Redactable Blockchain using Enhanced Chameleon Hash Function

1st Kondapally Ashritha
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
ashrithakondapally@gmail.com

2nd Sindhu M
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
m_sindhu@cb.amrita.edu

3rd Lakshmy KV
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
kv_lakshmy@cb.amrita.edu

***Abstract*—Immutability is a core principle of blockchain platforms that help all participants to have an exact global log of transactions. Immutability is easier to achieve in permissioned blockchain platforms where a trusted group are the validators. In public platforms, this vital property of blockchain can be achieved only through decentralized economic mechanisms like Proof-of-Work(PoW). Blockchain along with the concept of asset tokenization provides liquidity to markets like stock trading and exchange other intangible objects. With changes in regulatory policies and/or government laws, there could be a requisite to alter the contents in such markets. We propose an idea of using Chameleon hash functions that will enable modification of a block without changing other block contents. In our proposal, the trapdoor key used to redact a block is split among major validators and is reconstructed using Multi-Party Computation(MPC). Redaction happens when the major validators agree the suggested changes by digitally signing the proposal. This eliminates the need to rely on a trusted party. We also propose an idea of using second trapdoor key that will be with the creator of the block. This can be used in scenarios where the block redaction should not happen without the consent of the creator.**

***Index Terms*—Chameleon hash function, Blockchain, Multi-party Computation, Ephemeral key, Secret Sharing, Digital Signatures**

I. Introduction

Blockchain technology brings together two important aspects that shape the new internet. The idea of decentralization along with de-duplication has found enormous applications ranging from basic financial transactions to complex smart contracts. Blockchain technology is built in such a way that once the data is published in the chain, it cannot be removed or modified [1]. This key principle of immutability is very important in order to achieve data integrity in a trust less distributed network. Even though 100% immutability can never be theoretically promised in blockchain environment, strong economic practices like Proof-of-Work (PoW), Proof-of-Stake (PoS) gives a practical assurance that only one common ledger is maintained. In permissioned blockchain platforms like Ripple where the validators are fixed, it is easy to achieve immutability. It is always debatable to provide power to certain people to modify the already existing contents of a blockchain as that would hinder the normal users to use the blockchain where they have to trust unknown parties. Hence before adding the feature of redactablilty in distributed ledger platforms like blockchain, it is important to make sure that the possibility of validators cheating the network is minimum.

Since blockchain is basically a trust less distributed system, 100% cheating immunity cannot be guaranteed. In these systems, the probability of a user cheating is inversely proportional to the amount of investments he has made on the particular blockchain platform. For example, in Bitcoin, the probability of a miner who owns 20% mining power of the overall network will cheat is very minimal since he has a considerable amount of economic interest on bitcoin whereas a normal user with almost zero mining power will not have lots at stake even if bitcoin platform falls. Thus the power of redactability should only be given to those who owns a substantial amount of mining power.

In blockchain 1.0, which is mainly designed for carrying out financial transactions, the need for reversing a transaction is not required. But with the concepts of asset tokenization where most of the tangible and intangible objects of the real world are converted into crypto tokens and binded with the blockchain platform, there could occur scenarios where redaction is necessary. For example, consider a scenario where a land asset is converted into a crypto token and published in the blockchain, later if the land laws of respective country changes, the blockchain platform will be obliged to change the content related to that asset. Redactability of blockchain can be achieved by generating blocks using a special type of hash function called Chameleon Hash Function (CHF) [2]. Chameleon hash functions will have corresponding trapdoor key using which one or many blocks can be redacted depending upon the design.

In private or permissioned blockchain, the key can be in the hands of a Trusted Third Party (TTP) who can solely redact the blocks in the blockchain. This trusted third party can be determined based upon voting of validators who are fixed in case of permissioned blockchain. For example, in Ripple, the 18 validator nodes can vote among themselves and elect a node as the TTP. However, such kind of voting mechanism is not possible in a public network where there are no fixed validators for voting and there exists a very minimal trust between the users [3]. In such a scenario, we are proposing to split the trapdoor key into n shares and distribute among the major miners. The probability that the miners will pool together and cheat on the network is very less because of the economic investments they are putting in. For example, in Bitcoin, the trapdoor key can be split into 7 shares and distributed among the top 7 miners, ranked based upon the number of blocks published.

The paper is organized in the following manner: Section II describes basic cryptographic notations and definitions that are used throughout this paper. Section III explains the existing model and flaws in it. Section IV describes the proposed architecture, security aspects of the new model. In Section V, we look into different applications of the proposed architecture. The paper concludes in Section VI where future works and challenges are discussed.

II. Cryptographic Terminologies

A. Chameleon Hash Function

Chameleon hash functions are special type of cryptographic hash functions which can generate hash collisions efficiently with the knowledge of the trapdoor key.

Consider a message m and a randomly chosen parameter r. Given a trapdoor key tk, the trapdoor holder can find the pairs (m, r) and (m', r') using chameleon hash function such that $CH(m, r) = CH(m', r')$ where $CH()$ represents chameleon hash function. Note that the messages m and m' should be different i.e., $m \neq m'$.

A standard chameleon hash function consists of efficient algorithms such as Key Generation (HG), Hash Generation (H), Hash Verification (HV) and Hash Collision (HC) which is specified as follows:

$$CH = (HG, H, HV, HC)$$

1) $\boldsymbol{HG(1^n) = (hk, tk)}$**:** It takes the security parameter $n \in N$ as input, and gives public hash key hk and a secret trapdoor key t as output.

2) $\boldsymbol{H(hk, m) = (h, z)}$**:** It takes hash key hk, a message $m \in M$, and a random value $r \in R$ as input, and produces a pair (h, z) that contains the hash value h and a check string z.

3) $\boldsymbol{D = HV(hk, m, (h, z))}$**:** It takes a message $m \in M$, the hash value h, and a check string z as input, and outputs a bit D that equals 1 if (h, z) is a valid hash pair for the message m, otherwise D equals 0.

4) $\boldsymbol{C = HC(t, (h, m, z), m')}$**:** It takes the trapdoor key t, a valid tuple (h, m, z), and a new message $m' \in M$ as input, and outputs a new check string z' such that

$$HV(hk, m, (h, z)) = HV(hk, m', (h, z')) = 1$$

B. Secret Sharing

Secret Sharing is a group oriented cryptographic method that allows shifting the knowledge of secret information from a single entity to a group of entities. It facilitates a secure way of storing and transmitting confidential information as shares [4]. Specific conditions should be satisfied to reveal the secret information. Each of n participants is provided with one share each, and any group of t (threshold) or more participants can combine their shares to generate the secret information.

For example, if a user wants to divide a secret S into two parts, he can choose a random S_1 as first secret and determine S_2 by performing an XOR operation with S and S_1 as inputs.

Shamir's threshold scheme is based on polynomial interpolation that allows any k out of n participants to recover the secret. The secret S is divided into n shares $S_1, \ldots, S_n$ in such a way that:

1) It is easy to reconstruct the secret S with the knowledge of any k or more S_i shares.
2) The secret remains completely undetermined with the knowledge of any $k - 1$ or fewer S_i shares.

This scheme is known as (k, n) threshold scheme. If $k = n$, then every share S_i of the secret S is required to reconstruct the secret.

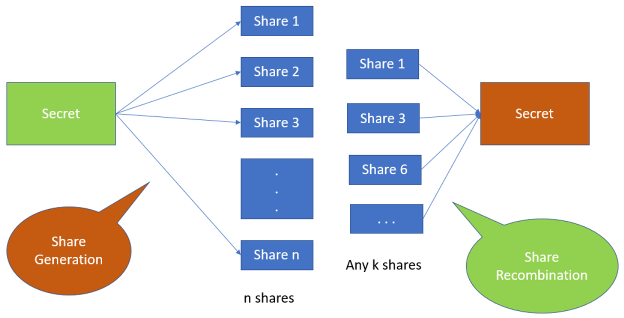

Fig. 1. (k, n) secret sharing scheme

Shamir's secret sharing scheme is a type of Linear Secret Sharing (LSS) scheme that is known to have lot of vulnerabilities. Therefore, to improve the security we are proposing to use a Non-Linear Secret Sharing (NLSS) scheme.

C. Secure Multi-party Computation

Multi-Party Computation (MPC) is a method in which the parties jointly compute a function over their private inputs [5]. The inputs are obtained in the form of shares from secret sharing scheme.

Consider a Multi-party computation in which n users, $u_1, u_2, ..., u_n$ each have their input data, $d_1, d_2, ..., d_n$ respectively. All the users compute the value of a public function f on their input data: $f(d_1, d_2, ..., d_n)$ while keeping their own inputs secret.

For example, consider two parties Alice and Bob, with inputs A and B respectively. To find out the highest value of the two inputs, without revealing their individual values to each other, consider the following two cases:

1) There exists a trusted party to whom the participants send their respective inputs and the trusted party calculates the maximum value amongst the two and reports the same to both Alice and Bob.
2) When there is no trusted party, Alice and Bob performs a two party computation (a generic model of Multi-party computation), with each other and then result will only reveal who has the highest value not the input figures of individual parties.

D. Digital Signatures

A Digital Signature is a mathematical scheme that is used for protecting the authenticity of digital documents. A signature is said to be valid if and only if it provides authentication, non-repudiation as well as integrity. They are widely used in financial transactions to detect forgeries. Digital signatures usually use asymmetric cryptography which gives the receiver a trust that the message was sent by the original sender and is not tampered.

A digital signature [6], [7] scheme typically consists of 3 phases namely Key generation (G), Signing (S) and Signature Verification (V).

1) **Key Generation:** A Security parameter r is randomly and 1^r is given as input. Public key pk and its corresponding private key sk is obtained as output.

$$(pk, sk) \leftarrow G(1^n)$$

2) **Signing:** The private key sk, that is generated is used for signing a message m and produces a signature t.

$$t \leftarrow S(sk, m)$$

3) **Signature Verification:** The message m, public key pk and signature t is given as input to the verification algorithm and it either accepts (D=0) or rejects (D=1) the message.

$$D \leftarrow V(pk, x, t)$$

For correctness, S and V must satisfy

$$Pr[(pk, sk) \leftarrow G(1^n), V(pk, x, S(sk, m)) = accepted] = 1$$

III. Existing System

In the existing model of redactable blockchain [8], the trapdoor key k is generated initially and is split into n shares by using Linear Secret Sharing scheme. The n shares are distributed among the top n validators. To redact a block, all the n validators need to perform a secure Multi-party computation to regenerate the trapdoor key. By using the key k, the validator can redact one block or any number of blocks. During the block verification process, the validators compute hash collision of the new and the previous content so that the hash does not change. (Refer Figure 2)

A block contains three pieces of information, the $PrevHash$, $Transactions$ and a random parameter $Randomness$. The $PrevHash$ and $Transactions$ values are determined by the miner and $Randomness$ is used in order to add redactability feature. The hash collision is computed by randomly trying out different values for $Randomness$ parameter so that the modified content along with $Randomness$ gives the same hash as it was before.

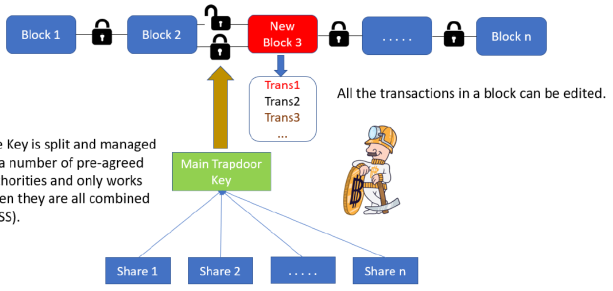
Block 1
Block 2
New Block 3
. . . . .
Block n
Trans1
Trans2
Trans3
...
All the transactions in a block can be edited.
The Key is split and managed by a number of pre-agreed authorities and only works when they are all combined (LSSS).
Main Trapdoor Key
Share 1
Share 2
. . . . .
Share n

Fig. 2. Redactable Blockchain

Consider a blockchain platform that uses previous hash value, *PrevHash*, the final merkle hash of proposed transactions, *Transactions* and a *random* value r. If a validator proposes changes to a block with hash value s, he can modify the previous hash value and/or the transaction contents that will modify the *Transactions* value. The random value r will be modified such that the final hash value s will not change. This is important because the next blocks in the blockchain are linked to blockchain with hash value s.

For example, if a validator want to change the *PrevHash* s and transactions Merkle root hash x and *random* value r of a block to new *PrevHash* s' and new transactions Merkle root hash x' , he has to determine a hash collision value r' that will make the final hash value unchanged using a chameleon hash function $CH()$. (Refer Figure 3)

$$CH(s, x, r) = CH(s', x', r')$$

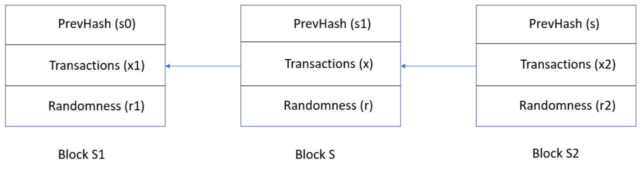

Fig. 3. Redactable Blockchain internal structure

A. Issues in Existing System

1) **No Content Verification:** During redaction, when a chosen validator redacts a block, the contents inside the block are not verified by other validators. They only calculate a hash collision value that will make sure that the overall hash value of the block remains unaltered. Thus, malicious validator can modify the contents as he wishes.
2) **Secret sharing:** The model is based on linearly splitting the secret shares and hence is vulnerable to many attacks like Tompa-Woll attack.

3) **Single trapdoor key:** Since only one trapdoor key is used, once a validator regenerates the trapdoor key [9], he can use it multiple times on multiple blocks.
4) **No consent of initial block publisher:** Redaction is done without the knowledge of the actual block publisher. In platforms where block reward is provided, there exists some scenarios where the publisher should also be involved while redacting the block.

IV. Proposed System

In this section, we will be focusing upon the issues of the existing model discussed in section III-A and propose design models that can reduce if not eliminate them.

A. Content Verification

If a validator wants to redact a block, he should first compute the trapdoor key by collaborating with other share holders of the key. Before publishing a block, the validator has to first broadcast the changes among the key holders. The chain will be redacted only if all the validators agree the suggested changes by digitally signing the newly proposed block. In this way, one validator cannot solely modify contents of any block without the agreement from other validators.

B. Secret Sharing

In Tompa-Woll attack, one of the validators can submit a false share such that only he can be able to obtain the correct secret. This is mainly due to the linearity property. To resist from this attacks, we are proposing an idea of using Non-Linear Secret Sharing [10], [11] instead of Shamir's secret sharing which is a type of Linear Secret Sharing scheme. Cheatings [12] can be corrected in non-linear schemes by using decoding techniques like Read-Solomon codes.

A Non-linear secret sharing scheme (k, n) for n parties such that

1) any k or more shares will regenerate the secret value.
2) no information about the secret can be obtained with $k-1$ or fewer shares.

An access structure is a function that is usually defined as the collection of shares that can be used for reconstructing a secret. Unlike Linear secret sharing, in Non-linear secret sharing scheme, the access structure cannot be realized because a non-linear function is used in a way that the structure of each and every share is different and hence a malicious user cannot regenerate the secret by submitting a false share.

C. Ephemeral Trapdoor Key

By using an ephemeral key that is specific to a particular block, along with the trapdoor key we can restrict the modifications a validator can make. Now, in order to redact a block, the validator will require both the trapdoor key that has to be regenerated by performing secure Multi-party computation with other validators as well as the ephemeral key that can be given to the initial proposer of the block.

A chameleon hash function with ephemeral trapdoor key [13] is a tuple of five algorithms $(ParGen, KGen, Hash, HashChk, Adapt)$

1) **ParGen**: A security parameter λ is taken as input and the public parameter pp_{ch} is obtained as output.
$$pp_{ch} \leftarrow ParGen(1^{\lambda})$$
2) **KGen:** The public parameter pp_{ch} is taken as input and the private key pk_{ch} and public key sk_{ch} are obtained as output.
$$(sk_{ch}, pk_{ch}) \leftarrow KGen(pp_{ch})$$
.
3) **Hash:** It takes the public key pk_{ch}, and a message m as input and produces a hash h, randomness r, and the ephemeral trapdoor key tk as output.
$$(h, r, tk) \leftarrow Hash(pk_{ch}, m)$$
.
4) **HashChk:** It takes as input the public key pk_{ch}, a message m, a hash h, and randomness r. It outputs a decision bit $d \in \{true, false\}$, which indicates whether the hash is correct or not.
$$d \leftarrow HashChk(pk_{ch}, m, r, h)$$
5) **Adapt:** It takes sk_{ch}, the old message m, the old randomness r, the new message m', the hash h, and the trapdoor information tk as input and outputs a new randomness r' for the message m'.
$$r' : r' \leftarrow Adapt(sk_{ch}, m, m', r, h, tk)$$

D. Consent from block publisher

By keeping the ephemeral key with the initial block proposer, the validator will require permission from the parent block publisher to make changes to a particular block. (Refer Table I)

TABLE I
COMPARISON OF EXISTING AND PROPOSED MODELS

Property	Existing Model	Proposed Model
Secret Sharing	LSS	NLSS
Ephemeral Key	No	Yes
Content Verification	No	Yes
Multiple Block Redaction	Yes	No
Initial Publisher Approval	No	Yes

E. Architecture

In the proposed system, we are using an ephemeral trapdoor key K that is specific to a particular block S. This key K will be with the initial proposer of the block S. The transactions x inside the block S is locked with the key K. By this way, to modify the transaction x to x', the permission of the initial block publisher is required. (Refer Figure 4). The main trapdoor key is split using a Non-linear secret sharing scheme and shared among the top n validators of the blockchain

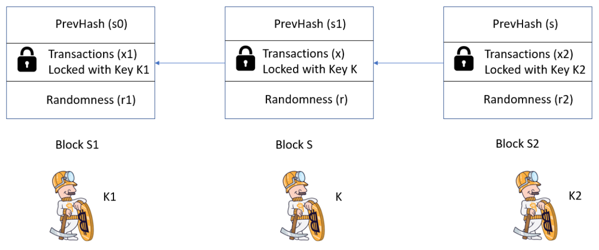

Fig. 4. Block Structure in Redactable Blockchain using Ephemeral trapdoor key

platform. To redact a block, all the validators will have to perform a secure Multi-party computation to regenerate the secret trapdoor key. Further, before publishing the modified block, all the share holders of the trapdoor key will have to agree to the changes by digitally signing the transaction. By this way, we can make sure that while redacting a block all the major validators and also the initial block publisher verify and agree with the modified content.

1) Deletion of a Block: There could be scenarios where a particular block S contain entire list of spam transactions. In such cases, the validators could either delete the transactions x of the block S and hence publish the block S with transactions $x' = 0$. This is similar to the architecture proposed in section IV-E. This solution is not ideal because we are storing a block S with no content in it. (Refer Figure 5) By slightly modifying the architecture in section IV-E, we can delete the entire block from the blockchain. For example, if a validator wants to delete a block S which has a succeeding block $S2$ and a preceding block $S1$, he can achieve it by changing the *PrevHash* of block $S2$ from s to $s1$.

In such cases, the *PrevHash* should also be locked with the ephemeral key. The ephemeral key K of the block to be deleted is used to remove the link of the block S from the blockchain.

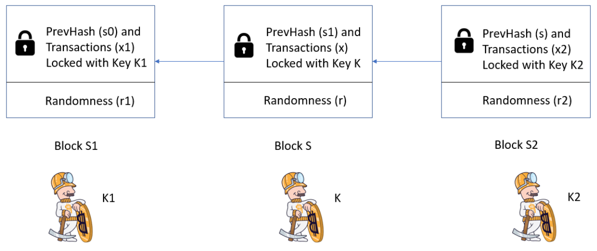
PrevHash (s0) and Transactions (x1) Locked with Key K1
Randomness (r1)
PrevHash (s1) and Transactions (x) Locked with Key K
Randomness (r)
PrevHash (s) and Transactions (x2) Locked with Key K2
Randomness (r2)
Block S1
Block S
Block S2
K1
K
K2

Fig. 5. Block Deletion in Blockchain

2) Revoking the Ephemeral Key: In Public blockchain platforms where anyone can mine a block, the ephemeral key will be lost if the validator with relatively low mining power mines a block and then leaves the network. Now, it will be impossible to redact a block as the ephemeral key is not available. To eradicate such circumstances, we propose a ephemeral key revocation scheme where the miner will have to relinquish the secret ephemeral key after a particular amount of time.

With little modifications, this can also be extended to regenerate the main trapdoor key where a holder of one share is planning to leave the network.

V. Applications

Redactability is easier to achieve in private blockchain because the validators are fixed whereas in a public blockchain where anyone could mine a block. In this section, we will explain about the two different proposals based on the type of blockchain in detail.

A. Permissioned Blockchain

Consider a permissioned blockchain with n permanent validators. Here, the trap- door key k will be split into n shares using non-linear secret sharing. In this case, we do not need an ephemeral key since any block will be published by the fixed validators who already have the share of the main trapdoor key.

B. Public Blockchain

In a public blockchain like Bitcoin, anyone could join a network and publish a block, it is theoretically impossible to split the main trapdoor key and give shares to all the miners. By specifically considering Bitcoin where a pool of seven miners publish most of the blocks, the trapdoor key can be divided into seven shares and distributed among them. But, for a particular block, the publisher could be someone who is not among the top seven miners. Thus, we need an ephemeral trapdoor key to make sure that the initial proposer is also involved in the redaction process.

C. Other Applications of Chameleon Hash Functions

1) Sanitizable Signatures: There exist some environments where the different users can access same data depending on their role. For example, consider a medical report in which few portions of the data remain confidential and only higher authorities can be able to access it. The authentication and integrity is usually achieved by using digital signatures. Sometimes there can be situations where an authorized third party should modify the content in the document. But, if the original signer of the document is not available or his/her key is expired etc., then authorized third party should be able to sign the document with a valid signature on behalf of original signer without contacting him/her. This can be achieved by using Sanitizable Signatures. The signer and the trusted party agree upon the mutable portions of the document before the modification such that the trusted party can modify only those portions.

Consider a case where a trusted party wants to modify a document. The original signer of the document t will partition it into some n blocks. The signer selects some blocks out of n blocks and signs it by computing the chameleon hash using the public key pk of third party. Now, as the private key sk is with the third party only he can be able to compute hash collision and thus modify the selected portions of the block without changing the signature i.e., the signature is valid.

2) Wireless Sensor Networks (WSN): Sensors are used in Wireless Sensor Network (WSN) to collect the information about temperature, sound, etc., at different locations, process and send it back to the destination. Usually, a Multi- hop network contains n number of nodes in which each and every node contains one or more sensors. Sensor nodes can transfer the information to other sensor nodes via a base station. The information should not be altered during the transmission. If two nodes say $n1$ and $n2$ wants to communicate with each other, they must authenticate themselves before data transfer. Authentication can be achieved by using Chameleon hash function in which both the nodes $n1$ and $n2$ share a key k through secure channel such that only the key holders can compute the hash collisions.

3) Chameleon Signatures: In chameleon signature scheme, the signer of a particular message m uses chameleon hash function to compute the hash h. This hash can be signed by using any signing algorithm. For example, if user A wants to send a message to user B such that no other users should be able to know information about the message, then user A uses the chameleon hash function of user B, computes hash h and signs it using any digital signature algorithm. When user B receives the signature he can be able to verify that the signature is valid by computing hash collision. In this way, authentication is guaranteed and the signature is non-transferable.

VI. Conclusion

The proposed model guarantees content verification as well as improves the security of secret sharing mechanism by using a Non-linear methods of secret sharing. By introducing a second trapdoor key, this model also make sure that the redaction does not happen without consent of the initial publisher of the block. The content in the block can be verified using digital signatures Since, ephemeral key is created for every block, key management will be an issue. By introducing Keyless schemes, system can be further improved.

References

[1] Zyskind, Guy, and Oz Nathan. ”Decentralizing privacy: Using blockchain to protect personal data.” Security and Privacy Workshops (SPW), 2015 IEEE. IEEE, 2015.

[2] Bellare, Mihir, and Todor Ristov. ”A characterization of chameleon hash functions and new, efficient designs.” Journal of cryptology 27.4 (2014): 799-823.

[3] Ambili, K. N., M. Sindhu, and M. Sethumadhavan. ”On Federated and Proof Of Validation Based Consensus Algorithms In Blockchain.” IOP Conference Series: Materials Science and Engineering. Vol. 225. No. 1. IOP Publishing, 2017.

[4] Feldman, Paul. ”A practical scheme for non-interactive verifiable secret sharing.” Foundations of Computer Science, 1987., 28th Annual Symposium on. IEEE, 1987.

[5] Goldreich, Oded. ”Secure multi-party computation.” Manuscript. Preliminary version 78 (1998).

[6] Rivest, Ronald L., Adi Shamir, and Leonard Adleman. ”A method for obtaining digital signatures and public-key cryptosystems.” Communications of the ACM 21.2 (1978): 120-126.

[7] https://en.wikipedia.org/wiki/Digital_signature

[8] Ateniese, Giuseppe, et al. ”Redactable blockchain-or-rewriting history in bitcoin and friends.” Security and Privacy (EuroS&P), 2017 IEEE European Symposium on. IEEE, 2017.

[9] Ateniese, Giuseppe, and Breno de Medeiros. ”On the key exposure problem in chameleon hashes.” International Conference on Security in Communication Networks. Springer, Berlin, Heidelberg, 2004.

[10] Zhang, WeiGuo, Xi Sun, and JunPo Yang. ”On constructing 1-cheating immune secret-sharing functions.” International Journal of Computer Mathematics 89.1 (2012): 30-34.

[11] dela Cruz, Romar, and Huaxiong Wang. ”Cheating-immune secret sharing schemes from codes and cumulative arrays.” Cryptography and Communications 5.1 (2013): 67-83.

[12] Pieprzyk, Josef, and Xian-Mo Zhang. ”On cheating immune secret sharing.” Discrete Mathematics and Theoretical Computer Science 6.2 (2004): 253-264.

[13] Camenisch, Jan, et al. ”Chameleon-hashes with ephemeral trapdoors.” IACR International Workshop on Public Key Cryptography. Springer, Berlin, Heidelberg, 2017.

[14] Nakamoto, Satoshi. ”Bitcoin: A peer-to-peer electronic cash system.” (2008).

[15] Antonopoulos, Andreas M. Mastering Bitcoin: unlocking digital cryptocurrencies. ” O’Reilly Media, Inc.”, 2014.