Received: 14 February 2021 | Revised: 3 May 2021 | Accepted: 11 May 2021

DOI: 10.1002/int.22502

RESEARCH ARTICLE

WILEY

Exploring the redaction mechanisms of mutable blockchains: A comprehensive survey

Di Zhang[1] | **Junqing Le**[1] | **Xinyu Lei**[2] | **Tao Xiang**[3] | **Xiaofeng Liao**[3]

[1]College of Electronic and Information Engineering, Southwest University, Chongqing, China

[2]Department of Computer Science and Engineering, Michigan State University, Michigan, United States

[3]Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, and College of Computer Science, Chongqing University, Chongqing, China

Correspondence
Xiaofeng Liao, Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, and College of Computer Science, Chongqing University, 400044 Chongqing, China.
Email: xfliao@cqu.edu.cn

Funding information
National Key Research and Development Program of China, Grant/Award Number: 2018AAA0100101; National Natural Science Foundation of China, Grant/Award Numbers: 61932006, 61772434; Chongqing Technology Innovation and Application Development Project, Grant/Award Number: cstc2020jscx-msxmX0156

Abstract

Blockchain technology has attracted tremendous interest from both industry and academia. It is typically used to record a public history of transactions (e.g., payment/smart contract data), but storing nonpayment/contract data in transactions has been common. The ability to store data unrelated to payment/contract such as illicit data on blockchain may be abused for malicious purposes. For example, one may use blockchain to store the data related to child pornography and copyright violations, which are publicly visible and immutable. Moreover, an immutable blockchain is not suitable for all blockchain-based applications. So far, numerous redaction mechanisms for the mutable blockchain have been developed. In this paper, we aim at conducting a comprehensive survey that reviews and analyzes the state-of-the-art redaction mechanisms. We start by giving a general presentation of blockchain and summarize the typical methods of inserting data in blockchain. Next, we discuss the challenges of designing the redaction mechanism and propose a list of evaluation criteria. Then, redaction mechanisms of the existing mutable blockchains are systemically reviewed and analyzed based on our evaluation criteria. The analyses include algorithmic overviews, performance limitations, and security vulnerabilities. Finally, the comparisons and analyses provide new insights into

these mechanisms. This survey will provide developers and researchers a comprehensive view and facilitate the design of future mutable blockchains.

KEYWORDS
chameleon hash, consensus mechanism, data insertion, privacy leakage, redactable blockchain

1 | INTRODUCTION

Recently, the blockchain technology has gained intensive research interests from both academia and industry. It has been adopted by various practical applications, such as copyright dispute resolutions,[1] product traceability,[2] electronic voting,[3] storage services,[4] healthcare services,[5] tracking products along supply chain,[6] and data management[7] in Internet-of-Things (IoT).[8] A recent study[9] shows that the worldwide spending on blockchain solutions is expected to grow from 1.5 billion dollars annually in 2018 to an estimated 15.9 billion dollars annually by 2023.

Blockchain maintains a distributed ledger of transactions (e.g., payment/smart contract data[10,11]) without any trusted central authority. It is typically managed by a peer-to-peer (P2P) network with multiple participating nodes. Blockchain allows each participating node independently appending new transactions while ensuring that participating nodes agree on a unified history of transactions. Transactions are stored securely and transparently based on cryptographic primitives. Immutability is a definitive feature of the traditional blockchain technology. It guarantees the data integrity (any transaction cannot be modified) and hence facilitates the data auditing process.

Nevertheless, it is also desirable to develop techniques to support redacting data in blockchains due to the three reasons. The first reason is that the immutability feature can be abused for malicious purposes, such as to store and disseminate illicit data/code. For example, Matzutt et al.[12] have found at least 8 files related to sexual content stored in Bitcoin blockchain, two of them containing 274 links to child pornography, 142 of which refer to darknet services. Erasing these illicit objects stored in the blockchain may be an important requirement and legally obliged for law enforcement agencies like Interpol.[13] The second reason is that the immutable blockchain cannot capture all emerging blockchain-based applications. More and more blockchain-based applications call for a certain degree of flexibility for data redaction. Examples include but not limited to (i) The data stored on blockchain may relate to users' privacy sensitive data (e.g., healthcare and insurance records).[14] The users may want to erase their sensitive data from the blockchain and update data whenever needed (e.g., an amending contract service).[15] (ii) It is required to delete unnecessary data to save space in IoT-based blockchain systems.[16-19] The third reason is that the requirement for redactable blockchain has also stimulated by regulations. According to the European General Data Protection Regulation (GDPR),[20] data of users have a "right to be forgotten," that is, any user can erase her/his personal data and copies. Obviously, the data anonymously appended on blockchain should also have the right to be rewritten.[21] To sum up, redacting data in blockchain under specific circumstances has evolved to an important requirement.

At present, numerous mechanisms have been developed to provide some degree of flexibility for data redaction on blockchains.[22] In this paper, we aim at conducting a comprehensive

survey for these existing redaction mechanisms. According to the design methodology of these redaction mechanisms, the state-of-the-art mutable blockchains can be divided to four categories: (i) consensus-based, (ii) chameleon hash-based, (iii) *meta*-transaction-based, and (iv) pruning-based, as summarized in Table 1. Then we abstract algorithmic overviews and analyze security vulnerabilities and performance limitations of the four types of mutable blockchain, respectively. The related work of this survey includes academic papers, white papers, and official documents.

First, the general structure and features of blockchain are reviewed. To analyze the threats of inserting arbitrary data in blockchain, we investigate where and what kind of data can be inserted. As a consequence, we figure out the challenges to be addressed when designing a redaction mechanism and propose a list of evaluation criteria to analyze the existing redaction mechanisms. Next, we give the general procedures of the four categories of mechanisms and discuss their advantages and disadvantages. For each mechanism, we give a brief overview and detailed analyses. Especially, the potential vulnerabilities and limitations that may be caused by the mechanism are analyzed. Then, we comprehensively compare the existing mechanisms based on the evaluation criteria and analyses. Finally, based on our detailed analyses and discussions, we present further challenges and research directions. Our survey will provide developers and researchers a comprehensive view and facilitate the design of the future mutable blockchains.

In summary, we have three main contributions in this survey.

(1) We provide an overview of the existing redaction mechanisms. We divide the state-of-the-art mechanisms into four categories. For mechanisms in each category, we analyze their security and performance in detail.
(2) We figure out the challenges of redacting data on the blockchain and propose a list of evaluation criteria to analyze the security and performance of mutable blockchains. The proposed criteria can be used for evaluating future proposals.
(3) We compare these mutable blockchains comprehensively and give out further research directions.

The remaining part of this survey is organized as follows. The following Section 2 reviews the framework and features of blockchain. The content insertion methods and the threats of malicious data insertion are discussed in Section 3. Next, the design challenges and evaluation criteria for redaction mechanisms are summarized in Section 4. Then, the reviews of algorithms and analyses of mechanisms are mainly presented from four categories in Section 5. Section 6

TABLE 1 Summary of redaction mechanisms in this survey

Category	Work	Key mechanism
Consensus-based	[23-25]	Rely on a rule to agree on a new state of the redacted transaction/block by voting-based consensus.
Chameleon Hash-based	[26-34]	The state of the redacted transaction/block remains intact based on the chameleon hash function.
Meta-transaction-based	[35,36,16]	Generate a new version transaction to replace the previous transaction.
Pruning-based	[17,37]	Delete transactions/blocks when a predefined condition (e.g., lifetime) is satisfied.

focuses on comparisons of performance and security for the representative mutable blockchains, and discusses the future research directions. Finally, Section 7 concludes this paper.

2 | OVERVIEW OF BLOCKCHAIN PROTOCOL

In this section, a background of the blockchain system is concisely reviewed to introduce the fundamental structure, key features, and security requirements. Note that the typical Bitcoin blockchain is taken as an example to present the description.

2.1 | Blockchain background

Blockchain is a distributed ledger recording a growing list of transactions and generated by a group of nodes who make up a P2P network. These nodes maintain a unified chain history according to the consensus protocol. In this section, we first introduce the fundamental component (i.e., nodes and network), and then present the data structure of the transaction and the block. Finally, the consensus protocol is concisely introduced.

2.1.1 | Node and network

Nodes can be any computing devices connected to the blockchain network. They propagate transactions and blocks through the P2P network. A node can be a user who exchanges assets via receiving and sending transactions, and also become a miner who is in charge of generating new blocks (i.e., mining) subject to a consensus protocol. Moreover, motivated by the block rewards and transaction fees,[38] miners continuously generate the new blocks to maintain the system. The contribution of a miner to the blockchain is proportional to its possession of resources, such as computation power, token wealth, and storage space. In the network, miners collect and verify transactions to maintain the blockchain. Especially, *full nodes* store complete blockchain starting from the genesis block, and fully verify all rules of the consensus protocol to ensure blockchain data integrity. Alternatively, *lightweight nodes* store the headers of all blocks, and only verify the authenticity of transactions.

With respect to the access control of the blockchain network, there are two main categories: *permissionless* and *permissioned* blockchain. In the permissionless setting, anyone are allowed to flexibly join and leave blockchain network without any authorization. Moreover, each node does not have any trusted assumptions, and generates transactions and blocks using a valid pseudonym. But in a permissioned blockchain, nodes need to be authorized and even reveal identity. A minority of nodes are usually elected to verify transactions and generate blocks.

2.1.2 | Transaction

A transaction mainly consists of inputs and outputs[39] which have defined the origin (from the sender) and destination (to the recipient) of the transaction, respectively. Each transaction references one or more previously unspent transaction outputs. Specifically, each transaction may have one or more inputs, and each input is a reference to an output of the previous

transaction. Moreover, each output can only ever be referenced at one time. If an output has been spent repeatedly, the subsequent transaction referring to this output is in conflict and invalid. The conflict transactions can be easily detected according to the timestamps.

The input script is composed of a previous transaction hash TxID, an output index n, a field *ScriptSig*, the length of *ScriptSig*, and a sequence number. The output script contains an amount of Bitcoin, a field *ScriptPubKey*, and the length of *ScriptPubKey*. As shown in Table 2, the script structures of four typical transactions are presented, containing data and operands. For example, the field *ScriptSig* of the P2PKH transaction contains a signature and the public key of the sender. The signature is computed by the private key of the sender, and verified to check transaction integrity and ownership by the public key. The public key is encoded as a field *ScriptPubKey* of the output script in an unspent transaction which the sender received previously. It is also encoded as an address in Bitcoin blockchain, providing the anonymity of the sender so that the sender and recipient can transact online without revealing their true identities. Moreover, the field *ScriptSig* of transaction Tx1 should be verified by the field *ScriptPubKey* of transaction Tx0, that is, the sender of transaction Tx1 is able to spend the unspent output **Out**[**0**] of Tx0 only when the sender possesses the private key that can match the public key. As a result, transactions appended on the blockchain are connected to digital signature. As shown in Figure 1, Tx0 → Tx1 → Tx3 can be regarded as a signature chain. The signature chains make that the history of all transactions can be tracked in the blockchain system. Moreover, they are usually verified to check the ownership and integrity of transactions.

A transaction will undergo five processes in the network[40] before it is appended to the blockchain. As shown in Figure 2, the transaction is first generated by users (or miners). Then it will be propagated and verified in the P2P network. The detailed processes are presented as follows.

- *Generation.* A sender generates and signs a transaction that is paid to a recipient.
- *Transaction propagation.* The transaction is propagated in the P2P network and then received by all nodes.
- *Validation.* After nodes receive transactions, they verify transactions according to the consensus protocol. If a transaction is valid, it is temporarily stored in the mempools of nodes. Then the valid transactions are integrated into a Merkle tree to generate a new block. Otherwise, the invalid transactions are discarded.

TABLE 2 Script structures of four typical transactions

	ScriptPubKey	***ScriptSig***
P2PKH	OP_DUP OP_HASH160 ⟨*PubKeyHash*⟩ OP_EQUALVERIFY OP_CHECKSIG	⟨*Signature*⟩⟨*PubKey*⟩
P2SH	OP_HASH160 ⟨*ScriptHash*⟩ OP_EQUAL	⟨*Signatures*⟩{*RedeemScript*}
	ScriptPubKey	***Witness***
P2WPKH	0 $\langle PubKeyHash \rangle_{20Byte(HASH160)}$	⟨*Signature*⟩⟨*PubKey*⟩
P2WSH	0 $\langle RedeemScript \rangle_{32Byte(HASH256)}$	⟨*Signatures*⟩{*RedeemScript*}

Note: The P2SH and P2WSH can be the M-of-N multisignature transaction.

Abbreviations: ⟨*Signatures*⟩, including M signatures; {*RedeemScript*}, containing N PubKeys; P2PKH, Pay-to-PubkeyHash; P2SH, Pay-to-ScriptHash; P2WPKH, Pay-to-Witness-PubkeyHash; P2WSH, Pay-to-Witness-ScriptHash.

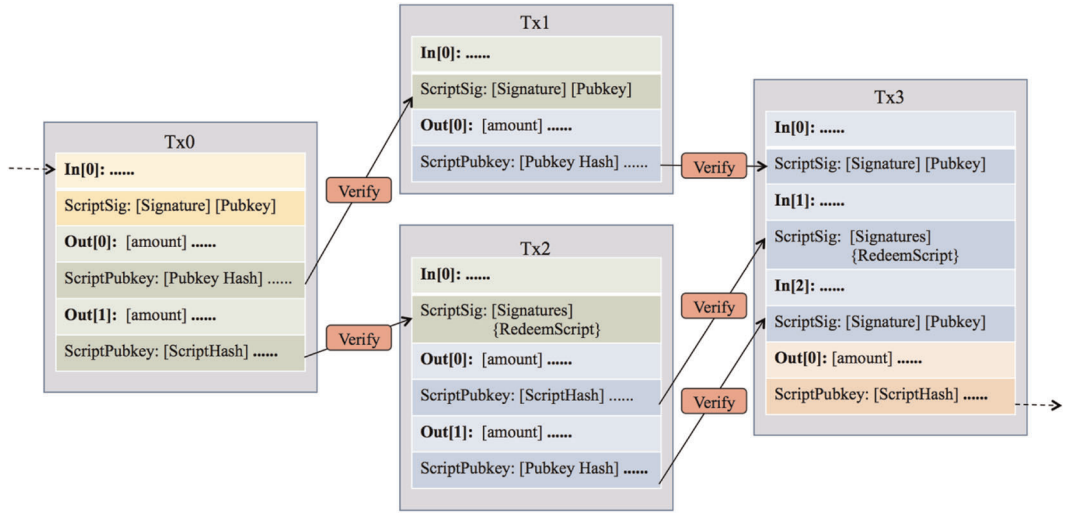

FIGURE 1 An example showing the references and verification between typical P2PKH and P2SH transactions. Transaction Tx1's input is a reference to the output **Out[0]** of transaction Tx0. The public key *pubkey* of the transaction Tx1's can verify Tx1's signature and its hash value should match the transaction Tx0's *ScriptPubkey*. Bitcoins are payed to transaction Tx3 along with the chain of Tx0 → Tx1(Tx2) → Tx3 [Color figure can be viewed at wileyonlinelibrary.com]

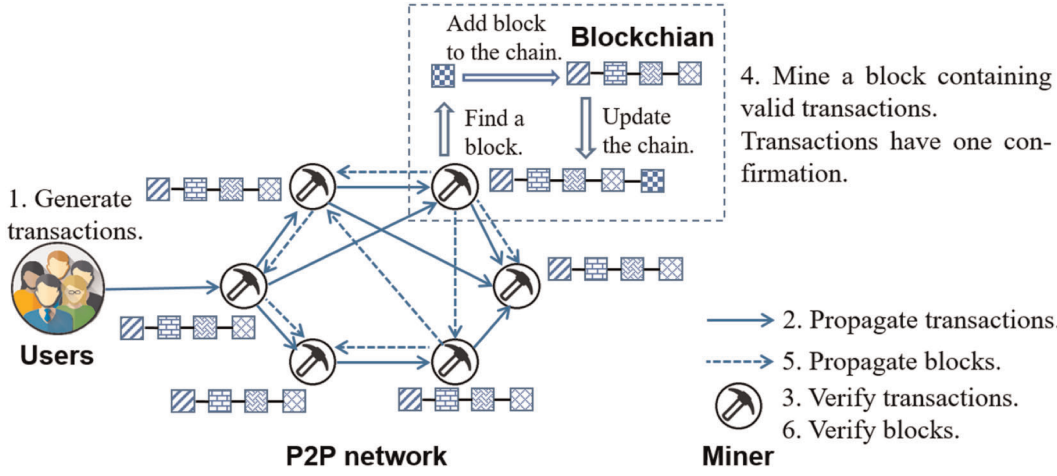

FIGURE 2 The processes of transactions in blockchain. The valid transactions are propagated in the network twice and appended to the blockchain finally [Color figure can be viewed at wileyonlinelibrary.com]

- *Block propagation.* After the new block containing a set of the valid transactions is found, it is also propagated in the P2P network. Nodes verify the new block. If the new block is valid, nodes update their own replica of the chain and mine on this chain. The propagation is subject to unexpected network delays, which will affect blockchain security. The long network delays will increase the occurrence of forks and lead to hashing power wasted and miner's losses.
- *Confirmation.* Once a transaction is contained by a block, it has one confirmation. After nodes receive the block, they verify the block according to the consensus protocol. If it is

valid, nodes will attach it to the blockchain and mine on this chain. As soon as another new block is mined on the same chain, the transaction has two confirmations, and so on. The confirmations of the transaction are more, the probability of being discarded is lower. Besides, the confirmation time is affected by the block-size[41] and network connectivity.[42]

Finally, once the transaction is validated and appended on a block, it has one *confirmation*. The number of transaction confirmations creases with the subsequent blocks being mined. The more the confirmations are, the lower the probability of the transaction being discarded is.

2.1.3 | Block

A block is generated to record transactions by miners. The generation of the block is subjected to the consensus protocol. For example, the block of the typical Bitcoin is defined by the Nakamoto consensus protocol[43] (also called as proof-of-work). In the Bitcoin protocol, the block encloses a block header and a set of valid transactions. As shown in Figure 3, the valid transactions recorded in the block are integrated into a Merkle Tree (MT). The MT root is included in the block header, denoted as $\mathcal{BH}$, which also contains a hash *PrevHash* of the previous block header, a nonce, and other essential information. The hash *PrevHash* is stored as the reference to the previous block. Therefore, the blockchain can be regarded as a sequence of blocks connected by a hash chain in chronological order. Secondly, the hash of each block header should satisfy a proof-of work (PoW) puzzle defined by the consensus protocol, that is,

$$SHA256(SHA256(\mathcal{BH})) \leq Target, \tag{1}$$

where parameter $Target \in \mathbb{N}$ defines the difficulty level of the puzzle. It is adjusted per 2016 blocks to maintain the time interval of generation at 10 min on average. The time interval also shows the trade-off between security and throughput[44,45] in Bitcoin blockchain. Miners race to compute the eligible hash for a new block by varying the nonce. Obviously, the probability of mining the new block is proportional to the hashing power the miner owning. In addition, miners always mine on the longest chain that has taken the most hashing power to build.

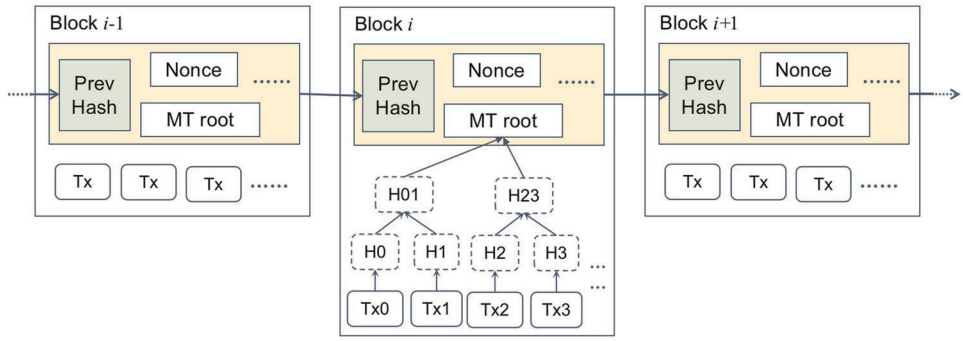

FIGURE 3 The blocks of the Bitcoin blockchain [Color figure can be viewed at wileyonlinelibrary.com]

2.1.4 | Consensus protocol

It is the key technology that ensures all honest nodes of blockchain agree on a unified transaction history or the same state of data without any trusted central authority. It usually specifies the mechanisms of message verifying and decision making for each node. Currently, a variety of consensus protocols are proposed based on different network sources, such as PoW, proof-of-stake (PoS), proof-of-activity (PoA), Byzantine fault tolerant (BFT), and hybrid BFT algorithms. Specifically, in PoW protocols, each node independently solves the PoW puzzle to generate blocks. The valid transactions can be listed on the block and the valid block can be confirmed in the main chain finally. Moreover, the honest nodes always mine on the main chain. The main chain is selected according to the rule of protocol. For example, a longest chain is selected as the main chain in the Bitcoin blockchain protocol. However, this type of PoW protocol and its variants[46,47] always have high computation cost and low throughput. In PoS protocols,[48,49] blocks are generated by the stake holders. Moreover, the chance to propose a block for a PoS miner is proportional to its stake value. The PoA protocol[50] is a hybrid PoW–PoS, where blocks are generated by the stake holders who are related with a pseudo-random sequence such that their chance to be in this sequence is proportional to the volume of stakes owned. In BFT-based protocols,[51-54] every consensus participant can propose the new block including a common set of transactions which are agreed on by a group of participants. This type of consensus protocols can achieve much high transaction throughput, but incur the exploding communication overhead. No matter which kind of the consensus protocols, the goal is to ensure that a majority of honest nodes can agree on a unified view of the blockchain history.

2.2 | Key features

Based on the above background of blockchain, some fundamental features are highlighted below.

- *Anonymity*: Users transact by using pseudonyms that are generated by their secret keys instead of the real identity or physical address. Thus, the real identities of traders are unexposed.
- *Traceability*: The references between transactions provide tracks of all transactions in the blockchain. That is, it is easy to determine where the transaction comes from and flows into.
- *Immutability*: The signatures ensure the correctness and integrity of transactions and the MT structure provides an efficient integrity check. The hash chain prevents blocks from maliciously tampering with.

An example of the traceability shown in Figure 1, the recipient of $\mathsf{Tx3}$ can trace his Bitcoin along with the chain $\mathsf{Tx0} \rightarrow \mathsf{Tx1}(\mathsf{Tx2}) \rightarrow \mathsf{Tx3}$ back to $\mathsf{Tx0}$. Besides, another example of the immutability is shown in Figure 4. Assume that the transaction Tx_i framed in red is modified on the nth block. Then hash values of this branch from Tx_i to the MT root become different from those replicated by other nodes. The header hash of modified block also is different from the one stored in the $(n + 1)$th block. Therefore, the modification is invalid and hard to be approved unless there are nodes possessing more than 51% hashing power to accept the modification and re-generate an alternative chain from the $(n + 1)$th block. In summary, the traditional blockchain ensures the data immutability.

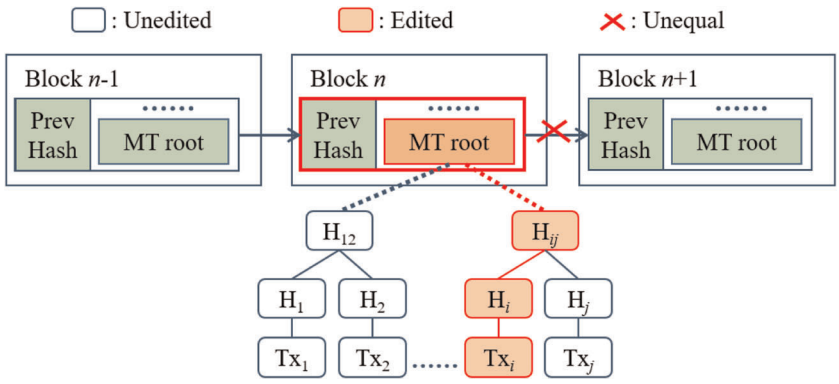

FIGURE 4 An example of blockchain immutability. If one of the transactions is edited, the MT root will be different. As a consequence, the header hash of the block n is not equal to the field of PrevHash recorded in the block $n + 1$. This edit is easily detected and cannot be approved by miners [Color figure can be viewed at wileyonlinelibrary.com]

2.3 | Security properties

A well-developed blockchain system should satisfy three security properties that are defined in Reference [55].

- *Chain growth.* The chain of honest nodes grows at least as fast as the number of blocks they produce, whatever strategy the adversary employs.
- *Chain quality.* The ratio of adversarial block contribution in a sufficiently long and continuous part of an honest node's chain is bounded. Furthermore, in a sufficient quality chain, the amount of hashing power controlled by the adversary controls is bounded.
- *Common prefix.* A blockchain guarantees the common prefix property if the result of cutting off k blocks from the end of an honest node's chain is a prefix of another honest node's chain. It states that the honest nodes should possess a large common prefix, and two subchains have a common prefix before exceeding k blocks.

The security properties ensure the *agreement* and *validity* of blockchain with a high probability. The agreement property requires that honest nodes eventually agree on a single chain, and the validity requires that a majority of blocks originate from the honest nodes.

3 | ANALYSIS OF DATA INSERTION

In this section, we elaborate the mainstream methods of inserting data in blockchain, the type of inserting data, and potential impacts of insertion in detail.

3.1 | Data insertion methods

The data insertion methods in Bitcoin blockchain are explicitly introduced below.

- *Input script.* A coinbase transaction, the first transaction created by miner in a new block, claims the block reward for generating the block. Its sole input provides up to 100 B (bytes) for storing arbitrary data. For example, the genesis block of Bitcoin contains a text in its coinbase transaction. Besides, the field *RedeemScript* of P2SH input script is also a convenient method of storing text. This method can store up to 1.5 kB data.[12] Moreover, the transaction storing data by the above two methods is still spendable.
- *Output script*. Users may encode the semantic data, such as a sentence and an image, as her/his address, called as a fake address. It is indistinguishable from the random binary hash and can receive transactions. Unfortunately, the Bitcoins the fake address receiving are unspendable and stored in UTXO database permanently. Because it is infeasible to find a private key corresponding to the semantic data by solving the elliptic curve discrete logarithm problem (ECDLP). The user cannot sign a new transaction to spend these Bitcoins. For example, the image and tribute text of *Nelson Mandela* are encoded as multiple fake addresses in Bitcoin blockchain.[56] Encoding data as fake address is an efficiently method. The web-based service *CryptoGraffiti* [57] can store up to 50 KiB data via multiple P2PKH output scripts within a single transaction. For big files/images, they can be stored in multiple output scripts in a fragmented manner.
- *OP_RETURN*. A special field OP_RETURN of the transaction can store 80 B arbitrary data.[58,59] At first, it is designed to avoid UTXO database bloat caused by the increasing transactions that are permanently unspendable. But now, it is often used to store the semantic data. Obviously, this method for data insertion has no impact on the transaction, which is still spendable.

No matter which method is used to insert data, the sender must spend some spendable Bitcoin to ensure the transaction can be appended on blockchain. We give some examples of the above-mentioned methods in Table 3. It also illustrates that the method inserting data in the output script is more efficient and popular.

3.2 | Arbitrary data insertion

Generally, arbitrary data can be inserted in the blockchain, including words, characters, images, texts, codes, web links, emails, and so on. This is because there is no general way to

TABLE 3 Data insertion methods, efficiency and examples

Method	Efficiency	Examples
Coinbase	Poor	Genesis block
P2SH input script	High	P2SH Injectors[60]
P2X output script (Fake address)	High	Apertus.io[61]
		CryptoGraffiti[57]
		Satoshi Uploader[56]
		Eternity Wall[62]
OP_RETURN	Poor	Blockstore[63]

Note: P2X: includes P2PK, P2PKH, P2MS, P2SH, P2WPK, P2WPKH and so on.

distinguish legal hash values from arbitrary binary data. Moreover, miners generally do not care about the decoded content of script data, because they can program to verify transactions and blocks.

After decoding, the content of these data may be legitimate, such as the identifiable voting flags,[64] the digital notary document,[65] and the right management data.[66] For example, the assets metadata and notary document are stored in the field OP_RETURN by the service Blockstore.[63] The service Eternity Wall[62] helps someone to write Valentine's day messages into blockchain. On the other hand, the data may be improper and even illicit for malicious purposes, such as the individual privacy data[67] that refers to people's real ID, the healthcare records that contain personal disease and genetic information, the illicit materials that involve child pornographies, the music metadata that is copyright infringement, the political points that may cause troubles, and malicious codes that are harmful and even steal users' Bitcoins. For instance, a malicious code that demonstrates a potential cross-site scripting (XSS) attack is found in the output script of the transaction included in block 251768 of Bitcoin.[56] Matzutt et al.[12] have also found the data related to child pornography and links to darknet services in Bitcoin blockchain.

3.3 | The threats of malicious data insertion

With the growing number of services used to insert data in the blockchain, the vulnerabilities caused by the abused ability of data insertion are becoming apparent. Therefore, the potential threats are discussed from the following aspects.

- *For nodes*. The illegal/harmful data are propagated and locally replicated by participating nodes, which will cause troubles to participants, such as violation of laws. The honest nodes become the assistors of online crime based on blockchain. What's worse, honest nodes may bear the blame for possession of illicit data on their local devices.
- *For victim*. The victim can be a user or even a nonuser of the blockchain. The illegal/harmful data can put victim at risk, such as disseminating victim's sensitive information using blockchain. The information is publicly visible. Moreover, the immutability of blockchain violates victim's right to erase personal data and its copies.
- *For blockchain*. The blockchain system may be disrupted with the increasing illegal/harmful data inserted in the blockchain. As the chain grows, the probability that these data in block are discarded from the blockchain system is diminished. To avoid locally replicating these data, fewer and fewer nodes store the complete blockchain. If there are not enough full nodes in the blockchain, its security and decentralization will be affected.

In summary, the threats of abusing the ability of data insertion cannot be underestimated. The current blockchain ecosystem can not provide perfect accountability for the stored data. It is infeasible to ask all miners to check the content of all data before they append transactions to the blockchain. Moreover, the way that the consensus protocol increases the cost of storing data in the blockchain also cannot prevent data from being intentionally inserted. Therefore, it is eager to find a scalable solution to cope with these data.

4 | THE DESIGN CHALLENGES AND EVALUATION CRITERIA FOR MUTABLE BLOCKCHAINS

To analyze and compare the existing redaction mechanisms in detail, the challenges of redacting data in blockchains are summarized, and some evaluation criteria for redaction mechanisms are proposed in this section.

4.1 | The challenges of redacting data in blockchain

Intuitively, redaction is in conflict with the immutability of blockchain. Therefore, redaction should be carried out under strict constrains. At first, the redaction mechanism should control who can edit data under which circumstances. Then, redaction should not have an impact on system's *security* and *consistency* (As discussed in Section 4.2). The redaction mechanism should also focus on how to verify the redacted transactions/blocks. Specifically, the challenges of redacting data in blockchain can be described as follows.

- It is hard to efficiently redact data in an immutable blockchain that is maintained by independently distributed nodes. The redacted transactions/blocks cannot be approved by nodes.
- Redaction breaks the consistency, which will undermine security guarantees provided by immutability blockchain and cause unexpected vulnerabilities, such as the double-spending attacks and forks.
- Redaction may be carried out at the expense of network resources, such as computation power, bandwidth, and storage. Moreover, it may be incompatible with the popular blockchain protocols, such as Bitcoin and Ethereum.

4.2 | The evaluation criteria for redaction mechanisms

The state-of-the-art mechanisms and frameworks for redacting data in blockchain have been proposed by utilizing various techniques.[68] To comprehensively analyze and compare them, some evaluation criteria are defined in this section.

(1) *Validity*. According to the requirements discussed in Section 2.3, the validity requests that the redacted transaction/block should satisfy the policy defined by the consensus protocol. If the redacted transaction/block is approved in the network, it will be accepted as an honest redaction by nodes.
(2) *Consistency*. It requests that all honest nodes always maintain a consistent view of the chain and the redaction should not damage the validity and connectivity of related transactions/blocks in the past and future. Otherwise, the consistency is damaged as well as the traceability. For example, assume that a honest redaction operation edits the transaction Tx0 as Tx0* in Figure 1. The hash of the redacted transaction Tx0* is different from the one spent by transaction Tx1 and Tx2. Analogously, the hash of new block containing the redacted transaction Tx0* is also different from the one referenced in the next block. As a result, the future transaction Tx1 and Tx2 become invalid, and there are two versions of the same height block concurrently. Thus, although the redaction is honest, it breaks the consistency.

(3) *Security properties*. Redaction mechanisms should not affect the security properties of the blockchain discussed in Section 2.3, including the chain growth, chain quality, and common prefix, which are defined by Garay et al.[55] and Kiayias et al.[45,69] These security properties are fundamental security requirements for a "health" blockchain system (also called as a robust transaction ledger).
(4) *Double-spending attack*. In blockchain system, nodes may earn profit by repeatedly spending the same transaction, that is, double-spending. However, as the definition of the transaction in Section 2.1, a valid transaction cannot be maliciously spent more than one time. If a transaction is referenced multiple times, these related transactions are in conflict. Therefore, the blockchain system is required to detect and resist the double-spending attack.[70] Similarly, the mutable blockchains should can resist the double-spending attack. Specifically, a malicious node may try to spend the funds of a spent transaction again by utilizing the redaction mechanism. For example, assume that the transaction Tx0 in Figure 1 has been redacted as Tx0*. When the new transaction created to spend Tx0* is confirmed, the funds of the transaction Tx0 are spent again. Therefore, a sound redaction mechanism should not increase any chances of double-spending attacks.
(5) *Tamper attack*. The attacker may tamper with data by redacting requests. This attack can affect the trust of the blockchain system. Therefore, a sound mechanism should ensure that honest nodes can detect and disapprove the malicious redaction. Analogously, it will be analyzed whether mechanisms increase the chances of attackers' tampering with data in the blockchain system.
(6) *Denial of service attack*. The malicious nodes may try to flood the network with superfluous redaction requests in an attempt to overload the blockchain system with redacted transactions/blocks confirmation. Then, legitimate requests will be prevented from being executed.
(7) *Consensus delay*. The consensus delay emerges when different honest nodes hold with a different set of redacted transactions/blocks and they cannot fast agree on the final state of the block/chain. It is usually related to the redaction verification and approval processes. The long time delay will result in more forks and vulnerabilities.
(8) *Bandwidth consumption*. Generally, the redacted transaction/block is asked to be broadcast in the P2P network. The bandwidth consumption hence is defined as the additional consumption of broadcasting a revision.
(9) *Additional PoW*. It is defined as the hashing power consumption of redacting transaction/block once. In the redaction mechanisms, the redacted block may be asked to resolve the PoW puzzle again. If the hashing power consumption of the revision is high, miners may prefer to ignore this redaction request.
(10) *Efficiency*. The efficiency is measured by the time taken to accept/approve the revision.
(11) *Compatibility*. It mainly discusses whether mechanisms are compatible with the popular Bitcoin protocol or not. If the mechanism can work with the set of instructions provided by Bitcoin scripts, it is compatible with the Bitcoin protocol.
(12) *Functionality*. It requires that the redaction mechanisms should support dynamic data operations such as modification, deletion, and insertion. Besides, the dynamic nodes should also be considered because the joining and leaving of nodes may have an impact on the execution of mechanisms.

Algorithm 1: CRMs general procedure

Input: Chain $\mathcal{C}$
Output: Redacted chain $\mathcal{C}'$

```
   /* Request                                                              */
1  Generate an edit request message;
   /* Edit                                                                 */
2  Edit non-monetary data of transaction Tx in block B of chain C;
3  Mine a new block B' with the new state containing the edited transaction Tx*;
   /* On-chain voting                                                      */
4  Broadcast and verify the block B';
5  while during voting period do
6      if Pow & Block consistency & Valid chain then
7          Vote for the block B';
8      else
9          Discard block B';
10     end
11 end
   /* Update                                                               */
12 if Votes ≥ preset threshold then
13     Redaction is approved;
14     Replace block B with block B';
15 end
16 Update chain C to chain C'
```

5 | REDACTABLE BLOCKCHAIN MECHANISMS

In this section, we comprehensively review the redaction mechanisms of the state-of-the-art mutable blockchains from four categories, that is, consensus-based, chameleon hash-based, mate-transaction-based, and pruning-based. Then, we analyze these mechanisms based on our criteria. Note that the redaction operations include edit/modification and removal/deletion.

5.1 | Consensus-based redaction mechanisms (CRMs)

CRMs rely on common rules to agree on the final state of the redacted transactions/blocks. Their general procedure is abstracted in Algorithm 1. Redaction is usually performed by on-chain voting on a new state of the redacted block. If nodes are eventually in agreement on the new state, the redaction is approved by honest nodes. We review and analyze three consensus-based redaction mechanisms as follows in detail.

5.1.1 | Hard fork

It is used to modify the history data, correct important security risks in code, add new functionality, roll back transactions, and reverse the effects of hacking. In 2016, Ethereum has used a hard fork to roll back transactions and restored millions of funds that are stolen via "The DAO" attack. As shown in Figure 5, a fork emerges when two or more versions of the same height block concurrently, meaning that alternative chains emerge. Generally, the accidental forks following the same consensus rules can be resolved because the attackers' chain is hard to

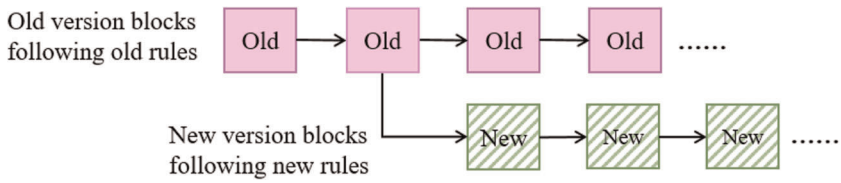

FIGURE 5 An example of hard fork. Hard fork results from a group of nodes following the new rules. It can lead to a permanent chain with the new versions [Color figure can be viewed at wileyonlinelibrary.com]

be longer than the honest one (i.e., 51% attack). However, the hard fork following new consensus rules results in a permanent chain that is run independently of the old version. It is also usually used to split in blockchain cryptocurrency. For instance, Bitcoin Gold and Bitcoin Cash hard forked from Bitcoin in 2017.

Analysis: Hard fork does not exactly erase the transaction history. Both the new and old versions of the blockchain can still be maintained by their own nodes. That is, the old version is still publicly visible. Moreover, the hard fork would take massive computing power and bandwidth to regenerate the chain by resolving PoW puzzles of the subsequent blocks. The cost of editing history data by using the hard fork is much high. Therefore, it is not suitable for frequent rewriting/erasing data in the blockchain system.

5.1.2 | Consensus-based voting (CV) chain

It is proposed to redact data in the permissionless blockchain by Deuber et al.[23] in 2019. To maintain the connectivity between the redacted block and the next block, CV chain extends the structure of block header to accommodate a new field to record an *old state*, that is, the block state (hash) before performing redaction. Moreover, blocks are linked by two hash chains in the CV chain. One is the current state (i.e., the current PrevHash after redacting), the other is the old state. The next block always holds the link to the previous block by referencing the old state. For multiple redactions, the old state consists of all states of all revisions. In addition, CV chain ensures the consistency, that is, honest nodes keep a unified view of the redaction, by a publicly *verifying and then voting* rule. The redacted block/transaction and the redaction request transaction are broadcast in the P2P network. As the general procedure shown in Algorithm 1, miners verify them according the consensus rules. Then, each miner who finds the new next block can vote for a valid revision by recording the redacted block hash in her/his block during the voting period. After the voting period (such as 1024 continuous blocks), the revision is eventually approved if the edited block gets enough votes. Finally, honest node updates their replicas by replacing the block with the edited block.

Analysis: In the CV chain, the ingenious structure of two hash chains and the voting-based mechanism ensure the connectivity and validity of redacted blocks so the consistency is guaranteed. Moreover, the on-chain voting mechanism provides the public verification to ensure accountability and antidouble-spending attack, and so on. It can also resist the DoS attack because of the cost of the transaction fee of the redaction request. Besides, the redacting privilege is decentralized to a group of nodes who find the next block. However, the voting period should be enough long to ensure the honest redaction can be approved eventually, which will cause nonscalable and poor efficiency. For example, it is asked there are more than 50% of 1024 consecutive blocks voting for the redacted block. That is, an honest redaction will take 7 days on average to be approved. Multiple/frequent redacting

blocks will result in longer time consensus delays and even more forks. Finally, the CV chain is obviously incompatible with the typical Bitcoin protocol due to the change of block structure.

5.1.3 | Reparo

Reparo, proposed by Thyagarajan et al. in 2020,[24] performs redactions in a repair layer on top of any blockchain system (permissioned and permissionless). Redaction is requested by a repair message that indicates the block hash, the object, and the chain state. Then the repair message is off-chain deliberated by verifying the state changes in the repair layer. The final deliberation results as a repair witness will be voted on-chain by miners. The voting method is the same as that of the CV chain, that is, the witness getting enough votes during the voting period will be approved. To ensure the block/transaction consistency, Reparo stores the old version and the approved repair message in their databases, respectively. These databases are also broadcast and maintained by nodes in the P2P network. In addition, the repair message also spends coins to be recorded in the blockchain.

Analysis: Reparo is instantiated in typical Ethereum, Bitcoin, and Cardano so it has better compatibility. It also maintains consistency by recording the old state stored in associated databases instead of the block header. Although Reparo does not change the typical data structure of the blockchain, it will consume more additional bandwidth, storage, and computation to maintain databases. Besides, Reparo adopts the same processes of the CV chain such as on-chain voting and public verification. It can also ensure system security and has the similar performance defects, such as the long voting period, the consensus delay, and poor efficiency.

Besides, Marsalek et al.[25] have designed a correctable blockchain architecture which is also composed of two hash chains, that is, the standard chain and the correction chain. The standard chain stores the original information and the correction chain consisting of correction blocks is used to store the correction information. Analogously, the consensus-enforced voting mechanism is also used to make a decentralized decision on requested data corrections. After the end of the voting period, a correction block containing the successful correction information is used to replace the redacted block with the same height in the standard chain. Obviously, this mechanism of Marsalek et al. also suffers from these issues such as long voting period and poor efficiency.

In summary, consensus-based redaction mechanisms do not rely on any trust assumption and are applied to the permissionless blockchain. They ensure the validity, consistency, and security of the system by the voting-based consensus. Moreover, they can resist the DoS attack because each redaction request spends coins in the form of a transaction fee. However, the redaction usually carries out at the expense of additional bandwidth and computation (or other resource). Nodes have to resolve the PoW puzzle (assuming the data of Bitcoin blockchain is redacted) of the redacted block and broadcast the request transaction and the redacted block/transaction. A long time of voting period can ensure the security of the system, but may result in consensus delay. Especially, the time of consensus delay will become much longer when multiple redactions are concurrently requested. Therefore, this type of redaction mechanism usually has poor efficiency and scalability.

5.2 | Chameleon hash-based redaction mechanisms

This type of mechanism redacts blockchain data without hard forking based on a trapdoor collision-resistant hash function, that is, the chameleon hash (CH) function[71] defined as Table 4. The CH function ensures that (i) Anyone possessing the public key can compute the

TABLE 4 Algorithms of CH function

$\mathsf{CHKey}(1^{\kappa})$: The probabilistic key pair generation algorithm CHKey takes as input a security parameter $\kappa \in \mathbb{N}$ and outputs the public and trapdoor key (pk, sk).
$\mathsf{CHash}(pk, m)$: The probabilistic hashing algorithm CHash takes as input the public key pk and a message $m \in \mathcal{M}$, and outputs a randomness r and hash h.
$\mathsf{CHVer}(pk, m, (h, r))$: The deterministic verification algorithm CHVer takes as input the public key pk, a message $m \in \mathcal{M}$, hash h and randomness r, and outputs a bit $b \in \{0, 1\}$ indicating if the hash h is valid.
$\mathsf{CHCol}(sk, (m, h, r), m')$: The probabilistic collision algorithm CHCol takes as input the trapdoor key sk, message m, hash h, randomness r and an additional message m', and outputs randomness r' such that $\mathsf{CHVer}(pk, m', (h, r')) = 1$.

CH value; (ii) It is infeasible to find two different inputs mapped to the same output without the trapdoor key except with negligible probability; (iii) The parties possessing the trapdoor key can easily find collisions for every input.

The general procedure of CH-based redaction mechanisms (CH-RMs) is outlined in Algorithm 2. Based on the CH function and its variants, CH-RMs ensure that the redacted transaction/block hash remains intact. First, the transactions/blocks are generated based on the CH function. Then, nodes owning the trapdoor key can easily get an additional randomness r' by computing collisions for any redacted data. Apparently, CH-RMs do not require to solve the PoW puzzle again, but the key management is critical to system security. Hence, this type of mechanism usually focuses on controlling redaction based on variously malleable cryptographic primitives. Here, several typical redaction mechanisms are introduced in detail.

Algorithm 2: CH-RMs general procedure

Input: Chain C
Output: Redacted chain C'

/* Key generation */
Join the blockchain network with a key pair;
/* Transaction/block generation */
Generate transactions/blocks based on the CH function (or its variants) in the chain C;
/* Request */
Request to redact non-monetary data in a transaction/block of the chain C;
/* Redacting */
Redact the transaction/block;
Run the algorithm CHCol with the trapdoor key;
CHCol returns an additional randomness r' for the redacted transaction/block;
Broadcast the redacted transaction/block with r';
/* Update */
if *Valid r' & Validly redacted transactions/block* **then**
    Update the chain C to C' that contains the redacted transaction/block;
end

5.2.1 | Standard chameleon hash-based (SCH) chain

It is proposed based on the SCH function by Ateniese et al.[26] in 2017. SCH chain adds a lock to each hash chain between blocks, which makes the immutable blockchain become a redactable one with the trapdoor key. In the block generation party of Algorithm 2, the inner SHA-256 function of the double hash of Equation (1) is replaced with the SCH function, that is,

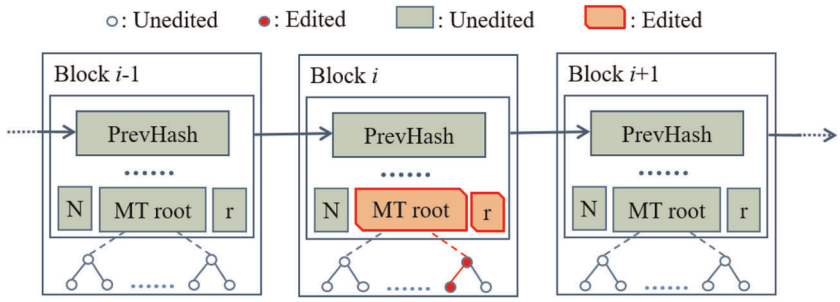

FIGURE 6 The block-level redaction structure of Standard chameleon hash based chain. After a transaction is edited in block i, the MT root will change. Miner finds an additional r by running the algorithm CHCol for block i so that the header hash of edited block i is still equal to the hash PrevHash stored in block $i + 1$. MT, merkle tree [Color figure can be viewed at wileyonlinelibrary.com]

$$SHA256(\mathsf{CHash}(pk, \mathcal{BH}; r)) \leq Target, \tag{2}$$

where pk is the public key and r is the randomness defined in Table 4. Moreover, the block structure of the SCH chain is also extended to add a field to record the randomness as shown in Figure 6. The holders of the trapdoor key can efficiently compute the additional randomness r' for the redacted block i by running the collision algorithm CHCol. The randomness r' ensures that the redacted block header hash keeps the same as the one before redacting. Therefore, the hash chain remains intact. In addition, SCH chain supports modifying/deleting data on more blocks, denoted as redaction in block-level.

Analysis: In the SCH chain, the block hash after any redaction operations is unchanged so the block consistency is maintained. However, redaction damages the validity of transactions listed in the block and even other related transactions. The transaction consistency and security properties will be breached when blocks are removed from the blockchain. For example, the chain quality will become bad because of the removal of many honest blocks. What is worse, the adversary can easily launch double-spending and tampering attacks because the traceable transaction history is erased. The attackers also can launch the DoS attack by flooding the network with many edited blocks to be verified in this system. Besides, this mechanism suffers from the key management problem. Once the trapdoor key is exposed, the ability to redact data may be abused. Therefore, SCH chain may need a central authority and rely on a secure multiparty computation (MPC) scheme to ensure system security. On the other hand, the SCH function suffers from a key exposure problem, that is, any user can find other collision or even recover the trapdoor key once the collision is public. At present, the construction of a key exposure-free chameleon hash function for mutable blockchain is an ongoing research.

5.2.2 | Attribute-controlled policy-based chameleon hash (APCH) chain

It is designed to control who is able to redact data in a fine-grained way by Derler et al.[27] in 2019. Based on an enhanced CH function[72] and a ciphertext-policy attribute-based encryption (CP-ABE) scheme,[73] the authors have proposed a policy-based chameleon hash (PCH) function associated with the attributes of participants to control the redaction privilege. In the APCH chain, transactions are specified as modifiable and nonmodifiable. The PCH hash values of

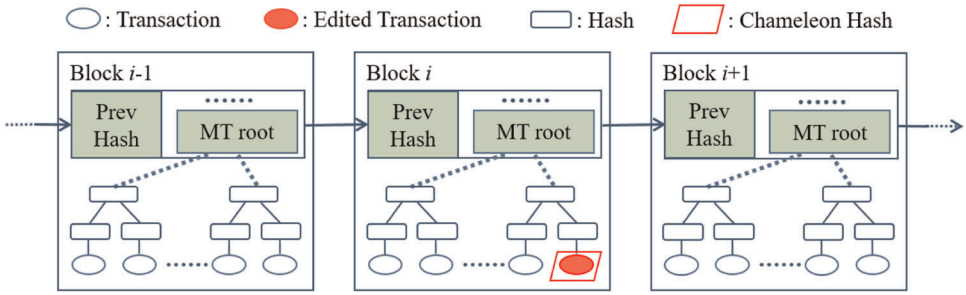

FIGURE 7 The Tx-level redaction structure based on CH function and its variants. Assume that a transaction of block i is edited. Then miner finds an additional r by running the algorithm CHCol for the transaction so that the edited transaction hash is unchanged [Color figure can be viewed at wileyonlinelibrary.com]

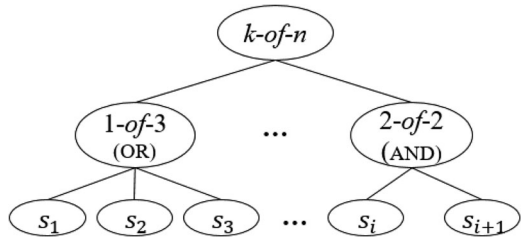

FIGURE 8 The structure of access tree. It is structured by a set of attributes. An **OR** gate is composed of attributes S_1, S_2, and S_3. An **AND** gate is composed of attributes S_i and S_{i+1}

modifiable transactions are taken as inputs to integrate into the Merkle tree with nonmodifiable transactions, as shown in Figure 7. The redaction is performed in the transaction-level (Tx-level). As shown in Figure 8, the set of attributes structures an access tree, where each leaf node describes an attribute and each non-leaf node represents a threshold gate.

For the k-of-n ($1 \leq k < n$) threshold gate composed of n attributions, it can be constructed as an **AND** gate if the threshold value $k = n$ and it can be an **OR** gate if $k = 1$. The participant can get the right trapdoor key if her/his attributes can satisfy the strategy specified by the access structure. Finally, the participant owning trapdoor key can easily compute the randomness r' for the redacted transaction by computing the collision (shown in Table 4).

Analysis: The PCH function elegantly provides a fine-grained redaction control based on the trapdoor key access-control. It is suitable for the permissioned blockchain where participants are asked to join the system with authorized attributes. Moreover, the CP-ABE is IND-CCA2 secure and the PCH function is strongly indistinguishable and collision-resistant, which ensures system security, that is, attackers are hard to recover the trapdoor key to launch the double-spending attack and the tampering attack. The APCH chain supports redaction in Tx-level, which will not affect the security properties of the system. Nevertheless, the signature becomes invalid after redaction. The validity and the consistency of those redacted transactions that cannot be re-signed because the sender is offline will be damaged. On the other hand, the PCH function may face the potential risk of collusion attacks. Participants may collude with each other to get a union of attributes that may satisfy the strategy. Besides, the update and sharing of the access tree are not considered in the APCH chain. In terms of compatibility, the PCH function can be served as a layer on top of blockchains so it is compatible with typical blockchains.

5.2.3 | Redactable consortium blockchain (RCB) for IIoT

It is proposed to ensure the validity of redacted transaction by Huang et al.[28] in 2019. RCB focuses on resigning the redacted transaction when the sender is off-line in a consortium blockchain. It enhances the SCH function by being associated with the identities of the participants (denoted as TCH function). Then, the participant with the correct identity can effectively compute the additional randomness with the help of all holders who have trapdoor keys. Following the hash-and-sign paradigm, transactions are hashed by the TCH function and then signed by an accountable-and-sanitizable chameleon signature (ASCS) algorithm, as shown in Figure 9. The participant with the correct identity can compute the same signature to validate the redacted transaction. Therefore, the signature chain between transactions can be maintained. Analogously, RCB also uses the TCH function to hash the modified transactions before they are integrated into the Merkle tree.

Analysis: RCB is managed by a group of equally-powerful participants who join the system with the authorized identities. This group of trusted participants is in charge of verifying the edited transaction/block and then reaching consensus on the final state of the edited chain. Thus, the redaction ability cannot be abused maliciously. Moreover, the ASCS algorithm ensures the validity of the edited transaction so the consistency is guaranteed. However, there are some limitations in RCB. First, it is obvious that the redaction ability is in centralized control by minimal trusted parties. Second, the process of computing the collision needs the co-operation of all trapdoor keys holders, which will increase bandwidth cost and reduce redaction efficiency and flexibility. Third, RCB still suffers from key management problem which affects system security. Finally, the ASCS algorithm cannot be compatible with the typical blockchain, and the computing-limited and storage-limited IIoT devices cannot efficiently perform complex cryptographic primitives.

5.2.4 | Self-redactable blockchain (SRB) for IoT

SRB is designed to redact data without any co-operation in the blockchain system by Huang et al.[29] in 2020. It also enhances CH function by associating with identities and a cycle time, denoted as RCH function. Based on the RCH function, it designs a revocable chameleon signature (RCS) scheme. Analogously, transactions are generated based on the RCH function and the RCS scheme. SRB allows periodically redacting data in the blockchain by the participant who owns the correct identity and trapdoor key during the cycle time. Any redaction consists of one private redaction with a trapdoor key and $(n-1)$ public redactions if there are n transactions appended after the redacted transaction. The public redaction that the signature is generated to authenticate transactions can be freely conducted by anyone.

Analysis: SRB is also built to maintain the signature of the edited transaction when the sender of the modified transaction is off-line. The security and performance of SRB are similar to those of RCB via analogical reasoning. Besides, SRB is also not adopted for IoT devices because it introduces the complex cryptographic primitives.

Besides, some CH-based work have been designed for other applications. Zhang et al[30] have designed an industrial data management scheme based on a dual-blockchain architecture which is also constructed by the chameleon hash function. This scheme is analogous to the SCH chain. The trapdoor key used to compute the collision for the modified data is also shared based on the verifiable secret sharing technology. Zhang et al.[31] have also proposed a rechain scheme to rewrite blockchain history which is maintained by the edge nodes in IoT. This scheme based on threshold trapdoor chameleon hash function can efficiently save space of edge devices and overcome the general issues

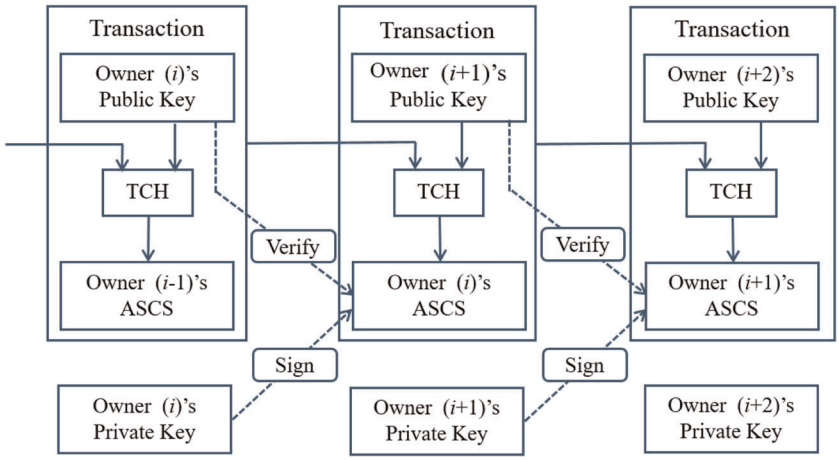

FIGURE 9 The signature chain of RCB. The SHA256 and elliptic curve digital signature algorithm (ECDSA-Secp256k1) of Bitcoin protocol are replaced with TCH and ASCS in RCB, respectively. The transaction still can structure a chain via the signature verification. ASCS, accountable-and-sanitizable chameleon signature [Color figure can be viewed at wileyonlinelibrary.com]

in IoT such as resource-constraint. At the same time, Zhang et al. have also constrainted the rewrite operations by the chronological order of historical transactions. Samelin et al.[34] have proposed a policy-based sanitizable signature scheme that can ensure the validity of the redacted transaction. Based on the access structures defined over attributes, the redaction operations are controlled and accountable in the blockchain system. Wu et al.[32] have proposed a key-exposure free chameleon hash functions based on the lattice, which is the first work to construct the quantum-resistant chameleon hash function. Moreover, they have focused on solving the redaction-misuse problem which may happen in the generic framework of redactable blockchains.

In summary, CH-based redaction mechanisms (CH-RMs) aim to implement data redaction in specific and exceptional circumstances. The main techniques which are used by the above-mentioned schemes are summarized in Table 5. These schemes maintain the consistency by computing a hash collision for the edited block/transaction and ensure the security based on the common cryptographic hardness assumptions, such as DLP and CDHP. In detail, the collision is computed to keep the redacted block/transaction hash or the signature of the redacted transaction unchanged. Moreover, CH-RMs do not require to resolve the PoW puzzle again for the new state of the redacted block. Redaction can be efficiently made on one or more blocks. The redaction privilege is in centralized control by a group of authorized participants owning trapdoor keys. Hence, the risk of the redaction ability abused is reduced. However, the key management and key exposure are the problems to be solved in CH-RMs. Some schemes overcome these problems by increasing the complexity of the function, which will cause high consumption of network resources.[33]

5.3 | Meta-transaction-based redaction mechanisms

This type of redaction mechanism is controlled by fiat and *mate*-transaction (*mate*-Tx) that is a special type transaction issued to trigger redaction. As shown in Algorithm 3, *mate*-Tx-based redaction mechanisms (MTxRMs) avoid introducing heavy cryptographic primitives and the long voting period. Three MTxRMs are reviewed and analyzed below in detail.

TABLE 5 Summary of chameleon-hash-based redaction mechanisms

Schemes	Identity-based	Attribute-based	Secret sharing-based	Free key-exposure	Assumptions
[26]	×	×	✓	✓	DLP
[27]	×	✓	×	✓	DLP
[29]	✓	×	×	✓	CDHP
[28]	✓	×	×	✓	CDHP
[30]	×	×	✓	✓	DLP
[33]	×	×	✓	✓	DLP
[31]	✓	×	✓	✓	CDHP
[32]	×	×	✓	✓	SIS
[34]	×	✓	✓	✓	SXDH

Abbreviations: CDHP, computational Diffie-Hellman problem; DLP, discrete logarithm problem; SIS, small integer solution problem; SXDH, standard symmetric external Diffie-Hellman assumption.

Algorithm 3: MTxRMs general procedure

Input: Chain $\mathcal{C}$

Output: Redacted chain $\mathcal{C}$

/* Request */

1. Generate a mate-transaction to trigger redaction;

/* Redacting */

2. Verify the mate-transaction according to the predefined policy;
3. Perform the redaction;

/* Update */

4. Update the view of the chain $\mathcal{C}$ to chain $\mathcal{C}$;

5.3.1 | μchain

It is proposed to withdraw coins from spent transactions by Puddu et al.[35] in 2017. *μchain* makes blockchain data redactable by issuing new *meta*-Txs, that is, a mutant transaction and a extending transaction. At first, μchain divides all transactions, including currency transactions and smart contract deployment transactions, into immutable and mutable transactions. There is a mutable transaction specified as active in each round and redacted subject to a policy established by the sender. The policy specifies the redacted objects, the redactor, and the time window. Then, the redactor generates the mutant and extending transactions to trigger and implement redaction. After the extending transaction is approved via an efficient off-chain voting procedure, the mutant transaction replaces the active transaction. Besides, μchain encrypts the mutable transaction to hide alternative transaction history. The secret key is shared by a dynamic proactive secret sharing (DPSS) scheme,[74] which satisfies the perfect security with a threshold of $t < \frac{n}{3} - \epsilon$ corrupted parties (for n parties and any constant $\epsilon > 0$).

Analysis: μchain allows that the redacted transaction has an impact on the past and future transactions so it breaches the consistency. Moreover, the block inserted the mutant and

extending transactions cannot be confirmed in the network. In terms of security, μchain satisfies the security properties and resists the double-spending, tampering, and DoS attacks. In terms of performance, it has to consume more computation and bandwidth for transaction encryption and keys management. Concretely, encrypting transactions undermines the transparency of the blockchain and limits the verification enforcement of currency transactions, so it also reduces the auditability of the data written in the blockchain. In the μchain, a malicious user can establish a policy to prevent the transaction from redaction. Furthermore, the replacement of μchain actually is performed by changing the view of transactions. That is, participants can read the old version with the decryption key or change the view back to the old version.

5.3.2 | MOF-BC

It is proposed to provide flexible memory management for IoT devices by Dorri et al. in 2019.[16] MOF-BC supports to removal or summarize specific transactions of blockchain over a period in large scale networks by issuing a remove transaction. The removal transaction records the hash of the deleted transaction and the hash of the previous transaction to ensure the transaction consistency. Then it is signed by the redactor to verify her/his eligible redacting privilege. After that, miners authorized by a Certificate Authority (CA) verify the removal transactions over a period. If the removal transaction appended to the blockchain, the removal operation will be executed at the IoT devices. Moreover, MOF-BC rewards users to encourage them to remove their transactions. Analogously, a summary transaction is created to summarize users' transactions, an example of summarizing transactions of Figure 1 shown in Figure 10. MOF-BC summarizes transactions tracing the signature chains of transactions. To ensure the consistency and validity, it stores some important information about the summarized transactions, such as the MT root, the time-stamps, and the order.

Analysis: The MOF-BC focuses on removing history data. The removal/summary transaction stores the information about removed/summaried transactions to ensur the consistency and validity. However, the MOF-BC depending on the verification of CA is vulnerable and also suffers from the key management problem. It also cannot resist the double-spending and DoS attacks. The malicious user can spend the funds of the spent transaction that has been removed/summarized. Besides, removal/summary transactions are associated with the same

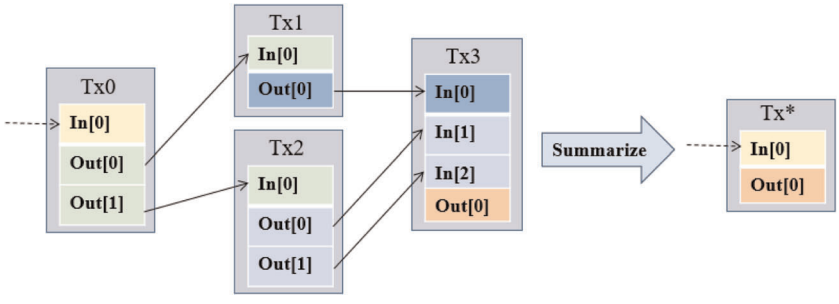

FIGURE 10 An example of summarizing transactions. The transactions Tx0, Tx1, Tx2, and Tx3 can be summarized as the transaction Tx*. The input and output of Tx* are the input of Tx0 and the output of Tx3, respectively [Color figure can be viewed at wileyonlinelibrary.com]

private key. If the private key is exposure, anyone can request to remove/summarize any transactions.

5.3.3 | Functionality-preserving local erasure (FPLE)

It is proposed by Florian et al.[36] in 2019. FPLE focuses on the erasure of the improper data stored in transaction outputs from the local storage of nodes. According to the analysis of Section 3.1, the transaction outputs containing arbitrary data are very unlikely to be spent in the foreseeable future and overwritten in the local UTXO database. To safely erase these data, FPLE modifies the field *ScriptPubKey* of the transaction output and designs an erasure database to store the information before modification. The erasure database can help to check that the relevant inputs have not been edited. It can also verify the subsequent transactions that reference (spend) the edited transaction. After check and verification, the original transaction will be replaced with the edited transaction and physically erased.

Analysis: Instead of changing the protocol to carry out blockchain redaction, FPLE focuses on erasing data locally and checking the sequence transactions after the transactions have been redacted. It can ensure the security properties and consistency of the system. However, the erasure database source must be trusted and ensure that it stores the original data of the edited transaction. That is, to ensure the credibility of the erasure database source, the erasure database may be maintained by a trusted third party or a minority of trusted nodes. The decentralization and security of the Bitcoin system may be affected. Besides, it relies on the strong assumption that nodes honestly perform the local erasure.

In summary, the *mate*-Tx-based redaction mechanisms also suffer from key management and trusted source of the database. They still rely on the help of trusted parties. For example, once attackers get the keys, they can efficiently redact transactions for malicious purposes. Besides, editing the local data synchronously when data is edited on the chain is a useful research topic.

5.4 | Pruning-based redaction mechanisms (PRMs)

In addition to the above emerging redaction mechanisms, the PRMs are designed for specific application scenarios. The general procedure of PRMs is outlined in Algorithm 4. It also avoids heavy cryptographic primitives. In this section, two mechanisms are presented.

Algorithm 4: PRMs general procedure

Input: Chain $\mathcal{C}$

Output: Redacted chain $\mathcal{C}'$

/* Request */

1. Request to prune transactions/blocks if a condition is satisfied;

/* Redacting */

2. Prune transactions/blocks according to the predefined policy;

/* Update */

3. Update the chain $\mathcal{C}$ to chain $\mathcal{C}'$;

5.4.1 | LiTiChain

It aims at saving the storage of edge nodes in the IoT ecosystem and is proposed by Pyoung et al. in 2019.[17] In the IoT ecosystem, edge nodes have limited storage and computing power so that they are likely to run out of memory and can not run the full blockchain. Therefore, the LiTiChain overcomes these problems by deleting expired blocks. In the LiTichain, edge nodes reach a distributed consensus based on the practical byzantine fault tolerance algorithm,[51] where the corruption tolerance threshold t satisfies $n \geq 3t + 1$ for n nodes. Then, each transaction/block of the LiTichain is maintained during a lifetime, that is, the time from the *creation* to the preset *endtime*. If the lifetime of a transaction/block expires, it can be deleted from the chain. For this special case that the expired block contains unexpired transactions, LiTichain renews these unexpired transactions by appending them to a new block. After that, LiTichain structures a graph according to the endtime ordering of blocks, where the child block always has earlier endtime than that of its parent block. Each block header contains two references hash to ensure the block consistency. One is the hash *PrevHash* of the previous block and the other is the hash *ParentBlockHash* of the parent block in the endtime ordering graph. When some blocks are deleted, the rest blocks can still connect with each other.

Analysis: The deletion of the outdated blocks and transactions reduces the percentage of honest blocks in the chain. Obviously, the chain quality and common prefix are also undermined. Besides, the LiTiChain cannot maintain transaction consistency. It cannot ensure the deletion of transactions does not affect the past and future transactions. The deletion without consistency will result in security vulnerabilities. For example, malicious nodes easily launch the double-spending and forge transactions that are legitimately deleted from the chain. It also cannot resist the DoS attack. What is worse, LiTiChain does not record the hash of the deleted blocks/transactions, which causes that the traceability and integrity of the chain cannot be guaranteed.

5.4.2 | CoinPrune

It is proposed based on a snapshot technique by Matzutt et al. in 2020.[37] CoinPrune also aims at reducing blockchain size to optimize the storage and bandwidth requirements. Specifically, CoinPrune creates a snapshot for constant intervals (e.g., every 10,000 blocks). The snapshot consists of the block headers and serialized UTXOs set of pruned blocks. Then, it is publicly announced to be reaffirmed during a time window. If the reaffirmations are more than the preset threshold, the snapshot is valid and accepted. Otherwise, it is invalid and pruning is delayed. The new joining nodes only need to require the reaffirmed snapshot and a few full blocks to synchronize with the system.

Analysis: CoinPrune can reduce disk space and synchronization time, but frequent pruning causes an expensive overhead. Although it does not actually delete the data from the blockchain, it provides a solution for some nodes to avoid storing improper data in local devices. However, the nodes running the blockchain based on the snapshot are similar to the lightweight nodes. CoinPrune also needs full nodes to store full blockchain. The consistency and security are guaranteed by these full nodes. Moreover, this way will result in the centralization if the full nodes account for a very low percentage of the network. Even so, pruning is a feasible solution and an active field of research.

In summary, PRMs implicitly assume that the pruned data are independent of the remaining or future transactions. Otherwise, the validity of the remaining or future transactions is hard to be

guaranteed. Besides, PRMs perform redaction for flexible memory management at the expense of system security, so they are suitable for applications with weak security requirements.

5.5 | Other emerging work

In addition to the above-mentioned four categories of redactable blockchains, some emerging work that may be not classified is also reviewed concisely in this section.

Grigoriev et al.[75] have designed a redactable private blockchain based on the RSA assumption. The idea, where the one-way function values of the original data and the redacted data are the same, is the same as that of the type of chameleon hash-based redaction blockchain. This mechanism has achieved the fundamental security requirements including chain growth, chain quality, and common prefix. However, it also suffers from the issue that it is lacking in a rule to select a central authority from nodes. Bilzhause et al.[76] have considered the signature verification of the redacted transaction based on the sanitizable signature scheme in redactable blockchains. They allow any semitrusted entity to modify the signed data in a controlled way and preserve the authenticity of the subsequently modified content. Chen et al.[77] have introduced a method to compress the size of transactions in the Bitcoin blockchain by replacing hash pointers with index pointers. Unfortunately, this method may destroy the authenticity and traceability of transactions appended on the blockchain. Arthur et al.[78] have discussed the pros and cons of a fraction of recent work on redactable blockchains. Then they have given some recommendations for organizations contemplating enterprise-level applications of this technology. However, they have not discussed and analyzed the performance of redactable blockchains at technique level.

6 | COMPARISONS AND DISCUSSIONS

In this section, we present a comprehensive comparison for some aforementioned mechanisms in terms of framework, security, and performance. After that, we discuss future research directions of the mutable blockchain.

6.1 | Comparisons

Based on the evaluation criteria defined in Section 4.2 and the detailed analyses of Section 5, we compare some typical mutable blockchains in terms of the security and performance. The comparisons are shown in Table 6.

From the comparisons, we find that there are a few mechanisms that are suitable for permissionless blockchains and support redaction at the block-level. Most redaction mechanisms are designed with the security assumption such as cryptographic hardness assumptions. Besides, most redaction mechanisms can ensure the validity of the edited transaction/block except for the SCH chain,[26] the APCH chain,[27] and the LiTiChain.[17]

In terms of security, most mechanisms can satisfy the security properties and have the capability of antiattacks except for those mechanisms supporting the removal operation (e.g., SCH chain,[26] MOF-BC,[16] and LiTiChain[17]). Removing blocks/transactions from the blockchain system will cause vulnerabilities in terms of agreement and validity. Moreover, the edited block/transaction is broadcast and publicly verified in the network to reduce the risk of

TABLE 6 Summary about redaction mechanism comparative analysis

Proposals	Consensus-based		CH-based				Meta-Tx-based		Pruning-based	
	23	24	26	27	28	29	35	16	17	37
Key mechanism	Voting	Voting	CH	Att-CH	ID-CH	ID-CH	Meta-Tx	Meta-Tx	LifeTime	Snapshot
Network	P-less	Both	Both	P-ed	P-ed	P-ed	Both	P-ed	P-ed	P-less
Redaction	Tx-level	Tx-level	B-level	Tx-level	Tx-level	Tx-level	Tx-level	Tx-level	B-level	B-level
Security assumption	×	×	DLP	DLP	CDHP	CDHP	Corruption tolerance $t < \frac{n}{3} - \epsilon$	CA	Corruption tolerance $n \geq 3t + 1$	×
Validity	✓	✓	×	×	✓	✓	✓	✓	×	✓
Block consistency	✓	✓	✓	✓	✓	✓	×	×	×	✓
Transaction consistency	✓	✓	×	×	✓	✓	×	✓	×	×
Chain growth	✓	✓	×	✓	✓	✓	✓	×	×	✓
Chain quality	✓	✓	×	✓	✓	✓	✓	×	×	✓
Common prefix	✓	✓	×	✓	✓	✓	✓	×	×	✓
Antidouble spending	✓	✓	×	×	✓	✓	✓	×	×	✓
Anti-Tampe-ring attack	✓	✓	Key-based	×	Key-based	Key-based	Key-based	Key-based	✓	✓
Anti-DoS	Tx-fee	Tx-fee	×	✓	✓	✓	Tx-fee	×	×	×
Bandwidth consumption	Tx&B	Tx&B	B	Tx	Tx	Tx	Txs	Tx	×	B
Additional PoW	✓	✓	×	×	×	×	✓	×	×	✓

(Continues)

TABLE 6 (Continued)

Proposals	Consensus-based		CH-based				Meta-Tx-based		Pruning-based	
	23	24	26	27	28	29	35	16	17	37
Consensus delay	V-p	V-p	×	×	×	×	V-p	×	×	×
Efficiency	Poor	Poor	High	High	High	High	Medium	Medium	Medium	poor
Compatibility	×	✓	×	✓	×	×	×	×	×	✓

Abbreviations: *CH: Chameleon hash function. Att-CH: Attribute-based CH function. ID-CH: Identity-based CH function. Meta-Tx: Meta-transaction. P-less: Permissionless. P-ed: Permissioned. Both: Permissionless and permissioned. Tx(s): Transaction(s). B: Block. Tx&B: Transaction and Block. DLP: Discrete logarithm Problem. CDHP: Computational Diffie-Hellman Problem. CA: Certificate Authority. Key-based: Anti-attack ability relies on the security of key management. Tx-fee: The mechanism can resist the DoS attack because of the cost of a transaction fee for each redaction request. V-p: The consensus delay may emerge during the voting period.

double-spending in most mechanisms. However, the key management is a problem to be solved in most mechanisms (e.g., RCB[28] and μchain[35]). Their capability of antitampering attack depends on the security of keys. Besides, some mechanisms resist the DoS attack by the cost of a transaction fee for every redaction request (e.g., CV chain[23] and Reparo[24]).

In terms of performance, CH-based mechanisms can implement data redaction with high efficiency and do not suffer from consensus delay. They compute the hash collision for the edited transaction/block in milliseconds and broadcast the edited it in the network in seconds (at least 12 s[79]). However, the mechanisms based on voting consensus redact data in the blockchain at the expense of high computation and bandwidth consumption. Moreover, they are poor efficient because they require enough time to vote on the edited block. For example, the voting period of the CV chain[23] is preset as the 1024 consecutive blocks, that is, about 7 days on average (Assume that there is one block generated per 10 min on average). During the long voting period, miners cannot fast agree on the state of the chain so that consensus delay emerges. Besides, in the *mate*-Tx-based mechanisms, the redaction is accomplished until the transaction generated to redact data is appended to the blockchain. This process may take at least 10 min so the redaction efficiency of *mate*-Tx-based mechanisms is medium. On the other hand, to ensure system security, more network resource consumption is ineluctable. From Table 5, most mechanisms have to broadcast the edited transaction/or block for verification, and some mechanisms also need to resolve the PoW puzzle.

In terms of compatibility, most redaction mechanisms are almost incompatible with the Bitcoin protocol. This is because some mechanisms need to change the data structure of the block/transaction, such as the CV chain[23] and the SCH chain.[26] Some mechanisms need to change the cryptographic primitives of the Bitcoin protocol, such as RCB[28] and SRB.[29]

Besides, we compare the state-of-the-art redaction mechanisms in terms of functionality. The results are summarized in Table 7. From the summary, it is obvious that this type of consensus-based redactable blockchains is Comprehensive. It can support dynamic data operations and dynamic nodes. Especially, the redaction can be performed normally in the consensus-based mechanism, even though a fraction of nodes leave the P2P network. However, if the departure of nodes occur in the CH-based and Meta-Tx-based mechanisms, the redaction cannot be ensured and even the system will stop working.

6.2 | Future research directions

At present, the research progress on designing mutable blockchain is still in its infancy. Designing the redaction mechanism for various applications is still interesting research directions. Future research on the redaction mechanism can focus on the below aspects.

TABLE 7 Summary about the functionality of redaction mechanisms

	Dynamic data operations			Dynamic nodes	
Proposals	**Insertion**	**Modification**	**Deletion**	**Joining**	**Leaving**
Consensus-based	✓	✓	✓	✓	✓
CH-based	✓	✓	✓	✓	×
Meta-Tx-based	×	✓	✓	✓	×
Pruning-based	×	×	✓	✓	✓

Abbreviations: ✓, support; ×, nonsupport.

(1) *Consistency*. How to preserve the consistency of the redacted chain is an important topic for designing redaction mechanisms. The existing works ensure it by storing the state (hash) of the transaction/block before redacting. However, this method still has some vulnerabilities with the removal operation. It is hard for the recipient of the removed transaction to be verified whether she/he can spend this transaction. Moreover, the consistency of the transaction chain may be violated. Transactions referring to the removed transactions may be invalid.
(2) *Security*. The other security requirement of mutable blockchains is preventing the redaction ability from maliciously abusing. Legitimate redaction may be requested for malicious purposes. Many mechanisms consider the redacted chain (i.e., the chain includes the redacted block/transaction) as the preferred one to mine the next block. Consider a scenario where attackers want to double-spending by forks. If attackers draw the contributions of honest nodes by legitimate redaction requests, the fork chain is possible to be longer than the honest chain.
(3) *Consensus delay*. The consensus delay directly affects the agreement of the blockchain system. The longer time delay will cause more forks and vulnerabilities. In most existing works, it still is a problem to be solved. For example, the attackers can launch a DoS attack by frequently requesting different and legal edits for the same block. Many legitimately redacted blocks that are approved by the policy of mechanism are held by many miners. Therefore, delays are increased and more forks emerge.
(4) *Consumption*. Most existing works carry out redaction at the expense of consuming network resources. Many blockchain-based applications cannot provide such network resources, such as the blockchain-based IoT/IIoT system.[30,31,80] According to the real requirements of applications, the mechanisms are designed to get a trade-off between security, redaction, and network resources consumption.
(5) *Application*. It is also an interesting topic to design redaction mechanisms for various blockchain-based applications.[81-84] For example, the redactable blockchains supporting removal are used to optimize the memory of edge devices in the IoT system.[19,85] The chameleon hash-based redactable blockchain is used to implement fine-grained access control with the attribute update[86] and also used to manage users' identities and authenticate users in the mobile networks.[87] In these applications, some focus on strong security while some require efficiency and functionality. Therefore, future research on redactable blockchains should consider the requirements of the practical applications.

7 | CONCLUSION

This paper has conducted a survey on mutable blockchains. The purpose of making blockchain redactable is to edit the objectionable data that may harm others and even break the law. We first analyze the impact of data insertion on the blockchain, including the inserted data type, the position where the data is inserted, and the harm caused by inserting data. Next, we summarize the challenges of designing the redaction mechanism and propose a list evaluation criteria. Then, we comprehensively review and analyze the state-of-the-art redaction mechanisms which are presented from four categories. Moreover, we compare these mechanisms with respect to security, limitations, and performance in detail. Our algorithm outlines, security analyses, and performance comparisons on these mutable blockchains in this survey can help researchers and developers to design mechanisms in the future.

ACKNOWLEDGEMENTS

This study is supported in part by National Key Research and Development Program of China (Grant no. 2018AAA0100101), in part by National Natural Science Foundation of China (Grant no. 61932006, 61772434, and 61806169), in part by Chongqing technology innovation and application development project (Grant no. cstc2020jscxmsxmX0156), and in part by the scholarship from China Scholarship Council (CSC).

ORCID

Di Zhang https://orcid.org/0000-0002-4875-1319
Junqing Le https://orcid.org/0000-0003-2240-5263
Xinyu Lei https://orcid.org/0000-0001-8799-7875
Tao Xiang https://orcid.org/0000-0002-9439-4623
Xiaofeng Liao https://orcid.org/0000-0002-3372-9207

REFERENCES

1. Ma Z, Jiang M, Gao H, Wang Z. Blockchain for digital rights management. *Future Gener Comput Syst*. 2018; 89:746-764.
2. Aitzhan NZ, Svetinovic D. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Trans Depend Secure Comput*. 2016;15(5):840-852.
3. Khan KM, Arshad J, Khan MM. Investigating performance constraints for blockchain based secure e-voting system. *Future Gener Comput Syst*. 2020;105:13-26.
4. Xu Y, Ren J, Zhang Y, Zhang C, Shen B, Zhang Y. Blockchain empowered arbitrable data auditing scheme for network storage as a service. *IEEE Trans Services Comput*. 2019.
5. De Aguiar EJ, Faiçal BS, Krishnamachari B, Ueyama J. A survey of blockchain-based strategies for healthcare. *ACM Comput Surveys*. 2020;53(2):1-27.
6. Hayrutdinov S, Saeed A, Tong L. Blockchain technology adoption via contractual coordination mechanisms. In: *2020 the 3rd International Conference on Blockchain Technology and Applications*, Association for Computing Machinery, New York, NY, ICBTA 2020; 2020:19-26.
7. Abidi MH, Alkhalefah H, Umer U, Mohammed MK. Blockchain-based secure information sharing for supply chain management: Optimization assisted data sanitization process. *Int J Intell Syst*. 2020;36(1):260-290.
8. Sun Z, Wang Y, Cai Z, Liu T, Tong X, Jiang N. A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing. *Int J Intell Syst*. 2021.
9. Liu S. Global blockchain solutions spending 2017-2023; 2019. [Online]. https://www.statista.com/statistics/800426/worldwide-blockchain-solutions-spending/
10. Zhang D, Le J, Mu N, Liao X. An anonymous off-blockchain micropayments scheme for cryptocurrencies in the real world. *IEEE Trans Syst Man Cybernet Syst*. 2020;50(1):32-42.
11. Wang S, Ouyang L, Yuan Y, Ni X, Han X, Wang FY. Blockchain-enabled smart contracts: Architecture, applications, and future trends. *IEEE Trans Syst Man Cybernet Syst*. 2019;49(11):2266-2277.
12. Matzutt R, Hiller J, Henze M, Ziegeldorf JH, Müllmann D, Hohlfeld O, Wehrle K. A quantitative analysis of the impact of arbitrary blockchain content on bitcoin. In: *International Conference on Financial Cryptography and Data Security*, Springer; 2018:420-438.
13. Tziakouris G. Cryptocurrencies-a forensic challenge or opportunity for law enforcement? An interpol perspective. *IEEE Security Privacy*. 2018; 16(4): 92-94.
14. Conti M, SandeepKumar E, Lal C, Ruj S. A survey on security and privacy issues of bitcoin. *IEEE Commun Surveys Tutorials*. 2018;20(4):416-3452.
15. Huang K, Zhang X, Mu Y, Rezaeibagha F, Du X. Scalable and redactable blockchain with update and anonymity. *Inform Sci*. 2021;546:25-41.
16. Dorri A, Kanhere SS, Jurdak R. Mof-bc: a memory optimized and flexible blockchain for large scale networks. *Future Gener Comput Syst*. 2019;92:357-373.
17. Pyoung CK, Baek SJ. Blockchain of finite-lifetime blocks with applications to edge-based iot. *IEEE Internet Things J*. 2019.

18. Ali MS, Vecchio M, Pincheira M, Dolui K, Antonelli F, Rehmani MH. Applications of blockchains in the internet of things: a comprehensive survey. *IEEE Commun Surveys Tutorials*. 2018;21(2): 1676-1717.
19. Mendki P. Blockchain enabled iot edge computing: Addressing privacy, security and other challenges. In: *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, Association for Computing Machinery, New York, NY, ICBCT'20; 2020:63-67.
20. GDPR. General Data Protection Regulation; 2016. [Online]. https://gdpr-info.eu
21. Campanile L, Iacono M, Marulli F, Mastroianni M. Designing a GDPR compliant blockchain-based iov distributed information tracking system. *Inform Process Manage*. 2021; 58(3): 102511.
22. Politou E, Casino F, Alepis E, Patsakis C. Blockchain mutability: challenges and proposed solutions. *IEEE Trans Emerging Topics Comput*; 2019.
23. Deuber D, Magri B, Thyagarajan SAK. Redactable blockchain in the permissionless setting. In: *2019 IEEE Symposium on Security and Privacy*; 2019:124-138.
24. Thyagarajan SAK, Bhat A, Magri B, Tschudi D, Kate A. Reparo: Publicly verifiable layer to repair blockchains. arXiv preprint arXiv:2001.00486. 2020. http://arXiv.org/abs/arXiv:2001.00486
25. Marsalek A, Zefferer T. A correctable public blockchain. In: *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*; 2019:554-561.
26. Ateniese G, Magri B, Venturi D, Andrade E. Redactable blockchain—or—rewriting history in bitcoin and friends. In: *2017 IEEE European Symposium on Security and Privacy*; 2017:111-126.
27. Derler D, Samelin K, Slamanig D, Striecks C. Fine-grained and controlled rewriting in blockchains: chameleon-hashing gone attribute-based. *IACR Cryptol ePrint Archive*. 2019;2019:406.
28. Huang K, Zhang X, Mu Y, Wang X, Yang G, Du X, Rezaeibagha F, Xia Q, Guizani M. Building redactable consortium blockchain for industrial internet-of-things. *IEEE Trans Industr Inform*. 2019; 15(6):3670-3679.
29. Huang K, Zhang X, Mu Y, Rezaeibagha F, Du X, Guizani N. Achieving intelligent trust-layer for internet-of-things via self-redactable blockchain. *IEEE Trans Industr Inform*. 2020;16(4):2677-2686.
30. Zhang C, Ni Z, Xu Y, Luo E, Chen L, Zhang Y. A trustworthy industrial data management scheme based on redactable blockchain. *J Parallel Distributed Comput*. 2021;152:167-176.
31. Zhang J, Lu Y, Liu Y, Yang X, Qi Y, Dong X, Wang H. Serving at the edge: a redactable blockchain with fixed storage. In: Wang G, Lin X, Hendler J, Song W, Xu Z, Liu G, eds. *Web Information Systems and Applications*. Cham: Springer International Publishing; 2020:654-667.
32. Wu C, Ke L, Du Y. Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain. *Inform Sci*. 2021;548:438-449.
33. Ashritha K, Sindhu M, Lakshmy KV. Redactable blockchain using enhanced chameleon hash function. In: *2019 5th International Conference on Advanced Computing Communication Systems*; 2019:323-328.
34. K, Samelin, Slamanig D. Policy-based sanitizable signatures. In: Jarecki S ed, *Topics in Cryptology—CT-RSA 2020*. Cham: Springer International Publishing 2020: 538-563.
35. Puddu I, Dmitrienko A, Capkun S. μ chain: How to forget without hard forks. *IACR Cryptol ePrint Archive*. 2017:106.
36. Florian M, Henningsen S, Beaucamp S, Scheuermann B. Erasing data from blockchain nodes. In: *2019 IEEE European Symposium on Security and Privacy Workshops*; 2019:367-376.
37. Matzutt R, Kalde B, Pennekamp J, Drichel A, Henze M, Wehrle K. How to securely prune bitcoin's blockchain. arXiv preprint arXiv:2004.06911. 2020. http://arXiv.org/abs/arXiv:2004.06911
38. Li J, Yuan Y, Wang FY. Bitcoin fee decisions in transaction confirmation queueing games under limited multi-priority rule. In: *2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*; 2019:134-139.
39. Okupski K. Bitcoin developer reference. *Eindhoven*; 2014.
40. Belotti M, Božić N, Pujolle G, Secci S. A vademecum on blockchain technologies: when, which, and how. *IEEE Commun Surveys Tutorials*. 2019;21(4):3796-3838.
41. Kawase Y, Kasahara S. Transaction-confirmation time for bitcoin: queueing analytical approach to blockchain mechanism. In: Yue W, Li QL, Jin S, Ma Z, eds. *Queueing Theory and Network Applications*. Cham: Springer International Publishing; 2017:75-88.

42. Zhao W, Jin S, Yue W. Analysis of the average confirmation time of transactions in a blockchain system. In Phung-Duc T, Kasahara S, Wittevrongel S, eds. *Queueing Theory and Network Applications*. Cham: Springer International Publishing; 2019: 379-388.
43. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system; 2008. [Online]. https://bitcoin.org/bitcoin.pdf
44. Akbari E, Zhao W, Yang S, Luo X. The impact of block parameters on the throughput and security of blockchains. In: *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, Association for Computing Machinery, New York, NY, ICBCT'20; 2020:13-18.
45. Kiayias A, Panagiotakos G. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019; 2015. https://eprint.iacr.org/2015/1019
46. Eyal I, Gencer AE, Sirer EG, Van Renesse R. Bitcoin-ng: A scalable blockchain protocol. In: *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, USENIX Association, NSDI'16; 2016:45-59.
47. Sompolinsky Y, Zohar A. Secure high-rate transaction processing in bitcoin. In Böhme R, Okamoto T, eds. *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer; 2015:507-527.
48. Bentov I, Gabizon A, Mizrahi A. Cryptocurrencies without proof of work. In Clark J, Meiklejohn S, Ryan PY, Wallach D, Brenner M, Rohloff K, eds. *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer; 2016:142-157.
49. Gaži P, Kiayias A, Zindros D. Proof-of-stake sidechains. In *2019 IEEE Symposium on Security and Privacy (SP)*; 2019:139-156.
50. Bentov I, Lee C, Mizrahi A, Rosenfeld M. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Perform Eval Rev.* 2014;42(3):34-37.
51. Castro M, Liskov B, et al. Practical byzantine fault tolerance. In: *OSDI*, Vol 99; 1999:173-186.
52. Duan S, Reiter MK, Zhang H. Beat: Asynchronous bft made practical. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Association for Computing Machinery, New York, NY, CCS'18; 2018:2028-2041.
53. Kokoris-Kogias E, Jovanovic P, Gailly N, Khoffi I, Gasser L, Ford B. Enhancing bitcoin security and performance with strong consistency via collective signing. In: *Proceedings of the 25th USENIX Conference on Security Symposium*, USENIX Association, SEC'16; 2016:279-296.
54. Xiao Y, Zhang N, Lou W, Hou YT. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun Surveys Tutorials*. 2020.
55. Garay J, Kiayias A, Leonardos N. The bitcoin backbone protocol: analysis and applications. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer; 2015:281-310.
56. Shirriff K. Hidden surprises in the Bitcoin blockchain and how they are stored: Nelson Mandela, Wikileaks, photos, and Python software; 2014. [Online]. http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html#ref8
57. Hyena. CryptoGraffiti; 2017. [Online]. https://cryptograffiti.info/
58. Bartoletti M, Pompianu L. An analysis of bitcoin op_return metadata. *Lecture Notes in Computer Science*. 2017.
59. Matzutt R, Henze M, Ziegeldorf JH, Hiller J, Wehrle K. Thwarting unwanted blockchain content insertion. In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*, IEEE; 2018:364-370.
60. Le Calvez A. Non-standard P2SH scripts; 2015. [Online]. https://medium.com/@alcio/non-standard-p2sh-scripts-508fa6292df5
61. HugPuddle. Apertus – Archive data on your favorite blockchains; 2013. [Online]. http://apertus.io/
62. Casatta R. Eternity wall; 2018. [Online]. https://github.com/eternitywall/eternitywall.com
63. Blockstack. Blockstore; 2015. [Online]. https://github.com/blockstack-packages/blockchain-id-deprecated/wiki/Blockstore
64. Sun G, Dai M, Sun J, Yu H. Voting-based decentralized consensus design for improving the efficiency and security of consortium blockchain. *IEEE Internet Things J*. 2021;8(8):6257-6272.
65. Peng W, Yi L, Fang L, XinHua D, Ping C. Secure and traceable copyright management system based on blockchain. In: *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*; 2019: 1243-1247.
66. Gao J, Yu H, Zhu X, Li X. Blockchain-based digital rights management scheme via multiauthority ciphertext-policy attribute-based encryption and proxy re-encryption. *IEEE Syst J*. 2021:1-12.
67. Truong NB, Sun K, Lee GM, Guo Y. GDPR-compliant personal data management: a blockchain-based solution. *IEEE Trans Inform Forensics Security*. 2020; 15: 1746-1761.

68. Dousti MS, Küpçü A. Moderated redactable blockchains: a definitional framework with an efficient construct. In Garcia-Alfaro J, Navarro-Arribas G, Herrera-Joancomarti J, eds. *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham: Springer International Publishing; 2020:355-373.
69. Kiayias A, Russell A, David B, Oliynykov R. Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz J, Shacham H, eds. *Advances in Cryptology—CRYPTO 2017*. Cham: Springer International Publishing; 2017:357-388.
70. Saad M, Spaulding J, Njilla L, Kamhoua C, Shetty S, Nyang D, Mohaisen D. Exploring the attack surface of blockchain: a comprehensive survey. *IEEE Commun Surveys Tutorials*. 2020;22(3):1977-2008.
71. Krawczyk H, Rabin T. Chameleon signatures. In: *2000 Network and Distributed System Security Symposium (NDSS)*; 2000.
72. Camenisch J, Derler D, Krenn S, Pöhls HC, Samelin K, Slamanig D. Chameleon-hashes with ephemeral trapdoors. In: *IACR International Workshop on Public Key Cryptography*, Springer; 2017:152-182.
73. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: *2007 IEEE Symposium on Security and Privacy*; 2007:321-334.
74. Baron J, ElDefrawy K, Lampkins J, Ostrovsky R. How to withstand mobile virus attacks, revisited. In: *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*; 2014:293-302.
75. Grigoriev D, Shpilrain V. Rsa and redactable blockchains. Cryptology ePrint Archive, Report 2020/069; 2020. https://eprint.iacr.org/2020/069
76. Bilzhause A, Pöhls HC, Samelin K. Position paper: The past, present, and future of sanitizable and redactable signatures. In: *Association for Computing Machinery*, New York, NY, ARES'17; 2017.
77. Chen X, Lin S, Yu N. Bitcoin blockchain compression algorithm for blank node synchronization. In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*; 2019:1-6.
78. Carvalho A, Merhout JW, Kadiyala Y, Bentley J II. When good blocks go bad: managing unwanted blockchain data. *Int J Inform Manage*. 2021;57(4):102263.
79. Croman K, Decker C, Eyal I, et al. On scaling decentralized blockchains. In: *International Conference on Financial Cryptography and Data Security*, Springer; 2016:106-125.
80. Chen W, Xu Z, Shi S, Zhao Y, Zhao J. A survey of blockchain applications in different domains. In: *Proceedings of the 2018 International Conference on Blockchain Technology and Application*. Association for Computing Machinery, New York, NY, ICBTA 2018; 2018:17-21.
81. Abuidris Y, Kumar R, Wenyong W. A survey of blockchain based on e-voting systems. In: *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*, Association for Computing Machinery, New York, NY, ICBTA 2019; 2019:99-104.
82. Yang W, Guan Z, Wu L, Du X, Lv Z, Guizani M. Autonomous and privacy-preserving energy trading based on redactable blockchain in smart grid. In: *GLOBECOM 2020—2020 IEEE Global Communications Conference*; 2020:1-6.
83. Wang H, He D, Choo KKR, Chen X. Blockchain-based multi-party proof of assets with privacy preservation. *Inform Sci*. 2021;547:609-621.
84. López D, Farooq B. A multi-layered blockchain framework for smart mobility data-markets. *Transport Res Part C: Emerging Technol*. 2020;111:588-615.
85. Qi S, Lu Y, Zheng Y, Li Y, Chen X. CPDS: enabling compressed and private data sharing for industrial internet of things over blockchain. *IEEE Trans Industr Inform*. 2021;17(4):2376-2387.
86. Yu G, Zha X, Wang X, et al. Enabling attribute revocation for fine-grained access control in blockchain-iot systems. *IEEE Trans Engineering Manage*. 2020:1-18.
87. Xu J, Xue K, Tian H, Hong J, Wei DSL, Hong P. An identity management and authentication scheme based on redactable blockchain for mobile networks. *IEEE Trans Vehicular Technol*. 2020:1.

How to cite this article: Zhang D, Le J, Lei X, Xiang T, Liao X. Exploring the redaction mechanisms of mutable blockchains: a comprehensive survey. *Int J Intell Syst*. 2021;36:5051-5084. https://doi.org/10.1002/int.22502