

Received June 3, 2018, accepted July 19, 2018, date of publication August 1, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2861944

Mchain: A Blockchain-Based VM Measurements Secure Storage Approach in IaaS Cloud With Enhanced Integrity and Controllability

BO ZHAO, PEIRU FAN¹, AND MINGTAO NI

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Corresponding author: Peiru Fan (fanpeiru@foxmail.com)

This work was supported in part by the National Key Basic Research Program of China (973 Program) under Grant 2014CB340600 and in part by the Wuhan FRONTIER Program of Application Foundation under Grant 2018010401011295.

ABSTRACT Virtual machine (VM) measurements data in IaaS cloud play a crucial role in integrity evaluation and decision making. Hence, the secure storage for these data has attracted more attention recently. This paper proposes a novel approach, named Mchain, to enhance the integrity and controllability of the secure storage. Especially, to enhance the integrity, a two-layer blockchain network is introduced. In the first layer, after the production, the data packages are first verified by leveraging a correspondence between a package and a policy, and a one-to-one relation among a VM, a user, and a node. After that, we propose a consensus achievement algorithm to construct a semi-finished block on a candidate block arranged by data packages. Meanwhile, the semi-finished block is distributed to all nodes, which can provide a certain integrity. In the second-layer, tamper-resistant metadata is generated by performing PoW tasks on the semi-finished block, resulting in strong integrity. Further, to enhance the controllability, a revisable user-defined policy-based encryption method with KP-ABE is proposed. It helps to flexibly control the scope of authorized verifiers. The experimental results on six scenarios with simulated data set show that the proposed approach is appealing in integrity and controllability, and the time overhead of data storage.

INDEX TERMS VM measurements, secure storage, blockchain, IaaS cloud, integrity.

I. INTRODUCTION

Cloud computing is widely used by governments and companies worldwide. It supports massive data storage and high performance computation. The major advantages of cloud computing include resource allocation on demand, pay as use and dynamic provision. It can greatly reduce the cost of business deployment and management. However, migrating data and applications outside administrative control to a cloud delegates the responsibility too. The cloud is supposed to provide equal or stronger security protection. Nevertheless, various tasks submitted by different users are assigned to a same cloud environment, introducing risks of security breaches. Besides, the cloud services rely heavily on various dependencies, creating more holes of vulnerabilities and incompatibilities. Therefore security concerns about cloud computing are at the forefront of researches [1], and security has become a quite important factor to decision makers.

Virtual machines (VM) are basic working units in IaaS cloud [2], [3]. They perform the delegated tasks. Therefore the trustworthiness of VMs is important. It closely relates to the security of an IaaS cloud. VM measurements are one kind of representative evidences. They are critical, and becoming a rather appealing target for cyber-attacks. How to ensure the security over such data has always been the focus of studies.

The VM measurements have different forms: hash values, properties, logs or test reports [4], [5]. Lots of work have been proposed to secure the storage of the measurements. Trusted hardware units (TPM [6], SGX [7] and TrustZone [8]) and huge data center are two popular techniques. As the former, the hardware-assisted methods have natural tamper-resistant capabilities, so the integrity and confidentiality of data can be protected very well. Nonetheless, the data access need specific commands or requests which cannot be processed in parallel. Besides the internal storage spaces are limited. These flaws restrict the application of trusted hardware units

in VM measurements secure storage. They cannot satisfy the requirements of flexible data access, parallel processing and massive data storage. As the second technique, a huge data center is controlled by one single authority normally. It is easy to focus on security guarantees. Cryptographic tools and access control policies are classical and effective. Whereas, they may fail when there are internal vulnerabilities. The compromise of keys and policies will allow imperceptible violations to data, leading unpredictable consequences (unknown manipulation). Hence it is necessary to seek new approaches to protect VM measurements in IaaS Cloud.

Blockchain is a new and attractive technology for data storage. It has fascinating properties of distributed consensus, tamper-resistance and non-repudiation. Researchers have made comparisons between the performances of blockchain-based Bitcoin and centralized electronic payment system from three aspects: cost, energy consumption and security [9]. The results showed that the Bitcoin system performed better in counter-attacks of data tampering and zombie networks with same deployment costs. Thus we consider applying the blockchain technique to secure the VM measurements data in IaaS cloud. However there remain some challenges: poor access control and time-intensive PoW tasks (which will be detailed in Section 3). On the basis of addressing these issues, we propose a Mchain construction approach to achieve our goal.

The major contributions are as follows:

(1) A Mchain design employing a two-layer blockchain network to enhance integrity. The validation of data and policy packages in the first-layer and the PoW tasks in the second-layer together provide a strong integrity guarantee.

(2) An access control mode based on revisable policies to restrict the scope of authorized verifiers. Users can update a policy for a VM at any given time.

(3) Appropriate performance of data storage latency from users' perspective. By separating a consensus achievement process from the original blockchain and hiding the time-intensive PoW tasks in the background, the confirmation latency of data packages could be reduced greatly from users' perspective.

The rest of the paper is organized as follows. Section 2 reviews the background and related work. Section 3 provides the challenges and assumption. The design and details of Mchain are presented in Section 4. The analysis of integrity, access control and performance are illustrated in Section 5. In Section 6 the experimental results are demonstrated, and followed by conclusions and discussions in Section 7.

II. BACKGROUND AND RELATED WORK

A. DATA SECURE STORAGE

VM measurements are critical evidences in the evaluation of an IaaS cloud. Lots of works have invested in the secure storage of such data. Among them, there are two major techniques: trusted hardware units and centralized huge data centers.

In the first technique TPM (Trusted Platform Module) [10] is the most representative. It generates measurements for a platform and extends them into a set of PCRs (Platform Configuration Registers). Later a virtual implementation of TPM, vTPM [11] appeared to fit the virtualization demands. Intel SGX [12] is another trusted unit. It allows instantiating enclaves to protect regions of partial (or all) address spaces of an application. TrustZone [13] is an arm secure architecture at first. A secure world is designed to isolate risky direct accesses. With these hardware-enforced approaches the data integrity and confidentiality can be protected very well. However they are not fully fitted to the VM measurements protection in IaaS cloud due to some limits: the additional hardware requirements will increase the cost; the non-universal data access interfaces will hinder the usability; and the limited internal storage spaces in trusted hardware will obstruct the traceability.

In the second technique huge data centers adopt various techniques like cryptographic tools, access control policies and backup technologies in data protection. Encryptions and signatures provided by cryptographic tools are against forgeries [14], [15]. Access control policies are used to restrict the exposure scope of data, e.g. [16]–[18]. Appropriate replication strategies in backup are exploited to ensure data availability and integrity. These techniques make it difficult for attackers. Nonetheless, since the data intensive huge center has made itself an extremely valuable target, there remain risks. In addition, the management failures and corrupted insiders are inevitable, causing more security holes. Hence, it is necessary to seek a new approach to ensure the secure storage of VM measurements in IaaS cloud.

B. BLOCKCHAIN

A new blockchain technology has attracted tremendous interests recently. It was originally used in a digital currency Bitcoin with fascinating tamper-resistance and non-repudiation properties for ledger storage [19]. Transactions are recorded and validated at any time without a centralized regulator. Researchers have demonstrated the blockchain could be applied to multiple settings. For instance, an open-source permissioned network Multichain has been developed to assure high transaction throughput on a private cloud [20]. A middleware Tierion was provided to upload and publish data records into Blockchain network [21]. A blockchain based naming and storage system on top of Namecoin was proposed by Blockstack Labs in Princeton University [22]. The blockchain technology also could be applied to the fields of data storage for good integrity and auditability. For example, Guy et al. implemented an automated access control manager without requiring trust in a third party [23]. They used blockchain transactions in their system to carry access instructions and protect personal data. An architecture ProvChain was presented to collect and verify cloud data provenance by embedding them in blockchain transactions [24]. A blockchain-based database in cloud environments was proposed to

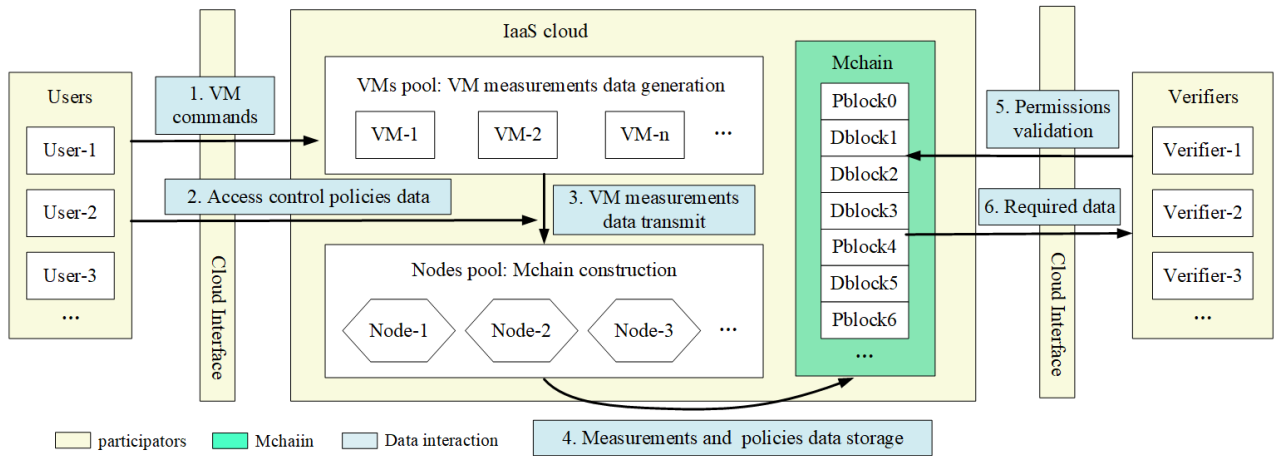


FIGURE 1. Overview of the Mchain design.

provide the desired guarantees on data integrity, performance and stability [25]. These solutions are designed specifically for their own application scenarios. However, when applying the blockchain technology to the secure storage of VM measurements in IaaS cloud, new challenges will arise. We will discuss them in the next section.

III. CHALLENGES AND ASSUMPTION

A. CHALLENGES

The blockchain technology has shown attractive natures on bitcoin ledger protection with tamper-resistance and non-repudiation properties. However, when applying it to the VM measurements secure storage application scenario, new challenges arrive.

1) POOR ACCESS CONTROL

VM measurements data in IaaS cloud are owned by users. They may contain sensitive information. The access to such data should be restricted. Nevertheless, the original blockchain for Bitcoin is a permissive network with a public distributed ledger. Anyone can download, look through and verify a replica of the data. The inconsistency with the access control demand on VM measurements data makes one noticeable challenge.

2) TIME INTENSIVE POW TASK

The confidence of data integrity on blockchain relies on the time-intensive PoW tasks from the mining process. But the mining causes a main defect in inefficiency of data storage: high confirmation latency and low throughput. The latency is a time interval between data submission and storage confirmation. In Bitcoin it is ten minutes, approximately equal to the computation time of a block. The throughput is about seven transactions per second [26]. Comparing with classical data storage, the blockchain is rather poor in efficiency. This forms another challenge of delivering the same integrity

guarantee to VM measurements secure storage with appropriate performance.

B. ASSUMPTION

Attacks on VM measurements data can have many forms: eavesdropping, tampering, forgery, replay, etc. But they have a common goal: manipulating the evidence data to hide other attack tracks. For example, by concealing compromised VMs from being detected, successful lurking backdoors or malicious codes can bring sustainable benefits for attackers. We adopt the same trust assumption as that in the original blockchain network. A small amount of computing power in IaaS cloud is compromised. Thus the VM measurements data are confronted with attacks before or after the storage.

IV. MCHAIN DESIGN

A. OVERVIEW

We propose a VM measurements secure storage approach named Mchain to address the aforementioned two challenges. The description begins with an overview, as shown in Fig. 1. There are three entities: users, IaaS cloud and verifiers. The users control VMs remotely to perform tasks. They store VM measurements on Mchain. The IaaS cloud provides data storage and validation services for users and verifiers. It is built physical nodes, on which the VMs are running. The verifiers are interested in accessing measurements histories of a target VM. Any participants like users, new authorized customers or cloud administrators who want to access VM measurements are classified as verifiers.

From the Fig. 1 in the IaaS cloud there are a Nodes pool and a VMs pool. The Nodes pool contains all physical hosts in the cloud, and the VMs pool contains all virtual machines running on the hosts. Mchain represents the storage location of the VM measurements. It takes up some spaces from the Nodes pool. Physical the VMs pool, Nodes pool and Mchain are merged together. But here for simplicity, three isolated parts are depicted to represent the logical relationships.

There are six procedures in the Mchain design: (1) VM commands. Users issue the commands remotely to initiate VM measurements. (2) Access control policies. Users define the policies to restrict the exposure scope of data. (3) VM measurements data. These data are generated for VMs and transmit inside the cloud. (4) Data storage. The Nodes pool responds for storing the VM measurements data and policies on Mchain. (5) Permissions validation. Verifiers are verified against the corresponding policies to check the permissions. (6) Required data. Only valid verifiers could acquire the required data. The most different place between Mchain and traditional data secure storage approaches lies in procedure four and five.

Our approach is illustrated from three aspects: package production and validation, Mchain construction and access control. They are detailed in following sections.

B. PACKAGE PRODUCTION AND VALIDATION

1) KEY ESTABLISHMENT AND PACKAGE PRODUCTION

Before storing VM measurements on Mchain, credentials are required for both users and VMs. They are created when a user account is registered and a VM is instantiated. We use a public-private key pair to identify the credential. Also an encryption key is needed to protect data confidentiality. We detail the keys as follows.

User Identity Key Pair (PK_u, SK_u): The public-private key pair is a credential for a user u . The SK_u is used to sign all measurements data of VMs belonging to the u , and the PK_u is used to validate and locate them.

VM Identity Key Pair (PK_v, SK_v): The public-private key pair is a credential for a VM v . The SK_v is used to sign all measurements data corresponding to the v , and the PK_v is used to validate and locate them.

VM Encryption Key K_{uv}: The encryption key is created on the basis of a customized VM access control policy P_{uv} with a KP-ABE [27] method, in which case the user is u and the VM is v . The policy is defined by users and revisable at any time. It can be regarded as a discrete function related to time. To identify which policy is used currently, the hash values of a corresponding policy are attached to the end of cipher texts.

If a piece of VM measurements data D_{uv} is ready to store on Mchain, the cipher texts are denoted as C_{uv} .

$$C_{uv} = K_{uv}(D_{uv})$$

If the hash function is denoted as $HASH$ and the policy changes at the time tc , the hash result of a policy P_{uv} is denoted as $H_{uv}(tc)$.

$$H_{uv}(tc) = HASH(P_{uv}(tc))$$

Signatures for cipher texts C_{uv} by a VM v and a user u are denoted as S_v and S_u respectively.

$$S_v = SK_v(C_{uv}), S_u = SK_u(C_{uv})$$

In our Mchain construction, a time server is required to determine the sequence of VM measurements data.

A timestamp t is attached to the end of a data package. Then a packetized data is:

$$PA_{uv} = C_{uv} || H_{uv}(tc) || S_v || S_u || t$$

In the Mchain design, each package needs to travel on the cloud network. The nodes perform package validation and block building tasks too. The source nodes of packages and blocks need to be identified. Thus it is necessary to create a credential for each node in the cloud.

Node Identity Key Pair (PK_n, SK_n): The public-private key pair is a credential for a Node n . The SK_n is used to sign all packages sent from the n , and the PK_n is used to validate them.

A signature for cipher texts C_{uv} by the source node n is denoted as S_n .

$$S_n = SK_n(C_{uv})$$

Then a VM measurements data package is denoted as PA_{uv} .

$$PA_{uv} = C_{uv} || H_{uv}(tc) || S_v || S_u || S_n || t$$

Similarly for a policy P_{uv} , the corresponding package is denoted as PPA_{uv} .

$$PPA_{uv} = P_{uv} || H_{uv}(tc) || S_v || S_u || S_n || t$$

The VM measurements and policy data package share the same format with different bodies. In data packages the contexts are cipher texts of VM measurements, while in policy packages are policies. In the following validation we take the data package as an example to make illustration. The validation of a policy package is similar.

2) PACKAGE VALIDATION

Under load balancing strategies in IaaS cloud, VMs are possibly scheduled to different nodes as time lapses. The correspondences between VMs and nodes are changing with time. But no matter how a VM migrates, only the node hosting it at the moment has the right to sign its packages. Similar relation exists between the user and the VM. During the entire life-time of a VM, its user barely changes. Even though it does, there is still a correspondence between the new user and the VM. In this case, it is also true that only the current user has the right to sign a VM measurements data package. In an IaaS cloud maintaining the correspondences among users, VMs and nodes is a necessary job. It is easy to know for every node. Thus with the correspondences the packages are validated. If there are violations, we can infer that wrong users or nodes have participated in the package production process. This implies an attack. The validation of a package is shown in Fig. 2.

For convenience of description, we use the packages that have been stored on the Mchain in the depiction. The dash lines denote the package location process from the blocks. The correspondence between a package and a then effective policy is verified explicitly. If a key generated from an old

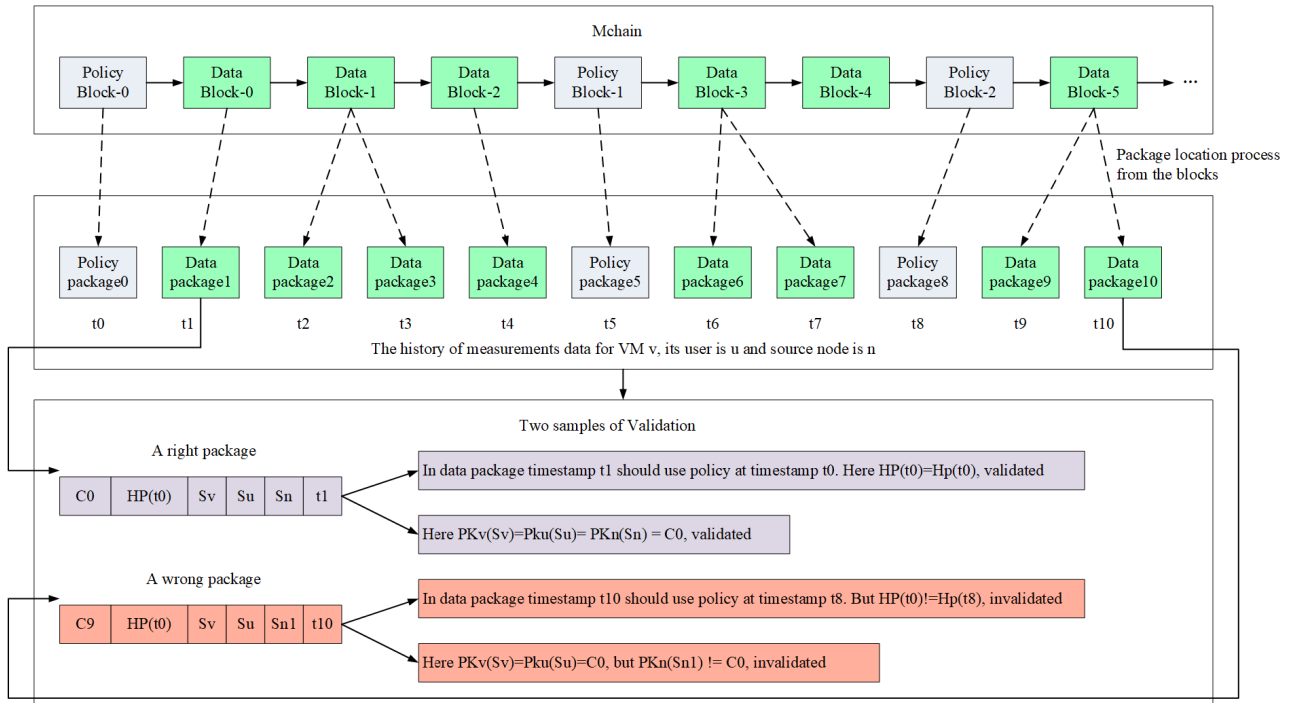


FIGURE 2. Validation of a package.

policy $Huv(tc)$ is used to encrypt a new data package generated at time t , it is a violation. The package is invalid. Besides, the correspondence among the user, VM and source node is verified implicitly. If all three signatures Sv , Su and Sn of a package are right with their own public keys PKv , PKu and PKn , the package is valid.

When a data package travels around the cloud network for validation, there might be attack attempts. For instance, if a package is tampered and uploaded to the network, it would fail the validation. Because a malicious node without a right private key SKn could not forge a right signature. Similarly, if a manipulated package is encrypted with an overdue key, it would fail the validation because of the contradiction to the new policy on Mchain. In this way only a valid package will be accepted. An invalid package will be rejected and alarmed immediately, which impose further obstacles to cover attack tracks.

We give an algorithm to describe the process of package validation as follows.

In algorithm 1, the violations are expressed by two error values 1 and 2 at a coarse-grained level. The specific type of violated correspondence is not distinguished. More error values could be added in step 6 to make a classification at a fine-grained level. It is easy to implement practically. We did not discuss it here.

In some cases the two correspondences are considered private and should not be directly used in a validation. To address this concern, other techniques like homomorphic encryption [28] can be introduced to offer protection. We did not discuss it here either.

Algorithm 1 Package Validation

- 1 procedure Validate ($PAuv, Puv(tc), PKv, PKu, Pkn$)
- $flag = 0$
- 2 The $PAuv$ is a new coming data package, $Puv(tc)$, PKv, PKu and PKn are public information known to all nodes.
- 3 If ($Huv(tc) = HASH(Puv(tc) \& \& t > tc)$)
- The $PAuv - Puv$ correspondence is ok.
- 4 Else $flag = 1$, jump to step 10
- 5 If ($PKv(Sv) == PKu(Su) == Pkn(Sn) == Cuv$)
- The $VM - user - node$ correspondence is ok
- 6 Else $flag = 2$, jump to step 10
- 7 All correspondences are ok, the package is valid.
- 8 There are some violations, the package is invalid.
- 9 Return $flag$
- 10 End procedure

C. MCHAIN CONSTRUCTION

Integrity is the most predominant property when the VM measurements data are stored as evidences. Concurrent access is another necessary demand in an IaaS cloud. The blockchain is promising in protecting both integrity and usability. But the time-intensive built-in PoW tasks result in a high confirmation latency to data storage. To improve the efficiency, we adopt a two-layer blockchain network [25], [29]. The workflow of Mchain construction is shown in Fig. 3. A semi-finished block is output from the first-layer, and a corresponding mature block is output from the second-layer, which is stored on the Mchain.

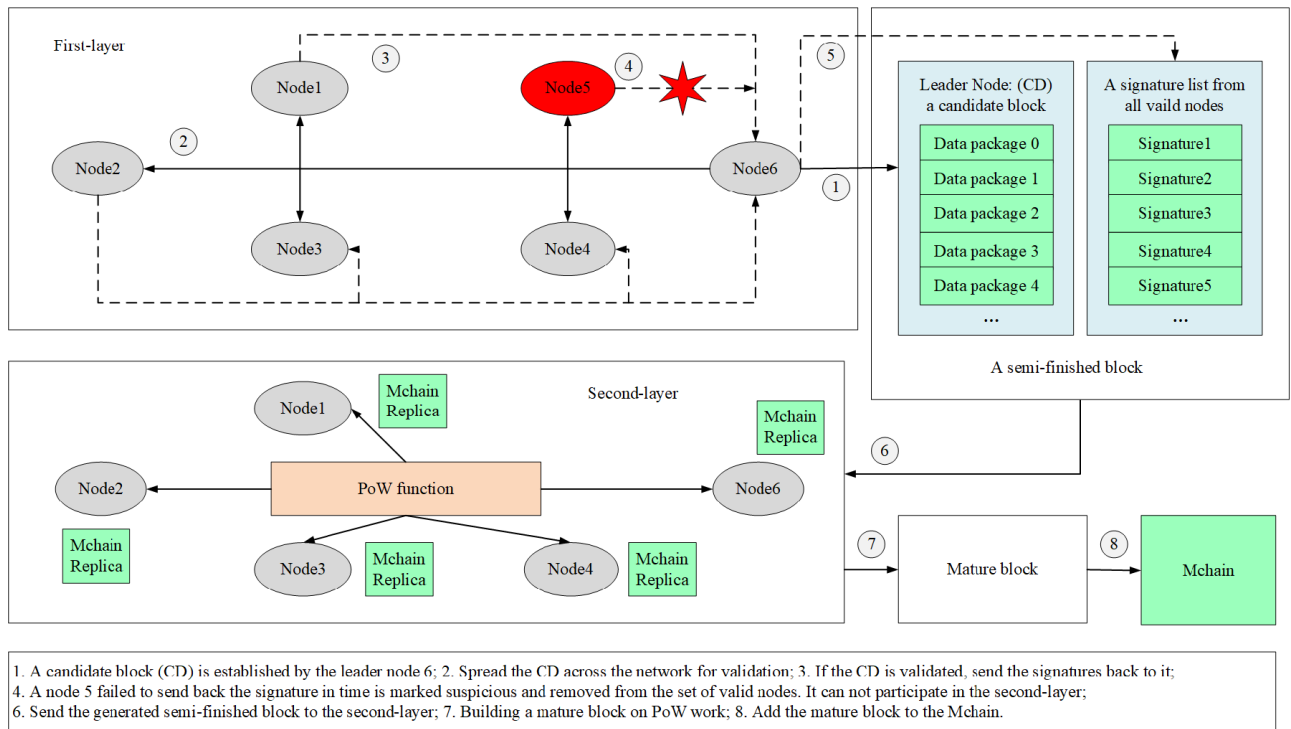


FIGURE 3. Mchain construction.

There are three stages in Mchain construction: candidate block establishment, consensus achievement and block building. They are illustrated below.

1) CANDIDATE BLOCK ESTABLISHMENT

A candidate block contains a group of packages ready to get consensus achievement. It is established by a leader node. The selection of a leader node is detailed in Section C-3.

There are two types of packages: VM measurements data and access control policies. Then there are two types of blocks: data block and policy block. They are stored on one Mchain alternatively. With different data and policies production rates, the arrangement of data and policy blocks forms different storage patterns, as shown in Fig. 4.

For intensive VM measurements, a candidate data block can always be established in a waiting time window t_{dw} . The window is selected based on performance. We use a time sequence rule to shorten the list of candidate packages. The longest waiting package ranks the top in a candidate data block. The rule can ensure a limited and acceptable waiting time for each package. It explains the right upper case in Fig. 4.

For sparse policies, a candidate policy block may have to wait a long, unpredictable time before it is fulfilled. However, this is not practical. A new revised policy should take effect as soon as possible after it is submitted. We use an argument t_{pw} to denote the acceptable waiting time before a revised policy effects. A candidate policy block is constructed based on an old one. When a new policy arrives, it replaces a corresponding old one, or adds it. Then the candidate waits for a time

window t_{pw} . During the period, the candidate block is open for new policies. And when the time window is closed, it is established and goes to the first-layer. Note that the maximum size of a policy block is not limited for now. Due to the one-to-one correspondence between a VM and a policy, the policies and VMs have the same number. All policies are arranged in one candidate block. They can be separated in practical implementation.

Therefore there are two candidate package buffers on each node. One is for data and the other is for policies. They are counted individually.

Though the VM measurements data are intensive in most cases, they could be sparse occasionally. When in a data waiting time window t_{dw} the accumulated data packages are not enough to fulfill a standard candidate data block. A smaller candidate is acceptable. A maximum size of a data block is limited, while a minimum is not. It explains the right lower case in Fig. 4.

2) CONSENSUS ACHIEVEMENT

All candidate blocks established in the last section go to the first-layer of Mchain design for confirmation. We present a lightweight distributed consensus algorithm to assure low confirmation latency and high data storage throughput. Our aim is to reach a quick and reliable agreement on a candidate block among all nodes. Time is divided into rounds. The consensus achievement work is performed in each round.

In a round, a candidate block travels around the cloud network. When it arrives a node, all packages are verified. If validated, a valid mark is tagged. The mark is a signature

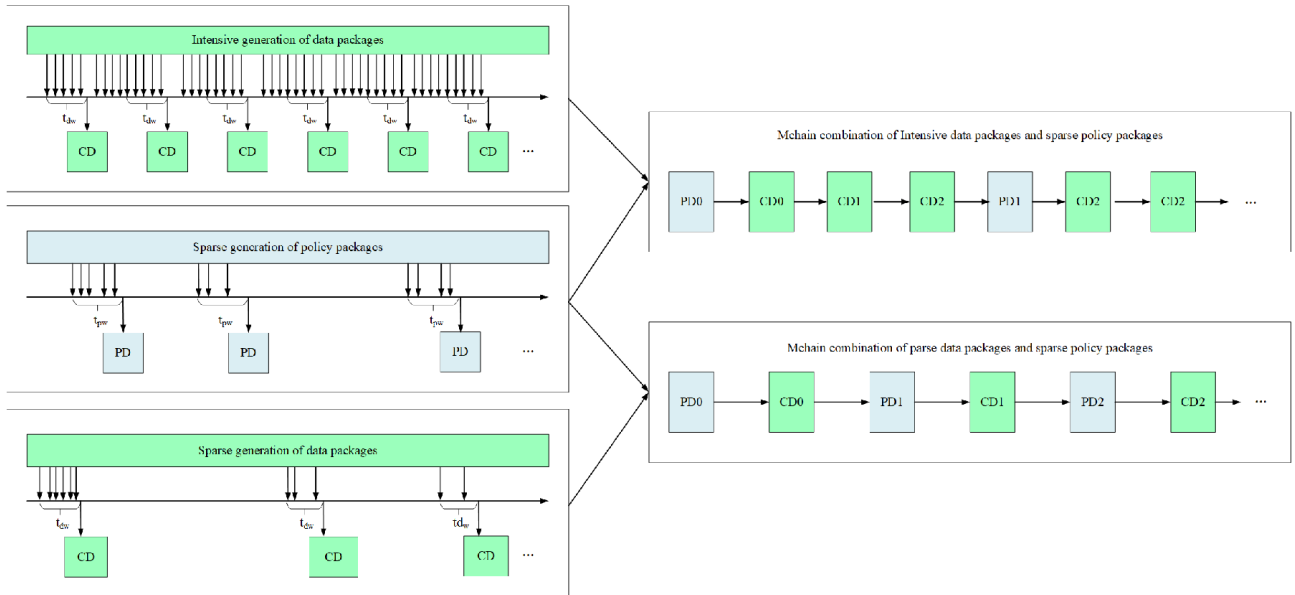


FIGURE 4. Two blocks storage patterns on Mchain.

produced by the node. If all valid marks are gathered, a consensus on the candidate block is achieved. It is then confirmed as a semi-finished block and sent to the second-layer.

If any package in the candidate block is invalid, it is rejected. The leader in this round is marked as suspicious, and removed from the valid nodes pool. The incident would be alarmed to the cloud administrator. A random selection algorithm then starts to choose a new leader.

We give an algorithm to describe the process of consensus achievement as follows.

When a candidate block is confirmed, a semi-finished block is output, and the round is over. It goes to the second-layer of Mchain for storage.

3) BLOCK BUILDING

All valid marks tagged on a semi-finished block in the first-layer guarantees integrity and authenticity to some extent. But it still needs the mining process to ensure strong integrity. We accomplish the PoW tasks in the second-layer of Mchain. A mature block is built on a semi-finished block and stored on the Mchain.

Unlike the permissionless Bitcoin blockchain, the Mchain second-layer network is permissioned. As the system is running, only valid (honest) nodes are allowed to participate in this process.

When a semi-finished block reaches the second-layer, all valid nodes begin the mining process. They construct a block header for the semi-finished block and try to find an eligible nonce. In the original blockchain, the nonce is to satisfy a difficulty, which is of the same use in Mchain. The difficulty is originally defined as the number of first zeros in the hash values of a block header. But in our work there are two types of blocks, so we need to adjust the difficulty slightly.

Algorithm 2 Consensus Achievement

- 1 procedure Consensus(candidate packages buffer)
- 2 $flag = 0$
- 3 Elect the leader node $nleader$ in this round.
- 4 The leader node establishes a candidate_block from the local candidate package buffer.
- 5 Sign the candidate_block with the private key $SKnleader$ and broadcast it on cloud network.
- 6 When the candidate_block arrives a new node $nnew$.
- 7 For package in candidate_block
- 8 If (Validate(package) == True)
- 9 Proceed the next package in candidate_block, $flag = 0$.
- 10 Else $flag = 1$
- 11 End For
- 12 If $flag = 0$
- 13 The candidate_block is valid, sign it with its private key $SKnnew$, and tag a valid mark.
- 14 Else the candidate_block is invalid, mark the node $nleader$ as suspicious, alarm the cloud administrator, and start a random election algorithm to choose a new leader, jump to step 5.
- 15 Once all valid marks from all valid nodes are gathered, the candidate_block is confirmed as a semi-finished block.
- 16 return $flag$
- 17 End procedure

The first bit of hash values for a block header is used as a flag. Zero means a data block, and one means a policy one. If a parameter $Diff$ represents the number of the first bits in block header hash values, the first flag bit is counted too.

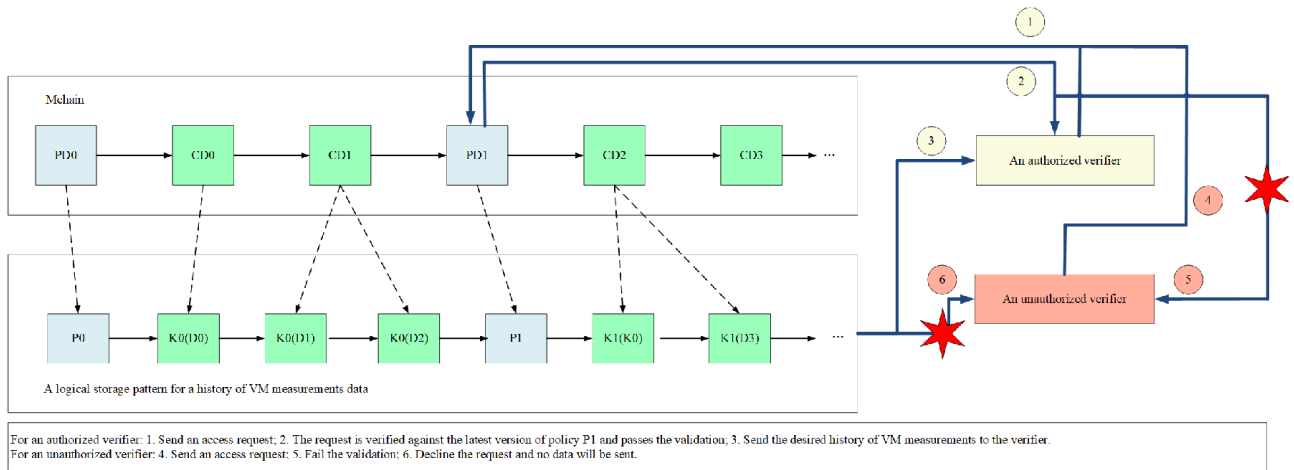


FIGURE 5. A sketch for data access control.

For example, if $Diff = 15$, then the first 15 bits in a data block header hash values are zeros, and that in a policy block are 1 one followed by 14 zeros.

The main structure of a block is similar to the original blockchain. It is composed of two parts: a header and a body. The header contains seven fields: current block hash, previous block hash, Merkle root, timestamp, $Diff$, height and nonce. The current block hash is computed from values in all the following fields. The Merkle root is the root value of a binary hash tree created on all packages in a semi-finished block. Each package is hashed as a Merkle tree node. All the packages constitute the block body.

Since the aim of our Mchain design is to store the VM measurements, the nodes do not have to compete for rewards. They can search complementary random number spaces cooperatively, forming an improvement to performance too. When a nonce is found, the semi-finished block becomes mature. The mature block is then sent to the cloud network and accepts validation from every other node. When it is validated, the mature block will be added to the local copy of Mchain. Occasionally there are two blocks built in one round, forming forks. The issues is solved with the same solution in original blockchain. After a certain time a longest chain will be naturally selected as the primary Mchain.

If a mature block is stored on Mchain, it is considered immutable and irreversible. The block hash value would act as metadata for proving and validating the integrity of VM measurements and policies.

D. ACCESS CONTROL

In an IaaS cloud all VM measurements data belonging to different users are stored on one Mchain. There might be sensitive information in these data. Access to them should be restricted. To achieve this goal, we use a KP-ABE encryption method to implement protection. In this method, if a verifier wants to obtain the measurements of a VM, he/she has to submit a request with granted permissions. The request is

validated against the current effective policy on Mchain. Only a validated verifier can get desired data packages and decrypt them.

There still remains an issue. When accessing a history of measurements data of a VM, the encryption keys might be different, because the corresponding access control policies are revisable. Thus the authorized verifier who meets the latest policy will fail the old ones, and the old data packages cannot be decrypted. To deal with it, the previous key will be encrypted with the new key to form a data package. It is stored on Mchain along with other measurements. In this way even when an authorized verifier retrieve the history of VM measurement data for a specified time, he/she can obtain the old keys at the same time, and then the old data packages are accessible. A sketch for data access control is presented in Fig. 5.

The latest policy block on Mchain is the only interface to access the VM measurements data. Only validated requests are responded. If a user wants to modify the scope of verifiers, he/she just needs to update a new access control policy. Besides, to prevent possible collision attacks for same policies, a one-way increasing value is adopted to control the policy version. Hence even for the same policies, the keys are not the same.

V. ANALYSIS

A. INTEGRITY

Integrity is essential for VM measurements data. We adopt a two-layer blockchain network to ensure.

The first-layer provides a weak integrity. A consensus achievement algorithm is performed. The integrity and authenticity of data and policy packages are validated. If an attacker wants to fake a semi-finished block, He/she needs to do following things: disabling the leader node, deceiving the random election algorithm to be elected as the next leader, raising a candidate block and gathering all valid marks (signatures). And if an attacker wants to manipulate

a semi-finished block, all replicas on all nodes in the first-layer need to be modified. Obviously, either of the two attacks costs a lot of efforts. But they remain possible. The integrity offered in the first-layer is not strong enough. More protections are required.

The second-layer provides a strong integrity. A mature block is built on the PoW tasks cooperatively by all nodes. The more blocks stored on top of a block, the safer it is. Because the efforts to subvert the integrity of a PoW-based blockchain is close to infinite [30]. Thus it is barely infeasible to manipulate the stored data on Mchain. In this way a strong integrity is guaranteed.

B. ACCESS CONTROL

We use a combination of a KP-ABE encryption and a user-defined revisable policy to enhance the access control. With the encryption, the VM measurements data are stored in cipher texts. The encryption key relates closely to a user-defined policy. The policy is revisable at any time, providing flexible control on the scope of verifiers.

An authorized verifier can download all measurements data of a VM. Although the encryption keys are different for the revisable policies, the old keys are stored along with the data packages. When the cipher texts are decrypted, the old keys can be obtained immediately. In this way an authorized verifier can access all measurement data unimpededly.

The newest policy block on Mchain is the only interface to control access. A verifier who fails the validation cannot download any contents from the Mchain. Attackers inside the IaaS cloud can access a Mchain copy. However, unless the newest policy is satisfied, the data can be decrypted neither.

Besides, a one-way increasing value is introduced to handle the version of policies. Whenever the policy updates, the value increments by 1. In this way for two identical policies, the keys are different, preventing collision attacks.

C. PERFORMANCE

The time-intensive PoW tasks lead to high confirmation latency and low throughput for data storage in blockchain. To solve this issue, the introduced two-layer blockchain network separates data confirmation from the PoW tasks. We perform a consensus achievement algorithm in the first-layer to realize the fast confirmation of data and policy packages. When a semi-finished block appears, it queues up to be built on the Mchain. The storage process is over from the perspective of users. Thus the confirmation latency is greatly reduced, and the throughput increases. Since users do not necessarily concern how blocks are built, unless they are interested. Hence by hiding the PoW tasks in the background of the second-layer, the performance could be improved.

VI. EVALUATION

To evaluate the effectiveness and performance of the Mchain design, we prepare a dataset of reports to simulate different situations of VM measurements data. The reports are generated by a popular Linux integrity analysis software

Tripwire2.4 [31]. A report is regarded as a piece of VM measurements data. It results from an integrity check on a configuration file. 294 VM configuration files are put in one folder to simulate the group of objects to be measured. Repeated checks are performed to generate a history of the measurements. By reading a report, a verifier can know the integrity of a VM configuration.

Instead of using a standard Bitcoin client, we implement a simplified version covering the key functions of Mchain: producing a package, establishing a candidate block and finding a right nonce to satisfy the desired difficulty. The programming language used in our experiments is Python 2.7.13 [32]. OpenSSL 0.9.8zg [33] is applied to provide the crypto library. An openssl instruction `genrsa` is used to generate a public-private key pair. We used SHA256 for the hash function, and AES 128 for the symmetric encryption. As to the KP-ABE based encryption key [34], we use a random 128 bit key that is simply related to a policy to make a simulation.

The experiments are carried out in six scenarios. Detailed information on nodes hardware and software are given in Table 1. The description of scenarios are given in Table 2.

TABLE 1. Details of experimental settings.

Computer	Hardware and software information
C1	CPU Intel(R) Pentium G3250@3.20GHz, 4GB RAM, OS OSX 10.9.5.
C2	CPU Intel(R) XeonE5-2603v4@1.70GHz, 64GB RAM, OS Ubuntu 16.04.2 LTS.CPU Intel(R)
C3	XeonE5-2620v3@2.40GHz, 64GB RAM, OS CentOS release 6.8.

TABLE 2. Illustration of six scenarios.

Scenarios	Description
S1	A cloud with two nodes C1 and C2. C1 is honest and C2 is malicious.
S2	A cloud with two nodes C1 and C2. C1 is honest and C2 is malicious.
S3	A cloud with three nodes C1, C2 and C3. They are all honest.
S4	A cloud with a single node C1, it is honest
S5	A cloud with a single node C2, it is honest
S6	A cloud with two nodes C1 and C2, they are honest

A. EFFECTIVENESS

The effectiveness of Mchain is demonstrated with the ability of tamper resistance. In the first-layer, the integrity of packages is validated against two correspondences: the one-to-one relation between a VM, a user, a node and that between a package and a policy. These correspondences rely on the formal proofs in cryptography, requiring no more experiments. Here we focus on the attempted manipulation attacks on data to be stored on Mchain in the second-layer.

Specifically, the attacks happen inside the IaaS cloud. As assumed in Section 3.2, the nodes can be divided into two parts: malicious and honest. We assume all the malicious nodes are laying low to successfully lurk to the second-layer. If an attacker wants to tamper a stored package on Mchain, he/she has to figure out how many blocks a fake Mchain fork

has fallen behind. The number of blocks that a fake Mchain behind a right one is one important argument to impact the attack success rate. In addition, the proportion of malicious computing power occupied in the IaaS cloud is important too. It directly impacts the attack results. Hence we consider two arguments: the number of blocks a fake Mchain falls behind N and the proportion of malicious computing power.

According to [19], if there are six blocks stored on top of one, the deeper block is considered immutable. It implies that if the gap of block number between the honest and fake Mchain is greater than 6, the longer chain is unbeatable. In Bitcoin system a fork can never go this far before it is eliminated. But in our design, when reducing the difficulty of building a block, it is much easier to appear forks. Thus we set 12 as the gap threshold. This value is used to judge the attack results. If the honest Mchain exceeds the fake one by 12 blocks, the attack fails, and vice versa.

We select the S1 and S2 scenario in this test. Both S1 and S2 consist of two nodes C1 and C2. But the proportions of malicious computing power are different. C1 has a higher CPU frequency, representing the higher computer power. Thus in S1 when C1 is honest and C2 is malicious, it is a simulation of malicious nodes in minority. An opposite situation happens in S2, it is a simulation of malicious nodes in majority. The *Diff* is 15. The tests in each case are repeated 100 times. The experimental results are shown in Fig. 6, where the unit of time is second (s).

The fake (red) and honest (green) Mchain start to build a block at the same height. After an average period of time 80.1134s, there are 30.61 blocks built on C1 and 17.58 built on C2. The block number is an average result of repeated experiments in 100 times. Thus they are not integers. The average building time of every block is 3.1416s on C1, and 5.2171s on C2. In the 100 tests, the honest Mchain on C1 can always exceed the fake one on C2 with 12 blocks upon a time. The fake Mchain can never replace the honest one in the cloud. Therefore the attacks always fail. In case 7 the $N = 7$. It means that the target data package has stored on the honest Mchain. The fake Mchain has fallen behind the honest one by 7 blocks. After an average period of time 41.3728s, there are 21.77 blocks built on C1 and 8.89 on C2. The average building time of every block on C1 is 3.2892s, while that on C2 is 5.4098s. In the 100 tests, the honest Mchain on C1 can always exceed the fake one on C2 by 12 blocks upon a time. The fake Mchain will never replace the honest one in the cloud. The attackers always fail. Therefore when there are a small proportion of malicious nodes (computing power), the primary Mchain can always stay in honest with our two-layer blockchain-based technology.

Fig.6 (b) shows the attack results in S2. The fake Mchain has fallen behind the honest one by different number of blocks: 2, 3, 4, 5, 6 and 7. They are also represented by black rectangles. Take case 3 as an example. The $N = 3$. It means the fake (red) Mchain has fallen behind the honest (green) one by 3 blocks. After a period of time 92.9148s, there are 26.85 mature blocks built on C1 and 13.20 built on C2. The average building time of every block on C1 is 3.3989s, while that on C2 is 8.1117s. In the 100 tests, the fake Mchain exceeded the honest one 81 times. It means that there are some probabilities of successful attacks in this context. Although there are 19 times the attacks failed, it is because that a nonce searching process contains a certain amount of luck. If the honest Mchain is always lucky, it could exceed the fake fork by 12 blocks. This may happen, but with a low probability. Given enough time, the attack would succeed eventually. Just like the following cases, the attack success rates in case 5 and 6 are up to 100%. Thus when the assumption in Section 3.2 is violated and the malicious computing power is in majority, the attacks are possible.

In our Mchain design, the nodes entering the second-layer network are permissioned. Abnormal nodes are removed in the first-layer. Although some malicious nodes might lurk carefully, unless they have occupied the major computing power of an IaaS cloud, the attacks would never succeed. The efforts of violating our assumption of Mchain is great. Considering the financial benefits, it is not worthy. Thus to this point, our Mchain could provide perfect integrity to VM measurements data storage.

B. PERFORMANCE

The time overhead in Mchain construction mainly reflects from two aspects: candidate block confirmation and block building. The experiments are in two parts too.

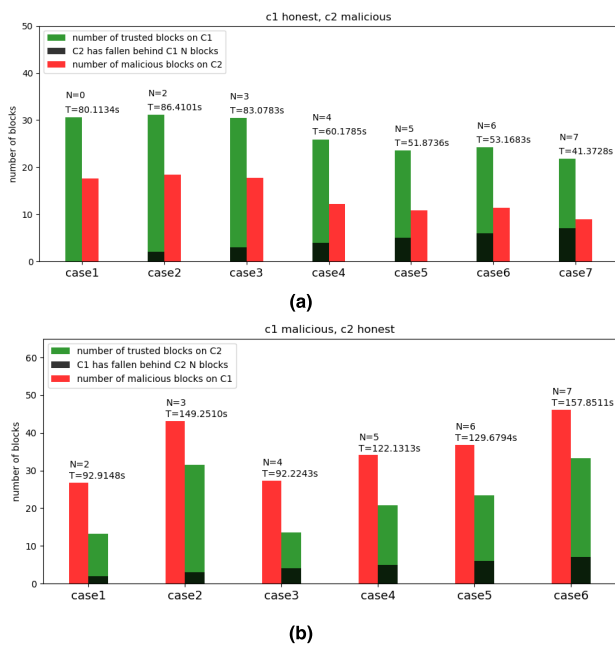


FIGURE 6. Attacking results in scenario s1 and s2. (a) S1. (b) S2.

Fig.6 (a) shows the attack results in S1. The fake Mchain has fallen behind the honest by different number of blocks: 0, 2, 3, 4, 5, 6 and 7. They are represented by black rectangles. Take case 1 as an example. In this case the $N = 0$. It means the target data package is in current semi-finished block.

1) CANDIDATE BLOCK CONFIRMATION

The production of a semi-finished block could be viewed as a sign of successful data storage from the perspective of users according to the analysis in Section 5.3. We measure the time of achieving an agreement on a candidate block to evaluate the confirmation performance. The experiment is performed in scenario S3, a cloud with three nodes. There are two cases of data transmission: in order and concurrent. In the first case a candidate block is verified and forwarded on nodes one by one. While in the second case the process is performed concurrently. The size of a package is 2.96K bytes in our experiment environment. The number of packages in a candidate block ranges between 20 and 200 with a step 20. The experimental results are shown in Fig. 7, where the unit of time is millisecond (ms).

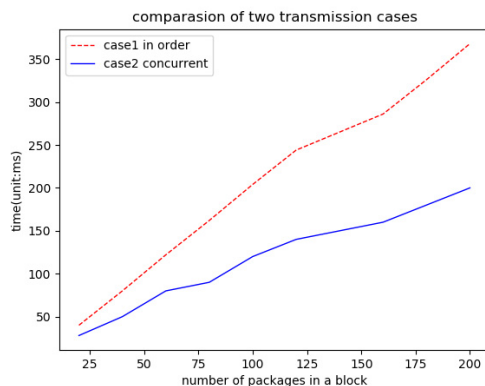


FIGURE 7. Confirmation time in S3.

Intuitively, it is observed that the confirmation time of a candidate block is positively correlated to the size of a block (the number of packages) without considering the network factor. It is relieved from the heavy intensive PoW tasks in the original blockchain technology. The unit of time here is millisecond, one or two order of magnitude less than that of block building (second in Section 6.1). By separating the mining work out, the data confirmation time is largely reduced from the perspective of users.

2) BLOCK BUILDING

The block building time in second-layer is influenced by two parameters: total computing power of a cloud and the difficulty *Diff*. We perform multiple tests on a same dataset with 2000 blocks (123M bytes) in three scenarios S4, S5 and S6. The difficulty values are set to 16, 17, 18 and 19. Each test is repeated 1000 times. The results are shown in Fig. 8.

The time overhead in different scenarios is represented by rectangles with different colors. When *Diff* = 16, the block building time in S4 is 7411s, 9628s in S5, and 5293s in S6. Since in S6 the cloud with two nodes C1 and C2 has the overwhelming power among the three scenarios, its time overhead is supposed to be the least. The experimental results confirm this. It is also found that the block building time is positively correlated to the total computing power, and negatively to

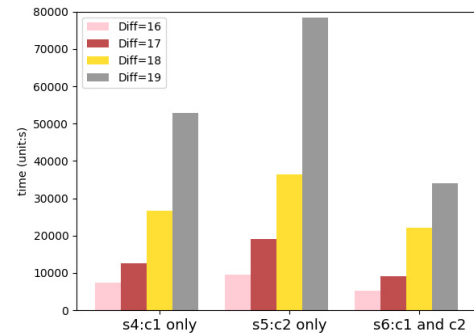


FIGURE 8. Block building time in S4, S5 and S6.

the difficulty. When the total computing power is fixed, the time to build a block could be adjusted by varying the difficulty value according to the practical requirements.

In addition to the data and policy packages, there are some metadata generated for each block in the process of Mchain construction. 1Mbytes bytes data are added in our tests in total. Comparing with the size of VM measurements data and policies, the amount of metadata is small. Hence the extra introduced storage overhead is acceptable.

VII. CONCLUSIONS AND DISCUSSION

In this paper we first discuss the specific requirements of VM measurements secure storage in IaaS cloud. The new blockchain technology has shown promising natures to meet these demands, along with two challenges: controllability and performance. To solve these issues, we present a Mchain approach. The major advantage is that the integrity and controllability are well balanced with appropriate performance. To enhance the integrity, a two-layer blockchain-based network is introduced. In the first layer, the data packages are verified against two correspondences: the one-to-one VM-user-node and package-policy relation. After that, a consensus achievement algorithm is proposed to construct semi-finished blocks on the candidate blocks arranged by data packages. In the meanwhile, the semi-finished blocks are distributed to all nodes to provide a certain integrity. In the second-layer, tamper-resistant metadata are generated by performing PoW tasks on the semi-finished blocks to ensure a strong integrity. Further, to enhance the controllability, a revisable user-defined policy based a KP-ABE encryption method is proposed to flexibly restrict the scope of verifiers. We conduct two types of experiments on six scenarios with simulated dataset to make evaluation. The experimental results showed that the proposed approach is appealing in integrity and controllability, as well as the time overhead of data storage.

The future plan of improving our Mchain design is extending the current design to a general secure storage approach for more data types with a better and flexible access control. The PoW function used in current work depends on finding an eligible nonce, which is the same as Bitcoin system. It only generates tamper-resistant metadata. We expect to find

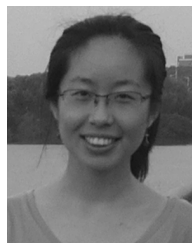
new functions that achieve similar goal with more practical usage. Besides, the correspondences used in package validation might be considered sensitive in some contexts. Finding new validation proofs is another interesting job. We hope our research work will yield a new insight to the use of blockchain technology.

REFERENCES

- [1] P. Dinadayalan, S. Jegadeeswari, and D. Gnanambigai, "Data security issues in cloud environment and solutions," in *Proc. IEEE World Congr. Comput. Commun. Technol.*, Feb./Mar. 2014, pp. 88–91.
- [2] Q. Duan, Y. Yan, and A. V. Vasilakos, "A survey on service-oriented network virtualization toward convergence of networking and cloud computing," *IEEE Trans. Netw. Service Manage.*, vol. 9, no. 4, pp. 373–392, Dec. 2012.
- [3] B. Guan, J. Wu, Y. Wang, and S. U. Khan, "CIVSched: A communication-aware inter-VM scheduling technique for decreased network latency between co-located VMs," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 320–332, Jul./Sep. 2014.
- [4] T. Zhang and R. B. Lee, "CloudMonatt: An architecture for security health monitoring and attestation of virtual machines in cloud computing," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit. (ISCA)*, vol. 43, no. 3, Jun. 2015, pp. 362–374.
- [5] L. Chen, R. Landfermann, H. Löhr, M. Rohe, A.-R. Sadeghi, and C. Stübke, "A protocol for property-based attestation," in *Proc. ACM Workshop Scalable Trusted Comput.*, 2006, pp. 7–16.
- [6] P. Fan et al., "An improved vTPM-VM live migration protocol," *Wuhan Univ. J. Natural Sci.*, vol. 20, no. 6, pp. 512–520, 2015.
- [7] A. Baumann, M. Peinado, G. Hunt, K. Zmudzinski, C. V. Rozas, and M. Hoekstra, "Secure execution of unmodified applications on an untrusted host," in *Proc. Symp. Oper. Syst. Princ.*, 2013, p. 1.
- [8] S. Zhao, Q. Zhang, G. Hu, Y. Qin, and D. Feng, "Providing root of trust for ARM trustzone using on-chip SRAM," in *Proc. 4th Int. Workshop Trustworthy Embedded Devices*, 2014, pp. 25–36.
- [9] J. Becker, D. Breuker, T. Heide, J. Holler, H. P. Rauer, and R. Böhme, "Can we afford integrity by proof-of-work? Scenarios inspired by the bitcoin currency," in *The Economics of Information Security and Privacy*. Rochester, NY, USA: Social Science Electronic, 2013, pp. 135–156.
- [10] L. Chen and M. Ryan, "Attack, solution and verification for shared authorisation data in TCG TPM," in *Proc. Int. Workshop Formal Aspects Secur. Trust*, 2009, pp. 201–206.
- [11] C. Chen, H. Raj, S. Saroui, and A. Wolman, "cTPM: A cloud TPM for cross-device trusted applications," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement.*, 2014, pp. 1–16.
- [12] F. Schuster et al., "VC3: Trustworthy data analytics in the cloud using SGX," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 38–54.
- [13] D.-A. Suciuc and R. Sion, "Droidsentry: Efficient code integrity and control flow verification on trustzone devices," in *Proc. Int. Conf. Control Syst. Comput. Sci.*, May 2017, pp. 156–158.
- [14] S. Rajak and A. Verma, "Secure data storage in the cloud using digital signature mechanism," *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 1, no. 4, p. 489, 2012.
- [15] J. Chen and H. Wee, "Semi-adaptive attribute-based encryption and improved delegation for Boolean formula," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, 2014, pp. 277–297.
- [16] L. Zhou, V. Varadharajan, and M. Hitchens, "Trust enhanced cryptographic role-based access control for secure cloud data storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2381–2395, Nov. 2015.
- [17] R. Saikerehana and A. Umamakeswari, "Secure data storage and data retrieval in cloud storage using cipher policy attribute based encryption," *Indian J. Sci. Technol.*, vol. 8, no. 9, pp. 318–325, 2015.
- [18] R. M. Jogdand, R. H. Goudar, G. B. SayedPratik, and B. Dhamanekar, "Enabling public verifiability and availability for secure data storage in cloud computing," *Evolving Syst.*, vol. 6, no. 1, pp. 55–65, 2015.
- [19] *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Mar. 25, 2018. [Online]. Available: <https://cryptohuge.com/en/bitcoin-paper.html>
- [20] *Multichain Private Blockchain White Paper*. Accessed: Mar. 25, 2018. [Online]. Available: <http://www.multichain.com/download/MultiChain-White-Paper.pdf>
- [21] *Tierion Api*. Accessed: Mar. 25, 2018. [Online]. Available: <https://tierion.com/app/api>
- [22] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2016, pp. 181–194.
- [23] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.
- [24] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proc. IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2017, pp. 468–477.
- [25] E. Gaetani, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Blockchain-based database to ensure data integrity in cloud computing environments," in *Proc. Italian Conf. Cybersecur.*, 2017, pp. 1–10.
- [26] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Proc. Privacy Secur. Risk Trust*, 2011, pp. 1318–1326.
- [27] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
- [28] D. Boneh and C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.
- [29] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, "Bitcoin-NG: a scalable blockchain protocol," in *Proc. USENIX Conf. Netw. Syst. Design Implement.*, 2016, pp. 45–59.
- [30] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2015, pp. 281–310.
- [31] *Tripwire*. Accessed: Mar. 25, 2018. [Online]. Available: <https://www.ibm.com/developerworks/cn/aix/library/au-usingtripwire>
- [32] *Python*. Accessed: Mar. 25, 2018. [Online]. Available: <https://www.python.org/>
- [33] *OpenSSL*. Accessed: Mar. 25, 2018. [Online]. Available: <https://www.openssl.org/>
- [34] H. Dang, Y. L. Chong, F. Brun, and E.-C. Chang, "Fine-grained sharing of encrypted sensor data over cloud storage with key aggregation," *IACR Cryptol. ePrint Arch.*, 2015.



BO ZHAO was born in 1972. He received the Ph.D. degree in computer software and theory from Wuhan University, Wuhan, China, in 2006. He is currently the Vice-President of the School of Cyber Science and Engineering, Wuhan University. He is also the Vice Director of the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan University. He is the author of two books and has published over 70 articles. His research interests include information system security, trusted computing, embedded system, and cloud computing security. He is a senior member of CCF.



PEIRU FAN was born in 1990. She received the B.Sc. degree in electronic information science and technology and English from Shanxi University, Taiyuan, China, in 2011, and the M.Sc. degree in signal and information processing from Tianjin polytechnic University, Tianjin, China, in 2014. She is currently pursuing the Ph.D. degree in information security with Wuhan University, China. Her current interest lies in the area of virtualization, cloud computing, and security.



MINGTAO NI was born in 1977. He received the B.Sc. degree in mechatronics from the Wuhan University of Technology, China, in 1998, and the M.Sc. degree in software engineering from the University of Electronic Science and Technology of China in 2008. He is currently pursuing the Ph.D. degree in information security with Wuhan University, China. His research interests include embedded system, IoT security, and mobile device security.