the deleted data. Then, the increment

Δ of the size of the transaction caused by redaction operations with respect to the original transaction is $\Delta = 128 - L$ bytes. For example, Tx1 is deleted by 983 bytes and stores 128 bytes r, so the increment is -855 after being redacted, meaning that the redaction operation reduces the size of Tx1 by 855 bytes. In the worst case such as Tx3, there is no data to be deleted, so the increment is $+128$ bytes, meaning that the size of Tx3 is increased by 128 bytes. The maximum increment size is constrained by the size of randomness r. Therefore, the cost of broadcasting the redacted transaction is low. In summary, applying DACH to Bitcoin will result in minor communication costs.

7.2 Implementation

To simulate the effectiveness of our DACH, we implement DACH and evaluate running time on a Macbook Pro laptop with 2.4 GHz Intel Core i5, 8 GB RAM, and 512 SSD. We compile in Python language and use Charm-Crypto 0.5 [37] which is based on PBC library, GMP library, and OpenSSL library for all algebraic operations and communications. In the implementation, we use SS512 curve with a 512-bit base field for pairing. The hash functions mapping any input to the group are provided the Charm-Crypto framework. Fig. 6 has shown the comparison results of our DACH and DPCH [17] in terms of running time. In the first five sub-figures, the x-axis means the number of attributes that satisfy the policy. In the last sub-figure, the x-axis means the number of attributes that are delegated. Besides, the delegate algorithm is proposed in redactable blockchains for the first time so there is no comparison test.

8 Related Work

This section reviews the related work from two categories.

Cryptography-Based. In 2017, Ateniese et al. [12] have first argued the importance and necessity of rewriting history in blockchain and proposed a feasible scheme based on the chameleon hash (CH) [28]. In this scheme, the traditional SHA256 is replaced by the CH fuction and a block-level redaction can be performed by computing the CH collisions for the redacted block. The trapdoor key of CH is managed by a full trusted authority in a centralized setting or shared among some fixed users in a decentralized setting.

To achieve a fine-grained and controlled redaction, Derler et al. [13] have proposed a secure transaction-level scheme in 2019. Based on a ciphertext-policy attribute-based encryption (CP-ABE) [38], Derler's scheme manages the key used for redactions with an access policy encoded over attributes. The user whose attributes satisfy the policy can redact the transaction. Moreover, attributes are verified by a full trusted authority. Huang et al. [14, 15] manage the key of CH by associating an identity. They have built redactable blockchains by proposing a threshold chameleon hash (TCH) for IIoT [14] and a revocable chameleon hash (RCH) for IoT [15], respectively. In their schemes, the redaction privilege is fixed to a party holding the identity.

To achieve decentralized key management, Ma et al. [17] have proposed a decentralized policy-based CH by applying a multi-authority ABE scheme. The keys used for redactions are controlled by multiple authorities. After that, Samelin et al. [39] have proposed a policy-based sanitizable signature (P3S) to guarantee signature validity of redacted blockchain in 2020. According to the hash-then-sign paradigm, the P3S scheme makes that the signature of the redacted transaction is the same as the original one. Besides, Xu et al. [16] have proposed a k-time modifiable and epoch-based redactable blockchain based on CH and digital signature. This scheme mitigates and penalizes malicious behaviors by limiting the rewriting privilege and making a time-locked deposit.

In this work, we focus on security and functionality in CH-based redactable blockchain. The proposed SDR-chain achieves a fine-grained and decentralized access control of redactions. Besides, the SDR-chain can support dynamic and ensure forward/backward security when any authority is offline. Thus, this work is regarded as a better solution to blockchain redacting than the previous works.

Non-Cryptography-Based. Puddu et al. have first proposed μchain [8] to implement redaction by issuing new version transactions. Redaction is constrained by a time window and the historic versions are hided by encryption. The one holding the secret keys can recover the original transaction. Besides, μchain needs to mutate all affected transactions causing by modification to tackle transaction consistency. As a result, the links between blocks are broken.

To support transaction deletion, Dorri et al. [9] have proposed a MOF-BC scheme to flexibly manage IoT data by issuing a removal/summary transaction. MOF-BC ensures blockchain consistency by recording all states of removed/summarized transactions. Therefore, the bits of the removal/summary transaction are usually much longer.

To maintain block connectivity, Deuber et al. [10] have proposed a consensus-based voting scheme, which maintains links between blocks by storing the old state of the redacted block in the block header. Nodes vote on the final state of the redacted block on-chain during a period (1024 consecutive blocks, i.e., over a week). This scheme is apparently significant in the redactable blockchain ecosystem but it is impractical to frequently modify the block.

To make the redaction solution compatible with various blockchains such as Bitcoin and Ethereum, Thyagarajan et al. [11] have proposed Reparo which storing old states in a database as a data repair layer on top of the blockchain. Li et al. [40] have proposed a redactable blockchain protocol supporting various network environment such as POW and POS. Beside, the sequential works focus on local redaction [41] and applications [42].

In this work, the proposed SDR-chain neither needs to store the old state of the redacted block nor re-computes the PoW puzzle for the redacted block. Without the long voting period, the SDR-chain does not suffer from consensus delay. Moreover, the SDR-chain achieves historic security which prevents the modified history from being reconstructed.

9 Conclusion

In this paper, we have tackled the issues of redacting transaction history in the blockchain and proposed the SDR-chain. We first propose a dynamic and decentralized attribute-based chameleon hash (DACH) function to make blockchain redactable in a decentralized and controlled way. In DACH-based SDR-chain, redactors can redact the historic

transactions by computing the DACH collisions. The redaction is controlled by a group of dynamic nodes instead of a single full trusted center. Besides, the designed delegation algorithm of DACH achieves the functionality and security, where any authority can leave and join the network securely. According to the security and performance analysis, our DACH is secure and has low computation complexity and communication cost, and our DACH-based SDR-chain satisfies the designed goals. In summary, our SDR-chain can achieve dynamic support, scalable, and secure transaction-level redactions in a fine-grained control and decentralized way.

ACKNOWLEDGMENTS

This work is supported in part by the National Key R&D Program of China (No. 2022YFB3103500), in part by National Natural Science Foundation of China (Grant no. 61932006, 62202071, U20A20176, 62072062), in part by China Postdoctoral Science Foundation (Grant no. 2022M710520, 2022M710518), in part by Natural Science Foundation of Chongqing, China (Grant no. CSTB2022NSCQ-MSX1217, CSTB2022NSCQ-MSX0358, cstc2022ycjh-bgzxm0031), in part by Special Foundation for Chongqing Postdoctor, China (Grant no.2021XM2030), in part by Sichuan Science and Technology Program (Grant no. 2021YFQ0056), and in part by Natural Science Foundation (Grant no. CNS-2153393).

REFERENCES

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008.

[2] K. Gai, Y. Zhang, M. Qiu, and B. Thuraisingham, "Blockchain-enabled service optimizations in supply chain digital twin," *IEEE Transactions on Services Computing*, pp. 1–12, 2022.

[3] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–27, 2020.

[4] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3548–3558, 2019.

[5] P. Taylor, "Global blockchain solutions spending 2017-2024," 2022. [Online]. Available: https://www.statista.com/statistics/800426/worldwide-blockchain-solutions-spending/.

[6] D. Zhang, J. Le, N. Mu, and X. Liao, "An anonymous off-blockchain micropayments scheme for cryptocurrencies in the real world," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 32–42, 2020.

[7] H. Wang, Y. Liu, Y. Li, S.-W. Lin, C. Artho, L. Ma, and Y. Liu, "Oracle-supported dynamic exploit generation for smart contracts," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1795–1809, 2022.

[8] I. Puddu, A. Dmitrienko, and S. Capkun, "μchain: How to forget without hard forks." *IACR Cryptology ePrint Archive*, vol. 2017, p. 106, 2017.

[9] A. Dorri, S. S. Kanhere, and R. Jurdak, "Mof-bc: A memory optimized and flexible blockchain for large scale networks," *Future Generation Computer Systems*, vol. 92, pp. 357–373, 2019.

[10] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," in *IEEE Symposium on Security and Privacy (S&P)*, 2019, pp. 124–138.

[11] S. A. K. Thyagarajan, A. Bhat, B. Magri, D. Tschudi, and A. Kate, "Reparo: Publicly verifiable layer to repair blockchains," in *International Conference on Financial Cryptography and Data Security (FC)*, 2021, pp. 37–56.

[12] G. Ateniese, B. Magri, D. Venturi, and E. Andrade, "Redactable blockchain – or – rewriting history in bitcoin and friends," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017, pp. 111–126.

[13] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based." in *26th Annual Network and Distributed System Security Symposium (NDSS)*, 2019.

[14] K. Huang, X. Zhang, Y. Mu, X. Wang, G. Yang, X. Du, F. Rezaeibagha, Q. Xia, and M. Guizani, "Building redactable consortium blockchain for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3670–3679, 2019.

[15] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, X. Du, and N. Guizani, "Achieving intelligent trust-layer for internet-of-things via self-redactable blockchain," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2677–2686, 2020.

[16] S. Xu, J. Ning, J. Ma, X. Huang, and R. H. Deng, "K-time modifiable and epoch-based redactable blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4507–4520, 2021.

[17] J. Ma, S. Xu, J. Ning, X. Huang, and R. H. Deng, "Redactable blockchain in decentralized setting," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1227–1242, 2022.

[18] D. Zhang, J. Le, X. Lei, T. Xiang, and X. Liao, "Exploring the redaction mechanisms of mutable blockchains: A comprehensive survey," *International Journal of Intelligent Systems*, vol. 39, no. 6, pp. 5051–5084, 2021.

[19] General Data Protection Regulation. [Online]. Available: https://gdpr-info.eu.

[20] K. Shirriff, "Hidden surprises in the bitcoin blockchain and how they are stored: Nelson mandela, wikileaks, photos, and python software," 2014. [Online]. Available: http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html#ref8.

[21] R. Matzutt, J. Hiller, M. Henze, J. H. Ziegeldorf, D. Müllmann, O. Hohlfeld, and K. Wehrle, "A quantitative analysis of the impact of arbitrary blockchain content on bitcoin," in *International Conference on Financial Cryptography and Data Security (FC)*, 2018, pp. 420–438.

[22] C. K. Pyoung and S. J. Baek, "Blockchain of finite-lifetime blocks with applications to edge-based iot," *IEEE Internet of Things Journal*, 2019.

[23] Y. Lu, J. Zhang, Y. Qi, S. Qi, Y. Zheng, Y. Liu, H. Song, and W. Wei, "Accelerating at the edge: A storage-elastic blockchain for latency-sensitive vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2021.

[24] G. Tian, Y. Hu, J. Wei, Z. Liu, X. Huang, X. Chen, and W. Susilo, "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

[25] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009–2030, 2020.

[26] Y. Zhang, K. Gai, J. Xiao, L. Zhu, and K.-K. R. Choo, "Blockchain-empowered efficient data sharing in internet of things settings," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3422–3436, 2022.

[27] C. Lin, X. Huang, and D. He, "Ebcpa: Efficient blockchain-based conditional privacy-preserving authentication for vanets," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2022.

[28] H. Krawczyk and T. Rabin, "Chameleon signatures," in

Proceedings of the Network and Distributed System Security Symposium (NDSS), 2000.
[29] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2011, pp. 568–588.
[30] A. Beimel, "Secure schemes for secret sharing and key distribution," in *PhD thesis*, 1996.
[31] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2011, pp. 568–588.
[32] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors," in *IACR International Workshop on Public Key Cryptography (PKC)*, 2017, pp. 152–182.
[33] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-rsa-inversion problems and the security of chaum's blind signature scheme," *Journal of Cryptology*, vol. 16, no. 3, pp. 185–215, 2003.
[34] G. Ateniese and B. de Medeiros, "On the key exposure problem in chameleon hashes," in *International Conference on Security in Communication Networks (SCN)*, 2004, pp. 165–179.
[35] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "Modeling the impact of network connectivity on consensus security of proof-of-work blockchain," in *IEEE Conference on Computer Communications (INFOCOM)*, 2020, pp. 1648–1657.
[36] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2015, pp. 281–310.
[37] Charm-Crypto 0.5. [Online]. Available: https://github.com/JHUISI/charm.
[38] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2007, pp. 321–334.
[39] K. Samelin and D. Slamanig, "Policy-based sanitizable signatures," in *Cryptographers' Track at the RSA Conference (CT-RSA)*, 2020, pp. 538–563.
[40] X. Li, J. Xu, L. Yin, Y. Lu, Q. Tang, and Z. Zhang, "Escaping from consensus: Instantly redactable blockchain protocols in permissionless setting," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–20, 2022.
[41] R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, and K. Wehrle, "How to securely prune bitcoin's blockchain," in *IFIP Networking Conference (Networking)*, 2020, pp. 298–306.
[42] J. Xu, K. Xue, H. Tian, J. Hong, D. S. L. Wei, and P. Hong, "An identity management and authentication scheme based on redactable blockchain for mobile networks," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.

Di Zhang received the Ph.D. degree degree in intelligent computing and information processing from Southwest University, Chongqing, China, 2021. From December 2018 to December 2020, she was a visiting scholar at Virginia Polytechnic Institute and State University. She is currently a research assistant with the College of Computer Science, Chongqing University. Her research interests include applied crypto, cloud computing security and blockchain.

Junqing Le received the B.S. degree in software engineering from Southwest Jiaotong University, Chengdu, China, in 2014, and the M.S. degree in signal and information processing and Ph.D. degree in intelligent computing and information processing from Southwest University, Chongqing, China, in 2017 and 2021, respectively. From May 2019 to May 2021, he was a visiting scholar at George Mason University. At present, he is a research assistant with the College of Computer Science, Chongqing University. His research interests include privacy protection, privacy machine learning, cloud computing security, and blockchain.

Xinyu Lei received the Ph.D. degree with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, in 2021. He is currently an Assistant Professor with the Department of Computer Science, Michigan Technological University, Houghton, MI, USA. He worked as a Research Assistant with the Texas A&M University at Qatar, Doha, Qatar, in 2013. In 2017, he worked as a Research Intern with Ford Motor Company, Dearborn, MI, USA. His current research focuses on machine learning and cybersecurity.

Tao Xiang received the B.Eng., M.S., and Ph.D. degrees in computer science from Chongqing University, Chongqing, China, in 2003, 2005, and 2008, respectively. He is currently a Professor with the College of Computer Science, Chongqing University. He has published over 100 papers on international journals and conferences. He also served as a referee for numerous international journals and conferences. His research interests include multimedia security, cloud security, and privacy and cryptography.

Xiaofeng Liao received the BS and MS degrees in mathematics from Sichuan University, Chengdu, China, in 1986 and 1992, respectively, and the PhD degree in circuits and systems from the University of Electronic Science and Technology of China in 1997. At present, he is a professor at Chongqing University and the Dean of College of Computer Science. He is also a Yangtze River Scholar of the Ministry of Education of China. From November 1997 to April 1998, he was a research associate at the Chinese University of Hong Kong. From October 1999 to October 2000, he was a research associate at the City University of Hong Kong. From March 2001 to June 2001 and March 2002 to June 2002, he was a senior research associate at the City University of Hong Kong. From March 2006 to April 2007, he was a research fellow at the City University of Hong Kong. Professor Liao is associate editors for IEEE Transactions on Cybernetics and IEEE Transactions on Neural Network and Learning Systems, and he holds 4 patents, publishes 4 books and over 300 international journal and conference papers. His current research interests include neural networks, nonlinear dynamical systems, cryptography, and privacy protection.