HIBEChain: A Hierarchical Identity-based Blockchain System for Large-Scale IoT

Zhiguo Wan, Wei Liu, Hui Cui

***Abstract*—Internet-of-Things enables interconnection of billions of devices, which perform autonomous operations and collect various types of data. These things, along with their generated huge amount of data, need to be handled efficiently and securely. Centralized solutions are not desired due to security concerns and scalability issue. In this paper, we propose HIBEChain, a hierarchical blockchain system that realizes scalable and accountable management of IoT devices and data. HIBEChain consists of multiple permissioned blockchains that form a hierarchical tree structure. To support the hierarchical structure of HIBEChain, we design a decentralized hierarchical identity-based signature (DHIBS) scheme, which enables IoT devices to use their identities as public keys. Consequently, HIBEChain achieves high scalability through parallel processing as blockchain sharding schemes, and it also implements accountability by use of identity-base keys. Identity-based keys not only make HIBEChain more user-friendly, they also allow private key recovery by validators when necessary. We provide detailed analysis of its security and performance, and implement HIBEChain based on Ethereum source code. Experiment results show that a 6-ary, (7,10)-threshold, 4-level HIBEChain can achieve 32,000 TPS, and it needs only 9 seconds to confirm a transaction.**

***Index Terms*—Blockchain, IoT, Key Management, Identity-based Signature**

I. Introduction

IoT connects various types of physical devices, ranging from wireless temperature sensors, smart meters, and intelligent appliances to connected vehicles. According to Gartner and ABI Research, over 20 billion IoT devices will be deployed by 2020 [1]. However, secure and efficient management of a large number of devices for such a huge network turns out to be an extremely challenging task.

The traditional approach to manage IoT devices with a centralized server, e.g. a dedicated cloud, has a number of drawbacks. First, the centralized mode leads to the single point of failure, and the whole system breaks down if the centralized server is compromised. This mode also makes the centralized server an attractive attack target. Second, users place too much trust on the centralized server as a third party, which can fully control and access the users' IoT devices. The centralized server could abuse this privilege by arbitrarily controlling IoT devices and obtaining sensitive information from these devices. In addition, centralized servers, e.g. the cloud, may be far from IoT devices, so they are unable to respond quickly to IoT devices. A better strategy is to take a decentralized approach instead of the centralized one like cloud computing.

Recently, the decentralized blockchain technology underlying Bitcoin [2] has attracted a lot of interests from both industry and academia. A blockchain is a completely decentralized ledger without relying on any trusted party. It is formed by connecting a block with its predecessor through hashing. Each block is generated through some consensus mechanism (e.g. proof-of-work or Practical Byzantine Fault Tolerance (PBFT) algorithm [3]) by the participants of the blockchain system. Many blockchain-based cryptocurrencies and systems have been developed following the similar design. More importantly, the blockchain, as a decentralized trustworthy infrastructure, has found much more applications in many areas, including fintech, supply chain, smart manufacturing and IoT systems.

Blockchain has a number of attractive features that make it well-suited for IoT. First, blockchain is decentralized without a single point of failure, so it is more secure and robust than traditional centralized solutions. Therefore, blockchain can be used as a decentralized control center for IoT devices, which take commands from the blockchain instead of a centralized server. This can effectively prevent emerging attacks aiming to take control of these IoT devices. Second, blockchain enables self-enforceable smart contracts, which facilitate cooperation among IoT devices although they may not fully trust each other (e.g. robots owned by different organizations collaborate to accomplish some task). Furthermore, blockchain-based cryptocurrency offers a convenient payment method, and it can also be used as an incentive mechanism for autonomous IoT devices.

There are a number of projects attempting to apply blockchain to IoT systems. A framework ADEPT by IBM [4] employs the blockchain technology to realize a decentralized, autonomous data and device control system for large-scale IoT networks. IBM also integrates blockchain with its Watson IoT platform [5], so that IoT data can be shared with a private blockchain. Supply chain tracing can also be implemented using blockchain on Watson IoT platform. China Unicom, ZTE corporation, Alibaba and MIIT [6] have recently proposed a blockchain framework of things as a decentralized service platform, aiming to improve IoT applications and services. It is expected that blockchain can help to improve trust, accountability, transparency and efficiency of IoT systems. A number of IoT-oriented blockchain projects have been initiated in recent years, such as Filament [7] and IOTA [8].

Unfortunately, blockchain cannot be directly applied on

Z. Wan and W. Liu are with the School of Computer Science and Technology, Shandong University, Qingdao, China.

H. Cui is with Discipline of Information Technology, Mathematics and Statistics, Murdoch University, Perth, Australia and Data61, CSIRO, Melbourne, Australia.

large-scale IoT systems due to its poor scalability, i.e. low transaction throughput. Currently, the Bitcoin blockchain can only process at most 7 TPS (transactions per second), and Ethereum can only process at most 25 TPS. Some improvements have been proposed for Bitcoin blockchain, e.g. increasing the block size from 1MB to 20 MB, but the improvement is very limited.

Transaction throughput is not the only issue for applying the blockchain technology in IoT systems. The enormous transaction data generated by the IoT cannot be processed efficiently by current blockchain systems. Currently, mainstream blockchain systems like Bitcoin and Ethereum use a single blockchain to manage transactions, which leads to difficulty in managing large amount of transaction data. For instance, the Bitcoin blockchain is of size around 200GB now, and every full node needs to store a complete copy of the blockchain. Clearly, an efficient blockchain system is required to deal with large volume IoT transaction data.

More importantly, cryptographic keys are not well managed in current blockchain systems, which results in malicious abuses and difficulties in blockchain management. In public blockchain systems like Bitcoin, participants generate their cryptographic keys by themselves, which are not bound to their real identities. Indeed, Bitcoin has been used for illegal activities, including money washing, drug trading, ransomwares etc. Such an approach achieves strong anonymity, but it could be abused by adversaries to launching attacks. For the sake of security and accountability, it is important for a blockchain-based IoT system to have an efficient and secure identity management scheme.

To tackle the above problems, we propose HIBEChain, a hierarchical permissioned blockchain system, with the aim to achieve efficiency, scalability and accountability for IoT systems. HIBEChain consists of multiple blockchains forming a hierarchical tree structure, with each blockchain being a node on the tree structure. Each leaf blockchain is independent and needs not to synchronize with other blockchains. It is managed by edge servers as the miners (also called validators), which process transactions generated by IoT devices within its domain. Intermediate blockchains do not store transaction from end devices, but only confirm blocks of their child blockchains. Similar to the hierarchical domain name system (DNS), HIBEChain is highly scalable and well-suited for large-scale IoT systems.

As a permissioned blockchain, HIBEChain adopts a decentralized hierarchical identity-based key management (DHIBS) scheme, which achieves secure and user-friendly key management. For IoT devices, the identities are unique and meaningful strings, instead of random bit strings. Using these identities as public keys, HIBEChain is much more user-friendly and public key verification is not required anymore. Meanwhile, malicious abuses can be effectively prevented as identities can be easily traced.

Our contributions can be summarized as follows:

- We propose HIBEChain, a hierarchical blockchain scheme, to implement decentralized, scalable and efficient management for IoT systems. To the best of our knowledge, this is the first IoT-oriented hierarchical blockchain system that contains multiple blockchains. HIBEChain is well-suited for processing large-scale transactions and data for IoT systems.
- In accordance with HIBEChain, we design an efficient and scalable signature scheme DHIBS based on the hierarchical identity-based encryption algorithm. At every level of HIBEChain, identity-based private key shares are generated cooperatively by a group of validators, instead of a single validator. DHIBS and the PBFT consensus algorithm of the blockchain can be seamlessly integrated together by using the same threshold.
- We provide a formal security model and a rigorous security proof for the DHIBS scheme. In addition, we provide in-depth discussion on consistency and liveness of our PBFT-based consensus mechanism used in HIBEChain.
- We fully implement HIBEChain as well as the digital signature scheme DHIBS based on Ethereum. We then conduct comprehensive experiments to evaluate its performance. Experiment results show that a 4-level HIBEChain can achieve 32,000 TPS and 9 seconds transaction processing latency.

The reminder of this paper is structured as follows. We review related work on blockchain in the next section, and then provide preliminaries on cryptographic techniques in Section III. We then present a decentralized hierarchical identity-based signature scheme to be used in HIBEChain in Section IV. We describe HIBEChain in detail in Section V, and analyze its properties including scalability, accountability and security in Section VI. Implementation and experimental results are presented in Section VII, followed by concluding remarks in the end.

II. Related work

A. Blockchain Consensus and Sharding

Mainstream consensus mechanisms for blockchains include proof-of-work (PoW), proof-of-stake (PoS), directed acyclic graphs (DAG) and Byzantine fault tolerance (BFT). Consensus *consistency* and *liveness* are two key properties to ensure security and robustness for blockchains.

Bitcoin and Ethereum [9] adopt the PoW mechanism to establish consensus. Analysis of [10] shows that Bitcoin PoW mechanism satisfies consistency and liveness. PoW mechanisms have very low transaction throughput (7 and 25 TPS for Bitcoin and Ethereum respectively). To solve the scalability problem and achieve high transaction throughput, more efficient consensus algorithms have been proposed. Proof-of-stake (PoS) consensus algorithm [11] demands peers of the blockchain to stake their assets to generate the next block, so the computation cost is significantly reduced. Delegated PoS (DPoS) consensus algorithm [12] extends PoS by selecting a group of delegates to execute PoS algorithm, so as to improve blockchain throughput. DAG is designed for IoT-oriented blockchain systems [8], but its security has not been rigorously proved.

Different from PoW/PoS algorithms proposed for public blockchains, permissioned blockchains have different consensus algorithms. Practical Byzantine Fault Tolerance (PBFT)

algorithm [3] is used in Hyperledger Fabric [13], and its safety (consistency) and liveness have been rigorously proved. Permissioned blockchain Ripple [14] and Stellar [15] also have their own consensus algorithms.

To improve transaction processing throughput, sharding mechanisms have been proposed for general-purpose blockchains. It is done by dividing transactions into mutually disjoint partitions and processing these partitions in parallel. Luu et al. [16] proposed Elastico, a sharding scheme for public blockchains like Bitcoin. Each shard is processed by a committee with a consensus algorithm like PBFT, in parallel with other shards. Then the consensus results from all committees are processed by a final committee to reach final consensus. Ethereum [17] also intends to adopt a PoS algorithm Casper with sharding to increase throughput.

RSCoin [18] employs a similar sharding approach, but in a centralized manner. In RSCoin, a mintette is responsible for collecting transactions from end users and collating them into lower-level blocks, while a central bank collects lower-level blocks from these mintettes to produce higher-level blocks and the blockchain.

A recent proposal RapidChain [19] achieves significantly higher throughput by sharding public blockchains, and it tolerates Byzantine faults up to 1/3 of all participants. A shard is formed by randomly selecting members for each committee, so that the fraction of corrupted members of each committee is less than 1/2. A synchronous Byzantine consensus protocol is then executed in each committee to reach consensus.

Another sharding scheme OmniLedger [20] achieves high throughput and resilience against corruption up to 1/4 of all participants for public blockchains. OmniLedger combines verifiable random function and public-randomness protocol to randomly select validators for each shard.

Parallel-chains [21] provide a general composition approach for sharding PoS and PoW blockchains. The idea is for the same validators (miners) to parallelly maintain multiple disjoint blockchains, which are then merged into the final blockchain. It has been rigorously proved that parallel-chains are robust against adaptive corruptions and can achieve near optimal throughput. However, parallel-chains do not shard communication or storage as RapidChain does, so its scalability is limited in this regard.

If the above blockchain sharding schemes are used in IoT systems, the randomly-chosen validators may be far from the IoT devices that generate transactions. Thus it will be hard for the validators to respond to IoT devices timely.

B. Blockchain and IoT

IBM's ADEPT project [4] investigated how to employ blockchain to manage large-scale IoT systems early in 2015. Since then, many blockchain startups have proposed different ideas and solutions to integrate blockchain with IoT systems [22]. But these solutions lack theoretic foundations and rigorous analysis, so their security is still in question. For example, IOTA [8] has been questioned by researchers [23].

In recent years, how to adopt blockchain in IoT systems has attracted attention of academic researchers, but research in this direction is still in the early phase. Some research works focus on potential opportunities and challenges in adopting blockchain in IoT systems [24], some describe a blockchain architecture or platform for specific IoT systems [25], and some others study specific research problems. For example, Boudguiga et al. [26] employ blockchain to design an IoT software/firmware update scheme, which is very important in the lifetime of IoT systems. Hammi et al. [27] design an efficient decentralized authentication scheme for IoT devices using blockchain.

All the above works demand a scalable and highly efficient blockchain system, which is a challenging problem our paper attempts to address.

C. Hierarchical Identity-based Encryption and Decentralized Key Generation

The core component of our proposed scheme is a decentralized hierarchical identity-based signature scheme (DHIBS), which is related to hierarchical identity-based encryption (HIBE) and decentralized key generation. Here we review a number of schemes that are closely related to our proposal. HIBE schemes generalize from IBE to realize an organizational hierarchy. Boneh et al. [28] proposed a HIBE scheme with constant size ciphertexts, which is the starting point of this work. But this HIBE scheme is not a signature scheme, as required by our proposal. To realize hierarchical identity-based signature (HIBS), Chow et al. [29] has proposed an efficient scheme. However, both the HIBE scheme by Boneh et al. and the HIBS scheme by Chow et al. are in centralized mode, not fitting into the decentralized scenario of blockchain.

To decentralize identity-based encryption (IBE), several schemes with distributed private-key generation have been proposed in [30]. In these decentralized IBE schemes, only the master key is shared among a set of authorities, and they just need to generate a private key for an end user. However, for a decentralized HIBS scheme, the private key at each level is required to be shared among the corresponding authorities. To the best of our knowledge, there is no DHIBS scheme proposed in the literature, and how to design such a scheme and prove its security remains an open problem.

III. Preliminaries

A. Bilinear Map and Bilinear Diffie-Hellman Assumptions

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p, g be a generator of $\mathbb{G}$. Then an efficiently computable bilinear map e is defined as $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$.

Computational Bilinear Diffie-Hellman (BDH) Assumption. Let $g \in \mathbb{G}$, and $\alpha, \beta, \gamma \in \mathbb{Z}_p^*$ where p is a prime number, the BDH problem is defined as follows:

$$\text{BDH} : \text{given } g, g^{\alpha}, g^{\beta}, g^{\gamma}, \text{compute } e(g,g)^{\alpha\beta\gamma} \in \mathbb{G}_1.$$

Computational Bilinear Diffie-Hellman Inversion (BDHI) Assumption. Let $g \in \mathbb{G}$ be a generator and $\beta \in \mathbb{Z}_p^*$, the l-BDHI is defined as follows:

$$l\text{-BDHI} : \text{given } g, g^{\beta}, g^{(\beta^2)}, ..., g^{(\beta^l)}, \text{compute } e(g,h)^{1/\beta}.$$

Computational weaker Bilinear Diffie-Hellman Inversion (wBDHI) Assumption. Let $g, h \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, the weak BDHI assumption is defined as follows:

l-wBDHI : given $g, h, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^l)}$, compute $e(g,h)^{1/\alpha}$.

Computational wBDHI* Assumption. Let $g, h \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$, the weak BDHI assumption is defined as follows:

l-wBDHI* : given $g, h, g^{\alpha}, g^{(\alpha^2)}, ..., g^{(\alpha^l)}$, compute $e(g,h)^{(\alpha^{l+1})}$.

B. Hierarchical Identity-based Signature

Below we briefly describe the hierarchical identity-based signature (HIBS) scheme, which is built on the hierarchical identity-based cryptosystem proposed in [28] and complemented with the signature generation technique in [31].

Let $\mathbb{G}$ be a bilinear group of prime order p and e is a bilinear map. Let the identity ID of depth k be $(ID_1, ID_2, ..., ID_k)$, where $ID_i \in \{0,1\}^*$, and $I_i = H(ID_i)$ for $i \in [k]$. $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a cryptographic hash function.

- HIBS.Setup(l): For a given maximum depth l, the public parameters are $params = (g, g_1, g_2, g_3, h_1, ..., h_l)$, and the master secret is $mk = g_2^{\alpha}$. Here $g, g_2, g_3, h_1, ..., h_l$ are random elements in $\mathbb{G}$, and $g_1 = g^{\alpha}$ where α is a random element in $\mathbb{Z}_p^*$.
- HIBS.KeyGen($d_{\mathsf{ID}|_{k-1}}, \mathsf{ID}|_k$): The private key for $\mathsf{ID}|_{k-1} = (ID_1, ID_2, ..., ID_{k-1})$ is

$$\begin{aligned} d_{\mathsf{ID}|_{k-1}} &= (g_2^{\alpha} \cdot (h_1^{I_1} \cdots h_{k-1}^{I_{k-1}} \cdot g_3)^r, g^r, h_k^r, ..., h_l^r) \\ &= (a_0, a_1, b_k, ..., b_l), \end{aligned}$$

  where $I_i = H(ID_i)$ for $i \in [k]$ and $r \in \mathbb{Z}_p^*$ is a random number. Then the private key for $\mathsf{ID}|_k = (ID_1, ID_2, ..., ID_k)$ is generated as:

$$\begin{aligned} d_{\mathsf{ID}|k} &= (a_0 \cdot b_k^{I_k} \cdot (h_1^{I_1} \cdots h_k^{I_k} \cdot g_3)^t, a_1 \cdot g^t, b_{k+1} \cdot h_{k+1}^t, \\ &\quad ..., b_l \cdot h_l^t) \\ &= (g_2^{\alpha} \cdot (h_1^{I_1} \cdots h_k^{I_k} \cdot g_3)^{r+t}, g^{r+t}, h_{k+1}^{r+t}, ..., h_l^{r+t}). \end{aligned}$$

- HIBS.Sign($d_{\mathsf{ID}|_k}, M$): To sign a message $M \in \{0,1\}^*$ with the private key $d_{\mathsf{ID}|_k} = (a_0, a_1, b_{k+1}, ..., b_l)$, it chooses a random number $s \in \mathbb{Z}_p^*$, computes $h = H'(M)$ and $x = a_0 \cdot h^s$ where $H' : \{0,1\}^* \rightarrow \mathbb{G}$ is a cryptographic hash function. Then it obtains $y = a_1$, $z = g^s$, and outputs the signature $\sigma = (x, y, z, y_{k+1}, ..., y_l)$, where $y_i = b_i$ for $i = k+1, \ldots, l$.
- HIBS.Verify($\mathsf{ID}|_k, M, \sigma$): To verify a signature $\sigma = (x, y, z, y_{k+1}, \ldots, y_l)$, it computes $h = H'(M)$ and verifies the following equation:

$$e(g,x) = e(g_1, g_2) \cdot e(y, h_1^{I_1} \cdots h_k^{I_k} \cdot g_3) \cdot e(z, h).$$

The security of the above HIBS scheme is similar to that in [28], which is sketched in the Appendix.

C. Pedersen's Verifiable Secret Sharing

Pedersen's verifiable secret sharing scheme [32] enables n participants to obtain a share x_i corresponding to a (t,n)-secret sharing of a random value x. At the end of the protocol, the random value x is kept secret from all participants, and x_i is known by participant i only. Pedersen's VSS scheme does not need any trusted third party and it is completely decentralized. More importantly, the participants can verify validity of received shares, so as to detect invalid messages sent by malicious participants. Later we will use Pedersen's VSS scheme as a building block in our protocol, and use (t,n)-Pedersen-VSS to denote a Pedersen's VSS scheme for threshold (t,n).

IV. DHIBS: Decentralized Hierarchical Identity-based Signature

In this section, we describe DHIBS that implements a decentralized hierarchical identity-based cryptosystem. DHIBS is used in our hierarchical blockchain system to realize efficient key management and block consensus for IoT devices. Different from aforementioned HIBS schemes, the master key and each private key of DHIBS are shared among a set of authorities, instead of being held by a single authority. Holding only a secret share (referred to as shadow from now on), an authority can generate either a partial signature or a private key share. The higher-level authorities can collaborate to generate (authorize) shadows for authorities at the lower level, and the IoT devices can obtain their private identity-based keys from their direct parent authorities.

However, it is challenging to generate lower-level secret shadows from higher-level shadows in a decentralized manner without reconstructing the private key, because the higher level may have a different threshold value than the lower level. It is important to note that generating private keys from shadows is much easier, since there is no need for re-randomization which is required for shadow generation.

TABLE I
Notations for DHIBS

$\mathsf{ID}\|_k$	Hierarhical identity for level k, i.e. $(ID_1, ID_2, ...ID_k)$, where $ID_i \in \{0,1\}^*$
(t_k, n_k)	Threshold secret sharing parameter for level k
$\mathcal{T}_k$	Set of indexes of k-level authorities of size t_k
$L(i)$	Lagrange interpolation coefficient, $\prod_{j \neq i, j \in \mathcal{T}} \frac{j}{j-i}$ for some $\mathcal{T}$
$d_{\mathsf{ID}\|_k}^{(i)}$	Key share of authority with index i and identity $\mathsf{ID}\|_k$
$d_{\mathsf{ID}\|_k}^{(i,j)}$	Key share generated by authority i for j with identity $\mathsf{ID}\|_k$
$v_{\mathsf{ID}\|_k}^{(j)}$	Verification key for partial signature by authority j with identity $\mathsf{ID}\|_k$
$[\![x]\!]_{t_k}^i$	Threshold secret share for authority i with threshold t_k

A. The Key Technique: Randomize-then-Reconstruct

It is straightforward to reconstruct the private key at the higher level and then generate shadows for lower-level authorities. But this will lead to disclosure of the private key of the higher level, thus losing the advantage of decentralization. In DHIBS, we re-randomize the shadow at the higher level and then reconstruct the private shadow at the lower level, so the generated partial shadow is masked by random numbers.

A valid HIBS private key is in the form of

$$d_{\mathsf{ID}|_{k-1}} = (g_2^{\alpha} \cdot (h_1^{I_1} \cdots h_{k-1}^{I_{k-1}} \cdot g_3)^r, g^r, h_k^r, ..., h_l^r),$$

DHIBS.Setup(l, n_0, t_0)

1. Select $\mathbb{G}, \mathbb{G}_1, p$ and g, where p is the group order and g is the generator;
2. Root authorities follow (t_0, n_0)-Pedersen-VSS to share a master secret α. Root authority i obtains its secret shadow α_i of α, denoted as $\alpha_i = [\![\alpha]\!]^i_{t_0}$, and its master secret shadow is generated as $d^{(i)}_{\mathsf{ID}|_0} = (g_2^{\alpha_i}, 1, 1, ..., 1) \in \mathbb{G}^{l+2}$. Each authority also generates and pubishes a public verification key $v^{(i)}_{\mathsf{ID}|_0} = g^{\alpha_i}$.
3. Generate public parameters $params = (g, g_1, g_2, g_3, h_1, \cdots, h_l)$ where $g_1 = g^{\alpha}$ and $g_2, g_3, h_1, \cdots, h_l$ are randomly selected from $\mathbb{G}$.

DHIBS.PartialShadowGen$(d^{(i)}_{\mathsf{ID}|_{k-1}}, ID_k, j, \mathcal{T}_{k-1})$

1. All $(k-1)$-level authorities from $\mathcal{T}_{k-1}$ follow (t_{k-1}, n_{k-1})-Pedersen-VSS to share a secret r', with authority i holding a private share r'_i. According to Pedersen-VSS, r' is kept secret from everyone while only authority i knows r'_i.
2. All $(k-1)$-level authorities from $\mathcal{T}_{k-1}$ determine the threshold t_k for $\mathsf{ID}|_k$, select $t_k - 1$ coefficients $c_1, c_2, \cdots, c_{t_k-1} \in \mathbb{Z}_p$ for $f(x) = c_1 x + c_2 x^2 + \cdots + c_{t_k-1} x^{t_k-1}$, and compute $f(j)$. The authority also computes $g^{f(j)}$ for verification key calculation.
3. Authority i, holding a private shadow $d^{(i)}_{\mathsf{ID}|_{k-1}} = (a_0, a_1, b_k, \cdots, b_l)$, computes $d^{(i,j)}_{\mathsf{ID}|_k} = (a_0 g_2^{f(j)} \cdot (h_1^{I_1} \cdots h_{k-1}^{I_{k-1}} g_3)^{f(j)} (h_1^{I_1} \cdots h_k^{I_k} g_3)^{r'_i} b_k^{I_k} (h_k^{I_k})^{f(j)}, a_1 g^{f(j)} g^{r'_i}, b_{k+1} h_{k+1}^{f(j)} h_{k+1}^{r'_i}, \cdots, b_l h_l^{f(j)} h_l^{r'_i})$.
4. Authority i outputs $d^{(i,j)}_{\mathsf{ID}|_k}$ and $g^{f(j)}$.

DHIBS.ShadowRecon$(\{d^{(i,j)}_{\mathsf{ID}|_k}\}_{i \in \mathcal{T}_{k-1}}, g^{f(j)})$

1. The j-th authority at k-level uses Lagrange interpolation to reconstruct its secret shadow. From $d^{(i,j)}_{\mathsf{ID}|_k} = (a_0^{(i)}, a_1^{(i)}, b_{k+1}^{(i)}, ..., b_l^{(i)}), i \in \mathcal{T}_{k-1}$, the j-th authority computes $d^{(j)}_{\mathsf{ID}|_k} = (\prod_{i \in \mathcal{T}_{k-1}} (a_0^{(i)})^{L(i)}, \prod_{i \in \mathcal{T}_{k-1}} (a_1^{(i)})^{L(i)}, \prod_{i \in \mathcal{T}_{k-1}} (b_{k+1}^{(i)})^{L(i)}, ..., \prod_{i \in \mathcal{T}_{k-1}} (b_l^{(i)})^{L(i)}) = (g_2^{\alpha''} (h_1^{I_1} \cdots h_k^{I_k} g_3)^{r''}, g^{r''}, h_{k+1}^{r''}, \cdots, h_l^{r''})$, where $\alpha'' = [\![\alpha]\!]^j_{t_k}$ and $r'' = [\![r_{k-1} + r']\!]^j_{t_k}$.
2. The j-th authority at k-level computes and publishes the verification key $v^{(j)}_{\mathsf{ID}|_k} = g^{\alpha''} = g^{\alpha + f(j)}$.

DHIBS.ShadowSign$(d^{(i)}_{\mathsf{ID}|_k}, M)$

1. The k-level authorities follow (t_k, n_k)-Pedersen-VSS to generate a random number $s \in \mathbb{Z}_p$, and each holds a secret share $s_i = [\![s]\!]^i_{t_k}$. They also compute $h = H'(M)$.
2. Authority i, with private share $d^{(i)}_{\mathsf{ID}|_k} = (a_0, a_1, b_{k+1}, ..., b_l)$, computes $x_i = a_0 \cdot h^{s_i}$, $y_i = a_1$ and $z_i = g^{s_i}$.
3. Output the partial signature $\sigma_i = (x_i, y_i, z_i)$.

DHIBS.PartialSigVerify$(\sigma_i, v^{(i)}_{\mathsf{ID}|_k})$

1. Parse σ_i as (x_i, y_i, z_i), and check if $e(g, x_i) = e(v^{(j)}_{\mathsf{ID}|_k}, g_2) e(y_i, h_1^{I_1} \cdots h_k^{I_k} g_3) e(z_i, h)$.

DHIBS.SignRecon$(\{\sigma_i\}_{i \in \mathcal{T}})$

1. Use Lagrange interpolation to reconstruct the final signature $\sigma = (\prod_{i \in \mathcal{T}_k} (x_i)^{L(i)}, \prod_{i \in \mathcal{T}_k} (y_i)^{L(i)}, \prod_{i \in \mathcal{T}_k} (z_i)^{L(i)}) = (x, y, z)$.

DHIBS.Verify$(\mathsf{ID}|_k, M, \sigma)$

1. For a signature (x, y, z), compute $h = H'(M)$ and verify the following equation: $e(g, x) = e(g_1, g_2) \cdot e(y, h_1^{I_1} \cdots h_k^{I_k} \cdot g_3) \cdot e(z, h)$.

Fig. 1. The Decentralized HIBS Scheme

so we use the Pedersen-VSS scheme to share it among a group of authorities. Authority i's share (shadow) is

$$d^{(i)}_{\mathsf{ID}|_{k-1}} = (g_2^{\alpha^*_i} \cdot (h_1^{I_1} \cdots h_{k-1}^{I_{k-1}} \cdot g_3)^{r^*_i}, g^{r^*_i}, h_k^{r^*_i}, ..., h_l^{r^*_i}),$$

where α^*_i and r^*_i are shares of α and r respectively, with regard to the same threshold value t_{k-1}.

The key idea is for each authority at level $k-1$ to generate a partial shadow, i.e. *a share of its shadow*, with regard to the threshold t_k of level k. For instance, the share of α^*_i for authority j at level k is generated as $\alpha^*_i + f(j)$ where $f(j) = c_1 x + c_2 x^2 + ... + c_{t_k-1} x^{t_k-1}$ for some randomly chosen coefficients $c_1, ...c_{t_k-1}$. After the lower-level authority j receives enough partial shadows, it can reconstruct its own shadow

$$\alpha''_j = \sum_i L(i)(\alpha^*_i + f(j)) = \alpha + f(j), \tag{1}$$

where $L(i) = \prod_{j \neq i} \frac{j}{j-i}$ is the Lagrange coefficient for level k. Equation (1) holds because of an important property of Lagrange coefficients, i.e. $\sum_i L_k(i) = 1$.

Since $f(j)$ is known by other authorities, it is crucial to protect the private shadow as well as α^*_i from others. HIBS protect the private keys by re-randomization using a random number, and we decentralize this process and protect α^*_i using a share of the random number.

Finally, the shadow of authority j at k-level can be reconstructed following the above approach:

$$d^{(j)}_{\mathsf{ID}|_k} = (g_2^{\alpha''_j} \cdot (h_1^{I_1} \cdots h_k^{I_k} \cdot g_3)^{r''_j}, g^{r''_j}, h_{k+1}^{r''_j}, ..., h_l^{r''_j}),$$

where α''_j and r''_j are shares of α and r' respectively, with regard to the threshold value t_k. As a result, DHIBS realizes decentralized private key resharing from shares without recovering the private key. It enables HIBEChain to manage identity-based keys and carry out cryptographic operations in a decentralized way.

B. The Construction of DHIBS

DHIBS consists of the following 7 algorithms, while the concrete construction is given in Fig. 1:

- DHIBS.Setup: This algorithm generates public parameters and master secret shares (shadows) for authorities

at the root for a given threshold and a given maximum depth. The master secret is shared between root authorities, and it is unknown to any coalition of authorities less than the threshold.

- DHIBS.PartialShadowGen: This algorithm enables an upper-level authority to generate a *partial* shadow for a lower-level authority. With multiple partial shadows from a set of indexes of upper-level authorities, denoted $\mathcal{T}$, a lower-level authority can reconstruct its complete shadow. The size of $\mathcal{T}$ is exactly the threshold value of the upper level.
- DHIBS.ShadowRecon: This algorithm enables an authority that has obtained all partial shares from the set of upper-level authorities $\mathcal{T}$ to reconstruct its complete secret share.
- DHIBS.ShadowSign: This algorithm enables an authority holding a secret shadow to generate a partial signature on M.
- DHIBS.PartialSigVerify: This algorithm verifies a partial signature on M generated by an authority.
- DHIBS.SignRecon: This algorithm reconstructs a complete signature from multiple partial signatures.
- DHIBS.Verify: This algorithm verifies a complete signature reconstructed from partial signatures, same as the verification algorithm presented in the last section.

The security proof of the above DHIBS scheme is provided in Section VI.

V. HIBEChain: Hierarchical Identity-based Blockchain System

In this section we first provide an overview of HIBEChain, and then present the system model and security model. After that, we describe the design of HIBEChain, including its hierarchical system structure, transaction management, and hierarchical identity management.

A. Overview of HIBEChain

We abandon the single blockchain design for the entire IoT system, which is the approach taken by current solutions. Instead, we propose a hierarchical blockchain structure for IoT, inspired by the hierarchical systems like DNS and PKI. HIBEChain aims to achieve scalable, real-time and autonomous device and data management for IoT systems.

As illustrated in Fig 2, HIBEChain is a tree structure of blockchains, very similar to the tiered Internet structure. Each blockchain is given an ID except the root blockchain, and its private key is shared via (t, n)-threshold secret sharing among a group of validators. For example, the blockchain with ID "Beijing" at level 1 is maintained by 13 validators, who share the private key through (9,13)-threshold secret sharing scheme. That is, 9 out of these 13 validators can recover the private key corresponding to "Beijing". IoT devices, like the one with ID "Beijing, Haidian, ID_1", have a complete private key corresponding to their IDs.

HIBEChain involves two different participants, whose roles and responsibilities are explained as follows:

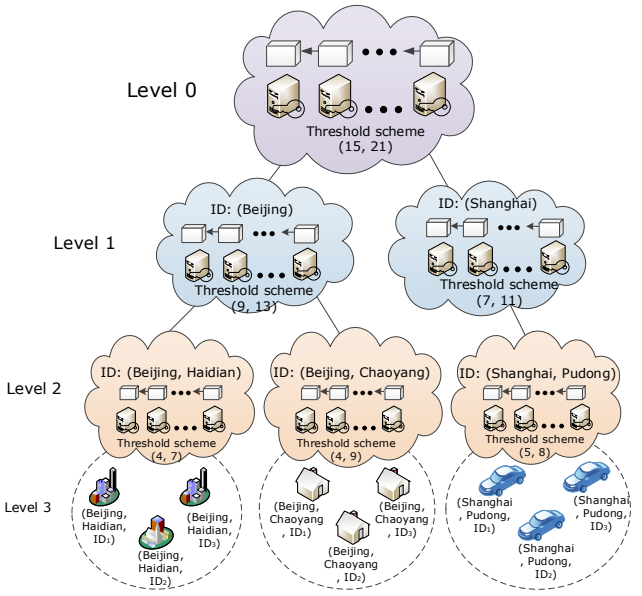


Fig. 2. An illustrated example of HIBEChain of depth 3. Each blockchain has an ID except the root blockchain, and its private key is shared via (t, n)-threshold secret sharing among a group of validators. Each IoT device has a complete private key corresponding to its identity.

- *Validators*: Each leaf blockchain is managed by a group of validators, like the miners in Bitcoin or Ethereum. They are edge servers of the edge computing and they are close to IoT devices, so they can respond to IoT devices quickly. Their responsibility is to verify transactions and maintain the decentralized consensus ledger. Validators of higher-level blockchains also serve as authorities to generate identity-based key shares for validators of lower-level blockchains.
- *IoT Devices*: They are clients at the bottom level of the hierarchy, and it does not store the blockchain data. They register with leaf blockchains with their identities, and then obtain corresponding private keys. Each IoT device is assumed to have a credential associated with their unique identities issued from some authority. IoT devices interact the corresponding leaf blockchain by sending transactions to and taking commands from the leaf blockchain.

HIBEChain relies on DHIBS to manage identity-based keys. This function of validators is equivalent to that of authorities in DHIBS. The root validators (validators at the root blockchain) of HIBEChain establish and share a master secret using Pedersen-VSS, but none of them know the master secret. Then they use their secret shares to generate identity-based private key shares (shadows) for each second-level validator without recovering the master secret. These validators can in turn generate private shadows for even lower-level validators, and so on and so forth. It is important to note that the master secret and private keys are never recovered by validators in this process.

Each leaf blockchain is in charge of IoT devices within a local domain, e.g. a smart campus, a smart community or even

a smart city. Periodically, the validators of a leaf blockchain collect and verify transactions submitted by things within the domain, and run the PBFT consensus algorithm to create a new block for the blockchain. During execution of PBFT, they also broadcast partial signatures over the new block, and enough partial signatures (more than the threshold value) can be combined into a complete signature. Then the signed block header is submitted to the higher blockchain as a transaction, which is called header transaction. The higher-level blockchain checks the signed header transactions sent from lower-level blockchains, and run the PBFT consensus algorithm to form a higher-level block from these block headers. This process continues until the root validators reach consensus.

As such, each leaf blockchain only process and record a very small fraction of transactions of the whole IoT, while non-leaf blockchains do not process transactions from IoT devices. A large number of transactions are distributed among different blockchains, instead of recording everything in a single blockchain. For this approach, it is critical to keep consistency of transactions among blockchains. In HIBEChain, we design a special mechanism to ensure consistency like the single blockchain system.

HIBEChain relies on DHIBS to manage identity-based keys and uses them to realize scalable and accountable blockchain management with 5 phases: system initialization, validator authorization, IoT device registration, transaction processing and hierarchical consensus establishment. Details about these steps will be provided soon in this section.

B. Threat Model, Assumptions and Design Goals

We assume a Byzantine threat model in which the adversary can compromise no more than 1/3 validators of each blockchain. The compromised validators will act arbitrarily, instead of following the specified protocol. They may collude with each other to coordinate their attacks, including injecting, modifying and dropping messages during participating the protocol. The underlying network communication is semi-synchronous with bounded transmission delay, and the network may drop, delay, duplicate messages or deliver them out of order.

For a consensus algorithm to achieve agreement among a set of validators, it should satisfy consistency and liveness as defined below.

Definition 1. *(Consistency of HIBEChain) If a correct validator of a blockchain in HIBEChain outputs a sequence of blocks $b_i, (i = 0, ..., l)$, then all correct validators of the same blockchain output the same sequence of blocks $b_i, (i = 0, ..., l)$.*

Definition 2. *(Liveness of HIBEChain) If all correct validators of a blockchain in HIBEChain initiate the consensus algorithm, then all correct validators of this blockchain output some block eventually.*

The design goals of HIBEChain include:

- Scalability: The system should scale to the IoT network size in terms of communication, computation and storage. Namely, the required resources (communication, computation and storage) for maintaining the IoT network is linear to the network size n.
- Accountability: Whenever there is a repudiation or abuse of the system, the system should be able to identify the thing(s) according to the corresponding transaction(s).
- Agreement: The consensus algorithm of HIBEChain should satisfy the *consistency* and *liveness* property given more than 2/3 of the validators are honest and correct.
- Security: The signature scheme of HIBEChain should satisfy unforgeability and robustness against adaptive identity and chosen message attacks.
- Recoverability: The private keys should be recoverable when they are lost.

C. HIBEChain

HIBEChain consists of 5 phases: system initialization, validator authorization, IoT device registration, transaction processing and hierarchical consensus. The root blockchain is at level 0, the blockchains of the next level are at level 1, and so on and so forth. Each blockchain is denoted as $\mathcal{C}_{\mathsf{ID}|_k}$ where k is the level and $\mathsf{ID}|_k$ is the blockchain identity. We use the identity format in DHIBS, i.e. the identity at depth k is $\mathsf{ID}|_k = (ID_1, ID_2, ..., ID_k)$. $\mathsf{ID}|_k$ is a globally unique string, just like a domain name in the DNS system.

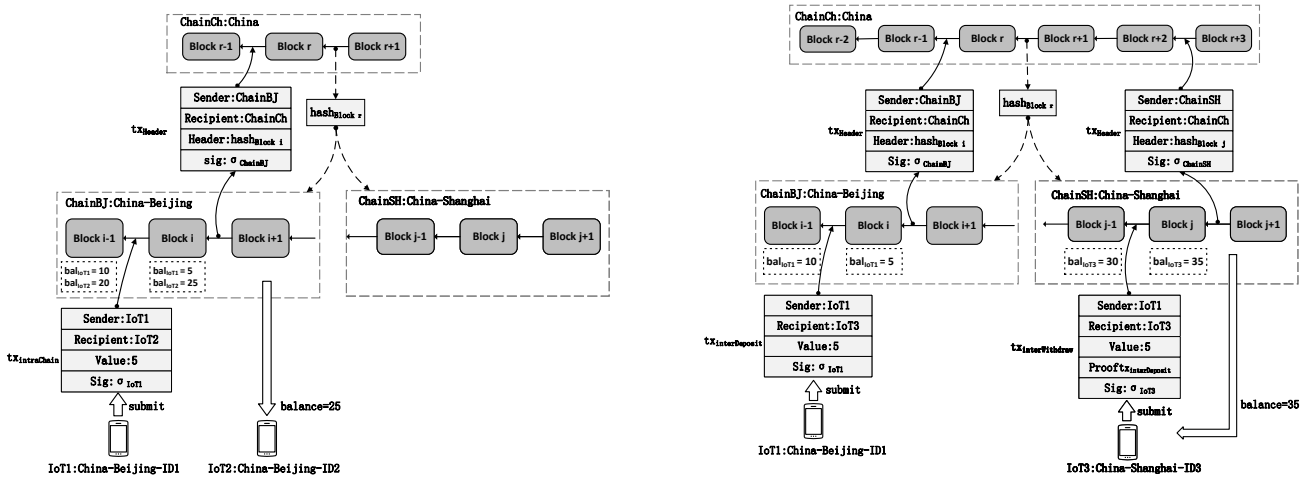
Within blockchain $\mathcal{C}_{\mathsf{ID}|_k}$, there are $n_{\mathsf{ID}|_k}$ validators and each one is denoted as $\mathsf{V}_{\mathsf{ID}|_k,i}$ where i is the validator's index. Validators are equivalent to authorities in DHIBS. Each blockchain also has a threshold $t_{\mathsf{ID}|_k}$ which is $\lceil \frac{2}{3} n_{\mathsf{ID}|_k} \rceil$ by default. It means at least $t_{\mathsf{ID}|_k}$ validators can establish consensus, so as to be compatible with PBFT.

System Initialization. This step initializes public system parameters and keys for validators of the root blockchain as follows:

(1) n_0 validators of the root blockchain are selected as root validators and they determine the hierarchy depth l and threshold t_0.
(2) The root validators run $\mathsf{DHIBS.Setup}(l, n_0, t_0)$ to generate public parameters $params = (g, g_1, g_2, g_3, h_1, ..., h_l)$ and private shadows $d^{(0)}_{\mathsf{ID}|_0} = (g_2^{\alpha_i}, 1, 1..., 1) \in \mathbb{G}^{l+2}$.
(3) The public keys are published throughout the whole system, while the private shadows (secret shares) are kept secret by each validators. No one knows the master secret α.

Validator Authorization. Similar to the domain name system, each blockchain obtains its identity from its parent blockchain in this phase. A candidate validator with index j of a blockchain $\mathcal{C}_{\mathsf{ID}|_k}$ needs to obtain authorization from multiple validators of the parent blockchain $\mathcal{C}_{\mathsf{ID}|_{k-1}}$.

(1) The candidate validator indexed by j selects $\mathcal{T}_{k-1}$, a set of validators of size $t_{\mathsf{ID}|_{k-1}}$ of $\mathcal{C}_{\mathsf{ID}|_{k-1}}$, to request authorization.
(2) Each validator from $\mathcal{T}_{k-1}$ runs $\mathsf{DHIBS.ShadowGen}$ $(d^{(i)}_{\mathsf{ID}|_{k-1}}, ID_k, j, \mathcal{T}_{k-1})$, and returns the results $d^{(i,j)}_{\mathsf{ID}|_k}$ to the requester. This is done with a secure channel or offline.



(a) Intra-Chain transaction processing

(b) Inter-Chain transaction processing

Fig. 3. Transaction Processing in HIBEChain. (a) shows intra-chain transaction processing, and (b) shows inter-chain transaction processing.

(3) After having received all partial shadows from upper-level validators, the candidate validator reconstructs its private shadow by running DHIBS.KeyRecon$(\{d^{(i,j)}_{\mathsf{ID}|_k}\}_{i\in\mathcal{T}_{k-1}})$. With a valid private shadow, the validator can collaborate with other validators to authorize validators at the next lower level.

IoT Device Registration. An IoT device with identity ID_k can register itself with a leaf blockchain (e.g. $\mathcal{C}_{\mathsf{ID}|_{k-1}}$) to join HIBEChain as follows.

(1) The IoT device sends its identity $\mathsf{ID}|_k$ along with the credential to a set of validators $\mathcal{T}_{k-1}$ of size $t_{\mathsf{ID}|_{k-1}}$ of the leaf blockchain $\mathcal{C}_{\mathsf{ID}|_{k-1}}$ to request registration.
(2) Each validator verifies the device's credential, and runs DHIBS.ShadowGen$(d^{(i)}_{\mathsf{ID}|_{k-1}}, ID_k, j, \mathcal{T}_{k-1})$ with threshold $t_{\mathsf{ID}|_k} = 1$.
(3) The IoT device reconstructs its private *key* corresponding to its full identity using DHIBS.KeyRecon$(\{d^{(i,j)}_{\mathsf{ID}|_k}\}_{i\in\mathcal{T}})$. Note: the IoT device is able to reconstruct the complete private key, while validators can only reconstruct private shadows, which are shares of the private key of the corresponding blockchain.

Transaction Processing. HIBEChain has three types of transactions: *intra-chain*, *inter-chain* and *header* transactions. An intra-chain transaction's sender and recipient are from the same leaf blockchain, whereas an inter-chain transaction's sender and recipient are from different leaf blockchains. The header transaction is composed of a block header with a signature generated by validators at a lower-level blockchain, and is submitted to the higher-level blockchain like a normal intra-chain transaction.

Both intra-chain transactions and header transactions are processed in the same way as Ethereum, except that DHIBS is used instead of ECDSA. As illustrated in Fig. 3(a), they are processed as follows:

(1) Sender submits an intra-chain payment transfer transaction.
(2) Each validator checks whether the sender has enough balance to make the transfer according to the local leaf blockchain (for intra-chain transactions only).
(3) Then each validator invokes DHIBS.Verify to check the signature in the transaction.
(4) If both checks are successful, the transaction is deemed valid and inserted into the next block to be assembled to the local blockchain.
(5) The sender's account and the recipient's account are updated according to this transaction.

As for the inter-chain transaction, we adopt an approach similar to OmniLedger's Atomix protocol [20] to achieve atomicity, but under the account model (in contrast to OmniLedger's UTXO model) as illustrated in Fig. 3(b):

(1) Sender submits an inter-chain payment transfer transaction to deposit a amount to sender blockchain.
(2) Each validator of the sender blockchain checks whether the sender has enough balance to make the transfer according to the sender blockchain.
(3) Each validator of the sender blockchain invokes DHIBS.Verify to check the signature in the transaction.
(4) If both checks are successful, the transaction is deemed valid and inserted into the next block to be assembled to the sender blockchain.
(5) The sender's account is updated according to this transaction.
(6) The recipient submits another inter-chain payment transfer transaction, containing a Merkle proof for the sender's inter-chain transaction, to the recipient blockchain, intending to withdraw the received amount to the recipient's account.
(7) Each validator of the recipient blockchain checks validity of the Merkle proof and the withdraw transaction.
(8) If the check is successful, the recipient's account is updated accordingly.

More explanation for Fig. 3 is provided later in this section.

Hierarchical Consensus Establishment. HIBEChain establishes consensus layer-by-layer, from leaf blockchains up until the root blockchain.

First, validators of each leaf blockchain synchronize transactions and compose a candidate block. Then they execute the PBFT algorithm to establish consensus on this block. During execution of the PBFT algorithm, validators generate partial signatures for the block after verifying all transactions inside it, using DHIBS.ShadowSign. The partial signatures σ_i are broadcast along with the block. When enough partial signatures are received, a complete signature can be reconstructed and then the corresponding block is deemed valid.

After the consensus is reached for a leaf blockchain, the validators report the resultant block header to the validators of the parent blockchain. The parent validators also receive the block headers from other children, and check validity of the block headers using DHIBS.Verify. Then they execute the PBFT algorithm on the received block headers to generate a new block. This process continues until the consensus is also reached at the root blockchain.

VI. Discussion and Analysis

In this section we discuss scalability, accountability, security and recoverability of HIBEChain. Then we present possible optimization and improvement techniques for HIBEChain.

A. Scalability

HIBEChain achieves scalability by employing a hierarchical structure of blockchains, with each blockchain handling a small part of transactions of the system. Essentially, this strategy is a type of sharding, similar to the sharding schemes studied by Elastico [16] and RapidChain [19]. Thus, HIBEChain is highly scalable as all leaf blockchains can process transactions in parallel. Transaction processing throughput (TPS, transactions per second) of HIBEChain is roughly proportional to the number of blockchains, so it can be increased infinitely theoretically. Meanwhile, HIBEChain is also highly scalable in data storage. A transaction is only stored by the validators in the corresponding leaf blockchain, instead of spreading to validators of other blockchains.

We analyze communication and computation costs of a full k-ary L-level HIBEChain (excluding the end-device level), in which each blockchain is maintained by n validators. Then this HIBEChain has $(k^L-1)n/(k-1)$ validators in total and k^{L-1} leaf blockchains.

From the perspective of key management, the computation and communication costs are listed in Table II. According to this table, both computation and communication costs of HIBEChain are modest, determined by the level, thresholds and the number of validators.

As for transaction processing, HIBEChain has an obvious advantage over singleton blockchains like existing sharding schemes: leaf blockchains can process a part of transactions in parallel. Different from existing sharding schemes, HIBEChain adopts a hierarchical structure, on which non-leaf blockchains do not process transactions from IoT devices. Assume the same parameters as in Table II and T TPS for each blockchain. Then HIBEChain's TPS is $k^{L-1}T$ whereas existing sharding schemes' TPS is $(k^L-1)T/(k-1)$, and their ratio is

$$\frac{k^L-k^{L-1}}{k^L-1}.$$

TABLE II
Computation and Communication Costs of HIBEChain

	Computation	Communication
Setup	-	$O(n^2)$
ShadowGen	$2(L-l+2)$ Exp.	$O(n^2)$
KeyRecon	$(L-l+2)\cdot t$ Exp.	-
ShadowSign	2 Exp.	$O(n^2)$
SignRecon	$3t$ Exp.	-
Verify	4 Pairings	-

Note: Assume all blockchains have the same number of validators and the same threshold. L denotes maximum hierarchy level, l denotes current level, t denotes threshold value, n is the number of validators of a single blockchain. Exp denotes exponentiation and pairing denotes bilinear pairings.

For an IoT network with $L=5$ and $k=10$ (consists of $\sim$10000 shards or leaf blockchains), this ratio is 0.90, and it is close to 1 for large IoT systems. This means HIBEChain's TPS is very close to that of a sharding scheme for large-scale systems.

It is important to reduce inter-chain transactions for sharding schemes, as inter-chain transactions involves additional verification. HIBEChain naturally achieves it since a leaf blockchain is in charge of devices in a local domain. As IoT devices interact most with others nearby, inter-chain transactions are significantly reduced.

B. Accountability

In HIBEChain, every participant must prove its identity on registration, so as to obtain an identity-based private key. The identities are used as public keys, and thus every transaction can be traced back to responsible parties. Therefore, HIBEChain explicitly achieves accountability with the identity-based cryptosystem, in contrast to anonymous public/private key pairs used in existing blockchain systems. Because of lack of accountability of current blockchain systems, Know your customer (KYC) regulation has to be enforced to ensure accountability off-chain.

Note that the higher-level validators can collaborate to recover a private key or generate a valid signature, which is the inherent key escrow problem of IBE. Nevertheless this problem is largely mitigated since at least t validators must collaborate to do it.

A drawback of HIBEChain would be lack of anonymity, but mitigation measurements can be used to preserve privacy. One way to protect privacy of transactions is through access control as Hyperledger Fabric [13] does. As HIBEChain is a permissioned blockchain system, managed by validators at different levels, the blockchain data can be encrypted by the managing validators to prevent eavesdropping. Different blockchains can be encrypted by different keys to achieve flexible access control.

Another potential solution is ZK-SNARK (Zero Knowledge Succinct Non-interactive ARgument of Knowledge). This technique has been successfully used in ZeroCash [33] to hide the sender, the recipient and the amount of a transaction. So similarly, ZK-SNARK can be used in HIBEChain to protect sender, recipient and amount privacy of transactions.

C. Security

1) Adversarial Model: The Security requirement of a decentralized hierarchical identity-based threshold signature (DHIBS) scheme includes both unforgeability and robustness [34]. To begin with, we briefly review the notion of unforgeability against selective identity and chosen message attacks for DHIBS schemes (i.e., the UF-sID-CMA security) between an adversary algorithm $\mathcal{A}$ which is assumed to be a probabilistic Turing machine and a challenger algorithm $\mathcal{C}$.

- **Initialization**. $\mathcal{A}$ outputs a target identity $\mathsf{ID}^* = (ID_1, ID_2, ..., ID_m)$ where $m \leqslant l$. $\mathcal{A}$ intends to forge a signature of ID^*.
- **Setup**. $\mathcal{C}$ runs DHIBS.Setup with a given (l, n_0, t_0) to generate the public parameters $params$ and the master secret mk. $\mathcal{C}$ sends the public parameters $params$ to $\mathcal{A}$.
- **Query**. $\mathcal{A}$ issues a number of queries to $\mathcal{C}$, and $\mathcal{C}$ answers as follows.
  - On receiving a partial shadow generation query on an identity $\mathsf{ID}|_k$ with parameters (i, j), algorithm $\mathcal{C}$ runs DHIBS.PartialShadowGen to produce a partial shadow $d^{(i,j)}_{\mathsf{ID}|_k}$ for the identity $\mathsf{ID}|_k$. $\mathcal{C}$ sends the private shadow $d_{\mathsf{ID}|_k}$ to $\mathcal{A}$.
  - On receiving a shadow reconstruction query on an identity $\mathsf{ID}|_k$ with parameter i, algorithm $\mathcal{C}$ runs DHIBS.ShadowRecon to produce a shadow $d^{(i)}_{\mathsf{ID}|_k}$ for the identity $\mathsf{ID}|_k$. $\mathcal{C}$ sends the private shadow $d_{\mathsf{ID}|_k}$ to $\mathcal{A}$.
  - On receiving a partial signature query on an identity $\mathsf{ID}|_k$ with parameter (i, M), $\mathcal{C}$ runs DHIBS.ShadowSign to produce a partial signature σ_i for M. $\mathcal{C}$ sends the partial signature σ_i to $\mathcal{A}$.
  - On receiving a signature query on an identity $\mathsf{ID}|_k$, algorithm $\mathcal{C}$ runs DHIBS.SignRecon to produce a signature σ for M. $\mathcal{C}$ sends the signature σ to $\mathcal{A}$.
- **Forge**. $\mathcal{A}$ outputs a tuple (ID^*, σ^*, M^*) for σ^* being a valid signature for the message M^* under the identity ID^*. The restriction is that $\mathcal{A}$ must not have made more than t_k (the corresponding threshold value) shadow generation queries for the challenge identity ID^* or any prefix of the challenge identity ID^*. Also it must not have made more than t_k (the corresponding threshold value) partial signature queries for the message M^* on the challenge identity ID^*. Otherwise, it is trivial for $\mathcal{A}$ to forge the required signature, because it can recover the corresponding private key or the required signature.

The advantage of algorithm $\mathcal{A}$ in the UF-sID-CMA game is defined to be $\mathsf{Adv}_{\mathcal{A}} = \Pr[\mathsf{DHIBS.Verify}(params, \mathsf{ID}^*, \sigma^*, M^*) = 1]$.

Definition 3. *(UF-sID-CMA) A DHIBS scheme is said to be UF-sID-CMA secure if algorithm $\mathcal{A}$'s advantage in the UF-sID-CMA game is negligible, i.e.,* $\mathsf{Adv}_{\mathcal{A}} < \epsilon$.

Now we define the robustness of the DHIBS scheme.

Definition 4. *(Robustness) A DHIBS scheme is said to be robust if it computes a correct output even in the presence of a malicious attacker that makes the corrupted signature generation servers deviate from the normal execution.*

2) Security Analysis:

Theorem 1. *Assuming that $n_k \geq 2t_k - 1$ for every level k in the proposed DHIBS scheme, then it is robust in the presence of up to $t_k - 1$ corrupted signature generators.*

Proof: Regarding the DHIBS.PartialShadowGen algorithm, it is straightforward to see that t_k honest signature generators are required for the signature generation and at most $t_k - 1$ signature generators can be corrupted, which requires that $n \geq 2t_k - 1$. In addition, the corrupted signature generators cannot destroy the functionality of the DHIBS.PartialShadowGen algorithm, and consequently the status of all the uncorrupted signature generators. Therefore the DHIBS.PartialShadowGen algorithm is robust if $n_k \geq 2t_k - 1$.

Concerning the DHIBS.ShadowSign algorithm, suppose that there are at most $t_k - 1$ signature generators corrupted and the number of uncorrupted signature generators is at least $n_k - (t_k - 1) \geq t_k$. Every corrupted signature generator can either be halted or issues an invalid partial signature. For the first case, the corrupted signature generators do not generate any signature and will not harm the execution of the uncorrupted signature generators. In the second case, if a partial signature is invalid, then it should be detected and excluded from the final signature as in the definition. Therefore, all the uncorrupted signature generators (at least t_k) are still able to output a valid signature in the presence of up to $t_k - 1$ corrupted signature generators, and the proposed DHIBS scheme is robust if $n_k \geq 2t_k - 1$. ■

Theorem 2. *Assuming that the underlying HIBS scheme is UF-sID-CMA secure, then the proposed DHIBS scheme is UF-sID-CMA secure.*

Proof: Assume that there exists an adversary algorithm $\mathcal{A}_0$ that breaks the UF-sID-CMA security of the given DHIBS scheme. Then we can build an adversary algorithm $\mathcal{A}_1$ that breaks the UF-sID-CMA security of the underlying HIBS scheme. Let $\mathcal{C}_1$ be the challenge algorithm in the UF-sID-CMA security game of the underlying HIBS scheme.

- **Initialization**. $\mathcal{A}_1$ initializes $\mathcal{A}_0$, which outputs a target identity $\mathsf{ID}^* = (ID_1, ID_2, ..., ID_m)$ where $m \leqslant l$. Then $\mathcal{A}_1$ forwards ID^* to $\mathcal{C}_1$.
- **Setup**. $\mathcal{A}_1$ receives the public parameters $params$ from $\mathcal{C}_1$, and forwards them to $\mathcal{A}_0$.
- **Query**. $\mathcal{A}_0$ issues a number of queries to $\mathcal{A}_1$, and $\mathcal{A}_1$ answers as follows.
  - Shadow query on identity $\mathsf{ID}|_k$ with parameter i:
    - ► If $\mathsf{ID}|_k \not\preceq \mathsf{ID}^*$ (i.e. $\mathsf{ID}|_k$ is not equal to or a prefix of ID^*), $\mathcal{A}_1$ makes a private key query on an identity $\mathsf{ID}|_{k-1}$ to $\mathcal{C}_1$. After receiving the private key $d_{\mathsf{ID}|_{k-1}} = (a_0, a_1, b_k, ..., b_l)$ from $\mathcal{C}_1$, $\mathcal{A}_1$ selects a random functions $f(x) = c_1 x + c_2 x^2 + ... + c_{t_{k-1}} x^{t_{k-1}-1}$. Then $\mathcal{A}_1$ computes $d^{(i)}_{\mathsf{ID}|_k} = (a_0 g_2^{f(i)} (h_1^{I_1} ... h_k^{I_k} g_3)^{f(i)}, a_1 g^{f(i)}, b_{k+1} h_{k+1}^{f(i)}, ..., b_l h_l^{f(i)})$ = $(a'_0, a'_1, b'_k, ..., b'_l)$, and answers the query with

$d^{(i)}_{\mathsf{ID}|_k}$.

- ▶ Otherwise, $\mathsf{ID}|_k \preceq \mathsf{ID}^*$, then $\mathcal{A}_0$ can only ask for $t_k - 1$ shadows. Suppose the set of indices $\mathcal{A}_0$ wants to query is IdxSet. For $i \in \mathsf{IdxSet}$, $\mathcal{A}_1$ can generate the shadow as $d^{(i)}_{\mathsf{ID}|_k} = (g_2^{\alpha_i^\dagger}(h_1^{I_1}...h_k^{I_k}g_3)^{r_i^\dagger}, g^{r_i^\dagger}, h_{k+1}^{r_i^\dagger}, ..., h_l^{r_i^\dagger})$, where $\alpha_i^\dagger$ and $r_i^\dagger$ are random numbers chosen by $\mathcal{A}_1$. The corresponding verification key is $v^{(i)}_{\mathsf{ID}|_k} = g^{r_i^\dagger}$.
  For an index $j \notin \mathsf{IdxSet}$, $\mathcal{A}_1$ only publishes a verification key $v^{(j)}_{\mathsf{ID}|_k}$. According to the scheme specification, $g_1 = g^\alpha = \prod_{i\in\mathsf{IdxSet}} (g^{\alpha_i^\dagger})^{L(i)} \cdot (g^{\alpha_j^\dagger})^{L(j)} = \prod_{i\in\mathsf{IdxSet}} (v^{(i)}_{\mathsf{ID}|_k})^{L(i)} \cdot (v^{(j)}_{\mathsf{ID}|_k})^{L(j)}$. So $\mathcal{A}_1$ can computes and outputs the verification key for j as $\left(\frac{g_1}{\prod_{i\in\mathsf{IdxSet}} (v^{(i)}_{\mathsf{ID}|_k})^{L(i)}}\right)^{\frac{1}{L(j)}}$.

– Partial shadow generation query on identity $\mathsf{ID}|_k$ with parameters (i, j):

- ▶ If $\mathsf{ID}|_k \not\preceq \mathsf{ID}^*$, $\mathcal{A}_1$ does the same in answering the shadow query to obtain a private shadow $d^{(i)}_{\mathsf{ID}|_k} = (a'_0, a'_1, b'_k, ..., b'_l)$. Then $\mathcal{A}_1$ runs $\mathsf{DHIBS.PartialShadowGen}$ to produce a partial shadow $d^{(i,j)}_{\mathsf{ID}|_k}$ for the identity $\mathsf{ID}|_k$. $\mathcal{C}$ sends the partial shadow $d^{(i,j)}_{\mathsf{ID}|_k}$ to $\mathcal{A}_0$.
- ▶ Otherwise, $\mathsf{ID}|_k \preceq \mathsf{ID}^*$, then $\mathcal{A}_0$ is allowed to ask for only $t_{k-1} - 1$ partial shadows. In this case, $\mathcal{A}_1$ composes the $t_{k-1} - 1$ shadows in the same way of answering shadow reconstruction queries, then $\mathcal{A}_1$ can generate partial shadows as required by invoking $\mathsf{DHIBS.PartialShadowGen}$ with the secret shadows.

– Signature query on identity $\mathsf{ID}|_k$ on message M: $\mathcal{A}_1$ forwords the query to $\mathcal{C}_1$, and answers the query with the result received from $\mathcal{C}_1$.

– Partial signature query on identity $\mathsf{ID}|_k$ with parameter (j, M):

- ▶ If $\mathsf{ID}|_k \not\preceq \mathsf{ID}^*$, $\mathcal{A}_1$ computes the required shadow as above and uses it to generate the partial signature for $\mathcal{A}_0$.
- ▶ If $\mathsf{ID}|_k \preceq \mathsf{ID}^*$ but $M \neq M^*$, $\mathcal{A}_1$ composes a signature query over M and forwards it to $\mathcal{C}_1$. Suppose $\mathcal{C}_1$'s answer is $\sigma = (x, y, z)$, and IdxSet ($|\mathsf{IdxSet}| = t_k - 1$) is the set of authorities whose shadows have been leaked to $\mathcal{A}_0$. $\mathcal{A}_1$ generates a random number s and shares it among authorities IdxSet and any $j \notin \mathsf{IdxSet}$ with (t_k, n_k)-secret sharing scheme. For $i \in \mathsf{IdxSet}$, $\mathcal{A}_1$ generates the partial signature as $\sigma_i = (x_i, y_i, z_i)$ using authority i's shadow. For the partial signature $\sigma_j = (x_j, y_j, z_j)$, the first component x_j (same for y_j and z_j) can be generated as $\left(\frac{x}{\prod_{i\in\mathsf{IdxSet}} x_i^{L(i)}}\right)^{\frac{1}{L(j)}}$.
- ▶ If $\mathsf{ID}|_k \preceq \mathsf{ID}^*$ and $M = M^*$, $\mathcal{A}_0$ is allowed to ask for only $t_k - 1$ partial signatures in this case (otherwise $\mathcal{A}_0$ can reconstruct the complete signature). So $\mathcal{A}_1$ can use the $t_k - 1$ private shadows to generate the partial signatures as required.

- **Forge**. $\mathcal{A}_0$ outputs a tuple (ID^*, σ^*, M^*) for σ^* being a valid signature for the message M^* under the identity ID^*. $\mathcal{A}_1$ forwards (ID^*, σ^*, M^*) to $\mathcal{C}_1$.

To sum up, if algorithm $\mathcal{A}_0$ can break the UF-sID-CMA security of the proposed DHIBS scheme with non-negligible probability, then algorithm $\mathcal{A}_1$ can break the UF-sID-CMA security of the underlying HIBS scheme with non-negligible probability. ■

D. Agreement

Theorem 3 (Consistency of HIBEChain)**.** *HIBEChain achieves consistency for asynchronous networks if corrupted validators within each blockchain of HIBEChain are no more than $f < 1/3$ of all validators.*

Proof: It is easy to see that all leaf blockchains of HIBEChain achieve consistency since they run the PBFT algorithm, whose safety (consistency) has been rigorously proved. As a result of the PBFT algorithm, each leaf blockchain outputs a new block with a complete HIBS signature in the block header. The higher-level validators collect blocks with HIBS signatures from lower-level blockchains, and then run the PBFT algorithm again to produce a higher-level block with a new HIBS signature. So the higher-level blockchain also achieves consistency for the same reason. This procedure continues until the root blockchain reaches consensus using the PBFT algorithm. That is, consistency is achieved at all levels of HIBEChain. ■

Theorem 4 (Liveness of HIBEChain)**.** *HIBEChain achieves liveness for semi-synchronous networks if corrupted validators within each blockchain of HIBEChain are no more than $f < 1/3$ of all validators.*

Proof: All leaf blockchains of HIBEChain also achieve liveness due to liveness of the PBFT algorithm in semi-synchronous networks. Then the leaf validators submit block headers to higher-level blockchains, which also use the PBFT algorithm for consensus establishment. This procedure continues until the root blockchain, so every blockchain of HIBEChain achieves liveness. ■

E. Recoverability

A desirable feature of DHIBS is that the private key of an end user/device can be recovered by multiple validators re-executing $\mathsf{DHIBS.ShadowGen}$. That is, the private key can be regenerated with validators more than the threshold in case of key lost. In blockchains like Bitcoin or Ethereum, once the private key is lost, it is infeasible to recover it. The cryptocurrency associated with the lost private key cannot retrieved anymore. This also implies that the escrow problem in IBE schemes are also solved by DHIBS with decentralization.

Based on the above analysis, we provide a comparison between HIBEChain with other sharding schemes in Table III.

TABLE III
COMPARISON OF HIBECHAIN WITH EXISTING SHARDING SCHEMES

Protocol	RSCoin[18]	Elastico[16]	OmniLedger[20]	RapidChain[19]	Parallel-Chains[21]	HIBEChain((3,4)-threshold)
Type	Permissioned	Permissionless	Permissionless	Permissionless	Permissionless	Permissioned
Key Recoverability	No	No	No	No	No	**Yes**
Key Management	No	No	No	No	No	**Yes**
Layers	2	2	2	2	1	$\boldsymbol{L}$
Latency	1 sec	800 sec	1.5 sec	8 sec	N.A.	$1.3L$ sec

VII. IMPLEMENTATION AND PERFORMANCE EVALUATION

A. Implementation

We implement HIBEChain based on Ethereum in about 6000 lines of code in Golang. Specifically, we use the Pairing-Based Cryptography (PBC) library [35] to implement DHIBS and a Go wrapper to use PBC [36] in our implementation.

In our implementation, we use a symmetric elliptic curve $y^2 = x^3 + x$ of order p, where p is 160 bits in length. The hash function H is instantiated with SHA256, and we use the first 160 bits of the output; H' is realized by using SHA256 to obtain an element $h' \in \mathbb{Z}_p$ just like H and then output $g^{h'}$.

We connect all blockchains in HIBEChain to form a tree structure, whose depth can be adjusted as needed. Each blockchain in the tree can have any number of child blockchains. In the simplest case, HIBEChain can only consist of a root blockchain.

A hierarchical identity ID are implemented as a string, and the identity of a parent is always a prefix of its children. For easy implementation, a hierarchical ID is in the form like "123", with its parent's ID being "12" and its grandparent's ID "1".

We adopt the PBFT algorithm as the consensus mechanism of HIBEChain. The primary replica finalizes a block and then starts the PBFT algorithm for this block. The PBFT algorithm requires $3f + 1$ replicas to be able to tolerate f failed nodes. This parameter perfectly matches the threshold of DHIBS, so we can set the threshold as $(2f + 1, 3f + 1)$. Moreover, each blockchain can have arbitrary thresholds and arbitrary number of validators.

To support intra-chain transactions, inter-chain transactions and block header transactions, we design four new transactions: $\mathsf{tx}_{\mathsf{intraChain}}$, $\mathsf{tx}_{\mathsf{interDeposit}}$, $\mathsf{tx}_{\mathsf{interWithdraw}}$ and $\mathsf{tx}_{\mathsf{Header}}$.

B. Experimental Results

We deploy our system on Aliyun ecs.r6.xlarge virtual machines, each of which has 4 vCPU and 32GB memory. Within each Aliyun virtual machine, we runs at most 10 independent docker containers as blockchain nodes. The number of running nodes of HIBEChain reaches at most 2,600. We develop a suite of automatic testing tools using python to aid our experiments. With these tools, we measure the performance of the DHIBS scheme and evaluate the transaction latency and throughput for different thresholds, branches (number of child blockchains) and tree depths.

We first test benchmarks of our DHIBS scheme. We provide the performance of each algorithm of DHIBS for different (t, n)-threshold. As showed in Fig. 4, the computation time for Setup, KeyRecon and SignRecon increases steadily as the threshold increases, while ShadowGen, ShadowSign and Verify have almost fixed time costs. This conforms to our analytical results summarized in Table II.

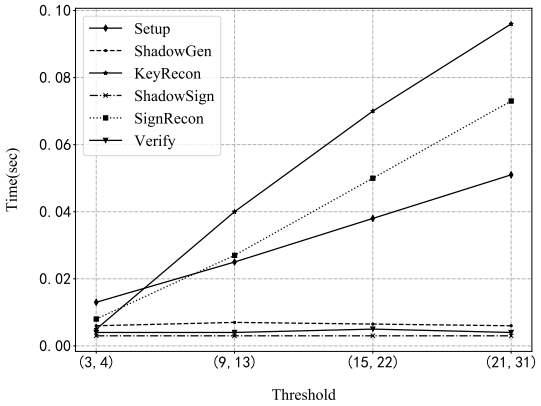


Fig. 4. Time costs for algorithms of DHIBS in different threshold settings.

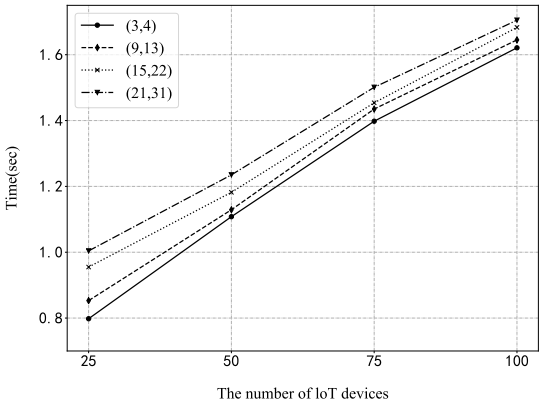


Fig. 5. Time cost for different number of IoT devices to obtain hierarchical identity-base private keys in different thresholds settings. It measures the latency from the moment the IoT devices send shadow requests to their parent blockchain until they all reconstruct their private keys.

We separately configure four single blockchains with threshold $(3, 4)$, $(9, 13)$, $(15, 22)$ and $(21, 31)$. Then we set their blockchain ID "HIBE". Under each single blockchain, we deploy some independent nodes (their ID is set "HIBE-001", "HIBE-002"..., and each has the threshold $(1, 1)$) as its children to simulate different numbers of IoT devices connected to HIBEChain. All nodes simultaneously send shadow requests to the parent blockchain and reconstruct their private keys. We measure the latency for 25, 50, 75 and 100 nodes respectively. Fig. 5 shows that 100 IoT devices can obtain hierarchical identity-base private keys in about 1.7 seconds from HIBEChain with threshold $(21, 31)$. The time cost increases uniformly as the number of IoT devices increases, and the

cost for larger thresholds is slightly larger than the smaller ones.

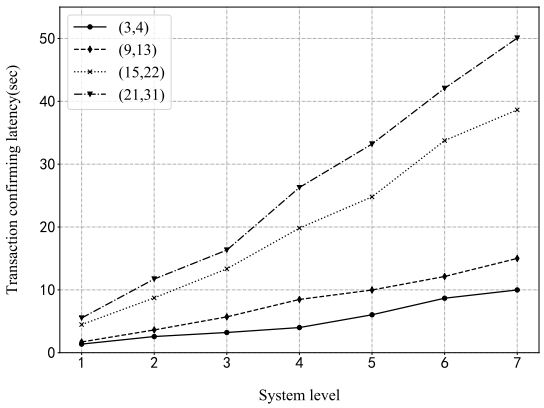


Fig. 6. Transaction confirmation latency for in different thresholds settings and different system level.

To measure the transaction confirmation latency (from the transaction is issued by the IoT device until it is confirmed by the root blockchain), we set four thresholds and for each threshold we deploy HIBEChain with tree depth from 1 to 7 (each blockchain has only one child blockchain). Fig. 6 shows that the transaction confirmation latency for a 1-ary, 7-level, $(3, 4)$-threshold HIBEChain system is about 10 seconds. For a 1-ary, 7-level, $(21, 31)$-threshold HIBEChain system, the transaction latency is about 50 seconds, which is about 5 times of the $(3, 4)$-threshold HIBEChain.

Fig. 7 shows detailed processing time costs at each level. It consists of four parts: block preparation latency, consensus latency, txHeader generation latency and txHeader processing latency. One-level processing time with threshold $(3, 4)$ is about 1.4 seconds and 7.4 seconds for threshold $(21, 31)$. We should notice that the txHeader processing latency is influenced significantly by the parent blockchain's consensus latency. This is because when the txHeader reaches the parent blockchain, it has to wait until the parent blockchain's consensus is finished. Therefore, the average latency is about half of the parent blockchain's consensus latency.

We also evaluate how the threshold influences the transaction confirmation latency by setting different t in the (t, n) threshold. Instead of setting the threshold as $(2f + 1, 3f + 1)$, we set t as 10, 12, 14, 16, 18, 20 for a 1-ary, 4-level, $(t, 20)$-threshold HIBEChain system. Fig. 8 shows that the transaction confirmation latency at the $(10, 20)$-threshold is about 12.4 seconds, while at the $(20, 20)$-threshold it is about 20.3 seconds. Because a larger t leads to more time cost for txHeader generation, consensus and txHeader processing latency as discussed above.

Fig. 9 shows the throughput for a single leaf blockchain under different threshold settings. The TPS decreases as the threshold increases since a larger threshold leads to more consensus delay. Therefore, we can deploy the leaf blockchain with a specific threshold according to the actual TPS requirement. For example, if IoT devices within a region frequently communicate with each other, we can deploy a leaf blockchain with a small threshold. If they communicate with each other

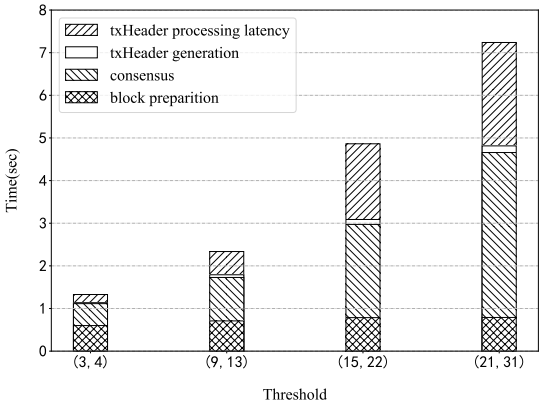


Fig. 7. The processing latency at each level. It consists of four parts: block preparation latency, consensus latency, txHeader generation latency and txHeader processing latency. Block preparation includes transaction verification, transaction preprocess and block packing. Consensus is to reach an agreement with the block by running the PBFT algorithm. TxHeader generation latency is to collect partially-signed $\mathsf{tx}_{\mathsf{Header}}$s and reconstruct a $\mathsf{tx}_{\mathsf{Header}}$ with a complete signature. Txheader processing latency refers to the time cost for transmitting the $\mathsf{tx}_{\mathsf{Header}}$ to the parent blockchain and the latency before the $\mathsf{tx}_{\mathsf{Header}}$ is handled by the parent blockchain.

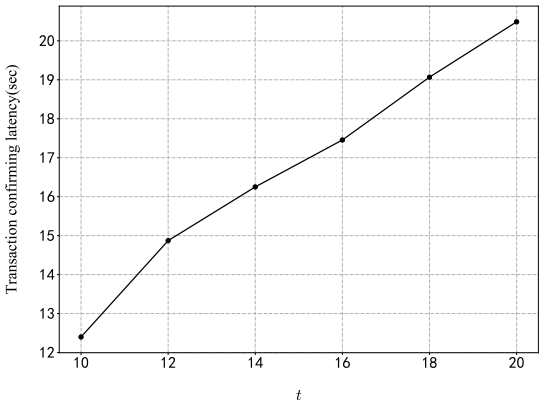


Fig. 8. How t in threshold $(t, 20)$ influences the transaction confirmation time of a 1-ary, 4-level HIBEChain system.

infrequently, we can deploy a leaf blockchain with a large threshold which can enhance the security.

To explore the scalability of the HIBEChain, we fix the tree depth and the threshold, and we vary the branch number of the tree. Fig. 10 shows the scalability of a 4-level, $(7, 10)$-threshold system when its branch grows from 1 to 6, and the number of leaf blockchains reaches 1, 8, 27, 64, 125 and 216 respectively. The result implies the HIBEChain is highly scalable and efficient, as the TPS grows to 32,000 for the 6-ary case while the transaction confirmation latency stays almost unchanged at about 9 seconds.

Finally, we measure the time costs to complete the intra-chain transaction and the inter-chain transaction. Fig. 11 shows that in a 4-level, $(7, 10)$-threshold system, the confirmation latency for $\mathsf{tx}_{\mathsf{intraChain}}$, $\mathsf{tx}_{\mathsf{interDeposit}}$ and $\mathsf{tx}_{\mathsf{interWithdraw}}$ is about 9 seconds. That implies it takes 9 seconds to complete an intra-chain transaction and 18 seconds to complete an inter-chain transaction.

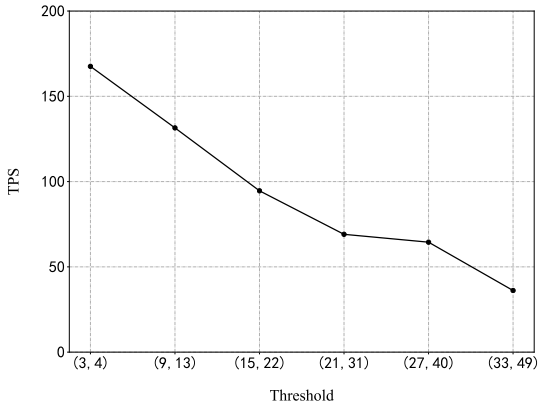


Fig. 9. Throughput for single leaf blockchain under different threshold settings. The TPS for a leaf blockchain with the (3,4)-threshold setting reaches 170.

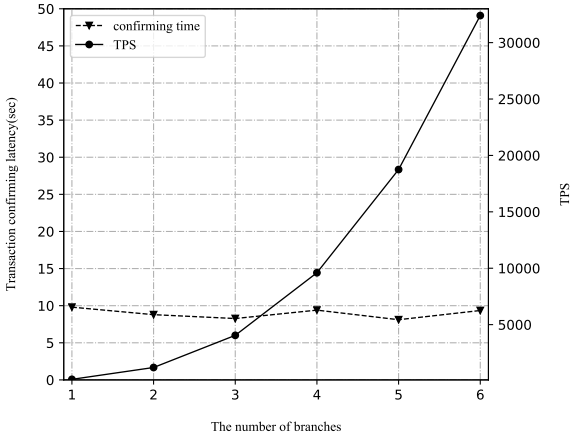


Fig. 10. Scalability for HIBEChain with 4-level, $(7, 10)$-threshold. TPS grows as the ary increases (TPS reaches 32,000 at 6-ary), while the transaction confirmation latency stays unchanged (about 9 seconds).

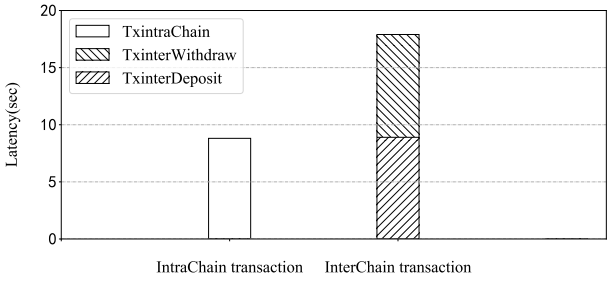


Fig. 11. Time to complete an intra-chain transaction (using $\mathsf{tx}_{\mathsf{intraChain}}$) and an inter-chain transaction (using $\mathsf{tx}_{\mathsf{interDeposit}}$ and $\mathsf{tx}_{\mathsf{interWithdraw}}$). The later is about one time longer than the former.

VIII. Conclusion

In this paper we have presented HIBEChain, a hierarchical blockchain system, which is based on a decentralized hierarchical identity-based signature scheme for IoT systems. The hierarchical structure makes HIBEChain highly scalable and efficient in processing the huge amount of transactions in large-scale IoT systems. HIBEChain also enjoys great user-friendliness by using identity-based keys, and private keys can be recovered by validators when necessary. We have analyzed performance and security of HIBEChain, and implemented HIBEChain based on Ethereum source code. Experiment results showed that HIBEChain is highly efficient in key management and consensus establishment, and its high throughput is suitable for IoT system.

Though HIBEChain adopts the permissioned blockchain using the PBFT consensus algorithm, its design can be easily extended to permissionless blockchains with other consensus algorithms.

References

[1] InformationWeek, "Gartner: 21 billion iot devices to invade by 2020." https://www.informationweek.com/mobile/mobile-devices/gartner-21-billion-iot-devices-to-invade-by-2020. Accessed: 2018-07-23.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." http://bitcoin.org/bitcoin.pdf, 2008.

[3] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[4] S. Panikkar, S. Nair, P. Brody, and V. Pureswaran, "Adept: An iot practitioner perspective," 2015.

[5] IBM, "Iot platform-ibm watson iot." https://www.ibm.com/internet-of-things/spotlight/watson-iot-platform. Accessed: 2018-07-23.

[6] ForkLog, "Alibaba, zte and china unicom to create a blockchain framework for the internet of things." http://forklog.net/alibaba-zte-and-china-unicom-to-create-a-blockchain-framework-for-the-internet-of-things/. Accessed: 2018-07-23.

[7] Filament. https://filament.com/. Accessed: 2018-07-23.

[8] S. Popov, "The tangle." https://iota.org/IOTA_Whitepaper.pdf, 2016. Accessed: 2018-12-05.

[9] Ethereum. https://ethereum.org/. Accessed: 2018-07-23.

[10] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Proc. Eurocrypt'17*, p. 643–673.

[11] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake." http://peerco.in/assets/paper/peercoin-paper.pdf, 2012.

[12] D. Larimer, "Delegated proof-of-stake (dpos)," *Bitshare whitepaper*, 2014.

[13] E. Androulaki, C. Cachin, K. Christidis, C. Murthy, B. Nguyen, and M. Vukolić, "Hyperledger fabric proposals: Next consensus architecture proposal," 2016.

[14] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm." https://ripple.com/files/ripple_consensus_whitepaper.pdf, 2014.

[15] D. Mazières, "The stellar consensus protocol: A federated model for internet-level consensus." https://www.stellar.org/papers/stellar-consensus-protocol.pdf, 2015.

[16] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC CCS*, pp. 17–30, 2016.

[17] Ethereum, "Sharding faqs." https://github.com/ethereum/wiki/wiki/Sharding-FAQs. Accessed: 2018-07-23.

[18] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[19] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proc. ACM CCS'18*, pp. 931–948, 2018.

[20] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *Proc. 2018 IEEE Symposium on Security and Privacy (S&P)*, pp. 19–34, 2018.

[21] M. Fitzi, P. Gaži, A. Kiayias, and A. Russell, "Parallel chains: Improving throughput and latency of blockchain protocols via parallel composition." Cryptology ePrint Archive, Report 2018/1119, 2018. https://eprint.iacr.org/2018/1119.

[22] A. Reyna, C. Martin, J. Chen, E. Soler, and M. Diaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, 2018.

[23] J. Ito, "Our response to "A Cryptocurrency Without a Blockchain Has Been Built to Outperform Bitcoin"." https://www.media.mit.edu/posts/iota-response/. Accessed: 2019-2-10.

[24] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *Journal of Network and Computer Applications*, vol. 125, pp. 251–279, January 2019.
[25] A. Bahga and V. Madisetti, "Blockchain platform for industrial internet of things," *Journal of Software Engineering and Applications*, vol. 9, pp. 533–546, 2016.
[26] A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards Better Availability and Accountability for IoT Updates by means of a Blockchain," in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW'17)*, 2017.
[27] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of Trust: A decentralized blockchain-based authentication system for IoT," *Computers & Security*, vol. 78, pp. 126–142, 2018.
[28] D. Boneh, X. Boyen, and E. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proceedings of EUROCRYPT*, pp. 440–456, 2005.
[29] S. S. M. Chow, L. C. K. Hui, S. Yiu, and K. P. Chow, "Secure hierarchical identity based signature and its application," in *Proc. of ICICS'04*, vol. 3269, pp. 480–494, 2004.
[30] A. Kate and I. Goldberg, "Distributed private-key generators for identity-based cryptography," in *Security and Cryptography for Networks*, pp. 436–453, Springer Berlin Heidelberg, 2010.
[31] K. G. Paterson and J. C. N. Schuldt, "Efficient identity-based signatures secure in the standard model," in *Proceedings of ACISP, 2006*, pp. 207–222, 2006.
[32] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Proceedings of CRYPTO'91*, pp. 129–140, 1991.
[33] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy*, pp. 459–474, 2014.
[34] J. Baek and Y. Zheng, "Identity-based threshold signature scheme from the bilinear pairings," in *Proceedings of ITCC 2004.*, vol. 1, pp. 124–128 Vol.1, April 2004.
[35] B. Lynn, "The pairing-based cryptography library." https://crypto.stanford.edu/pbc/.
[36] Github, "The pbc go wrapper." https://github.com/Nik-U/pbc.