IPANM: Incentive Public Auditing Scheme for Non-Manager Groups in Clouds

Longxia Huang, Junlong Zhou, Gongxuan Zhang, Senior Member, IEEE, Jin Sun, Tongquan Wei, Senior Member, IEEE, Shui Yu, Senior Member, IEEE, and Shiyan Hu, Senior Member, IEEE

Abstract—Cloud storage services give users a great facility in data management such as data collection, storage and sharing, but also bring some potential security hazards. An utmost importance is how to ensure the integrity of data files stored in the cloud, particularly for user groups without trusted managers. Existing literature focuses on integrity checking for groups with managers who have lots of permissions. To overcome the shortage of public auditing for non-manager user groups in clouds, we develop a novel framework IPANM that integrates \((t, n)\) threshold technology, blinding technology, and incentive mechanism to realize an incentive privacy-preserving public auditing scheme. In IPANM, the data integrity is guaranteed by our \((t, n)\) threshold signature based public auditing and the data privacy during public auditing is protected by the blinding technology. The generation of signatures can be accelerated by our blockchain-aided incentive mechanism that mobilizes the initiative of signers in the signature generation by rewarding the contributed signers. We formally prove the security of our IPANM and conduct numerical analysis and evaluation study to validate its high efficiency. The experimental results demonstrate that IPANM has lower overheads of storage, communication, and computation as compared to the state-of-the-art technique IAID-PDP and NPP.

Index Terms—incentive, threshold signature, public auditing, blockchain, cloud computing.

1 INTRODUCTION

With the help of crowd sensing [1], the users of smart end-devices in the community are able to collect data about the surrounding environment such as noise levels and mapping. These users have the same power in providing a safe and comfortable living environment for the community. After data collection, the users need to share the data with other members by cloud services. To ensure the correctness of the shared data, integrity check is needed. Due to its massive data storage and processing capacity, cloud computing has become a ubiquitous computing service [2] and it can be used in crowd sensing to store the messages collected by users. As we know, the users that use the cloud service are in general resource-constrained [3] and they participate in the generation of information together with other devices in the same group to provide high-quality and high-coverage measurements of crowd sensing [1]. In such a scenario, the presence of group managers prevents users from participating in data sharing with equal power. Thus, the power centralization problem of managers needs to be solved. In addition, if group users are assigned equal power, the low enthusiasm issue of users in crowd sensing cannot be ignored. Overall, for the above-mentioned scenario we need to ensure the integrity of stored data, solve the problem of power centralization, and improve users’ enthusiasm.

Cloud storage service allows the devices frequently resort to the cloud for outsourced data storage and sharing, which would bring a series of security challenges inevitably [4]. Among all the challenges, the primary concern is the integrity of outsourced files since the data stored in the cloud may be spoiled by hardware failures, manual operation errors, or external attack [5]. What’s worse, the cloud may conceal the fact that the data has been damaged to maintain its reputation [6]. Beside using hardware to prevent malicious behavior, public auditing has been a widely used technique to ensure the integrity of outsourced data due to its efficiency in verification. Different from the traditional integrity checking scheme, there is no need to download all file message for verification in public auditing since public auditing runs a challenge-and-response protocol that can realize sampling inspection with a high detection rate. What’s more, in public auditing the verification of integrity is run by a third party, which greatly reduces the workload of group users. However, an inherent problem of public auditing is that the third party attempts to acquire the data privacy during this process [7], [8]. Solving this problem necessitates privacy protection in designing a public auditing scheme for data integrity verification [9], [10].

In addition to utilizing homomorphic authentication [11] in verification to protect the challenged data, we introduce blinding technology in verification to protect data privacy since it supports confidentiality [12]. Note that traditional method only masks the message with random numbers which still may lead to the leakage of sensitive data [13]. In
this paper, the messages got by the auditor are further blind-
ed by blinding factors. Since blinding technology supports confiden-
tiality, the auditor cannot get the information of the original messages without knowing the blinding factors. It’s worth noting that blinding operation does not affect the correctness of validation with the aid of bilinear pair technology.

Apparently, the manager in a group is assigned too much power and may abuse the power to frame an innocent user or harbor a malicious user [14], which will frustrate the correctness of validation with the aid of bilinear pair technology.

Data privacy during public auditing is protected using blinding technology.

2) We deploy a novel incentive mechanism in IPANM, which encourages the users in the group to sign the message with rewards. To realize the incentive mechanism, blockchain is used to record the information of signers and determine the users who should be awarded.

3) We verify the effectiveness of the IPANM through theoretical analyses with respect to incentive, homomorphic authentication, correctness, public auditing, robustness, unforgeability, collision resistance, file privacy, and auditing efficiency.

4) We carry out extensive experiments to evaluate the performance of the IPANM in reducing the overhead of storage, computation, and communication. Simulation results demonstrate that the IPANM outperforms the state-of-the-art technique IAID-PDP.

The remainder of this paper is organized as follow. Section 2 introduces the related work and Section 3 reviews the preliminary knowledge used in our method IPANM. Section 4 presents the system model and our design objective, and Section 5 describes the IPANM in detail. To have a full investigation of the effectiveness of the IPANM, we provide the theoretical analysis in Section 6 and show the experimental results in Section 7. Concluding remarks and discussions on future work are given in Section 8 and Section 9 separately.

2 Related Work
Considerable research efforts have been devoted to the design of public auditing schemes for verifying the integrity of files stored in the cloud. Most of these schemes are designed based on Provable Data Possession (PDP) that allows both a data owner and a public verifier to operate integrity auditing without downloading the entire data from the cloud. For example, Wang et al. [13] proposed a PDP-based mechanism, Oruta, which allows public auditing for data shared in the cloud. Wang et al. [17] then pointed that Oruta is able to calculate the verification information for auditing the integrity of shared data, but it cannot resist the group members changing attack. Hence, the authors developed a new scheme called IAID-PDP [17] that not only resists the attack but also gives awards to the crime-reporter. However, due to the adoption of ring signature, the auditing time of Oruta and IAID-PDP linearly increase with the size of user group, thus cannot audit data shared among a large number of users.

To tackle this issue induced by ring signature, group signature is utilized in the proposed method Knox [18] to construct homomorphic authenticators, such that the auditing time using Knox is independent of the group size. However, Knox uses a group manager that is assigned a higher authority than users, thus does not support public auditing and is not suitable for groups of members with mutual equality. To solve this problem, NPP [14] uses $(t, n)$ threshold signature to eliminate the abuse of single-authority power by distributing the power to multi-manager. In this paper, we employ the $(t, n)$ threshold signature that not only supports public auditing but also allows $n$
equal members to take part in generating a valid signature. However, inefficient signature generation may occur as the equal members may be inactive in the generation of keys and signatures. In addition, the incentive mechanism is integrated into our \((t, n)\) threshold signature based public auditing scheme, in order to generate the signature as soon as possible. The incentive mechanism encourages users to take part in signature generation with rewards, which are granted using the technology of blockchain [15] with decentralization, tamper-resistant and traceability. What’s more, the effectiveness of blockchain-based incentive mechanism that users are encouraged to forward and receive packets with the incentive to increase their coins has been proved [19]. Inspired by the incentive mechanism in blockchain which reduces reward by half periodically, we propose a simple yet effective incentive mechanism that the numbers of awards are defined and gives more rewards to the users who generate signatures earlier in IPANM to solve the negative attitude caused by non-manager.

Public auditing schemes must take privacy protection into account due to its inherent problem that the public verifier is naturally curious about the sensitive information on the files and will attempt to obtain private information from the message received during the auditing. To this end, a lot of studies have focused on the design of privacy-preserving public auditing to support the privacy protection. To realize sensitive information hiding, Shen et al. [9] introduced a sanitizer to sanitize the sensitive data blocks and further transform the corresponding signatures for auditing. Xue et al. [5] utilized blockchain to directly replace the public verifier by employing the nonces in a blockchain for the automatic verification of challenge messages. Wang et al. [15] also discarded the public verifier and used blockchain to design the smart contract for public cloud storage auditing based on the non-interactive public PDP. For public auditing schemes with the public verifier, a privacy-aware protocol [20] is proposed to realize the privacy protection by guaranteeing that the third party auditor is not allowed to learn any knowledge about the data content stored in the cloud server during the efficient auditing process. Similarly, to realize the privacy protection of multisource IoT data, Tian et al. [2] developed a “zero-knowledge privacy” technique to achieve the privacy protection by ensuring that the verifier cannot obtain any information from publicly available data. However, public auditing using the above methods are not efficient due to their high communication and computation cost. In this paper, we utilize blinding technology to achieve privacy protection.

3 Preliminaries

Before showing the details of our proposed IPANM, we first introduce cryptographic primitives and their corresponding properties from existing literatures that we evaluate in IPANM.

3.1 Bilinear Map and Security Assumption

In IPANM, the integrity checking of files is based on the properties of bilinear map and the security proof is under Computational Diffie-Hellman assumption in random oracle model 2.

Definition 1. (Bilinear Map) [21]. Let \(G_1\) and \(G_2\) be two multiplicative cyclic groups of prime order \(q\), and \(g\) be a generator of \(G_1\). Then a bilinear map can be represented by \(e: G_1 \times G_1 \rightarrow G_2\), which has the following properties.

1. For any elements \(u, v \in G_1\) and any primes \(a, b \in \mathbb{Z}_q\), the equation \(e(a, b)^{uv} = e(u^{ab}, v^b)\) holds, where \(\mathbb{Z}_q\) is the set of prime numbers.
2. For any elements \(u_1, u_2, v \in G_1\), the equation \(e(u_1, v) = e(u_1, v)\cdot e(u_2, v)\) holds.
3. There must exist a method that can calculate the map \(e\) efficiently, which is non-trivial in the form of \(e(g, g) \neq 1\).

Given a certain elliptic curve, the bilinear maps defined above can be easily built. To understand our proposed scheme IPANM, the readers only need to know the properties of bilinear maps given in Definition 1. Thus, the details on how to construct bilinear maps from elliptic curves are omitted, which can be found in the literature [21]. In addition, the security analysis of our IPANM is based on the Computational Diffie-Hellman (CDH) assumption, which is described in Definition 2.

Definition 2. (CDH Assumption). The CDH assumption is based on CDH problem, which is defined as: given \(g, g^x, g^y \in G\) as input, where \(x, y \in \mathbb{Z}_q\) are unknown, outputs \(g^{xy}\). The CDH assumption indicates that for any probabilistic polynomial time algorithm \(\mathcal{A}\), the probability of using \(\mathcal{A}\) to solve the CDH problem is negligible, which is represented by

\[
\text{Adv}^{\text{CDH}}_{\mathcal{A}} = \Pr[\mathcal{A}(g, g^x, g^y) = g^{xy} : x, y \leftarrow \mathbb{Z}_q, g \leftarrow G].
\]

3.2 Homomorphic Verifiable Tags

Homomorphic verifiable tags [13], which enable a verifier to check the integrity of a file without downloading the entire file, are fundamental to develop public auditing mechanisms. In addition to unforgeability, a homomorphic verifiable signature scheme should satisfy the properties of blockless verification and non-malleability. Let \((pk, sk)\) represent a pair of the signer’s public and private key, and \(\sigma_1\) and \(\sigma_2\) represent the signatures on block \(m_1\) and \(m_2\), respectively, where \(m_1, m_2 \in \mathbb{Z}_p\). After defining these variables, the three properties are described as below.

- **Unforgeability:** only the users with the right private key can generate valid signatures.
- **Blockless verification:** without downloading the data, a verifier can verify the correctness of data in the cloud with a linear combination of blocks via a challenge-and-response protocol. Specifically, given \(\sigma_1\) and \(\sigma_2\), random values \(y_1, y_2 \in \mathbb{Z}_q\), and a linear combined block \(m' = y_1m_1 + m_2y_2\), the verifier is able to check the correctness of \(m'\) without knowing \(m_1\) and \(m_2\).
- **Non-malleability:** given the signatures, a user who does not have the valid secret keys cannot generate a valid signature on the combined blocks by using the given signatures. Specifically, given \(\sigma_1\) and \(\sigma_2\), random primes \(y_1, y_2 \in \mathbb{Z}_q\), and a linear combined block \(m' = y_1m_1 + m_2y_2\), the user cannot generate a valid \(\sigma'\) by combining \(\sigma_1\) and \(\sigma_2\) if he/she does not have \(sk\).


3.3 \((t, n)\) Threshold Signature

As pointed out earlier, \(t\) valid and honest users are required to generate a signature on a message. The generation of \((t, n)\) threshold signature is implemented at two steps. At the first step, each user gets his/her secret key sharings to produce a partial signature, and public key sharings to verify the correctness of the produced partial signature. At the second step, \(t\) valid partial signatures are collected and used to aggregate a final signature on the message. Note that the generated final signature should have the property of unforgeability, which indicates that the probability of an adversary \(A\) winning a game with the help from a challenger \(C\) is no larger than an arbitrarily small value \(\epsilon\) [22]. The game operates as follows.

1. \(A\) selects \(t-1\) members to corrupt and the other \(n-t+1\) members always keep honest during the whole game.
2. \(A\) is also allowed to query signing oracle and hash oracle to obtain the public key sharings of users and signatures of files during the game.
3. \(A\) submits hash query on the pair of identifier and block \((id, m)\) that has never been presented to the signing oracle, and generates the signature \((id', m', \sigma')\) of \((id', m')\) with the results received from \(C\).

The probability that \(A\) outputs a valid signature is formulated as

\[
\text{Succ}_{A}(k) = \Pr[\text{Verify}(id', m', \sigma') = 1] < \epsilon, \quad \text{where} \quad k\text{ is the security parameter.}
\]

3.4 Blinding Technology

In the traditional public auditing schemes, the auditor selects several random numbers \(\{v_1, v_2, ..., v_c\}\) and sends them to the cloud server for checking file integrity. Once getting these random numbers, the cloud server computes the proof message as \(p = \sum_{i \in [1, c]} v_i m_i\) with the stored file message \(m_i\) and returns it to the auditor. The challenge-and-response protocol ensures the cloud server cannot use the previous proof message to pass the auditing. However, the auditor may try to get the file message with the help of the protocol as the auditor is semi-trusted [2]. For example, in the first auditing, the auditor requires the cloud server to generate the proof message \(p_1 = \sum_{i \in [1, c]} v_i m_i\) with a random number set as \(\{v_1, v_2, ..., v_c\}\). Then, the auditor initiates the second validation with the random number set as \(\{v_1, v_2, ..., v'_1, ..., v_c\}\) to challenge the same file message. Finally, the auditor has two proof messages as \(p_1 = v_1 m_1 + ... + v_i m_i + ... + v_c m_c\) and \(p_2 = v_1 m_1 + ... + v'_1 m_i + ... + v_c m_c\). Using the two proof messages, the auditor can compute the file message as \(m_i = (p_2 - p_1)/(v'_i - v_i)\).

To be against the leakage of data privacy, the auditor can adopt encryption techniques to encrypt the messages. In this paper, we adopt a commonly accepted encryption technique, blinding technology, to protect the sensitive information in the files during the integrity verification. Blinding technology is extracted from blind signature, a form of digital signature in which the content of a message is disguised (blinded) before it is signed [23]. It is a commonly accepted encryption technique and has been widely used in applications where sender privacy is important, and exists for many public key signing protocols [23]. The philosophy behind blind signature is using a randomly selected blinding factor (typically in the form of random number) to encrypt the transmitted message [23]. Specifically, the blinding process is described as follows. In each public auditing, the verifier selects a fresh random number \(v\) and sends it to the cloud to prevent the cloud from passing the integrity checking using the previously transmitted messages [13]. In addition, to ensure the verifier cannot get the information in the obtained message \(vm\), the cloud further chooses a random factor \(b\) and blinds the message by

\[
m' = vm + b.
\]

Using the blinded message \(m'\), the verifier cannot get any information about the original message \(m\) but still can verify the integrity.

3.5 Ethereum

Ethereum is a global, open-source and decentralized application platform with Turing complete scripting language that allows users to create, deploy, and run smart contracts [24]. The smart contract is a tiny executable code that can be automatically executed without a trusted third party once the smart contract is deployed. The properties of smart contract such as decentralization and self-executability make it possible to guarantee the fairness of incentive in our scheme. In this paper, we design a simple yet effective incentive mechanism and use the Ethereum blockchain to realize it.

4 System Model and Design Objective

A secure cloud storage is consisted of public auditing scheme and signature scheme, where the former one is based on the second one. Therefore, in this section, we first introduce our proposed signature generation model that realizes the generation of keys and signatures, then present the public auditing model which is used to verify the integrity of files stored in the cloud, and finally describe the design objectives of our scheme.

4.1 Signature Generation Model

To realize a robust incentive public auditing scheme, we propose a model of signature generation that is carried out at three major steps: threshold key generation, signature sharing generation, and signature generation. In the first step, each user in the group is allowed to take part in the generation of keys. During this process, users who take part in and make contributions to key generation by generating right sharings, are selected as members in qualified user group \(QU\) to generate the secret key sharings for all users. Clearly, the qualified user group \(QU\) is a sub-group of the user group. In the second step, each user is able to generate the signature sharing on files after getting the secret key sharing. Any \(t\) signers (users in the group) producing the signature sharings constitute a sign group \(SG\). Note that a signer can belong to multiple signer groups. In the last step, the aggregator selects a signer group and uses their signature sharings to generate the final signature. The flowchart of signature generation is shown in Fig. 1.
As one can see from Fig. 1, the robustness of the system is gained from two aspects. First, not all the users (i.e., only the qualified users) are needed to participate in the generation of secret key and secret key sharings. Second, the system has multiple signer groups and all of them can be used to generate the signature. In addition to robustness, the system has the advantage of being incentive. That is, the server first employs blockchains to record the signatures uploaded by aggregators in a chronological order, then selects the earliest valid signature recorded as the final signature, and rewards the users who contribute to the selected signature in the end.

4.2 Public Auditing Model

The cloud data storage is composed of three entities: 1) a group of users, who have a large quantity of files to be stored in the cloud or have the right to download these files; 2) the cloud server, which has sufficient storage capacity and computation resources but may inadvertently modify and delete the stored data due to hardware failures or manual operation errors. To maintain its reputation, the cloud may conceal the fact that the data is damaged [7]. In this case, the cloud is not fully trusted [25]; 3) a verifier (e.g., a third party auditor providing expert data auditing services), who is capable of checking the integrity of shared data stored in the server through a challenge-and-response protocol, but is semi-trusted (honest-but-curious [2]) since he/she may launch passive attacks, such as data theft and analysis, rather than active attacks. There are members in the user group, where each user has the access to group files and is entitled to check the integrity of files stored in the cloud.

Public auditing can greatly mitigate users’ burden on checking the integrity of shared data, thus has been widely used in cloud computing and big data analysis. In general, public auditing operates as follows. At first, a user sends an auditing request to the public verifier, which immediately sends an auditing challenge message to the cloud server once receiving the request. Based on the data stored, the cloud server will reply an auditing proof of the possession of shared data to the public verifier after receiving the challenge. Finally, the public verifier validates the data by checking the correctness of the proof and then returns the result back to the user. The process of public auditing is described in Fig. 2.

4.3 Security Threat Model

The security of data stored in the cloud server may be broken by two types of threats. That is, the integrity threat from the cloud and the data privacy threat in public auditing. Both threats are described in detail as below.

- **Integrity Threat** [9]: Data integrity should be guaranteed to persuade users of using cloud storage. As data owners will no longer have access to their files if they delete their local files after uploading them to the cloud, the data stored in the cloud suffers from both internal and external threats. Internal threats include inadvertently modification and deletion by the cloud due to hardware failures or manual operation errors. Outside attackers also break the integrity of data. What’s worse, the cloud can conceal the fact that the data has been damaged, in order to maintain their interests and reputation. Therefore, to ensure the retrievability of the data, integrity verification for the data in the cloud is an important issue.

- **Data Privacy Threat** [2]: Data privacy threat means the data leakage occurring in the public auditing. This threat is induced by the employment of a semi-trusted verifier who may attempt to obtain the file information, including sensitive data, from the received verification message.

4.4 Design Goals of IPANM

To realize an incentive file storage system in the cloud, our proposed scheme IPANM needs to achieve the following security and performance guarantees.

1) Incentive: ensure that members who contribute to the signature stored in the cloud will get their awards fairly.

2) Correctness: ensure that valid public key sharings, valid signature sharings, and valid signatures generated by group users can pass the verification.
5 DESIGN OF OUR PROPOSED IPANM

This section first briefly describes the proposed IPANM, and then presents the details on how to construct IPANM.

5.1 Overview of IPANM

As an integration of \((t, n)\) threshold signature, blinding technology, and blockchain-aided incentive mechanism, the proposed IPANM can provide an incentive privacy-preserving public auditing for non-manager user groups in clouds. At a high level, the proposed IPANM is composed of three systems: signature generation system, verification system, and incentive system. The signature generation system is built to provide signatures on files for checking data integrity, which is realized by five steps: \textit{Set-up}, \textit{Threshold Key Generation}, \textit{Signature Sharing Generation}, \textit{Signature Aggregation}, and \textit{Signature Verification}. The verification system is deployed to verify the correctness of files stored in the cloud, which is realized by \textit{Public Auditing}. The incentive system is developed to encourage signers to take part in signing, which is realized by an \textit{Incentive Mechanism} that grants rewards to contributed signers. The high-level overview of IPANM is shown in Fig. 3, and the details of IPANM are given in the following subsections.

5.2 Construction of IPANM

5.2.1 Set-up

This initialization algorithm is run by the system. Let \(p\) denote a sufficient large prime and \(e : G_1 \times G_1 \rightarrow G_2\) denote a bilinear pairing, where \(G_1\) and \(G_2\) are two cyclic multiplicative groups with the same prime order \(p\). Let \(H\) and \(H_1\) denote two one-way hash functions, which are defined as \(H : \{0,1\}^* \rightarrow G_1\) and \(H_1 : \{0,1\}^* \rightarrow Z_p\), respectively. Let \(f\) denote a pseudo-random function and \(\pi\) denote a pseudo-random permutation, which are defined as \(f : Z_p \times \{1, 2, \cdots, n\} \rightarrow Z_p\) and \(\pi : Z_p \times \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}\), respectively. Two generators \(g\) and \(g_1\) are both selected from group \(G_1\), then the global public parameter set \(S_p = (p, G_1, G_2, e, g, g_1, H, H_1, f, \pi)\) can be constructed.

5.2.2 Threshold Key Generation

During this stage, the secret key sharing \(\alpha_i\) and public key sharing \(p_i\) of each user \(U_i\) are generated with the help of all group users. Note that only qualified users \((\text{i.e., } U_d\text{ in Fig. 1})\) in IPANM are allowed to generate the secret value sharing and public key sharing. Qualified users are the participants whose behavior is honest. It’s clear that not all the users are needed to participate in this process and any \(t\) qualified users form a qualified group. Thus IPANM can tolerate the faults made by users in the generation of the two sharings. The generation of the two sharings is described below.

- \textit{Generate secret key }\(\alpha\)\textit{ and secret key sharing }\(\alpha_i\). Since only qualified users are allowed to participate in the generation of secret key and secret key sharing, we need to identify the qualified users at first. The identification operates as follows. First, each user \(U_i\) selects a random polynomial \(f_i(x) = a_{i0} + a_{i1}x + a_{i2}x^2 + \cdots + a_{i(t-1)}x^{t-1}\), where \(a_{ik}\) \((1 \leq i \leq n, 0 \leq k \leq t-1)\) is coefficient. Subsequently, user \(U_i\) broadcasts variable \(C_{ik}\), which is equal to \(g^{a_{ik}}\), calculates the secret value sharing \(s_{ij}\) by \(s_{ij} = f_i(j)\), and sends \(s_{ij}\) to user \(U_j\) secretly where \(j \neq i\). After receiving \(s_{ij}\) from user \(U_i\), user \(U_j\) verifies the validity of \(s_{ij}\) by

\[
g^{a_{ij}} = \prod_{k=0}^{t-1} (C_{ik})^{j_k}, \quad 1 \leq i \leq n. \quad (2)
\]

When there is something wrong with the secret key sharing, it results in false of Eq. (2). Thus the user with the wrong sharing key should be warned and this bad behavior of user \(U_i\) will be broadcast to all users to abandon the message generated by the user. Clearly, user \(U_i\) cannot be regarded as a qualified one. Finally, a user is deemed as disqualified if the user has been reported by no less than \(t\) other users, and qualified otherwise. The group of qualified users is represented by \(QU\). Afterwards, the secret key \(\alpha\) and secret key sharing \(\alpha_i\) of user \(U_i\) can be derived by

\[
\alpha = \sum_{i \in QU} a_{i0} \quad (\text{mod } p), \quad (3)
\]

\[
\alpha_i = \sum_{j \in QU} s_{ji} \quad (\text{mod } p). \quad (4)
\]
5.2.4 Signature Sharing Aggregation

After getting enough signature sharings, an aggregator generates the signature of each file block \( \sigma_s \) and broadcasts it to all the other users who (e.g., user \( U_j \)) will check the validity of \( A_{ik} \) using

\[
g_{\omega} = \prod_{k=0}^{t-1} A_{ik} j^k. \tag{5}
\]

If Eq. (5) does not hold, the invalidity of \( A_{ik} \) will be recorded and \( t \) valid users will cooperate to reconstruct the right \( A_{ik} \) and re-broadcast it. Note that we introduce two kinds of blockchains in this paper. One is used to record invalid operations in the process of signature generation, and the other one is for incentive. Second, each user \( U_i \) in the user group broadcasts his/her public key sharing \( p_i \) to all the other users who (e.g., user \( U_j \)) will check the correctness of \( p_i \) using

\[
p_i = \prod_{j \in QU} \prod_{k=0}^{t-1} A_{jk} j^k. \tag{6}
\]

If Eq. (6) holds, the key sharing is valid and accepted. Otherwise, the user must send a wrong parameter \( p_i \) but the right parameter \( p_i \) will be calculated using the same equation Eq. (6). In addition, this bad behavior will be sent to the system to record. The correct public key sharing that passes the verification or the one re-generated by Eq. (6) is set as the final public key sharing \( p_i \). The public key \( pk \) is computed as \( pk = g^{p_i} \). In this step, the identity information of the users who generate the valid key sharings is recorded in the blockchain in chronological order. The content is used to realize incentive which encourages users to take part in and then improves the stability of the system.

5.2.3 Signature Sharing Generation

Each user is allowed to run this algorithm to generate signature sharing. During this stage, each user \( U_i \) generates his/her own signature sharing \( \sigma_{is} \) for each file block \( m_s \) with an identifier \( id_s \) using

\[
\sigma_{is} = (H(id_s) \cdot g_1^{m_s})^\alpha_i, \tag{7}
\]

where the block \( m_s \) is a part of files stored in the cloud, represented by \( M = \{m_1, m_2, \ldots, m_t\} \). The signature sharings of \( t \) qualified users can be used to generate the signature of each file block \( m_s \).

5.2.4 Signature Sharing Aggregation

After getting enough signature sharings, an aggregator generates the final signature with this algorithm. During this stage, an aggregator \( U_a \) who comes from the user group, aggregates the final signature of block \( m_s \) when he/she receives \( t \) valid signature sharings. Before aggregating the final signature, \( U_a \) needs to check the correctness of the received signature sharings using

\[
e(\sigma_{is}, g) = e(H(id_s) \cdot g_1^{m_s}, p_i). \tag{8}
\]

The aggregator records the identity information of signers who generate the valid signature sharings in chronological order. Once no less than \( t \) valid signature sharings are received, the aggregator uses the set \( SG \) to record the signers of these sharings and aggregates the final signature using

\[
\sigma_s = \prod_{i \in SG} \sigma_{is} \lambda_i = (H(id_s) \cdot g_1^{m_s})^{\alpha_i}, \tag{9}
\]

where \( \lambda_i = \prod_{(j \in SG, j \neq i)} \frac{1}{m_j} \) is a Lagrangian interpolation coefficient. Finally, the aggregator records the signature \( \sigma_s \) as the identity \{id\} \( i \in SG \) of signers in the blockchain.

5.2.5 Signature Verification

To ensure the validity of the signature, each user is allowed to run this verification algorithm. The correctness of final signature of file block \( m_s \) can be checked by a verifier using

\[
e(\sigma_s, g, d) = e(H(id_s) \cdot g_1^{m_s}, pk). \tag{10}
\]

However, each file block may have multiple signatures since all the users are entitled to select \( t \) valid sharings and generate the signature. To address this issue, this algorithm selects only the earliest valid signature recorded in the blockchain, and sends it with its file to the cloud in the format of \( \{(m_s, id_s, \sigma_s)\}_{s \in [1, t]} \).  

5.2.6 Public Auditing

The public auditing algorithm realizes the challenge-and-answer protocol operated by a verifier and the cloud server. When user \( U_i \) attempts to check the integrity of file \( M \), the user or a verifier (if the verifier is delegated by the user) will send a challenging set \( \text{chal} \) to the cloud. Subsequently, based on the data in set \( \text{chal} \) and the file \( M = \{m_1, m_2, \ldots, m_t\} \), block identifier \( id_1, id_2, \ldots, id_t \) and signature \( \{\sigma_1, \sigma_2, \ldots, \sigma_t\} \) stored in the server, the cloud generates a file possession proof \( Pro \) and sends it to the user or the verifier. Finally, the user or the verifier can verify the integrity of file \( M \) with the proof \( Pro \). The details of generating and verifying the proof are shown below.

1) Generate file possession proof \( Pro \). User \( U_i \) or a verifier sends challenge message \( \text{chal} = \{c, k_1, k_2\} \) to the cloud, where \( c \in [1, t] \) is the number of challenged blocks and \( k_1, k_2 \in Z_q \) are two random numbers. After receiving the challenge, the cloud first calculates \( t = \pi_{k_1}(j), v_i = f_{k_2}(j) \), where \( 1 \leq j \leq c \). The cloud then aggregates all the blocks’ tags \( \varphi = \Pi_{i \in T} \sigma_{iv} \in G_1 \). At last, the cloud picks a random number \( r \rightarrow Z_q^* \) calculates \( R = e(g_1, pk)\gamma, \gamma = H_{1}(R), \mu = (r+\gamma) \cdot \sum_{l \in T} v_{ml} \) (mod \( p \)), and sends audit proof \( Pro = \{(id_{l1}, l, \mu, \varphi, R)\} \) to the user or verifier.

2) Verify file possession proof \( Pro \). After getting the proof \( Pro \), the verifier first computes

\[
t = \pi_{k_1}(j), v_i = f_{k_2}(j), 1 \leq j \leq c
\]

and then checks the proof by

\[
R \cdot e(\varphi, g) = e((\Pi_{l \in T} H(id_{lv}))^{\gamma} \cdot g_1^{\mu}, pk). \tag{11}
\]

If the equation holds, the file \( M \) stored in the cloud is deemed as valid.
is realized by recording the identities of signers who contribute to the final signatures in the blockchain, where the records are unchangeable. The fairness is maintained by allowing each user to take part in the signature generation and always picking the earliest generated valid signature. The initiative of participating in the signature generation is mobilized by rewarding the signers who contribute to the final signatures.

### 6.1 Homomorphic Authentication

Homomorphic authentication is the basic tool to construct public auditing mechanism [18], and it meets the demands of blockless verifiability and non-malleability.

**Theorem 1.** The proposed scheme IPANM supports homomorphic authentication.

**Proof.** According to the properties of homomorphic authentication, the proposed scheme is required to support blockless verifiability and non-malleability [13]. We discuss blockless verification first. For two blocks \(m_1\) and \(m_2\), given the public key \(pk\), their identifiers \(id_1\) and \(id_2\), their signatures \(\sigma_1\) and \(\sigma_2\), and two random numbers \(y_1\) and \(y_2\), a verifier would have the ability to check the correctness of a block \(m' = y_1m_1 + y_2m_2\) by checking whether the following equation holds:

\[
e \left( \prod_{i=1}^{2} \sigma_i^{y_i}, g \right) = e \left( \prod_{i=1}^{2} (H(id_i)^y_i \cdot g_{\alpha}^{m_i}), pk \right). \tag{12}
\]

Note that this checking procedure does not require knowing in advance the value of blocks \(m_1\) and \(m_2\). Based on the properties of bilinear maps, the proof of Eq. (12) is described as follows:

\[
e \left( \prod_{i=1}^{2} \sigma_i^{y_i}, g \right) = e \left( \prod_{i=1}^{2} (H(id_i)^y_i \cdot g_{\alpha}^{m_i}), g \right) = e \left( \prod_{i=1}^{2} (H(id_i)^y_i), g_{\alpha} \right) = e \left( \prod_{i=1}^{2} (H(id_i)^{y_i} \cdot g_{\alpha}^{m_i}), pk \right). \tag{13}
\]

It is clear that IPANM supports blockless verification. We then prove that an attacker not having the private key cannot generate a valid signature \(\sigma'\) for a combined block \(m' = y_1m_1 + y_2m_2\) by combining \(\sigma_1\) and \(\sigma_2\) with \(y_1\) and \(y_2\). The difficulty of this step lies in the fact that hash function \(H\) is a one-way hash function. Let \(\theta = [H(id')g^{\alpha}]^\alpha\) denote the signature of block \(m'\). As we have \(\sigma_1^{y_1} \cdot \sigma_2^{y_2} = \prod_{i=1}^{2} (H(id_i)^{y_i} \cdot g_{\alpha}^{m_i})^\alpha\), if \(\theta\) can pass the verification, apparently we have \(\prod_{i=1}^{2} H(id_i)^{y_i} = H(id')\), which contradicts to the assumption that \(H\) is a one-way hash function.

Therefore, IPANM is a homomorphic authenticable scheme.

### 6.2 Correctness Analysis

We analyze the correctness [26] from three respects: valid public key sharings, valid signature sharings and signatures are able to pass the verification. According to [13], proving
6.3 Public Auditing

Public auditing ensures a verifier to check the integrity of files stored in the cloud with the received message.

Theorem 5. If the cloud does store the valid file $M$ and follow the phases of verification, a verifier is able to correctly check the integrity of shared data $M$ with the proof message

$$P_{ro} = \{\{id_i\}_{i\in \{\mu, \varphi, R\}}\}$$

Proof. With storing the valid file $M = \{m_1, m_2, \ldots, m_t\}$, block identifier $\{id_1, id_2, \ldots, id_t\}$ and block signature $\{\sigma_1, \sigma_2, \ldots, \sigma_t\}$, the cloud generates file possession proof $P_{ro}$. Based on the knowledge in Theorem 1 and given the proof message $P_{ro} = \{\{id_i\}_{i\in \{\mu, \varphi, R\}}\}$ calculated by the cloud, the verifier is able to check the integrity of files using Eq. (11). The reason is that if the proof message is computed by the corresponding files, the cloud can pass the verification which is proven as follows.

$$R \cdot e(\varphi, g) = R \cdot e(\varphi, g)$$

$$= R \cdot e(\Pi_{i\in I} \alpha_i, \gamma) \cdot g$$

$$= e(g_1, pk)^r \cdot e(\Pi_{i\in I} (H(id_i) \cdot g_1 \cdot \gamma), g)$$

$$= e(g_1, pk) \cdot e(\Pi_{i\in I} (H(id_i) \cdot g_1^{\gamma}), g)$$

$$= e(g_1, pk) \cdot e(\Pi_{i\in I} (H(id_i) \gamma), g)$$

$$= e(\Pi_{i\in I} (H(id_i) \gamma), g)$$

$$= e(\Pi_{i\in I} (H(id_i) \gamma), g)$$

6.4 Robustness

Robustness means that IPANM is flexible and tolerates the participation of some inactive parties in the generation of key sharings and signatures. Firstly, we show the flexibility of IPANM. Specifically, in key sharings generation, we define a set $QU$ which determines the value of the global secret key. On one hand, the size of $QU$ is set as required, which allows the participation of inactive users. On the other hand, as different set $QU$ generates different secret key, it's easy for an organization to sign different levels of files with different secret keys. Specifically, when two different offices $a$ and $b$ have two files $A$ and $B$ to generate signature, members in office $a$ construct a set $QU_a$ which corresponds to $sk_a$ and that of office $b$ is $sk_b$. The phases of signing file $A$ is independent of signing file $B$ and each user in the organization is able to take part in both tasks. Therefore, IPANM is flexible. Then, we prove that IPANM tolerates inactive parties in key sharings generation as follows.

Theorem 6. The signature scheme is robust when less than $t$ users are corrupted by an adversary [22].

Proof. A user corrupted by an adversary is under the control of this adversary. Instead of following the right phases described in Section 5, the user can have only two operations: reject results outputting or broadcast wrong results.

A robust threshold signature scheme tolerates some dishonest signers’ participation during the signature generation. Assume that $X = \{U_i\}_{i\in \{1, 2, \ldots, t\} - 1}$ denotes the set of corrupted users, and $Y = \{U_i\}_{i\in \{n\} - (n \geq 2t - 1)}$ denotes the set of honest users. First, during the generation of public key sharing, if one corrupted user $U_i \in X$ does not generate the sharing, the other honest users collaborate to reconstruct $U_i$’s polynomial since each honest user $U_j \in Y$ owns $s_{ij}$. Then, the honest users compute $A_{ik} (0 \leq k \leq t - 1)$ and the correct $p_i = g^{\gamma}$ according to Eq. (6). It can be observed that the number of honest users is at least $t$. That is the reason why $n \geq 2t - 1$ is required. Otherwise, if $A_{ik}$ broadcasted by user $U_i$ is wrong, $U_i$ cannot pass the verification in Eq. (5), and then the honest users reconstruct it. If user $U_i$ broadcasts a wrong $p_i$, $U_i$ cannot pass the verification in Eq. (6), and then the right one can be reconstructed by the same Eq. (6).

To conclude, valid public key sharings can be generated when less than $t$ users are corrupted by adversaries. Then,
in the phase of signature sharing generation, the number of honest users is greater than \( n - (t - 1) \geq t \) when users in set \( X \) refuse to broadcast their sharings. Thus, a valid signature can be aggregated with the remaining valid signature sharings. If users in set \( X \) broadcast invalid signature sharings, they cannot pass the verification and would not be used in signature aggregation. Therefore, the signature scheme is robust.

6.5 Unforgeability

Unforgeability is an essential requirement in \((t, n)\) threshold signature which ensures that only legal parties can generate valid signatures.

**Theorem 7.** The proposed scheme is unforgeable if there is no such adversary that gets the public key \( pk \) as well as the adaptively chosen signatures of \( k \) messages \( \{m_1, \ldots, m_k\} \), can produce a valid signature on a new message \( m \) that is different from \( \{m_1, \ldots, m_k\} \) with non-negligible probability.

**Proof.** Suppose that we have an adversary \( A \) who is able to corrupt less than \( t \) users and a challenger \( C \) who answers adversary \( A \)'s query to hash function and signing oracle. We define a sequence of games, \( Game_0, Game_1, \ldots, Game_5 \), where \( Game_0 \) is an actual game and the rest games are modified attack games. Let \( C \) denote the event that the adversary wins in \( Game_1 \) in this proving process. With these games, we show that adversary \( A \) forges a valid threshold signature by solving the CDH problem in \( G \).

**Game_0:** Adversary \( A \) is able to access a random oracle \( O_H \) and a signing oracle \( O_S \), as well as to corrupt \( t - 1 \) users in set \( X \) defined in **Theorem 6**. The goal of \( A \) is to forge a signature sharing which is capable of passing the verification. We use \( C_0 \) to denote the event that the forged signature passes the verification. Accordingly, we have

\[
\epsilon = \text{Succ}_A = \Pr[\text{Forge}_0].
\]  

**Game_1:** The corrupted users in set \( X \) are \( U_1, U_2, \ldots, U_{t-1} \). Referring to the probability of guessing right defined in [22], we have

\[
\Pr[\text{Forge}_1] = \frac{(t - 1)! \cdot (n - t + 1)!}{n!} \Pr[\text{Forge}_0].
\]  

**Game_2:** First, challenger \( C \) selects a number \( a \leftarrow Z_p \) randomly and replaces the public key \( pk = g^a \) and secret key \(\alpha = a\). Then, we perform the following steps for each user in set \( X \):

1. For all users in set \( X \), challenger \( C \) randomly selects \( \alpha_1', \alpha_2', \ldots, \alpha_{t-1}' \in Z_p \) as their secret key sharings and set their public key sharings as \( \alpha_i = g^\alpha \). The public key is set as \( pk = g^a \).
2. For other honest users in set \( Y \), we compute

\[
p_i' = \frac{1}{L(i,i) \cdot (pk - \sum_{j=1}^{i-1} L(j,i)p_j)},
\]

where \( L(x,y) \) is defined in [27] and \( x, y \in [1, n] \).
3. The challenger selects another \( t - 1 \) random numbers \( \alpha_1, \alpha_2, \ldots, \alpha_{t-1} \in Z_p \) to set the corrupted users' public key sharings \( p_i = g^\alpha \) and secret key sharing \(\alpha_i = x_1\).

4. For honest users in set \( Y \), challenger \( C \) selects \( p_1, p_{1+1}, \ldots, p_{t-1} \in G_1 \) as their public key sharings, and selects \( p_n = pk - \sum_{i=1}^{n-1} p_i \) as the public key sharing of user \( U_n \).
5. Finally, challenger \( C \) aggregates the secret key and public key by integrating \( \alpha_i \) and \( p_i \) to obtain \(\alpha_i \) and \( p_i \), respectively.

With the preceding construction, the distributions of these values keep identical to the distribution in \( Game_1 \). The adversary cannot distinguish the corrupting \( t - 1 \) members from the real game. Thus, we have

\[
\Pr[\text{Forge}_2] = \Pr[\text{Forge}_1].
\]  

**Game_3:** Adversary \( A \) is allowed to query \( q_b \) times in this game. On a hash query issued by \( A \), in order to embed the challenge \( g^b \) into an oracle answer, challenger \( C \) selects a random number \( r \leftarrow Z_p \) and computes \( H(id)g_1^m = (g_1^b)^r \) when a fresh query \( (id, m) \) is created. Then, we store \( (id, m, r, (g_1^b)^r) \) in the hash list and return \( H(id)g_1^m \) as the answer to hash query. As \( r \) is randomly selected from \( Z_p \), this game is therefore perfectly indistinguishable from the previous one. As a result, we have

\[
\Pr[\text{Forge}_3] = \Pr[\text{Forge}_2].
\]  

**Game_4:** In this game, if and only if \((id, m)\) has not been queried before, and the outputs of \((id, m)\) passes the verification, challenger \( C \) would keep executions. As \( H(id)g_1^m \) is uniformed in \( G_1 \), the probability of this event in this game is \( 1/p \). Therefore, we have

\[
\Pr[\text{Forge}_4] - \Pr[\text{Forge}_3] \leq \frac{1}{p}.
\]  

**Game_5:** In this game, adversary \( A \) is allowed to query \( q_s \) times. For any message which has not been queried before, challenger \( C \) returns \( H(id)g_1^m = (g_1^b)^r \). We use \( \sigma = (H(id)g_1^m)^{1/r} \) to pass the verification. Given \( g_1^b \) and \( g_1^a \), \( C \) is able to output \( g_1^b \) with an adversary of \( \Pr[\text{Forge}_5] \):

\[
\Pr[\text{Forge}_5] - \Pr[\text{Forge}_4] \leq \frac{q_s}{p}.
\]  

Putting Eqs. (14)-(19) all together, we conclude that the advantage of adversary \( A \) to win all the games is \( \epsilon' \). Clearly, if adversary \( A \) has a non-negligible advantage of generating a forgery, \( \text{Adv}_{\epsilon'}^{\text{CDH}} = \epsilon' \) is an non-negligible advantage to solve the CDH problem, which contradicts to the CDH assumption on \( G_1 \).

\[
\epsilon' \leq \frac{(t - 1)! \cdot (n - t + 1)!}{n!} \cdot \epsilon - \frac{q_s}{p}.
\]  

Therefore, it is computationally infeasible for adversary \( A \) to generate a forgery. In other words, an adversary is unable to forge a valid signature in IPANM.

6.6 Data Privacy

Public auditing scheme should protect the data from concealing to the verifier.

**Theorem 8.** A public auditor cannot learn the knowledge of blocks during the verification of data integrity.
Proof. During the verification, the public auditor sends challenge message $\text{chal} = \{(i, v_i)\}_{i \in I}$ to the cloud and receives proof message $\text{Pro} = \{(id_{ik})_{i \in I}, \mu, \varphi, R\}$. If a public verifier gets the combined message $\mu' = \sum_{i \in I} \nu_i m_i$, the verifier can get the content of data by collecting a sufficient number of linear combinations of message signatures and solving the resultant linear equations. For this reason, we execute a blind operation on it as $\mu = \mu + \gamma \mu'$, which masks $\mu'$ with a random element $r \leftarrow \mathbb{Z}_p$ by random masking technology. If the public auditor wants to solve bilinear equations, the auditor must get the value of $r$. In addition to guessing it with the probability $1/p$, the auditor can alternatively obtain $r$ from $R = e(g_1, pk)^r$ through computing the value of $e(g_1, pk)$. However, computing $r$ is as difficult as solving the Discrete Logarithm problem [13] in $\mathbb{G}_1$, which has been demonstrated to be computationally infeasible. In other words, the attacker cannot get the $m$ by solving a set of linear equations. In this manner, IPANM protects data privacy.

7 Performance Analysis and Evaluation

In this section, we first analyze the storage, computation, and communication costs of IPANM, and then evaluate the performance of IPANM in experiments.

7.1 Storage Overhead Analysis

Table 1 lists the storage overhead of each party in the proposed scheme, where “User $U$ (B)” means the statement of the user who hasn’t got the secret key sharing. As shown in Table 1, $t$ is the threshold value, $n$ denotes the number of users, $\ell$ is the number of file blocks, and $\mid | \mid$ means the length.

| Storage overhead | User $U$ (B) | $t|Z_p| + (t + 2)(n - 1)|G_1| + |G_1| + |Z_p| + |G_1| |
|------------------|-------------|---------------------------------------------|
| User $U$         | $t|Z_p| + (t + 2)(n - 1)|G_1| + |G_1| + |Z_p| + |G_1| |
| Aggregator       | $(t + 1)|ID| + 2|G_1| + |Z_p| + |G_1| |
| Cloud            | $t|Z_p| + (t + 2)(n - 1)|G_1| + |G_1| |

In the generation of threshold key, each user needs to choose a random polynomial. Accordingly, the user has to store the coefficients with a storage overhead of $t|Z_p|$. In addition, the secret value sharings $s_{ij}$ ($j \in [1, n], i \neq j$) lead to $(n - 1)|G_1|$ storage overhead. Each user $U_i$ will receive $s_{ji}$ and $C_{jk}$ from other users, which requires $(n - 1)|G_1|$ and $(n - 1)|G_1|$ storage spaces, respectively. Moreover, before user $U_i$ generates the secret key sharing with the list of $Q$, $U_i$ requires a storage space of $t|Z_p| + (t + 2)(n - 1)|G_1|$ for verification and generation of sharing. Finally, when user $U_i$ generates the secret key sharing and public key sharing, $U_i$ only needs to store these two sharings $\sigma_i$ and $p_i$ with a storage overhead of $|Z_p| + |G_1|$. As an aggregator, user $U_a$ who aggregates the signature $\sigma_i$ of block $m_i$ has to store $t$ IDs of valid signature sharings’ signers for distributing award when $\sigma_i$ is accepted. Since user $U_a$ needs to store $\{\sigma_i, (1_{ij}, \ldots, \sigma_{ij}), (ID_{1_{ij}}, \ldots, ID_{ij}), id_i\}$ for block $m_i$, an additional storage space $(t + 1)|ID| + |G_1|$ is required.

The cloud only needs to store file blocks and their corresponding identifiers and signatures. Thus, for a file $M$ with $\ell$ blocks, storing $\{(m_1, \ldots, m_\ell), (id_1, \ldots, id_\ell), (\sigma_1, \ldots, \sigma_\ell)\}$ on cloud has the cost of $t|Z_p| + |id| + |G_1|$ storage space. In our scheme, all signatures are generated by an identical secret key $\alpha$ and the integrity verification only needs public key $pk$. As a result, the cloud does not need to store signer information, which is different from existing public auditing schemes [13], [17].

7.2 Communication Overhead Analysis

Based on the generation of signature and challenge-and-response interaction in public auditing, we give the communication cost of the proposed protocol from three parts: threshold key generation, signature aggregation and public auditing which can be further divided into two steps. Before detailing the simulation, we introduce the meanings of the notations: $n$ denotes the number of users, $t$ is the threshold value, $d$ is the number of users in set $QU$ generating secret key sharings, $c$ is the number of challenging blocks, and $\mid | \mid$ means the length.

In the step of threshold key generation, first, each user $U_i$ broadcasts $C_{ik}$ ($k \in [0, t - 1]$) and sends $s_{ij}$ ($1 \leq j \leq n, j \neq i$) to user $U_j$ secretly. The valid user $U_j$ in set $QU$ sends $A_{jk}$ ($k \in [0, t - 1]$) to help other users verify the correctness of their public key sharings. At last, each user $U_i$ broadcasts the public key sharing. Assume that the number of valid users is $d$, the communication overhead in this step is determined as $nt|G_1| + n(n - 1)|Z_p| + dt|G_1| + n|G_1|$.

In the step of signature aggregation, a signature aggregator requires $t$ valid signature sharings to generate the final signature $\sigma_a$ for block $m_a$. Each user $U_i$ in set $SG$ sends $(id_{ik}, \sigma_a, ID_i)$ to the aggregator. Therefore, the communication cost of signature aggregation is $t|id| + |ID| + |G_1|$.

The public auditing consists of two steps: proof generation and proof verification. In the first step, the verifier sends challenge $\text{chal} = \{(i, v_i)\}_{i \in I}$ to the cloud while in the second step, the verifier receives audit proof $\text{Pro} = \{(id_{ik})_{i \in I}, \mu, \varphi, R\}$ from the cloud. Thus, the communication cost of these two steps are $2c|Z_p|$ and $c|Z_p| + 2|G_1| + |G_2|$, respectively.

Table 2 lists the communication cost (size of messages from senders to receivers) of four steps in IPANM.

| Communication overhead | Key generation | $nt|G_1| + n(n - 1)|Z_p| + dt|G_1| + n|G_1|$ |
|------------------------|----------------|---------------------------------------------|
| Signature aggregation  | $t|id| + |ID| + |G_1|$ |
| Proof generation       | $2c|Z_p|$ |
| Proof verification     | $c|Z_p| + 2|G_1| + |G_2|$ |

7.3 Computation Overhead Analysis

In the computation overhead analysis, we use $E_{Z_p}$, $E_{G_1}$, and $E_{G_2}$ to denote the computation time of exponentiation operation in $Z_p$, $G_1$, and $G_2$, respectively. $M_{Z_p}$, $M_{G_1}$, and $M_{G_2}$ to denote the computation time of multiplication operation in $Z_p$, $G_1$, and $G_2$, respectively. $Hash$ to denote the computation cost of a Hash operation in $G_1$, and $Pair$ to
denote the computation overhead cost of a pair operation. We analyze the computation overhead from the following two aspects: signature generation and public auditing.

7.3.1 Signature Generation

During the secret key sharing generation, each user $U_i$ ($i \in [1, n]$) is required to compute $C_{ik}$ ($k \in [0, t]$) and $s_{ij}$ ($1 \leq j \leq n; j \neq i$). The computation costs are $tE_{G_1}$ and $(n-1)[(t-2)Z_p + (t-1)Z_p]$, respectively. The cost for verifying a single $s_{ij}$ is $E_{G_1} + (t-1)(M_{G_1} + Z_p)$. During public key sharing generation, each user $U_i$ in set $QU$ has already computed $C_{ik}$, which is equal to $A_{ik}$. The verification of $A_{ik}$ costs $E_{G_1} + (t-1)(M_{G_1} + (t-1)Z_p)$. The computation cost of generating a final public key sharing $p_{i1}$ and verifying the correctness of $p_{i1}$ is $E_{G_1}$ and $(dt + t-1)M_{G_1} + (t-1)E_{Z_p}$, respectively.

In addition, $t$ signature sharings which pass the verification are aggregated to generate the final signature with a cost of $(2t-1)M_{G_1}$. Verifying the final signature leads to $2Pair + E_{G_1} + M_{G_1} + Hash$. Verifying a signature sharing costs $2Pair + E_{G_1} + M_{G_1} + Hash$. The computation costs are $2Pair + (c+1)M_{G_1} + (c+3)E_{G_1} + cHash$ to check the correctness. The computation overhead of public auditing is summarized in TABLE 4.

7.4 Experimental Results

In addition to the performance analysis, we also carry out extensive experiments to evaluate the effectiveness of our IPANM by comparing it with IAID-PDP [17] and NPP [14]. IAID-PDP and NPP both solve the problem of centralization caused by group managers. Specifically, IAID-PDP utilizes ring signature to construct a public auditing scheme with incentive and provides group users with equal power and NPP distributes the power of management to multiple managers. In order to make a fair comparison, we adopt the same simulation settings in comparison of IPANM, IAID-PDP and NPP. Note that NPP belongs to group signature based scheme, we only show the differences between them in the process of public auditing.

We adopted the Pairing Based Cryptography (PBC) [27] library to simulate the cryptographic operations in our scheme IPANM. All the experiments are tested on an Intel Core i7 processor of 3.40 GHz, and the Ubuntu operating system. Without loss of generality, we test our scheme over 1,000 times. For the setup of benchmarking schemes [13], [14], the size of $ID$, $G_1$, $G_2$, and $Z_p$ are all set to 160bit, and the size of block tag $id$ is set to 40bit. We assume the size of a file is 10M, thus the number $\ell$ of blocks in the shared data is set to 524,288 when the size of block $m_{i1}$ is set to 160bit. According to the observation of our previous work [6], we need to set the number $c$ of selected blocks in an auditing task to 460, in order to ensure that the detection probability is greater than 99%. Note that the detection probability is greater than 95% when $c = 300$. Since the proposed scheme IPANM consists of an incentive mechanism and a public auditing method, we implement two sets of simulation experiments to evaluate IPANM. In the first set of simulation experiments, to show the performance of the incentive mechanism, we compare the storage overhead of IPANM and IAID-PDP [17], analyze the communication overhead of generating keys and signatures, and compare the computation overhead of IPANM and IAID-PDP [17] in the signature generation. In the second set of simulation experiments, to demonstrate the performance of the public auditing method, we compare the communication and computation overheads of IPANM with that of IAID-PDP [17] and NPP [14].

To estimate the gas cost of the proposed incentive mechanism, we deploy it on the locally simulated Ethereum network built by Ganache [28]. Ganache provides a fast and flexible way to build a virtual blockchain which makes developers more focus on the performance of the deployed smart contracts without considering the influence of complex network circumstances. In the implementation, the Ganache runs on a laptop with 2.5GHz Intel(R) Core(TM) i5-3210M CPU, 8GB memory, and Windows 10 operating system. The size of the elements in records such as $(id_{i1}, pk_{i1}, N_{OA})$ is set to 40bit, 160bit, and 40bit, respectively.

7.4.1 Evaluation of Incentive Mechanism

In IPANM, we employ blockchain to record the signatures for further incentive, which will bring extra overhead such as: 

<table>
<thead>
<tr>
<th>Step</th>
<th>Computation overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof generation</td>
<td>$Pair + 2M_{Z_p} + (c+1)M_{G_1} + cE_{G_1} + E_{G_2} + Hash$</td>
</tr>
<tr>
<td>Proof verification</td>
<td>$2Pair + (c+1)M_{G_1} + (c+3)E_{G_1} + cHash$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 3: Computation overhead of signature generation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SK sharing G</td>
</tr>
<tr>
<td>SK sharing V</td>
</tr>
<tr>
<td>PK sharing G</td>
</tr>
<tr>
<td>PK sharing V</td>
</tr>
<tr>
<td>SS G</td>
</tr>
<tr>
<td>SS V</td>
</tr>
<tr>
<td>Signature G</td>
</tr>
<tr>
<td>Signature V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 4: Computation overhead of public auditing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
</tr>
<tr>
<td>Proof generation</td>
</tr>
<tr>
<td>Proof verification</td>
</tr>
</tbody>
</table>
as storage overhead, communication overhead, computation overhead and gas cost. To show the performance of incentive mechanism, we compare the cloud storage overhead of IPANM and IAI-D-PDP [17], analyze the communication overhead of generating keys and signatures, and compare the computation overhead of IPANM with that of IAI-D-PDP [17] and NPP [14].

Comparison of the Storage Overhead: In the evaluation of storage overhead, we compare the cloud storage overhead of IPANM and IAI-D-PDP under the varying size n of user groups (n is set to 12, 24, 36, 48, 60, 72, 84, 96). Due to the blockchain based incentive mechanism in IPANM, in addition to storing file blocks and their corresponding identifiers and signatures such as \(\{m_i, id_i, \sigma_i\}\) for block \(m_i\), which costs \(480t\) bits in IPANM, we also need to store the message generated during the process of key generation and signature generation. In the generation of threshold key, each user first selects a random polynomial which costs \(160t\) bits and the secret value sharings \(s_{ij}\) (\(j \in [1, n], i \neq j\)) lead to \(160(n-1)\) bits. Each user then spends \(160(n-1)\) bits and \(160t(n-1)\) bits storage spaces to store the received \(s_{ij}\) and \(C_{jk}\) from other users, respectively. Moreover, when user \(U_i\) requires a storage space of \(160m_i+320n-320\) bits for verification and generation of sharing. Finally, when user \(U_i\) generates the secret key sharing and public key sharing, \(U_i\) needs to store these two sharings \(\alpha_i\) and \(p_i\) with a storage overhead of 320 bits. In the generation of signature, the aggregator \(U_a\) needs to store \(\{\sigma_i, \sigma_{i1}, \ldots, \sigma_{it}, ID_{t1}, \ldots, ID_{tn}, id_t\}\) for block \(m_t\), an additional storage space \(320(t+1)\) bits is required. The cloud only needs to store file blocks and their corresponding identifiers and signatures. The storage overhead is \(480t\) bits. We let IPANM-1 and IPANM-2 denote the IPANM method when \(t\) is set to \(n/2\) and \(n/3\), respectively. We calculate the storage overheads of the cloud using IPANM-1, IPANM-2, and IAI-D-PDP when the size of user group is increased from 12 to 96.

The results listed in TABLE 5 demonstrate that the storage cost savings of IPANM-1 and IPANM-2 can be up to 85.496% and 89.567%, respectively, even if we introduce blockchain into IPANM to record the relevant information produced during the process of key and signature generation. The overhead savings achieved by IPANM-1 and IPANM-2 is due to that IAI-D-PDP needs to store \(n - 1\) more signatures than IPANM. The results also indicate that the cloud storage overhead grows with the increase in the size of use group. This is because that the storage overhead highly depends on the number \(t\) of signers coming from use group.

### TABLE 6: Comparison of computation overhead in signature generation.

<table>
<thead>
<tr>
<th></th>
<th>n = 12</th>
<th>n = 24</th>
<th>n = 36</th>
<th>n = 48</th>
<th>n = 60</th>
<th>n = 72</th>
<th>n = 84</th>
<th>n = 96</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1 = n/2</td>
<td>0.482</td>
<td>0.986</td>
<td>1.49</td>
<td>1.994</td>
<td>2.498</td>
<td>3.002</td>
<td>3.506</td>
<td>4.01</td>
</tr>
<tr>
<td>t2 = n/3</td>
<td>0.233</td>
<td>0.473</td>
<td>0.713</td>
<td>0.953</td>
<td>1.193</td>
<td>1.433</td>
<td>1.673</td>
<td>1.913</td>
</tr>
<tr>
<td>IAI-D-PDP (ms)</td>
<td>0.316</td>
<td>0.632</td>
<td>0.947</td>
<td>1.263</td>
<td>1.583</td>
<td>1.903</td>
<td>2.223</td>
<td>2.543</td>
</tr>
<tr>
<td>Advantage-1 (%)</td>
<td>51.660</td>
<td>52.028</td>
<td>52.148</td>
<td>52.207</td>
<td>52.242</td>
<td>52.626</td>
<td>52.926</td>
<td>53.096</td>
</tr>
<tr>
<td>IPANM-2 (ms)</td>
<td>0.153</td>
<td>0.313</td>
<td>0.473</td>
<td>0.663</td>
<td>0.793</td>
<td>0.953</td>
<td>1.113</td>
<td>1.273</td>
</tr>
<tr>
<td>Advantage-2 (%)</td>
<td>68.258</td>
<td>68.256</td>
<td>68.255</td>
<td>68.255</td>
<td>68.255</td>
<td>68.254</td>
<td>68.254</td>
<td>68.254</td>
</tr>
</tbody>
</table>

![Fig. 4: Communication overhead of key generation and signature aggregation.](image-url)
without communication overhead since its signature is only determined by the signer, we only show the communication overhead of IPANM in the generation of keys and signatures without comparison. For each operation, the communication overhead is measured by the size of messages from senders to receivers.

In the evaluation of communication overhead, we first compute the communication overhead of IPANM in the generation of keys and signatures. Assume that each user \( U_i \) broadcasts \( C_{ijk} \) and sends \( s_{ij} \) to user \( U_j \) secretly during the threshold key generation. The valid user \( U_j \) in set \( QU \) needs to send \( A_{jk} \) for the verification of their public key sharings. At last, each user \( U_j \) broadcasts the public key sharing. Thus, the communication overhead of threshold key generation is \( 160(n + n^2 + d) \) bits where \( d \) is the number of valid users. In signature aggregation, the aggregator costs 400 + 320 bits to get \( t \) valid signature sharings and the information of the signed block for generating the final signature. Since the communication overhead of key generation and signature aggregation in IPANM depends on the size \( d \) of qualified group \( QU \) and the value of \( t \), we investigate the communication overhead in the scenario of \( d = t = \frac{n+1}{2}, \quad d = t, \quad \text{and} \quad n \) where \( t \) increases 0 to 100. Fig. 4 plots the communication overhead of key generation and signature aggregation. The results in the figure indicate that the communication overhead of signature aggregation is practically negligible regardless of the value of \( t \), whereas the communication overhead of key generation is non-negligible and grows with the increase of the value of \( t \).

Comparison of the Computation Overhead: In the evaluation of computation overhead in incentive mechanism, we compare the computation overhead of IPANM and IAID-PDP in the signature generation. Same to the evaluation of storage overhead, we let IPANM-1 and IPANM-2 denote the IPANM method when \( t \) is set to \( n/2 \) and \( n/3 \), respectively. \( n \) takes the value of \{12, 24, 36, 48, 60, 72, 84, 96\}. The comparison results (see TABLE 6) clearly demonstrate that the computation overhead of IPANM-1 and IPANM-2 are both lower than that of IAID-PDP. Same to the evaluation of storage overhead, IPANM-1 and IPANM-2 are utilized to denote the IPANM when \( t \) is set as \( n/2 \) and \( n/3 \), respectively, and Advantage-1 and Advantage-2 are utilized to represent the reduction (%) of computation overhead achieved by IPANM-1 and IPANM-2, respectively. TABLE 6 presents the computation overhead of signature generation using IAID-PDP, IPANM-1, and IPANM-2 under the varying size \( n \) of user group. The results clearly show that IPANM-1 and IPANM-2 are both computation-efficient as compared to IAID-PDP. Specifically, the computation overhead of IPANM-1 and IPANM-2 are about 52% and 68% lower than that of IAID-PDP, respectively.

Overhead of Blockchain: We evaluate the overhead (quantified by gas cost) of the proposed blockchain based incentive mechanism from two aspects: recording and reward. During the recording process, the system records the message of \{\( (id_s, pk_s, No_a) \}\}_{i=1}^{n/2} in the blocks of blockchain. As demonstrated in Fig. 5, the gas cost is 539932, 1079928, 2159920, 4319840, and 8639744 when the number of contributors \( t \) is set to \{4, 8, 16, 32, and 64\}, respectively. For the reward process, the gas cost is 58384, 94838, 167512, 312626, and 589220 when the number of contributors \( t \) is set to \{4, 8, 16, 32, and 64\}, respectively. Compared to the gas cost of data management [29], it’s obvious that the gas cost of the proposed incentive mechanism is acceptable.

7.4.2 Evaluation of Public Auditing

In this section, we compare our scheme IPANM with IAID-PDP and NPP from two parts: communication overhead and computation overhead to show the performance of public auditing.

Comparison of the Communication Overhead: In this step, we compare the communication overhead of IPANM with that of IAID-PDP and NPP in public auditing. The public auditing includes proof generation and proof verification. In proof generation, the challenge message costs 320c bits. The proof verification needs 480 + 160c bits to transmit the proof message. For one time interaction, the whole communication overhead of public auditing is \((160 * (c + 5 + 0 + d) + 20) / 8\) /1024 KB. Fig. 6 shows the communication overhead of public auditing in IPANM, IAID-PDP and NPP when the number \( c \) of selected blocks takes the value of 300 and 460. As can be seen from the figure, the communication overhead of public auditing using IPANM is much lower than that of IAID-PDP and NPP. This is because that IAID-PDP needs to send \( n - 1 \) more elements for a message block and the auditing in NPP uses a lot of auxiliary information as compared to IPANM. We can also find from Fig. 6 (a)-(b) that the communication overhead of public auditing using IPANM and NPP is independent of the number of users while the communication overhead of public auditing using IAID-PDP is linearly increased with the size of user group as shown in Fig. 6 (c)-(d). Fig. 6 (c) and (d) are extracted from Fig. 6 (a) and (b) respectively for clearer comparison.

Comparison of the Computation Overhead: To evaluate the performance, we compare the computation overhead of IPANM, IAID-PDP and NPP in public auditing. The public auditing includes proof generation and proof verification. During the proof generation, the verifier generates a challenge message once getting the verification request from a user. According to the received challenge message, the cloud generates the proof message \( Pro = \{id_s, t, \varphi, R\} \). The computation overhead of generating this proof message is \( Pr + 2Mz_p + (c - 1)M_G + eG + E_G + Hash \). After getting the proof message, the verifier checks the correctness.
with the computation overhead of $2Pair + (c + 1)M_{G_1} + (c + 3)E_{G_1} + cHash$. The results in Fig. 7(a)-(b) show that the computation overhead of public auditing using IPANM is a small constant and is independent of the size $n$ of user group. Unlike IPANM and NPP, IAID-PDP is linearly increased with the size $n$ of use group, thus has a higher computation overhead in the most cases ($n \geq 4$) than IPANM. Although the overhead of NPP is also constant, it needs more intermediate steps than IPANM to calculate auxiliary information. Fig. 7(c) plots the total computation overhead of public auditing under a fixed number of use group and the varying number $c$ of selected blocks. The results demonstrate that the total computation overhead of IPANM, IAID-PDP and NPP are all linearly increased with the number of blocks, and IPANM is the most computation-efficient as compared to IAID-PDP and NPP.

8 SUMMARY

In this paper, we propose a fair and incentive privacy-preserving public auditing scheme IPANM that supports non-manager groups and encourages all users to take part in the generation of signatures, especially for users in crowd sensing. The main contribution of IPANM is our development of an incentive based $(t, n)$ threshold signature based public auditing scheme for non-manager groups where users are assigned the same power for non-manager groups in the cloud.

9 FUTURE WORK

In this paper, the proposed fair and incentive privacy-preserving public auditing scheme IPANM is designed for static groups without managers. IPANM is able to achieve the security and performance guarantees such as incentive, correctness, public auditing, robustness, unforgeability, as well as file privacy and efficiency. However, it does not support dynamic groups that allow group users to leave and new members to join. What’s more, considering that dynamic groups are also very common in real-world applications, they should also be taken into account in the future work. The main challenge in designing a public auditing
scheme for dynamic groups is to realize secure and efficient group dynamic. Therefore, in the future we will extend our proposed public auditing scheme IPANM to dynamic groups from the perspectives of security and efficiency as described below.

1) **Security:** secure dynamic groups should ensure forward security and backward security in secure key updates. Forward security requires that the revoked users cannot participate in the signature generation with the previous keys, and backward security requires that the newly added users cannot use the current key to crack the group information before joining in the group. In addition to providing forward and backward security, implementing secure key updates in dynamic groups without managers needs to be realized.

2) **Efficiency:** users in the dynamic groups are allowed to leave. For these leaving users, the system needs to update their associated file messages and re-sign these file messages, both of which have a large computation overhead. To reduce the computation overhead, designing an efficient file message and signature updating method for leaving users of dynamic groups is a necessity.

Undeniably, group dynamic has been supported in a lot of auditing schemes (e.g., [6], [25], [30], [31]). However, most of the existing schemes either focus on re-signing the files belonging to the revoked user (e.g., [6], [25]) efficiently or aim at solving the problem of the collusion between the cloud and the revoked group users (e.g., [30], [31]). In addition, they do not consider forward security, backward security, and non-manager groups. Therefore, these schemes are not suitable for our future work of designing a public auditing scheme that supports forward security, backward security, and group dynamic for non-manager groups.

**REFERENCES**


Longxia Huang received the Ph.D. degree in Computer Science and Technology from Nanjing University of Science and Technology, Nanjing, China, in 2019. She was a Visiting Scholar with the Deakin University, Melbourne, Australia, in 2018. She is currently a Lecture with the School of Computer Science & Communication Engineering, Jiangsu University, Jiangsu, China. Her current research interests include blockchain technology, privacy protection, data sharing and cloud computing.

Junlong Zhou (S'15-M'17) received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014–2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include embedded systems, edge computing and Internet-of-Things, and cyber physical systems, where he has published over 60 refereed papers. He has been program vice-chairs, publication chairs, publicity chairs, section chairs, and TPC members for numerous conferences. Dr. Zhou serves as an Associate Editor for the Journal of Circuits, Systems, and Computers and the IET Cyber-Physical Systems: Theory & Applications, a Subject Area Editor for the Journal of Systems Architecture: Embedded Software Design, and a Guest Editor for 5 ACM/IEEE/Elsevier/Wiley Journals.

Gongxuan Zhang (SM'12) received the B.S. degree in electronic computer from Tianjin University in 1983 and the M.S. and Ph.D. degrees in Computer Application from the Nanjing University of Science and Technology in 1991 and 2005, respectively. He was a Senior Visiting Scholar in Royal Melbourne Institute of Technology from 2001.9 to 2002.3 and University of Notre Dame from 2017.7 to 2017.10. Since 1991, he has been with the Nanjing University of Science and Technology, where he is currently a professor in the School of Computer Science and Engineering. His current research interests include multicore and parallel processing and distributed computing, and cyber space security. He has served on the program committees in a couple of conferences. He has over 80 publications and has led or participated in 40 research projects.

Jin Sun (M’17) received the B.S. and M.S. degrees in computer science from Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Arizona in 2011. Between January 2012 and September 2014, he was with Orora Design Technologies, Inc. as a member of technical staff. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include high-performance computing, integrated circuit modeling and analysis, and computer-aided design.

Tongquan Wei (M’11) received the Ph.D. degree in electrical engineering from Michigan Technological University, Houghton, MI, USA, in 2009. He is currently an Associate Professor with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interests include real-time embedded systems, green and reliable computing, cloud computing and Internet of Things. Dr. Wei has been a Regional Editor for the Journal of Circuits, Systems, and Computers since 2012. He served as a Guest Editor for several special sections of the IEEE Transactions on Industrial Informatics and ACM Transactions on Embedded Computing Systems.

Shui Yu (SM’12) received the Ph.D. degree from Deakin University, Burwood, VIC, Australia, in 2004. He currently a Professor with the School of Software, University of Technology Sydney, Broadway NSW 2007, Australia. He has published over 150 fully reviewed papers, including top journal papers and top conference papers. He has authored one monograph entitled Distributed Denial of Service Attack and Defence (Springer) and edited a book entitled Networking for Big Data (CRC, 2015). His current research interests include big data, network security, privacy and forensics, networking theory, and mathematical modeling. He targets on narrowing the gap between theory and applications using mathematical tools. Dr. Yu is active in research services in various roles. He serves on the editorial board of IEEE Transactions of Parallel and Distributed Systems, IEEE Communications Surveys and Tutorials, IEEE Access, and three other international journals. He served as a leading Guest Editor for a number of special issues, including Networking for Big Data in IEEE Network in 2014 and Big Data for Networking on IEEE Network in 2015. He is also involved in organizing international conferences, such as the TPC Co-Chair of IEEE Big Data Computing Service and Application in 2015, the Founder of International Workshop on Security and Privacy in Big Data (usually with the IEEE INFOCOM), the Symposium Co-Chair for CISS of IEEE ICC 2014 and IEEE INCC 2014, the Publication Chair for IEEE Globecom 2015 and IEEE INFOCOM 2016, and a TPC Member for INFOCOM 2012–15.

Shiyuan Hu (SM’10) received his Ph.D. in Computer Engineering from Texas A&M University in 2008. He is Professor and Chair in Cyber-Physical System Security at University of Southampton. His research interests include Cyber-Physical Systems and Cyber-Physical System Security, where he has published more than 100 refereed papers. He is an ACM Distinguished Speaker, an IEEE Systems Council Distinguished Lecturer, an IEEE Computer Society Distinguished Visitor, a recipient of the 2017 IEEE Computer Society TCSC Middle Career Researcher Award, the 2014 U.S. National Science Foundation (NSF) CAREER Award, and the 2009 ACM SIGDA Richard Newton DAG Scholarship. His publications have received a few distinctions, which includes the 2018 IEEE Systems Journal Best Paper Award, the 2017 Keynote Paper in IEEE Transactions on Computer-Aided Design, and the Front Cover in IEEE Transactions on Nanobioscience in March 2014. He is the Chair for IEEE Technical Committee on Cyber-Physical Systems. He is the Editor-In-Chief of IET Cyber-Physical Systems: Theory & Applications. He serves as an Associate Editor for IEEE Transactions on Computer-Aided Design, IEEE Transactions onIndustrial Informatics, IEEE Transactions on Circuits and Systems, ACM Transactions on Design Automation for Electronic Systems, and ACM Transactions on Cyber-Physical Systems. He has served as a guest editor for eight IEEE/ACM Journals such as Proceedings of the IEEE and IEEE Transactions on Computers. He has held chair positions in many IEEE/ACM conferences. He is a Fellow of IET and a Fellow of British Computer Society.