Iraqi Journal for Computer Science and Mathematics

Journal Homepage: http://journal.esj.edu.iq/index.php/IJCM
e-ISSN: 2788-7421 p-ISSN: 2958-0544

Enhancing Efficiency and Fine-Grained Control in Redactable Blockchains with EPBCHF

Shams Mhmood Abd Ali[1,2], Mohd Najwadi Yusoff [1,*], Hasan Falah Hasan[1,3]

[1]School of Computer Science, Universiti Sains Malaysia (USM),
Main Campus, Gelugor, Penang, 11700, Malaysia.
[2]School of Arts, Al-iraqiya University,
AlAzhamiya - Haibba Khatoon Street, Baghdad, Iraq.
[3]School of Engineering, Al-iraqiya University,
AlAzhamiya - Haibba Khatoon Street, Baghdad, Iraq.

*Corresponding Author: Mohd Najwadi Yusoff.

DOI: https://doi.org/10.52866/ijcsm.2024.05.03.010
Received February 2024; Accepted April 2024; Available online July2024

ABSTRACT: Blockchain technology has presented a promising decentralized paradigm to preclude trusted third parties' dominancy. It is a transparent and distributed ledger initially designed for digital cryptocurrencies while currently extended to serve various industries. However, Blockchain immutability presents challenges, as it can be misused for storing illicit content, violating privacy regulations, and limiting data management flexibility. Policy Based Chameleon Hash Function (PBCH) has transformed blockchain rewriting contents concept via permitting modifiers to amend certain transaction since they possessed fundamental privileges satisfying certain access policy. However, PBCHF suffers from efficiency issues due to its reliance on Chameleon Hash ephemeral Trapdoor (CHET) and Attribute-Based Encryption (ABE), significantly impacting overall efficiency. We propose the Efficient Policy-Based Chameleon (EPBCHF) construction by replacing CHET with Chameleon-Hashes by Dual Long-Term Trapdoors (CHDLTT) to address these challenges. Additionally, we introduce an enhanced encryption scheme resilient against chosen-ciphertext attacks (CCA) without compromising overall efficiency. Modelling EPBCHF proves practical instantiation accompanied by rigorous security proofs. Our construction provides a fine-grained redactable blockchain in comparison to the currently proposed solutions. The evaluated results confirm that the proposed EPBCHF is scalable and efficient due to having the ability to handle unlimited transaction volumes additionally, data is efficiently processed without further overhead meanwhile data size consistency reflects a robust memory management due to predicted memory size, network bandwidth and storage requirement for future growth thereby, EPBCHF is proven to be reliable and scalable.

Keywords: Blockchain Technology; Redactable Blockchains; Chameleon Hash (CHF); Dual Long-Term Trapdoors(DLTTF); Attribute-Based Encryption (ABE); Efficiency; Fine-Grained Redaction.

1. INTRODUCTION

Blockchain is a modern, cutting-edge technology that has become a prime example of decentralized computing paradigms. Its primary objective is to reduce the influence of trusted third parties. It is a transparent, distributed, and public ledger with several chained blocks built specifically to backbone digital cryptocurrency since its founding. Simultaneously, a wide range of vital industries are currently considering its services [1], where it has acquired academic and industry attention resulting in the emergence of numerous applications such as copyright conflict resolution [2], product traceability [3], voting [4], storage services [5], healthcare sector [6], supply chain [7] and IoT data management [8] due to Its unique formation is constituted by P2P networked participants who disseminate transactions to be validated by powerful elected nodes named miner meanwhile validation, and transaction inclusion within a block are regulated by consensus algorithm [9],[10] via complex mathematical puzzles solved by miners in order to grant them that right. Additionally, security and decentralization are blockchain's primary features. Security ensures content integrity or immutability via prohibiting content modification once approved leads to be settled in a block, while decentralization

refers to the possession of a distributed ledger locally at each participant's side; thereby, any suspicious activities can be disclosed [11].

1.1 MOTIVATION

Immutability is a substantial content soundness assurance mechanism that, in essence, ensures blockchain ledger transparency and integrity, where any block content modifications require unanimous approval from all participants. However, it is unnecessary to be without passive impact, which also impacts blockchain growth due to protecting illicit content. Matzut et al. [12], in 2018, eight records contained inappropriate sexual content; additionally, records with links to child abuse exploitation were connected to the dark web, thus discouraging further users from investing in blockchain due to possible legal and ethical consequences. Additionally, regulatory demands like the European General Data Protection Regulations (GDPR) [13], [14] which rise as immutability confronted issue in which GDPR imposes the handling the freedom of data amending by the rightful owners and thus, redactable blockchain is an urgent demand to be considered. Redactable blockchain is a variant that permits data modification but with predefined constraints, which further increases the credibility of different user types. Moreover, handling data management securely and anonymously is essential, specifically in specific sectors such as healthcare banking, and even optimizing systems efficiency as in IoT Blockchain systems [15][16].

1.2 EXITING SOLUTION

Several redactable blockchain proposals have been suggested to modify content without violating its transparency or integrity. Ateniese et al. (2017) [17] have proposed a block-level redactable mechanism that relies, in essence, on replacing the standard one-way collision resistance hash function with a Chameleon Hash Function (CHF) where it has the advantage of acquiring a pair of public and trapdoor keys. The Trapdoor key is leveraged for collision generation, resulting in modifying blockchain content; however, block-level redaction can be an issue due to granting the trapdoor key for the entire block in which all content can be amended without leaving a trace further, the transaction owner is unaware of any changed performed therefore, an extra access control restrictions are demanded. Derler et al. (2019) [19] have benefited from Ateniese's proposal drawback via designing Policy Based Chameleon Hash Function (PBCH). They primarily present an immaculate transaction level redaction mechanism aided with fine-grained access control through combining Chameleon Hash Ephemeral Trapdoor (CHET) [20] with Attribute-Based Encryption (ABE) [21]. The CHET comprises two keys. First is a long-termed key, generated by a central authority and distributed to all permitted modifiers in the setup phase, while the transaction owner produces the short-term trapdoor key. The short-term trapdoor key is encrypted using one of the ABE variants associated with a predefined access policy. Modifiers that satisfy previously set attributes are enabled to decrypt the trapdoor key and modify the relevant transaction.

1.3 RESEARCH PROBLEM

PBCH represents a significant cryptographic tool to redact data within redactable blockchain; nevertheless, its shortcomings require careful consideration, especially efficiency and its effectiveness, due to inherited CHET and ABE efficiency limitations. Current paradigms employ CHET that, in essence, produces an individual ephemeral trapdoor key for each mutable transaction regardless of whether a single or a set of transactions are directed to be modified by the same modifier or under a similar access policy; thereby, security will be strengthened, and in contrast, efficiency is affected. Furthermore, despite ABE's security strength, it remains vulnerable to Chosen-Ciphertext Attacks (CCA) [23]. These implications are significant obstacles in the light of increasing demands on solid encryption in ranged applications such as blockchain in order to secure transmitted data wherein; several enhancements have been performed to mitigate these vulnerabilities, but it comes at the expense of the system efficiency [23][25] Meanwhile, current real-world extensively require a further balance among security and efficiency.

1.4 THIS WORK

In this research, we aim to improve computational efficiency with a preserved security level by employing Krenn et al. (2018)[22] proposed Chameleon-Hashes with Dual Long-Term Trapdoors Function (CHDLTTF). It is a unique advancement and an ideal replacement to the CHET mechanism that enables the usage of the similar ephemeral trapdoor key reusability, exceptionally when a set of mutable transactions are meant to be modified by a similar attribute handler or access policy where it leads to optimizing efficiency via reducing computational overhead and in contrast, security is maintained. The Boneh-Katz transformation (BK) technique [26][27] is also employed within the ABE scheme, where it reinforces its security to resist CCA attacks while computational efficiency is assured.

1.5 CONTRIBUTION

- EPBCHF Formulization: EPBCHF introduces an accurate formal definition that supports PBCH by employing the CHDLTTF. Other concepts, such as full indistinguishability and collision resistance within an adversarial context, are also introduced.

- EPBCHF Implementation: A full implementation is conducted while a performance assessment between our proposal, traditional PBCH, and real-world applications.

1.6 RELATED WORKS

The current state of the art indicates a great interest in data redaction within blockchain via tackling impacting factors; meanwhile, proposed solutions have been broadly classified into Chameleon and non-chameleon redaction mechanisms [28]. Non-chameleon solutions encountered efficiency limitations such as computation time, network bandwidth, and storage overhead. Additionally, it imposes a seriously compromising blockchain infrastructure, resulting in further cost burdens. Chameleon based redaction mechanism was substantially dedicated to permissioned blockchain by Ateniese et al. (2017) [17], which proposed block-level rewriting with coarse-grained access control built upon the usage of CHF[18] and Public Key Infrastructure (PKI). The employment of a central controlling authority is fundamental due to its role in granting and editing modifiers' privileges using secret keys. Subsequently, numerous other researchers' efforts drew inspiration from Ateniese's proposal. They primarily focused on addressing single-key exposure [29], [30],[31],[32],[33],[34],[35],[36],[37],[38],[39] resulted in key management mechanisms using Shamir's secret sharing or Multiparty Computation (MPC) however, fine-grained access control to support privilege rewriting management is not considered. Derler et al.(2019) [19] introduced a transaction rewriting level and fine-grained access control construction in redactable blockchain, also applied in permissioned settings where it relies on conventional hash function using Merkle root, unlike Ateniese's mechanism. They have originated a cryptographic notion called Policy-Based Chameleon Hash Function (PBCHF), derived from the combination of CHET[20] and ABE. CHET comprises a long-term trapdoor and a short-term ephemeral trapdoor. Central authority adhered to creating a long-term trapdoor and provided it to each modifier, while each transaction was assigned a short-term ephemeral trapdoor derived by the owner. Meanwhile, it is encrypted using the ABE scheme, ensuring that modifiers cannot perform rewriting without both trapdoors. Numerous redactable blockchains have been proposed based on PBCHF, offering various features such as accountability [40], [23], [41], punishment [42], [43], decentralization [44], [45], and revocability [25],[46],[24] mechanisms. For instance, Tian et al. (2020) [40] introduced PBCHF, emphasizing accountability by integrating CHET, CP-ABE, and digital signatures. However, weak accountability is compelled by the proposed solution, which links modified transactions to their modifiers via leaked keys. Guo et al. (2021) [23] suggested an Auditable Outsource Computation (OORB-AOC) scheme for hybrid online and offline rewriting. However, it lacks revocation monitoring for misbehaved modifiers. To remove harmful data, Hou et al. [41] presented a novel design for a redactable blockchain based on PBCHF and sanitizable signatures. Although their solution allows blockchain rewriting, it does not address data revocation or decentralized methods. The absence of punishment mechanisms in PBCHF also enables faulty modifiers to evade consequences.

Xu et al. (2021) [42] presented a blockchain redaction concept known as k-time modifiable and epoch-based redactable blockchain (KERB). The timeline in KERB is divided based on epochs; additionally, transaction modifiers are demanded to issue a locked deposit based on the time window; therefore, it cannot be claimed only beyond invalidating rewriting privilege. Modifiers allow 'k' modification operations within each epoch. Exceeding this limit leads to extracting the modifier's secret key and losing the locked deposit. Chen and Gao (2022) [43] introduced CDEdit, enabling controllable redaction permissions and various editing types. However, centralization and the absence of a punishment mechanism hinder its performance. To facilitate the transaction rewriting privilege in a decentralized environment, Zhang et al. (2021) [44] introduced a Multi-Authority ABE with CHF, while Ma et al. (2022) [45] presented a Decentralized Policy-Based Chameleon Hash (DPCH). Both solutions have leveraged the CHET algorithm to rewrite data with decentralized ABE to manage editing privileges.

Revoking access in PBCHF through CP-ABE encryption presents challenges due to the overhead of re-keying and re-encryption. Panwar et al. (2021) [25] have suggested the blockchain redaction method in permissioned settings with the trace and revoke ability. Traceability establishes a relation among modified transactions and their modifiers using dynamic group signature, while revocability can be used to withdraw the rewriting privilege from misbehaved modifiers. However, linear complexity is concerning due to the number of unrevoked modifiers within the revocation process. Moreover, secret key updates must be securely supplied within established channels. In contrast, Jia et al. (2021) [46] introduced hierarchical revocation. However, their approach assumes a semi-trusted modifier and does not comprehensively address accountability. Xu et al. (2021) [24] proposed a new cryptographic concept called Revocable Policy-Based Chameleon Hash (RPCH). RPCH is constructed from a new revocable ABE and CHET, whereas revocable ABE operates with logarithmic complexity and transfers updated secret keys through public channels. [21]. Additionally, Non-interactive zero-knowledge proofs (NIZK) can be utilized to verify encrypted trapdoors, as demonstrated by Ateniese et al. (2017) [17], Derler et al. (2020)[21], Panwar et al. (2021)[25], and Tian et al. (2020)[40]. Figure 2 provides an overview of the current state of PBCHF solutions, applications, mechanisms, advantages, and existing challenges. Table 1 illustrates an analysis of current solutions inspired by PBCHF and highlights specific gaps. Their extensive reliance on CHET utilization has a detrimental impact on overall efficiency. Implementing Fujisaki-Okamoto (FO) transformation algorithms mandates cipher validation via a re-encryption process to achieve the CCA security level. This procedure, as documented by Tian et al. [26] in 2022, also results in efficiency degradation. Our proposed solution

primarily focuses on enhancing efficiency by replacing CHET with CHDLTTF while concurrently achieving ABE with CCA security, as detailed in references [26], [27], and [47].

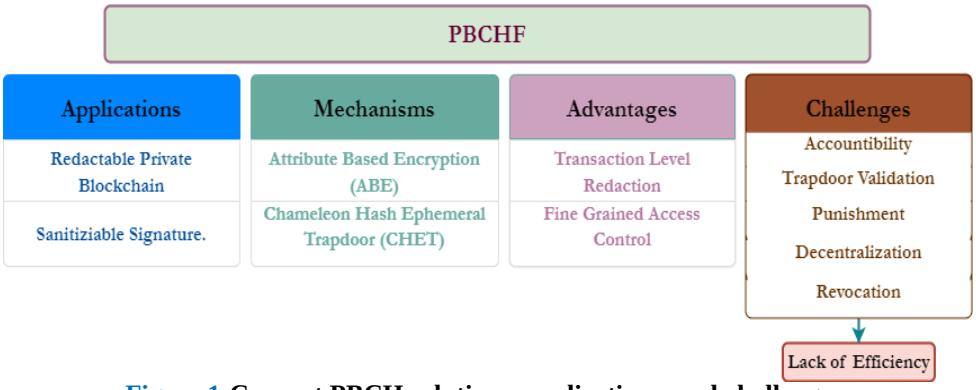


Figure 1:Current PBCH solutions, applications, and challenges

1.7 ROADMAP

The structure of the upcoming sections is outlined as follows: Section 2 presents a system overview. Section 3 delves into elucidating the cryptographic building block components utilized in our proposed solution. Section 4 will subsequently furnish an EPBCHF in detail. Section 5 delivers a specific comprehensive EPBCHF concrete construction. Section 6 will exclusively discuss results and comparisons; meanwhile, section 7 illustrates EPBCHF application, and final section 8 expresses the conclusion and future research directions.

Table 1. Current PBCH Solutions

Schema/Year	Chameleon Hash	CCA Mechanism	Methods	Advantage	Efficiency
Derler (2019) [19]	CHET	KEM, FO	PBCHF	Transaction level redaction, Fine-grained access control.	M
Tain (2020)[40]	CHET	-	APBCH	Accountability	M
Guo (2021) [23]	CHET	KEM, FO	APCH	Accountability, Enhanced Storage.	M
Huo (2021) [41]	CHET	PKE	PBCHF, DS	Accountability, Removing harmful data.	M
Xu(2021) [42]	CHET	-	PBPCHF	Accountability, Punishment.	M
Chen and Gao (2022) [43]	CHET	-	PBPCHF	Accountability, Punishment.	M
Zhang (2021)[44]	CHET	-	DPCH	Decentralization.	M
Wu(2021)[45]	CHET	KEM, FO	DPCH	Accountability, Decentralization.	M
Panwar(2021) [25]	CHET	KEM, FO	RPCH	Accountability, Decentralization, Anonymity, Revocation.	P
Jia (2021) [46]	CHET	-	HRCHET	Forward Secrecy, Backward Revocation Secrecy	M
Xu (2021) [24]	CHET	KEM, FO	RPCH	Backward Secrecy for Revocation	M
Our	**CHDLLT**	**BK**	**EPBCHF**	Enhance Efficiency in terms of execution time	**H**

2. SYSTEM OVERVIEW

The proposed system comprises three distinct participants. The Transaction Owner (TO) is responsible for appending transactions to the blockchain. Meanwhile, a set of Transaction Modifiers (TM) are adhered to modifying transactions based on their rewriting privileges, and finally, the Trusted Authority (TA) is committed to generating and distributing keys. In a decentralized context, the TA role can be assigned to any user to assign attributes to other users.

Modifier's numbers are deliberately kept small, as the rewriting privilege needs careful control and should not be granted to most users. The proposed system assumes a TA represents the central authority in a permissioned blockchain. The process begins with the transaction owner publishing modifiable transactions on the blockchain-based on two scenarios:

- **Multiple Transactions with the Same Access Policy:** If the TO has various transactions with the same access policy dedicated to similar modifiers, they utilize CHDLTTF to generate a second trapdoor key that can be used across multiple transactions, which is encrypted according to the predefined access policy using ABE, followed by the computation of a MAC to ensuring the ciphertext confidentiality and integrity. Modifiers equipped with the secret key satisfying the access policy can decrypt the trapdoor key, verify its validity, and proceed with the redaction process.
- **Other Scenarios:** the TO has either a single modifiable transaction or multiple transactions with different access policies, and a refreshed trapdoor key is generated for each transaction by following similar steps as previously mentioned. The validation burden is tremendously reduced in both scenarios on the TM side due to the demand to compute MAC to check the received ciphertext instead of encryption and re-encryption processes, thereby enhancing efficiency. Figure 3 illustrates the system overview.

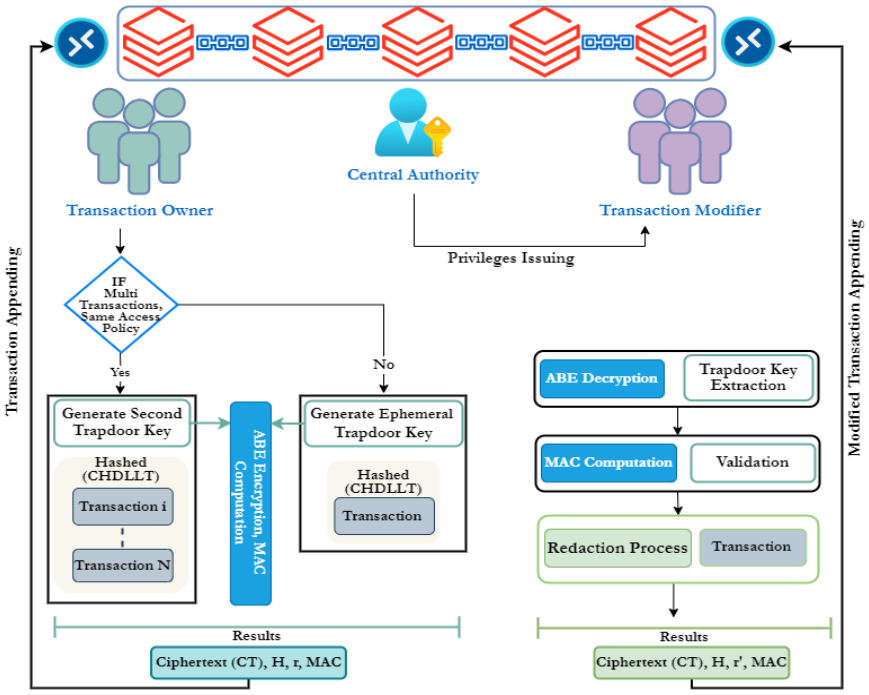


Figure 2: System Overview

3. CRYPTOGRAPHIC BUILDING BLOCKS

This section discusses the fundamental cryptographic components employed in the proposed EPBCHF construction, namely CHDLLT and ABE.

3.1 CHAMELEON-HASH WITH DUAL LONG-TERM TRAPDOORS FUNCTION

Chameleon-Hashes with Dual Long-Term Trapdoors Function (CHDLTTF), introduced by Krenn et al. in 2018 [22], is an advanced and extended cryptographic primitive than Chameleon-hashes [18]. From an external perspective, Chameleon hashes, and standard collision-resistant hash functions appear to be similar; however, Chameleon hash possesses an intriguing property, a specific secret key, known as a trapdoor, that aids the possibility of discovering arbitrary collisions within the hash function. To address the security challenges stemming from the knowledge of this long-term trapdoor, the CHET was first presented by Camenisch et al. [20]. CHET mitigates the risk of discovering collisions by holders due to its association with long-term trapdoors by introducing a second ephemeral trapdoor. However, the CHET generates an ephemeral trapdoor for each new hashing operation.

In contrast, CHDLTTF builds upon these principles and eliminates the necessity to generate a new second trapdoor for each new hashing operation. Instead, it offers hashing authority the flexibility to choose whether to issue a new second trapdoor or utilize an available one. This enhancement broadens the applicability of CHETs, making them more versatile and suitable for various scenarios. A set of six crucial algorithms define this cryptographic primitive, as shown in Definition 1, as outlined below [22].

Definition 1: Chameleon-Hashes with Dual Long-Term Trapdoors Function (CHDLTTF)

In this definition, we provide the aforementioned CHDLTTF algorithms in the following:

Algorithm	Description
CHDLTTF.PPGen (1^{λ})	The security parameter λ serves as the algorithm's input. It produces the public parameter pp_{CH} . (performed in TA).
CHDLTTF.KGen (pp_{CH})	PP_{CH} is the input for this algorithm, which produces a key pair, denoted as (Mpk_{CH}, Msk_{CH}) where (Mpk_{CH}), represents the master hashing key, and (Msk_{CH}), as master secrete trapdoor key. (perform TA)
CHDLTTF.SKGen (pp_{CH})	This algorithm's input is (pp_{CH}) , and it produces a key pair, denoted as (Spk_{CH}, Ssk_{CH}) , where (Spk_{CH}) is the second hash key, and (Ssk_{CH}) is the second trapdoor. (Performed by a hashing party).
CHDLTTF.HGen (Mpk_{CH}, Spk_{CH}, m)	It seeks the input of (Mpk_{CH}, Spk_{CH}) a message (m) and produces CHF as (hash value) random r (performed in the hashing party).
CHDLTTF.HVer $(Mpk_{CH}, Spk_{CH}, m, CHF, r)$	Inputs are $Mpk_{CH}, Spk_{CH}, m, CHF$ *and* r whereas outputs are $b \in \{0,1\}$. (Performed in verifying party)
CHDLTTF.Hcol $(Msk_{CH}, Ssk_{CH}, m, m', CHF, r)$	Input $Msk_{CH}, Ssk_{CH}, m, m', CHF$ and r meanwhile, random r' is the output. (perform in collision party).

Correctness

Note the verification algorithm consistently validates the given hash and produces $\perp$ if the hash is not valid. Moreover, CHDLLTF is proven to be correct for all values as For all $\lambda \in$,

$$
\begin{aligned}
&pp_{CH} \leftarrow \textbf{CHDLTTF.PPGen}(1^{\lambda});\\
&(Mpk_{CH}, Msk_{CH}) \leftarrow \textbf{CHDLTTF.KGen}(pp_{CH});\\
&(Spk_{CH}, Ssk_{CH}) \leftarrow \textbf{CHDLTTF.SKGen}(pp_{CH});\\
&(CHF, r) \leftarrow \textbf{CHDLTTF.HGen}(Mpk_{CH}, Spk_{CH}, m);\\
&r' \leftarrow \textbf{CHDLTTF.Hcol}(Msk_{CH}, Ssk_{CH}, m, m', CHF, r).
\end{aligned}
$$

We have the following:

$$\textbf{CHDLTTF.HVer}(Mpk_{CH}, Spk_{CH}, m, CHF, r) = \textbf{CHDLTTF.HVer}(Mpk_{CH}, Spk_{CH}, m', CHF, r') = 1.$$

For security assurance, CHDLTTF must demonstrate indistinguishability, making it challenging for any potential adversary to distinguish between the randomness generated during the hashing process and collisions. Additionally, CHDLTTF satisfies the collision resistance requirement, posing a formidable challenge for adversaries who lack access to both the master and second trapdoors when attempting to discover hash collisions. Please refer to the source in [22] for a complete and detailed definition.

3.2 ATTRIBUTE-BASED ENCRYPTION

Attribute-based encryption (ABE) is a cryptographic encryption scheme employed in cryptography to offer precise control over access to encrypted data. Unlike conventional cryptographic methods, ABE regulates access to encrypted information based on standard cryptographic keys and specific attributes or characteristics linked to users or data entities. Key-Policy ABE (KP-ABE) is a variant of ABE where access control policies are tied to cryptographic keys. Data owners define these policies, and users are issued keys that align with these predefined policies. Users can decrypt data solely if their keys are in line with the policy associated with the encrypted data. Furthermore, Ciphertext-Policy ABE (CP-ABE) represents another variant, where data owners encrypt their data with specific policies, and users are furnished with keys corresponding to their attributes. Users can access and decrypt data exclusively if their attributes match the policy linked with the encrypted data. CP-ABE greatly influences scenarios where data is shared among multiple users, each possessing

distinct attributes. In such cases, data owners can stipulate who can access their data based on user attributes, allowing for granular control over data access. The adaptability of CP-ABE in determining access policies makes it particularly suitable for various applications where data owners require precise control over data access[48]. A CP-ABE scheme encompasses the following four fundamental algorithms as in Definition 2. Like numerous encryption schemes, ABE schemes are commonly crafted to withstand CPA attacks. These attacks involve adversaries selecting plaintexts and observing the resulting ciphertexts. Achieving security against CCA attacks represents a more formidable objective and necessitates incorporating supplementary security measures and considerations. CCA-secure ABE schemes, while offering a higher level of security, often come with heightened computational complexity and potential trade-offs in efficiency. Ensuring protection against CCA attacks typically involves additional complexities to fortify encryption. For a comprehensive definition of ABE with CCA security, please refer to the paper [19].

Definition 2: Attribute-Based Encryption (ABE)

The ABE algorithms are illustrated in this section as follows:

Algorithms	Description
ABE.Setup $(1^{\lambda}, n)$	A security parameter λ and the number of participants n form the input, generating a key pair (Mpk_{ABE}, Msk_{ABE}), which Mpk_{ABE} represents the master public key and Msk_{ABE} represents the master secret key for CP-ABE (performed in trusted authority).
ABE.KGen (Msk_{ABE}, S_i)	It takes (Msk_{ABE}) the set of attributes based on participants as input in order to achieve the individual secret key sk_i. (performed by trusted authority).
ABE.Encrypt $(Mpk_{ABE}, \mathrm{A}, m)$	(Mpk_{ABE}), access structure A and message m are the required inputs, while the resulting ciphertext CT is the output. (Performed by encryptor side).
ABE.Decrypt (Mpk_{ABE}, CT, sk_i)	It requires $(Mpk_{ABE}, sk_i, and\ \mathrm{CT})$ input and outputs m or $\perp$ (performed by decryptor message).

3.3 ABE WITH CCA SECURITY

In this section, we delve into fortifying the security of our CP-ABE encryption scheme by achieving CCA security in order to improve security while preserving efficiency. Our approach entails the fusion of CP-ABE with the BK transformation, introducing a Message Authentication Code (MAC) function and incorporating a Key Encapsulation Mechanism (KEM) referred to as the weak Commitment. KEM provides an extra layer of security to our encryption process. Within this framework, we leverage two independent hash functions, denoted as 'h' and 'H.' The initial hash function 'h' generates a public commitment based on confidential random value 'r,' while the subsequent function 'H' derives a key (hid) to be utilized as a commitment (hiding process) using similar 'r' according to the equation *(key=h(r), hid=H(r), bid=r).* The *'r'* is thoughtfully integrated into the message encryption process. When encrypting a message,' as represented by the equation $CT = (hid \oplus (m \circ bid))$, we compute a MAC using *'key'* applied to the resultant ciphertext, ensuring the genuineness and integrity of our encrypted message. Upon decryption of the ciphertext, recipients initially retrieve both the message and the confidential random value. Subsequently, they verify whether the public commitment aligns with the *'key'* and whether the MAC appropriately validates the "*hid"*; consequently, this approach effectively ensures both CCA security and message authenticity within our encryption scheme. This algorithm draws inspiration from the concepts introduced in [47], which combine BK transformation [27] and RSA and Tian et al. [26] concepts represented by demonstrating the revocable Attribute-Based Encryption. Please refer to [27],[47] and [26] for further readings. A CP-ABE scheme with CCA security is constructed as Definition 3, utilizing the following fundamental algorithms:

Definition 3: ABE with CCA Security proposed by Tian et al. [26]

Algorithm	Description
ABE. Setup $'(1^{\lambda}, n)$ performed by TA.	A security parameter λ and the participants' numbers n are the inputs, producing a keypair mpk'_{ABE} as a public and secret key msk_{ABE}.
ABE.KGen (msk_{ABE}, S_i) performed by TA.	It takes (msk_{ABE}) and the set of attributes S_i based on participant i as input in order to achieve individual secret key sk_i.

ABE.Encrypt $(mpk_{ABE}^{'}, \mathrm{A}, m)$ Performed by the encryptor side.	$mpk_{ABE}^{'}$, access structure A and message m are the required inputs while resulting in ciphertext CT, hiding factor (hid), and the tag is represented as MAC_{tag} with are outputs as the following : **Encryption Process:** The sender initiates the process by encapsulating a random value using (KEM) through $\left(pub, 1^{\lambda}\right)$ which the pub is the public parameter to obtain (Key, hid, bid) ; randomness r is chosen from $\mathbf{Z}_p$, resulting in an encapsulation schema $(key = h(r), hid = H(r), bid = r)$, then applies CP-ABE with CPA security by selecting randomness $r^{'}$ from $\mathbf{Z}_p$ to produce ciphertext CT_1, where $CT_1 = \textbf{ABE.Encrypt}\left(\mathrm{A}, r', mpk_{ABE}\right)$, then proceeds to compute ciphertext CT_2, where $CT_2 = G(r^{'} \oplus (m \circ bid))$, where G represents a Pseudorandom Generator (PRG). MAC_{tag} is calculated by (CT_1, CT_2) using Key.
ABE.Decrypt $(mpk_{ABE}^{'}, \mathrm{A}, m)$ Performed by message decryptor	**ABE.Decrypt** $(mpk_{ABE}, CT, hid, MAC_{tag}, sk_i)$: It requires $(mpk_{ABE}, sk_i, MAC_{tag}$, hid, and CT as $(\mathrm{CT}_1, \mathrm{CT}_2))$ input and outputs m or $\perp$ the following : **Decryption Process:** The decryptor utilizes its secret key sk_i to decrypt CT_1 by applying the algorithm in CP-ABE, resulting in $r' =$ **ABE.Decrypt**(CT_1, sk_i, mpk_{ABE}) . This value is used to decrypt CT_2 obtain $r' = m \circ bid$, and compute $hid = H(bid)$. The output is m if $hid = H(bid)$, and the verification of the MAC_{tag} on (CT_1, CT_2) under key Key is valid.

4. EFFICIENT POLICY-BASED CHAMELEON HASH FUNCTION

4.1 FORMAL DEFINITION

This section presents three parties involved in our EPBCHF construction, TO, TM, and TA. Our formal definition consists of six consecutive algorithms, as shown in definition 4.

Definition 4: Efficient Policy-Based Chameleon Hash Function

This definition demonstrates the algorithms that contribute to building the proposed EPBCHF.

Algorithm		Description
EPBCHF.Setup $(1^{\lambda}, n)$ *Performed by TA*	**Input**	λ security parameter and n number of participants
	Run	$PP_{CH} \leftarrow$ **CHDLTTF.PPGen**(1^{λ}); $(Mpk_{CH}, Msk_{CH}) \leftarrow$ **CHDLTTF.KGen**(PP_{CH}); $(Mpk_{ABE}, Msk_{ABE}) \leftarrow$ **ABE.Setup**$(1^{\lambda}, n)$.
	Return:	$(Mpk_{CH}, Mpk_{ABE}) \leftarrow Mpk_{EPBCHF}$ ***and*** $(Msk_{CH}, Msk_{ABE}) \leftarrow Msk_{EPBCHF}$
EPBCHF.KGen $\left(Msk_{EPBCHF}, S_i\right)$ Performed by TA	**Input**	Msk_{EPBCHF}, parsed as (Msk_{CH}, Msk_{ABE}) an attribute set $S_i \in \mathrm{U}$, while resulted sk_{ii}.
	Run	$sk_i \leftarrow$ **ABE.KGen**(Msk_{ABE}, S_i)
	Return:	$sk_{ii} = (Msk_{CH}, Sk_i)$

EPBCHF.SKGen (PP_{CH}, A)
Performeb by TO

Input Public Parameter (PP_{CH}) and access policy (A), generates the key pair (Spk_{CH}, Ssk_{CH}) where (Spk_{CH}) is the second hash key, and (Ssk_{CH}) is the second trapdoor key.

Run $(Spk_{CH}, Ssk_{CH}) \leftarrow$ **CHDLTTF.SKGen** (pp_{CH})
$CT, MAC_{tag} \leftarrow$ **ABE.Encrypt'**$(Mpk_{ABE}(Ssk_{CH}, \mathrm{A}))$.

Return: $(Spk_{CH}, CT, MAC_{tag})$, *Where* ***CT*** *is the Ciphertext.*

EPBCHF.Hash (Mpk_{CH}, Spk_{CH}, m)
Performed by TO

Run $(CHF, r, \mathrm{CT}, \mathrm{MAC}_{tag}) \leftarrow$ **CHDLTTF.HGen**(Mpk_{CH}, Spk_{CH}, m)

Return: (*random* r ,*CHF* hash value, CT, MAC_{tag})

EPBCHF.Verify $(Mpk_{CH}, Spk_{CH}, m, CHF, r)$
Performed by TO, TA, TM

Run Input $(Mpk_{CH}, Spk_{CH}, m, CHF)$.

Run **CHDLTTF.HVer**$(Mpk_{CH}, Spk_{CH}, m, CHF, r) = 1$

Return: *return* 1 if the following check is true; *otherwise,* 0. bit $b \in \{0,1\}$

EPBCHF.Hcol $\begin{pmatrix} sk_{ii}, r', m, CHF, r, \\ CT, MAC_{tag} \end{pmatrix}$
Performed by TM

Input parse sk_{ii}, as (sk_i , Msk_{CH}), and m', m, CHF, r, CT, MAC_{tag}

Run *Check whether* :
EPBCHF.Verify$(Mpk_{CH}, Spk_{CH}, m, CHF, r) = 1$
$Ssk_{CH} \leftarrow$ **ABE.Decrypt**(sk_i, CT, MAC_{tag})
$r =$ **CHDLTTF.HCOL**$(Msk_{CH}, Ssk_{CH}, m, m', CHF, r)$
return $\perp$, ***if*** $Ssk_{CH} = \perp$
return $\perp$, ***if***
EPBCHF.Verify $(Mpk_{CH}, Spk_{CH}, m', CHF, r') = 0$ and r', ***otherwise***.

Return: (randomness r' ,chameleon hash CHF)

Correctness

The EPBCHF is correct if for all:

$(Mpk_{EPBCHF}, Msk_{EPBCHF}) \leftarrow$ **EPBCHF.Setup**$(n, 1^{\lambda}), \lambda \in$;
$sk_{ii} \leftarrow$ **EPBCHF.KGen**$(Msk_{EPBCHF}, S_i), S_i \in \mathrm{U}$;
$(Spk_{CH}, CT, MAC_{tag}) \leftarrow$ **EPBCHF.SKGen**$(PP_{CH}, \mathrm{A}), \lambda \in$;
$(r, CHF, CT, MAC_{tag}) \leftarrow$ **EPBCHF.Hash**$(Mpk_{CH}, Spk_{CH}, m), m \in \mathrm{M}$;
$(r') \leftarrow$ **EPBCHF.Hcol**$(sk_{ii}, m', m, r, CHF, CT, MAC_{tag})$.

We have That,

EPBCHF.Verify$(Mpk_{CH}, Spk_{CH}, m, CHF, r) =$ **EPBCHF.Verify**$(Mpk_{CH}, Spk_{CH}, m', CHF, r') = 1$

4.2 SECURITY MODEL

In this section, we delve into the security aspects of EPBCHF, particularly emphasizing indistinguishability and collision resistance. Indistinguishability guarantees that an adversary cannot conclusively differentiate between the randomness generated by the Chameleon hash and that produced by the collision algorithm. We will present a security experiment to demonstrate this property as follows:

	Analysis Steps
Indistinguishability	$\mathbf{Exp}^{Ind}_{EPBCHF,\mathrm{A}}\left(1^{\lambda}\right):\left(Msk_{EPBCHF},Mpk_{EPBCHF}\right)\leftarrow\mathbf{EPBCHF.Setup}\left(n,1^{\lambda}\right);b\leftarrow\{0,1\}$ $b^{'}\leftarrow \mathrm{A}^{\mathrm{O}_{\text{Hash or Hcol}}(Msk_{EPBCHF},\cdots,b)}\left(Mpk_{EPBCHF}\right),\underline{return}\ 1,\textbf{if}\ (b^{'}=b),\textbf{else}\ \underline{return}\ 0$ $Oracle\mathrm{O}_{\text{Hash or Hcol}}\left(sk_{ii},S_i,m,m^{'},\mathrm{A},b\right):$ $sk_{ii}\leftarrow\mathbf{EPBCHF.KGen}\left(Msk_{EPBCHF},S_i\right);\left(Spk_{CH},CT,MAC_{tag}\right)\leftarrow\mathbf{EPBCHF.SKGen}\left(pp_{CH},\mathrm{A}\right)$ $\left(r_0,CHF_0,CT^0,MAC^0{}_{tag}\right)\leftarrow EPBCHF.Hash\left(Mpk_{CH},Spk_{CH},m^{'}\right);$ $\left(r_1,CHF_1,CT^1,MAC^1{}_{tag}\right)\leftarrow EPBCHF.Hash\left(Mpk_{CH},Spk_{CH},m\right);$ $r_1\leftarrow EPBCHF.Hcol\left(sk_{ii},m^{'},m,CHF_1,r_1,CT^1,MAC^1{}_{tag}\right);\underline{return}\left(r_b,CHF_b,CT^b,MAC^b{}_{tag}\right)$

The EPBCHF scheme is deemed indistinguishability secure if the chance of any polynomial-time adversary $\mathcal{A}$ is negligible, as defined below:

$$\mathrm{Adv}^{IND}_{EPBCHF,\mathrm{A}}\left(1^{\lambda}\right)=\left|\Pr\left[\mathrm{Exp}^{IND}_{EPBCHF,\mathrm{A}}(1)=1\right]-1/2\right|.$$

Collision resistance states that an adversary cannot find collisions for computed hashes with access policies they are dissatisfied with. This property is demonstrated by the provided security experiment below:

Collision Resistance Analysis Steps
$\boldsymbol{Exp}^{CR}_{EPBCHF,A}\left(1^{\lambda}\right):\left(Msk_{EPBCHF},Mpk_{EPBCHF}\right)\leftarrow\mathbf{EPBCHF.Setup}\left(n,1^{\lambda}\right);\text{set}\ \Lambda_1,\Lambda_2,\Lambda_3\leftarrow\varnothing$
$\left(m^*,r^*,m^{*'},r^{*'},CHF^*\right)\leftarrow\mathrm{A}^{\mathrm{O}}\left(Mpk_{EPBCHF}\right),\textbf{where}\ \mathrm{O}\leftarrow\left\{\mathrm{O}_{KGen},\mathrm{O}_{KGen'},\mathrm{O}_{SKGen},\mathrm{O}_{\text{Hash}},\mathrm{O}_{\text{Hcol}}\right\}.$
$\textbf{if}\ 1=\mathbf{EPBCHF.Verify}\left(Mpk_{CH},Spk_{CH},m^*,CHF^*,r^*\right)=\mathbf{EPBCHF.Verify}\left(Mpk_{CH},Spk_{CH},m^{*'},CHF^*,r^{*'}\right)$
$\textbf{and}\left(CHF^*,\mathrm{A}\right)\in\Lambda_3,\ \textbf{for some}\ \mathrm{A}\ \textbf{and}\ \mathrm{m}^*\neq m^{*'}\ \textbf{and}\ (\mathrm{A}\cap\Lambda_1=\varnothing)\textbf{and}\left(CHF^*,m^*,.\right)\notin\Lambda_3,\ \underline{return}\ 1,\textbf{otherwise}\ \underline{return}\ 0$
$\textbf{Oracle}\ \mathrm{O}_{KGen}\left(Msk_{EPBCHF},S_i\right):sk_{ii}\leftarrow\mathbf{EPBCHF.KGen}\left(Msk_{EPBCHF},S_i\right);\Lambda_1\leftarrow\Lambda_1\cup\left\{S_i\right\},\ \underline{return}\ sk_{ii}$
$\textbf{Oracle}\ \mathrm{O}_{KGen'}\left(Msk_{EPBCHF},S_i\right):sk_{ii}\leftarrow\mathbf{EPBCHF.KGen}\left(Msk_{EPBCHF},S_i\right);\Lambda_2\cup\left\{S_i\right\},i\leftarrow i+1$
$\textbf{Oracle}\ \mathrm{O}_{SKGen}\ \left(pp_{CH},\mathrm{A}\right):\left(Spk_{CH},CT,MAC_{tag}\right)\leftarrow EPBCHF.SKGen\left(pp_{CH},\mathrm{A}\right),\Lambda_3\leftarrow\Lambda_3\cup\{\mathrm{A}\}$
$\textbf{Oracle}\ \mathrm{O}_{\text{Hash}}\ \left(Mpk_{CH},Spk_{CH},m\right):\left(CHF,r,CT,MAC_{tag}\right)\leftarrow\mathbf{EPBCHF.Hash}\left(Mpk_{CH},Spk_{CH},m\right)$
$\Lambda_3\leftarrow\Lambda_3\cup\left\{(CH,m)\right\},return\left(CHF,r,CT,MAC_{tag}\right)$
$\textbf{Oracle}\ \mathrm{O}_{\text{HCOL}}\ \left(m,m^{'},CHF,r,CT,MAC_{tag},sk_{ii}\right):\textbf{if}\ \left(sk_{ii}\right)\notin\Lambda_2\ \textbf{for some}\ Msk_{EPBCHF},\ \underline{return}\perp.$
$r^{'}\leftarrow\mathbf{EPBCHF.HCOL}\left(pk,sk_{ii},m,m^{'},CHF,r,CT,MAC_{tag}\right)\textbf{if}\ (CHF,m,\mathrm{A})\in\Lambda_3,$
$\text{for some}\ A,\textbf{let}\ \Lambda_3\leftarrow\Lambda_3\cup\left\{\left(CHF,m^{'},\mathrm{A}\right)\right\},\underline{return}\ r^{'}.$

The EPBCHF scheme is collision-resistant if the advantage for any polynomial-time adversary $\mathcal{A}$ is negligible, as defined below:

$$\mathrm{Adv}^{CR}_{EPBCHF,\mathrm{A}}\left(1^{\lambda}\right)=\Pr\left[\mathrm{Exp}^{CR}_{EPBCHF,\mathrm{A}}\left(1^{\lambda}\right)=1\right]$$

5. CONCRETE CONSTRUCTION INSTANTIATION

This section introduces the specific concrete construction and compares the results.

5.1 CONCRETE CONSTRUCTION

This section introduces a practical instantiation of EPBCHF tailored specifically for redactable blockchains. Our approach builds upon CHDLTTF, instantiated with a Discrete Logarithm (DL)--based solution and an efficient CP-ABE algorithm type known as FAME [23]. FAME incorporates asymmetric prime-order Type-III pairing and achieves adaptive security, relying on the standard decision linear assumption. It also boasts essential features such as accommodating unlimited ABE universes and ensuring consistent decryption speed. To further enhance FAME's capabilities and align it with the CCA security model, we apply the BK transformation developed by [7], [26]. In adapting this scheme to meet our specific requirements, we incorporate the BK scheme and efficient MAC, which seamlessly align with our objectives. For message authentication in our case, we employ a CBC-MAC with a 128-bit AES as the underlying block cipher to provide CCA security to FAME. The subsequent algorithm sections elaborate on the instantiation of the EPBCHF scheme construction.

Definition 5: EPBCHF Concrete Construction

Setup Algorithm EPBCHF.Setup $(1^{\lambda}, n)$

Inputs: Security parameter λ , Number of users n

⇒ **Group Generation**: Execute GroupGen (1^{λ}) to generate $\mathcal{G}$ defined as $\mathcal{G} = (p, \mathrm{G}, \mathrm{H}, \mathrm{G_T}, \hat{e}, g, h)$. Consider a bilinear pairing: $\hat{e} = \mathrm{G} \times \mathrm{H} \to \mathrm{G_T}$, where g and h are the generator G,and H respectively.

⇒ **Master Secret/Public Key Pair Generation for CHDLTTF**: Execute $(Mpk_{CH}, Msk_{CH}) \leftarrow CHDLTTF.KGen(PP_{CH})$; to generate the master secret/public key pair $(Msk_{CH} = S, Mpk_{CH} = h^S)$ where $S \in \mathbb{Z}_p$ and $PP_{CH} \leftarrow CHDLTTF.PPGen(1^{\lambda})$;

⇒ **Hash Functions Setup: Pick** $H : \{0,1\}^* \to \mathrm{G}, H_1 : \{0,1\}^* \to \mathbb{Z}_p$

⇒ **CP-FAME Key Generation: Pick** $(a_1, a_2, f_1, f_2) \leftarrow \mathbb{Z}_p$, $(e_1, e_2, e_3) \leftarrow \mathbb{Z}_p$, **Compute:** $T_1 = h^{a_1}, T_2 = h^{a_2}$, $\mathrm{K_1} := \hat{e}\{g, h\}^{e_1 a_1 + e_3}$, $\mathrm{K_2} := \hat{e}\{g, h\}^{e_2 a_2 + e_3}$, $mpk_{ABE} := \{g, h, T_1, T_2, K_1, K_2, H, H_1\}$, $msk_{ABE} := \{a_1, a_2, f_1, f_2, g^{e_1}, g^{e_2}, g^{e_3}\}$

Output: EPBCH Composite Key Generation, including $Mpk_{EPBCHF} := \{Mpk_{CH}, mpk_{ABE}\} = \{h^S, g, h, T_1, T_2, K_1, K_2, H, H_1\}$, and $Msk_{EPBCHF} := \{Msk_{CH}, msk_{ABE}\} = \{S, a_1, a_2, f_1, f_2, g^{e_1}, g^{e_2}, g^{e_3}\}$

EPBCHF.KGen (Msk_{EPBCHF}, S_i):

Inputs: (Msk_{EPBCHF}) and attribute set (S_i) of participant i

⇒ Pick $(\delta_1, \delta_2) \in \mathbb{Z}_p, \delta = \delta_1 + \delta_2$, compute $\mathrm{sk_0} := \left(h^{f_1 \delta_1}, h^{f_2 \delta_2}, h^{\delta}\right)$,

⇒ for all $s \in S_{i,j}$ and $z = \{1,2\}$, pick $\varpi_s, \varpi' \in \mathbb{Z}_p$, compute $\mathrm{sk}_{s,z} := H(s1z)^{f_1 \delta_1 / a_z} \cdot H(s2z)^{f_2 \delta_2 / az} \cdot H(s3z)^{\delta / az} \cdot g^{\varpi_s / a_z}$, where Set $\mathrm{sk}_s := \left\{\mathrm{sk}_{s,1}, \mathrm{sk}_{s,2}, g^{-\varpi_s}\right\}$,

⇒ Comput $sk'_z := g^{e_z} \cdot H(011z)^{f_1 \delta_1 / a_z} \cdot H(012z)^{f_2 \delta_2 / a_z} \cdot H(013z)^{\delta / a_z} \cdot g^{\varpi' / a_z}$ set $\mathrm{sk}' := \left(\mathrm{sk}'_1, \mathrm{sk}'_2, g^{e_3} \cdot g^{-\varpi'}\right)$. The decryption key for S_i is: $sk_i := \left(\mathrm{sk}_0, \{\mathrm{sk}_s\}_{s \in S_i}, sk'\right)$

Output: $sk_{ii} := \{S, sk_i\}$

EPBCHF.SKGen (PP_{CH}, A)

Inputs: Public parameter (PP_{CH}) , access policy A

⇒ **Compute CHDLTTF Second Key:** Run $(Spk_{CH}, Ssk_{CH}) \leftarrow CHDLTTF.SKGen\ (pp_{CH})$; to acquire the second trapdoor/hash keys pair: $Ssk_{CH} = T, Spk_{CH} = h^{H_1(T)}$, where $T \in \mathbb{Z}_p$.

⇒ **Encryption Process:** Select random values $\tau_1, \tau_2 \leftarrow \mathbb{Z}_p$ and calculate $\tau = \tau_1 + \tau_2$: Compute $ct_0 := \left(T_1^{\tau_1}, T_2^{\tau_2}, h^{\tau}\right)$. Suppose A has ro rows and co columns. For $\forall i \in \{1, \ldots, ro\}, t = 1, 2, 3$

$$\mathrm{ct}_{i,t} := H(\pi(i)t1)^{\tau_1} \cdot H(\pi(i)t2)^{\tau_2} \cdot \prod_{j=1}^{co} \left[H(0\,jt1)^{\tau_1} \cdot H(0\,jt2)^{\tau 2}\right]^{\mathrm{A}_{i,j}}$$

$\mathrm{ct}_i := \left(\mathrm{ct}_{i,1}, \mathrm{ct}_{i,2}, \mathrm{ct}_{i,3}\right)$.Compute, $CT_1 := K_1^{\tau_1} \cdot K_2^{\tau_2} \cdot m$; where m is message = (Ssk_{Ch}). Finally, outputs $CT_1 = \left(\mathrm{ct}_0, \{\mathrm{ct}_i\}_{i \in co}, ct'\right)$.

⇒ **Encapsulation Scheme (KEM):** Select randomness $r \in \mathbb{Z}_p$, compute $(Key = H(r), hid = H_1(r), bid = r)$, where hid is used as a hiding factor, bid as a biding factor. Choose r' , by applying $r' = H\ ((Ssk_{CH}))$, compute $CT_2 = G(r') \oplus (msg \circ bid)$, where G is a PRG and $msg = Ssk_{CH}$.

⇒ **Calculate** MAC_{tag} : Compute MAC_{tag} on (CT_1, CT_2) using Key .

Output: $\left(Spk_{CH}, (\mathrm{CT\ as}\ (CT_1, CT_2, MAC_{tag}, hid)\right)$

Hash Algorithm $(Mpk_{CH}, Spk_{CH}, m_i)$

Input: Master hash key Mpk_{CH} , Second hash key Spk_{CH} , set of messages (transactions) as m_i , access policy A , following performed :

⇒ ***IF*** (reusable trapdoor key == true), **then call** (EPBCHF.SKGen (PP_{CH}, A) one time calling for all transactions), ***otherwise*** (call EPBCHF.SKGen (PP_{CH}, A) multiple times for each transaction), which behaves similarly to ephemeral trapdoor key.

⇒ **Chameleon Hash Computation CHDLTTF:** For each message or transaction m_i in the set Select a random $r_i \in \mathbb{Z}_p$ *and c*alculate the chameleon hash CHF_i for each m_i as, $CHF_i = Mpk_{CH}^{r_i} \square Spk_{CH}^{m_i}$. By repeating these steps for each m_i in the set of messages or transactions, you can calculate CHF_i r_i for all of them.

Output: $(CHF_i, r_i, Spk_{CH}, (\mathrm{CT\ as}\ (CT_1, CT_2, MAC_{tag}, hid)))$

EPBCHF.Verify

Inputs: $(Mpk_{CH}, Spk_{CH}, CHF_i, r_i, m_i)$.

Check $CHF_i = Mpk_{CH}^{r_i} \square Spk_{CH}^{m_i}$.

Output: return 1 if successful; otherwise, 0.

EPBCHF.Adapt
Redaction by TM

Inputs: It takes the (sk_i) secret key of the modifier, the public key as (Mpk_{CH}, Spk_{CH}) a new message r_i' , message m_i , ciphertext $CT\ as\ (CT_1, CT_2, MAC_{tag}, hid)$, randomness r_i chameleon hash CHF_i as input:

⇒ Check ***if*** $1 = EPBCHF.Verify\ (Mpk_{CH}, Spk_{CH}, r_i, CHF_i, m_i)$.

⇒ Decrypt CT_1 based on (ABE - FAME) as follows:

Run ABE. Decrypt take $(mpk_{ABE}, sk_i, MAC_{tag}$, and CT as $(\mathrm{CT}_1, CT_2), hid)$ **Outputs** m or $\perp$. If S in (sk_i) satisfies A in CT_1 Additionally, there are constants $\{\gamma_i\}_{i\in I}$ that satisfies $\sum_{i\in I}\gamma_i M_i = (1, 0\ldots\ldots, 0)$, Compute:

$$n := \mathrm{ct}'.\hat{e}(\prod_{i\in I} ct_{i,1}^{\gamma_i}, sk_{0,1}).\hat{e}(\prod_{i\in I} ct_{i,2}^{\gamma_i}, sk_{0,2}).\hat{e}(\prod_{i\in I} ct_{i,3}^{\gamma_i}, sk_{0,3})$$

$$m := \hat{e}(\mathrm{sk}_1'.\prod_{i\in I} \mathrm{sk}_{\pi(i),1}^{\gamma_i}, \mathrm{ct}_{0,1}).\hat{e}(\mathrm{sk}_2'.\prod_{i\in I} \mathrm{sk}_{\pi(i),2}^{\gamma_i}, \mathrm{ct}_{0,2}).\hat{e}(\mathrm{sk}_3'.\prod_{i\in I} \mathrm{sk}_{\pi(i),3}^{\gamma_i}, \mathrm{ct}_{0,3})$$

where $\mathrm{sk}_{0,1}, \mathrm{sk}_{0,2}, \mathrm{sk}_{0,3}$ denote the first, second, and third element of sk_0 the same rule applies to ct_0 , and n, m are the output, then derive the second trapdoor $Ssk_{CH} = \{msg\}$ from n , m .

⇒ Verify MAC_{tag} : Calculate $r' = H(Ssk_{CH})$ where (r' is computed based on Ssk_{CH}); this value is used to decrypt CT_2 , obtain bid based on r' , $CT_2 = (mobid)$ and compute $hid = H(bid)$. The output is m if $hid = H(bid)$, and the verification of the MAC_{tag} on (CT_1, CT_2) under Key is valid.

⇒ Compute collision r_i' : $r_i' := r_i + \left(m_i - m_i'\right) \cdot H_2^{(Ssk_{CH})} / sk_{CH}$.

Output: $(m_i', CHF_i, r_i', MAC_{tag}, CT, hid)$

6. RESULTS AND COMPARISONS

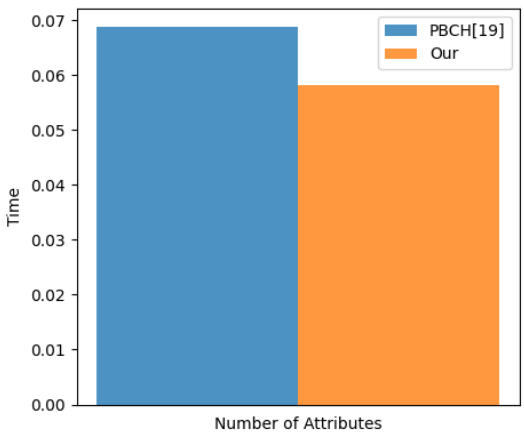
In this section, we evaluate EPBCHF compared to the conventional PBCHF proposed by Derler [19], focusing on measuring efficiency in terms of algorithm execution time. Additionally, we provide a summary of key distinctions between the two systems as follows:

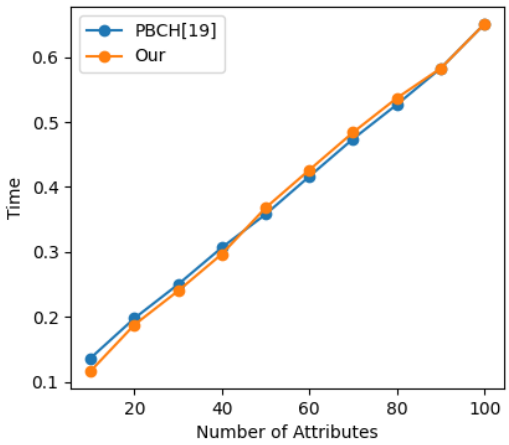
- In the PBCHF system, the Transaction Owner (TO) employs the CHET mechanism to calculate the hash for each transaction. This process involves generating and encrypting a trapdoor key within the hashing algorithm. In contrast, the proposed EPBCHF introduces a novel algorithm named the Second Trapdoor Generation Algorithm (SKGen). The SKGen algorithm operates independently of the hashing mechanism and is executed on the TO's side. Consequently, in our design, the hashing algorithm is exclusively dedicated to computing the transaction hash, relying on the second trapdoor key. By introducing the SKGen algorithm, we emphasize its potential to significantly reduce execution time and enable the reuse of the second trapdoor key when multiple transactions share similar access policies or possess the same modifier. As a result, our hash execution time is notably improved, achieving a time of 1.95 ms for attributes within the range of 10-100. This represents a significant enhancement compared to PBCHF, where hash computation consumes 539.6 ms for attributes within the same range. However, it is essential to acknowledge that including SKGen introduces an additional computational overhead, resulting in an execution time of 480 ms for attributes spanning from 10 to 100. In our evaluation, we find this increase in execution time to be acceptable, particularly when considering the trapdoor key's reusability in scenarios where transactions share identical access policies or modifiers. This reusability implies that the key is computed only once, effectively reducing the overall computational overhead. We are confident that this moderate increase in overhead does not significantly impact system efficiency, as the TO retains control and maintains a balance within the system.
- Computation Time Reduction on the Transaction Manager (TM) Side: Our construction reduces the time burden on the TM side by replacing the recalculation of ciphertext trapdoor key validation with the use of the MAC technique, which is faster and more efficient. Simultaneously, if the TM receives several transactions from the same TO, it only needs to decrypt the ciphertext once, increasing time efficiency.

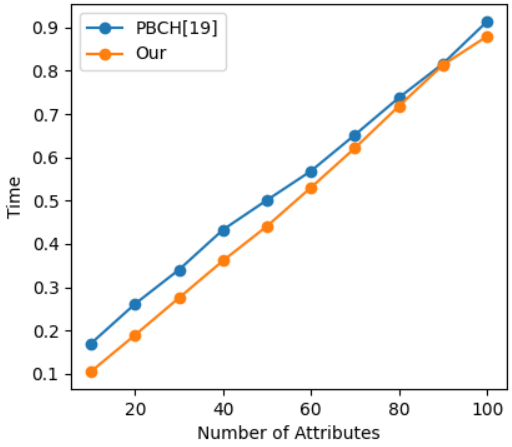
We executed the implementation of our EPBCHF scheme and conducted an exhaustive performance evaluation on a PC equipped with a 64-bit Ubuntu operating system (version 22.04), an Intel Core i5 processor, and 16 GB of RAM.

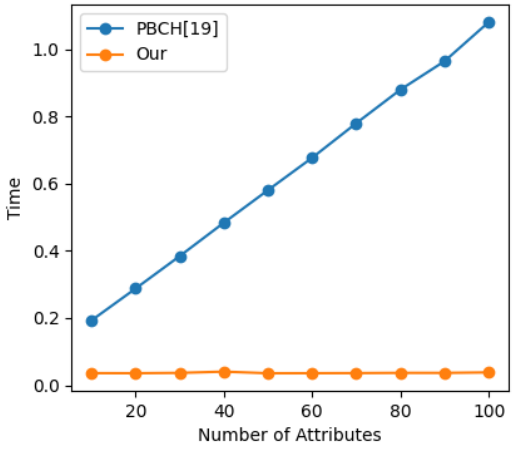
Throughout our benchmarking process, we diligently scrutinized all aspects of the EPBCHF algorithm, introduced necessary parameter variations, and harnessed pairing groups based on the MNT224 curve. Our construction was realized using the standardized interfaces provided by the Charm framework (version 0.491). The implementation ran seamlessly on Python version 3.7, hosted within an Anaconda environment (version 2.4.0). We conduct two experiments types:

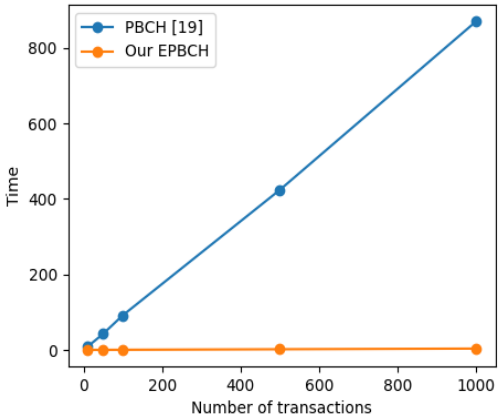
1. The first experiment focused on making transactions constant and set them to be (One Transaction) while attributes are variable and range from 10 to 100. algorithms have been tested wherein values acquired are for Setup algorithm achieved (0.05 s) due to the use of CHDLTTF-based DL instead of CHET-based RSA in PBCHF, KeyGen and Verify algorithms also resulted in an identical to PBCHF of 0.033 and 0.00217s respectively. The TO has executed SKGEN and Hash algorithms with an average of 0.442s for a single transaction. Finally, the collision algorithm reached 0.038, which was applied to the modifier side.
2. The second experiment considers the SKGen algorithm that offers great addition via generating one trapdoor key for each hash computation, which can be employed in numerous transaction hashing processes. Consequently, unlike PBCHF [19], several benefits can be achieved via reusing similar trapdoor keys with multiple transaction hashing operations, resulting in preserved computational time and communication overhead, further increasing proposed construction flexibility due to their independence from any other hashes generated. Similar conditions have been selected to test 1000 transactions on both EPBCHF and PBCH, where the results acquired indicate an average time of 3.4s in EPBCHF versus PBCHF, which was established to 870s. It is noted that a linear time increase in EPBCHF is related to the number of transactions, indicating the system is scaling linearly rather than exponentially, wherein 1000 transactions are consuming 100 times 10 transactions; therefore, it is considered a linear relationship. Unlike PBCHF, where the relation is not linear due to the time taken to process a similar number of transactions is not 10 times; thereby, it is a sublinear scaling relation due to different time intervals have been measured, first 10-50 transactions then 100 transactions which almost indicate scalability is undermined within the transactions increment. Furthermore, the data size measured was 144 KB for the 1000 transactions, showing a linear increment through noticed doubling data size due to transactions number doubling. CPU employment again indicates a linear relation with transaction volume; however, EPBCHF is further enhanced and nearly constant with bigger volumes due to the noted results, which show that 10 transactions consume 83.3% versus 100% for 50 transactions and above. Meanwhile, PBCH results evidently demonstrate a fixed 100% CPU usage for all transaction volumes, which refers to a passive relationship between them, and thus, the performance bottleneck is obvious, leading to being unscaled and inefficient, specifically with large transaction volumes nutshell, PBCH might be ideal for small transaction loads that might not require full CPU capacity. The aforementioned observations prove that EPBCHF is scalable and efficient due to having the ability to handle unlimited transaction volumes additionally, data is efficiently processed without further overhead meanwhile, data size consistency reflects robust memory management due to predicted memory size, network bandwidth, and storage requirement for future growth thereby, EPBCHF is proven to be reliable and scalable. Figures 3 to 9 visually represent the testing results and discussed observations.

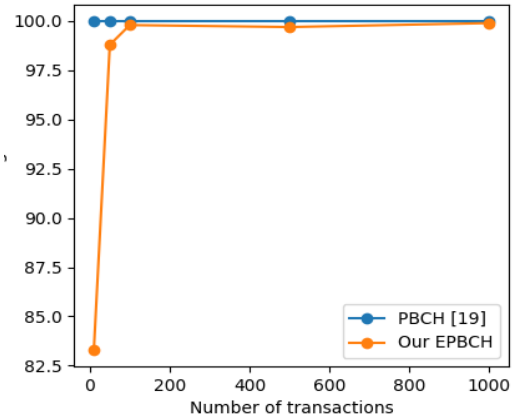


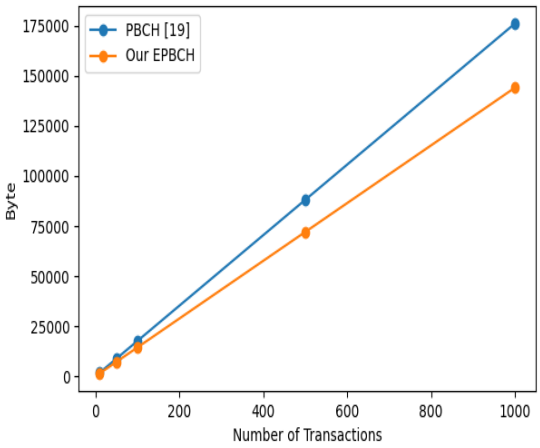



PBCH[19]
Our
Time
Number of Attributes


PBCH[19]
Our
Time
Number of Attributes


PBCH [19]
Our EPBCH
Time
Number of transactions


PBCH [19]
Our EPBCH
Number of transactions


PBCH [19]
Our EPBCH
Byte
Number of Transactions

7. APPLICATION OF EPBCHF

We demonstrate the effective application of our EPBCHF construction within existing blockchain systems, allowing for transaction modifications while maintaining blockchain integrity. Every block in the blockchain contains a summarized illustration of a collection of transactions, achieved through the Merkle tree root hash (TX-ROOT), consolidating entire transactions within the block [49]. Since a transaction is added by a specific user, like $T_x(i,1)$, to the blockchain, it is compulsory for a message or recorded data to undergo hashing leveraging the Hash (Mpk, m) function. As depicted in Figure 4, a block B_i that combines the following transactions, $T_x(i,1)$, $T_x(i,2)$, $T_x(i,3)$, and $T_x(i,4)$, where $T_x(i,1)$ is a mutable transaction that includes an access policy denoted as A, selected by the user. The rest of the transactions are hashed utilizing any one-way hash function referred to as H. In cases where $T_x(i,1)$ requires modification, possessing a secret key sk(A) satisfying A, a modifier is enabled to find a collision r' for CHF value, and r is replaced with r'. It's important to realize that the r, CT, and MAC_{tag} are not integrated into the hash calculation; Instead, these elements are supplied as non-hashed components within the transaction or block. Moreover, the blockchain's linking hash function remains unchanged as 'H,' and the previous hash values retain their original state. The implications of incorporating the EPBCHF construction into a mutable blockchain can be briefed as follows:

- Minimal Overhead: Modifying rewritten transactions introduces minimal overhead in terms of Merkle tree generation and chain validation. This efficiency results from storing r and CT in the non-hashed segment of rewritten transactions. One-way hash functions like SHA256 are employed for aggregating transactions and connecting blocks.
- Feasibility in Permissioned Blockchains: The EPBCHF Construction is applicable in permissioned blockchains like Hyperledger and Ripple, as confirmed by prior research [19]. The assumption of transaction authority (TA) is reasonable since the authority governs attribute-based users who can contribute to consensus on the system state, validate block transactions, and restrict access to authorized users capable of modifying transactions.
- Manageable Storage Costs: Storage costs are considered reasonable for blockchain applications, especially when the number of mutable transactions within a block is assumed to be limited.

In conclusion, integrating our EPBCHF construction into a mutable blockchain offers several advantages, including minimal overhead, suitability for permissioned blockchains, and manageable storage costs. This makes it a practical solution for transaction modifications within a blockchain.

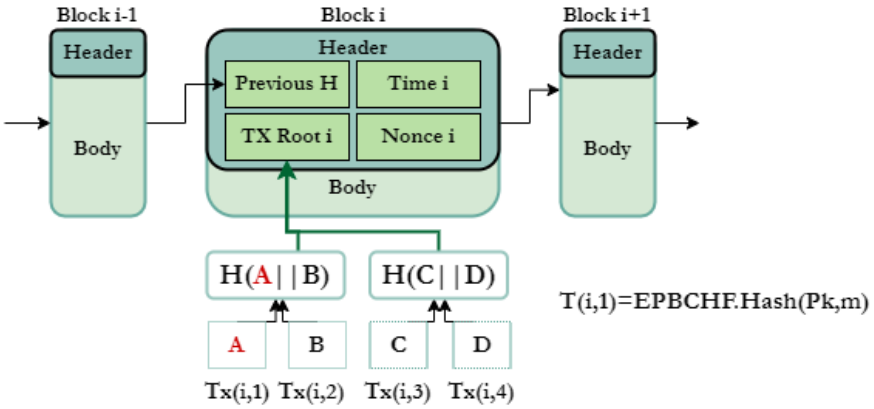


Figure 4. EPBCHF Application.

8. CONCLUSION

In conclusion, this paper has presented the design of a redactable blockchain with fine-vgrained access control, drawing inspiration from PBCH, which revealed efficiency challenges. To address these challenges, we introduced the novel EPBCHF construction by replacing the traditional CHET with the CHDLLT primitive. This modification allows transaction owners to compute a single trapdoor key when multiple modifiable transactions share the same access policy or are modified by the same entity. This enhancement significantly reduces computation time compared to regular PBCH, which computes fresh trapdoor keys for each transaction. The second improved aspect states, in essence, that the modifier is required to decrypt and validate each trapdoor key in PBCH, where it added massive transaction computation burdensome while our construction has replaced the validation of the trapdoor key by MAC, which has been proven efficient and fast. Future research directions should explore the extension of EPBCHF with an efficient revocation mechanism to prevent malicious activity by modifiers when modifying multiple transactions with a single trapdoor key.

This research marks a significant stride toward enhancing the efficiency of the redaction process in redactable blockchain systems, making them more suitable for real-world applications.

Funding

None

ACKNOWLEDGEMENT

None

CONFLICTS OF INTEREST

None

REFERENCES

[1] N. Sabah, A. Sagheer, and O. Dawood, “Survey: (Blockchain-Based Solution for COVID-19 and Smart Contract Healthcare Certification),” ijcsm, pp. 1–8, Jan. 2021, doi: 10.52866/ijcsm.2021.02.01.001.

[2] Z. Ma, M. Jiang, H. Gao, and Z. Wang, "Blockchain for digital rights management," Future Generation Computer Systems, vol. 89, pp. 746–764, Dec. 2018, doi: 10.1016/j.future.2018.07.029.

[3] N. Z. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams," IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 5, pp. 840–852, Sep. 2018, doi: 10.1109/TDSC.2016.2616861.

[4] K. M. Khan, J. Arshad, and M. M. Khan, "Investigating performance constraints for blockchain-based secure e-voting system," Future Generation Computer Systems, vol. 105, pp. 13–26, Apr. 2020, doi: 10.1016/j.future.2019.11.005.

[5] N. Khan, H. Aljoaey, M. Tabassum, A. Farzamnia, T. Sharma, and Y. H. Tung, "Proposed Model for Secured Data Storage in Decentralized Cloud by Blockchain Ethereum," Electronics, vol. 11, no. 22, Art. no. 22, Jan. 2022, doi: 10.3390/electronics11223686.

[6] B. Wang and Z. Li, "Healthchain: A Privacy Protection System for Medical Data Based on Blockchain," Future Internet, vol. 13, no. 10, Art. no. 10, Oct. 2021, doi: 10.3390/fi13100247.

[7] N. C. K. Yiu, "An Overview of Forks and Coordination in Blockchain Development," Feb. 2021, doi: 10.13140/RG.2.2.36579.07207.

[8] I. Al_Barazanchi et al., "Blockchain Technology Based Solutions for IOT Security," Iraqi Journal For Computer Science and Mathematics, vol. 3, no. 1, Art. no. 1, Jan. 2022, doi: 10.52866/ijcsm.2022.01.01.006.

[9] X. Xu, D. Zhu, X. Yang, S. Wang, L. Qi, and W. Dou, "Concurrent Practical Byzantine Fault Tolerance for Integration of Blockchain and Supply Chain," ACM Transactions on Internet Technology, vol. 21, no. 1, 2021, doi: 10.1145/3395331.

[10] G. Tziakouris, "Cryptocurrencies - A forensic challenge or opportunity for law enforcement? An INTERPOL perspective," IEEE Security and Privacy, vol. 16, no. 4, 2018, doi: 10.1109/MSP.2018.3111243.

[11] R. Qamar and B. A. Zardari, "A Study of Blockchain-Based Internet of Things," Iraqi Journal For Computer Science and Mathematics, vol. 4, no. 1, Art. no. 1, 2023, doi: 10.52866/ijcsm.2023.01.01.003.

[12] R. Matzutt et al., "A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018. doi: 10.1007/978-3-662-58387-6_23.

[13] P. Bai, S. Kumar, K. Kumar, O. Kaiwartya, M. Mahmud, and J. Lloret, "GDPR Compliant Data Storage and Sharing in Smart Healthcare System: A Blockchain-Based Solution," Electronics, vol. 11, no. 20, 2022, doi: 10.3390/electronics11203311.

[14] B. Schellinger, F. Völter, N. Urbach, and J. Sedlmeir, "Yes, I Do: Marrying Blockchain Applications with GDPR," in Proceedings of the 55th Hawaii International Conference on System Sciences, 2022. doi: 10.24251/hicss.2022.563.

[15] S. A. Salman, S. Al-Janabi, and A. M. Sagheer, "Security Attacks on E-Voting System Using Blockchain," Iraqi Journal for Computer Science and Mathematics, vol. 4, no. 2, 2023, Accessed: Sep. 13, 2023. [Online]. Available: https://www.iasj.net/iasj/article/270134

[16] E. Politou, F. Casino, E. Alepis, and C. Patsakis, "Blockchain Mutability: Challenges and Proposed Solutions," IEEE Trans. Emerg. Topics Comput., vol. 9, no. 4, pp. 1972–1986, Oct. 2021, doi: 10.1109/TETC.2019.2949510.

[17] G. Ateniese, B. Magri, D. Venturi, and E. R. Andrade, "Redactable Blockchain - Or - Rewriting History in Bitcoin and Friends," in Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017, 2017. doi: 10.1109/EuroSP.2017.37.

[18] H. Krawczyk and T. Rabin, "Chameleon Hashing and Signatures.," IACR Cryptology ePrint Archive, vol. 1998, no. October, 1998.

[19] D. Derler, K. Samelin, D. Slamanig, and C. Striecks, "Fine-Grained and Controlled Rewriting in Blockchains: Chameleon-Hashing Gone Attribute-Based," 2019. doi: 10.14722/ndss.2019.23066.
[20] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors and applications to invisible sanitizable signatures," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2017. doi: 10.1007/978-3-662-54388-7_6.
[21] S. Agrawal and M. Chase, "FAME: Fast Attribute-based Message Encryption." 2017. Accessed: Sep. 13, 2023. [Online]. Available: https://eprint.iacr.org/2017/807
[22] S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-Hashes with Dual Long-Term Trapdoors and Their Applications." 2018. Accessed: Sep. 13, 2023. [Online]. Available: https://eprint.iacr.org/2018/971
[23] L. Guo, Q. Wang, and W. C. Yau, "Online/Offline Rewritable Blockchain with Auditable Outsourced Computation," IEEE Transactions on Cloud Computing, 2021, doi: 10.1109/TCC.2021.3102031.
[24] S. Xu, J. Ning, J. Ma, G. Xu, J. Yuan, and R. H. Deng, "Revocable Policy-Based Chameleon Hash," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021. doi: 10.1007/978-3-030-88418-5_16.
[25] G. Panwar, R. Vishwanathan, and S. Misra, "ReTRACe: Revocable and traceable blockchain rewrites using attribute-based cryptosystems," in Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT, 2021. doi: 10.1145/3450569.3463565.
[26] Y. Tian, A. Miyaji, K. Matsubara, H. Cui, and N. Li, "Revocable Policy-Based Chameleon Hash for Blockchain Rewriting," The Computer Journal, p. bxac083, Jul. 2022, doi: 10.1093/comjnl/bxac083.
[27] D. Boneh and J. Katz, "Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption," in Topics in Cryptology – CT-RSA 2005, A. Menezes, Ed., in Lecture Notes in Computer Science, vol. 3376. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 87–103. doi: 10.1007/978-3-540-30574-3_8.
[28] S. M. Abd Ali, M. N. Yusoff, and H. F. Hasan, "Redactable Blockchain: Comprehensive Review, Mechanisms, Challenges, Open Issues and Future Research Directions," Future Internet, vol. 15, no. 1, 2023, doi: 10.3390/fi15010035.
[29] K. Huang et al., "Building redactable consortium blockchain for industrial internet-of-things," IEEE Transactions on Industrial Informatics, vol. 15, no. 6, 2019, doi: 10.1109/TII.2019.2901011.
[30] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, X. Du, and N. Guizani, "Achieving Intelligent Trust-Layer for Internet-of-Things via Self-Redactable Blockchain," IEEE Transactions on Industrial Informatics, vol. 16, no. 4, 2020, doi: 10.1109/TII.2019.2943331.
[31] J. Zhang et al., "Serving at the edge: A redactable blockchain with fixed storage," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020. doi: 10.1007/978-3-030-60029-7_58.
[32] W. Lv, S. Wei, S. Li, and M. Yu, "Verifiable Blockchain Redacting Method for a Trusted Consortium with Distributed Chameleon Hash Authority," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2020. doi: 10.1007/978-3-030-66046-8_24.
[33] L. Liu, L. Tan, J. Liu, J. Xiao, H. Yin, and S. Tan, "Redactable Blockchain Technology Based on Distributed Key Management and Trusted Execution Environment," in Communications in Computer and Information Science, 2021. doi: 10.1007/978-981-16-7993-3_23.
[34] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, and X. Du, "Scalable and redactable blockchain with update and anonymity," Information Sciences, vol. 546, 2021, doi: 10.1016/j.ins.2020.07.016.
[35] C. Zhang, Z. Ni, Y. Xu, E. Luo, L. Chen, and Y. Zhang, "A trustworthy industrial data management scheme based on redactable blockchain," Journal of Parallel and Distributed Computing, vol. 152, 2021, doi: 10.1016/j.jpdc.2021.02.026.
[36] C. Wu, L. Ke, and Y. Du, "Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain," Information Sciences, vol. 548, 2021, doi: 10.1016/j.ins.2020.10.008.
[37] R. Matzutt, V. Ahlrichs, J. Pennekamp, R. Karwacik, and K. Wehrle, "A Moderation Framework for the Swift and Transparent Removal of Illicit Blockchain Content," in IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2022, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICBC54727.2022.9805508.
[38] W. Gao, L. Chen, C. Rong, K. Liang, X. Zheng, and J. Yu, "Security Analysis and Improvement of a Redactable Consortium Blockchain for Industrial Internet-of-Things," The Computer Journal, vol. 65, no. 9, 2022, doi: 10.1093/comjnl/bxab080.
[39] J. Wei et al., "A Redactable Blockchain Framework for Secure Federated Learning in Industrial Internet of Things," IEEE Internet of Things Journal, vol. 9, no. 18, 2022, doi: 10.1109/JIOT.2022.3162499.
[40] Y. Tian, N. Li, Y. Li, P. Szalachowski, and J. Zhou, "Policy-based Chameleon Hash for Blockchain Rewriting with Black-box Accountability," in ACM International Conference Proceeding Series, 2020. doi: 10.1145/3427228.3427247.

[41] H. Hou, S. Hao, J. Yuan, S. Xu, and Y. Zhao, "Fine-Grained and Controllably Redactable Blockchain with Harmful Data Forced Removal," Security and Communication Networks, vol. 2021, 2021, doi: 10.1155/2021/3680359.
[42] S. Xu, J. Ning, J. Ma, X. Huang, and R. H. Deng, "K-Time Modifiable and Epoch-Based Redactable Blockchain," IEEE Transactions on Information Forensics and Security, vol. 16, 2021, doi: 10.1109/TIFS.2021.3107146.
[43] X. Chen and Y. Gao, "CDEdit: A Highly Applicable Redactable Blockchain with Controllable Editing Privilege and Diversified Editing Types," May 2022, [Online]. Available: http://arxiv.org/abs/2205.07054
[44] Z. Zhang, T. Li, Z. Wang, and J. Liu, "Redactable Transactions in Consortium Blockchain: Controlled by Multi-authority CP-ABE," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021. doi: 10.1007/978-3-030-90567-5_21.
[45] J. Ma, S. Xu, J. Ning, X. Huang, and R. H. Deng, "Redactable Blockchain in Decentralized Setting," IEEE Transactions on Information Forensics and Security, vol. 17, 2022, doi: 10.1109/TIFS.2022.3156808.
[46] Y. Jia, S. F. Sun, Y. Zhang, Z. Liu, and D. Gu, "Redactable Blockchain Supporting Supervision and Self-Management," in ASIA CCS 2021 - Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security, 2021. doi: 10.1145/3433210.3453091.
[47] J. Brown, J. M. G. Nieto, and C. Boyd, "Efficient CCA-Secure Public-Key Encryption Schemes from RSA-Related Assumptions," in Progress in Cryptology - INDOCRYPT 2006, R. Barua and T. Lange, Eds., in Lecture Notes in Computer Science, vol. 4329. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 176–190. doi: 10.1007/11941378_13.
[48] S. Xu, G. Yang, and Y. Mu, "Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation," Information Sciences, vol. 479, pp. 116–134, 2019, doi: https://doi.org/10.1016/j.ins.2018.11.031.

[49] H. F. Hasan, M. N. B. Yusoff, and S. M. A. Ali, "Bitcoin Layer Two Scaling Solutions: Lightening Payment Channels Network Comprehensive Review, Mechanisms, Challenges, Open Issues and Future Research Directions," Iraqi Journal For Computer Science and Mathematics, vol. 5, no. 1, Art. no. 1, Jan. 2024, doi: 10.52866/ijcsm.2024.05.01.003.