Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors

Yuan Zhang, *Student Member, IEEE*, Chunxiang Xu, *Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

Abstract—The deployment of cloud storage services has significant benefits in managing data for users. However, it also causes many security concerns, and one of them is data integrity. Public verification techniques can enable a user to employ a third-party auditor to verify the data integrity on behalf of her/him, whereas existing public verification schemes are vulnerable to *procrastinating auditors* who may not perform verifications on time. Furthermore, most of public verification schemes are constructed on the public key infrastructure (PKI), and thereby suffer from certificate management problem. In this paper, we propose a certificateless public verification scheme against procrastinating auditors (CPVPA) by using *blockchain technology*. The key idea is to require auditors to record each verification result into a transaction on a blockchain. Because transactions on the blockchain are time-sensitive, the verification can be time-stamped after the transaction is recorded into the blockchain, which enables users to check whether auditors perform the verifications at the prescribed time. Moreover, CPVPA is built on certificateless cryptography, and is free from the certificate management problem. We present rigorous security proofs to demonstrate the security of CPVPA, and conduct a comprehensive performance evaluation to show that CPVPA is efficient.

Index Terms—Cloud storage, data integrity, certificateless public verification, procrastinating auditors, blockchain

1 Introduction

With cloud storage services, users outsource their data to cloud servers and access that data remotely over the Internet [1], [2]. These services provide users an efficient and flexible way to manage their data, while users are free from heavy local storage costs [3], [4], [5]. Although users enjoy great benefits from these services, data outsourcing has also incurred critical security issues [6], [7], [8]. One of the most important security concerns is data integrity [9], [10]. Unlike traditional data management paradigm, where users store their data locally, users would not physically own their data once having outsourced the data to cloud servers. Hence, users are always worried about the data integrity, i.e., whether the outsourced data is well maintained on cloud servers.

The integrity of outsourced data is being put at risk in practice [11], [12]. For example, the cloud servers may always conceal incidents of data corruption for good reputation, or may delete a part of data that is never accessed to reduce the storage costs [13], [14]. Furthermore, an external adversary may tamper with the outsourced data for financial or political reasons [15]. Hence, it is necessary to verify the integrity of outsourced data periodically. The verification can be performed by the users themselves. However, this lays a heavy communication burden on users to retrieve and verify the data.

Public verification techniques enable users to outsource the data integrity verification to a dedicated third-party auditor. The auditor periodically checks the data integrity, and informs the users that the data may be corrupted once the checking fails [16]. In most of public verification schemes, the auditor is assumed to be honest and reliable. If the auditor is compromised, these schemes would be invalidated. For example, an irresponsible auditor may always generate a good integrity report without performing the verification to avoid the verification costs. In such a way, the auditor is virtually non-existent. Furthermore, a compromised auditor may be incentivized by cloud servers to generate a bias verification result to deceive the users for profits. To resist the compromised auditor, the users are required to audit the auditor's behaviors [17], [18], [19]: After each verification, the auditor records the information used to verify the data integrity, which enables the user to audit the validity of the auditor's behavior.

In existing public verification schemes, the auditor needs to perform the verification periodically so that the data corruption can be detected as soon as possible. Actually,

- *Y. Zhang is with the Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China, and also with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: ZY_LoYe@126.com.*
- *C. Xu are with the Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China. E-mail: chxxu@uestc.edu.cn.*
- *X. Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 2W1, Canada. E-mail: xlin08@uoguelph.ca.*
- *X. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 21 Sept. 2018; revised 20 Feb. 2019; accepted 26 Mar. 2019. Date of publication 29 Mar. 2019; date of current version 3 Sept. 2021. (Corresponding authors: Yuan Zhang and Chunxiang Xu.) Recommended for acceptance by X. Deng. Digital Object Identifier no. 10.1109/TCC.2019.2908400

periodical verification can reflect the state of integrity of the outsourced data in each period, which enables the user to find the data corruption within the period. For example, in a cloud-assisted electronic health system, the outsourced electronic health records (EHRs) are sensitive [20], [21], [22], and should be verified periodically to guarantee their correctness. Any time the EHRs are corrupted, the healthcare provider can find it within the period, stops to use the corrupted EHRs, and attempts to recover the EHRs at once. This can protect the healthcare provider against losses caused by the EHR corruption as far as possible. However, an irresponsible auditor may procrastinate on the scheduled verification, due to network failures, system errors, or request from the cloud server.[1] We call this auditor a *procrastinating auditor*, it deviates from the original objective of public verification schemes, i.e., detecting data corruption as soon as possible. It might be too late to recover the data loss or damage if the auditor procrastinates on the verification. In fact, the procrastinating auditor also cannot be detected in the public verification schemes, even though malicious auditors can be detected there [17], [18], [19].

Furthermore, most public verification schemes are built on the public key infrastructure (PKI), where the auditor has to maintain the users' certificates to select correct public keys for verifications. Consequently, these schemes suffer from the certificate management problem (including certificate revocation, storage, distribution, and verification). It is inefficient in practice [23], [24].

In this paper, we present the first certificateless public data integrity verification scheme that resists malicious and procrastinating auditors, dubbed CPVPA. The key idea behind CPVPA is to use *blockchain-based currencies* (i.e., *on-chain currencies*), such as Bitcoin and Ethereum [25], [26], [27], which provide a secure way to conduct transactions without a central authority (i.e., bank). In CPVPA, the auditor is required to create a new transaction after each verification, where the information corresponding to the verification is integrated into the transaction, and the auditor conducts the transaction. After the transaction is recorded into the blockchain, the user can verify the time when the verification was performed by checking the generation time of the transaction. We stress that for a blockchain system, the more participants in it, the stronger security guarantee it can provide. Therefore, we construct CPVPA on a well-established and widely-used blockchain system (e.g., Ethereum), rather than a newly created one. Moreover, CPVPA is built on the certificateless cryptography [28] and avoids the certificate management problem. Specifically, the contributions of this work are summarized as follows.

- We analyze existing public verification schemes, and demonstrate that existing schemes cannot resist a procrastinating auditor who may not perform the data integrity verification on schedule and deviate from the original objective of public verification schemes, i.e., detecting the data corruption as soon as possible.
- We present a certificateless public verification of data integrity scheme, namely CPVPA, to resist malicious and procrastinating auditors, where each verification performed by the auditor is time-stamped by integrating it into a transaction of a blockchain, e.g., Ethereum. Such a mechanism enables the user to check the time when the auditor performs the verification. CPVPA addresses the certificate management problem existing in most of existing public verification schemes.
- We present rigorous security proofs to prove that CPVPA is secure against the procrastinating auditor defined in this paper and the semi-trusted server and the malicious auditor defined in the strongest model [14], [17], [18]. We also provide a comprehensive performance evaluation, and demonstrate that CPVPA has a constant communication overhead, and is efficient in terms of computational overhead on both the server side and the user side.

The remainder of this paper is organized as follows. We motivate CPVPA in Section 2, and define the system model, adversary model, and design goals and present preliminaries in Section 3. We propose CPVPA in Section 4 and analyze its security in Section 5. We provide the performance evaluation in Section 6, review the related work in Section 7, and draw the conclusions and present some open problems in Section 8.

2 Problem Statement

2.1 Public Verification of Data Integrity

The key idea of the public verification technique [13], [14], [15] is that the user (i.e., data owner) splits the data into multiple blocks, computes a signature for each one, and outsources the data blocks as well as corresponding signatures to the cloud server. When the auditor verifies the data integrity, it chooses a random subset of all data blocks (e.g., sample 300 blocks from 10,000 ones) and sends the sampled blocks' indexes (as a challenging message) to the cloud server. The cloud server responses with the corresponding proof, the auditor checks the integrity of challenged blocks by verifying the validity of the proof. If the verification passes, the integrity of entire data set is ensured. The key technique used here is aggregated signature [29], which enables the auditor to verify multiple blocks simultaneously without downloading the data.

In public verification schemes, after data outsourcing, the user sets a verification period (i.e., the frequency at which the auditor performs the verification). Then the auditor verifies the outsourced data integrity at the corresponding time. In practice, the auditor generates a verification report containing multiple verification results (corresponding to multiple periods, we call these periods an epoch). If in any period the verification result is "*Reject*", it means that the data may be corrupted and the auditor needs to inform the user at once. Otherwise, the auditor generates a verification log and provides the user with the log at the end of each epoch. Since the auditor can verify the data integrity without the user's participation, the user can assign the auditor to perform the verification with any period as needed. In other words, from the user's perspective, if the outsourced data is corrupted,

1. When the data corruption occurs, the cloud server may collude with the auditor, where it asks the auditor for halting the scheduled verification, and gains much more time to retrieval the outsoucrced data for good reputation.

the longest delay within which she/he needs to find the data corruption should be the verification period.

We stress that the frequency at which the auditor checks the data integrity would not be very high in practice, due to the following reasons. First, the auditor serves multiple users simultaneously. If users require the auditor to perform the data integrity verification with a high frequency, e.g., performing the verification every hour, the auditor would bear a heavy communication and computation burden. Furthermore, the higher frequency to perform the data integrity verification, the more costs to employ the auditor. From a pragmatic standpoint, users would not require the auditor to perform data integrity verification with a high frequency in existing scenarios. Second, performing the data integrity verification with a high frequency would also cause heavy verification burden on the cloud server. As pointed out by Armknecht et al. [17], if integrating security mechanisms into existing cloud systems incurs considerable costs on the cloud service providers, most of the providers would not accept liability for the corresponding security guarantees in their Service Level Agreements (SLAs) and only ensure the service availability.

2.2 On the Vulnerability of Existing Public Verification Schemes against Procrastinating Auditors

In most of existing public verification schemes [14], [16], [30], auditors are assumed to be honest and reliable. This means that the auditor would honestly follow the prescribed schemes, and performs the verification reliably.[2]

These schemes cannot resist malicious auditors. The most trivial attack a malicious auditor can perform is that it always generates a good integrity report without verifying the data integrity to avoid the verification burden. To thwart such the attack, the user is able to audit the auditor's behavior at the end of each epoch. However, a more tricky attack still exists in the mechanism: the auditor colludes with the cloud server, and always generates bias challenging messages such that only the data blocks which are well maintained are verified, this avoids revealing the data corruption. To resist this attack, the challenging messages should not be predetermined by any participant. Existing schemes [17], [18], [19] utilize Bitcoin to generate the challenging messages to ensure the randomness of challenged data blocks, where the auditor extracts the hash value of the latest block from the Bitcoin blockchain, and generates the challenging message according to the security parameter and the extracted hash value. Since in the Bitcoin blockchain the hash value of a block generated at a future time is unpredictable, this ensures that the auditor cannot generate a bias challenging message to deceive the user, and enables the user to efficiently audit the auditor's behavior.

However, such the mechanism is vulnerable to a procrastinating auditor. Assuming the agreed verification period is 1 day, and an epoch is 1 month (i.e., 30 days), this means that the auditor checks the outsourced data integrity one time per day, and the user audits the auditor's behaviors one time per month. Normally, the auditor would perform the verification every day and generate a verification report every 30 days. For a procrastinating auditor, it would not perform the verification on the first 29 days, and would perform the verification 30 times on the last day, where the challenging messages in each verification of the first 29 days can be regenerated in the 30th day. As such, the verification report only reflects the most recent (the 30th day's) state of integrity for the outsourced data. This deviates from the public verification's original target: if the outsourced data is corrupted, the data owner is able to find it within 1 day (i.e., one verification period).

To resist the procrastinating auditor, a straightforward solution is to require the user to audit the auditor's behaviours in a random time interval. However, before the user audits the correctness of auditor's behaviours, she/he needs to interact with the auditor to obtain the data that records the auditor's behaviours for the auditing, this sufficiently gives rise to forge the data for the auditor and cloud server. As such, a procrastinating auditor can pass the user's auditing by colluding with the cloud server. Another straightforward solution is to introduce a trusted service provider who provides a time-stamping service [33]. After each verification, the auditor is required to query the time-stamping on the information, which is used to check the data integrity, and is used to be audited by the user to prove the correctness of its behavior. This enables the information to be time-sensitive, and therefore can resist the procrastinating auditor. Nevertheless, the security of this mechanism relies on the security and reliability of the time-stamping service provider, and the provider here becomes a single point of failure. Furthermore, the provider has to bear heavy communication and computation burden in the case of multiple users and auditors. As such, how to resist the procrastinating auditor without introducing any trusted entity is a very challenging problem.

2.3 On the (in)efficiency of PKI-Based Public Verification Schemes

Most public verification schemes are built on PKI, where a fully trusted certificate authority issues participants' certificates, and the auditor has to maintain users' certificates to select correct public keys for verifications. However, certificate management (including revocation, storage, distribution, and verification) is inefficient in practice [18], [28]. Therefore, removing the certificate management problem could be economic and favorable in practice.

3 Definitions and Preliminaries

3.1 System Model

As depicted in Fig. 1, four different entities are involved in CPVPA: cloud user (data owner), cloud server, third-party auditor (TPA), and key generation center (KGC).

- User: The user is the data owner, who outsources her/his data to the cloud server and accesses the outsourced data as needed. After data outsourcing, the user employs a TPA, agrees a verification period with TPA, and let TPA periodically verify the data integrity.

2. Some works [15], [31], [32] assume that auditors are honest but curious, however, from the perspective of data integrity verification, there is no difference between these two assumptions, since the auditors would not deviate from the prescribed schemes.

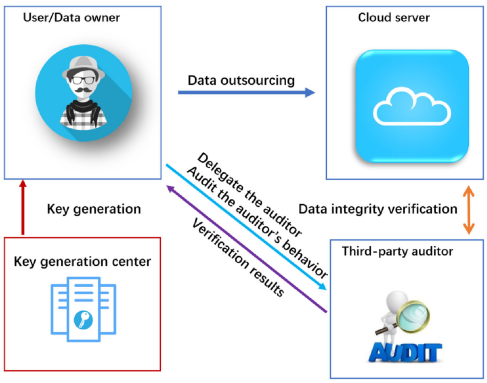


Fig. 1. System model.

- Cloud server: The cloud server is subject to a cloud service provider. It provides storage services for users. It has a huge amount of storage space.
- TPA: TPA works for the user. It generates the verification results based on the proof information from the cloud server, and detects the data corruption as soon as possible. The communication between TPA and other entities is authenticated.
- KGC: The KGC is subject to an authority. It generates a partial private key for the user by using the user's identity.

A formal definition of CPVPA is given in the following:

Definition 1. *CPVPA is composed of five algorithms*, Setup, Store, Audit, LogGen, *and* CheckLog.

- *Setup*. In this algorithm, necessary parameters used in the subsequent algorithms are generated.
- *Store*. In this algorithm, a user outsources the data to a cloud server. The user needs to generate verification tags (i.e., signatures) that enable a TPA to verify the data integrity. Furthermore, the correctness of outsourced data should be confirmed by the cloud server.
- *Audit*. This algorithm enables TPA to check the data integrity, and allows the cloud server to prove that the outsourced data is well maintained.
- *LogGen*. This algorithm enables TPA to generate a log file, which records the TPA's verification information.
- *CheckLog*. This algorithm enables the user to audit the TPA's behavior by checking the validity and correctness of the log file.

3.2 Threat Model

In the threat model, we consider threats from three different aspects: semi-trusted servers, misbehaved auditors, and malicious users.

Semi-Trusted Servers. The cloud server is a semi-trusted entity. We follow the existing threat model of cloud servers [14] with the integration of the threat model of certificateless cryptography [28], [34]. It may hide the incident of data corruption by forging a proof information to deceive TPA. The cloud server may become two types of adversaries:

1) Type I adversary $\mathcal{A}_{\mathrm{I}}$: He is able to replace the public key of the user with a value of his choice, but he cannot access to the KGC's master key.
2) Type II adversary $\mathcal{A}_{\mathrm{II}}$: He is able to access the KGC's master key but cannot perform public key replacement.

Misbehaved Auditors. We extend the existing threat models of malicious auditors [17], [18]. TPA may be compromised, which means that TPA may hide an incident of data corruption from the user by colluding with the cloud server. Furthermore, TPA may deviate from the prescribed verification period, and may not perform the verification on schedule.

Malicious Users. We follow the existing threat model of malicious users [17], [18]. The only attack the user may perform is that he uploads incorrect verification tags (signatures) to circumvent the cloud server.

3.3 Challenges and Design Goals

In this paper, we target at designing a secure public verification of data integrity scheme for cloud storage systems, where two challenges should be addressed:

1) How to resist the procrastinating TPA without introducing any trusted participant. Existing schemes assume that TPA would perform the data integrity verification at the prescribed time. However, procrastinating auditors would not detect the data corruption as soon as possible, and it might be too late to recover the data loss or damage. Such the procrastination is hardly to be detected without a trusted participant's help.
2) How to avoid the certificate management. As discussed before, certificate management is cumbersome and costly in practice. Enabling TPA to verify the data integrity without managing users' certificates could be economic and favorable in practice.

To enable secure verification of outsourced data integrity in cloud storage under the aforementioned model, the following objects should be achieved.

- Efficiency: The communication and computation overhead should be as efficient as possible; TPA is able to verify the data integrity without managing the users' certificates and bearing a priori bound on the number of verification interactions; TPA should be stateless, and is not required to maintain and update state during verification.
- Security: When a cloud server passes the TPA's verification, it must possess the specified data intact; A malicious TPA and a procrastinating TPA cannot deceive the user; Collusion between any two participants cannot break the security of the proposed scheme.

3.4 Notation

Given two integers $u, v \in N, (u \leq v)$, where N is natural number set, we denote by $[u, v]$ the set $\{u, u+1, u+2, \ldots, v\}$. Given a finite set T, $|T|$ denotes the number of components in T. Given two bit-strings x and y, $x||y$ denotes their concatenation.

3.5 Bilinear Pairing

Give two multiplicative groups G and G_T (they have the same prime order p), e: $G \times G \rightarrow G_T$ is a bilinear pairing if it has three properties:

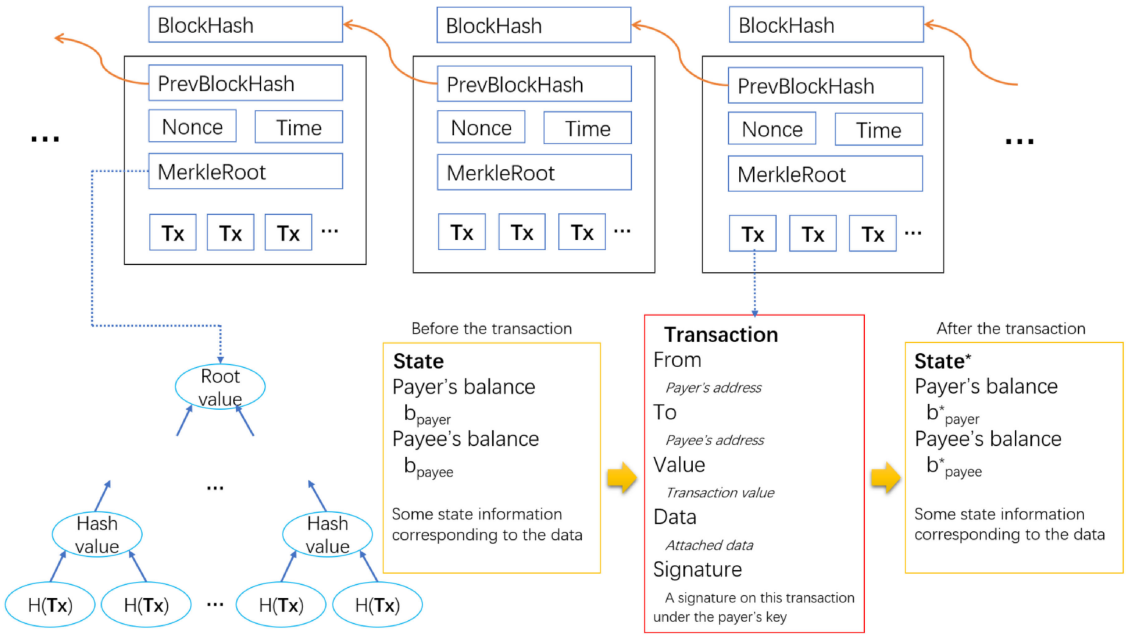


Fig. 2. A simplified Ethereum blockchain.

1. Bilinearity: $e(g^u, q^v) = e(g, q)^{uv}$, $\forall g, q \in G,\ u, v \in Z_p^*$.
2. Non-degeneracy: $\forall g, q \in G$; $g \neq q$, $e(g, q) \neq 1$.
3. It is efficient to compute e.

Computation Diffe-Hellman (CDH) problem in G: given G and one of its generator g, for any unknown $a, b \in Z_q^*$, given g^a and g^b; compute g^{ab}.

3.6 Public Blockchain and On-Chain Currencies

A blockchain is composed of multiple data elements, in which each data element is called a *block*. All blocks form a chain, where the security is guaranteed by utilizing a cryptographic hash function. Each block typically contains a hash pointer as a link to a previous block, a timestamp, and transaction data [25]. Only if a transaction's validity is verified, it can be recorded into the block. Generally, blockchains can be classified into two categories: *private blockchains* and *public blockchains*. In private blockchains (including consortium blockchains), the verifications are performed by authorized participants, who may be employed by the blockchain managers or the managers themselves. In public blockchains, the verifications can be performed by any participant in the network: a transaction can be recorded into a block, only if it has been verified and accepted by a considerable majority [35], [36], [37].

The most prominent manifestation of public blockchain is on-chain currencies, e.g., Bitcoin [25] and Ethereum [26]. In these currencies, the public blockchain is utilized to record a ledger that efficiently keeps track of transactions between two participants in a distributed manner. Furthermore, such a ledger is publicly verifiable and secure against modification of chained blocks, the participants who verify the transaction and maintain the blockchain are called *miner*. Actually, the more miners that participate in a public blockchain system, the stronger security guarantee the blockchain system has. In this work, we utilize the Ethereum blockchain to construct CPVPA, since Ethereum is more expressive than other on-chain currencies and Ethereum is the one of most popular blockchain systems.

A simplified Ethereum blockchain is depicted in Fig. 2: Transactions are denoted by Tx; the hash value of current block is denoted by BlockHash; the hash value of the previous block is denoted by PrevBlockHash; Nonce denotes a solution of proof-of-work (PoW) puzzles detailed later; Time denotes the timestamp to indicate when the block is appended to the blockchain; MerkleRoot denotes the root value of a Merkle hash tree [38] formed by all transactions in current block. The ledger of Ethereum can be thought of as a state transition system, where there is a "state" consisting of the ownership status of all existing Ethers (which are the underlying value token) and a "state transition function" that takes a state and a transaction as input, and outputs a new state which is the result. When a new block is added into the chain, all transactions recorded in the block should be verified first, and then miners compute a valid nonce (denoted by Nonce in Fig. 2) such that the hash value of the block is less than or equal to a value provided by the Ethereum system. This process is a proof of work and is well known as "Mining". The first miner who finds the nonce broadcasts the block of transactions together with this nonce. Other participants can verify that the nonce is a valid solution, and hence add the new block to their blockchain. Once the block is added to the chain, all the corresponding state information has been updated. More technique details can be found in [26].

In Ethereum, the state is made up of objects called "account". Generally, Ethereum includes two categories of accounts: externally owned ones and contract ones. Externally owned accounts are controlled by private keys. Contract accounts are controlled by their contract code. Each smart contract corresponds to a contract account, after the contract is deployed and contained by the Ethereum blockchain, one can send message to the contract account (i.e., transferring Ethers from an account to the contract account) to trigger the execution of the smart contract, where the data included in the "data field" can be set as the input of the contract. For the transaction between two external

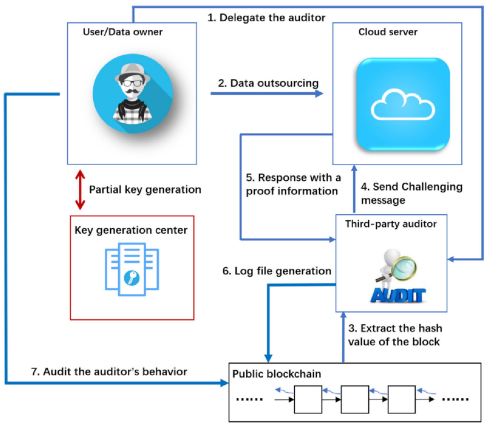


Fig. 3. The proposed CPVPA.

owned accounts, once the transaction is recorded into the blockchain, the accounts' balances are updated. Note that the transactions between two external accounts in Ethereum also include a "data" field. The user who conducts the transaction (i.e., the payer shown in Fig. 2) can set the data field to be any binary data she/he chooses.

There are three fundamental properties in secure blockchain systems [39], [40], [41]:

1) φ-chain consistency. Blockchains of any two honest miners at any point in time during the mining execution can differ only in the last φ blocks.
2) (ι, φ)-chain quality. For an honest miner's blockchain, the fraction of blocks mined by honest miners in any sequence of φ or more successive blocks is at least ι. In other words, the probability that any φ successive blocks on the blockchain are generated by an adversarial miner whose hashrate is less than 51 percent of the network's mining hashrate can be negligible. In Ethereum, $\varphi \geq 12$.
3) Chain growth. The number of blocks that are appended to the blockchain during any given time interval is deterministic. In other words, the blockchain' height would steadily increase with respect to both short and long terms.

With the above fundamental properties as well as the inherent characteristics of PoW-based consensus algorithm, we derive two properties from the Ethereum blockchain and adopt them to construct CPVPA.

- Unpredicted hash value.[3] The hash value (denoted by BlockHash in Fig. 2) of each block on the blockchain only can be determined after a valid nonce is computed and verified by all miners. Once the block is appended to the blockchain, its hash value is deterministic and resistant to modification. This means that given a point in time $time$, if $time$ is a point in the past, the hash value of the latest block that has appeared since $time$ on the blockchain is publicly verifiable and can be extracted efficiently; if $time$ is a point in the future, the hash value is computationally unpredictable. We stress that *the fact that the hash value of blocks generated in the future cannot be predicted does not means that the hash value cannot be biased by an adversary who has infinite budget*. Since the adversary can incentivize a miner who first mine a block to throw the newly mined block away and continue to mine if the hash value of the block does not meet the adversary's requirement [42].
- Time-sensitive data state. For a transaction with a string Ti as its Data value, if a block including this transaction is accepted by a majority of miners and is chained to the blockchain, the string Ti is then time-stamped. It means that Ti was created earlier than the time that the block is appended to the blockchain. Thus Ti is time-sensitive. Furthermore, due to the property of chain growth, it is feasible to derive the time when the block was appended to the blockchain from the height of the block. Specifically, if a transaction with Ti is recorded into the blockchain, anyone can extract the corresponding block's height (which is denoted by t_1) to determine whether Ti is generated no later than $BlockTime_{t_1}$, where $BlockTime_{t_1}$ denotes the height-derived time. By doing so, an Ethereum blockchain provides an efficient and secure service for time-stamping a digital document but without introducing any trusted authority [33].

4 The Proposed CPVPA

4.1 Overview

We utilize the Ethereum blockchain as the underlying public blockchain. Fig. 3 shows the proposed CPVPA.

CPVPA consists of two phases. In the first phase, the auditor verifies the integrity of outsourced data. In the second phase, the user audits the auditor's behavior.

In the first phase, the verification period is determined by the user. For a point in time when the data integrity should be verified, TPA first extracts the hash values of φ successive blocks that are the latest ones confirmed on the Ethereum blockchain, where φ denotes the number of blocks deep used to confirm a transaction (in the Ethereum, $\varphi = 12$), and these hash values are denoted by $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$. Then TPA generates a challenging message on $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$, and sends it to the cloud server. Upon receiving the challenging message, the cloud server computes the corresponding proof. TPA checks the validity of the proof to verify the data integrity. If the checking fails, TPA informs the user that the data may be corrupted; Otherwise, TPA sets $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$ and the proof as a log entry, stores the entry to a log file, and creates a transaction that transfers 0 deposit from its account to the user's account,[4] wherein the data field is set to the hash value of the entry. In this paper, for the sake of simplicity, we assume the transaction is recorded to the block whose height is $t + \varphi + 1$ and PrevBlockHash $= Bl_{t+\varphi}$, as shown in Fig. 4.

In the second phase, the user audits the TPA's behavior in a much longer period compared with the verification period. We first show how is a single entry (without loss of generality, $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$ and the corresponding

3. Other PoW-based on-chain currencies, such as Bitcoin, also have this feature [17].

4. The Ethereum blockchain allows a payer to conduct a transaction, wherein the transaction value is 0.

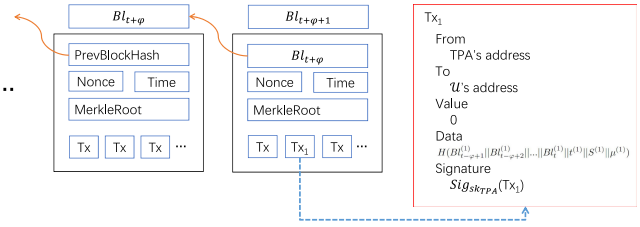


Fig. 4. The Transaction Tx_1 in PVPA.

proof) in the log file audited by the user. The user first determines the verification time that TPA should perform data integrity verification. Then she/he obtains $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$ from the Ethereum blockchain according to the agreed verification time, and extracts the hash value of the entry from the transaction. Next she/he regenerates the challenging message on $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$, and checks the validity of the corresponding proof by using the challenging message generated by herself/himself. If the checking passes, it means that TPA performs the verification correctly. Multiple entries can be audited simultaneously, and the auditing costs can be amortized over these entries, which we show in Section 6.

4.2 Construction of CPVPA

A user $\mathcal{U}$, a TPA, a cloud server $\mathcal{C}$, and a KGC are involved in CPVPA. The user $\mathcal{U}$ has an identity $ID_{\mathcal{U}}$.

Phase 1. TPA verifies the data integrity.

Setup. Based on the security parameter ℓ, KGC determines the system parameters as follows:

- Determine the bilinear pairing $e: G \times G \to G_T$, in which G and G_T are multiplicative groups with the same prime order p and g is a generator of G.
- Choose a random $\lambda \in Z_p$ as the master key and computes $P_M = g^{\lambda}$ as its public key.
- Choose five hash functions $H(\cdot)$, $H_1(\cdot)$, $H_2(\cdot)$, $H_3(\cdot)$, $H_4(\cdot)$, where $H(\cdot): \{0,1\}^* \to Z_p, H_1(\cdot) \sim H_4(\cdot): \{0,1\}^* \to G$ are cryptographic hash functions.
- Choose a pseudorandom permutation $\pi_{key}(\cdot)$ and a pseudorandom function $f_{key}(\cdot)$ [43].

The system parameters are $\{e, G, G_T, g, p, H(\cdot), H_1(\cdot) \sim H_5(\cdot), \pi_{key}(\cdot), f_{key}(\cdot)\}$.

The KGC generates the partial private key for $\mathcal{U}$ using $ID_{\mathcal{U}}$ as follows:

- Compute $Q_{\mathcal{U},0} = H_1(ID_{\mathcal{U}}, 0)$ and $Q_{\mathcal{U},1} = H_1(ID_{\mathcal{U}}, 1)$.
- Compute $D_{\mathcal{U},0} = Q_{\mathcal{U},0}^{\lambda}$ and $D_{\mathcal{U},1} = Q_{\mathcal{U},1}^{\lambda}$.

$\mathcal{U}$ chooses a random $x_{\mathcal{U}} \in Z_p^*$ and computes $pk_{\mathcal{U}} = g^{x_{\mathcal{U}}}$.

$\mathcal{U}$'s signing key is $ssk_{\mathcal{U}} = \{x_{\mathcal{U}}, D_{\mathcal{U},0}, D_{\mathcal{U},1}\}$, and the corresponding public key is $spk_{\mathcal{U}} = \{pk_{\mathcal{U}}, ID_{\mathcal{U}}\}$.

Store.

- $\mathcal{U}$ transfers his/her data M into n blocks: $M = \{m_i\}_{1 \le i \le n}$.
- $\mathcal{U}$ randomly chooses an element $name \in Z_p$ for file naming, randomly chooses a one-time number $\Delta \in Z_p$, and computes $\tau = H(name||n||\Delta||spk_{\mathcal{U}})$.
- $\mathcal{U}$ randomly chooses $r \in Z_p^*$, and computes $R = g^r$, $V = H_3(\Delta), W = H_4(\Delta)$.
- For each $i \in [1, n]$, $\mathcal{U}$ computes $T_i = H_2(i||\tau||R)$, and $S_i = (D_{\mathcal{U},0} \cdot V^{x_{\mathcal{U}}})^{m_i} \cdot (D_{\mathcal{U},1} \cdot W^{x_{\mathcal{U}}})^{H(i||\tau||R)} \cdot T_i^r$.
- $\mathcal{U}$ outsources $\tilde{M} = \{M, \{S_i\}_{i=1,2,\ldots,n}, R, \Delta\}$ to $\mathcal{C}$.
- After receiving $\tilde{M}$, $\mathcal{C}$ first computes $\tau = H(name||n||\Delta||spk_{\mathcal{U}})$, and then it verifies the correctness of the data by checking

$$e\left(\prod_{i=1}^{n} S_i, g\right) = e\left(\prod_{i=1}^{n} (Q_{\mathcal{U},0}^{m_i} \cdot Q_{\mathcal{U},1}^{\hbar_i}), P_M\right) \times e\left(\prod_{i=1}^{n} (V^{m_i} \cdot W^{\hbar_i}), pk_{\mathcal{U}}\right) e\left(\prod_{i=1}^{n} T_i, R\right),$$

where $\hbar_i = H(i||\tau||R)$. If the checking passes, $\mathcal{C}$ accepts $\tilde{M}$.

Audit. To check the integrity of outsourced data, TPA first generates a challenging message as follows:

- Extract $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$ from the Ethereum blockchain based on the current time, where φ denotes the number of blocks deep used to confirm a transaction, t denotes the height of the block that is latest confirmed at the current time.
- Set $(\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}, t)$ as the challenging message and send it to $\mathcal{C}$.

After receiving $(\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}, t)$ from TPA, $\mathcal{C}$ first checks the validity, i.e., whether $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t\}$ are the correct ones on the blockchain. If the checking fails, $\mathcal{C}$ rejects; Otherwise, $\mathcal{C}$ generates the proof information as follows:

- Compute $k_1 = h_1(Bl_{t-\varphi+1}||Bl_{t-\varphi+2}||\ldots||Bl_t)$ and $k_2 = h_2(Bl_{t-\varphi+1}||Bl_{t-\varphi+2}||\ldots||Bl_t)$, where $h_1(\cdot)$ is a secure hash function and maps $\{0,1\}^*$ to the key space of $\pi_{key}(\cdot)$, and $h_2(\cdot)$ is a secure hash function and maps $\{0,1\}^*$ to the key space of $f_{key}(\cdot)$.
- Compute $i_\xi = \pi_{k_1}(\xi), v_{i_\xi} = f_{k_2}(\xi), \quad \xi = 1, 2, \ldots, c$, where c is the number of data blocks that should be checked, and is determined by ℓ.
- Compute $S = \prod_{\xi=1}^{c} S_{i_\xi}^{v_{i_\xi}}, \mu = \sum_{\xi=1}^{c} v_{i_\xi} m_{i_\xi}$.
- Send $\{S, R, \mu, \Delta\}$ to TPA.

Upon receiving the proof information from $\mathcal{C}$, TPA verifies the data integrity as follows:

- Compute $\tau = H(name||n||\Delta||spk_{\mathcal{U}})$.
- Compute $k_1 = h_1(Bl_{t-\varphi+1}||Bl_{t-\varphi+2}||\ldots||Bl_t)$ and $k_2 = h_2(Bl_{t-\varphi+1}||Bl_{t-\varphi+2}||\ldots||Bl_t)$.
- Compute $i_\xi = \pi_{k_1}(\xi), v_{i_\xi} = f_{k_2}(\xi), \xi = 1, 2, \ldots, c$.
- Check whether

$$e(S, g) = e\left(Q_{\mathcal{U},0}^{\mu} \cdot \prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi}\hbar_{i_\xi}}, P_M\right) e\left(\prod_{\xi=1}^{c} T_{i_\xi}^{v_{i_\xi}}, R\right) \times e\left(V^{\mu} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi}\hbar_{i_\xi}}, pk_{\mathcal{U}}\right), \quad (1)$$

where $\hbar_{i_\xi} = H(i_\xi||\tau||R), T_{i_\xi} = H_2(i_\xi||\tau||R)$.

- If the Equation (1) holds, TPA outputs the verification result as Accept; Otherwise, TPA outputs the verification result as Reject.

Phase 2. The user checks the TPA's behavior.

LogGen. TPA generates a log file as follows:

TABLE 1
The Log File Λ in CPVPA

R			
$Bl_{t-\varphi+1}^{(1)}, Bl_{t-\varphi+2}^{(1)}, \ldots, Bl_t^{(1)}$	$t^{(1)}$	$S^{(1)}$	$\mu^{(1)}$
$Bl_{t-\varphi+1}^{(2)}, Bl_{t-\varphi+2}^{(2)}, \ldots, Bl_t^{(2)}$	$t^{(2)}$	$S^{(2)}$	$\mu^{(2)}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$
$Bl_{t-\varphi+1}^{(l)}, Bl_{t-\varphi+2}^{(l)}, \ldots, Bl_t^{(l)}$	$t^{(l)}$	$S^{(l)}$	$\mu^{(l)}$

- For each verification task, generate an entry as $\{Bl_{t-\varphi+1}, Bl_{t-\varphi+2}, \ldots, Bl_t, t, S, \mu\}$.
- Store the entry to a log file Λ in chronological order, as shown in Table 1.
- Compute the hash value (hereinafter, we take the first verification task as an example)

$$\theta_{t_1} = H(Bl_{t-\varphi+1}^{(1)} || Bl_{t-\varphi+2}^{(1)} || \ldots || Bl_t^{(1)} || t^{(1)} || S^{(1)} || \mu^{(1)}).$$

- Generate a transaction Tx_1 shown in Fig. 4, where the *data* field is set to θ_{t_1}, and upload it to the Ethereum blockchain. Ideally, Tx_1 is recorded in the block whose height is $t+\varphi+1$ and hash value is $Bl_{t+\varphi+1}^{(1)}$.

CheckLog. We first show that how to check the validity of a single entry (e.g., the first row) in Λ:

- $\mathcal{U}$ first acquires $t^{(1)}$ and $t^{(1)}+\varphi+1$, and derives the physical time when the verification is performed from $t^{(1)}$ and $t^{(1)}+\varphi+1$. If the time does not match the agreed one, $\mathcal{U}$ sets the checking result as $Reject$.
- $\mathcal{U}$ acquires $Bl_t^{(1)}$ from the Ethereum blockchain, and extracts θ_{t_1} from the block whose hash value is $Bl_{t+\varphi+1}^{(1)}$. If the extraction fails, $\mathcal{U}$ sets the checking result as $Reject$.
- $\mathcal{U}$ checks that whether θ_{t_1} matches the entry in the first row of Λ.
- $\mathcal{U}$ computes $\tau = H(name||n||\Delta||spk_{\mathcal{U}})$, $i_\xi^{(1)} = \pi_{k_1}(\xi)$, $v_{i_\xi}^{(1)} = f_{k_2}(\xi)$, where

$$k_1 = h_1\Big(Bl_{t-\varphi+1}^{(1)} || Bl_{t-\varphi+2}^{(1)} || \ldots || Bl_t^{(1)}\Big),$$
$$k_2 = h_2\Big(Bl_{t-\varphi+1}^{(1)} || Bl_{t-\varphi+2}^{(1)} || \ldots || Bl_t^{(1)}\Big).$$

- $\mathcal{U}$ checks

$$\begin{aligned} e(S^{(1)}, g) = {} & e\left(Q_{\mathcal{U},0}^{\mu^{(1)}} \cdot \prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi}^{(1)} \hbar_{i_\xi}^{(1)}}, P_M\right) \\ & \times e\left(V^{\mu^{(1)}} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi}^{(1)} \hbar_{i_\xi}^{(1)}}, pk_{\mathcal{U}}\right) \\ & \times e\left(\prod_{\xi=1}^{c} (T_{i_\xi}^{(1)})^{v_{i_\xi}^{(1)}}, R\right), \end{aligned} \tag{2}$$

where $\hbar_{i_\xi}^{(1)} = H(i_\xi^{(1)} || \tau || R)$, $T_{i_\xi}^{(1)} = H_2(i_\xi^{(1)} || \tau || R)$. If the checking fails, $\mathcal{U}$ sets the auditing result as $Reject$; Otherwise, the auditing result is $Accept$.

In practice, a user needs to audit multiple entries to ensure the correctness of the TPA's behaviour during an epoch. To reduce the auditing costs, multiple entries in Λ can be audited simultaneously. Specifically, the user picks a random b-element subset of the set $[1, l]$. Let $\{l_1, l_2, \ldots, l_b\}$ be the picked subset. The user then audits the TPA's behaviour by checking

$$\begin{aligned} e\left(\prod_{\eta=1}^{b} S^{(l_\eta)}, g\right) = {} & e\left(\prod_{\eta=1}^{b} Q_{\mathcal{U},0}^{\mu^{(l_\eta)}} \cdot \prod_{\eta=1}^{b}\prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi}^{(l_\eta)} \hbar_{i_\xi}^{(l_\eta)}}, P_M\right) \\ & \times e\left(\prod_{\eta=1}^{b} V^{\mu^{(l_\eta)}} \cdot \prod_{\eta=1}^{b}\prod_{\xi=1}^{c} W^{v_{i_\xi}^{(l_\eta)} \hbar_{i_\xi}^{(l_\eta)}}, pk_{\mathcal{U}}\right) \\ & \times e\left(\prod_{\eta=1}^{b}\prod_{\xi=1}^{c} \Big(T_{i_\xi}^{(l_\eta)}\Big)^{v_{i_\xi}^{(l_\eta)}}, R\right). \end{aligned}$$

If the check fails, $\mathcal{U}$ sets the auditing result as $Reject$; Otherwise, the auditing result is $Accept$.

Correctness. The correctness of CPVPA depends on the correctness of Equations (1) and (2), and we provide the correctness proof in the following:

$$\begin{aligned} e(S, g) &= e\left(\prod_{\xi=1}^{c} S_{i_\xi}^{v_{i_\xi}}, g\right) \\ &= e\left(\prod_{\xi=1}^{c} Q_{\mathcal{U},0}^{v_{i_\xi} m_{i_\xi}}, P_M\right) e\left(\prod_{\xi=1}^{c} V^{v_{i_\xi} m_{i_\xi}}, pk_{\mathcal{U}}\right) \\ &\quad \times e\left(\prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi} \hbar_{i_\xi}}, P_M\right) e\left(\prod_{\xi=1}^{c} W^{v_{i_\xi} \hbar_{i_\xi}}, pk_{\mathcal{U}}\right) \\ &\quad \times e\left(\prod_{\xi=1}^{c} T_{i_\xi}^{v_{i_\xi}}, R\right) \\ &= e\left(Q_{\mathcal{U},0}^{\mu} \cdot \prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi} \hbar_{i_\xi}}, P_M\right) \\ &\quad \times e\left(V^{\mu} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi} \hbar_{i_\xi}}, pk_{\mathcal{U}}\right) e\left(\prod_{\xi=1}^{c} T_{i_\xi}^{v_{i_\xi}}, R\right). \end{aligned}$$

4.3 Remark and Further Discussion

In CPVPA, we do not set the timestamp recorded in the block as the transaction time (i.e., when the verification is performed), since the timestamp recorded in the Ethereum block might suffer from up to 15 minutes error. In CPVPA, the user derives the transaction time from the height of the block which includes the transaction, due to the property of chain growth discussed in Section 3.6.

Now, we show how to compute the height-derived physical time when a block was generated. An Ethereum block averagely takes 15.85579752 seconds (this is counted based on the data released by Etherscan[5]) to be mined from the time of genesis block (which was generated on 2015-07-30) to the time of block with height 5,784,426 (which is the last block generated on 2018-06-15). We denote $time_{5784426}$ the height-derived physical time that transactions included in the block with height 5,784,426 were generated. $time_{5784426} = time_0 + 15.85579752 \times 5784426$ (seconds), where $time_0$ is the physical time that the genesis block of Ethereum was created (2015-07-30, 03:26:13 PM +UTC).

In CPVPA, the main goal of utilizing blockchain-based currencies is to resist procrastinating auditors. CPVPA also

5. https://etherscan.io

achieves an appealing feature, which we believe might be of independent interest. The auditor cannot pre-perform the scheduled verifications due to the unpredictability of block's hash value. In other words, if the auditor is required to verify the data integrity at a point in time, it has to perform the verification at the point in time. Our proposed mechanism used to resist procrastinating auditors is well compatible with most of existing public verification of data integrity schemes.

We stress that we construct CPVPA on PoW-based blockchain systems, such as Bitcoin and Ethereum. Theoretically, CPVPA can also be constructed on the blockchain systems that are based on other consensus algorithms (e.g., proof of stake (PoS) [44]), since the φ-chain consistency, (ι, φ)-chain quality, and chain growth are fundamental properties for secure blockchain systems. However, due to the differences on the block structure of the PoS-based blockchains, constructing CPVPA on different PoS-based blockchains requires different constructions. Furthermore, as discussed before, for a public blockchain system, the more participants in it, the stronger security guarantee it can provide. The number of participants in existing PoS-based blockchain systems, e.g., CARDANO [44], is much less than that in Ethereum. Therefore, if we construct CPVPA on these blockchains, the costs that the adversary breaks the security of CPVPA are reduced significantly.

5 Security Analysis

Lemma 1. *The signature $\sigma_i = \{S_i, R\}$ in CPVPA is existentially unforgeable under adaptively chosen-message attack.*

To prove this lemma, we first define two games corresponding to the type I adversary and type II adversary, respectively.

Game I (for adversary $\mathcal{A}_{\mathrm{I}}$):

Setup: a challenger $\wp$ runs the *Setup* algorithm and obtains the public parameters. $\wp$ sends the public parameters to $\mathcal{A}_{\mathrm{I}}$.

Query:

- Public-Key-Replacement queries $PKR(ID_{\mathcal{U}}, spk'_{\mathcal{U}})$: $\mathcal{A}_{\mathrm{I}}$ is able to choose a new public key $spk'_{\mathcal{U}} = \{Q_{\mathcal{U},0}, Q_{\mathcal{U},1}, pk^*_{\mathcal{U}}\}$ and sets $spk'_{\mathcal{U}}$ as the new public key of $\mathcal{U}$. $\wp$ will record this replacement.
- Sign queries $S(\Delta_i, m_i, ID_{\mathcal{U}}, spk_{\mathcal{U}})$: $\mathcal{A}_{\mathrm{I}}$ can request $\mathcal{U}$'s signature on a message m_i under a state information Δ_i. On receiving a query $S(\Delta_i, m_i, ID_{\mathcal{U}}, spk_{\mathcal{U}})$, $\wp$ generates the corresponding signature σ_i, and sends σ_i to $\mathcal{A}_{\mathrm{I}}$.

Forgery: For the $ID_{\mathcal{U}}$, $\mathcal{A}_{\mathrm{I}}$ outputs the corresponding public key $spk'_{\mathcal{U}}$, a messages m^*, a state information Δ^*, and a signature σ^*.

We say that $\mathcal{A}_{\mathrm{I}}$ wins *Game I* if and only if:

1) σ^* is a valid signature on m^* with state information Δ^* under $ID_{\mathcal{U}}$ and $spk'_{\mathcal{U}}$.
2) m^* is not submitted during the sign queries.

Game II (for adversary $\mathcal{A}_{\mathrm{II}}$):

Setup: a challenger $\wp$ runs the *Setup* algorithm and obtains the secret and public parameters. $\wp$ sends the public parameters and the KGC's master key to $\mathcal{A}_{\mathrm{II}}$.

Query:

- Sign queries $S(\Delta_i, m_i, ID_{\mathcal{U}}, spk_{\mathcal{U}})$: $\mathcal{A}_{\mathrm{II}}$ can request $\mathcal{U}$'s signature on a message m_i under a state information Δ_i. On receiving a query $S(\Delta_i, m_i, ID_{\mathcal{U}}, spk_{\mathcal{U}})$, $\wp$ generates the corresponding signature σ_i, and sends σ_i to $\mathcal{A}_{\mathrm{II}}$.

Forgery: For the $ID_{\mathcal{U}}$, $\mathcal{A}_{\mathrm{II}}$ outputs a messages m^*, a state information Δ^*, and a signature σ^*.

We say that $\mathcal{A}_{\mathrm{II}}$ wins *Game II* if and only if:

1) σ^* is a valid signature on m^* with state information Δ^* under $ID_{\mathcal{U}}$ and $spk_{\mathcal{U}}$.
2) m^* is not submitted during the sign queries.

Proof. We first prove that the advantage that $\mathcal{A}_{\mathrm{I}}$ wins *Game I* is negligible.

Let $\wp$ be a CDH attacker who receives a random instance (g, g^a, g^b) of the CDH problem in G and needs to compute g^{ab}. Here, we show that if $\mathcal{A}_{\mathrm{I}}$ is able to forge a signature with a probability ϵ, $\wp$ is able to solve the CDH problem by using $\mathcal{A}_{\mathrm{I}}$'s output with the same probability. Due to space limitation, we only show the proof sketch and omit some interaction details. In fact, this proof follows the proof of [34].

At the beginning of the game, $\wp$ sets $P_M = g^a$ as an instance of the CDH problem, and simulates $H(\cdot)$, $H_1(\cdot) \sim H_4(\cdot)$ as random oracles. In the game, $\wp$ sets $Q_{\mathcal{U},0} = g^{\alpha^*_{\mathcal{U},0}} \cdot g^{\alpha'^*_{\mathcal{U},0} b}$, $Q_{\mathcal{U},1} = g^{\alpha^*_{\mathcal{U},1}} \cdot g^{\alpha'^*_{\mathcal{U},1} b}$, $T = g^{\varsigma^*}$, $V = g^{\beta^*}$, and $W = g^{\gamma^*}$, where $\alpha^*_{\mathcal{U},0}, \alpha'^*_{\mathcal{U},0}, \alpha^*_{\mathcal{U},1}, \alpha'^*_{\mathcal{U},1}, \varsigma^*, \beta^*, \gamma^* \in Z_p^*$ are chosen randomly. For any signature query on any message m under any state information, $\wp$ responses with the corresponding signature as follows.

- $\wp$ sets $H(R) = -(m\alpha'^*_{\mathcal{U},0})/\alpha'^*_{\mathcal{U},1}$.
- $\wp$ randomly chooses $r \in Z_p$, computes $R = g^r$ and

$$S = g^{\alpha^*_{\mathcal{U}} m \cdot a} \cdot g^{(\beta^* m) x_{\mathcal{U}}} \cdot g^{-((m\alpha'^*_{\mathcal{U},0} \cdot \alpha^*_{\mathcal{U},1})/\alpha'^*_{\mathcal{U},1}) \cdot a} \times g^{-((\alpha'^*_{\mathcal{U},0} \gamma^* m)/(\alpha'^*_{\mathcal{U},1})) x_{\mathcal{U}}} \cdot g^{\varsigma^* r}.$$

- $\wp$ responses with (S, R).

Finally, $\mathcal{A}_{\mathrm{I}}$ outputs a tuple $\{m^*, \sigma^*, \Delta^*, pk^*_{\mathcal{U}}\}$, where $\sigma^* = \{S^*, R^*\}$ is a valid signature of m^* under Δ^* and $pk^*_{\mathcal{U}}$. Since σ^* is a valid signature, we have

$$\begin{aligned} e(S^*, g) = {} & e(Q^{m^*}_{\mathcal{U},0} \cdot Q^{H^*}_{\mathcal{U},1}, P_M) \\ & \times e((V^*)^{m^*} \cdot (W^*)^{H^*}, pk^*_{\mathcal{U}}) \\ & \times e(T^*, R^*), \end{aligned} \tag{3}$$

where $H^* = H(R^*)$, $T^* = H_2(R^*)$. Here, we discuss the security of a single signature, hence the form H^* is slightly different from the one in CPVPA. This difference should not reduce the security of S_i. Note that

$$\begin{aligned} e(S^*, g) = {} & e(g^{m^* \alpha^*_{\mathcal{U},0} a} \cdot g^{m^* \alpha'^*_{\mathcal{U},0} ab} \cdot g^{H^* \alpha^*_{\mathcal{U},1} a} \cdot g^{H^* \alpha'^*_{\mathcal{U},1} ab} \\ & \times (pk^*_{\mathcal{U}})^{m^* \beta^*} \cdot (pk^*_{\mathcal{U}})^{H^* \gamma^*} \cdot (R^*)^{\varsigma^*}, g). \end{aligned}$$

We have

$$g^{ab} = (S^* \cdot (g^{m^* \alpha^*_{\mathcal{U},0} a} \cdot g^{H^* \alpha^*_{\mathcal{U},1} a} \cdot (pk^*_{\mathcal{U}})^{m^* \beta^*} \times (pk^*_{\mathcal{U}})^{H^* \gamma^*} \cdot (R^*)^{\varsigma^*})^{-1})^{(m^* \alpha'^*_{\mathcal{U},0} + H^* \alpha'^*_{\mathcal{U},1})^{-1}}.$$

By this way, $\wp$ can solve the CDH problem.

For $\mathcal{A}_{\text{II}}$, let $\wp$ be a CDH attacker who have received a random instance (g, g^a, g^b) of the CDH problem and needs to compute g^{ab} by using $\mathcal{A}_{\text{II}}$'s output.

At the beginning of *Game II*, $\wp$ randomly selects λ, $x_{\mathcal{U}} \in Z_p^*$ and obtains the public parameters. Then $\wp$ sends λ and the public parameters to $\mathcal{A}_{\text{II}}$. In *Game II*, $\wp$ sets $T = g^{\beta^*}$, $V = g^{\gamma^*} \cdot g^{\gamma'^* a}$, $W = g^{\varsigma^*} \cdot g^{\varsigma'^* a}$, and $pk_{\mathcal{U}} = g^{x_{\mathcal{U}} b}$, where $\beta^*, \gamma^*, \gamma'^*, \varsigma^*, \varsigma'^* \in Z_p^*$ are chosen randomly. For any signature query on any message m under any state information, $\wp$ responses with the corresponding signature as follows.

- $\wp$ sets $H(R) = -(\gamma'^* m)/(\varsigma'^*)$.
- $\wp$ randomly choose $r \in Z_p$, computes $R = g^r$ and

$$S = H_1(\mathcal{U}, 0)^{\lambda} \cdot H_1(\mathcal{U}, 1)^{\lambda} \cdot g^{\gamma^* m x_{\mathcal{U}}} \times g^{-((\varsigma^* \gamma'^* m)/\varsigma'^*) x_{\mathcal{U}}} \cdot g^{\beta^* r}.$$

- $\wp$ outputs (S, R) as the response.

Finally, $\mathcal{A}_{\text{II}}$ outputs a tuple $\{m^*, \sigma^*, \Delta^*\}$, where $\sigma^* = \{S^*, R^*\}$ is a valid signature of m^* under Δ^*.

Since σ^* is a valid signature, it satisfies the Equation (3). By our setting, we also have

$$e(S^*, g) = e(Q_{\mathcal{U},0}^{\lambda m^*} \cdot Q_{\mathcal{U},1}^{\lambda H^*} \cdot g^{x_{\mathcal{U}} b m^* \gamma^*} \times g^{x_{\mathcal{U}} b m^* \gamma'^* a} \cdot g^{x_{\mathcal{U}} b H^* \varsigma^*} \times g^{x_{\mathcal{U}} b H^* \varsigma'^* a} \cdot (R^*)^{\beta^*}, g),$$

where $Q_{\mathcal{U},0} = H_1(ID_{\mathcal{U}}, 0)$, $Q_{\mathcal{U},1} = H_1(ID_{\mathcal{U}}, 1)$, and $H^* = H(R)$. $\wp$ can solve the CDH problem

$$g^{ab} = (S^* \cdot (Q_{\mathcal{U},0}^{\lambda m^*} \cdot Q_{\mathcal{U},1}^{\lambda H^*} \cdot (R^*)^{\beta^*} \times g^{(x_{\mathcal{U}} m^* \gamma^* + x_{\mathcal{U}} H^* \varsigma^*) b})^{-1})^{(x_{\mathcal{U}} m^* r'^* + x_{\mathcal{U}} H^* \varsigma'^*)^{-1}}.$$

This concludes the proof of this lemma. □

Claim 1. CPVPA achieves the authenticity defined by [14], i.e., if the cloud server passes the TPA's verification, it must well maintains the specified data.

Proof. To prove the authenticity of CPVPA, we define a sequence of games with interleaved analysis.

Game 0. This game is simply the challenge game between TPA and the cloud server defined in Section 4.2.

Game 1. This game is the same as *Game 0*, with the exception of one difference. The adversary is trained to be able to forge a part of the proof information in *Audit*. Since $\sigma_i = \{S_i, R\}$ in CPVPA is existential unforgeable, here, the challenger records each proof information generated by the adversary, and declares failure and aborts if

1) the proof information is a valid one;
2) μ' in the proof information is different from the expected μ.

Analysis. As *Game 1* defined, in the case that the challenger aborts, the proof information generated by the adversary is $\{S, R, \mu', \Delta\}$. due to the correctness of CPVPA, we have

$$e(S, g) = e\left(Q_{\mathcal{U},0}^{\mu} \cdot \prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi} \hbar_{i_\xi}}, P_M\right) \times e\left(V^{\mu} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi} \hbar_{i_\xi}}, pk_{\mathcal{U}}\right) \times e\left(\prod_{\xi=1}^{c} T_{i_\xi}^{v_{i_\xi}}, R\right), \quad (4)$$

and for the adversary's output, we have

$$e(S, g) = e\left(Q_{\mathcal{U},0}^{\mu'} \cdot \sum_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi} \hbar_{i_\xi}}, P_M\right) \times e\left(V^{\mu'} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi} \hbar_{i_\xi}}, pk_{\mathcal{U}}\right) \times e\left(\prod_{\xi=1}^{c} T_{i_\xi}^{v_{i_\xi}}, R\right). \quad (5)$$

Since $\mu \neq \mu'$, $\bar{\mu} = \mu - \mu' \neq 0$. We further have

$$Q_{\mathcal{U},0}^{\mu} \neq Q_{\mathcal{U},0}^{\mu'}, V^{\mu} \neq V^{\mu'}.$$

Furthermore, we also have

$$e(Q_{\mathcal{U},0}^{\mu}, P_M) e(V^{\mu}, pk_{\mathcal{U}}) = e(Q_{\mathcal{U},0}^{\mu'}, P_M) e(V^{\mu'}, pk_{\mathcal{U}}).$$

Rearranging terms yields

$$e(Q_{\mathcal{U},0}^{\mu\lambda} \cdot V^{\mu x_{\mathcal{U}}}, g) = e(Q_{\mathcal{U},0}^{\mu'\lambda} \cdot V^{\mu' x_{\mathcal{U}}}, P),$$

that is, $(Q_{\mathcal{U},0}^{\lambda} \cdot V^{x_{\mathcal{U}}})^{\mu} = (Q_{\mathcal{U},0}^{\lambda} \cdot V^{x_{\mathcal{U}}})^{\mu'}$. We set $\omega = Q_{\mathcal{U},0}^{\lambda} \cdot V^{x_{\mathcal{U}}}$, and for arbitrary ω, we can represent it as $\omega = (g^*)^{\zeta^*} \cdot (g'^*)^{\zeta'^*}$, where $\zeta^*, \zeta'^* \in Z_p$, $g^*, g'^* \in G$ are random elements. Similarly, there exists $\chi \in Z_p$ such that $g^* = (g'^*)^{\chi}$. Here, the CDH problem is that given g'^* and $g^* = (g'^*)^{\chi}$, computing χ. By our setting, the solution of CDH problem is $\chi = -(\zeta'^*/\zeta^*)$.

Game 2. This game is the same as *Game 1*, with the exception of one difference. The adversary is trained to be able to forge any part of proof information in *Audit*. That is, the challenger records each proof information generated by the adversary, and declares failure and aborts if

1) the proof information $\{S', R', \mu', \Delta\}$ is a valid one;
2) the proof information $\{S', R', \mu', \Delta\}$ is different from the expected $\{S, R, \mu, \Delta\}$.

Analysis. We consider the user as the challenger. At the beginning of the game, the challenger sets $P_M = g^s$ as an instance of the CDH problem. Then the challenger randomly chooses $\alpha_0, \alpha_0', \alpha_1, \alpha_1' \in Z_p$ and sets $H_i = -m_i \alpha_0' \alpha_1'^{-1}$, $Q_{\mathcal{U},0} = g^{\alpha_0} \cdot (g')^{\alpha_0'}$, and $Q_{\mathcal{U},1} = g^{\alpha_1} \cdot (g')^{\alpha_1'}$. For random values r and $x_{\mathcal{U}}$, the challenger computes (S_i, R_i)

$$R_i = g^{r_i}$$
$$S_i = (D_{\mathcal{U},0} \cdot V^{x_{\mathcal{U}}})^{m_i} \cdot (D_{\mathcal{U},1} \cdot W^{x_{\mathcal{U}}})^{\hbar_i} \cdot T_i^r = g^{(m_i \alpha_0 - m_i \alpha_0' \alpha_1 \alpha_1'^{-1}) s} \cdot V^{m_i x_{\mathcal{U}}} \times (W^{m_i \alpha_0' \alpha_1'^{-1} x_{\mathcal{U}}})^{-1} \cdot T_i^r$$

The challenger continues to interact with the adversary. When the challenger aborts, the received proof information is $\{S^{'}, R^{'}, \mu^{'}, \Delta\}$, while the expected one is $\{S, R, \mu, \Delta\}$. Due to the correctness of CPVPA, we have the Equation (3) and

$$e(S^{'}, g) = e\left(Q_{\mathcal{U},0}^{\mu^{'}} \cdot \prod_{\xi=1}^{c} Q_{\mathcal{U},1}^{v_{i_\xi}\hbar_{i_\xi}}, P_M\right) \times e\left(V^{\mu^{'}} \cdot \prod_{\xi=1}^{c} W^{v_{i_\xi}\hbar_{i_\xi}}, pk_{\mathcal{U}}\right) \cdot e\left(\sum_{\xi=1}^{c} (T_{i_\xi}^{'})^{v_{i_\xi}}, R^{'}\right).$$

By our setting, $(S^{'}, R^{'}) \neq (S, R)$. Clearly, $\mu \neq \mu^{'}$, and we define $\bar{\mu} = \mu^{'} - \mu$. Here, the CDH problem is that given $(g, g^{'}, g^{s})$ computing $(g^{'})^{s}$.

Now we have $e(S^{'}/S, P) = e(Q_{\mathcal{U},0}^{s\bar{\mu}} \cdot V^{x_{\mathcal{U}}\bar{\mu}} \cdot \prod_{\xi=1}^{c}((T_{i_\xi}^{'})^{r^{'}}/T_{i_\xi}^{r})^{v_{i_\xi}}, g)$. Recall that $Q_{\mathcal{U},0} = g^{\alpha_0} \cdot (g^{'})^{\alpha_0^{'}}$, we can further get $S^{'}/S = g^{s\bar{\mu}\alpha_0} \cdot (g^{'})^{s\bar{\mu}\alpha_0^{'}} \cdot V^{x_{\mathcal{U}}\bar{\mu}} \cdot \prod_{\xi=1}^{c}((T_{i_\xi}^{'})^{r^{'}}/T_{i_\xi}^{r})^{v_{i_\xi}}$. Finally, the CDH solution

$$(g^{'})^{s} = (S^{'} \cdot S^{-1} \cdot (g^{s\bar{\mu}\alpha_0})^{-1} \cdot (V^{x_{\mathcal{U}}\bar{\mu}})^{-1} \times \prod_{\xi=1}^{c}((T_{i_\xi}^{'})^{r^{'}} \cdot T_{i_\xi}^{r})^{v_{i_\xi}})^{(\bar{\mu}\alpha_0^{'})^{-1}}.$$

This concludes the proof. □

Lemma 2. *In a PoW-based public blockchain system, such as Bitcoin and Ethereum, an adversary cannot pre-determine the hash value of the block generated at a future time.*

Proof. We first introduce the preimage resistance of a cryptographic hash function.

A hash function is preimage resistant if given a uniform y it is infeasible for a PPT adversary to find a value x such that $H(x) = y$ [43].

With the preimage resistance of hash function, the proof of this lemma is straightforward.

Assuming the block of underlying blockchain has the form shown in Fig. 4, where $Bl_t = H(Bl_{t-1}||\mathsf{Nonce}||\mathsf{Time}||\mathsf{MerkleRoot})$, and $H(\cdot)$ is a secure hash function with preimage resistance. If the adversary can pre-determine Bl_t, the preimage resistance of $H(\cdot)$ is broken.

This concludes this lemma. □

Claim 2. CPVPA is able to thwart malicious auditors who can perform two attacks to break the security:

1) Forging an entry in the log file Λ to pass the user's audit;
2) Sampling bias challenging messages to generate bias verification results.

Proof. For the first case, we will discuss it in two aspects.

First, TPA forges an entry $\{Bl_{t-\varphi+1}, \ldots, Bl_t, S^{'}, R^{'}, \mu^{'}\}$ that can pass the user's auditing. However, due to the authenticity of CPVPA, it is computationally infeasible to generate the entry. Specifically, since we have proved that if the data loss occurs, the cloud server can pass the auditor's verification with negligible probability. If TPA is able to forge the entry that passes the user's auditing, the cloud server is able to break the authenticity of CPVPA by following the TPA's method.

Second (for the procrastinating auditor), at the time $BlockTime_t + time_1, time_1 > 0$, TPA forges an entry to convince the user that the entry is generated at the time $BlockTime_t$. In this attack, TPA needs to generate a transaction and record it to a block that has been chained in the Ethereum blockchain. However, due to the security of Ethereum, this attack is computationally infeasible.

For the second case, we will analyze it in three aspects.

First, the hash values of blocks used to compute the challenging messages cannot be pre-determined and specified by any entity, due to *Lemma 2*.

Second, the blocks used to compute the challenging messages would not be generated by the adversary, due to (ι, φ)-chain quality of the Ethereum blockchain [39], [45].

Third, the only attack the malicious auditor can perform is to bias the hash value of the blocks by incentivizing the miners who mine a new block to throw the newly mined block away and continue to mine if the hash value of the block does not meet the adversary's requirement. Here, the adversary's requirement to the biased hash value is that the indexes of challenged blocks that generated on the hash value of the blocks exclude the corrupted ones, i.e., for $\xi = 1, 2, \ldots, c$, $i_\xi = \pi_{h_1(Bl_{t-\varphi+1}||Bl_{t-\varphi+2}||\cdots||Bl_t)}(\xi)$ would not include the indexes of corrupted blocks. Note that the adversary can bias $\{i_\xi\}(\xi = \{1, 2, \ldots, c\})$ only by biasing Bl_t. Now, we compute the probability that the adversary successfully biases Bl_t. The adversary model and game model follow the ones proposed by Pierrot et al. [42].

For illustration, we assume an adversary $\mathcal{A}$ who aims at biasing Bl_t to break the security of CPVPA. In our setting, the indexes of corrupted data blocks form a set ε, $\mathcal{A}$ wins whenever anyone of indexes of challenged blocks generated by Bl_t do not fall in ε. Let $\Upsilon : \varepsilon \rightarrow \{0, 1\}$ be the characteristic function that predicates whether a given value meets $\mathcal{A}$'s requirement.

We denote by P the probability for a extracted hash value of a block Bl to satisfy $\Upsilon(Bl) = 1$. We stress that P *is essentially a probability that the corrupted data set can pass the auditor's verification in CPVPA.* As evaluated by [9], [16], [31], if ρ fraction of data is corrupted, then randomly (uniformly) sampling c blocks would reach the detection probability $P_{detec} = 1 - (1 - \rho)^c$. Therefore,

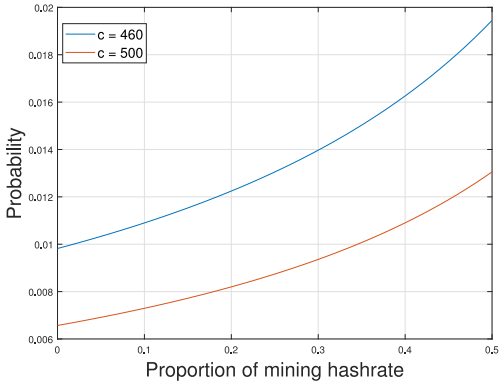
$$P = 1 - P_{detec} = (1 - \rho)^c. \tag{6}$$

$\mathcal{A}$'s strategy here is straightforward: he will focus all his computational power on mining the next block and incentivize the miners to throw the newly mined blocks away and continue to mine if $\Upsilon(Bl) = 0$, where the hash value of the newly mined block is denoted by Bl.

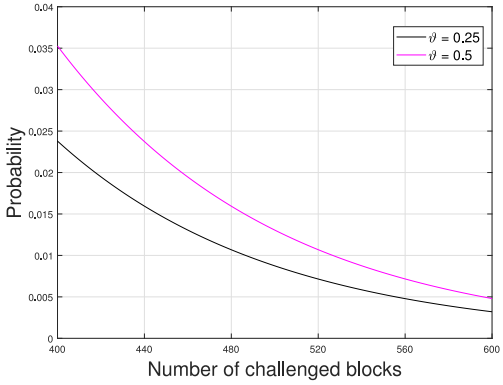
We denote by $P_{\mathcal{A}}$ the probability that $\mathcal{A}$ wins. According to [42], we know that

$$P_{\mathcal{A}} = \frac{P}{1 - \vartheta(1 - P)}, \tag{7}$$

where ϑ denotes the proportion of $\mathcal{A}$'s mining hashrate. In our security analysis, we assume $\vartheta < 51\%$, otherwise, the security of underlying blockchain would be broken.



(a) Probability that the adversary wins w.r.t. mining hashrate



(b) Probability that the adversary wins w.r.t. number of challenging blocks

Fig. 5. Probability that the adversary wins.

According to Equations (6) and (7), we have

$$P_{\mathcal{A}} = \frac{(1-\rho)^c}{1-\vartheta(1-(1-\rho)^c)}. \tag{8}$$

We show the probability that $\mathcal{A}$ wins in Fig. 5, where $\rho = 1\%$. For instance, when $\vartheta = 1/4$, $\rho = 1\%$, $c = 460$, the probability that $\mathcal{A}$ wins is 0.01305; When ϑ is set to $1/2$ and others parameters remain the same, $P_{\mathcal{A}}$ is increased from 0.01305 to 0.01983. Note that $\vartheta = 1/2$ denotes that the adversary is very powerful, but the probability that $\mathcal{A}$ wins is still very small.

The above analysis demonstrates that CPVPA is able to thwart malicious auditors.

This concludes the proof of this claim. □

In CPVPA, it is easy to prove that the security holds under the cases that the user colludes with TPA to circumvent the cloud server, and the user colludes with the server to circumvent TPA. Here we would not elaborate on it in details.

6 Performance Evaluation

We evaluate the performance of the proposed scheme in terms of communication overhead and computation overhead. All experiments are conducted on a computer with the Window 10 system, an Intel Core 2 i5 CPU, and 8 GB DDR 3 of RAM. The code is implemented by utilizing C language and MIRACL Library. The security level is chosen to be 80 bits for the evaluation, which means that the pairing is based on the MNT curve,[6] its base field size is 159 bits and its embedding degree is 6.

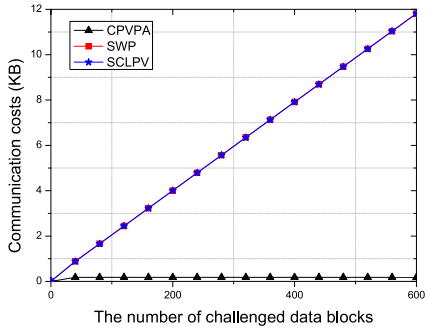


Fig. 6. The communication overhead.

TABLE 2
Notation of Operations

Symbol	Operation
$Hash_G$	mapping a value into G
Exp_G	exponent operation in G
Mul_G	group operation in G
Mul_{Z_p}	multiplication in Z_p
$Pair_{G_T}$	computing $e(g,q)$ where $g,q \in G$
C_f	computing a PRF $f(\cdot)$
Add_{Z_p}	addition in Z_p
$Hash_{Z_p}$	hash a value into Z_p

In [14], there is a discussion about the difference on the sector number. For the sake of brevity, we only provide the atomic operation analysis for the case that the sector number is 1 in the following. Furthermore, we assume that the total number of data blocks is less than 10,000.

6.1 Communication Overhead

We first show the communication costs between TPA and the cloud server in Fig. 6, and also give a comparison with SWP [14]. In CPVPA, the communication costs between TPA and the cloud server are constant, while those in SWP is proportional to the number of challenged data blocks.

6.2 Computation Overhead

We specify the computational costs with respect to cryptographic operations in Table 2.

We show the computation costs on the server side of CPVPA, SWP [14], SCLPV [18], and CLPA [24] in Table 3, where c denotes the total number of challenged data blocks. Since the challenging messages in CPVPA are computed on the hash value of the Ethereum blockchain, this requires additional computation costs. However, these additional costs ensure the (low) constant communication costs between the server and TPA, and also protects CPVPA from malicious or/and procrastinating auditors.

6. In reality, the MNT curve is utilized to construct Type-3 pairings, i.e., $e: G_2 \times G_1 \rightarrow G_T$. Because Type-3 pairings are more efficient than Type-1 pairings (i.e., $G \times G \rightarrow G_T$) in the same security level, in this section, we use the Type-3 pairing to analyze the performance. This would not affect the feasibility and security of CPVPA.

TABLE 3
Computation Costs on the Server Side

	Computation costs on the server
SWP [14]	$c \cdot Exp_G + c \cdot Mul_G + c \cdot Mul_{Z_p} + c \cdot Add_{Z_p}$
CLPA [24]	$c \cdot Exp_G + c \cdot Mul_G + c \cdot Mul_{Z_p} + c \cdot Add_{Z_p}$
SCLPV [18]	$2c \cdot Exp_G + 2c \cdot Mul_G + c \cdot Mul_{Z_p} + c \cdot Add_{Z_p}$
CPVPA	$c \cdot Exp_G + c \cdot Mul_G + c \cdot Mul_{Z_p} + c \cdot Add_{Z_p} + 2c \cdot C_f$

TABLE 4
Computation Costs on the TPA Side

	Computation costs on TPA
SWP [14]	$2 \cdot Pair_{G_T} + (c+1) \cdot Exp_G + c \cdot Mul_G + c \cdot Hash_G$
CLPA [24]	$2 \cdot Pair_{G_T} + (c+3) \cdot Exp_G + (c+3) \cdot Mul_G + (c+1) \cdot Hash_G + 2 \cdot Hash_{Z_p}$
SCLPV [18]	$4 \cdot Pair_{G_T} + (2c+4) \cdot Exp_G + (2c+2) \cdot Mul_G + 5 \cdot Hash_G + 2c \cdot Hash_{Z_p} + 2c \cdot Mul_{Z_p}$
CPVPA	$4 \cdot Pair_{G_T} + (3c+2) \cdot Exp_G + 3c \cdot Mul_G + (c+4) \cdot Hash_G + (2c+3) \cdot Hash_{Z_p} + 2c \cdot Mul_{Z_p} + 2c \cdot C_f$

We show a comparison of computation costs on the auditor side in Table 4, and show the computation delay on the TPA of CPVPA in Fig. 7.

Compared with existing schemes, CPVPA requires the user to audit the TPA's behavior at the end of each epoch. The costs of user's auditing are the same as the TPA's verification costs, when the user checks a single entry. However, when the user checks multiple entries simultaneously, the auditing costs can be amortized over these entries and reduced significantly. We show the user's auditing delay in Fig. 8.

According to the analysis, we can see that the user is able to audit the TPA's behavior within 1 minute in the case of 50 entries. Furthermore, as discussed before, the period that the user audits the TPA's behavior is much longer than the one that TPA verifies the data integrity, and these additional costs ensure that CPVPA can resist procrastinating auditors. Therefore, these computational costs on the user can be accepted in practice.

From the system perspective, CPVPA does not require the user, the auditor, and the cloud server to be a full node of Ethereum network, since there is a light client protocol of Ethereum.[7] It allows a user to conduct Ethereum transactions without maintaining the Ethereum blockchain. The block information of the Ethereum blockchain is released by multiple sites and systems in real time, such as Etherscan[8] and Etherchain.[9] This allows the user, the auditor, and the cloud server to securely extract the block information from the Ethereum blockchain. Therefore, the user, the auditor, and the cloud server in CPVPA would not incur a heavy burden in terms of communication and computation costs.

7 Related Work

To ensure the integrity of data stored on a remote and untrusted server, Juels et al. [30] presented the "proofs of retrievability" (POR) technique. However, in [30], public

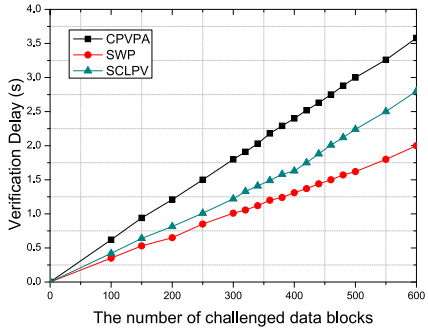


Fig. 7. The verification delay on the TPA side.

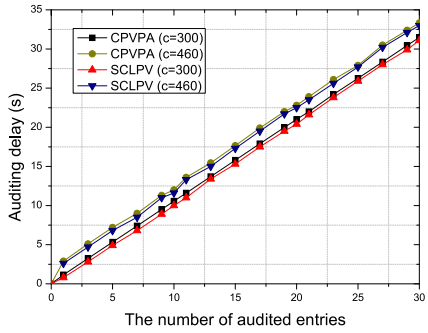


Fig. 8. The auditing delay on the user side.

verification is not considered, and hence the data owner needs to verify the data integrity periodically. This requires the data owner to keep online for verification. As such, the data owner has to bear heavy communication and verification burden to retrieve and use the data. In the mean time, Ateniese et al. [16] presented the "provable data possession" (PDP) technique, where the public verification was first considered: the data owner is able to employ a third-party auditor to check the data integrity on behalf of her/him. Subsequently, Shacham et al. [14] presented the first compact POR scheme with full proofs of security against arbitrary adversaries. Inspired by the Shacham et al.'s work, researchers have presented several public verification schemes with different features [9], [13], [15], [32], [46]. These schemes are constructed on the homomorphic signature technique [29].

Privacy preserving public verification has also attracted attentions in the recent literature. A privacy-preserving public verification scheme enables the auditor to verify the integrity of oursourced data without learning the content of the data. Most of existing privacy-preserving schemes, such as [9], [15], [31], [32], require the cloud server to utilize a random mask to blind the proof information such that the auditor can check the validity of the proof information without extracting the data content. For our scheme, to protect users' data against the auditor, we encrypt the data before generating the tags. Since in CPVPA, we consider that the auditor may collude with the cloud server, the cloud server would straightforward send the data to the auditor to violate the data privacy of users.

The above schemes are built on certificate-based cryptography and thereby are confronted with the certificate management problem. To avoid managing certificates in a public verification scheme, some schemes [10], [47] constructed on the identity-based signature schemes were

7. https://github.com/ethereum/mist/releases
8. https://etherscan.io/blocks
9. https://www.etherchain.org/

proposed. However, these schemes are subject to an inherent drawback: the key escrow problem [28]. Wang et al. [23] proposed the first certificateless public verification scheme. Then some certificateless public verification schemes with enhanced security were proposed [18], [24]. These schemes are constructed on the homomorphic certificateless signature schemes. Therefore, the auditor in these schemes does not need to manage the users' certificates without confronting with the key escrow problem.

Furthermore, in the existing schemes, the auditor is assumed to be honest and reliable. However, this is a very strong assumption, due to the fact that compromising auditors could be feasible in reality. Recently, Armknecht et al. [17] proposed the first PoR scheme that thwarts malicious auditors, and Zhang et al. [18] proposed the first public verification scheme with resistance against malicious auditors. These schemes cannot resist procrastinating auditors who may not perform the data integrity verification on schedule. A procrastinating auditor can deviate from the primary objective of public verification that detect the data corruption as soon as possible. It is worth clarifying that resistance against procrastinating auditors is vitally important for public verification schemes in practice. More detailed survey on public verification of data integrity can be found in [48].

8 Conclusion and Future Work

In this paper, we have proposed a certificateless public verification scheme against the procrastinating auditor, namely CPVPA. CPVPA utilizes on-chain currencies, where each verification performed by the auditor is integrated into a transaction on the blockchain of on-chain currencies. Furthermore, CPVPA is free from the certificate management problem. The security analysis demonstrates that CPVPA provides the strongest security guarantee compared with existing schemes. We have also provided a comprehensive performance evaluation, which demonstrates that CPVPA has a constant communication overhead and is efficient in terms of computation costs.

For the future work, we will investigate how to construct CPVPA on other blockchain systems. Since the main drawback of proofs of work is the energy consumption, constructing CPVPA on other blockchain systems (e.g., proofs-of-stake-based blockchain systems) can save energy. However, it requires an elaborated design to achieve the same security guarantee while ensuring the high efficiency. This remains an open research issue that should be further explored. We will also investigate how to use blockchains to improve cloud storage services in terms of security, performance, and functionality. As the outsourced data processing (e.g., outsourced computation and searching over encrypted data) has also played an important role in current information age, we will explore the integration of blockchain into existing schemes, which should have a deep impact on outsourced data processing.

Acknowledgments

This work is supported by in part by the National Key R&D Program of China under Grant 2017YFB0802000, in part by the National Natural Science Foundation of China under Grants 61872060 and 61370203, and in part by the China Scholarship Council. We would like to thank the editor and anonymous reviewers for their valuable comments. The first author would like to thank the colleagues from BBCR lab, University of Waterloo, for their valuable discussions.

References

[1] J. Yu, K. Wang, D. Zeng, C. Zhu, and S. Guo, "Privacy-preserving data aggregation computing in cyber-physical social systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 1, 2018, Art. no. 8.

[2] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.

[3] J. Li, H. Ye, W. Wang, W. Lou, Y. T. Hou, J. Liu, and R. Lu, "Efficient and secure outsourcing of differentially private data publication," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2018, pp. 187–206.

[4] L. Zhong, Q. Wu, J. Xie, J. Li, and B. Qin, "A secure versatile light payment system based on blockchain," *Future Generation Comput. Syst.*, vol. 93, pp. 327–337, 2019.

[5] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 4, pp. 870–885, Apr. 2019.

[6] K. Yang, K. Zhang, X. Jia, M. A. Hasan, and X. Shen, "Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms," *Inf. Sci.*, vol. 387, pp. 116–131, 2017.

[7] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Trans. Cloud Comput.*, accepted 2016, to appear, pp. 1–15, doi: 10.1109/TCC.2016.2647718.

[8] H. Yang, X. Wang, C. Yang, X. Cong, and Y. Zhang, "Securing content-centric networks with content-based encryption," *J. Netw. Comput. Appl.*, vol. 128, pp. 21–32, 2019.

[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. 14th Eur. Conf. Res. Comput. Secur.*, 2009, pp. 355–370.

[10] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Inf. Sci.*, vol. 472, pp. 223–234, 2018.

[11] K. Wang, J. Yu, X. Liu, and S. Guo, "A pre-authentication approach to proxy re-encryption in big data context," *IEEE Trans. Big Data*, accepted 2017, to appear, pp. 1–11, doi. 10.1109/TBDATA.2017.2702176.

[12] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Providing task allocation and secure deduplication for mobile crowdsensing via fog computing," *IEEE Trans. Depend. Sec. Comput.*, accepted 2018, to appear, pp. 1–13, doi. 10.1109/TDSC.2018.2791432.

[13] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, "Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 3, pp. 676–688, Mar. 2017.

[14] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.

[15] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[16] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[17] F. Armknecht, J. Bohli, G. O. Karame, and W. Li, "Outsourcing proofs of retrievability," *IEEE Trans. Cloud Comput.*, accepted 2018, to appear, pp. 1–15, doi: 10.1109/TCC.2018.2865554.

[18] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Trans. Comput. Soc. Syst.*, vol. 2, no. 4, pp. 159–170, Dec. 2015.

[19] Y. Zhang, C. Xu, H. Li, and X. Liang, "Cryptographic public verification of data integrity for cloud storage systems," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 44–52, Sep./Oct. 2016.

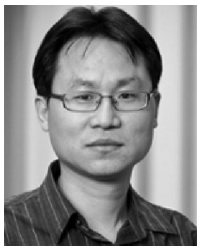
[20] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 754–764, Jun. 2010.

[21] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, accepted 2017, to appear, pp. 1–11, doi: 10.1109/TCC.2017.2769645.

[22] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4101–4112, Sep. 2018.
[23] B. Wang, B. Li, H. Li, and F. Li, "Certificateless public auditing for data integrity in the cloud," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2013, pp. 136–144.
[24] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Syst. J.*, vol. 12, no. 1, pp. 64–73, Mar. 2018.
[25] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[26] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.
[27] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surv. Tut.*, vol. 18, no. 3, pp. 2084–2123, Jul.–Sep. 2016.
[28] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2003, pp. 452–473.
[29] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2001, pp. 514–532.
[30] A. Juels and B. S. Kaliski, Jr, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 583–597.
[31] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[32] S. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Comput. Electr. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2014.
[33] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Proc. 10th Annu. Int. Cryptology Conf. Advances Cryptology*, 1990, pp. 437–455.
[34] L. Zhang, B. Qin, Q. Wu, and F. Zhang, "Efficient many-to-one authentication with certificateless aggregate signatures," *Comput. Netw.*, vol. 54, pp. 2482–2491, 2010.
[35] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 870–882, Apr. 2019, doi: 10.1109/TPDS.2018.2871449.
[36] Y. Zhang, X. Lin, and C. Xu, "Blockchain-based secure data provenance for cloud storage," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2018, pp. 3–19.
[37] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J. Liu, Y. Xiang, and R. Deng, "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, accepted 2018, to appear, pp. 1–15, doi: 10.1109/TPDS.2018.2881735.
[38] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Secur. Privacy*, 1980, pp. 122–134.
[39] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2015, pp. 281–310.
[40] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," *IACR Cryptology ePrint Archive*, vol. 2015, 2015, Art. no. 1019.
[41] R. Goyal and V. Goyal, "Overcoming cryptographic impossibility results using blockchains," in *Proc. Theory Cryptography Conf.*, 2017, pp. 529–561.
[42] C. Pierrot and B. Wesolowski, "Malleability of the blockchains entropy," *Cryptography Commun.*, vol. 10, no. 1, pp. 211–233, 2018.
[43] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton, FL, USA: CRC Press, 2014.
[44] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptology Conf.*, 2017, pp. 357–388.
[45] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas, "Bitcoin as a transaction ledger: A composable treatment," in *Proc. Annu. Int. Cryptology Conf.*, 2017, pp. 324–356.
[46] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Trans. Cloud Comput.*, accepted 2018, to appear, pp. 1–14, doi: 10.1109/TCC.2018.2851256.
[47] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.
[48] M. Sookhak, A. Gani, H. Talebian, A. Akhunzada, S. U. Khan, R. Buyya, and A. Y. Zomaya, "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–34, 2015.

Yuan Zhang (S'16) received the BSc degree in University of Electronic Science Technology of China (UESTC), Chengdu, China, in 2013. He is currently working toward the PhD degree in School of Computer Science and Engineering, University of Electronic Science Technology of China, and is a visiting PhD student in BBCR Lab, Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests are applied cryptography, data security, and blockchain technology. He is a student member of the IEEE.

Chunxiang Xu (M'06) received the BSc and MSc degrees in applied mathematics from Xidian University, Xi'an, China, in 1985 and 2004, respectively, and the PhD degree in cryptography from Xidian University, in 2004. She is currently a professor with the Center for Cyber Security, the School of Computer Science and Engineering, UESTC. Her research interests include information security, cloud computing security, and cryptography. She is a member of the IEEE.



Xiaodong Lin (M'09–SM'12–F'17) received the PhD degree in information engineering from the Beijing University of Posts and Telecommunications, China, and the PhD degree (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada. He is currently an associate professor with the School of Computer Science, University of Guelph, Canada. His research interests include computer and network security, privacy protection, applied cryptography, computer forensics, and software security. He is a fellow of the IEEE.

Xuemin Shen (M'97–SM'02–F'09) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, in 1990. He is currently a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the R.A. Fessenden Award in 2019 from IEEE, Canada, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015 and the Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and the Outstanding Performance Award 5 times from the University of Waterloo and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee chair/co-chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring, the chair for the IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking. He is the editor-in-chief of the *IEEE Internet of Things Journal* and the vice president on Publications of the IEEE Communications Society. He is a fellow of the IEEE.