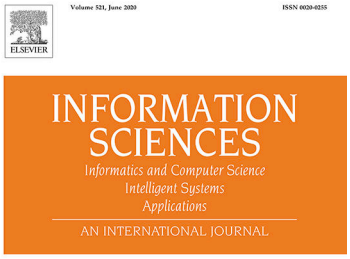
Journal Pre-proofs



Quantum Resistant Key-exposure Free Chameleon Hash and Applications in Redactable Blockchain

Chunhui Wu, Lishan Ke, Yusong Du

PII: S0020-0255(20)30995-6
DOI: https://doi.org/10.1016/j.ins.2020.10.008
Reference: INS 15922

Available online at www.sciencedirect.com
ScienceDirect

To appear in: *Information Sciences*

Received Date: 25 April 2020
Accepted Date: 7 October 2020

Please cite this article as: C. Wu, L. Ke, Y. Du, Quantum Resistant Key-exposure Free Chameleon Hash and Applications in Redactable Blockchain, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.10.008

Quantum Resistant Key-exposure Free Chameleon Hash and Applications in Redactable Blockchain

Chunhui Wu[a], Lishan Ke[b,*], Yusong Du[c,*]

[a]*School of Internet Finance and Information Engineering, Guangdong University of Finance, Guangzhou 510521, P.R. China*
[b]*School of Mathematics and Information Science, Guangzhou University, Guangzhou 510006, P.R. China*
[c]*School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, P.R. China*

Abstract

Blockchain technologies have attracted a large amount of attention recently, with immutability as a basic property. However, it is often desired to allow editing a transaction or a block in a controlled way. Chameleon hash function, with *enhanced collision-resistance* property, has recently found to be an important tool to construct redactable blockchain. This means that the traditional key-exposure free (double-trapdoor) constructions are unsuitable for the applications here. Although *single-trapdoor* key-exposure free chameleon hash functions naturally satisfy enhanced collision-resistance, they are very rare, and none is based on quantum-resistant assumptions.

In this paper, we propose two single-trapdoor key-exposure free chameleon hash functions based on lattice, without/with lattice trapdoors respectively, and show their applications in redactable blockchain. Our constructions do not need heavy cryptographic tools, such as encryption and NIZK, therefore are more compact and computational efficient than schemes following Ateniese et al.'s generic transformation framework of PKE+NIZK. Moreover, we introduce two mechanisms in order to prevent the misuse of redaction functionality in blockchain. We present a fully distributed key management mechanism for the first scheme, and solve the redaction-misuse problem which remains in blockchains using Ateniese et al.'s generic framework. We also suggest the voting strategy when applying our second scheme. Finally, we show how to efficiently integrate our chameleon hash with any blockchain technologies, with only minor changes to the current blockchains in use. For extend interests, our proposed chameleon hash functions are also suitable for constructing quantum-resistant chameleon signatures and off-line/on-line signatures.

Keywords: Chameleon hash, key-exposure, lattice-based cryptography, trapdoor function, redactable blockchain

1. Introduction

Chameleon hash function is a trapdoor collision resistant hash function. It has the same properties as a normal cryptographic hash function without the trapdoor, however, collisions can be easily computed once the trapdoor is known. Chameleon hash function has many traditional

*Corresponding author: kelishan@gzhu.edu.cn (Lishan Ke); duyusong@mail.sysu.edu.cn (Yusong Du)

applications, such as chameleon signatures [19] and off-line/on-line signatures [15]. Ateniese and de Medeiros [2] find that collision resistance is not sufficient for most of the applications of chameleon hash. The reason is that, though the chameleon hash is collision resistant, anyone seeing a collision may be able to find other collisions or even recover the trapdoor. This is known as the *key exposure* problem [3]. Researchers then propose many key-exposure free chameleon hash functions which are more suitable for applications in chameleon signatures [8, 10] and off-line/on-line signatures [9, 24].

In 2017, Ateniese et al. introduced a stronger notion of *enhanced collision-resistance* for chameleon hash functions and applied them to realize redactable blockchain [4]. Blockchain technology has now attracted a tremendous of attention [11], and redaction is a useful property in many scenarios [26]. The first large-scale application of blockchain is the decentralized cryptocurrency Bitcoin. Meanwhile, applications have gone far beyond cryptocurrencies, such as supply chains, copyright protection, logistics chains, asset digitization, healthcare et al., which are defined by different "smart contacts". Moreover, the chain's participants can reach an agreement by the consensus protocol. Each block is linked in the chain by including the hash of the previous block in it, additionally, valid transactions (such as monetary transactions in cryptocurrencies) or some other data (which are of interest and need to be recorded) are also included by means of a Merkle tree to accumulate them into a single hash.

1.1. Problem and Motivation

While immutability is a crucial property of blockchain, it is often desirable to redact its content in some specific conditions, such as removing improper or sensitive information from the blockchain, or updating a smart contact et al. [4]. It is even legally obliged for the blockchain to be editable. For example, "the right to be forgotten" is a key right of people imposed by the EU in the General Data Protection Regulation (GDPR)[1]. Under this regulation, it is illegal to use immutable blockchain where personal data are recorded. Various other legal regulations can be found in the U.S. Security and Exchange Commission's (SEC) Regulation S-P and Gramm-Leach-Bliley Act. Of course, the redaction should be under strict constraints.

Since redaction is becoming an important requirement and even be legally obliged in blockchains, how to realize this functionality in a secure, simple and efficient way is a critical issue. It is obvious that *hard fork* is not an option for this purpose. Fortunately, chameleon hash function is found to be a suitable tool recently. The Accenture Cooperation has applied a patent to use chameleon hash in building a redactable blockchain [1]. Though there are many key-exposure free chameleon hash functions based on various standard number-theoretic assumptions such as discrete log and factoring, they are insufficient for constructing redactable blockchain. Because a collision will leak the one-time trapdoor (though the long-term trapdoor is still kept secret), which allows one to compute other collisions, i.e., anyone can edit the block again after seeing a redaction. Ateniese et al. [4] introduce a new notion of *enhanced collision-resistance*, which is suitable for the application of redactable blockchain. After Ateniese et al.'s work, several chameleon hash functions are proposed in the enhanced collision resistance model [12, 18]. But no one can be resistant to quantum attacks. Since chameleon hash function is a fundamental tool in constructing redactable blockchain, it is crucial to propose efficient constructions based on quantum resistant assumptions, such as lattice-based assumptions.

[1] https://gdpr-info.eu/

Another key problem is the redaction-misuse in blockchain. The redaction functionality should not be misused without the agreement by most participants, so the chameleon trapdoor should be protected well. In other words, it cannot be leaked (as enhanced collision resistance ensured) and may also need to be shared in a distributed manner. Ateniese et al. [4] propose a generic transformation framework from key-exposure chameleon hashes to schemes satisfying enhanced collision resistance, based on a CPA-PKE and a tSE NIZK (true-Simulation Extractable Non-Interactive Zero-Knowledge) scheme. The PKE scheme is used to hide the random number r in chameleon hash to prevent key-exposure, i.e., the chameleon hash outputs $(h, (c, \pi))$ instead of (h, r), where c is an encryption of r. However, r has to be decrypted to compute a collision r', and then to compute its encryption c' and the corresponding NIZK π'. This process is centralized and exposes the collision r and r', and leads to the leakage of the trapdoor. So any participants can redact the block in any way.[2] Ateniese et al. [4] propose a distributed key management mechanism for their single-trapdoor key-exposure free chameleon hash to prevent redaction-misuse. However, it seems that the generic transformation and its two instantiations cannot support distributed key management and are unsuitable for blockchain applications. The same problem exists in the following works using this framework, such as [18]. Though a collision r' can be computed without decrypting r, r' will be exposed to the prover (redactor), who can compute the trapdoor and redact the block arbitrarily.

1.2. Related Works

Some other ways to realize redactable blockchain without chameleon hash functions are also proposed. Deuber et al. [13] use a consensus-based voting to decide whether the redaction should be permitted, similar to the idea in [20], and replace the block by "breaking" the chain. However, the old block should still be kept there in order to validate the chain, while the information was supposed to be rewritten. Therefore, the old information can not really be "forgotten". The same shortcoming occurs in Xu et al. [25], where a redactable blockchain in the permissionless setting is also proposed, with similar ideas as [13]. It is compatible with the proof-of-stake consensus protocol and the redaction voting is faster than [13]. Puddu et al. [23] construct a mutable blockchain in a different way. The sender prepares and submits a transaction set in order to make his transaction mutable. Only one transaction in the set is *activated*, while the other *inactive* transaction versions can be used as replacements later. In order to hide alternative transaction data, the inactive transactions are encrypted and the decryption keys are shared among the miners. The sender establishes a mutable policy for his transaction, which sets out how and by whom his transaction is allowed to be mutated. When a mutate request is broadcasted, the miners run a multi-party computation (MPC) protocol to reconstruct the decryption key and decrypt the appropriate version of the transaction, which becomes the new *active* transaction. However, the mutations have to be prepared beforehand and the mutation process needs the honest participation from the sender.

So key-exposure free chameleon hash function is the best choice to build redactable blockchain, in order to satisfy the "right to be forgotten". Meanwhile, this method can be compatible with all kinds of consensus protocols, and requires minimal changes to the blockchains in use. There are

[2]The basic chameleon hash they used is based on Pederson commitment. It is a key-exposure one, not a double-trapdoor scheme. So the long-term trapdoor will be exposed, and all blocks can be redacted in any way in this case.

many double-trapdoor key-exposure free chameleon hash functions available, constructed based on various number-theoretic assumptions, since the key-exposure problem addressed by Ateniese and de Medeiros [3]. They are usually constructed by the idea of tags, and contain two trapdoors, namely the long-term trapdoor and one-time trapdoor. The collision only leaks the one-time trapdoor, which can be seen as a signature on a one-time tag, while keeping the long-term trapdoor secret.

These double-trapdoor key-exposure free chameleon hash functions are suitable for applications in chameleon signatures and off-line/on-line signatures. When used in chameleon signatures, the key-exposure freeness can guarantee the non-transferability. Moreover, one-time trapdoor key leakage is also a useful property to keep the original signed message secret in the *denial protocol* of chameleon signature, where the judge can be convinced by the signer that some message m' is forged by given another message $m'' \neq m'$, without revealing the original signed message m (also know as message-hiding [2]). The reason is that, the signer cannot provide a different message m'' unless m' is a forgery, since it does not have the trapdoor, which is only known to the verifier. If m' is a forgery, the signer can compute the one-time trapdoor from the pair of collisions, i.e. m and m', and then compute a third collision m''. Using key-exposure free chameleon hash in off-line/on-line signatures can help saving the storage and computation cost because the chameleon hash and its corresponding signature computed off-line can be reused. Note that the one-time key will not be exposed here and other collisions cannot be computed (collisions means signature forgery and are not allowed in off-line/on-line signatures), because the collision pair will never be revealed, with one part of the collision computed off-line and kept secret.

However, double-trapdoor key-exposure free chameleon hash functions are useless in redactable blockchain. Fortunately, there is a special kind of key-exposure free chameleon hash, which contains one single trapdoor, satisfies the *enhanced collision-resistance* property defined by Ateniese et al. in [4]. But only few constructions of single trapdoor key-exposure free chameleon hash are known. One example is the scheme in [3] and recently used in redactable blockchain [4]. Moreover, Ateniese et al. propose a generic transformation from traditional collision resistant chameleon hash to enhanced collision resistant chameleon hash. The transformation uses a CPA-secure PKE to conceal the random number in chameleon hash, and a tSE NIZK to prove it. When instantiating, they use a standard NIZK and a CCA-secure PKE scheme to realize the tSE NIZK. The transformation framework is a little cumbersome. Most importantly, it cannot support distributed key management to prevent redaction-misuse. Derler et al. further propose an attribute-based access control method to determine who can modify the block [12]. An attribute-based encryption is used to encrypt the one-time trapdoor in the chameleon hash. Note that the chameleon hash is a special one, named *chameleon hash with ephemeral trapdoor* proposed by Camenisch and Derler et al. [6]. It requires not only the long-term trapdoor, but also one-time trapdoor to compute collisions, so the attribute-based access control is forced. However, the scheme in [12] does not include a NIZK, so it cannot guarantee that the ABE encrypts a valid one-time trapdoor of the chameleon hash, and it cannot prevent dishonest participants from uploading invalid transactions which cannot be rewritten. Li et al. [20] introduce a multi-party joint decision making method to redact blockchain, in order to prevent misuse of redaction functionality. Every participant in the alliance chain votes to decide whether the redaction should be performed, and then a special consensus mechanism is used to choose one user to execute the redaction. Khalili et al. [18] improve Ateniese et al.'s construction [4] with better efficiency, follow the same framework and cannot support distributed key management.

To the best of our knowledge, there are only few single-trapdoor key-exposure free chameleon hash functions exist [3, 12, 18]. So how to design such chameleon hash functions based on different assumptions is an important issue, especially constructions based on quantum resistant assumptions. Moreover, the existing schemes are not efficient and compact, which normally need encryption and NIZK. As described above, Ateniese et al.'s generic transformation [4] from key exposure chameleon hash needs CPA PKE+NIZK+CCA PKE, Derler et al.'s policy-based chameleon hash [12] needs 2 single-trapdoor key-exposure free chameleon hash+ABE[3]. Furthermore, Ateniese et al.'s NIZK-based generic transformation (also used in [18]) cannot support distributed key management, which is an important mechanism to prevent redaction-misuse.

1.3. Techniques and Our Contributions

In this paper, we construct two single-trapdoor key-exposure free chameleon hash schemes based on lattice without/with lattice trapdoors respectively, and use them to build redactable blockchains. Moreover, in order to prevent the misuse of redact functionality, with the support of a secure multi-party sum protocol we propose a distributed trapdoor sharing method to compute collisions in the first scheme, and combine the voting strategy as [13, 20, 25] with the second scheme, so the redaction functionality can be integrated with all types of blockchains easily, no matter they are private blockchain, consortium blockchain, or public blockchain.

A Chameleon Hash based on Lattice without Trapdoors. Only few constructions of single-trapdoor key-exposure free chameleon hash functions are known even under number-theoretic assumptions, since collisions will normally leak some information of the trapdoor. The tag-based technique is unsuitable for redactable blockchain applications here, though it is widely used in chameleon signatures and off-line/on-line signatures to prevent key-exposure problem. So we need to find other ideas to construct single-trapdoor key-exposure free chameleon hash.

We observe the similarities between collision computing and signature computing, and between hash collisions and twin signatures. Based on these observations, we combine the ideas of twin signature framework [22] and lattice signatures in [14], which implement a more efficient reject sampling scheme than [21], to construct an efficient chameleon hash. The enhanced collision resistance can be proved based on the twin signature framework, and computing a collision corresponds to computing a lattice signature, which requires the trapdoor.

A Chameleon Hash based on Lattice with Trapdoors. Many discrete logarithm related chameleon hash functions are constructed based on Pedersen commitment. However, Pedersen commitment has the key-exposure problem. In lattice-based environment, if we construct a chameleon hash using similar structure as Pedersen commitment, i.e., $\mathbf{h} = \mathbf{Ar} + \mathbf{Tc} \mod q$, where $\mathbf{T} = \mathbf{AX}$ and $\mathbf{c} = H(\mu)$, and set $\mathbf{X}$ as the trapdoor. Then the trapdoor $\mathbf{X}$ may be leaked if enough pairs of collisions $(\mu_i, \mathbf{r}_i)$ and $(\mu_i', \mathbf{r}_i')$ are revealed, say not less than n collisions, i.e., $\mathbf{Ar}_i + \mathbf{Tc}_i = \mathbf{Ar}_i' + \mathbf{Tc}_i'$ and $\mathbf{AX}(\mathbf{c}_i - \mathbf{c}_i') = \mathbf{A}(\mathbf{r}_i' - \mathbf{r}_i)$ with $i = 1, 2, \ldots, n$. Then $\mathbf{X}$ could be determined by solving the linear system $\mathbf{XC} = \mathbf{R}$, where $\mathbf{C} = [\mathbf{c}_1 - \mathbf{c}_1', \mathbf{c}_2 - \mathbf{c}_2', \ldots, \mathbf{c}_n - \mathbf{c}_n']$ and $\mathbf{R} = [\mathbf{r}_1' - \mathbf{r}_1, \mathbf{r}_2' - \mathbf{r}_2, \ldots, \mathbf{r}_n' - \mathbf{r}_n]$.

Existing solutions, such as tag-based techniques, are widely used to avoid the key-exposure problem of Pedersen commitment. But it is unsuitable for the applications here. Fortunately, the trapdoor for lattice comes to rescue. The short basis $\mathbf{S}$ can be set as the trapdoor of the chameleon

[3]NIZK is not included for efficiency considerations, but may cause some problems.

Table 1: Comparison of chameleon hash functions in the Enhanced Collison Resistant (ECR) model.

Schemes	Security Model[1]	Assumptions[2]	Dis. Key Management	Hash Computation[4]	Collision Computation[4]	Random Size[6]
[3]	GGM	–	Yes	$2E_{\mathbb{Z}_p^*}$	$1E_{\mathbb{Z}_p^*}$	$3\|\mathbb{Z}_q\|$
[4, §4.4.2]	ROM	DDH	No	$17E_{\mathbb{G}}$	Inefficient inversion[5]	$12\|\mathbb{G}\| + 7\|\mathbb{Z}_p\|$
[4, §4.4.3]	SM	SXDH	No	$51E_{\mathbb{G}_1}$	Inefficient inversion[5]	$6\|\mathbb{G}_1\| + 12\|\mathbb{G}_2\|$
[18, §5]	SM	SXDH	No	$25E_{\mathbb{G}_1}$	$1E_{\mathbb{G}_1}$	$8\|\mathbb{G}_1\| + 4\|\mathbb{G}_2\|$
[18, §6]	SM	PKE	No	$6E_{\mathbb{G}_1}$	$1E_{\mathbb{G}_1}$	$2\|\mathbb{G}_1\|$
Our scheme 1	GGM	ISIS	Yes	$1M$	$2M$	$\|\mathbb{Z}_{2q}^n\| + \|D_\sigma^m\|$
Our scheme 2	ROM	SIS	–[3]	$2M$	$1M + O(n^2)$	$\|D_\sigma^m\|$

[1] ROM denotes the Random Oracle Model, SM denotes the Standard Model, and GGM denotes the Generic Group Model.
[2] PKE denotes for Power Knowledge of Exponent here.
[3] A distributed SampleD algorithm is needed to support distributed key management in our second scheme.
[4] $E_{\mathbb{Z}_p^*}$ and $E_{\mathbb{G}}$ denote the exponentiation in $\mathbb{Z}_p^*$ and $\mathbb{G}$ respectively, and M denotes the matrix-vector multiplication.
[5] The invertible map $\Omega : \mathbb{Z}_p \to \mathbb{G}$ used in encryption is inefficient, as also stated in [18].
[6] By abuse of notation, we use $|D_\sigma^m|$ to denote the size of an element sampled from D_σ^m, which is less than $m \log(12\sigma)$ with overwhelming probability.

hash. In the chameleon hash $\mathbf{h} = \mathbf{Ar} + \mathbf{Tc} = \mathbf{A}(\mathbf{r} + \mathbf{Xc})$, the distribution of $\mathbf{r} + \mathbf{Xc}$ can be tailored to D_σ^m by using reject sampling, and $\mathbf{h}$ can be uniform in $\mathbb{Z}_q^n$. So computing a collision $\mathbf{r}'$ satisfying $\mathbf{Ar}' = \mathbf{h} - \mathbf{Tc}' \mod q$ and $||\mathbf{r}'|| \leq \beta$ is equivalent to the ISIS problem. Moreover, the trapdoor basis $\mathbf{S}$ will not be leaked by chameleon hash collisions, i.e., by the SIS instances $(\mathbf{r} - \mathbf{r}' + \mathbf{Xc} - \mathbf{Xc}')$ from $\mathbf{Ar} + \mathbf{Tc} = \mathbf{Ar}' + \mathbf{Tc}'$, so a third collision $\mathbf{r}''$ cannot be computed, and the enhanced collision resistance is guaranteed.

Mechanisms to Prevent the Misuse of Redaction Functionality. We introduce two mechanisms to control the behavior of redaction, the first one is the threshold secret sharing method, which provides indistinguishable redaction and is also used in [4, 23]; the second one is the consensus-based voting strategy, which naturally provides the property of public accountability and is also used previously in [13, 20, 25]. We propose an efficient distributed key generation algorithm, by separating the trapdoor into column vectors, and use a secure multi-party sum protocol to compute chameleon hash collisions. For the redaction voting mechanism, we compare different voting strategies. Deuber et al. [13] use lightweight collision-resistant hash to vote, while it requires a long voting period. Xu et al. [25] and Li et al. [20] use heavier tools as signatures (multi-signatures), and the voting can be done within one block. However, the redaction is realized by directly "breaking" the chain in [13, 25], and the chameleon hash function used in [20] is not quantum-resistant and has the key-exposure problem. We combine our chameleon hash functions with these voting mechanisms and avoid these problems.

We give a comparison of our proposed two quantum-resistant chameleon hashes with previous traditional constructions in the enhanced collision resistant (ECR) model in Table 1. Note that schemes using Ateniese et al.'s NIZK-based generic transformation cannot support distributed key management.

2. Preliminaries

2.1. Notations

In this paper, we assume that q is a small prime number and elements in $\mathbb{Z}_q$ are integers in the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$. Given a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we can define two full-rank m-dimensional integer lattices. The first one is

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{0} \mod q\},$$

which consists of integer vectors that are "orthogonal" (mod q) to the rows of $\mathbf{A}$. The second one is

$$\Lambda(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{A}^T\mathbf{x} \mod q \text{ for } \mathbf{x} \in \mathbb{Z}^n\},$$

which is generated by the (transposed) rows of $\mathbf{A}$.

Reject Sampling. Reject sampling is a fundamental building block in lattice-based schemes [21]. It can sample from an arbitrary probability distribution f, given a different source probability distribution g. The method works as follows. It accepts a sample x from source distribution g with probability $f(x)/(M \cdot g(x))$, where M is some constant. The process is restarted if it is not accepted. It is proved that the reject sampling produces exactly the distribution of f, if $f(x) \leq M \cdot g(x)$ for all x. Moreover, since the expected number of times for the procedure to restart is M, it is important to make M as small as possible. Ducas et al. [14] propose an improved reject sampling by tailoring the function g as bimodal Gaussians, such that it resembles the target function f much better.

Discrete Gaussian Distribution. The continuous Gaussian distribution over $\mathbb{R}$ with center $c \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}$ is defined as $\rho_{c,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-(x-c)^2}{2\sigma^2})$ for $x \in \mathbb{R}$, and more generally over $\mathbb{R}^m$ as $\rho^m_{\mathbf{c},\sigma}(\mathbf{x}) = (\frac{1}{\sqrt{2\pi\sigma^2}})^m \exp(\frac{-||\mathbf{x}-\mathbf{c}||^2}{2\sigma^2})$ for $\mathbf{x}, \mathbf{c} \in \mathbb{R}^m$. If the center $\mathbf{c} = \mathbf{0}$, we will omit it and simply write $\rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution over $\mathbb{Z}$ centered at c is defined as $D_{c,\sigma}(x) = \rho_{c,\sigma}(x)/\rho_\sigma(\mathbb{Z})$, and more generally, over $\mathbb{Z}^m$ as $D^m_{\mathbf{c},\sigma}(\mathbf{x}) = \rho^m_{\mathbf{c},\sigma}(\mathbf{x})/\rho^m_\sigma(\mathbb{Z}^m)$.

Notice that the quantity $\rho^m_\sigma(\mathbb{Z}^m) = \sum_{\mathbf{z}\in\mathbb{Z}^m} \rho^m_\sigma(\mathbf{z})$ is just a scaling quantity so as to make the function into a probability distribution. Also notice that the scaling factor is the same for all $\mathbf{v}$, i.e., $\rho^m_{\mathbf{v},\sigma}(\mathbb{Z}^m) = \rho^m_\sigma(\mathbb{Z}^m)$.

2.2. Definitions

Definition 1. *(**Chameleon Hash Function with Enhanced Collision Resistance**) A chameleon hash function is formalized by the following three PPT algorithms:*

- $\mathsf{KeyGen}(\lambda)$*: on input the security parameter* $\lambda \in \mathbb{N}$*, the algorithm generates the hash/trapdoor key pair* (HK, TK)*.*
- $\mathsf{CH}(HK, \mu, r)$*: on input a hash key* HK*, a message* μ*, and an auxiliary random number* r*, outputs a hash value* h *with fixed length.*
- $\mathsf{UF}(TK, \mu, r)$*: on input the trapdoor key* TK*, a message* μ*, and an auxiliary random number* r*, the Universal Forge algorithm outputs a second message* μ' *along with an auxiliary parameter* r'*, s.t.*

$$\mathsf{CH}(HK, \mu', r') = \mathsf{CH}(HK, \mu, r).$$

The chameleon hash function satisfies the following properties:

Enhanced Collision Resistance. This property is introduced in [4], intuitively, it means that it is hard to find collisions for the chameleon hash function given access to the collision finding oracle $\mathsf{UForgeO}$. Formally, the following probability is negligible for all PPT adversary $\mathcal{A}$:

$$Pr\left[\begin{array}{c|l} \mathsf{CH}(HK,\mu,r)=\mathsf{CH}(HK,\mu',r')=h & (h,(\mu,r),(\mu',r')) \xleftarrow{\$} \mathcal{A}^{\mathsf{UForgO}(\cdot)}(HK) \\ \wedge(\mu\neq\mu')\wedge(h\notin\mathcal{Q}) & (HK,TK)\leftarrow \mathsf{KeyGen}(\lambda) \end{array}\right]$$

where $\mathcal{Q}$ is the set of all hash values queried by $\mathcal{A}$ to its oracle, and oracle UForgO returns r' such that $CH(HK,\mu',r')=h$ upon input $((h,\mu,r),\mu')$ satisfying $CH(HK,\mu,r)=h$.

Notice that enhanced collision resistance implies collision resistance, which is required in normal chameleon hash functions.

Semantic Security. Nothing is revealed for the underlying message μ given its corresponding chameleon hash value h. Formally, information-theoretic semantic security states that the entropy $H[\mu|h]$ of the message conditioned on its chameleon hash equals the total entropy $H[\mu]$ of the message space.

2.3. Hardness Assumptions

The security of our two constructions are based on the SIS (Small Integer Solution) and ISIS (Inhomogeneous Small Integer Solution) problems.

Definition 2. *($\mathsf{SIS}_{q,n,m,\beta}$ **problem**) Given a uniformly random matrix $\mathbf{A}\xleftarrow{\$}\mathbb{Z}_q^{n\times m}$, an integer q, and a real β, find a non-zero vector $\mathbf{v}\in\mathbb{Z}_q^m$ such that $\mathbf{Av}=\mathbf{0} \mod q$ and $||\mathbf{v}||\leq\beta$.*

Definition 3. *($\mathsf{ISIS}_{q,n,m,\beta}$ **problem**) Given a uniformly random matrix $\mathbf{A}\xleftarrow{\$}\mathbb{Z}_q^{n\times m}$, an integer q, a real β, and a random syndrome $\boldsymbol{u}\xleftarrow{\$}\mathbb{Z}_q^n$, find a vector $\mathbf{v}\in\mathbb{Z}_q^m$ such that $\mathbf{Av}=\mathbf{u} \mod q$ and $||\mathbf{v}||\leq\beta$.*

In order to ensure that there exists a solution $\mathbf{v}$, the parameters q,n,m,β should satisfy $\beta\geq\sqrt{m}q^{n/m}$ and $m\geq 2n\log q$.

3. A Construction based on Lattice without Trapdoors

In this section, we integrate the ideas of Ducas et al.'s improved rejection sampling algorithm [14] and the twin signature framework [22], and construct an efficient single-trapdoor key-exposure free chameleon hash. Then we show how to share the trapdoor in order to prevent the misuse of redaction functionality in blockchain. Finally we analyse the security of our proposed scheme.

3.1. The Scheme

- KeyGen: Let $H:\{0,1\}^*\rightarrow\{\mathbf{v}:\mathbf{v}\in\{0,1\}^n,||\mathbf{v}||_1\leq\kappa\}$ be a cryptographic hash function. The long-term trapdoor key is a matrix $TK=\mathbf{X}\in\mathbb{Z}_{2q}^{m\times n}$ with small coefficients and the hash key is the matrix $HK=\mathbf{A}\in\mathbb{Z}_{2q}^{n\times m}$ such that $\mathbf{AX}=q\mathbf{I}_n \mod 2q$. Notice that the key pair satisfies an important property that $\mathbf{AX}=\mathbf{A}(-\mathbf{X})=q\mathbf{I}_n \mod 2q$. Such key pair can be generated easily and efficiently, refer [14] for details.

- CH: On input hash key HK, select $\mathbf{y} \in \mathbb{Z}_{2q}^n$ uniformly at random, sample $\mathbf{s} \xleftarrow{\$} D_\sigma^m$, and compute the chameleon hash of a message μ as:

$$\mathbf{h} = \mathbf{y} + \mathbf{A}\mathbf{s} + q\mathbf{c} \mod 2q \tag{1}$$

  where $\mathbf{c} = H(\mathbf{y}, \mu)$.

- UF: On input the long-term trapdoor key $TK = \mathbf{X}$, a message μ', sample $\mathbf{k} \xleftarrow{\$} D_\sigma^m$ and a bit $b \xleftarrow{\$} \{0, 1\}$, compute:

$$\mathbf{y}' = (\mathbf{h} + \mathbf{A}\mathbf{k}) \bmod 2q, \quad \mathbf{s}' = -\mathbf{k} + (-1)^b \mathbf{X}\mathbf{c}'.$$

  where $\mathbf{c}' = H(\mathbf{y}', \mu')$. Output $(\mathbf{y}', \mathbf{s}')$ with probability $1/(M \exp(\frac{-||\mathbf{X}\mathbf{c}'||^2}{2\sigma^2}) \cosh(\frac{\langle \mathbf{s}', \mathbf{X}\mathbf{c}' \rangle}{\sigma^2}))$, otherwise restart the UF procedure.

  For $(\mu', \mathbf{y}', \mathbf{s}')$ to be a valid collision, it is required that $||\mathbf{s}'|| \leq \eta\sigma\sqrt{m}$.

  It is obvious that $(\mu', \mathbf{y}', \mathbf{s}')$ and $(\mu, \mathbf{y}, \mathbf{s})$ are a pair of collisions because

$$\begin{aligned} \mathbf{y}' + \mathbf{A}\mathbf{s}' + q\mathbf{c}' &= \mathbf{h} + \mathbf{A}\mathbf{k} + \mathbf{A}(-\mathbf{k} + (-1)^b \mathbf{X}\mathbf{c}') + q\mathbf{c}' \\ &= \mathbf{h} + (-1)^b \mathbf{A}\mathbf{X}\mathbf{c}' + q\mathbf{c}' \\ &= \mathbf{h} + 2q\mathbf{c}' \\ &= \mathbf{h} \mod 2q \end{aligned}$$

3.2. Sharing the Trapdoor

In order to prevent the misuse of redaction functionality, we propose a method to share the trapdoor, in the key generation and universal forge algorithms in the semi-honest setting.

DisKeyGen: the distributed key generation algorithm is described as follows:

- all participants in the set $\mathcal{U}$ ($|\mathcal{U}| \leq n$) together choose a matrix $\mathbf{A}' \in \mathbb{Z}_{2q}^{n \times (m-n)}$ uniformly at random, written as

$$\mathbf{A}' = \begin{bmatrix} \mathbf{a}_1'^T \\ \mathbf{a}_2'^T \\ \vdots \\ \mathbf{a}_n'^T \end{bmatrix}_{n \times (m-n)}$$

  where $\mathbf{a}_i'^T \in \mathbb{Z}_{2q}^{1 \times (m-n)}, i = 1, \cdots, n$ is the row vector of $\mathbf{A}'$.

- each participant P_j chooses a uniformly random *column* vector $\mathbf{x}_j' = [x_{1j}' \; x_{2j}' \; \cdots \; x_{(m-n)j}']^T \in \mathbb{Z}_{2q}^{(m-n) \times 1}, j = 1, \cdots, n$ of matrix $\mathbf{X}' \in \mathbb{Z}_{2q}^{(m-n) \times n}$ with small coefficients, i.e., $\mathbf{X}' = [\mathbf{x}_1' \; \mathbf{x}_2' \; \cdots \; \mathbf{x}_n']$. Notice that $\mathbf{X}'$ is never revealed, and so with the secret key

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}' \\ -q\mathbf{I} \end{bmatrix}_{m \times n} . = \begin{bmatrix} \mathbf{x}_1' & \mathbf{x}_2' & \cdots & \mathbf{x}_n' \\ & & -q\mathbf{I} & \end{bmatrix}_{m \times n} = \begin{bmatrix} x_{11}' & x_{12}' & \cdots & x_{1n}' \\ \cdots & \cdots & \cdots & \cdots \\ x_{(m-n)1}' & x_{(m-n)2}' & \cdots & x_{(m-n)n}' \\ \ddots & & & \\ & & -q\mathbf{I} & \end{bmatrix}_{m \times n}$$

- each participant P_j computes and broadcasts $a''_{ij} = \mathbf{a}_i'^T \mathbf{x}'_j \mod 2q$, for $i = 1, \cdots, n$, then all participants in the set $\mathcal{U}$ recover

$$\mathbf{A}'' = \begin{bmatrix} a''_{11} & a''_{12} & \cdots & a''_{1n} \\ a''_{21} & a''_{22} & \cdots & a''_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a''_{n1} & a''_{n2} & \cdots & a''_{nn} \end{bmatrix}_{n\times n} = \mathbf{A}'\mathbf{X}',$$

  and output the public key $\mathbf{A} = [2\mathbf{A}'|2\mathbf{A}'' + q\mathbf{I}]$.

DisUF: the collision can be computed using a MPC protocol with distributed secret key.

- all participants in the set $\mathcal{U}$ together sample $\mathbf{k} \xleftarrow{\$} D_\sigma^m$, and compute $\mathbf{y}' = \mathbf{h} + \mathbf{A}\mathbf{k}$ and $\mathbf{c}' = [c'_1\ c'_2\ \cdots\ c'_n]^T = H(\mathbf{y}', \mu') \in \{0,1\}^n$ as the scheme presented in Section 3.1.
- all participants engage in a semi-honest MPC protocol for computing $\mathbf{s}' = -\mathbf{k} + (-1)^b \mathbf{X}\mathbf{c}'$. Specifically, each participant P_j contributes a column of $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ along with the coefficient c'_j, i.e., $c'_j\mathbf{x}_j$ with $j = 1, 2, \ldots, n$. Then by using a semi-honest secure multi-party sum protocol, all participants together compute

$$\mathbf{X}\mathbf{c}' = \sum_{j=1}^{n} c'_j \mathbf{x}_j,$$

  but do not reveal $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ to each other.

Discussion. Sharing the trapdoor in our scheme relies on a semi-honest secure multi-party sum protocol. For simplicity and for the quantum-resistance, we may use a multi-party unconditionally secure protocol [5, 7], which is information-theoretically secure as long as the number of users participating in the collusion attack is less than a half of total number of users.

3.3. Security Analysis

Theorem 1. *The proposed single-trapdoor chameleon hash function in Section 3.1 is a secure chameleon hash function with enhanced collision resistance and semantic security.*

The theorem is proved based on the following two lemmas.

Lemma 1. *The proposed single-trapdoor chameleon hash function in Section 3.1 satisfies enhanced collision resistance based on the* $\mathsf{ISIS}_{q,n,m,\beta}$ *assumption for* $\beta = \eta\sigma\sqrt{m}$ *(*$\eta \in [1.1, 1.4]$ *in practice).*

Proof. (sketch). Observe that the UF algorithm in our chameleon hash scheme has a similar form to the signing process of Nyberg-Rueppel signature [22, Appendix A], the difference is that we work in the lattice setting. Since the security proof of the twin Nyberg-Rueppel signature works in the generic computation model, the encoding $\sigma(x) = g^x \mod q$ can be replaced with $\sigma(\mathbf{x}) = \mathbf{A}\mathbf{x} \mod 2q$, $\mathbf{x} \xleftarrow{\$} D_\sigma^m$ based on lattice here. The $\mathsf{ISIS}_{q,n,m,\beta}$ assumption ensures that the encoding $\sigma(\mathbf{x})$ satisfies the framework of twin Nyberg-Rueppel signature. Notice that computing a collision means forging a twin Nyberg-Rueppel signature on message $\mathbf{h}$, so the proof of enhanced collision resistance can follow the existential unforgeability under adaptive chosen message attacks of twin

Nyberg-Rueppel signature (refer to [22, Appendix A] for the detailed proof), where the UForgeO oracle can be simulated as the signing oracle.

Another difference from the twin Nyberg-Rueppel signature is that, we substitute $\mathbf{c}' = H(\mathbf{y}', \mu')$ for $\mathbf{y}'$, and compute $\mathbf{Xc}'$ instead of $\mathbf{Xy}'$. The proof is formally the same, only with the difference that the collisions are found in the image of $H(\cdot)$ instead of in the whole ring $\mathbb{Z}_{2q}$. Therefore the hash function $H(\cdot)$ does not need to be programmed as a random oracle, but only need to be collision-resistant. □

Lemma 2. *The proposed single-trapdoor chameleon hash function in Section 3.1 satisfies semantic security.*

Proof. Recall that the chameleon hash is computed as $\mathbf{h} = \mathbf{y} + \mathbf{As} + q\mathbf{c} \mod 2q$, where $\mathbf{y}$ is a uniformly random vector from $\mathbb{Z}_{2q}^n$ and $\mathbf{c} = H(\mathbf{y}, \mu) \in \{0,1\}^n$. Notice that for $\mathbf{s} \xleftarrow{\$} D_\sigma^m$, the distribution of the syndrome $\mathbf{As} \mod 2q$ is statistically close to uniform over $\mathbb{Z}_{2q}^n$ [17, 14]. So the distribution of $\mathbf{y}+\mathbf{As}+q\mathbf{c} \mod 2q$ is statistically close to uniform over $\mathbb{Z}_{2q}^n$, Then $\mathbf{h}$ is independent of message μ, and our proposed chameleon hash satisfies semantic security. □

4. A Construction based on Lattice with Trapdoors

In this section, we propose another construction of single-trapdoor key-exposure free chameleon hash function, based on a similar structure as Pedersen commitment. However, the construction is not trivial, since Pedersen commitment has the key-exposure problem. We set the trapdoor using the technique of trapdoor lattice instead.

4.1. The Scheme

- KeyGen: Using the idea of Gentry, Peikert and Vaikuntanathan [17] to choose $(\mathbf{A}, \mathbf{S})$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ is statistically close to uniform and $\mathbf{S} \subset \Lambda^\perp(\mathbf{A})$ is a good basis with $||\tilde{\mathbf{S}}|| \leq L = m^{2.5}$. Here $\tilde{\mathbf{S}}$ denotes Gram-Schmidt orthogonalization of $\mathbf{S}$ and the bound L on $||\tilde{\mathbf{S}}||$ can be improved to $L = m^{1+\epsilon}$ for any $\epsilon > 0$. Let $\mathbf{X} \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ be a matrix with small coefficients and $\mathbf{T} = \mathbf{AX}$. Let $H : \{0,1\}^* \to \{\mathbf{v} : \mathbf{v} \in \{-1, 0, 1\}^n, ||\mathbf{v}||_1 \leq \kappa\}$ be a cryptographic hash function. Output the hash key $HK = (\mathbf{A}, \mathbf{T})$ and the trapdoor key $TK = \mathbf{S}$.

- CH: On input hash key HK, sample $\mathbf{r} \xleftarrow{\$} D_\sigma^m$, compute the chameleon hash as:

$$\mathbf{h} = \mathbf{Ar} + \mathbf{Tc} \mod q \tag{2}$$

  where $\mathbf{c} = H(\mu)$. Output $\mathbf{h}$ with probability $1/(M \exp(\frac{-||\mathbf{Xc}||^2 + 2\langle \mathbf{r}, \mathbf{Xc}\rangle}{2\sigma^2}))$.

- UF: On input the long-term trapdoor key TK, a message μ'. First, compute via linear algebra an arbitrary $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_q^m$ such that $\mathbf{At} = \mathbf{h} - \mathbf{Tc}' \mod q$, where $\mathbf{c}' = H(\mu')$. Then sample $\mathbf{v} \xleftarrow{\$} D_{-\mathbf{t},\sigma}^{\Lambda^\perp}$ using SampleD$(\mathbf{S}, \sigma, -\mathbf{t})$ and output $\mathbf{r}' = \mathbf{t} + \mathbf{v}$. The SampleD algorithm can be referred to [17, §4.2].

It is obvious that $(\mu', \mathbf{r}')$ and $(\mu, \mathbf{r})$ are a pair of collisions.

Discussion. Our scheme has a similar form to Pedersen commitment, which has the key-exposure problem. If $\mathbf{X}$ is used as the secret key it may be leaked when enough pairs of collisions $(\mu_i, \mathbf{r}_i)$ and $(\mu_i', \mathbf{r}_i')$ are revealed, say not less than n collisions. However, one can see that $\mathbf{X}$ is not the secret key or the trapdoor in our scheme. The short basis $\mathbf{S}$ is the trapdoor of the chameleon hash, and $\mathbf{S}$ will not be leaked even if $\mathbf{X}$ is revealed.

Parameters. The specific parameters of our scheme (including q, n, m, κ, σ) can follow the parameters of signature scheme presented in [21]. For instance, we use a prime q of length 25 bits, and take $n = 512$, $m = 8139$, $\kappa = 14$ and $\sigma = 15157$. Then the size of the hash value $\mathbf{h} \in \mathbb{Z}_q^n$ is 12800 bits, and size of the random vector $\mathbf{r}$ is about $m \cdot \log(12\sigma) \approx 142210$ bits. The specific parameters of our scheme in Section 3 of this paper could also be set similarly in this way.

4.2. Security Analysis

Theorem 2. *The proposed single-trapdoor chameleon hash function in Section 4.1 is a secure chameleon hash function with enhanced collision resistance in the random oracle model, and satisfies semantic security.*

The theorem is proved based on the following two lemmas.

Lemma 3. *The proposed single-trapdoor chameleon hash function in Section 4.1 satisfies enhanced collision resistance based on the $\mathsf{SIS}_{q,n,m,\beta}$ assumption for $\beta = (2\eta\sigma + 2d\kappa)\sqrt{m}$ ($\eta \in [1.1, 1.4]$ in practice) in the random oracle model.*

Proof. We will show that if there is an adversary $\mathcal{A}$ who can break the enhanced collision resistance, then we can construct an algorithm $\mathcal{B}$ who can solve the $\mathsf{SIS}_{q,n,m,\beta}$ problem, i.e., compute $\mathbf{v}$ such that $\mathbf{Av} = \mathbf{0} \mod q$ with $||\mathbf{v}|| \leq \beta$.

Denote $D_H = \{\mathbf{c} : \mathbf{c} \in \{-1, 0, 1\}^n, ||\mathbf{c}||_1 \leq \kappa\}$ as the range of the random oracle H. Generate keys as in KeyGen. The public key is $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{T} = \mathbf{AX} \in \mathbb{Z}_q^{n \times n}$, where $\mathbf{X} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ is a matrix with small coefficients. The trapdoor key $\mathbf{S}$ is a good basis of $\Lambda^{\perp}(\mathbf{A})$. The public key is given to the adversary $\mathcal{A}$. We will keep a list $\mathcal{L}$ of all the queries to H, so in case the same query μ_i is made twice, the previously answered $\mathbf{c}_i$ will be replied. Notice that if the random oracle H was not queried or programmed on some input μ, then $\mathcal{A}$ only has a $1/|D_H|$ chance of producing a $\mathbf{c}$ such that $\mathbf{c} = \mathbf{T}^{-1}(\mathbf{h} - \mathbf{Ar})$.

We can simulate the $\mathsf{UForgeO}$ as follows. When the adversary $\mathcal{A}$ queries with input $((\mathbf{h}_i, \mu_i, \mathbf{r}_i), \mu_i')$ such that $\mathbf{h}_i = \mathbf{Ar}_i + \mathbf{Tc}_i$, we sample $\mathbf{r}_i' \xleftarrow{\$} D_\sigma^m$ and program the random oracle $\mathbf{c}_i' = H(\mu_i') = \mathbf{T}^{-1}(\mathbf{h}_i - \mathbf{Ar}_i')$, and return $\mathbf{r}_i'$ as the collision.

If the adversary $\mathcal{A}$ queries the random oracle on μ_j, then we return a random element denoted as $\mathbf{c}_j \xleftarrow{\$} D_H$, and record it in the random list $\mathcal{L}$. Finally, $\mathcal{A}$ finishes running and outputs a pair of chameleon hash collision $(\mu, \mathbf{r})$ and $(\mu', \mathbf{r}')$. So we have $\mathbf{h} = \mathbf{Ar} + \mathbf{Tc} = \mathbf{Ar}' + \mathbf{Tc}'$. By re-arranging the above equality and plugging in $\mathbf{T} = \mathbf{AX}$, we obtain

$$\mathbf{A}(\mathbf{r} - \mathbf{r}' + \mathbf{Xc} - \mathbf{Xc}') = \mathbf{0}.$$

Since $||\mathbf{r}||, ||\mathbf{r}'|| \leq \eta\sigma\sqrt{m}$, and $||\mathbf{Xc}||, ||\mathbf{Xc}'|| \leq d\kappa\sqrt{m}$ we have that $||\mathbf{r} - \mathbf{r}' + \mathbf{Xc} - \mathbf{Xc}'|| \leq (2\eta\sigma + 2d\kappa)\sqrt{m}$, so the SIS problem is solved.

□

Lemma 4. *The proposed single-trapdoor chameleon hash function in Section 4.1 satisfies semantic security.*

Proof. Recall that the chameleon hash is computed as $\mathbf{h} = \mathbf{A}(\mathbf{r} + \mathbf{Xc}) \mod q$. Notice that using rejection sampling, the distribution of $(\mathbf{r} + \mathbf{Xc})$ is made to be D_σ^m for $\mathbf{r} \xleftarrow{\$} D_\sigma^m$, so the distribution of the syndrome $\mathbf{h} = \mathbf{A}(\mathbf{r} + \mathbf{Xc})$ is statistical close to uniform over $\mathbb{Z}_q^n$ [17, Lemma 5.2], and the conditional entropy $H[\mu|\mathbf{h}]$ is statistical close to $H[\mu]$. Therefore, our proposed scheme satisfies semantic security. □

5. Integrate with Blockchain

Our chameleon hash functions can be easily integrated with any type of blockchain, such as private, alliance, or public blockchain, regardless of the network types or consensus mechanisms.

5.1. Blockchain Basics

We use the notation in [16] to describe the blockchain. A block is a triple $B = \langle s, x, ctr \rangle$, where $s \in \{0,1\}^\kappa$, $x \in \{0,1\}^*$ and $ctr \in \mathbb{N}$. Block B is valid if

$$\mathbf{validblock}_q^D(B) := (H(ctr, G(s,x)) < D) \wedge (ctr < q) = 1.$$

$H, G : \{0,1\}^* \to \{0,1\}^\kappa$ are two collision-resistant hash functions, the parameter $D \in \mathbb{N}$ is the block's difficulty level, and $q \in \mathbb{N}$ is the number of hash queries that a user is allowed to make.

The blockchain $\mathcal{C}$ is a chain of blocks. The rightmost block is called the head of the chain, denoted by $\mathsf{Head}(\mathcal{C})$. A chain $\mathcal{C}$ with head $\mathsf{Head}(\mathcal{C}) = \langle s, x, ctr \rangle$ can be extended to a longer chain $\mathcal{C}' = \mathcal{C}||B'$ by appending a valid block $B' = \langle s', x', ctr' \rangle$ such that $s' = H(ctr, G(s,x))$, and the head of new chain $\mathcal{C}'$ is B'.

5.2. The Construction of Redactable Blockchain

Ateniese et al. [4] aim at *block-level* rewriting, i.e., rewriting entire blocks or cutting a subchain, while Derler et al. [12] target at *transaction-level* rewriting. The only difference is where the chameleon hash is applied. In Ateniese et al.'s block-level redactable blockchain, the internal hash function $G(\cdot)$ is replaced by a chameleon hash, while in Derler et al.'s transaction-level scheme, the chameleon hash function is used in building the Merkle tree (the Merkle tree is used in blockchain to accumulate transactions in a block). If a transaction is allowed to be edited later, it is first preprocessed by the chameleon hash, and then the output is used instead of the original transaction to build the Merkle tree. In both schemes, the randomness used in the chameleon function is stored in the block without aggregating in the Merkle tree.

We can construct both types of redactable blockchain following the frameworks in [4] and [12] respectively using our proposed chameleon hash functions, according to the needs in specific applications. We give an example with transaction-level rewrites in Fig.1. In this toy example, block B_i accumulates four transactions $T_{i,1}, \cdots, T_{i,4}$. Assume that transaction $T_{i,1}$ should be redactable, then it is preprocessed using our chameleon hash function, and the output is used to build the Merkle tree, together with other unredactable transactions which are processed normally.

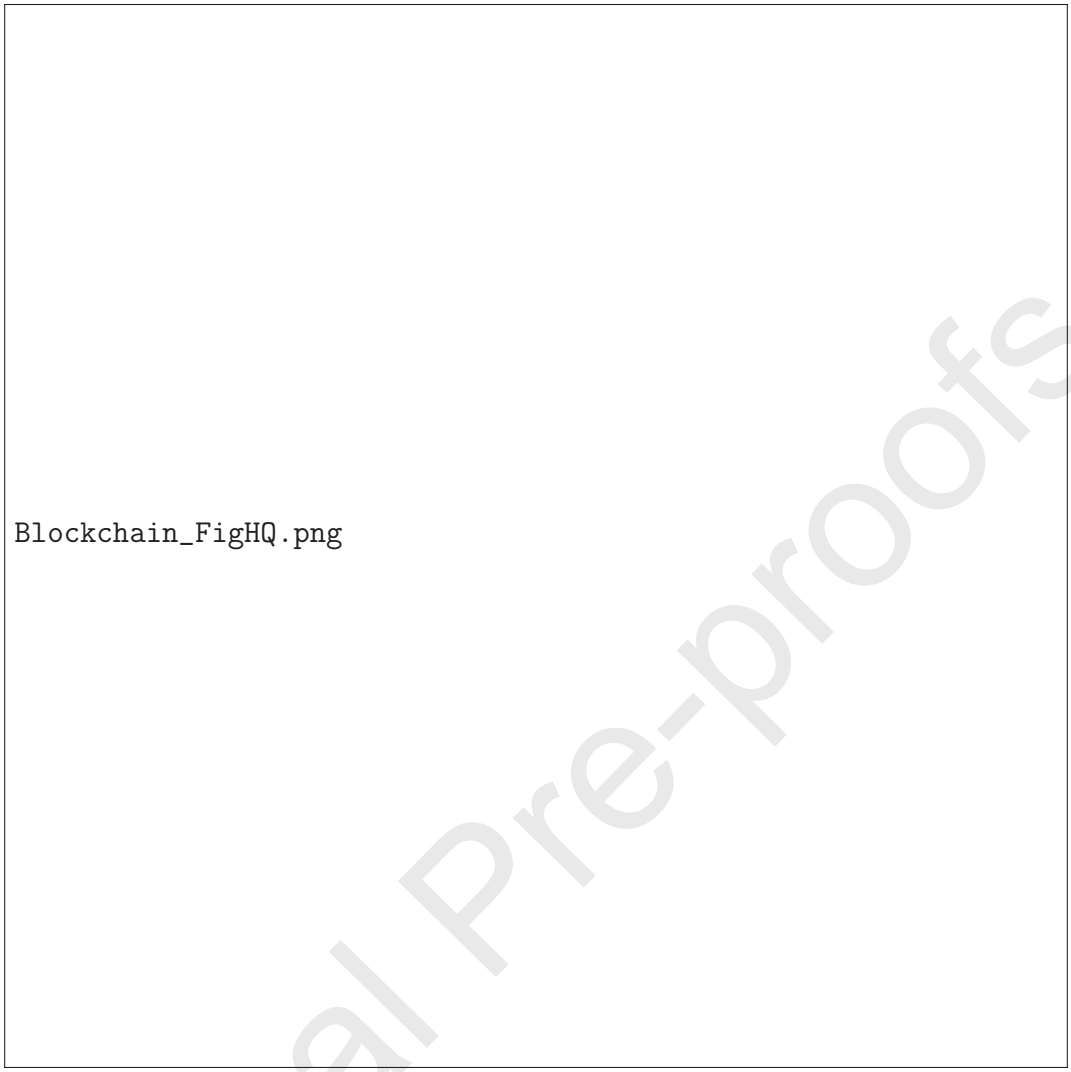


Figure 1: Using chameleon hash to construct redactable blockchain with transaction-level rewrites.

5.3. Prevent Misuse of Redaction Functionality

Since redaction is becoming an important requirement and may even be legally obliged in blockchains, a natural question is how to manage the trapdoor for the chameleon hash function, and to prevent misuse of redaction functionality. The answer to this question is dependent on applications. We specify how to manage the trapdoor in the following three types of blockchains.

- **Private blockchain.** This type of blockchain is widely used by the financial sector. Only a central authority is allowed to write on the blockchain, while the read permissions may be public or restricted. In this scenario, the trapdoor management is simple. The trapdoor key could be given to the central authority, who has the power to compute collisions and redact blocks.

- **Alliance blockchain.** This type of blockchain is widely used between enterprises and organizations, and has much more efficient consensus mechanisms. In this scenario, only

predetermined entities can participate in the consensus process, so the trapdoor key can be managed in the following two ways:

1. The trapdoor is shared among all the parties in the alliance by using our distributed key generation, and redactions can be realized using MPC, refer to Section 3.2.
2. The trapdoor is held by the redaction-authority ($\mathcal{RA}$). Meanwhile, a voting scheme, as in [13, 20, 25], is applied to decide whether a redaction could be allowed. If enough votes are collected, say more than half of the participants have voted to redact, then the redaction is operated by the $\mathcal{RA}$. Both our chameleon hash functions can combine with these voting techniques without "breaking" the blockchain.
   More specifically, Deuber et al. [13] simply use a collision-resistant hash function to construct a voting scheme, without using heavy cryptographic primitives, such as signatures. However, they directly "break" the blockchain in redaction, and the old chain should be kept and verified beforehand in chain validation. The same problem exists in [25], where multi-signature is used to shorten the voting period. Li et al. [20] also use signature technique to vote, and adopt chameleon hash to avoid "chain-breaking". However, the chameleon hash they used is not quantum-resistant and has the key-exposure problem.
   Our chameleon hash functions can combine with these voting mechanisms and avoid these problems. Note that by using the voting mechanism, the redactable blockchain offers an extra property of public verifiability and accountability.

- **Public blockchain.** This type of blockchain is open to everyone. Any party can send transactions to the network and have them included in the blockchain. Bitcoin is a good example of public blockchain. In this scenario, the trapdoor can either be shared among all parties of the network, or shared among a chosen subset of the parties. The former solution may not be very efficient due to the MPC protocol, while the latter one might be a practical solution, though not completely decentralized. Similar to the condition in Bitcoin, where the majority of the hashing power is controlled by a small amount of parties, but it still works well.

6. Conclusion

Redaction is a desired property in blockchain with many reasons, such as removing inappropriate content, supporting application requirements, and even be legally obliged. Chameleon hash function with enhanced collision-resistance is an important tool to construct redactable blockchain. In this paper, we propose two concrete single-trapdoor key-exposure free chameleon hash functions based on lattice, which naturally satisfy enhanced collision-resistance, and show their applications in redactable blockchain. To the best of our knowledge, this type of chameleon hash is rare and our constructions are the first instances based on quantum-resistant assumptions. Our constructions, without using encryption and NIZK, are more compact and computational efficient than the generic constructions. Besides, they can achieve distributed trapdoor management, which seems unsupportable in Ateniese et al.'s NIZK-based generic construction. Then, in order to prevent the misuse of redaction functionality, we introduce two different mechanisms. The first one is a distributed key sharing, which can be efficiently implemented in our first chameleon hash under the support of a semi-honest secure multi-party sum protocol. The second one is a more general

redaction voting mechanism, which naturally provides public accountability. Our chameleon hash functions can be efficiently integrated with any blockchain technologies, regardless of the consensus protocols and network models, and require only minor changes to the current blockchains in use. For further work, we are trying to present a distributed key generation and a MPC protocol for redaction for the second chameleon hash as well, so as to extend its flexibility in applications. It is also an interesting issue to enrich the applications of the proposed chameleon hash functions, such as quantum-resistant chameleon signatures and off-line/on-line signatures.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 61902081), the Natural Science Foundation of Guangdong Province, China (No. 2018A030313974), the Science and Technology Planning Project of Guangdong (No. 2018A050506087), and the Fundamental Research Funds for the Central Universities (No. 19lgpy217).

References

[1] Accenture. Accenture debuts prototype of 'editable' blockchain for enterprise and permissioned systems. https://newsroom.accenture.com/news/accenture-debuts-prototype-of-editable-blockchain-for-enterprise-and-permissioned-systems.htm.

[2] G. Ateniese and B. de Medeiros. Identity-based chameleon hash and applications. In *Finacial Cryptography and Data Security–FC 2004*, pages 164–180. LNCS 3110, Springer-Verlag, 2004.

[3] G. Ateniese and B. de Medeiros. On the key-exposure problem in chameleon hashes. In *Proc. of the 4th Conference on Security in Communication Networks–SCN 2004*, pages 165–179. LNCS 3352, Springer-Verlag, 2005.

[4] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. Redactable blockchain-or-rewriting history in bitcoin and friends. In *IEEE European Symposium on Security & Privacy*, pages 111–126, 2017.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[6] 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 *Public-Key Cryptography - PKC 2017*, 2017.

[7] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[8] X. Chen, F. Zhang, and K. Kim. Chameleon hashing without key exposure. In *Proc. of the 7th International Information Security Conference–ISC 2004*, pages 87–98. LNCS 3225, Springer-Verlag, 2004.

[9] X. Chen, F. Zhang, W. Susilo, and Yi Mu. Efficient generic on-line/off-line signatures without key exposure. In *ACNS 2007*, pages 18–30. LNCS 4521, Springer-Verlag, 2007.

[10] X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, and K. Kim. Identity-based chameleon hashing and signatures without key exposure. *Information Sciences*, 265:198–210, 2014.

[11] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[12] D. Derler, K. Samelin, D. Slamanig, and C. Striecks. Fine-grained and controlled rewriting in blockchains: Chameleon-hashing gone attribute-based. In *NDSS 2019*, 2019.

[13] D. Deuber, B. Magriy, S. Aravinda, and T. Krishnan. Redactable blockchain in the permissionless setting. In *IEEE Symposium on Security and Privacy 2019*, 2019.

[14] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In *Advances in Cryptology-CRYPTO 2013*, number 8042 in LNCS, pages 40–56. Springer-Verlag, 2013.

[15] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[16] J.A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, number 9057 in LNCS, pages 281–310. Springer-Verlag, 2015.

[17] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Association for Computing Machinery, 2008.

[18] M. Khalili, M. Dakhilalian, and W. Susilo. Efficient chameleon hash functions in the enhanced collision resistant model. *Information Sciences*, 510:155–164, 2020.

[19] H. Krawczyk and T. Rabin. Chameleon hashing and signatures. In *Network and Distributed System Security Symposium (NDSS)*, pages 143–154, 2000.

[20] P. Li, H. Xu, T. Ma, and Y. Mu. Research on fault-correcting blockchain technology. *Journal of Cryptologic Research*, 5(5):501–509, 2018.

[21] V. Lyubashevsky. Lattice signatures without trapdoors. In *Advances in Cryptology-EUROCRYPT 2012*, number 7237 in LNCS, pages 738–755. Springer-Verlag, 2012.

[22] D. Naccache, D. Pointcheval, and J. Stern. Twin signatures: an alternative to the hash-and-sign paradigm. In *8th ACM Conference on Computer and Communication Security (ACM CCS)*, pages 20–27. ACM Press, 2001.

[23] I. Puddu, A. Dmitrienko, and S. Capkun. μchain: How to forget without hard forks. Cryptology ePrint Archive. https://eprint.iacr.org/2017/106, 2017.

[24] C. Wu, X. Chen, and D. Long. A new effcient on-line/off-line threshold signature scheme. *Chinese Journal of Electronics*, 18(2):321–324, 2009.

[25] J. Xu, X. Li, L. Yin, B. Guo, H. Feng, and Z. Zhang. Redactable proof-of-stake blockchain with fast confirmation. Cryptology ePrint Archive. https://eprint.iacr.org/2019/1110, 2019.

[26] J. Xu, K. Xue, H. Tian, J. Hong, D. S. L. Wei, and P. Hong. An identity management and authentication scheme based on redactable blockchain for mobile networks. *IEEE Transactions on Vehicular Technology*, 69(6):6688–6698, 2020.