Serving at the Edge: A Redactable Blockchain with Fixed Storage

Jingning Zhang, Youshui Lu(✉), Yuhao Liu, Xu Yang, Yong Qi, Xinpei Dong, and Haoming Wang

School of Electrical and Information Engineering,
Xi'an Jiaotong University, Xi'an 710071, China
{jingning,lyuhao}@stu.xjtu.edu.cn, lucienlu@me.com,
sunshine561@163.com, qiy@mail.xjtu.edu.cn, dongxinpei@gmail.com,
wanghaomingwj@126.com

Abstract. As a promising approach to extend cloud resource and service on the Internet-of-Things (IoT), edge computing has attracted significant attention. However, edge computing faces challenges in its decentralized management and data reliability. To meet this gap, many approaches propose to use blockchain technology to enable distributed storage and computation at the edge nodes, thus guaranteeing reliable access and control of the network. However, the resource-constraint nature of the edge node makes it difficult to store the entire chain as the data volume increases. To address this issue, we propose Re-chain, a re-writable blockchain *with fixed storage*. Re-chain supports re-writing of the onchain historical transactions in chronological order without changing the block hash, as a result, the total size of the blockchain will not increase. Our protocol is consensus-based and uses the proposed threshold trapdoor chameleon hash (TTCH) to constraint re-write operations. With this regards, Re-chain achieves both decentralized re-writing design and fault-tolerance at the same time. We provide security analysis and evaluation experiments to demonstrate the feasibility of Re-chain, the results show that the performance of Re-chain is acceptable when it is executed at a medium scale.

Keywords: Internet of Things · Chameleon hash · Redactable blockchain · Edge computing

1 Introduction

As the rapid advancement in computing technologies has enabled a wide range of applications, edge computing proposes a novel model for providing computational resources close to billions of end devices at the edge of the network. Edge computing has numerous applications in the Internet of Things (IoT), including healthcare, smart grids, manufacturing, etc. [7]. Edge computing that scales to a large number of sites is a cheaper way to achieve scalability than servers in the corporate center. However, its heterogeneity and resource-constraint nature will

© Springer Nature Switzerland AG 2020
G. Wang et al. (Eds.): WISA 2020, LNCS 12432, pp. 654–667, 2020.
https://doi.org/10.1007/978-3-030-60029-7_58

bring security challenges. During data transmissions, some attacks (e.g., jamming attacks, sniffer attacks) could disable the links by congesting the network. Further, the data in edge networks are separate into many parts and stored in different storage locations, which may cause data reliability issues [8,20].

To address the inherent drawbacks above, many scholars use blockchain as a building block to integrate with edge computing in IoT systems [22,23,25]. With the blockchain technology, it is possible to build a distributed control at dozens of edge nodes. Thanks to the chain structure and consensus process, blockchain can protect the accuracy, consistency and validity of the collected IoT data transparently. The integration of blockchain and edge computing seems to be a win-win solution which can provide secure and reliable services.

However, the integration of blockchain and edge computing still encounters data storage capacity problem. Although the edge node could offer relatively large storage, as the collected data and transactions increase, the storage required for blockchain is ever-growing. As a result, edge nodes will eventually consume the entire storage. Current Bitcoin chain is more than 225 GB large, in the industrial IoT settings, the size of the chain could be even more significant. To address this problem, approaches such as Ethereum differentiate full node and light node. Only the full node stores the entire chain while the light node only stores the state.

Nevertheless, it brings centralization risk, since the full nodes may be malicious and the light nodes have no way to detect this [14,21]. Also, the edge node must be able to verify transactions and blocks, therefore it should store the full chain. Another solution is to overwrite the original chain directly, but it would break tamper-resistance property of the blockchain. The existing approaches cannot effectively solve storage issues.

To address the storage issue, we propose a redactable and reusable blockchain architecture *with fixed storage* called Re-chain. In collaborative edges, Re-chain allows the re-write operations from the earliest block seamlessly when the edge node reaches the maximum storage size, thus mitigating the storage limitation of the edge node. Moreover, the re-writing process is controlled by a consortium of edge nodes. Only approved by a sufficient number of edge nodes can the new transactions be re-written to the chain. Such a consensus mechanism brings trust to the system and increases the attacking overheads for the adversaries.

Our observation is that data and transactions in IoT scenario are time-sensitive. In specific, data generated by the IoT devices (such as sensor measurements, device logs, monitoring data and environmental data) may lose its value after some time. Based on this observation, Re-chain safely re-writes the new transactions in the earliest blocks in a seamless way. We can further use the cloud as a backup node to store the overwritten blocks, which ensures that the historical data and previous transactions are still accessible and verifiable, but it beyond the scope of this paper. Edge nodes in the near-end stores blocks generated in more recent periods and allow transaction query and verification.

Re-chain uses chameleon hash [15] to enable the re-write operation. The concept of chameleon hash was first proposed by Krawczyk and Rabin [3]. It is a

one-way hash function that contains public key and trapdoor, where hashing is parametrized by public key pk. As long as the trapdoor sk is not known, it is hard to find a collision. Conversely, if the trapdoor sk is known, the arbitrary collision can be efficiently found. However, the chameleon hash cannot be directly used in our scenario since the trapdoor needs to be managed by a centralized party. Instead, we require a consortium of edge nodes to reach consensus before re-writing a block, and the consensus process should be able to tolerate a considerable number of faulty nodes. To meet this gap, we propose threshold trapdoor chameleon hash (TTCH) by using multiple secret keys instead of a single fixed secret key to finding collisions. We incorporate TTCH into our consensus mechanism. Moreover, to achieve a higher performance in the throughput and reduce the large cryptographic overhead incurred during the consensus process, we construct two different hashes in the Merkel tree, TTCH hash function and SHA256, which only needs to calculate once of the TTCH collision during the block re-write process.

The main contributions of this work are summarized as follows:

- We propose a reusable and redactable blockchain called Re-chain, which can re-write the historical data and transaction of the earlier blocks without affecting the integrity of the original chain. Re-chain allows the most recent data and transactions on the blockchain accessible with a fixed size in space, addressing the storage issues for the large scale edge-based IoT system.
- We propose TTCH to achieve a consensus-based re-write operations, which allows a t-out-of-n edge nodes to compute a hash collision collaboratively in order to re-write block transactions.
- We instantiate a prototype implementation of Re-chain and TTCH, and evaluate the performance of the different operations through comprehensive experiments. The results demonstrate that Re-chain is practical when applying to a medium-scale IoT system.

2 Related Work

2.1 Integrated Blockchain and Edge Computing Systems

The study of [16] proposes a permissioned blockchain edge model to address privacy protections and energy security in smart grid, and they use the voting functionality of blockchain to validate the users' identities and smart contract to achieve optimal energy resource management. Sharma et al. [17] presented distributed blockchain cloud architecture with Software-Defined Networking (SDN) enabled edge computing, the SDN controllers based on blockchain is used for a low-latency service of computing resources. Qi et al. [24] use blockchain technology to build Cpds to prevent the participants from acquiring trusted traceability of products in industrial IoT data sharing arena.

In edge computing environments with a massive amount of data collected from IoT devices, the scalability issues hinder the practical feasibility of blockchain-based solutions. In the following, we summarized works on massive

data storage empowered by blockchain. InterPlanetary File System (IPFS) [18] is a decentralized and distributed file system with high integrity and robustness. IPFS runs over a peer-to-peer network to store and share data over the network nodes.

However, none of those above works has explicitly addressed the storage issue of ever-growing blockchain and the reuse of blockchain, existing approaches can only enhance the limited scalability, and security level will decrease accordingly. By contrast, Re-chain can gain storage space by constantly re-writing outdated historical transactions to ensure edge node acting as a full node.

2.2 Redactable Blockchain

Ateniese et al. [15] first proposed redactable blockchain by using chameleon hash (CH) to replace traditional hash function, so that block content can be re-written without causing hard forks. Huang et al. [10] proposed a threshold chameleon hash (TCH) for Industrial Internet-of-Things (IIoT) environment, and it allows a group of authorized sensors to re-write blockchain. However, in the re-write process, TCH requires a ring of k authorized sensors to compute collision one by one, which means, k sensors must behave correctly, and the system cannot tolerate malicious sensors, once a sensor being compromised, the entire system can fail.

In Re-chain, the re-write operation is governed by a consortium of edge nodes, which makes the process more controllable. Further, TTCH find collision through a threshold number of trapdoor keys rather than relies on a single trapdoor key, hence, has better resilience to key compromise. The entire re-write process relies on a consensus protocol similar to Proof of Authority (PoA), which can tolerate faulty nodes. When an edge node leading the re-write process, other edge nodes remain unlinkable, this will not bring higher latency and cost. We believe Re-chain achieves a higher decentralized design and acts more efficient.

3 Background

3.1 System Model

The existing multi-layer edge-IoT system mainly consists of four entities as identified below.

1. *Cloud*: Central cloud in the cloud layer can provide gigantic data storage and computational power. The cloud can back up the blocks which have been overwritten at the edge node in an off-chain manner, therefore it can still recover the overwritten data through the cloud and make the data accessible to the users.
2. *Edge node*: Edge nodes provide fixed storage and computational power which is higher than which of the end device's, but smaller than the cloud. We notice that Re-chain stores at the edge node which has fixed storage. Re-chain is used to record the transactions and data collected from end devices, along with resource management and data processing operations.

3. *Key authority*: Key authority is a trusted entity in the edge layer which is responsible for publishing the genesis block and generating secret keys for edge nodes.
4. *End device*: End devices (e.g., sensors, actuators) are used for collecting data in different circumstances. The storage and computational power are constrained, therefore the end device will send and store the collected data in the edge node.

3.2 Design Goals

Re-chain has the following design goals:

- **Liveness:** Re-chain must guarantee the liveness as long as the minimum number of edge nodes maintains liveness and honest, thus weak synchrony assumptions hold for the key distribution [2].
- **Collision-resistance:**Without knowing the trapdoor of TTCH, no adversary can efficiently find a collision for any pair of $(\hbar^*, R^*, m^*)$ and $(\hbar^*, R'^*, m'^*)$ with PPT algorithm.
- **Unforgeability:** Unforgeability requires that it is even intractable for the adversary who possesses secret keys to find collisions, this definition is stronger than and key-exposure freeness [4,5].
- **Efficiency:** When the space used for Re-chain does not reache the edge node's storage limit, Re-chain uses baseline consensus to write new blocks to the chain, while it reaches the storage limit, Re-chain re-write the historical blocks with our proposed proof-of-concept consensus protocol. The performance of re-writing consensus must be as efficient as basic consensus.

3.3 Threat Model

Our adversary's goal is to find arbitrary collision to break the security of the re-write process. The adversary may hold part of secret keys (less than the threshold) and try to compute collision by invalid credentials. The adversary may take any action within enough secret keys to obtain the hash trapdoor among all edge nodes it controls. We assume that the adversary cannot break standard cryptographic primitives and assumption, such as finding hash collisions or forging digital credentials. The adversary also cannot compromise the private keys of arbitrary domains. In our instantiation, we further assume that the adversary cannot control a majority of edge nodes (more than the threshold) in the blockchain network. We do not consider the privacy and access control of the data stored in the cloud in this paper, as works [9,11,19] are orthogonal with our work and they can integrated into Re-chain.

3.4 Preliminaries

Notations. Let g be a generator of a cyclic group $\mathbb{G}$ of order p a for a λ-bit prime p, an algorithm is efficient if it runs in probabilistic polynomial time (PPT) in

the length of its input. We write the integer modulo p as $\mathbb{Z}_p$ and $r \xleftarrow{R} \mathbb{Z}_p$ denote that r is chosen uniformly at random from $\mathbb{Z}_p$, ab represent the multiplication of two integers $a \in \mathbb{Z}_p$ and $b \in \mathbb{Z}_p$. Let BGGen be a PPT algorithm that returns BG $= (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a computable bilinear map.

Chameleon Hash

Definition 1 (Chameleon Hash). A chameleon hash with message space $\mathcal{M}$ contains five algorithms (HGen, Hash, HVer, HCol) specified as follows:

- HGen(1^λ). The algorithm HGen, on input a parameter λ output a public hash key hk and a secret key sk.
- Hash(m, hk). The algorithm Hash, on input the public hash key hk and a message $m \in \mathcal{M}$ and output the hash value and randomness r.
- HVer($m, hk, (\hbar, r)$). The algorithm HVer, on input $m, \hbar, r$ verify that $(\hbar, r)$ is a valid hash pair for message m.
- HCol($sk, (\hbar, m, r), m'$). The algorithm HCol, on input the secret key and $(\hbar, m, r)$, for a new message $m' \in \mathcal{M}$ return new randomness r' to satisfies

$$\mathrm{HVer(m, hk, (\hbar, r)) = HVer(m', hk, (\hbar, r'))}$$

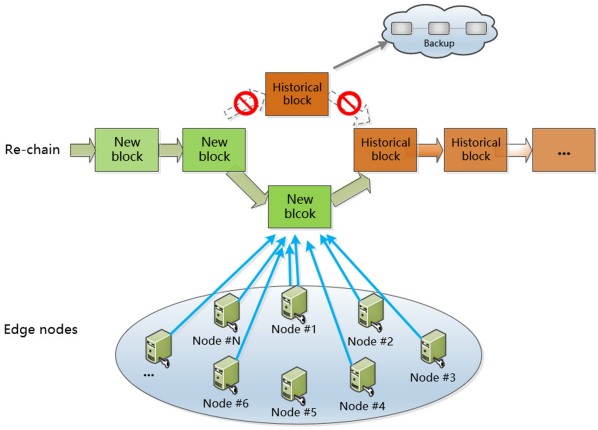


Fig. 1. Re-chain architecture

4 Re-chain Design

4.1 Re-chain Architecture

The core property of Re-chain is the re-writing of the historical transactions when reaching a consensus. We assume that the data collected by the end device

is time-sensitive, and the value of data is reducing as the time passing by. Meanwhile, the data collected earlier is less likely to be accessed by the user. After the IoTs system run a period of time, the storage of the edge node will eventually use up by the Re-chain as the collected data volume increases. The edge node will back up all the data to the cloud in case that the data may be requested in later time if the requested data has been overwritten at the edge node. As shown in Fig. 1, in the re-writing phase, Re-chain re-writes from the earliest block seamlessly, which will maintain the structure of the original chain without affecting the accessibility of the most recent transactions and data. The re-writing phase is controlled by the consortium of edge nodes while not by a single entity, it means that the transactions are validated in the same way as which for the ordinary transactions.

As for the block structure in Re-chain, like what is for most blockchains, each block consists of data records (e.g., transactions), block headers, and the source device signed individual transactions. Blocks are chained together by referencing previous blocks via the inclusion of the hash of the previous block header into the header of the current block. Also, block headers include a timestamp which corresponds to the time window of the current block, and integrity measurements, a Merkle tree [12]. In the Merkle Tree, each leaf node contains the hash of a transaction, while each non-leaf node carries with the hash of the concatenation of its child nodes' hashes. To construct our consensus protocol, we modified the hash used by the Merkle tree, for all intermediate tree nodes, we use SHA256, while for the Merkle root hash, we use our designed TTCH which will be discussed in Sect. 4-B. The Merkle root will then be used to verify the integrity of the data records.

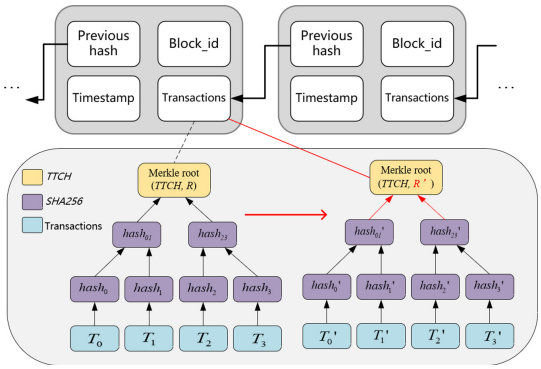


Fig. 2. The structure of blocks used by Re-chain and the Merkle tree update process

4.2 Threshold Trapdoor Chameleon Hash

Building Block of TTCH Scheme. Before giving the full scheme construction, we first recall the public coin chameleon hash proposed by Mojtaba Khalili et al.[1], their hash function satisfies DCDH assumption in the standard model. The scheme works in a billinear group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as described in Sect. 3-B.

- **K.Setup**(1^λ): Choose a billinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order p, where p is a λ-bit prime number. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2)$.
- **K.KeyGen**$(params)$: Choose $x \xleftarrow{R} \mathbb{Z}_p$ and $\widehat{h} \xleftarrow{R} \mathbb{G}_1$, set $h1 = g_1^x$ and $h2 = g_2^x$. Then set $\text{hk} = (h_1, h_2, \widehat{h})$ and $\text{tk} = x$.
- **K.Hash**(m, hk): Select a random number $r \xleftarrow{R} \mathbb{Z}_p$. compute $h = h_1^r \widehat{h}^m$ and $R = h_1^r$.
- **K.HVerify**(m, hk, R): Output $true$ if $e(\hbar/\widehat{h}^m, g^2) = e(R, h_2)$, otherwise output $false$.
- **K.Hcol**(tk, m, m'): For message m', computes new R' as follows:

$$R' = (\frac{\hbar}{\widehat{h}^{m'}})^{\frac{1}{x}}$$

The randomness R in this scheme belongs to a source group of a bilinear pairing and correctness of it can be verified by a pairing product equation. Based on the above scheme, we construct a scheme to support threshold trapdoor aggregate to achieve our design goals as the next section describes.

Construction of TTCH Scheme

- **TT.Setup**$(1^\lambda) \rightarrow (params)$: Choose a billinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order p, where p is a λ-bit prime number. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2)$
- **TT.KeyGen**$(params, t, n) \rightarrow (hk, (sk_1, .., sk_n))$: Pick $x \xleftarrow{R} \mathbb{Z}_p$ and $\widehat{h} \xleftarrow{R} \mathbb{G}_1$, set $h_1 = g_1^x$ and $h_2 = g_2^x$. Compute d s.t. $xd \equiv 1 \bmod p$. Pick a polynomial v of degree $t-1$ with coefficients in $\mathbb{Z}_p$, and set the constant term of v to d, which means $v(0) = d$. Issue to each edge node $i \in [1, ..., n]$ a secret key $sk_i = v(i)$, and public the hash key $hk = (h_1, h_2, \widehat{h})$.
- **TT.Hash**$(m, hk) \rightarrow (\hbar, R)$: On input a hash key hk and message m. Select a random number $r \xleftarrow{R} \mathbb{Z}_p$, compute $h = h_1^r \widehat{h}^m$ and $R = h_1^r$.
- **TT.HVerify**$(m, \hbar, hk, R) \rightarrow (true$ or $false)$: On input a committed message m, chameleon hash $\hbar$, hash key hk and randomness R, check whether $e(h/\widehat{h}^m, g_2) = e(R, h_2)$, if yes, output $true$, otherwise output $false$.
- **TT.Sign**$(sk_i, m', \hbar) \rightarrow (\sigma_i)$: The edge node i parses its secret key sk_i and a new message m', output:

$$\sigma_i = (\frac{\hbar}{\widehat{h}^{m'}})^{sk_i}$$

as a credential of message m'.

– **TT.Hcol**$((\sigma_1, .., \sigma_t), R, hk, \hbar, m) \rightarrow (\perp$ or $R')$: First make a check by running **Hverify**$(m, \hbar, hk, R)$, if check failed then return $\perp$. Otherwise, parse each σ_i for $i \in [1, .., t]$. compute Lagrange coefficient l:

$$l_i = \left[\prod_{j=1, j\neq i}^{t} (0-j)\right]\left[\prod_{j=1, j\neq i}^{t} (i-j)\right]^{-1} \mod \quad p$$

Then compute a new randomness $R' = \prod_{i=1}^{t} \sigma_i^{l_i}$.
Next, check whether equation: $e(\hbar/\widehat{h}^{m'}, g_2) = e(R', h_2)$ holds. If yes, output R', otherwise output $\perp$.

4.3 Consensus Protocol

In order to put the block re-writing operation to the hands of the consortium of edge nodes, we propose a consensus protocol inspired by Proof of Authority (PoA) [6]. Our consensus protocol can tolerate a number of malicious or failure edge nodes. We assume the majority of N edge nodes are honest, which means at least $N/2 + 1$ edge nodes are honest. The edge nodes are responsible for processing data and transactions from the end devices, and also executing the block re-writing operations on Re-chain. The protocol will run by round where in each round an edge node will be selected as proposal node. To prevent a single Byzantine node from attacking the network by imposing a large number of blocks, each edge node is allowed to propose only one block every $N/2 + 1$ blocks.

To run the consensus protocol, the key authority first generates system parameters during the system initialization. Then it sets the threshold parameter of TTCH function as $t = N/2+1$, and runs **TT.KeyGen** to generate N threshold secret keys $(sk_1, ..., sk_N)$ and a hash key hk. The key authority assigns the secret key to the corresponding edge nodes. Each edge node has its own secret key for issuing credential in consensus. We assume that this consensus protocol only applies to the re-writing operation when the storage space of the Re-chain is used up. The detailed process of the consensus protocol is listed in the following.

– *Step 1: Block proposal.* At the very beginning of the process, an edge node generates transactions with the collected data assembled and proposes a block as proposal node. The proposal node will verify the *Block_id* of the re-written block, and then re-write the earliest transactions. To form a new Merkle Tree, the proposed node calculates the hash corresponding to each new transaction and the hash of the intermediate node with SHA256, while the chameleon hash of the Merkle root remains the same as the original block. Therefore, the Merkle root hash cannot be verified by an intermediate node hash at this time. The proposal node then digitally signs the block and use the timestamp to guarantee authenticity and accuracy. Lastly, it broadcasts the proposed block to other edge nodes.

– *Step 2: Credential issuance.* Upon receiving the proposed block, other edge nodes which act as validators will first validate the signature and the *Block_id*, if verification fails, the proposed block will be dropped by the validators. Upon successfully authenticated, the validator node j will run the **TT.Sign** algorithm to sign the Merkle root's child nodes' hashes with its secret key sk_j, as Fig. 2 shows, the credential $\sigma_j = \mathbf{TT.Sign}(sk_j, \hbar'_{01}, \hbar'_{23}, \hbar_{root})$ will be issued by node j and sent to the proposal node, while the validated block will be temporarily saved by node j.
– *Step 3: Computing collision.* After collecting t,i.e. $(N/2+1)$ credentials (Including self-signed credential) $(\sigma_1, ..., \sigma_t)$ from the edge nodes, proposal node enters collision computation phase. By running $\mathbf{TT.Hcol}((\sigma_1, ..., \sigma_t), R, hk, \hbar_{root})$, while keeping the chameleon hash of Merkle root unchanged, the proposal node will calculate a new randomness R', which enables the $\hbar_{root}$ to be verifiable by its child nodes' hash. After that, the proposal node broadcasts the *commit* message with the new randomness R' to all the edge nodes in the network. If the proposal node does not collect t credentials within the pre-defined time limit, this consensus round will be terminated and enter into the next round of consensus.
– *Step 4: Commit.* When a validator receives the *commit* message with randomness R', it will first validate the Merkle root hash by running **TT.Hverify**, if verified, the validator will re-write the original block with the new validated block, and update the randomness corresponding to the Merkle root hash to R', and the consensus on re-writing is reached.

4.4 Security Analysis

The Security of TTCH Collision Resistance. We argue that if an adversary $\mathcal{A}$ can break collision resistance. Adversary $\mathcal{A}$ receive a divisible tuple (g_1^x, g_2^x, g_1^y), set $h_1 = g_1^x$, $h_2 = g_2^x$ and $\widehat{h} = g_1^y$, without knowing x, the $\mathcal{A}$ can find $(\hbar^*, R^*, m^*)$ and $(\hbar^*, R'^*, m'^*)$ as collision where $R^* = g_1^r$ and $R'^* = g_1^{r'}$, obtain the equation:

$$\begin{aligned} h_1^r \widehat{h}^{m^*} &= h_1^{r'} \widehat{h}^{m'^*} \\ \Rightarrow g_1^{xr} g_1^{ym^*} &= g_1^{xr'} g_1^{ym'^*} \\ \Rightarrow g_1^{xr} g_1^{\frac{y}{x}xm^*} &= g_1^{xr'} g_1^{\frac{y}{x}xm'^*} \\ \Rightarrow R^* g_1^{\frac{y}{x}m^*} &= R'^* g_1^{\frac{y}{x}m'^*} \end{aligned}$$

$\mathcal{A}$ divide both sides of the equation and obtain $g_1^{\frac{y}{x}} = (\frac{R'^*}{R^*})^{(\frac{1}{m^* - m'^*})}$. Since divisible CDH in billinear group is hard, so the adversary $\mathcal{A}$ cannot compute x from two collisions, our proposed TTCH is collision-resistant. **Unforgeability:** We consider two possible ways for adversary $\mathcal{A}$ to forge collision.

(1) The adversary $\mathcal{A}$ without valid credentials manages to compute collision. While forged or wrong credentials will not be able to get new random numbers through **TT.Hcol**.

(2) An adversary $\mathcal{A}$ that has successfully collected fewer than t credentials. While running **TT.Hcol** involves performing Lagrange interpolation. If $\mathcal{A}$ has fewer than t partial credentials, then they have fewer than t points, which makes the resulting the $t-1$ degree polynomial undetermined and impossible to compute collision.

The Security of Re-chain. The security of Re-chain is based on the security of TTCH.

(1) A malicious edge node $\mathcal{A}$ may attempt to calculate the hash trapdoor by continuous collecting the re-write operation records. While according to the property of Collision Resistance of TTCH, $\mathcal{A}$ cannot get trapdoor through hash tuples.
(2) A small subset of malicious edge node cannot corrupt re-write operation, since the threshold property of Re-chain implies that the adversary needs to corrupt at least t authorities for this attack to be possible.

5 Implementation and Experiments

In this section, we first implement our construction and present the evaluation result of the concrete TTCH. Then we instantiate a proof-of-concept prototype of Re-chain and evaluate its performance through experiments.

5.1 TTCH Implementation

We implement the construction described in Sect. 4 in python 3.6 using petlib and bplib. The bilinear pairing is defined over the Barreto-Naehrig [13] curve, using OpenSSL as arithmetic backend. All simulations are run on desktop computer with Intel i5-3210M CPU and 2 core processors running at 2.3 GHz and 4-GB RAM with 64-bits Linux system.

We first evaluate **TT.KeyGen**, **TT.Hash**, **TT.HVerify** and **TT.Sign**, we fix the message size to 1 KB, each of our results is taken by a mean of 1000 executions. As shown in Fig. 3, we can see that as the threshold parameter increases, the time spent in Hash and HVerify phases are constant, this is because the calculation process of these algorithms does not involve threshold parameter t. The average time cost of **Sign** increases linearly, it is reasonable as in Sign phase it has to get$t$ credentials, so that the algorithm will run t times with different secret keys. However, the time spent of the KeyGen phase increases as the threshold parameter increases, this is due to the **KeyGen** algorithm has to generate t secret keys.

The performance of **TT.Hcol** algorithm is depicted in Fig. 4(i), we demonstrate the linear relationship between threshold parameter and computing cost through the experiments by setting t value from 5 to 30. As it is shown in the figure, the time spent on running **Hcol** increases linearly when t increases. It is because as the t increases, the algorithm has to compute more parameters when finding the collision.

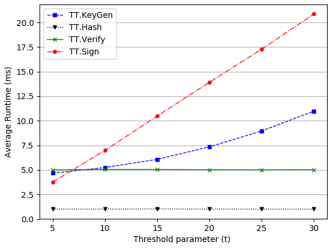
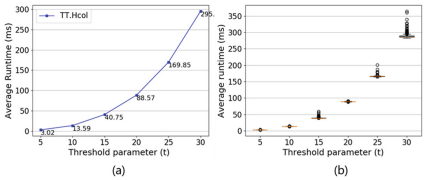
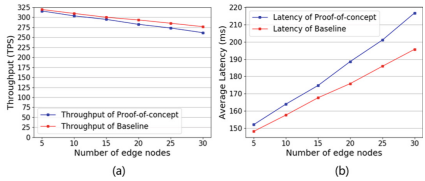


Fig. 3. Average time cost of **KeyGen**, **Hash**, **HVerify** and **Sign** in TTCH



(i) (a)time cost of Hcol. (b)Box-Plots of the Hcol



(ii) (a)throughput. (b) latency

Fig. 4. Performance of Re-chain

5.2 Re-chain Implementation

We instantiate a prototype of Re-chain with python 3.6. To demonstrate the performance of our proposed consensus protocol, We use the following baseline consensus protocol which is used when the space used for Re-chain is not reach the edge node's storage limit. The baseline consensus procedure are as follows: *1) step 1*: The proposal node propose a block and broadcasts it to the rest of edge nodes. *2) step 2:* The edge node will first validate signatures and *Block_id* of the proposed block, if verified, it will accept the proposal block and broadcasts the preset message to all the other edge nodes. *3) step 3:* If the proposal node receive over $N/2$ correct preset messages, the edge node will broadcast a commit message. *4) step 4:* A consensus is reached if the proposal node accepts $N/2+1$ (possibly including its own) commit messages.

We use Aliyun CES as the experimental platform, all nodes are running in the Docker containers on four servers, each server is equipped with two Intel(R) Xeon(R) Platinum 8269CY CPU at 2.50 GHz and 4 GB of RAM, the operating system is 64-bit Ubuntu 18.04.4 LTS. Each block contains 100 transactions. We test the performance of our proposed consensus protocol through the comparison experiments with the baseline consensus protocol. The experiments record the throughput and latency of both protocols with different numbers of nodes varying from 5 to 30. We use the HTTP protocol for the communication between different nodes. And we record the average result out of six tests for each experiment. As shown in Fig. 4(ii)(a), the results show that the throughput for our proposed consensus protocol is about 5% less than which of the baseline when

relatively fewer nodes in the system. However, the difference will become more significant when the number of nodes increases. Figure 4(ii)(b) shows that the latency gap is slightly larger between the proposed consensus and baseline consensus. It is reasonable as our proposed method requires more computational overhead in finding collisions and other cryptographic calculations such as digital signing. Moreover, as the number of nodes increases, the threshold t will also increase, therefore it requires more computational power to execute **TT.Hcol** to reach consensus during re-writing process.

6 Conclusion

In this paper, we proposed TTCH to build a reductable and reusable blockchain called Re-chain at the edge. Through Re-chain we address the storage problem of conventional blockchain caused by ever-growing information chunks. The proposed proof-of-concept consensus is used when Re-chain reaches the maximum storage size, and it empowers Re-chain to re-write historical blocks in a controllable and secure way while maintaining the connectivity of the chain, meanwhile, the consensus process can guarantee the liveness even if there are $N/2 - 1$ faulty edge nodes in the network. Re-chain can be applied to the consortium blockchain-based industrial IoT systems which have storage limitation issues. The experimental results showed that the re-write operation is efficient, and the performance of Re-chain is acceptable at a medium scale (under 30 edge nodes).

Acknowledgment. This research is supported by the National key R&D Program of China under Grant No. 2018YFB1402700. It is also partially supported by the National Natural Science Foundation of China under Grant No. 61672421.

References

1. Khalili, M., Dakhilalian, M., Susilo, W.: Efficient chameleon hash functions in the enhanced collision resistant model. Inf. Sci. **510**, 155–164 (2020)
2. Kate, A., Huang, Y., Goldberg, I.: Distributed key generation in the wild. Cryptology ePrint Archive, Report 2012/377 (2012). https://eprint.iacr.org/2012/377
3. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA (2000)
4. Ateniese, G., de Medeiros, B.: On the key exposure problem in chameleon hashes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 165–179. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30598-9_12
5. Chen, X., Zhang, F., Kim, K.: Chameleon hashing without key exposure. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 87–98. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30144-8_8
6. De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., Sassone, V.: PBFT vs proof-of-authority: applying the CAP theorem to permissioned blockchain. In: Proceedings of ITASEC, pp. 1–11 (2018)

7. Stanciu, A.: Blockchain based distributed control system for edge computing. In: 21st International Conference on Control Systems and Computer Science (2017)
8. Yang, R., Yu, F.R., Si, P., Yang, Z., Zhang, Y.: Integrated blockchain and edge computing systems: a survey some research issues and challenges. IEEE Commun. Surv. Tutor. **21**, 1–25 (2019)
9. Qi, S., Zheng, Y., Li, M., Liu, Y., Qiu, J.: Scalable industry data access control in RFID-enabled supply chain. IEEE/ACM Trans. Netw. **24**(6), 3551–3564 (2016)
10. Huang, K., et al.: Building redactable consortium blockchain for industrial Internet-of-Things. IEEE Trans. Ind. Inform. **15**(6), 3670–3679 (2019)
11. Qi, S., Zheng, Y., Li, M., Lu, L., Liu, Y.: Secure and private RFID-enabled third-party supply chain systems. IEEE Trans. Comput. **65**(11), 3413–3426 (2016)
12. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21
13. Kasamatsu, K.: Barreto-Naehrig curves (2014). https://tools.ietf.org/id/draft-kasamatsu-bncurves-01.html. Accessed 14 Aug 2014
14. Buterin, V.: A next generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper
15. Ateniese, G., Magri, B., Venturi, D., Andrade, E.: Redactable blockchain - or - rewriting history in bitcoin and friends. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P (2017)
16. Gai, K., Wu, Y., Zhu, L., Xu, L., Zhang, Y.: Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. IEEE Internet Things J. **6**(5), 7992–8004 (2019)
17. Sharma, P.K., Chen, M.-Y., Park, J.H.: A software defined fog node based distributed blockchain cloud architecture for IoT. IEEE Access **6**, 115–124 (2017)
18. Benet, J.: IPFS - content addressed, versioned, p2p file system (draft 3). https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf
19. Qi, S., Zheng, Y.: Crypt-DAC: cryptographically enforced dynamic access control in the Cloud. IEEE Trans. Dependable Secure Comput. **16**, 1 (2019)
20. Lu, Y., Qi, Y., Qi, S., Li, Y., Song, H., Liu, Y.: Say no to price discrimination: decentralized and automated incentives for price auditing in ride-hailing services. IEEE Trans. Mob. Comput. (2020). https://doi.org/10.1109/TMC.2020.3008315
21. Xue, S., Zhao, X., Li, X., Zhang, G., Xing, C.: A trusted system framework for electronic records management based on blockchain. In: Ni, W., Wang, X., Song, W., Li, Y. (eds.) WISA 2019. LNCS, vol. 11817, pp. 548–559. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30952-7_55
22. Tuli, S., Mahmud, R., Tuli, S., Buyya, R.: FogBus: a Blockchain-based lightweight framework for edge and fog computing. J. Syst. Softw. **154**, 22–36 (2019)
23. Ayoade, G., Karande, V., Khan, L., Hamlen, K.: Decentralized IoT data management using blockchain and trusted execution environment. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), pp. 15–22. IEEE (2018)
24. Qi, S., Lu, Y., Zheng, Y., Li, Y., Chen, X.: Cpds: enabling compressed and private data sharing for industrial IoT over blockchain. IEEE Trans. Ind. Inform. (2020). https://doi.org/10.1109/TII.2020.2998166
25. Samaniego, M., Deters, R.: Virtual resources & blockchain for configuration management in IoT. J. Ubiquit. Syst. Pervasive Netw. **9**(2), 1–13 (2017)