Traitor tracing scheme for relational databases based on blockchain

Weiqiang Jiang[a], Kedi Yang[b,c], Youliang Tian*[b,c]
[a]Information Security Center, China Mobile Communications Corporation, Beijing 10053, China; [b] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China; [c]College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

ABSTRACT

The traceability of data leakage remains a foundational challenge faced by big data. Traditional data tracing technology is mainly based on digital fingerprint to embed lengthy code into digital works such as video, while the structured data with limited embedding space has not been given adequate consideration. In this paper, we propose a chameleon short signature by improving Khalili's chameleon hash function and combining Boneh's signature algorithm to achieve a one-to-many signature with a shorter message length under the same security premise. Then, we construct a traitor tracing model based on the proposed signature and design a cascade chain to complete credible data sharing and undeniable leaker detection. Security and simulation analysis show that the traitor tracing model achieves trusted data sharing and efficient traitor tracing for structured data.

Keywords: Traitor tracing, chameleon hash, blockchain, watermarking

1. INTRODUCTION

Data sharing is an effective way to activate the huge value contained in different data, but the issue of data leakage accompanied by it has been haunted by experts and scholars. Therefore, the leaking traceability has become an open problem that needs to be resolved.

The credibility of data sharing is the primary prerequisite for achieving leak traceability. Blockchain as a distributed storage technology, each block in the chain can be abstractly described as a distributed ledger that introduces time attribute into it to form the time dimension, which improves the verifiability and traceability of transaction. Therefore, the blockchain is widely used to construct trusted data sharing and tracking schemes for Internet of Vehicles[1], supply chain[2], medical care [3-4], digital copyright protection[5] and other fields. Even though the combination of attribute-based encryption[6], federated learning[7] and other technologies with blockchain can well realize the verifiability and privacy of shared data, it is difficult to obtain the traceability after data leakage[8].

Digital watermarking is a technology about information hiding that embeds specific identity in the form of a bitstream into digital carrier without affecting the use value of the original works, which is a common method to complete copyright protection and anti-counterfeiting traceability. As an important branch of digital watermarking, database watermarking[9] is an effective means to realize the ownership protection of structured data[10-12]. A robust watermarking can still correctly extract the identity hidden in the digital works to realize the ownership confirmation even if a carrier is attacked to varying degrees[13-14]. But for watermarking, the ability to trace out the leaker is limited [15-19] in that it directly converts the user identity into bit stream and embeds these bits in the digital carrier. Digital fingerprint is a traitor tracking technology developed based on digital watermarking. The principle of this method is to embed the unique identification code[20] representing the buyer's identity into a digital carrier (such as DVD) to form a digital fingerprint[21-22], which achieve the binding of the buyer's identity and digital products. When illegal copies appear on the market, data owner can identify illegal users by extracting the fingerprinting from digital carriers to achieve the purpose of tracking down the traitor. To ensure anti-collusion, the length of the fingerprint code will expand as the number of users increases[23-24]. Embedding a long fingerprint code in a multimedia carrier (such as video, audio, image, etc.) with a lot of redundant space will not significantly change the imperceptibility in visual or auditory. However, the imperceptibility and reusability of the structured data (such as CSV) is seriously damaged if a large number of watermark codes are embedded into the digital work with limited embedding space. In addition, the monitor algorithm needs to detect all sharers when the potential leaker is unknown so as to result in extremely low efficiency in leaking detection. Therefore,

*youliangtian@163.com

International Conference on Computer Application and Information Security (ICCAIS 2021), edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 122600A · © The Authors. Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.2637501

there is an urgent need for an approach to track traitor through a shorter code length and higher detection efficiency when structured data is leaked.

Chameleon signature[25] is a one-to-many signature algorithm first proposed by Krawczyk and Rabin in 2000. It is consisted by a chameleon hash function and an ordinary signature scheme, which follows also the paradigm that "hash first and then sign". The generation of the message digest is completed by a special function, chameleon hash, which is a one-way function with trapdoor: the collision can be easily constructed when the trapdoor information is obtained; conversely, it is the same as the ordinary hash function and is collision resistant when there is no trapdoor. Therefore, the chameleon signature is not only undeniable and unforgeable, but also has characteristics for specific recipients, which is suitable for constructing a credible sharing scheme of one-to-many to achieve leak traceability. There is a defect of key exposure in the chameleon proposed earlier[26-27], that is, the signer's chameleon private key is likely to be leaked when calculating the collision, which weakens the security of the chameleon signature in a certain extent. For this, Feng et al.[28] and Chen et al.[29] introduced identity parameters to construct an identity-based chameleon-hash function signature. On this basis, Camenisch et al. [30] designed a hash function with ephemeral trapdoors to prevent the trapdoor holder from finding collision in "all-or-nothing" way in that the collision in the previous scheme is completely generated by the trapdoor holder. However, a series of encryption algorithms and zero-knowledge proofs are introduced in his scheme to avoid the leakage of private key and emphemeral key, which seriously reduces the computational efficiency of the Chameleon-hash. Later, Khalili et al.[31] analyzed the problem of low efficiency in many schemes[32-34] and constructed an enhanced collision-resistant and high-efficiency chameleon function based on bilinear mapping, which greatly shortens the length of the chameleon hash while solving the above problems. Although the above scheme can effectively avoid the key exposure and improve the execution performance to a certain extent, it still lacks a chameleon signature that possess a shorter coding length, a higher performance and the same security to realize trusted data sharing and traitor tracing for structured data.

In response, we present a chameleon short signature based on Khalili's chameleon hash [31] and Boneh's short signature[35] to ensure the non-repudiation of shared data, and construct a traceability chain with a cascading structure based on the characteristics of chameleon short signature to achieve effective traitor tracking in the big data scenario. The contributions in this work are as follows:

(1) *A novel chameleon short signature*. We design a chameleon signature with a shorter message length by combined Khalili's chameleon hash with Dan's short signature, and further improve efficiency while ensuring security so as to suit for the credible data sharing.

(2) *A traitor tracking model*. We design a cascade chain for shared data based on the characteristics of the proposed signature to ensure that all transactions on the same digital product belongs to the same transaction chain, which improves the detection efficiency of the illegal redistribution.

(3) *Trusted data sharing of structured data and efficient traitor tracking.* We embed the message of chameleon short signature between the data provider and the data buyer into the structured data and record the trading information on the blockchain to realize the non-repudiation of shared data. By extracting the watermark in the shared data and comparing the information in the cascaded chain to achieve efficient detection of traitors.

2. PRELIMINARIES

2.1 Notations

The main parameters involved in the chameleon short signature and the traitor tracking model in this article are shown in Table 1.

Table 1. Notations used in the paper.

Symbol	Description	Remarks	Symbol	Description	Remarks
x_h	Chameleon private key	$x_h \in Z_p^*$	ID_a	Identity to user A	$ID_a \in \{0,1\}*$
y_h	Chameleon public key	$y_h \in \mathbb{G}$	K_a	Watermarking key to user A	
$\tilde{y}_h$	Chameleon public key	$\tilde{y}_h \in \mathbb{G}$	S_a	Ordinary signature message to (ID_a, K_a)	
x_σ	Short signature private key	$x_\sigma \in Z_p^*$	M_a	User information to A	$M_a=\{ID_a, K_a, S_a\}$
y_σ	Short signature public key	$y_\sigma \in \mathbb{G}$	D	Digital carrier	
h	Chameleon hash	$h \in \mathbb{G}$	D_σ	Digital carrier embedded a signature	
σ_h	Chameleon short signature message	$\sigma_h \in \mathbb{G}$	B_i	The *i*-th copyright block	
M	Plaintext to be signed	$M \in \{0,1\}*$	B_{ij}	The *j*-th transaction block under B_i	
R	Check parameter *w.r.t* (h, M)	$R \in \mathbb{G}$	η	Match rate between bit strings	
$H_{\mathbb{G}}$	Global hash function	$H_{\mathbb{G}}: \{0,1\}* \to \mathbb{G}$	len_σ	Length of original signature	
H_p	Global hash function	$H_p: \{0,1\}* \to Z_p^*$	dis_{hc}	Hamming code distance between bit strings	

2.2 Chameleon hash

Chameleon hash function[25] *CH* = (*KeyGen*, *Hash*, *Check*, *Adapt*) can be consisted by four probability polynomial time algorithms, which are described as follows:

1. *KeyGen* (1^λ). The chameleon key generation algorithm on inputs the security parameter 1^λ to generate a public-private key pair (*pk*, *sk*), where *pk* and *sk* are related to the security parameter 1^λ.

2. *Hash* (*pk*, *m*). The hash generation algorithm takes as inputs the public key *pk* and the message *m*, it selects a randomness *r* to calculate the chameleon hash *h* and outputs (*h*, *r*).

3. $Check(h, m, r)$. The compatibility check algorithm takes as input the chameleon hash *h*, the message *m*, and the check value *r*, outputs a decision $b \in \{0,1\}$ indicating whether the (*h*, *m*, *r*) is compatible.

4. $Adapt(sk, h, m, r, m')$. The adapt algorithm on inputs private key *sk*, chameleon hash *h*, original plaintext message *m* and random number *r*, constructs the matching check parameter r' according to the collision message m', such that $Check(h, m, r) = Check(h, m', r') = 1$. Among them, (m, r) and (m', r') are called a pair of collisions.

2.3 Short signature

A short signature scheme[35] *BLS* = (*KeyGen*, *Sign*, *Verify*) is composed of three probability polynomial time algorithms, which are defined as follows:

1. *KeyGen* (1^λ): The key generation algorithm takes security parameter 1^λ as input and outputs the public key and the private key (*pk*, *sk*).

2. *Sign* (*M*, *sk*): The signature algorithm takes message $M \in \{0,1\}*$ and private key *sk* as input, and outputs the signature message σ of *M*;

3. *Verify* (*M*, σ, *pk*): The verification algorithm on inputs the signature message pair (*M*, σ) and the public key *pk*, outputs

a verification value $v \in \{0,1\}$. If $\hat{e}(g,\sigma) = \hat{e}(h, pk)$, then v = 1, means that the signature σ is a valid signature of the private key *sk* to the message *M*, otherwise v = 0. Among them, g is a public parameter.

2.4 Computational Diffie-Hellman assumption (CDH)

Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p related to the security parameter λ, where g is a generator of $\mathbb{G}$, on given $g, g^a, g^b \in \mathbb{G}$ for any $a,b \in Z_p$ to compute the $g^{ab} \in \mathbb{G}$. If the probability of successfully outputting $g^{ab} \in \mathbb{G}$ by polytime algorithm $\mathcal{A}$ is $\Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \varepsilon$, $\mathcal{A}$ has the advantage ε in $\mathbb{G}$ to solve the CDH problem.

Definition 1. We say that the Computational Diffie-Hellman Assumption (CDH) holds if no polytime algorithm has a non-negligible advantage in solving the CDH problem.

3. CHAMELEON SHORT SIGNATURE

This part, we improve the chameleon hash function[31] and combine the short signature scheme[35] to construct the chameleon short signature algorithm. It consists by seven parts and the framework is shown in Figure 1.

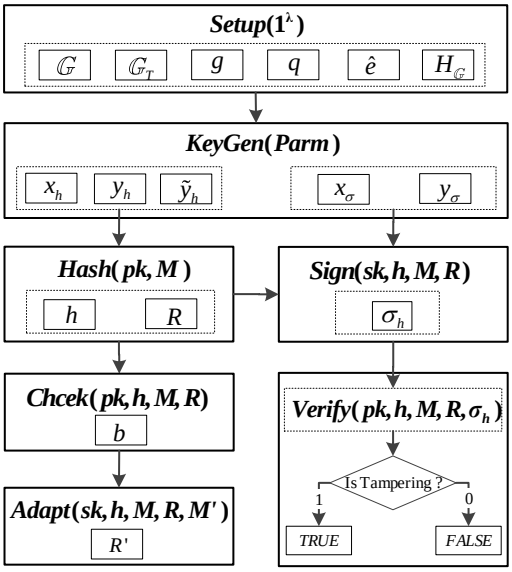

Figure 1. Chameleon short signature frame.

3.1 Algorithm description

During system initialization, ***Setup*** construct two groups $\mathbb{G}$, $\mathbb{G}_T$ and bilinear mapping $\hat{e}$ from security parameter λ. To get the keys, ***KeyGen*** generates the chameleon hash key pair $(x_h, y_h, \tilde{y}_h)$ and the short signature key pair (x_σ, y_σ) according to the public parameters. In the process of hash generation, ***Hash*** utilizes the chameleon public key $\tilde{y}_h \leftarrow pk$ to compute a chameleon hash h and its check parameter R with respect to the plaintext M. We can leverage ***Check*** to detect the compatibility of output parameters (h, M, R). When signing, ***Sign*** constructs a short signature message σ_h related to h from the signature private key $x_\sigma \leftarrow sk$. During the check of the signed message, ***Verify*** utilizes the signature public key $y_\sigma \leftarrow pk$ and the public parameter g to verify the legitimacy of σ_h with respect to h. To obtain the check parameters that suitable for the new plaintext M' and the chameleon hash h, ***Adapt*** first detects the compatibility of (h, M, R) by ***Check***, and then calculates the check parameters R' of the plaintext M' according to the chameleon private key $x_h \leftarrow sk$. The specific definition of the above algorithm is as follows:

(1) $Setup(\lambda) \rightarrow Parm$

Let $\mathbb{G}=\langle g\rangle$ be the gap group of order q, g be a generator of $\mathbb{G}$, and the prime number $q\geq 2^{\lambda}$. $\hat{e}:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$, for any $a,b\in\mathbb{G}$ such that $\hat{e}(g^a,g^b)=\hat{e}(g,g)^{ab}$. The system selects the global anti-collision hash function $H_{\mathbb{G}}:\{0,1\}^*\to\mathbb{G}$, and publishes the system parameters $Parm=\{\mathbb{G},\mathbb{G}_T,g,q,\hat{e},H_{\mathbb{G}}\}$.

(2) $KeyGen(Parm)\to(sk,pk)$

Set randomness $x_h\xleftarrow{R}Z_q^*$ and $x_\sigma\xleftarrow{R}Z_q^*$, calculate $y_h=g^{x_h}$, $\tilde{y}_h=g^{1/x_h}$, and $y_\sigma=g^{x_\sigma}$ respectively to output the following public-private key pair:

$$sk=\{x_h,x_\sigma\}$$

$$pk=\{y_h,\tilde{y}_h,y_\sigma\}$$

(3) $Hash(pk,M)\to(h,R)$

Let $m=H_{\mathbb{G}}(M)$ *w.r.t.* the plaintext *M*, the chameleon public key $\tilde{y}_h\leftarrow pk$, randomness $r\xleftarrow{R}Z_q^*$, calculate the chameleon hash *h* with equation (1), and the check parameter $R=g^r$.

$$h=m\cdot\tilde{y}_h^r\in\mathbb{G} \tag{1}$$

(4) $Check(pk,h,M,R)\to b\in\{0,1\}$

Parse the chameleon public key $\tilde{y}_h\leftarrow pk$, construct $m=H_{\mathbb{G}}(M)$, and then detect the compatibility of (h,m,R) according to equation (2). If the equation holds, output 1; otherwise, output 0.

$$\hat{e}(h/m,\ g)=\hat{e}(R,\ \tilde{y}_h) \tag{2}$$

(5) $Adapt(sk,h,R,M,M')\to R'$

If $Check(h,M,R)=0$ returns $\perp$, otherwise, let $m'=H_{\mathbb{G}}(M')$, the chameleon private key $x_h\leftarrow sk$, and compute the check parameter R' according to equation (3).

$$R'=(h/m')^{x_h} \tag{3}$$

(6) $Sign(sk,h,M,R)\to(S_\sigma)$

Check (*h*, *M*, *R*) before signing, and then calculate the short signature message σ_h with the signature private key $x_\sigma\leftarrow sk$ according to equation (4).

$$\sigma_h=h^{x_\sigma}\in\mathbb{G} \tag{4}$$

(7) $Verify(pk,h,R,M,\sigma_h)\to b\in\{0,1\}$

Check (*h*, *M*, *R*) before verification, then verify the legitimacy of the signature σ_h *w.r.t.* the chameleon hash h, the signature public key $y_\sigma\leftarrow pk$ and the public parameter g according to equation (5). if the equation holds, then b=1, otherwise b=0.

$$\hat{e}(h,y_\sigma)=\hat{e}(\sigma_h,g) \tag{5}$$

3.2 Security model

The security model of the chameleon short signature is composed of an enhanced Collision-Resistance game $CollRes_{\mathcal{A}}^{CDH}(\lambda)$ of the chameleon hash function and an Existential-Unforgeability game $EUF_{\mathcal{A}}^{CDH}(\lambda)$ of the short

signature scheme. Each game contains a challenger $\mathcal{B}$ and an adversary $\mathcal{A}$. The $\mathcal{B}$ simulates the operation of the system and answers the queries from the $\mathcal{A}$. The formal definitions of each game are shown in Figures 2a and 2b, respectively.

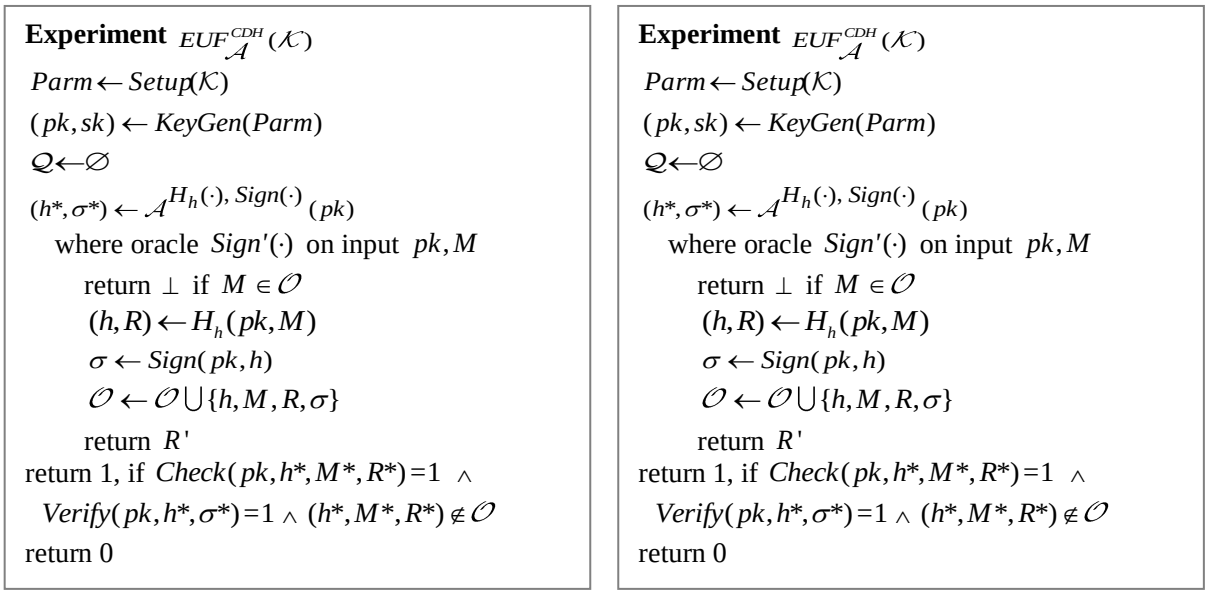

Figure 2. (a) Chameleon hash Collision-Resistance game; (b) Short signature Existential-Unforgeability game.

Collision Resistance. Collision resistance says, even if an adversary has access to an adapt oracle, it cannot find any collisions for messages other than the ones queried to the adapt oracle. Note, this is an even stronger definition than key-exposure, which only requires that one cannot find a collision for some new plaintext, i.e., for some auxiliary value for which the adversary has never seen a collision.

Definition 2 (Collision-Resistance). A chameleon-hash is collision-resistant, if for any polytime adversary $\mathcal{A}$ there exists a negligible function ω such that $\Pr[EUF_{\mathcal{A}}^{CDH}(\mathcal{K}) = 1] \leq \omega(\mathcal{K})$. The corresponding experiment is depicted in Figure 2a.

Existential-Unforgeability. The existential unforgeability of a digital signature means that an adversary cannot achieve a valid forged signature for at least one message even if the adversary has access to a sign oracle.

Definition 3 (EUF-CMA). A chameleon short signature scheme becomes existential unforgeability against adaptive selection message attack, referred to as EUF-CMA security, if for any polytime adversary $\mathcal{A}$ there exists a negligible function ω such that $\Pr[CollRes_{\mathcal{A}}^{CDH}(\mathcal{K}) = 1] \leq \omega(\mathcal{K})$. The corresponding experiment is depicted in Figure 2a.

4. TRAITOR TRACING MODEL

Blockchain relying on its decentralization, proof-tampering, openness and traceability, is extremely suitable for building a data sharing and leakage tracing framework. However, the blockchain usually stores the blocks in a sequential structure, which results in all the blocks on the chain need to be traversed when data querying for tracking, so the efficiency to traitor tracking is low. In response, first, we construct trusted a data sharing framework based on the chameleon short signature algorithm (3.1); Then, design a cascade chain to achieve the efficient tracing on the data leaker according to the constructed framework. Through the above design, we complete finally the trusted sharing to data and the efficient tracking to leaker.

4.1 Trusted sharing

The trusted sharing framework embeds the short chameleon signatures of both parties in the transaction as a watermark into the shared data to reduce the amount of signatures held in the entire system and the size of the watermark in the digital carrier, so as to provide conditions for effective tracking while ensuring the availability and robustness of the digital carrier. The trusted data sharing framework is shown in Figure 3.

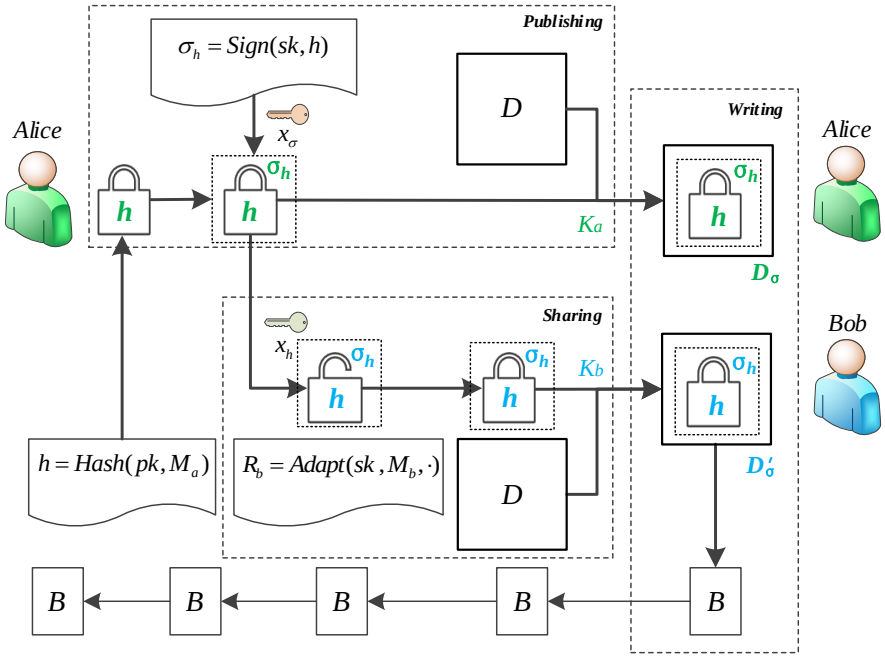

Figure 3. Shared traceability framework.

In publishing original data D, first, the data provider A generates a chameleon hash h based on its own identity information $M_a=\{ID_a, K_a, S_a\}$, where ID_a is the unique identifier of user A, K_a is the embedding key for watermark, and S_a is the regular signature to (ID_a, K_a); Then, the user A signs h with his own signature private key x_σ, and embeds the chameleon short signature σ_h as watermark into D through K_a to obtain the watermarked carrier D_σ containing the chameleon signature message; Finally, the provider publishes the parameter $\{ID_a, K_a, S_a, R_a, h, \sigma_h\}$ generated in the above process to the blockchain system to form copyright information about D.

To get a trusted copy of data D, user B first submits his identity information $M_b=\{ID_b, K_b, S_b\}$; Then, the provider A generates the check parameter R_b $w.r.t$ (h, M_b), and embeds σ_h into the original data D using B's watermark key K_b to obtain the watermarked carrier D_σ' so as to ensure that the transaction behavior of both parties is unforgeable and undeniable; Finally, the above-mentioned parameters $\{ID_b, K_b, S_b, R_b\}$ are all recorded on the blockchain to keep the traitor can be tracked when the data is leaked.

4.2 Efficient tracking

For achieve the efficient tracing to the data leaker, the cascade chain consists of three parts: the copyright chain, the transaction chain, and the copyright block index, the structure is shown in Figure 4. During the publishing of data D_i, the consensus node generates a new copyright block B_n with relevant parameters $\{ID_a, K_a, S_a, R_a, h, \sigma_h\}$ and appends B_n to the chain to form a copyright chain $\{B_i\}_{i=1,2,\cdots,n.}$ In the process of each sharing data D_i, the consensus node generates a transaction block B_{im} based on the parameters $\{ID_m, K_m, S_m, R_m\}$, and appends B_{im} below the corresponding copyright block B_i to form a transaction chain $\{B_{ij}\}_{j=1,2,\cdots,m}$, which makes sure that the transaction information of the same copyright data belongs to the same transaction chain. In order to further improve the query efficiency of traceability information, we let σ_h in the watermarked carrier as the query key $key \leftarrow H_p(\sigma_h)$ to build an index on the copyright block, where $H_p: \{0,1\}^* \rightarrow Z_p^*$. If someone checks the transaction, he can extract the signature σ_h in the carrier and quickly locate the copyright block through binary search to effectively obtain the copyright information and all transaction information in the corresponding transaction block.

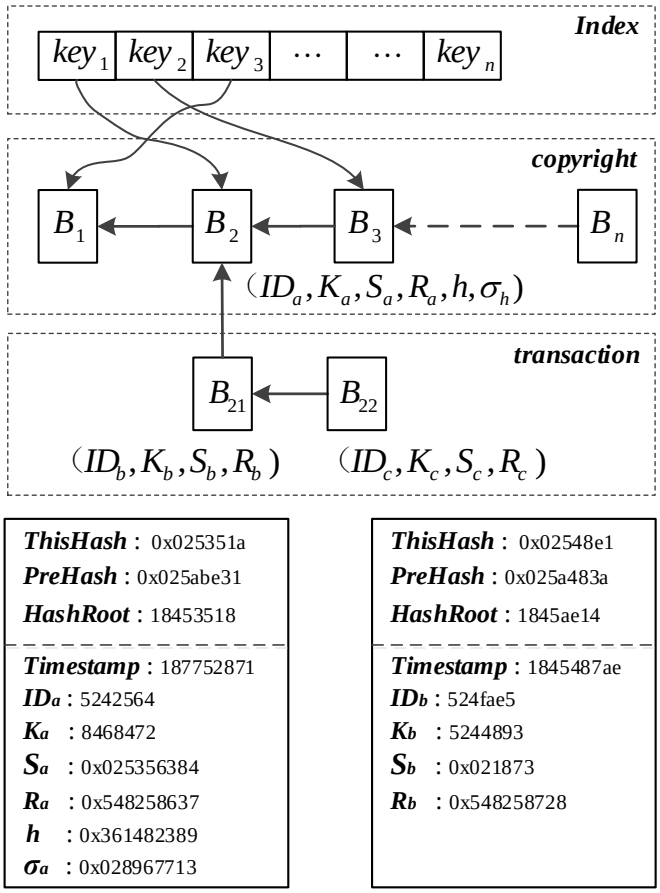

Figure 4. The cascade chain and its block structure.

If there is a suspected illegal copy D_σ^* on the market, the data owner can quickly query the corresponding copyright block B_i and all transaction blocks $\{B_{ij}\}_{j=1,2,\cdots,n}$ in the cascade chain with the keyword $key \leftarrow H_p(\sigma_h)$ to collect the copyright information $\{ID_i, K_i, S_i, R_i, h, \sigma_h\}$ and all transaction information $\{ID_{ij}, K_{ij}, S_{ij}, R_{ij}\}_{j=1,2,\cdots,m}$. To track down a traitor, the owner utilizes the corresponding watermark extraction algorithm to obtain the signature information $\sigma_h^* \leftarrow DeWater(D_\sigma^*, K_{ij})$ in the carrier D_σ^* through the watermark key K_{ij}, verifies the correctness of copyright information by $Verify(pk, h, M_{ij}, R_{ij}, \sigma_h^*)$, and checks the consistency of the transaction by $Check(h, M_{ij}^*, R_{ij}^*)$, where $M_{ij}^* = \{ID_{ij}^*, K_{ij}^*, S_{ij}^*\}$. If the above detects are passed, ID_{ij}^* can be inferred as a potential data leaker.

5. ANALYSIS

5.1 Security

This part mainly analyzes the credibility and traceability about the traitor tracking model based on the security of the proposed chameleon short signature (see the appendix for the security analysis). The credibility considered in this model refers to the trustworthiness of both parties in the transaction, that is, the transaction initiator is the data purchaser himself and the seller is the data provider himself. The Traceability considered in this traitor tracing model refers to the undeniability of both parties in the transaction, that is, the purchaser cannot deny that he is the transaction initiator and the provider cannot deny that he is the transaction executor.

Property 1 (credibility): Let the ordinary signature submitted by the data purchaser be unforgeable, if the chameleon signature in the traitor tracing model is existential unforgeability, the data sharing approach is credible.

Proof: From Section 4.1, the data purchaser *B* initiates a transaction to the data provider *A* with his ordinary signature message S_b. Because of the ordinary signature is unforgeable, the data provider *A* can confirm whether the initiator is the purchaser himself by verifying the legality of S_b. When the data purchaser *B* receives the shared copy from provider, he can extract the chameleon signature σ_b in the carrier by his own watermark key $K_b \leftarrow M_b = (ID_b, K_b, S_b)$ and verify the legality of (h, σ_b) to confirm that the shared data really comes from the data provider himself. Therefore, the sharing approach between the data purchaser and the provider is credible. (Property 1 is proved)

Property 2 (Traceability) Let the ordinary signature submitted by the data purchaser be unforgeable, if the chameleon hash is collision resistant and the chameleon signature is existential unforgeability, the data sharing approach is traceable.

Proof: It is known from Section 4.2 that if there is an illegal shared copy D_σ^* on the market, the data inspector can extract the chameleon signature σ_h^* through the purchaser's watermark key $K_b \leftarrow M_b$. Since the ordinary signature S_b about (ID_b, K_b) is unforgeable, the purchaser cannot deny that the transaction was initiated by him. Because of the chameleon hash *h* w.r.t. M_b is collision resistant, the inspector can check the compatibility of (h, M_b , R_b) to prevent the purchaser from denying that the illegal copy comes from himself. Since the provider's chameleon signature σ_h^* related to *h* is existential unforgeability, the provider cannot deny that the data carrier is authorized by himself. Thus, the data sharing approach between the data purchaser and the provider is traceable (Property 2 is proved).

5.2 Simulation

This experiment leverages 1 host (CPU is Intel Core i5 7500, memory is 8 GB, operating system is Windows 10) to simulate the big data platform and watermark center, and 4 hosts (CPU is Intel Core i3 2120, memory is 4 GB) , operating system is Windows 7) to simulate the consensus node. We choose C++ as the main programming language to build the chameleon short signature algorithm, the watermarking algorithm and the PBFT consensus. Based on the above experimental environment, we take 20,000 rows and 50 columns of structured data as a carrier, and conduct multiple data sharing and tracing experiments.

In order to analyze the performance of the improved chameleon hash, we conduct experiments on the proposed scheme and the Khalili scheme with different sizes of data, and the time consumption is shown in Figure 5. From it, we can observe that the time consumption of the *Hash* and the *Check* is reduced by about 15ms, the *Adapt* is reduced by about 45ms. The reason is that the proposed algorithm reduces the mapping operation of the plaintext hash and the inverse operation of the group $\mathbb{G}$, so the time comparison of each part is reduced except for the key generation *KeyGen*. Overall, there is a certain improvement in the performance of the proposed algorithm.

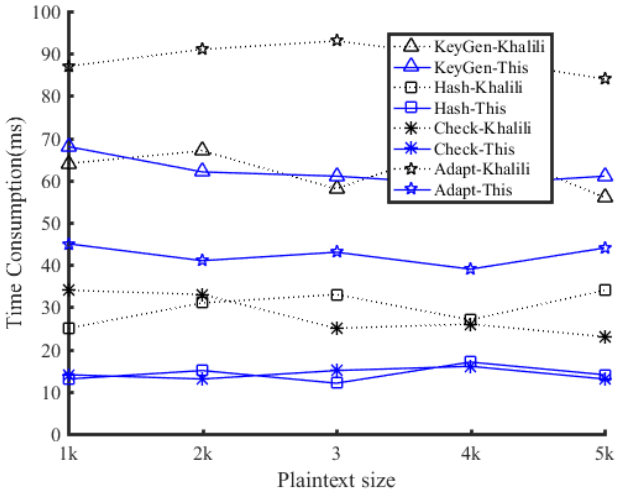

Figure 5. Comparison of the chameleon hash.

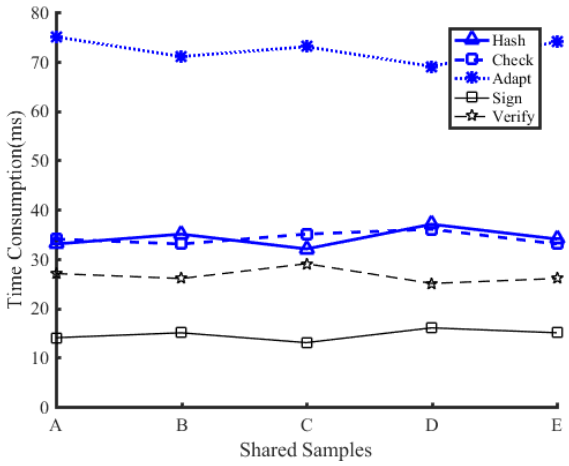

Figure 6. Consumption of the chameleon short signature.

We conducted 100 data sharing experiments on 5 published digital works to obtain the time cost of the chameleon short signature, and randomly selected 5 sharing instances of A, B, C, D and E from the 5 works for time analysis, which is shown in Fgure 6. It shows that the time consumption of the hash generation (*Hash*) and compatibility check (*Check*) about the proposed chameleon hash is approximately 35ms, the time cost of the adjustment to check parameter (*Adapt*) is approximately 75ms, and the time consumption of the signature (*Sign*) and verify (*Verify*) are maintained at 15ms and 25ms, respectively. On the whole, the execution time of the proposed chameleon short signature can well meet the practical requirements to achieve trusted sharing.

The traditional blockchain leverages a sequential chain to link blocks, while we use a cascade chain to connect blocks. In order to further compare and analyze the retrospective time cost of the two chain, we respectively conducted 100 data sharing on 5 digital work to ensure that 500 transaction blocks are recorded on the chain; then, each work randomly selects a leaked copy for tracing. For the same node, the consumption of leak detection in the two traceability chains is shown in Figure 7. What can be seen from the figure is that the detection efficiency of the cascade chain is about 3 times that of the traditional chain. The reason is that the cascaded chain only detects transaction blocks related to the original data, and does not compare all blocks, so the efficiency of leakage tracking is significantly better than that of the traditional chain.

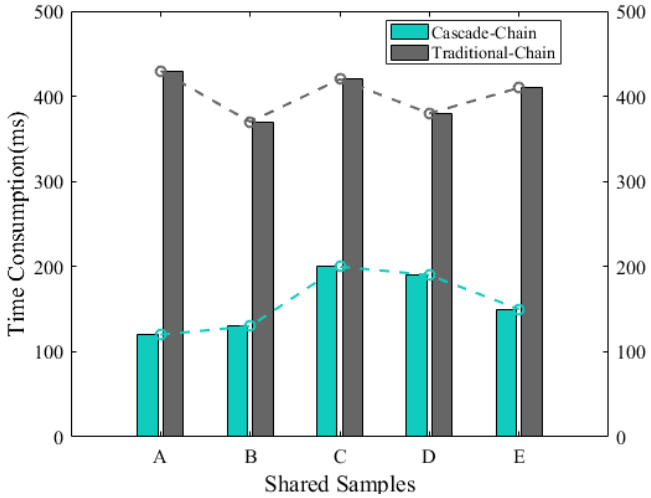

Figure 7. Consumption on leak detection.

In order to fully analyze the time consumption of the traitor tracking system, we embed the chameleon short signature into the digital carrier by the watermarking algorithm GAHSW[19], and write the transaction information on the chain by the PBFT algorithm. Based on the above design, we randomly selected five shared instances of A, B, C, D and E for

analysis. The overall performance is shown in Figure 8. The time consumption in data sharing includes embedded watermarks and consensus writing blocks, and the consumption in traitor tracing includes block reading and watermark detection. From the figure, we can see that the entire sharing and tracing time consumption is concentrated on the embedding and extraction of the watermark. Therefore, it is the key to improve the efficiency of the watermarking algorithm for improving the efficiency of traitor detection.

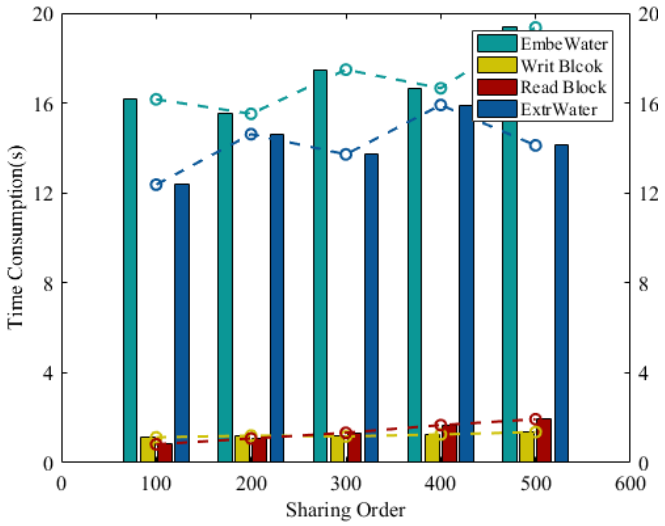

Figure 8. Consumption on traitor tracing system.

Malicious users usually attack authorized copies with watermarks to varying degrees to obtain illegal copies that cannot be held accountable. For a single attacker, row deletion and column deletion are the easiest attacks, but for multiple attackers, collusion combinations, maximum, minimum and average attacks are the most common attacks. Among them, the attack of collusion combination refers to each conspirator taking out the same amount of different data to form an illegal copy; the attack of maximum (minimum, average) is to use the maximum (minimum, average) of the element at the corresponding position in all authorized copies as the element of the illegal copy.

In order to analyze the traceability of the traitor tracking model under different watermarking algorithms, we respectively embed the chameleon short signature into the digital carrier based on four robust watermarking algorithms such as GAHSW[19], GADEW[15], RLW[11], RRW[14] and attack those carries to varying degrees. The detection effect is shown in Tables 2-4.

Table 2. Detection effect on single traitor.

Attack type	**Scheme**	**Degree of attack**				
		10%	**20%**	**30%**	**40%**	**50%**
Row delete	GAHSW	√	√	√	√	√
	GADEW	√	√	√	√	√
	RLW	√	√	√	√	×
	RRW	√	√	√	×	×
Column delete	GAHSW	√	√	√	√	√
	GADEW	√	√	√	√	×
	RLW	√	√	√	×	×
	RRW	√	√	√	√	×

Table 2 shows the detection results of digital carriers under different watermarking algorithms and different degrees of deletion attacks in a single attacker scenario. From the table, we can get that for the scheme GAHSW, whether it is row deletion or column deletion, the tracking algorithm can correctly detect the signed message about owners and consumers as long as the attack degree it suffers does not exceed 50%; And for the scheme GADEW, as long as the attack degree does not exceed 40%, it can also correctly detect the signed message. In comparison, the GAHSW algorithm can deal with the deletion attack in a single traitor well.

Table 3. The matching rate of two colluders.

Attack strategy	**Scheme**	**2 conspirators**				
		A ○	**B** ○	**C** ×	**D** ×	**E** ×
Combination replacement	GAHSW	93%	91%	45%	51%	46%
	GADEW	98%	95%	37%	42%	53%
	RLW	86%	91%	63%	46%	57%
	RRW	73%	79%	51%	36%	46%
Max	GAHSW	74%	77%	53%	55%	47%
	GADEW	83%	83%	43%	51%	48%
	RLW	76%	69%	51%	47%	34%
	RRW	63%	68%	39%	51%	48%
Min	GAHSW	79%	75%	42%	36%	51%
	GADEW	81%	76%	53%	61%	42%
	RLW	73%	77%	39%	48%	55%
	RRW	67%	72%	53%	46%	41%
Average	GAHSW	74%	69%	38%	54%	47%
	GADEW	79%	76%	48%	38%	42%
	RLW	76%	71%	46%	43%	48%
	RRW	65%	69%	54%	39%	54%

If it is defined that the matching rate between the original signature and the extracted signature is $\eta=(len_{\sigma}-dis_{hc})/len_{s}$, where len_{σ} is the length of the original signed message, and dis_{hc} is the distance of Hamming code between the original signed signature and the extracted. Under environment of the different colluders, the different attack strategy and the different watermarking algorithms, we let "○" means participating in collusion and "×" indicates not participating, then the matching rate under 2 colluders (A, B) and 3 colluders (A, B, C) is show in Tables 3 and 4, separately.

Table 4. The matching rate of three colluders.

Attack strategy	Scheme	3 conspirators				
		A ○	B ○	C ○	D ×	E ×
Combination replacement	GAHSW	81%	79%	84%	41%	36%
	GADEW	86%	83%	87%	43%	51%
	RLW	83%	79%	81%	59%	53%
	RRW	65%	67%	71%	49%	51%
Max	GAHSW	67%	65%	69%	53%	47%
	GADEW	76%	78%	76%	42%	37%
	RLW	61%	63%	62%	39%	46%
	RRW	59%	63%	58%	51%	47%
Min	GAHSW	68%	61%	64%	39%	48%
	GADEW	76%	78%	73%	51%	53%
	RLW	63%	67%	71%	48%	36%
	RRW	64%	58%	61%	37%	51%
Average	GAHSW	63%	71%	68%	54%	47%
	GADEW	73%	69%	75%	48%	39%
	RLW	62%	56%	64%	51%	48%
	RRW	61%	58%	63%	46%	38%

It can be seen from Table 3 that all the watermarking algorithms can well detect out potential traitors in the case of 2 colluders. From Table 4, we can get that if 3 colluders conduct the attack of combined substitution on the digital carrier, all algorithms can also detect out potential traitors well. However, for maximum and minimum attacks, the algorithms GAHSW, GADEW, and RLW have relatively better anti-collusion attacks; for average attacks, the algorithms GAHSW and GADEW are more resistant to collusion; On the whole, the efficiency of identifying colluders is GADEW > GAHSW > RLW > RRW, but the scheme GADEW modifies the original carrier to a greater extent and makes the data availability relatively low, so the GAHSW algorithm is more suitable for detecting data leakers in the traitor tracking model.

6. CONCLUSION

The traceability of data leaks in the big data environment is an issue that people have been paying attention to. This paper takes structured data as the main research point. A chameleon short signature is designed to complete trusted data sharing, and a cascade chain is established to effectively achieve traitor tracking. The security and efficiency of the scheme are analyzed through provable security model and experimental simulation. We hope to provide valuable reference information for related researchers.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China under Grant Nos. 61662009 and 61772008; Science and Technology Major Support Program of Guizhou Province under Grant No.20183001; Key Program of the

National Natural Science Union Foundation of China under Grant No.U1836205; Science and Technology Program of Guizhou Province under Grant No.[2019]1098; Project of High-level Innovative Talents of Guizhou Province under Grant No. [2020]6008; Science and Technology Program of Guiyang under Grant No.[2021]1-5; Graduate Research Foundation of Guizhou Province No.[2021].

REFERENCES

[1] Fan, K., Pan, Q., Zhang, K., Bai, Y., Sun, S., Li, H. and Yang, Y., "A secure and verifiable data sharing scheme based on blockchain in vehicular social networks," IEEE Transactions on Vehicular Technology, 69(6), 5826-5835 (2020).
[2] Qi, S., Lu, Y., Zheng, Y., Li, Y. and Chen, X., "Cpds: Enabling compressed and private data sharing for industrial IoT over blockchain," IEEE Transactions on Industrial Informatics, 17(4), 2376-2387 (2020).
[3] Chen, M., Qian, Y., Chen, J., Hwang, K., Mao, S. and Hu, L., "Privacy protection and intrusion avoidance for cloudlet-based medical data sharing," IEEE Transactions on Cloud Computing, 8(4), 1274-1283 (2020).
[4] Liu, X., Wang, Z., Jin, C., Li, F. and Li, G., "A blockchain-based medical data sharing and protection scheme," IEEE Access, 7, 118943-118953 (2019).
[5] Fu, W., Shi, H., Huang, J., Ji, Y. and Yan, S., "Spatial image data traceability and interaction mechanism based on alliance chain," IEEE 10th Inter. Conf. on Software Engineering and Service Science (ICSESS), 586-591 (2019).
[6] Pu, Y., Hu, C., Deng, S. and Alrawais, A., "RPEDS: A recoverable and revocable privacy-preserving edge data sharing scheme," IEEE Internet of Things Journal, 7(9), 8077-8089 (2020).
[7] Lu, Y., Huang, X., Zhang, K. and Maharjan, S., "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," IEEE Transactions on Vehicular Technology, 69(4), 4298-4311 (2020).
[8] Qureshi, A. and Megías, D., "Blockchain-based P2P multimedia content distribution using collusion-resistant fingerprinting," 2019 Asia-Pacific Signal and Information Processing Association Annual Summ. and Conf. (APSIPA ASC), 1606-1615 (2019).
[9] Agrawal, R. and Kiernan, J., "Watermarking relational databases," Proc. of the 28th Inter. Conf. on Very Large Data Bases (VLDB'02), 155-166 (2002).
[10] Shehab, M., Bertino, E. and Ghafoor, A., "Watermarking relational databases using optimization-based technique," IEEE Transactions on Knowledge and Data Engineering, 20(1), 116-129 (2008).
[11] Franco-Contreras, J., Coatrieux, G., Cuppens, F., Cuppens-Boulahia, N. and Roux, C., "Robust lossless watermarking of relational databases based on circular histogram modulation," IEEE Transactions on Information Forensics & Security, 9(3), 397-410 (2017).
[12] Gort, M., Olliaro, M., Cortesi, A. and Feregrino, C., "Semantic-driven watermarking of relational textual databases," Expert Systems with Applications, 167(2), 368-379 (2020).
[13] Kamran, M., Suhail, S. and Farooq, M., "A robust, distortion minimizing technique for watermarking relational databases using once-for-all usability constraints," IEEE Transactions on Knowledge and Data Engineering, 25(12), 2694-2707 (2013).
[14] Saman, I., Kamran, M. and Anwar, Z., "RRW-a robust and reversible watermarking technique for relational data," IEEE Transactions on Knowledge & Data Engineering, 27(4), 1132-1145 (2015).
[15] Gupta, G. and Pieprzyk, J., "Reversible and blind database watermarking using difference expansion," International Journal of Digital Crime and Forensics, 1(2), 42-54 (2008).
[16] Khurram, J. and Asifullah, K., "Genetic algorithm and difference expansion based reversible watermarking for relational databases," Journal of Systems and Software, 11(86), 2742-2753 (2013).
[17] Imamoglu, M., Ulutas, M. and Ulutas, G., "A new reversible database watermarking approach with firefly optimization algorithm," Mathematical Problems in Engineering, 1-14 (2017).
[18] Zhang, Y., Yang, B. and Niu, X. M., "Reversible water-marking for relational database authentication," Journal of Computers, 17, 59-65 (2006).
[19] Hu, D. H., Zhao, D. and Zheng, S. L., "A new robust approach for reversible database watermarking with distortion control," IEEE Transactions on Knowledge and Data Engineering, 31, 1024-1037 (2019).
[20] Boneh, D. and Shaw, J., "Collusion-secure fingerprinting for digital data," IEEE Transactions on Information Theory, 44(5), 1897-1905 (1998).

[21] He, S. and Wu, M., "Collusion-resistant video fingerprinting for large user group," IEEE Transactions on Information Forensics and Security, 2(4), 697-709 (2007).
[22] Li, M., Chang, H., Xiang, Y. and An, D., "A novel anti-collusion audio fingerprinting scheme based on fourier coefficients reversing," IEEE Signal Processing Letters, 27, 1794-1798 (2020).
[23] Fodor, G., Schelkens, P. and Dooms, A., "Fingerprinting codes under the weak marking assumption," IEEE Transactions on Information Forensics & Security, 13(6), 1495-1508 (2017).
[24] Fan, J., Gu, Y., Hachimori, M. and Miao, Y., "Signature codes for weighted binary adder channel and multimedia fingerprinting," IEEE Transactions on Information Theory, 67(1), 200-216 (2021).
[25] Krawczyk, H. and Rabin, T., "Chameleon signatures," *NDSS Symp.*, 143-154 (2000).
[26] Ateniese, G. and Medeiros, B., "Identity-based chameleon hash and applications," Lecture Notes in Computer Science, 167, 164-180 (2004).
[27] Ateniese, G. and Medeiros, B. D., "On the key exposure problem in chameleon hashes," Inter. Conf. on Security in Communication Networks, 165-179 (2005).
[28] Feng, B., Deng, R. H., Ding, X., Lai, J. and Zhao, Y., "Hierarchical identity-based chameleon hash and its applications," Inter. Conf. on Applied Cryptography & Network Security, 201-219 (2011).
[29] Chen, X., Zhang, F., Susilo, W., Tian, H., Li, J. and Kim, K., "Identity-based chameleon hashing and signatures without key exposure," Information Sciences, 265, 198-210 (2014).
[30] Camenisch, J., Derler, D., Krenn, S., Pöhls, H. C., Samelin, K. and Slamanig, D., "Chameleon-hashes with ephemeral trapdoors and applications to invisible sanitizable signatures," IACR Inter. Work. on Public Key Cryptography, 152-182 (2017).
[31] Khalili, M., Dakhilalian, M. and Susilo, W., "Efficient chameleon hash functions in the enhanced collision resistant model," Information Sciences, 510, 155-164 (2020).
[32] Gennaro, R., Gentry, C., Parno, B. and Raykova, M., "Quadratic span programs and succinct NIZKs without PCPs," Annual Inter. Conf. on the Theory and Applications of Cryptographic Techniques, 626-645 (2013).
[33] Ateniese, G., Magri, B., Venturi, D. and Andrade, E., "Redactable blockchain-or-rewriting history in bitcoin and friends," IEEE European Symp. on Security and Privacy, 111-126 (2017).
[34] Groth, J. and Maller, M., "Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks," Annual International Cryptology Conf., 581-612 (2017).
[35] Dan, B., Boyen, X. and Shacham, H., "Short group signatures," 24th Annual Inter. Cryptology Conf., 3152, 227-242 (2004).

APPENDIX

This part is to analyze the security of the proposed chameleon short signature based on the $CollRes_{\mathcal{A}}^{CDH}(\mathcal{K})$ and $EUF_{\mathcal{A}}^{CDH}(\mathcal{K})$ game model (3.2).

Theorem 1 Let $\mathbb{G}$ be the gap group and $H_{\mathbb{G}}$ be the random oracle on $\mathbb{G}$. If the *CDH* assumption holds on $\mathbb{G}$, then the chameleon hash function is collision resistance.

Specifically, suppose there is a polynomial adversary $\mathcal{A}$ that breaks the chameleon hash scheme with the advantage of $\omega(\mathcal{K})$, then there must be an adversary $\mathcal{B}$ to solve the *CDH* on $\mathbb{G}$ at least by the advantage of

$$Adv_{\mathcal{B}}^{CDH} \geq \frac{\omega(\mathcal{K})}{e \cdot q_H}$$

where e is the base of the natural logarithm, q_H is the maximum number of queries to $H_{\mathbb{G}}$.

Proof: From Section 3.2, the process of the game $CollRes_{\mathcal{A}}^{CDH}(\mathcal{K})$ between $\mathcal{A}$ and $\mathcal{B}$ is as follows.

(1) Adversary $\mathcal{B}$ runs $Setup(\mathcal{K})$ and $KeyGen(Parm)$ to generate the key pair (pk, sk), selects a random function $H_{\mathbb{G}} \xleftarrow{R} \{H : \{0,1\}^* \to \mathbb{G}\}$, and send the system parameters and public key to adversary $\mathcal{A}$.

(2) Adversary $\mathcal{A}$ can send any inquiry $H_{\mathbb{G}}(\cdot)$ about M' to adversary $\mathcal{B}$ and inquiry $Adatp'(\cdot)$ about the check parameters, adversary $\mathcal{B}$ response R' as the corresponding answer. During this process, q is the maximum number of times $\mathcal{A}$ queries $H_{\mathbb{G}}(\cdot)$, and $M' \notin \mathcal{O}$.

(3) Adversary $\mathcal{A}$ outputs a set of chameleon hash pairs $(h^*, M^*, R^*, M'^*, R'^*)$. If $Check(h^*, M^*, R^*)=1 \wedge Check(h^*, M'^*, R'^*)=1$, then the attack from $\mathcal{A}$ is successful, in which the check parameter R'^* of M'^* has not been queried by it before.

Analysis: Knowing $R' = (h / m')^x$ from equation (3) in Section 3.1, if $\mathcal{B}$ can find a certain r^*, such that $r^* = h / m'$, then $(r^*)^x = (h / m')^x$. If m' is the hash value of a certain plaintext M', then $(h / m')^x$ is the check parameter for (M', h). Since adversary $\mathcal{B}$ knows $\{g, h, y = g^x, m' = H_{\mathbb{G}}(M')\}$ and want to leverage $\mathcal{A}$ as a subroutine to attack the chameleon hash algorithm, its goal is to find out $R' = (r^*)^x = (h / m')^x$. In step (3) of the above game $CollRes_{\mathcal{A}}^{CDH}(\mathcal{K})$, (M'^*, R'^*) is generated by adversary $\mathcal{A}$, but $H_{\mathbb{G}}(M'^*)$ is generated by $\mathcal{B}$, so $\mathcal{B}$ can be set $r_{m'}^* = H_r(M'^*) = h / H_{\mathbb{G}}(M'^*)$. When $\mathcal{B}$ lets r^* be the potential hash of a target plaintext, his goal is to call $\mathcal{A}$ to calculate $(r^*)^x$ based on the triple (g, g^x, r^*), which is to solve the *CDH* problem. Throughout the game, $\mathcal{B}$ doesn't know which plaintext will be generated by $\mathcal{A}$ to forge a check parameter. Therefore, he has to make a guess that the *j*-th query H_r corresponds to $\mathcal{A}$'s final forged result.

For simplicity without loss of generality, we assume that 1). Adversary $\mathcal{A}$ must have asked (*h*, *M*, *R*) before asking $H_r(M')$; 2). Adversary $\mathcal{A}$ will not initiate the same query $H_r(M')$ twice to M'; 3). Adversary $\mathcal{A}$ must have asked $H_r(M')$ before querying the check parameters of M'; 4). Adversary $\mathcal{A}$ he must have asked (*h**, *M**, *R**) and $H_r(M'^*)$ before he outputs (M'^*, R'^*).

In the actual process, $\mathcal{B}$ implicitly regards $u = g^a$ in the known tuple $(g, u = g^a, r^*)$ as its own public key (in fact, $\mathcal{B}$ does not know the specific value of a), then $(r^*)^a$ is a forgery check parameter of a certain plaintext, namely $R'^* = (r^*)^a = (H_r(M'))^a = (h / H_{\mathbb{G}}(M'))^a$, where $(r^*)^a$ is forged by $\mathcal{A}$. To hide instance $u = g^a$, $\mathcal{B}$ needs to select a randomness t and send $u \cdot g^t$ to $\mathcal{A}$ as the public key of $\mathcal{B}$.

The following proves that the collision resistance game $CollRes_{\mathcal{A}}^{CDH}(\mathcal{K})$ of the chameleon hash can be reduced to the *CDH* problem.

(1) $\mathcal{B}$ sends the generator g of group $\mathbb{G}$ and the public key $y = u \cdot g^t = g^{a+t} \in \mathbb{G}$ to $\mathcal{A}$, where $t \xleftarrow{R} Z_q^*$, and the secret key corresponding to y is $a+t$. At the same time, $\mathcal{B}$ randomly selects $j \xleftarrow{R} \{1, 2, \cdots, q_H\}$ as the hypothetical index of the forged parameter, that is, the *j*-th query of H_r from $\mathcal{A}$ corresponds to the hash of the target plaintext M'.

(2) H_r query (at most q times), $\mathcal{B}$ creates an empty list H^{list} and let the five-tuple (h, M_i, r_i, y_i, b_i) be the element in it, which means that $\mathcal{B}$ has set $H_r(M) = h / H_{\mathbb{G}}(M_i) = r_i$. When $\mathcal{A}$ makes the *i*-th inquiry to $H_r(\cdot)$, $\mathcal{B}$ randomly selects $b_i \xleftarrow{R} Z_q^*$ and answers as follows:

·If $i = j$, return $y_i = r_i^* \cdot g^{b_i} \in \mathbb{G}$;

·Otherwise, calculate $y_i = g^{b_i} \in \mathbb{G}$.

Take y_i as the answer to the query $H_r(M_i)$, and append (h, M_i, r_i, y_i, b_i) to the list.

(3) *Adapt'* query (at most q times). In the process of $\mathcal{A}$ requesting check parameter of plaintext M' related to h, $\mathcal{B}$ let $M'=M_i$ be the *i*-th H_r query, and respond to the query in the following way:

·If $i \neq j$, then retrieval the tuple (h, M_i, r_i, y_i, b_i) in H^{list}, compute $R_i' = (u \cdot g^t)^{b_i}$, and return R_i' to $\mathcal{A}$. Because of $R_i' = (u \cdot g^t)^{b_i} = (g^{a+t})^{bi} = (g^{bi})^{a+t} = (y_i)^{a+t}$, R_i' is the check parameter constructed by r_i on M_i with secret key $a+r$.

·If $i = j$, then interrupt.

(4) $\mathcal{A}$ outputs (M'^*, R'^*).

·If $M'^* \neq M_j$, interrupt;

·Otherwise, $\mathcal{B}$ outputs $\frac{R'^*}{r^t u^{b_j} g^{b_j} g^{b_j t}}$ as $(r^*)^a$. For

$$R'^* = y_j^{(a+t)} = (r^* \cdot g^{b_j})^{(a+t)} = (r^*)^{(a+t)} \cdot g^{b_j(a+t)} = (r^*)^a r^t \cdot (g^{(a+t)})^{b_j} = (r^*)^a r^t \cdot u^{b_j} g^{b_j t}$$

If the guess i from $\mathcal{B}$ is correct and $\mathcal{A}$ finds a correct forgery, then $\mathcal{B}$ successfully solves the given CDH problem, that is, $\mathcal{B}$ finds $R'^* = (r^*)^a$ based on (g, g^a, r^*) with $\mathcal{A}$. The successful output from $\mathcal{B}$ is determined by the following 3 events:

$\mathcal{E}_1$: $\mathcal{B}$ isn't interrupted during $\mathcal{A}$'s parameter inquiry process.

$\mathcal{E}_2$: $\mathcal{A}$ produces a valid "plaintext-parameter" pair (M'^*, R'^*).

$\mathcal{E}_3$: $\mathcal{E}_2$ occurs and the subscript of M'^* in the corresponding five-tuple (h, M_i, r_i, y_i, b_i) is $i = j$.

$$\Pr[\mathcal{E}_1] = (1 - \frac{1}{q_H})^{q_H},\ \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = \omega(\mathcal{K}),\ \Pr[\mathcal{E}_3 \mid \mathcal{E}_1\mathcal{E}_2] = \Pr[i = j \mid \mathcal{E}_1\mathcal{E}_2] = \frac{1}{q_H}$$

So the advantage of $\mathcal{B}$ is:

$$\begin{aligned} \Pr[\mathcal{E}_1\mathcal{E}_3] &= \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \Pr[\mathcal{E}_3 \mid \mathcal{E}_1\mathcal{E}_2] \\ &= (1 - \frac{1}{q_H})^{q_H} \cdot \frac{1}{q_H} \cdot \omega(\mathcal{K}) \\ &\approx \frac{\omega(\mathcal{K})}{e \cdot q_H} \end{aligned}$$

Since the *CDH* problem holds on $\mathbb{G}$, the advantage $\frac{\omega(\mathcal{K})}{e \cdot q_H}$ of polynomial adversary $\mathcal{B}$ is negligible, so the chameleon hash algorithm is collision resistant. (Theorem 1 is proved)

Theorem 2 Let $\mathbb{G}$ be a gap group, and the chameleon hash function H_h is collision resistance on $\mathbb{G}$, if the *CDH* problem on $\mathbb{G}$ is difficult, the chameleon short signature is EUF-CMA.

Specifically, suppose there is an EUF-CMA adversary $\mathcal{A}$ who breaks the short signature scheme with the advantage of $\omega(\mathcal{K})$, then there must be an adversary $\mathcal{B}$ to solve the *CDH* on $\mathbb{G}$ at least by the advantage of

$$Adv_{\mathcal{B}}^{CDH} \geq \frac{\omega(\mathcal{K})}{e \cdot q_H}$$

where e is the base of the natural logarithm, q_H is the maximum number of queries to H_h.

Proof: The game process of $EUF_{\mathcal{A}}^{CDH}(\mathcal{K})$ is similar to the $CollRes_{\mathcal{A}}^{CDH}(\mathcal{K})$.

(1) Adversary $\mathcal{B}$ runs $Setup(\mathcal{K})$ and $KeyGen(Parm)$ to generate the key pair (pk, sk), selects a random function $H_{\mathbb{G}} \xleftarrow{R} \{H : \{0,1\}^* \to \mathbb{G}\}$, and send the system parameters and public key to adversary $\mathcal{A}$.

(2) Adversary $\mathcal{A}$ can request any chameleon hash query $H_h \leftarrow Hash(pk, M)$ about M, and $\mathcal{B}$ responds with (h, R) as the corresponding answer. $\mathcal{A}$ can request any signature query $Sign(\cdot)$ about h, and $\mathcal{B}$ responds with σ. During this process, q is the maximum number of times $\mathcal{A}$ queries $H_h(\cdot)$, and $M \notin \mathcal{O}$.

(3) The adversary $\mathcal{A}$ outputs a signature pair (h^*, σ^*). If $Verify(pk, h^*, \sigma^*) = 1$, then the attack from $\mathcal{A}$ is successful, in which the signature σ^* of the chameleon hash h^* has not been queried by it before.

Analysis: Knowing $\sigma = h^x$ from formula (3.4), if $\mathcal{B}$ can find a certain chameleon hash triple (h^*, M^*, R^*), such that $\sigma^* = (h^*)^x$, then σ^* is the signature of the chameleon hash h^* related to the plaintext M^*. Since adversary $\mathcal{B}$ knows $\{g,\ y = g^x,\ h = Hash(pk, M)\}$ and want to leverage $\mathcal{A}$ as a subroutine to attack the chameleon short signature, its goal is to find out $\sigma^* = (h^*)^x$. In step (3) of the game $EUF_{\mathcal{A}}^{CDH}(\mathcal{K})$, (h^*, σ^*) is generated by adversary $\mathcal{A}$, but h^* is generated by $\mathcal{B}$, so $\mathcal{B}$ can be set $h^* \leftarrow H_h(M^*) \leftarrow Hash(pk, M^*)$. When $\mathcal{B}$ lets h^* be the chameleon hash of a target "plaintext-parameter" pair (M^*, R^*), his goal is to call $\mathcal{A}$ to calculate $(h^*)^x$ based on the triple $(g,\ g^x,\ h^*)$, which is to solve the *CDH* problem. Throughout the game, $\mathcal{B}$ doesn't know which chameleon hash will be generated by $\mathcal{A}$ to forge a short signature, Therefore, he has to make a guess that the *j*-th query H_h corresponds to $\mathcal{A}$'s final forged result.

For simplicity without loss of generality, we assume that 1). Adversary $\mathcal{A}$ only initiates a plaintext chameleon hash query H_h, and doesn't initiate a adapt query; 2). Adversary $\mathcal{A}$ will not initiate the same query twice $H_h(M)$ to M; 3). Adversary $\mathcal{A}$ must have asked $h \leftarrow H_h(M)$ before requesting the signature; 4). Adversary $\mathcal{A}$ must have inquired about $h^* \leftarrow H_h(M^*)$ before outputs the signature (h^*, σ^*).

In the actual process, $\mathcal{B}$ implicitly regards $u = g^a$ in the known tuple $(g,\ u = g^a,\ h^*)$ as its own public key (in fact, $\mathcal{B}$ does not know the specific value of a), then $(h^*)^a$ is a forgery signature of the chameleon hash h^* related to (M^*, R^*), where h^* is generated by $\mathcal{B}$ Randomly, $(h^*)^a$ is forged by $\mathcal{A}$. To hide instance $u = g^a$, $\mathcal{B}$ needs to select a random number t and send $u \cdot g^t$ to $\mathcal{A}$ as the public key of $\mathcal{B}$.

The following proves the Existential-Unforgeability game $EUF_{\mathcal{A}}^{CDH}(\mathcal{K})$ of the short signature can be reduced to the *CDH* problem.

(1) $\mathcal{B}$ sends the generator g of group $\mathbb{G}$ and the public key $y = u \cdot g^t = g^{a+t} \in \mathbb{G}$ to $\mathcal{A}$, where $t \xleftarrow{R} Z_q^*$ and the secret key corresponding to y is $a + t$. At the same time, $\mathcal{B}$ randomly selects $j \xleftarrow{R} \{1, 2, \cdots, q_H\}$ as the hypothetical index of the forged signature, that is, the *j*-th query of H_h from $\mathcal{A}$ corresponds to the signature of the target hash h^*.

(2) H_h query (at most q times), $\mathcal{B}$ creates an empty list H^{list}, and let the tuple (M_i, y_i, b_i) be the element in it, which means that $\mathcal{B}$ has set $H_h(M_i) = Hash(pk, M_i) = y_i$. When $\mathcal{A}$ makes the *i*-th inquiry to $H_h(\cdot)$, $\mathcal{B}$ randomly selects $b_i \xleftarrow{R} Z_q^*$ and answers as follows:

·If $i = j$, return $y_i = (h^*) \cdot g^{b_i} \in \mathbb{G}$;

·Otherwise，calculate $y_i = g^{b_i} \in \mathbb{G}$.

Take y_i as answer to the query $H_h(M_i)$, and append (M_i, y_i, b_i) to the list.

(3) *Sign* query (at most q times). In the process of $\mathcal{A}$ requesting the signature of h related to plaintext M, $\mathcal{B}$ let $M=M_i$ be the i-th H_h query, and respond to the query in the following way:

·If $i \neq j$, then retrieval the tuple (M_i, y_i, b_i) in H^{list}, compute $\sigma = (u \cdot g^t)^{b_i}$, and return σ to $\mathcal{A}$. Because of $\sigma = (u \cdot g^t)^{b_i} = (g^{a+t})^{bi} = (g^{bi})^{a+t} = (y_i)^{a+t}$, σ is the signature constructed by the secret key $a + r$ on M_i.

·If $i = j$, then interrupt.

(4) $\mathcal{A}$ outputs (h^*, σ^*).

·if $M^* \neq M_j$ in triple (h^*, M^*, R^*), then interrupt;

·Otherwise, $\mathcal{B}$ outputs $\frac{\sigma^*}{h^t u^{b_j} g^{b_j} g^{b_j t}}$ as $(h^*)^a$. For

$$\sigma^* = y_j^{(a+t)} = (h^* \cdot g^{b_j})^{(a+t)} = (h^*)^{(a+t)} \cdot g^{b_j(a+t)} = (h^*)^a h^t \cdot u^{b_j} g^{b_j t}$$

If the guess i from $\mathcal{B}$ is correct and $\mathcal{A}$ finds a correct forgery, then $\mathcal{B}$ successfully solves the given *CDH* problem, that is, $\mathcal{B}$ finds $\sigma^* = (h^*)^a$ based on (g, g^a, h^*) with $\mathcal{A}$. Therefore, the advantage of $\mathcal{B}$ is the same as Theorem 1, which is:

$$Adv_{\mathcal{B}}^{CDH} \approx \frac{\omega(\mathcal{K})}{e \cdot q_H}$$

Since the *CDH* problem holds on $\mathbb{G}$, the advantage $\frac{\omega(\mathcal{K})}{e \cdot q_H}$ of polytime adversary $\mathcal{B}$ is negligible, so the short signature algorithm is EUF-CMA (Theorem 2 is proved).