Identity-Preserving Public Integrity Checking with Dynamic Groups for Cloud Storage

Aoting Hu, Rui Jiang, and Bharat Bhargava, *Fellow, IEEE*

Abstract—Despite a variety of security threats, cloud storage is on the increase especially when a group of users need to store and share data. Identity-privacy and user dynamic operation are of growing concern in public integrity checking (PIC) scheme. In this paper, we develop an identity-preserving public integrity checking scheme with dynamic groups (IPIC-DG) for cloud storage. Firstly, our IPIC-DG scheme can realize the whole anonymity. On the one hand, no one except the group manager can discover the real identity of users. On the other hand, even the manager, who issues user's secret key, is not capable of forging signatures on behalf of others. Secondly, we propose an anonymous public integrity verification protocol which not only supports integrity checking without retrieving whole data from the cloud, but also protects the signer's identity during the whole process. We utilize group signature to construct a homomorphic authenticator on each file block to guarantee the anonymous remote data integrity checking. Thirdly, our scheme supports a way of dynamic user operation that greatly improves the efficiency and feasibility of user revocation. At last, we formally prove our IPIC-DG scheme is IND-CCA security. Experimental results show that our work performs well in practical application.

Index Terms—Identity-preserving, public integrity checking, user revocation, cloud storage

1 Introduction

Advances in networking technology and an increase in the need for computing resources have prompted many organizations to outsource their storage and computing needs [1]. Cloud storage, which plays important role in cloud computing by moving data and access control relatively to the cloud, can avoid the costs of building and maintaining infrastructure on their own. Although the advantages of applying a public cloud infrastructure are obvious, it brings significant security and privacy problems. In fact, confidentiality and integrity are the most restricted hurdle to popularize the cloud storage.

Recently, more and more new anonymous applications are developed in cloud storage. E-auction is one of the application scenarios that requires the confidentiality of user's identity. EBay, Amazon.com, Yahoo possess the auction agent system that allows users to bid in public auction. In general, a group of bidders register to the authority to obtain their credentials. When auction begins, every bidder commits the price tag to auction house with the price signed by qualified bidder's secret credential. Before the final decision, every bidder can alter his bid continuously. At the end of auction, the successful bidder's identity is revealed by authority. The previous auction schemes [2], [3], [4], [5] are based on RSA algorithm. The signature length of their schems are quite long when achiving the same level of security. Besides, dynamic membership and public integriy checking are common flaws in exsiting works. Moreover, previous works [2], [3], [4], [5] can only prove anonymity under chosen plaintext attack.

In addition, data sharing on private cloud is a suitable scenario where every member is going to share resources anonymously among groups. In most of universities, private tracker station (PT station) is designed for students to share resources anonymously. In this station, students are allowed to upload resources anonymously. The manager is able to revoke the misbehaved members. New members are allowed to enter the station as long as they are approved by the manager. There exsists a third-party auditor who responsible for checking the integrity of data stored on station remotely. In order to support remote data integrity checking, some researches such as [6], [7], [8], [9], [10], [11], [12] and [13] were done. However, there are few of them support identity-preserving [6], [9]. Moreover, the dynamic group operation is not achievable in Oruta [6]. Member revocation and unforgeability are not guaranteed in Knox [9].

In order to solve identity preserving, dynamic membership and remote data integriy checking simultaneously for a data shaing system. In this paper, we propose an identity-preserving integrity checking scheme which can support dynamic groups in cloud storage. Our work is based on the following researches.

1.1 Related Works

To achieve anonymity (identity preserving) in groups, the concept of group signature was first introduced by Chaum and van Heyst in 1991 [14]. They provided an anonymous member authentication scheme, called group signature, to prove a user belong to a certain group. There was also

- *A. Hu and R. Jiang are with the School of Information Science and Engineering, Southeast University, Nanjing, Jiangsu 210096, China. E-mail: 531888137@qq.com, R.Jiang@seu.edu.cn.*
- *B. Bhargava is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA. E-mail: bbshail@purdue.edu.*

Manuscript received 12 Nov. 2017; revised 13 July 2018; accepted 21 July 2018. Date of publication 27 July 2018; date of current version 5 Aug. 2021. (Corresponding author: Rui Jiang.)
Digital Object Identifier no. 10.1109/TSC.2018.2859999

another solution for tackling identity preserving, called ring signature [15], which was applied in Oruta [6]. However, ring signature algorithm can not realize dynamic group operation. Besides, the signature length and signature generation complexity are linearly growing with the number of groups. Therefore, we focus on the group signature algorithms to realize identity preserving supporting dynamic group member in cloud storage.

As to the group signatures algorithm, dynamic user operation, which includes user join and user revocation, is still a stronghold need to be addressed. However, most of former group signature schemes that based on [16] are static, which means that the number of group members and their identities were fixed and frozen in the setup phase so that the scheme could not support real-time group member operation. A structure for dynamic group signature was proposed in [17]. The authors provided a structure and security model for dynamic group signatures. But they did not provide specific solution. Ateniese et al. [18] proposed a group signature scheme supporting strong notion of exculpability. In [18], only the data owner could sign the message. However, the authors did not deal with the user revocation problem. One simple solution [19], [20], [21] for user revocation is to update public key and secret key of unrevoked users every time after user revocation. However, it required the group manager to generate and distribute key frequently. Also, the previous signatures should be re-signed with new keys. So, this approach is unrealistic in large groups for cloud environment. Song [22] presented a forward secure group signature schemes which provided a solution to user exclusion without updating user's secret key. However, the verification time was linearly increasing with the number of revoked user. Camenisch and Lysyanskaya [23] proposed an accumulator-based revocable solution and incorporated it into ACJT [18] scheme. However, their scheme could not keep secrecy about archive *Eadd* and *Edelete* from revoked user. So, the revoked user could update the membership credentials itself. Dan Boneh and Hovav Shacham [24] provided a group signature scheme with verifier-local revocation by checking whether the signer belong to revocation list or not. However, it required O(R) costs in verification phase when R is the length of revocation list. Zhu et al. [25] proposed a data sharing scheme for dynamic groups by utilizing polynomial function, which triggered our method of dynamic user management. In this paper, we guarantee the real-time user join with strong excupability and efficient user revocation.

Except for revocation problem, classical group signature schemes [19], [20], [26], [27] are not suitable for cloud storage because they adopted hash values to generate the final signature. For data integrity checking, these schemes need to retrieve whole file and zero knowledge proof from cloud storage to verify the correctness of signatures generated by owners. Thus, it is a difficult task to apply group signature into cloud storage environment. However, the homomorphic authenticators are designed for solving such problems and make it suitable for remote data integrity checking senario.

In order to ensure the remote data integrity, provable data possession (PDP) [28] was proposed by Hovav Shacham and Brent Waters. They designed a basic public verification construction by hiring a Third-Party Auditor (TPA). Afterword, many improved remote data integrity checking schemes were proposed [6], [7], [8], [9], [10], [11], [12], [13]. For example, An ID-based RDIC scheme [12] was proposed to provide the efficient remote data integrity checking scheme. However, it cannot support either identity-preserving or efficient dynamic user operation. Moreover, the group manager is aware of user's secret key in Extract algorithm and he can forge signatures on behalf of any user. There are few schemes that can support identity-preserving [9], [6]. Oruta [6] was a typical public integrity verification scheme supporting privacy-preserving within groups. In the scheme, the authors applied ring signature [15] and designed the homormorphic authenticators to achieve public auditing. However, owing to the natural features of ring signature, the dynamic group operation was not achievable. Knox [9] was another privacy-preserving public auditing scheme. In the scheme, the authors applied group signature to construct homormorphic authenticators. However, the authors failed to guarantee strong exculpability so that the group manager was able to forge the signature of any members. In addition, although the authors claimed to support user revocation, they did not provide a specific way to update the group secret key.

1.2 Contributions

As far as we know, there is no scheme that realizes identity-preserving public integrity checking with efficient dynamic groups for cloud storage currently. In this paper, we develop an identity-preserving public integrity checking scheme with dynamic groups (IPIC-DG) for cloud storage.

The main contributions of our IPIC-DG scheme are:

1. Our IPIC-DG scheme achieves identity-preserving public integrity checking on cloud server based on homormorphic authenticators on group signature algorithm. The real identity of members will be treated confidentially during whole process of scheme. The data stored on cloud sever canbe examed by TPA remotely. We construct homormorphic authenticators based on improved group signature algorithms for each file blocks to ensure the integrity of data could be examed by TPA remotely without exposing the real identity of data owner.
2. Our IPIC-DG scheme supports dynamic membership for anonymous public integrity verification scheme. The dynamic user operation includes user join and revocation. No one including group manager is able to forge the signature of members. Once a user is revoked, (s)he can no longer upload data to the cloud server anymore. The previous data stored on cloud server do not need to be updated. The group manger constructs the polynomial function to manage group memberships and enrypts the secret group key with exponential coefficients of polynomial function.
3. We formally prove IPIC-DG shceme is correct, existential unforgeable under adaptive chosen-message attacks [29], and anonymous under Chosen Ciphertext Attack (CCA) in random oracle model. Also, we evaluate the security functions and performances of our scheme and compare it to state-of-art.

2 PRELIMINARIES

In this section, we introduce the bilinear group and some computational assumptions used in this paper.

2.1 Bilinear Groups

The bilinear maps [26] are defined as follows.

1. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two (multiplicative) cyclic groups of prime order p,
2. g_1 are generators of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$,
3. ψ is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi\ (g_2) = g_1$,
4. e is an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties:
   a) Bilinear: for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e\ (u^a, v^b) = e(u, v)^{ab}$;
   b) Non-degenerate: $\mathrm{e}(g_1, g_2) \neq 1$.

Generally speaking, we simply assume that ψ exists and efficiently computable. $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups for an elliptic curve $E/\mathbb{F}_q$. In this case, $\mathbb{G}_1 \in E/\mathbb{F}_q$ and $\mathbb{G}_2 \in E/\mathbb{F}_q$.

2.2 Computational Assumptions

1) The Discrete Logarithm Problem

Definition 1. *Given a random generator $g \in \mathbb{G}$, and an element $g^{\alpha} \in \mathbb{G}$, the problem is to output α. We define the advantage of an algorithm $\mathcal{A}$ in solving DL problem as:*

$$AdvDL \stackrel{\text{def}}{=} Pr[\mathcal{A}(g,\ g^{\alpha}) =\ \alpha :\ \alpha \in \mathbb{Z}_p^*].$$

2) The Decision Diffie-Hellman Problem (DDH) [30]

Definition 2. *Given a random generator $\in \mathbb{G}$, two random elements g^a, g^b in $\mathbb{G}$, and a candidate $X \in \mathbb{G}$, the problem is to distinguish betwwen the two distributions $(g,\ g^a,\ g^b,\ g^{ab})$ and $(g,\ g^a,\ g^b,\ X)$. We define the advantage of an algorithm $\mathcal{A}$ in solving DDH problem as:*

$$AdvDDH \stackrel{\text{def}}{=} Pr[\mathcal{A}(g,\ g^a,\ g^b,\ g^{ab}) =\ \text{“true”}] - Pr[\mathcal{A}\ (g,\ g^a,\ g^b,\ X) = \text{“}ture\text{”}]).$$

3) The External Diffie-Hellman Assumption (XDH) [31]

Definition 3. *Given three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, as well as a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, although the DDH problem is easy in $\mathbb{G}_2$, the XDH assumptions declare that DDH problem still hold in $\mathbb{G}_1$.*

4) The Strong Diffie-Hellman Assumption (SDH) [27]

Definition 4. *Given two isomorphic groups $\mathbb{G}_1, \mathbb{G}_2$ and a $(q+2)$ tuple $(g_1,\ g_2,\ {g_2}^{\gamma}, {g_2}^{\gamma^2}, \ldots,\ {g_2}^{\gamma^q})$, for a random generator g_2 of $\mathbb{G}_2$, a random generator g_1 of $\mathbb{G}_1$, $\gamma \in_R \mathbb{Z}_p^*$, and $\psi\ (g_2) = g_1$. The problem is to compute a pair $(x, {g_1}^{\frac{1}{x+\gamma}})$, where $x \in \mathbb{Z}_p^*$. We define the advantage of an algorithm $\mathcal{A}$ in solving SDH problem as:*

$$AdvSDH \stackrel{\text{def}}{=} Pr\Big[\mathcal{A}\ \Big(g_1, g_2,\ {g_2}^{\gamma}, {g_2}^{\gamma^2}, \ldots,\ {g_2}^{\gamma^q}\Big) = \Big(x, {g_1}^{\frac{1}{x+\gamma}}\Big)\ : x \in \mathbb{Z}_p^*\Big].$$

5) Co-Diffie-Hellman (co-CDH) problem [26]

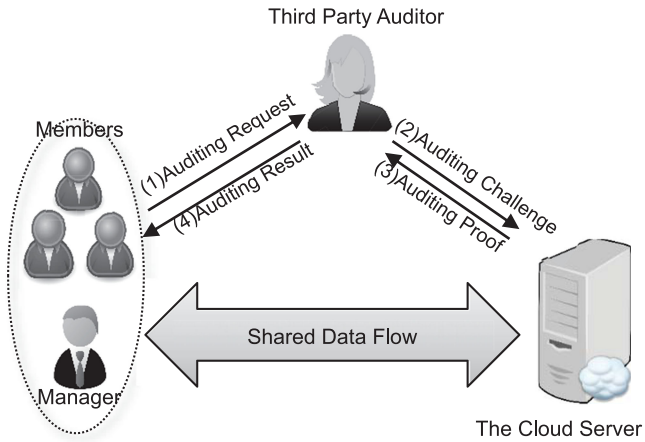


Fig. 1. System model.

Definition 5. *Computational co-Diffie-Hellman (co-CDH) on $(\mathbb{G}_1, \mathbb{G}_2)$: Given $g_2, {g_2}^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$ as input, compute $h^a \in \mathbb{G}_1$. We define the advantage of an algorithm $\mathcal{A}$ in solving the co-CDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ as*

$$Adv\ co\text{-}CDH \stackrel{\text{def}}{=} Pr\Big[\mathcal{A}\ (g_2, {g_2}^a, h) = h^a : a \in \mathbb{Z}_p^*, h \in \mathbb{G}_{\mathbf{1}}\Big].$$

Definition 6. *Two order p groups $(\mathbb{G}_1, \mathbb{G}_2)$ are (τ, t, ϵ)-bilinear group pair if they satisfy the following properties: (1) The map ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ can be computed in time at most τ. (2) A order-p group $\mathbb{G}_T$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ exsist, an e is computable in time at most τ. (3) No algorithm (t, ε)-breaks co-CDH on $(\mathbb{G}_1, \mathbb{G}_2)$.*

3 MODEL AND DEFINITIONS

In this section, we describe system model, definitions of algorithm, security model and Service-based application.

3.1 System Model

The system model is illustrated in Fig. 1. It contains three parties: a Third-Party Auditor (TPA) (also called public verifier), the Cloud Server Provider (CSP) including the storage server and the proxy server, and a group of members including group manager and ordinary members. The group manager is responsible for issuing signing key for new members. After registering to group manager, the members are able to sign the message and share data with cloud servers in an anonymous manner. To be specific, the manager is capable of revealing the identity of group member when the member violates the rules. However, the manager is incapable of signing the message on behalf of any other members. When signing the message, a member adopts Zero-Knowledge Proof of Knowledge (ZKPK) protocol for SDH* tuple to generate verification metadata and sends it with message to the cloud. Every member of the group is allowed to verify the correctness of data stored in cloud. A public verifier is specific to verify the correctness of data on behalf of members without knowing the signer's identity.

In order to ensure the public auditing, a public verifier randomly selects a number of blocks of the file stored on the cloud, then the cloud server generates the corresponding auditing proof on these blocks with stored metadata. After aggregating the proofs, the cloud server sends them to the

public verifier for the final judge. Given auditing proofs, the public verifier evaluates verification equation and gives back the decision for data integrity to client.

In user revocation process, the group manager firstly deletes corresponding identity from polynomial-time function, then he updates the group secret key and encrypt it with new exponential coefficient of polynomial function. Non-revoked member is able to decrypt the newest group secret key by using his own secret key. The revoked user is unable to generate a legal signature owing to the fact that he has been removed from polynomial function.

3.2 Definition of Algorithms

In this section, we improve the former dynamic group signature model that put forward in [17] to support remote data integrity checking. We separate the original GVf algorithm to Upload, Challenge, ProofGen and ProofVer algorithms. We also add revocation algorithms at last. The new model includes 9 algorithms that constitute our IPIC-DG scheme. It supports identity-preserving, dynamic membership and public integrity checking simultaneously.

Our model can be specified as a tuple GS = (Setup, Join, GSig, Upload, Challenge, ProofGen, ProofVer, Open, Revocation) of polynomial-time algorithms. Throughout, $k \in \mathbb{N}$ denotes the security parameter.

Setup is a probabilistic algorithm run by the group manager. It inputs a security parameter 1^k, and outputs public parameter $param$, group manager's issuing key ik (for issuing member's secret key) and opening key ok (for exposing misbehaved user).

Join ($param$, ik, ID_t) is a probabilistic algorithm run by the group manager and member. It inputs the issuing key ik, public parameter $param$, and member's identity ID_t, and outputs group member's secret key $sk_t = (x_t, y_t, A_t)$, polynomial function $f(x)$, group secret key K_g, the public key ρ, and EK.

GSig ($param$, sk_t, $\mathcal{M}$) is a probabilistic algorithm run by a group member t. It inputs the public parameter $param$, the group member t's signing key tuple (x_t, y_t, A_t) and blocked message $\mathcal{M}$, and outputs a set of metadata $\{\sigma_i\}_{1\le i\le n}$ and θ.

Upload ($param$, $\mathcal{M}$, $\{\sigma_i\}_{1\le i\le n}$, θ) is a deterministic algorithm run by the cloud server. It inputs public parameter $param$, the data $\mathcal{M}$, signature $\{\sigma_i\}_{1\le i\le n}$, θ, and outputs "Data accepted" or "Data denied".

Challenge ($param$, *filename*) is a randomized algorithm run by the TPA. It inputs filename and public parameter $param$, and outputs a challenge $chal$.

ProofGen ($param$, $chal$, $\{\sigma_i\}_{1\le i\le n}$) is a probabilistic algorithm run by the cloud server. It inputs public parameter $param$, and signature $\{\sigma_i\}_{1\le i\le n}$, and outputs aggregated proof P.

ProofVer ($param$, $chal$, P) is a deterministic algorithm run by the TPA. It inputs public parameter $param$, challenge $chal$, and proof P, and outputs 1 or 0 to report the judgement of file integrity.

Open ($param$, ok, σ_i) is a deterministic algorithm run by the group manager. It inputs public parameter $param$, opening key ok, and a homomorphic authenticated block σ_i, and outputs block signer's secret key A_t to reveal identity.

Revocation ($param$, $f(x)$, ID_t) is a probabilistic algorithm run by the group manager. It inputs public parameter $param$, polynomial function $f(x)$, and revoked user's identity ID_t, and outputs a new polynomial function $f'(x)$, new group secret key $K_g{}'$, its corresponding public key ρ' and EK'.

3.3 Security Model

In this section, we define the security models for IPIC-DG scheme in terms of the honest but curious TPA, the honest but curious CSP, the honest but curious group manager, revoked users, and the adversary. Firstly, the honest but curious TPA means that TPA woud verify the integrity of data stored on cloud honestly. However, TPA may curious about data owner's real identity and try to break anonymity game of Definition 8. Secondly, the honest but curious CSP always excute the protocol honestly but may try to forge the signature to prove the intact of data stored on cloud, which means break security game defined in Definition 7. CSP also curious about the real identity of data owner and try to break anonymity game of Definition 8. Thirdly, we assume the group manager is honest but curious. We believe the group manager would operate user join protocol justly but may curious about member's secret credential. But we assume the group manager never colludes with others. Fourthly, revoked user may try to upload the data after he is revoked. However, in our scheme, we do not allow the group manager collude with revoked user. Finally, we define the adversary $\mathcal{A}$ is an attacker including curious TPA, curious CSP, curious group manager, revoked users, and online intruder. The adversary $\mathcal{A}$ can moniter communication channel and collect information. (s)he may try to forge signatures on behalf of group members, that is break security game of Definition 7. Besides, (s)he may try to guess the real identity of group members to break security game of Definition 8.

1) *Unforgeability*. Our IPIC-DG scheme apply two kind of signature algorithm. The one is group signature algorithm based on ZKPK protocol for SDH* tuple, and the other is BLS signature algorithm [26]. Thus, the unforgeablility for our scheme has three meanings. Firstly, the adversarty $\mathcal{A}$ may try to construct a valid homormorphic authenticators σ_i of a block m_i. Secondly, the curious CSP may try to forge an aggregated auditing proof that can cheat the TPA without using metadata σ_i. Thirdly, the adversary $\mathcal{A}$ may try to generate a forgery proof θ on file M.

Definition 7. *Existential unforgeability under adaptive chosen-massage attacks security game is defined as follows:*

Setup: *The challenger runs Join algorithm and send adversary$\mathcal{A}$ public key PK, and keeps secret key SK.*

Queries: *Proceeding adaptively, $\mathcal{A}$ requests signatures with PK on at most q_s times of his choice $M_1, \ldots M_{q_s} \in \{0,1\}^*$. The challenge computes θ, and sends signature to $\mathcal{A}$.*

Output. *At last, $\mathcal{A}$ outputs a pair (M, θ), where M is not any of $M_1, \ldots M_{q_s}$. The adversary $\mathcal{A}$ win the game if $Verify\ (PK, M,\ \theta) =\ valid$ holds.*

We define $AdvSig_{\mathcal{A}}$ to be the probability that $\mathcal{A}$ wins in the above game.

2) *Anonimity*. The curious TPA, CSP, group members, the revoked users or online intruder may try to get

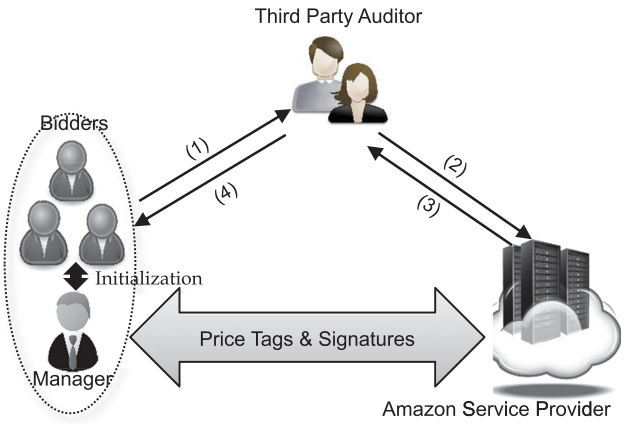


Fig. 2. E-auction system architecture. (1) Bidder auditing request, (2) TPA auditing challenge, (3) Amazon auditing proof, (4) Bidder auditing result.

the real identity of data owner. Our IPIC-DG scheme can ensure anonymity under chosen ciphertext attack in the random oracle model. Most of group signature schemes such as [19], [21], [24], [8], and [9] were CPA-full-annonymity, which means their schemes can only ensure anonymity under Chosen Plaintext Attack (CPA) in random oracle model. The adversary is not permitted to access open oracles under CPA.

The semantic security of chosen ciphertext attack (CCA) is equivalanet to Indistinguishability of Chosen Ciphertext Attack (IND-CCA) according to [32]. The cryptosystem is considered to have been broken if an adversary is able to find the two users $\mathcal{U}_0$ and $\mathcal{U}_1$ and a message m_0 such that it can distinguish between the signatures generated by $\mathcal{U}_0$ and $\mathcal{U}_1$.

We now define the chosen ciphertext attack precisely in security game which inlcudes what the adversary is capable of. A chosen plaintext attack is similar to chosen ciphertext attack, except that the adversay does not have access to open queries.

Definition 8. *IND-CCA Security Game is defined as follows:*

Setup: The challenger $\mathcal{B}$ runs the Setup algorithm to generate public parameter $param$, issuing key $ik = \gamma$, and opening key $ok = (\xi_1, \xi_2)$. The challenger $\mathcal{B}$ flips a coin d, then he gives the $param$, ik, half of opening key ξ_d to the adversary $\mathcal{A}$.

Queries: The adversary $\mathcal{A}$ can ask several queries to the challenger.

a) *Join Queries: The adversary $\mathcal{A}$ can ask for member's secret signing key sk_t, and the challenger $\mathcal{B}$ run Join algorithm to answer it.*
b) *Signing Queries: The adversary $\mathcal{A}$ queries the signature of file block m_i with user $\mathcal{U}_i$, the challenger $\mathcal{B}$ answers with the signature of m_i.*
c) *Open Queries: The adversary $\mathcal{A}$ asks for the signer's identity on a pair of (m_i, σ_i), the challenger runs the open algorithm to answers it with identity A_i. The adversary can ask as many queries as he want, as long as he does not ask the challenged one.*

Challenge. The adversary $\mathcal{A}$ sends a file block m_i and two users $\mathcal{U}_0, \mathcal{U}_1$ as a challenge. The challenger flips a coin $b \overset{R}{\leftarrow} \{0,1\}$ and generates the corresponding signature σ_b to the adversary $\mathcal{A}$.

Output. Finally, the adversary returns its guess b' to b.

We let $Adv_{anon} = \Pr[b' = b] - 1/2$. We say IPIC-DG is anonymous under chosen ciphertext attack if the function Adv_{anon} is negligible for any polynomial-time adversary $\mathcal{A}$.

3.4 Sevice-Based Application

Based on the system model and the algorithms at Section 3.2, our IPIC-DG scheme can be widely applied in the service-based cloud applications. One of example is E-auction system, which is illustrated in Fig. 2. Similar to Fig. 1, The system contains three parites: a group of bidders and a manager, amazon service provider, a Third Party Auditor (TPA). Firstly, before the first round of bidding, at the initiation phase, the manager runs Setup algorithm to determine initial parameters and public keys. Then, the manager runs Join algorithm to register all bidders and generates their secret keys. When auctions begins, the bidder runs GSig algorithm to commit price tag combined with anonymous signature. After that, according to Upload algorithm, the bidder sends price tag and anonymous signature to amazon service provider. Amazon verifys the correctness of data flows and stores them in the servers. After the auction ends, Amazon determines the winner's price tag and sent it to the manager. The manager runs Open algorithm to expose the real identity of winner by calculating with manager's secret key.

Every bidder can question the validity of price tag in the middle of auction or after the auction. A Third Party Auditor (TPA) is employed to represent bidders to audit the correctness of price tag stored in Amazon. The whole process contains three algorithms, which are the Challenge, ProofGen, and ProofVer. The TPA will faithfully perform auditing tasks and return the results whether or not the Amazon (the cloud service provider) is honest.

Every bidder can quit auction at anytime by employing Revocation Algorithm. She/he only need to inform the manager when she/he wants to leave. The manager will do the rest of jobs, which include deleting the bidder's information from polynomial function safely and updating the secret group key.

4 IPIC-DG Schemes

In this section, we propose our Identity-preserving Public Integrity Checking scheme for Dynamic Groups (IPIC-DG), which can support anonymity, strong exculpability, dynamic member operation and efficient public auditing. In general, the signature algorithm in our scheme based on ZKPK Protocol for SDH* tuple. Thus, we firstly present our ZKPK Protocol for SDH* tuple. Then we give construction of our IPIC-DG scheme in detail.

4.1 ZKPK Protocol for SDH* Tuple

In our scheme, based on [19], [33], we present new triple SDH* tuple for extending the original structure and providing strong exculpability.

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable bilinear map. g_1, g, h, k are independent generators of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, satisfying $g_1 = \psi(g_2)$. The secret key of group manager is $ik = \gamma$, which helps to generate the certificate triples (x, y, A), with $\mathrm{A} \in \mathbb{G}_1, x, y \in \mathbb{Z}_q^*$, such that $A^{x+\gamma} = g_1\, h^y$. Applying $\mathrm{e}(\cdot, g_2)$ on both sides, the triple (x, y, A) can be certificated by the equation:

$$e(A, g_2)^x e(A, w)\ e(h, g_2)^{-y} = e(g_1, g_2).$$

To prove the possession of the modified SDH (x, y, A) tuple, which we called SDH* tuple, we present the concrete ZKPK protocol as follows.

The prover randomly chooses $\alpha, \beta \in \mathbb{Z}_q^*$, and computes $T_1 = k^\alpha, T_2 = Ah^\alpha, T_3 = k^\beta, T_4 = Ag^\beta \in \mathbb{G}_1,\ \ z = x\alpha + y.$ In order to prove the knowledge of (α, β, x, z), the prover randomly chooses $r_\alpha, r_\beta, r_x, r_z \in \mathbb{Z}_q^*$, and computes $R_1 = k^{r_\alpha}$, $R_2 = e(T_2, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha} \cdot e(h, g_2)^{-r_z},\ R_3 = k^{r_\beta},\ R_4 = h^{r_\alpha}/g^{r_\beta}$. Finally, the prover sends $(T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$ to verifier.

The verifier randomly selects $c \in \mathbb{Z}_q^*$, and sends it back as a challenge to prover.

After receiving the challenge, the prover computes $s_\alpha = r_\alpha + c\alpha \bmod q, s_\beta = r_\beta + c\beta \bmod q, s_x = r_x + cx \bmod q, s_z = r_z + cz \bmod q$, and sends $(s_\alpha, s_\beta, s_x, s_z)$ to the verifier.

The verifier checks if the four equations hold: $\mathrm{k}^{s_\alpha} = R_1 \cdot {T_1}^c,\ \mathrm{k}^{s_\beta} = R_3 \cdot {T_3}^c,\ h^{s_\alpha}/g^{s_\beta} = R_4 \cdot (T_2/T_4)^c,\ e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} e(h, g_2)^{-s_z} e(T_2, w)^c / R_2 = e(g_1, g_2)^c$. If and only if the four equations hold, the verifier accepts the proofs. If one of those fails, the verifier denies the proofs and the verification process fails.

We can prove the above ZKPK Protocol for SDH* tuple is of completeness, soundness, and honest-verifier zero-knowledge property.

1) Completeness

Lemma 1. *Our ZKPK protocol is complete.*

Proof. An honest prover, who possess a valid triple (x, y, A), will be acknowledged.

$$\begin{aligned}
R_1 \cdot {T_1}^c &= k^{r_\alpha} k^{\alpha c} = k^{r_\alpha + c\alpha} = k^{s_\alpha} \\
R_3 \cdot {T_3}^c &= k^{r_\beta}\ k^{\beta c} = \mathrm{k}^{s_\beta} \\
R_4 \cdot\ (T_2/T_4)^c &= h^{r_\alpha}/g^{r_\beta}\,(Ah^\alpha/Ag^\beta)^c \\
&= h^{s_\alpha}/g^{s_\beta} e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} e(h, g_2)^{-s_z} e(T_2, w)^c / R_2 \\
&= e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_z} \cdot e(T_2, g_2)^{-r_x} \\
&\quad \cdot e(h, w)^{r_\alpha} \cdot e(h, g_2)^{r_z} e(T_2, w)^c \\
&= e(T_2, g_2)^{cx} \cdot e(h, w)^{-c\alpha} \cdot e(h, g_2)^{-cz} \cdot e(T_2, w)^c \\
&= e(Ah^\alpha, g_2)^{cx}\ e(h, {g_2}^\gamma)^{-c\alpha} e(h, g_2)^{-cz} e(Ah^\alpha, w)^c \\
&= e(A, g_2)^{cx}\ e(h, g_2)^{-c\alpha x} e(h, w)^{-c\alpha} e(h, g_2)^{-c(x\alpha + y)} \\
&\quad \cdot\ e(A, w)^c e(h, w)^{c\alpha} \\
&= (e(A, g_2)^x e(A, w) e(h, g_2)^{-y})^c \\
&= e(g_1, g_2)^c
\end{aligned}$$

These relations shows the verifier is able to confirm the rightness of $R_1, R_2,\ R_3, R_4$, as long as the prover is an honest one. □

2) Soundness

Lemma 2. *Our ZKPK protocol is sound.*

Proof. We prove that we can construct an efficient extractor for the protocol. First of all, the prover computes $T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4$. Then, for challenge value c, the prover calculates $s_\alpha, s_\beta, s_x, s_z$. Also, facing challenge value c' different from c, the prover responds with ${s_\alpha}', {s_\beta}', {s_x}', {s_z}'$. Both set of values satisfy the verification equations.

Let $\Delta c = c' - c, \Delta s_\alpha = {s'}_\alpha - s_\alpha, \Delta s_\beta = {s'}_\beta - s_\beta, \Delta s_x = {s'}_x - s_x, \Delta s_z = {s'}_z - s_z$. By dividing the two instances of verification equations, such as $\mathrm{k}^{{s_\alpha}'}/\mathrm{k}^{s_\alpha} = (R_1 \cdot {T_1}^{c'})/(R_1 \cdot {T_1}^c)$, it can be obtained that $\mathrm{k}^{\Delta s_\alpha} = {T_1}^{\Delta c}$, $\mathrm{k}^{\Delta s_\beta} = {T_3}^{\Delta c}$, $h^{\Delta s_\alpha}/g^{\Delta s_\beta} = (T_2/T_4)^{\Delta c}$, $e(T_2, g_2)^{\Delta s_x} \cdot e(h, w)^{-\Delta s_\alpha} \cdot e(h, g_2)^{-\Delta s_z} = (e(g_1, g_2)/e(T_2, w))^{\Delta c}$. The exponents are in a group of known prime order, the roots can be taken. Let $\tilde{x} = \Delta r_x/\Delta c, \tilde{\beta} = \Delta r_\beta/\Delta c,\ \tilde{\alpha} = \Delta r_\alpha/\Delta c, \tilde{z} = \Delta r_z/\Delta c$, it can be gotten that $T_1 = k^{\tilde{\alpha}},\ \ T_3 = k^{\tilde{\beta}},\ \ T_2 - T_4 = h^{\tilde{\alpha}}/g^{\tilde{\beta}}$ and $e(T_2, g_2)^{\tilde{x}} \cdot e(h, w)^{-\tilde{\alpha}} \cdot e(h, g_2)^{-\tilde{z}} = e(g_1, g_2)/e(T_2, w)$. Therefore, with $\tilde{A} = T_2 / h^{\tilde{\alpha}}$ and $\tilde{y} = \tilde{z} - \tilde{\alpha}\tilde{x}$, it can be gotten that $e(\tilde{A}, g_2)^{\tilde{x}} e(\tilde{A}, w) = e(g_1, g_2) e(h, g_2)^{\tilde{y}}$, which means that $(\tilde{A}, \tilde{x}, \tilde{y})$ is a valid certificate: $\tilde{A}^{\tilde{x}+\gamma} = g_1\ h^{\tilde{y}}$. Thus, the extractor obtains a legal tuple $(\tilde{A}, \tilde{x}, \tilde{y})$. □

3) Honest-verifier zero-knowledge property

Lemma 3. *With an honest verifier, our ZKPK protocol can be simulated under the XDH assumption.*

Proof. We prove that we can construct a polynomial-time simulator to output transcripts for our ZKPK protocol under the XDH assumption. First, we need to simulate the quadruple (T_1, T_2, T_3, T_4). To do so, we randomly choose $A \text{ in } \mathbb{G}_1, \alpha, \beta \in \mathbb{Z}_q^*$, and compute $T_1 = k^\alpha, T_2 = Ah^\alpha, T_3 = k^\beta, T_4 = Ag^\beta$. Because of the XDH assumption, this quadruple is indistinguishable from the output on prover. Then the simulator chooses a challenge $c \in \mathbb{Z}_q^*$, and responds the value $s_\alpha, s_\beta, s_x, s_z$. After that, the simulator computes the commitment values as $R_1 = \mathrm{k}^{s_\alpha} \cdot {T_1}^{-c}, R_2 = e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_z} \cdot (e(g_1, g_2)/e(T_2, w))^{-c}$, $R_3 = \mathrm{k}^{s_\beta} \cdot {T_3}^{-c}$, $R_4 = h^{s_\alpha}/g^{s_\beta} \cdot (T_2/T_4)^{-c}$. Hence, the overall transcript is indistinguishable from the distribution of our ZKPK protocol under the XDH assumption. □

4.2 IPIC-DG

In this section, we propose our IPIC-DG scheme in detail. Our IPIC-DG scheme consists of five phases, which are Initialization, Signature Generation, Public Integrity Checking, Open, and Revocation.

1) *Initialization.* Initialization phase includes two algorithms which are the Setup algorithm and Join algorithm. We also introduce the PKI environment to ensure each user $\mathcal{U}_t$ including the manager possess the secret and public key pair $(usk[t],\ upk[t])$, $(gmpk, gmsk)$.

Setup algorithm is operated by group manager. The group manager generates three order-p isomorphic multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. There exists an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. ψ is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$ where g_2 is a random generator of $\mathbb{G}_2$ and $w = {g_2}^\gamma$ with randomly choosing γ as group manger's issuing key. g_1, g, h, k are independent generators of $\mathbb{G}_1$ which satisfy $h = k^{\xi_1}, g = k^{\xi_2}$ by choosing $ok = (\xi_1, \xi_2)$ as opening key. $\mathcal{H}_1 : \{0, 1\}^* \to \mathbb{Z}_q^*$, $\mathcal{H}_2 : \{0, 1\}^* \to \mathbb{G}_1$ are two hash functions. In order to improve the efficiency of integrity verification, data $\mathcal{M}$ should be separated into n blocks, each block contains m segments. Therefore, $\mathcal{M}$ can be denoted as $\{m_{ij}, 1 \le i \le n, 1 \le j \le m, m_{ij} \in \mathbb{Z}_q^*\}$.

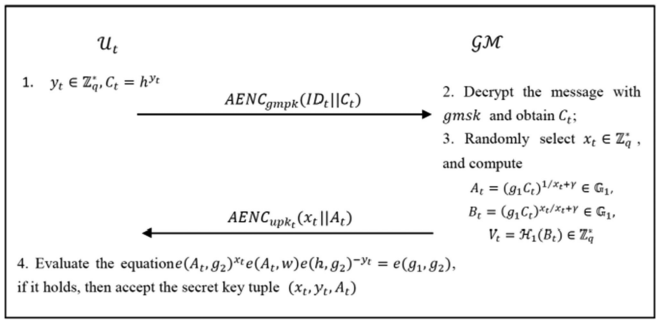


Fig. 3. Join protocol between user $\mathcal{U}_t$ and group manager $\mathcal{GM}$.

Basically, group public parameter is denoted as $param = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathcal{H}_1, \mathcal{H}_2, e, \psi, g_1, k, h, g, g_2, w; \varepsilon)$, and group manager's secret key are $gsk = (ik = \gamma$ and $ok = (\xi_1, \xi_2))$.

Join algorithm is run by the group manager and the user together. It inputs $param$ and ik, outputs user $\mathcal{U}_t$'s secret key tuple (x_t, y_t, A_t), a group key pair (K_g, ρ), and an encryption EK. The whole Join protocol is illustrated in Fig. 3. User $\mathcal{U}_t$ firstly chooses $y_t \in \mathbb{Z}_q^*$ at random, then he computes $C_t = h^{y_t}$, encrypts $(ID_t||C_t)$ with the group manager's public key $gmpk$, and sends the message $AENC_{gmpk}(ID_t||C_t)$ to the group manager. The group manager decrypts C_t with its secret key $gmsk$. Then the group manager selects random value $x_t \in \mathbb{Z}_q^*$ and computes A_t, B_t, V_t. The group manager encrypts secret key x_t and A_t with $\mathcal{U}_t$'s public key $upk[t]$, and sends the message $AENC_{upk[t]}(x_t||A_t)$ to user $\mathcal{U}_t$. After receiving the message, $\mathcal{U}_t$ decrypts the message with its secret key $usk[t]$. Finally, $\mathcal{U}_t$ obtains its secret key tuple (x_t, y_t, A_t) after passing equation $e(A_t, g_2)^{x_t} e(A_t, w) \cdot e(h, g_2)^{-y_t} = e(g_1, g_2)$. $AENC_{gmpk}$ denotes one kind of asymmetric encryption using $gmpk$ and $gmsk$ as a pair of public key and secret key.

After all users registered to the manager, the group manager creates the register table (Table 1) which includes user's ID_t and group polynomial token $\{x_t, A_t, V_t\}$. Then the group manager saves the table locally. The group manager generates the polynomial function $f(x) = \prod_{i=1}^{l}(x - V_i) = \sum_{i=0}^{l} a_i x^i \pmod q$, and the exponential function was $\{W_0, \ldots, W_l\} = \{\varepsilon^{a_0}, \ldots, \varepsilon^{a_l}\}$ by randomly selecting $\varepsilon \in \mathbb{Z}_q^*$. Then group manager picks a group secret key K_g and computes $EK = \{K_g \cdot W_0, W_1 \ldots, W_l\}$. At last, the group manager publishes the public key $\rho = g_2{}^{K_g}$ and EK.

2) *Signature Generation*. We use the idea of double encryption scheme [34] into our signature generation algorithms, to achieve anonymity under chosen ciphertext attack (not only CPA-full anonymity [24]). In Signature Generation phase, there are two algorithms which are the GSig algorithm and Upload algorithm.

In the **GSig** algorithm, it inputs user's signing key sk_t, public parameter *param*, and a message $\mathcal{M}$, and outputs signature of data. The whole process is as follows.

1. First, $\mathcal{U}_t$ randomly selects $\alpha, \beta \in \mathbb{Z}_q^*$ and computes double elgamal encryption $T_{t,1}, T_{t,2}, T_{t,3}, T_{t,4}$ of x_t, A_t as follows, $T_{t,1} = k^{\alpha}, T_{t,2} = A_t\ h^{\alpha}, T_{t,3} = k^{\beta}, T_{t,4} = A_t\ g^{\beta} \in \mathbb{G}_1$. Suppose $z = x_t\ \alpha + y_t$, then the signer proves the knowledge of (α, β, x_t, z).
2. In order to sign message $\mathcal{M}$, which is denoted as $\{m_{ij}, 1 \le i \le n, 1 \le j \le m\}$, U_t randomly picks four elements $r_{i,\alpha}, r_{i,\beta}, r_{i,x_t}, r_{i,z}$ in $\mathbb{Z}_q^*$ and computes $R_{i,1} = k^{r_\alpha}$, $R_{i,2} = e(T_{t,2}, g_2)^{r_{x_t}} \cdot e(h, w)^{-r_\alpha} \cdot e(h, g_2)^{-r_z}$, $R_{i,3} = k^{r_\beta}$, $R_{i,4} = h^{r_\alpha}/g^{r_\beta}$.
3. Given the file $\mathcal{M}$, $\mathcal{U}_t$ firstly chooses "filename" from $\mathbb{Z}_q^*$, then randomly picks $u_1, \ldots, u_m \xleftarrow{R} \mathbb{G}_1$. Let τ_0 be "$filename||n||u_1||\ldots||u_m$". For each block m_i, $\mathcal{U}_t$ computes

$$c_i = H_2\ (T_{t,1}||T_{t,2}||T_{t,3}||T_{t,4}||R_{i,1}||\ R_{i,2}||\ R_{i,3}||R_{i,4}) \cdot H_2(filename||i) \cdot \prod_{j=1}^{m} u_j{}^{m_{ij}}.$$

$1 \le i \le n, 1 \le j \le m$. After that, $\mathcal{U}_t$ computes $s_{i,\alpha} = r_{i,\alpha} + \mathcal{H}_1(c_i)\alpha \bmod q$, $s_{i,\beta} = r_{i,\beta} + \mathcal{H}_1(c_i)\beta \bmod q$, $s_{i,x} = r_{i,x_t} + \mathcal{H}_1(c_i)x_t \bmod q$, $s_{i,z} = r_{i,z} + \mathcal{H}_1(c_i) \cdot z \bmod q$. Finally, the signature of each block is $\sigma_i = (T_{t,1}, T_{t,2}, T_{t,3}, T_{t,4}, c_i, s_{i,\alpha}, s_{i,\beta}, s_{i,x}, s_{i,z})$.
4. In order to prove identity legitimacy, the signer $\mathcal{U}_t$ should obtain the group secret key K_g. Given A_t, the signer computes $B_t = A_t\ (g_1 C_t)^{x_t}$, $V_t = \mathcal{H}_1\ (B_t)$, then he obtains the secret group key as $K_g \cdot W_0 \cdot \prod_{j=1}^{l} (W_j)^{V_t{}^j} = K_g \cdot \varepsilon^{f(V_t)} = K_g$ from EK. Finally, the signer signs the file $\mathcal{M}$ with K_g as

$$\theta = (\mathcal{H}_2(\mathcal{M}))^{K_g}.$$

5. The signature of file $\mathcal{M}$ is $\{\{\sigma_i\}_{1 \le i \le n}, \theta\}$. Finally, the data owner uploads the file blocks, file tag and signatures of blocks $\{\{m_{ij}\}_{1 \le i \le n, 1 \le j \le m}, \tau_0, \{\sigma_i\}_{1 \le i \le n}, \theta\}$ to the cloud.

TABLE 1
Register Table

ID_1	A_1	x_1	V_1
ID_2	A_1	x_2	V_2
	...		
ID_l	A_l	x_l	V_l

In the *Upload* algorithm, having received data from user, the cloud server firstly verifies the user's legitimacy by checking whether the equation

$$e(\theta, g_2) = e(H_2(\mathcal{M}), \rho). \quad (1)$$

holds or not. Then, the cloud server checks the intact of data. When both user identity and data integrity are verified, the cloud server takes over the data and sends an acknowledgement to user.

3) *Public Integrity Checking*. During public integrity checking phase, the verifier delegates TPA to exam the integrity of data. We applied sampling auditing to support high-efficiency remote integrity checking while protecting identity privacy. In this phase, there are three algorithms which are the Challenge algorithm, ProofGen algorithm, and ProofVer algorithm.

In the *challenge* algorithm, having received requests from users, TPA randomly picks element $y_i \in \mathbb{Z}_q^*$,

where $i \in \mathcal{J}$, $\mathcal{J}$ is a c-elment subset, c denotes the challenge number choosen by TPA. Then it output an auditing request $chal = filename||\{(i, y_i)\}_{i\in\mathcal{J}}$ and sends it to the cloud.

In the *ProofGen* algorithm, having received the auditing request from the TPA, the cloud parses τ_0 as $(filename, n, u_1, \ldots, u_m)$ and locates the chosen blocks $\{\{m_{ij}\}, \tau_0, \{\sigma_i\}\}$ for $i \in \mathcal{J}$, computes $\mu_j = \sum_{i\in J} y_i m_{i,j} \in \mathbb{Z}_q^*$ for $j \in [1, m]$. Then the cloud server computes $R_{i,1} = k^{s_{i,\alpha}} \cdot {T_{t,1}}^{-\mathcal{H}_1(c_i)}$, $R_{i,4} = h^{s_{i,\alpha}}/g^{s_{i,\beta}} \cdot (T_{t,2}/T_{t,4})^{-\mathcal{H}_1(c_i)}$, $R_{i,2} = (e(g_1, g_2)e(T_{t,2}, w))^{-\mathcal{H}_1(c_i)} e(T_{t,2}, g_2)^{s_{i,x}} e(h, w)^{-s_{i,\alpha}} e(h, g_2)^{-s_{i,z}}$, $R_{i,3} = k^{s_{i,\beta}}\ {T_{t,\,3}}^{-\mathcal{H}_1(c_i)}$ for $i \in \mathcal{J}$. Then the cloud server computes

$$\delta = \prod_{i\in\mathcal{J}} {c_i}^{y_i} \cdot H_2\big(T_{t,1}, T_{t,2}, T_{t,3}, T_{t,4}, R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}\big)^{-y_i}$$

At last, the cloud server sends $P = (\{\mu_j\}_{j\in[1,m]}, \delta, \{u_j\}_{j\in[1,m]})$ to the TPA.

In the *ProofVer* algorithm, given an auditing proof P and challenge *chal*, the TPA checks the integrity of data by evaluating the equation

$$\delta = \prod_{l\in[1,m]} {u_l}^{\mu_l} \cdot \prod_{i\in\mathcal{J}} H_2(name||i)^{y_i}. \tag{2}$$

If (2) holds, the TPA sends "1" to user if verification success. Otherwise, the TPA sends "0" to user.

4) *Open.* In *Open* phase, there is only Open algorithm run by group manager. Given manager's secret key $ok = (\xi_1, \xi_2)$, a block signature σ_i from file $\mathcal{M}$, the group manager computes $T_{t,2}T_{t,4}/({T_{t,1}}^{\xi_1}{T_{t,3}}^{\xi_2}) = A_t$. He can decide the block signer's identity by searching for the register Table 1.
5) *Revocation.* In *Revocation* algorithm, when there is a revocation demand for expelling misbehaved user $\mathcal{U}_t$, the group manager modifies the polynomial functions as $f'(x) = f(x)/(\mathrm{x} - V_t) = \sum_{i=0}^{r-1} a_i' x^i (mod\ q)$, where the number of group member decreases to r-1. The corresponding exponential functions are changed to $\{{W_0}', \ldots, {W_{r-1}}'\} = \{{\varepsilon^{a_0}}', \ldots, {\varepsilon^{a_{r-1}}}'\}$. The group secret key K_g is changed to ${K_g}'$. The corresponding public key is $\rho' = {\varepsilon^{K_g}}'$ and the new encryption of ${K_g}'$ is $K' = \{{K_g}' \cdot {W_0}', {W_1}', \ldots, {W_{r-1}}'\}$.

5 Security Analysis

In this section, we formally prove the IPIC-DG scheme is correct, existential unforgble under adaptive chosen-message attacks, and anonymous under chosen ciphertext attacks.

5.1 Correctness

Correctness in our scheme should satisfy three requirements. Firstly, in public integrity checking phase, correct proof generated by the cloud server should be accepted by the TPA. Secondly, once user revocation finishes, the unrevoked user is able to update the secret group key with its credentials. Thirdly, an exposer is able to reveal the identity of block signer if it possesses the opening key ok.

Theorem 1. *Given any file $\mathcal{M}$, file tagτ_0 and its signature $\{\sigma_i,\ \theta\}$, a verifier is able to verify the integrity of data and non-revocability of signer.*

Proof. The verifier firstly parses file tag τ_0 to obtain file name "*filename*" and the public generators "$u_1||\ldots||u_m$". Let the file sectors are $\{m_{ij}\}$ and block authenticators are $\{\sigma_i\}$. For a prover who responds honestly to a query $filename||\{(i, y_i)\}_{i\in\mathcal{J}}$, firstly, he locates the chosen blocks and computes $R_{i,1}, R_{i,2}, R_{i,3},\ R_{i,4}$ of each block according to our ZKPK Protocol. Then the aggregated blocks and tokens are given as $\mu_l = \sum_{i\in J} y_i m_{i,l}$ and

$$\delta = \prod_{i\in\mathcal{J}} {c_i}^{y_i} \cdot H_2\big(T_{t,1}, T_{t,2}, T_{t,3}, T_{t,4}, R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}\big)^{-y_i}.$$

Finally, the verifier checks the equation (2)

$$\begin{aligned}\delta &= \prod_{i\in\mathcal{J}} \big({c_i}^{y_i} H_2\big(T_{t,1}, T_{t,2}, T_{t,3}, T_{t,4}, R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}\big)^{-y_i}\big)\\ &= \prod_{i\in\mathcal{J}} \left((H_2(name||i) \prod_{j=1}^{m} {u_j}^{m_{ij}})^{\ y_i} \right)\\ &= \prod_{i\in\mathcal{J}} H_2(name||i)^{y_i} \cdot \prod_{j\in[1,m]} {u_j}^{\sum_{i\in J} y_i m_{i,j}}\\ &= \prod_{j\in[1,m]} {u_j}^{\mu_j} \cdot \prod_{i\in\mathcal{J}} H_1(name||i)^{y_i}.\end{aligned}$$

Hence, the verification equation is satisfied.

Similarly, given file name "*filename*", public key $\rho = {g_2}^{K_g}$, the hash value $\mathcal{H}_2(filename)$, and $\mathcal{H}_1(M)$, the verifier checks the equation (1)

$$\begin{aligned}e\ (\theta, g_2) &= e\Big(\Big(\mathcal{H}_2(filename){g_1}^{\mathcal{H}_1(M)}\Big)^{\ K_g}, g_2\Big)\\ &= e\Big(H_2(filename){g_1}^{\mathcal{H}_1(M)}, {g_2}^{K_g}\Big)\\ &= e\Big(H_2(filename){g_1}^{\mathcal{H}_1(M)}, \rho\Big).\end{aligned}$$

Hence, the *non-revocability* equation is satisfied. □

Theorem 2. *Only unrevoked members are able to update the secret group key by using its credential, any revoked user is unable to do so if discrete logarithm assumption holds.*

Proof. When a user is revoked, the group manager updates the polynomial functions as $f'(x) = f'(x)/(x - V_t) = \sum_{i=0}^{r-1} a_i' x^i (mod\ q)$ and the new exponential coefficients is $\{{W_0}', \ldots, {W_{r-1}}'\} = \{{\varepsilon^{a_0}}', \ldots, {\varepsilon^{a_{r-1}}}'\}$. Having selected new group secret key ${K_g}'$, the group manager publishes $\{{K_g}' \cdot {W_0}',\ \ldots, {W_{r-1}}'\}$. As a non-revoked member, his secret key pair is (x_t, y_t, A_t), so he can compute $B_t = A_t\ (g_1 C_t)^{x_t}$, $V_t = \mathcal{H}_1\ (B_t)$, where $V_t \in \{V_i\}_{0<i<r-1}$. Let $x\ = V_t$, he can get

$$\begin{aligned}f'(x) &= {a_0}' + {a_1}' x + a_2' x^2 + \cdots + a_{r-1}\ x^{r-1} = \prod_{i=0}^{r-1} {a_i}'\ x^i\\ &= \prod_{i=0}^{r-1} (x - V_i) = \prod_{i=0}^{r-1} (V_t - \ V_i) = 0.\end{aligned}$$

For other value that not belong to $\{V_i\}_{0<i<r-1}$, the output of polynomial function is not zero.

To obtain the new group secret key $K_g{}'$, only the non-revoked member can figure out

$$K_g{}' \cdot W_0{}' \cdot \prod_{j=1}^{r-1} \left(W_j{}'\right)^{V_i{}^j} = K_g{}' \cdot \varepsilon^{f'(V_t)} = K_g{}'.$$

Therefore, it is proved that only unrevoked user can decrypt the secret group key. Suppose that there exist an adversary who can compute the group secret key $K_g{}'$, he may also find a solution to deal with DL problem for given $g_2, \rho = g_2{}^{K_g{}'}$ which contradicts the DL problem. □

Theorem 3. *Given the block signature σ_j, an exposer is able to reveal the identity of signer if and only if he possess the opening key pair $ok = (\xi_1, \xi_2)$.*

Proof. Given block signature σ_j from the block m_i, the exposer computes

$$(T_{t,2}T_{t,4})/\left(T_{t,1}{}^{\xi_1}T_{t,3}{}^{\xi_2}\right) = \frac{A_t h^{\alpha} \cdot A_t g^{\beta}}{k^{\alpha\xi_1} \cdot k^{\beta\xi_2}} = \frac{A_t\left(k^{\alpha\xi_1} \cdot k^{\beta\xi_2}\right)}{k^{(\alpha\xi_1+\beta\xi_2)}} = A_t.$$

After recovering secret key A_t, one can search the register table to find the identity of the signer. As an adversary, if he ever guesses ξ_1 or ξ_2 successfully, then he finds a solution to decrypt secret key A_t. However, at the same time, he finds the solution to DL problem. Because given h, k , computing ξ_1 is hard in DL problem. Hence, decrypting secret key A_t for an adversary is as hard as solving DL problem. □

5.2 Unforgeability

The signature of file $\mathcal{M}$ can be segmented into two parts, which are $\{\sigma_i\}_{1\le i\le n}$ and token θ. σ_i is homormorphic authenticator for each file blocks and the token θ is BLS signature on file $\mathcal{M}$. Theorem 4, 5 can formally prove the unforgeability of $\{\sigma_i\}_{1\le i\le n}$, and Theorem 6 can prove the unforgeability of θ.

Theorem 4. *It is computationally infeasible to construct a valid homormorphic authenticator σ_i for file block m_i without the knowledge of signing key tuple as long as SDH assumption holds.*

Proof. We can prove if there is an adversary $\mathcal{A}$ who can successfully forge a signature part σ_i for the file block m_i, then we can find the solution to deal with SDH problem. Based on the signature framework of forking lemma [35]), we normalize the block signature as $((m_i, T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4}), (R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}), c_i, (s_{i,\alpha}, s_{i,\beta}, s_{i,x}, s_{i,z}))$. According to the forking lemma in [9], after less than $2(1+4q_H)/\eta$ replays of the attack, with probability greater than 1/6, the adversary can produce a valid signature $((m_i, T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4}), (R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}), c_i, (s_{i,\alpha}, s_{i,\beta}, s_{i,x}, s_{i,z}))$. Then within time $\mathrm{T} \ll 9q_H T/\varepsilon$, a replay of this machine can outputs another valid signatures $((m_i, T_{i,1}, T_{i,2}, T_{i,3}, T_{i,4}), (R_{i,1}, R_{i,2}, R_{i,3}, R_{i,4}), c_i', (s_{i,\alpha}', s_{i,\beta}', s_{i,x}', s_{i,z}'))$ such that $c_i \neq c_i'$. Hence, according to Lemma 2, another valid certificate $(\tilde{A}, \tilde{x}, \tilde{y})$ can be gotten, which contradicts SDH assumption. □

Theorem 5. *In our IPIC-DG scheme, if Discrete Logarithm Assumption is hard in bilinear groups, it is computational infeasible for an untrusted cloud server to construct a forgery proof that can pass the verification check without knowing $\{m_i, \sigma_i\}$.*

Proof. If any adversary is able to pass through the verification equation without using correct and selected data blocks, then we will find a solution to DL assumption. If the cloud server wins the following security game, then it cheats the member about status of integrity of data.

Security Game: To begin with, a public verifier is delegated to verify the integrity of data stored on cloud. He generates an auditing challenge $filename||\{(j, y_j)\}_{\mathrm{j}\in\mathcal{J}}$ and sends to cloud. In fact, an honest cloud server should compute the auditing proof $(\{\mu_l\}_{l\in[1,m]},\ \delta,\ \tau_0)$ on selected blocks of file named "filename". Instead, the misbehaved cloud generates the forgery auditing proof $(\{\mu_l{}'\}_{l\in[1,m]},\ \delta,\ \tau_0)$ based on corrupted file $\mathcal{M}'$, where $\mu_l' = \sum_{j\in J} y_j m_{j,l}$, define $\Delta\mu_l = \mu_l' - \mu_l$, and at least one element of $\{\Delta\mu_l\}_{l\in[1,m]}$ is nonzero. We consider the adversary wins the security game as long as the corrupted auditing proof pass the verification equations.

Then, we show if the corrupted auditing proof can pass the verification equations, we can find the solution to DL problem simultaneously. Based on corrected auditing proof $(\{\mu_l\}_{l\in[1,m]},\ \delta,\ \tau_0)$, we obtain

$$\delta = \prod_{i\in[1,m]} u_i{}^{\mu_i} \cdot \prod_{\mathrm{k}\in\mathcal{J}} H_1(filename||k)^{y_k}.$$

Based on corrupted auditing proof $(\{\mu_l{}'\}_{l\in[1,m]},\ \delta,\ \tau_0)$, we have

$$\delta = \prod_{i\in[1,m]} u_i{}^{\mu_i{}'} \cdot \prod_{\mathrm{k}\in\mathcal{J}} H_1(filename||k)^{y_k}.$$

Then we find $\frac{\prod_{i\in[1,m]} u_i{}^{\mu_i}}{\prod_{i\in[1,m]} u_i{}^{\mu_i{}'}} = 1$, that is $\prod_{i\in[1,m]} u_i{}^{\Delta\mu_i} = 1$.

For two random element h, g in $\mathbb{G}_1$, there exists $x \in \mathbb{Z}_q^*$, such that $h = g^x$. Thus, for any u_i, it can be expressed as $u_i = g^{\delta_i}\ h^{\gamma_i}$ for $\delta_i, \gamma_i \in \mathbb{Z}_q^*$. Then we obtain

$$\prod_{i\in[1,m]} g^{\delta_i} h^{\gamma_i \Delta\mu_i} = g^{\sum_{i\in[1,m]} \delta_i\Delta\mu_i}\ \ h^{\sum_{i\in[1,m]} \gamma_i\Delta\mu_i} = 1$$

$$\mathrm{h} = g^{\frac{\sum_{i\in[1,m]} \delta_i\Delta\mu_i}{\sum_{i\in[1,m]} \gamma_i\Delta\mu_i}}, x = -\frac{\sum_{i\in[1,m]} \delta_i\Delta\mu_i}{\sum_{i\in[1,m]} \gamma_i\Delta\mu_i}.$$

Obviously, we find a solution x to DL problem, we know there is at least one $\Delta\mu_i \neq 0$ since $\mathcal{M}' \neq \mathcal{M}$. □

Theorem 6. *Suppose A is an (t, ε)-algorithm that can generate a forgery signature θ on file M. Then we find an (t', ε')-algorithm B that can solve the co-CDH problem with*

$$t' \le t + (q_H + 2q_S)c_{\mathbb{G}_1} \text{ and } \varepsilon' = \varepsilon/e(q_S+1).,$$

where $c_{\mathbb{G}_1}$ denotes one exponentiation time in $\mathbb{G}_1$, and is the base of the natural logarithm.

Proof. The unforgeability of signature θ is rely on the unforgeability of BLS signature algorithm. So, we prove the security of the signature algorithm in our scheme against existential forgery under adaptive chosen-message attacks in the random oracle model. We run the security game as follows.

Let g_2 be a generator of $\mathbb{G}_2$. Given $g_2, \rho \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$, where $\rho = g_2{}^{\pi}$, the goal of challenger B is to

output $h^{\pi}{}_{\circ}$. Firstly, challenger B gives the generator g_2 and public key $\rho {g_2}^r$ for random $r \in \mathbb{Z}_q^*$. Then the adversay A starts with query the random oracle $\mathcal{H}_2$.

Hash queries. Adversay A queries algorithm B with $M_i \in \{0,1\}^*$. Algorithm B flips a coin $c_i \in \{0,1\}$ so that the probability of $c_i = 0$ is $1/(q_s+1)$. If $c_i = 0$, the algorithm B computes $\omega_i = h \cdot \psi(g_2)^{b_i}$ for a random $b_i \in \mathbb{Z}_q^*$, otherwise, the algorithm B computes $\omega_i = \psi(g_2)^{b_i}$. Finally, the algorithm B responds to A with $\mathcal{H}_2\ (M_i) = \omega_i$.

Signature queries. Let M_i be the signature queries issued by adversay A. Algorithm B runs the Hash queries to obtain a $\omega_i \in \mathbb{G}_1$. If $c_i = 0$ then B reports failure and terminates, or it computes the signature as $\theta_i = \psi(\rho)^{b_i} \cdot \psi(g_2)^{r b_i} = {\omega_i}^{\pi + r} \in \mathbb{G}_1$. The signature θ_i is a valid signature corresponding to public key $\rho {g_2}^r$ given before. Algorithm B gives θ_i back to A.

Output. In the end, the algorithm A forges a massage-signature pair (M^*, θ^*) where M^* is never queried before. We assume θ^* is a valid signature of M^*. Then the algorithm B flips a coin c_i to generate a signature on M^* to ensure such a signature exists. If $c_i = 1$ then B fails. Otherwise, $c_i = 0$ and thus $\omega = h \cdot \psi(g_2)^b$ and $\theta = h^{\pi + r} \cdot \psi(g_2)^{r(b+\pi)}$ for a random $b \in \mathbb{Z}_q^*$. Finally, algorithm B computes $h^{\pi} = \theta / (h^{\pi} \cdot \psi(\rho)^b \cdot \psi(g_2)^{\pi b})$. Thus, we obtain the solution to co-CDH problem. The probability of that B produces the desired output is at least $\varepsilon/(e \cdot (q_S + 1))$, where

$$e = \lim_{n \to \infty} \left(1 - \frac{1}{q_S + 1}\right)^{q_S},$$

notes the probability ceiling that algorithm B answered A's signature queries consistently. $c_{\mathbb{G}_1}$ indicates one exponentiation time on $\mathbb{G}_1$, so the total running time is at most $t + (q_H + 2q_S)c_{\mathbb{G}_1}$. □

5.3 IND-CCA Anonymous

We use the security game defined in Section 3.3 to formally prove the anonymity under chosen ciphertext attack.

Theorem 7. *IPIC-DG is anonymous as long as XDH assumption holds. If there exists an adversary $\mathcal{A}$ who can break the anonymity game with advantageε within time t in the random oracle model, then one can find a(ε', t')-algorithm B that breaks XDH assmption after quering q_H times to random oracle $\mathcal{H}$ and q_S queries to signing oracle, where*

$$\varepsilon' \geq \varepsilon/4 - (q_H + q_S)/p^4$$
$$t' \leq t + 4T_{pairing} + (2 + (8+m)q_S) \cdot T_{exp}.$$

$T_{pairing}$ is the time of one pairing computation, T_{exp} denotes one time of a multi-exponentiation.

Proof. Given a tuple $(k, t = k^{\alpha}, U = k^u, V = t^v)$, if $u = v$, then it is a DDH tuple, otherwise, it is a random tuple. The simulator B chooses group manager's issuing key $ik = \gamma$, and computes $w = {g_2}^{\gamma}$. Then he flips a coin d, and defines, either $h = k^{\xi_1}, g = t (if\ d = 0)$, or $h = t, g = k^{\xi_2}\ (if\ d = 1)$. Thus, we only know half of opening key. Then the simulator gives public key $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, g_1, k, h, g, g_2, w)$ to the adversary.

Join-query. At any time, the adversary $\mathcal{A}$ can query group member U_i's certificates $(A_i, x_i,\ y_i)$.

TABLE 2
Security Comparison with Different Schemes

	Oruta [6]	XSGS [20]	Knox [9]	PS-OL [21]	Ours
Strong Exculpability	/	Yes	No	Yes	Yes
Sampling auditing	Yes	No	Yes	No	Yes
User Revocation	No	Yes	Yes	Yes	Yes
Identity-preserving	Yes	Yes	Yes	Yes	Yes
Server block access	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$
Anonymity Model	IND-CPA	IND-CCA	IND-CPA	IND-CPA	IND-CCA

Signing-queries. The adversary $\mathcal{A}$ queries the signature of file block m_i with user $\mathcal{U}_b$. The simulator B chooses independent random bits d', and will try to simulate the signature from $\mathcal{U}_b$: the ith request is answered, given a massage m_i, and certificate (x_b, y_b, A_b).

If $d = 0, T_1 = k^{\alpha}, T_2 = A_d,\ h^{\alpha}, T_3 = U, T_4 = A_b \cdot V$, for a random $\alpha \in \mathbb{Z}_q^*$.

If $d = 1, T_1 = U, T_2 = A_b\ V, T_3 = k^{\beta}, T_4 = A_{d'} \cdot g^{\beta}$, for a random $\beta \in \mathbb{Z}_q^*$.

Then simulator B computes the signature of m_i according to our Lemma 3, the probability of failure is less than $(q_H + q_S)/p^4$. If simulator B fails, he exits. Otherwise, he computes the signature $(T_1, T_2, T_3, T_4, c_i, s_{i,\alpha}, s_{i,\beta}, s_{i,x}, s_{i,z})$ and gives back to $\mathcal{A}$.

Open-queries. The adversary $\mathcal{A}$ is allowed to ask the signer of file block signature σ_i at any time except for the challenge one.

Challenge. The adversary $\mathcal{A}$ outputs a file block m_i and two users U_0, U_1 and sends the challenge to the simulator B. The simulator B randomly picks a coin $b \in \{0,1\}$ and generates the corresponding signature $\sigma_b = (T_{b,1}, T_{b,2}, T_{b,3}, T_{b,4}, s_{i,\alpha}, s_{i,\beta}, s_{i,x}, s_{i,z})$.

Output. Eventually, the adversary $\mathcal{A}$ returns its guess b′ for b. Then the simulator B takes $\mathcal{A}$'s output as his own guess to DDH tuple. If it is a DDH tuple $(\mathrm{u} = \mathrm{v})$, and $d' = b$, then it is a valid double ElGamal encryptions for A_b, the advantage of $\mathcal{A}$ in guessing b is ε. Otherwise $\mathcal{A}$ has no advantage in guessing b. Thus, the simulator B has advantage $\varepsilon/4$ in distinguishing DDH tuple or breaking XDH assumption. □

5.4 Security Comparison

We compare the security function of our scheme with the state-of-art [6], [20], [21], [9] and show our advantages in Table 2. Firstly, our scheme provides strong exculpability, which means only data owner can sign the message on behalf of himself. Knox [9] cannot prevent the group manager sign message on behalf of members. There is no group manager in Oruta [6], so we do not consider strong exculpability in Oruta. Secondly, our IPIC-DG scheme only access $O(1)$ blocks during integrity verification process while XSGS [20] and PS-OL [21] need to access $O(n)$ blocks. Thirdly, our scheme achieves efficient user revocation based on our polynomial function member management. Although XSGS [20] and PS-OL [21] claimed to support user revocation, they applied verifier-local revocation proposed in [24]. The drawback of their user revocation is that it costs $O(R)$ to prove that the user not belong to revocation lists, where R denotes the number of revocation lists. Knox [9] also suggested a way to support user revocation by updating group secret key when revocation happens. However, they failed to explain how

TABLE 3
Comparison on Signature Length per File

	Oruta [6]	XSGS [20]	Knox [9]	PS-OL [21]	Ours
Sig.Len	$dnl_{\mathbb{G}_1}$	$4l_{\mathbb{G}_1}+5l_q$	$(5l_{\mathbb{G}_1}+6l_q)n$	$3l_{\mathbb{G}_1}+5l_q$	$(5l_{\mathbb{G}_1}+4l_q)$n

normal member obtained the latest group secret key. Oruta does not support user revocation. Last but not least, as to security model, our scheme is anonymous under chosen ciphertext attack. However, [6], [20], [21], [9] only achieve anonymity under chosen plaintext attack.

6 Performance Analysis

6.1 Comparison of Schemes

Signature Length. The structure of our signature is $\{\{\sigma_i\}_{1\le i\le n},\theta\}$, where $\theta\in\mathbb{G}_1$ denotes the signature of file M,σ_i is the hormormorphic authenticators of each block. The length of θ is negligible for a large file including thousands of blocks. We compare our signature length on each file with [6], [9], [20], [21]. [6] and [9] are identity-preserving public auditing schemes. XSGS [20] and PS-OL [21] are anonymous signature schemes that do not support public auditing. Thus, the length of signature [20], [21] has nothing to do with block number n. Table 3 shows that our signature length is shorter than Knox [9]. Whats more, our scheme achieves high level of security, IND-CCA anonymity, which is better than [6], [9], [21]. As for Ourta [6], the signature length is positive linearly with number of group members, which makes it not suitable for large groups. Related Notations are presented in Table 4

Signature efficiency. As shown in Table 5, we compare the signature generation time for one file with Oruta [6] and Knox [9]. Related Notations are presented in Table 4. As can be seen, the signature generation time of Oruta is determined by the number of group members, block numbers per file and segment numbers per block. The signature generation time of Knox and ours are determined by the block numbers per file and segment numbers per block. In deed, our signature generation algorithm is the most efficient one comparing to [6] and [9].

Computaion costs for PIV. We compare the computation costs for Public Integrity Verification (PIV) process on CSP and TPA sides separately with Ortua [6] and Knox [9]. The cloud server is responsible for generating proofs for chosen files. In our IPIC-DG scheme, the proof generation costs $(c+3)\mathrm{MulExp}_{\mathbb{G}_1}+c\mathrm{MulExp}_{\mathbb{G}_T}+2\mathrm{Pair}$. Our computation costs for PIV on cloud server side is a little bit more than Knox [9], but far less than Oruta [6]. The reason is that we aggregate as many as proofs to one elements to reduce communication costs and computation cost on TPA side. During public integrity verification process, when TPA receives proofs from the cloud server, they implement ProofVer algorithm. Our IPIC-DG scheme costs $(\mathrm{c}+m)$ $\mathrm{MulExp}_{\mathbb{G}_1}$ for TPA to veifify the correctness of proof. Therefore, our scheme has the minimum amount of computation comparing to Oruta [6] and Knox [9].

TABLE 4
Notations

Symbol	Quality
n	Number of blocks per file
m	Number of segments per block
num	Number of files stored on cloud server
d	Number of group members
$l_{\mathbb{G}_1}$	Length of elements on $\mathbb{G}_1$
$l_{\mathbb{G}_T}$	Length of elements on $\mathbb{G}_T$
l_q	Length of elements on $\mathbb{Z}_q^*$
l_s	Length of challenge index
$\mathbf{MulExp}_{\mathbb{G}_1}$	One multi-exponentiation in $\mathbb{G}_1$
$\mathbf{MulExp}_{\mathbb{G}_T}$	One multi-exponentiation in $\mathbb{G}_T$
$\mathbf{MulExp}_{\mathbb{Z}_q^*}$	One multi-exponentiation in $\mathbb{Z}_q^*$
c	Challenge number
Pair	One paring operation $e:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$

Communication costs during PIV. We compare the communication costs during public integrity verification process with Oruta [6] and Knox [9] in Table 5. The communication cost during PIV contains two parts, on the one hand, the TPA sends challenge $chal$ to the cloud, on the other hand, the cloud server responds with aggregated proofs. The challenge sent by TPA is $chal=\{(i,y_i)\}_{i\in\mathcal{J}}$. It costs cl_q+cl_s in all three schemes. Unlike challenge, the proof varies with different schemes. Our IPIC-DG scheme only costs $ml_q+l_{name}+(m+1)l_{\mathbb{G}_1}$ for transmitting proof to TPA. Therefore, it was the smallest one comparing to Oruta [6] and Knox [9].

Revocation Efficiency. We analyze the revocation efficiency with the state of art [6], [9], [20], [21]in Table 6. Oruta [6] can not support user revocation ability owing to the decentralized signature algorithm (ring signature) it applied. We assume there exists proxy server employed by cloud server suggested in Panda [36]. It can re-generate signatures stored in cloud server on behalf of users. The computation costs of Knox [9] for re-generate signatures on cloud server is

TABLE 5
Comparison of Computation and Communication Overhead

		Oruta [6]	Knox [9]	Ours
Signature efficiency		$n(d-1)(m+d)\mathrm{Exp}_{\mathrm{G}_1}$	$8n\mathrm{MulExp}_{\mathrm{G}_1}+n\mathrm{MulExp}_{\mathrm{G}_T}+nm\mathrm{Exp}_{Z_p}+3n\mathrm{Pair}$	$((4+m)n+4)*\mathrm{MulExp}_{\mathbb{G}_1}+n\mathrm{MulExp}_{\mathbb{G}_T}+3\mathrm{Pair}$
Computation costs for PIV	The cloud server	$(m+dc)\mathrm{Exp}_{\mathrm{G}_1}$	$c\mathrm{MulExp}_{\mathbb{G}_1}$	$(c+3)\mathrm{MulExp}_{\mathbb{G}_1}+c\mathrm{MulExp}_{\mathbb{G}_T}+2\mathrm{Pair}$
	TPA	$(2m+c)\mathrm{Exp}_{\mathrm{G}_1}+(d+2)\mathrm{Pair}$	$8c\mathrm{MulExp}_{\mathbb{G}_1}+(2c+1)Exp_{\mathbb{Z}_q^*}+4\mathrm{Pair}$	$(c+m)\mathrm{MulExp}_{\mathbb{G}_1}$
Communication costs during PIV	Challenge	cl_q+cl_s	cl_q+cl_s	cl_q+cl_s
	Proof	$(m+d)l_{\mathbb{G}_1}+cl_q+ml_q$	$(6c+m)l_q+(4c+1)l_{\mathbb{G}_1}+cl_{\mathbb{G}_T}$	$ml_q+l_{name}+(m+1)l_{\mathbb{G}_1}$

TABLE 6
Revocation Efficiency

	Key update	Data update
Oruta [6]	/	/
Knox [9]	need	$O(n \cdot num)$
XSGS [20]	need	$O(num)$
PS-OL [21]	need	$O(num)$
Ours	need	no need

$O(n \cdot num)$. The revocation process of [20], [21] is similar to [19]. In these schemes, the group manager has to update public keys, members need to update secret keys and all data signed before need to re-sign by data owner itself. Hence, XSGS [20] and PS-OL [21] not only require $O(num)$ computational costs, but also consume $O(num)$ communication costs if data is stored on cloud. Our revocation process is better than the state-of-art, because we do not need to re-generate signatures that stored on cloud server. The group manager only updates group key $K_g{}'$ when user revocation happens. The member's personal credentials remain the same.

6.2 Exprimental Results

In this section, we make simulations to analyze the efficiency of IPIC-DG scheme. All simulations are conducted with C language on an Ubuntu 14.04 system with Intel Core i5-2140M processor running at 2.71 Ghz. The basic pairing algorithm is conducted through GNU Multiple Precision Arithmetic (GMP) library version 6.1.2 and Pairing Based Cryptography (PBC) library version 0.5.14. We choose type d MNT curve from PBC library. We apply d277699-175-167 curve in experiment to ensure approximately 1024-bit security in RSA cryptosystems. We set $l_q = 175, l_{\mathbb{G}_1} = 167$.

Signature Generation Efficiency. We simulate the signature time of 1 MB file and analyze the impact of n and d (the meaning of n and d refers to Table 4) on signature generation time. If the file size is set to 1 MB, then segment number per block can be expressed as $m = 1\ \text{MB}/(l_p * n)$. The results are shown in Fig. 4.

In Fig. 4a, signature time for a 1 MB file with the varying number of blocks n. When d is fixed and set to 5, the result shows our IPIC-DG signature generation efficiency is superior than that of Knox and Oruta. The reason is that we apply group signature algorithm rather than ring signature algorithm.

In Fig. 4b, when n is fixed and set to 250, signature generation time is linearly increasing with d in Oruta. However, the signature generation time of Knox and ours are constant and smaller than Oruta. The reason is our signature generation algorithm takes advantadge of group signature algorithm instead of ring signature algorithm.

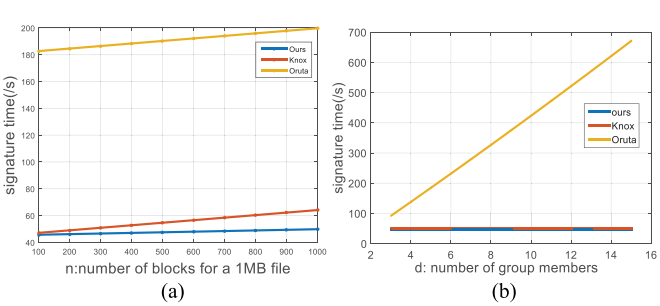


Fig. 4. Comparison of signature generation time for a 1 MB file. (a) Impact of n on signature time(s). (b) Impact of d on signature generation time (s).

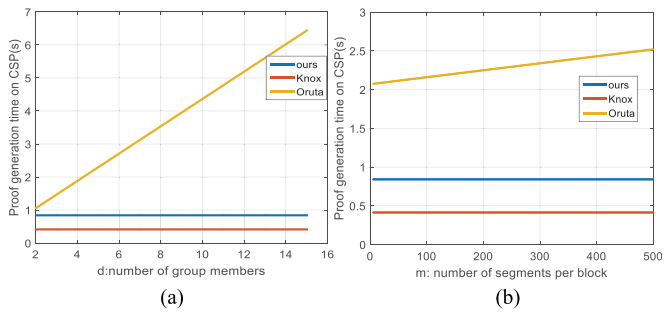


Fig. 5. Comparison of proof generation time on CSP side. (a) Impact of d on proof generation time. (b) Impact of m on proof generation time.

Computaion costs for PIV on CSP side. We simulate the proof generation time during PIV on the CSP side. We assume the number of blocks for a file is 10000, that is $n = 10000$. We set challenge number $c = 460$ to achieve detection rate at 99 percent when the cloud server has deleted 1 percent of the blocks [37]. The results are shown in Fig. 5.

In Fig. 5a, when m is fixed and set to 250, the proof generation time of Oruta [6] is linearly increasing with d. The proof generation time on the cloud server of IPIC-DG and Knox are constant. The reason is that our proof generation time is irrelevant to the size of group membership. Our proof generation time is a little bit longer than Knox [9]. The reason is that we should aggregate proofs on the cloud server to ease the computation costs on TPA and decrease the communication costs, which Knox [9] should not.

In Fig. 5b, when d is fixed and set to 5, the proof generation time of Oruta [6] is linearly increasing with m. The proof generation time on the cloud server of IPIC-DG and Knox are still constant because proof generation time of Knox and IPIC-DG are irrelevant to the way that file were segmented.

Computaion costs for PIV on TPA side. We simulate the verification time during public integrity verification process on the third part auditor (TPA) side. We also set $n = 10000$ and $c = 460$ to keep detection rate 99 percent. The results are shown in Fig. 6.

In Fig. 6a, when m is fixed and set to 250, the verification time of Oruta [6] is linearly increasing with d. The verification time of our IPIC-DG is the shortest. The reason is that the computation costs of our IPIC-DG are irrelevant to group size.

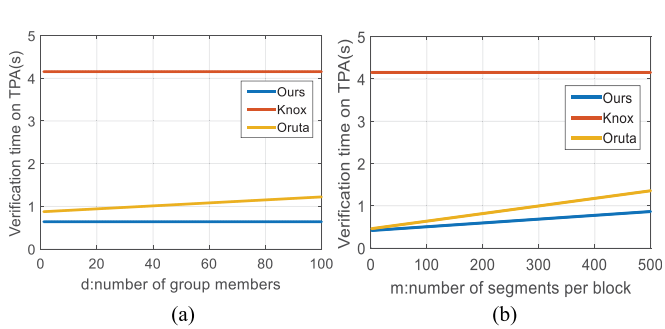


Fig. 6. Comparison of verification time on TPA side. (a) Impact of d on verification time. (b) Impact of m on verification time.

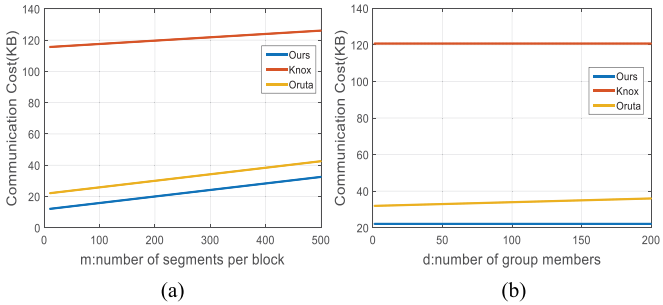


Fig. 7. Comparison of communication costs during PIV. (a) Impact of m on communication costs. (b) Impact of d on communication costs.

In Fig. 6b, when d is fixed and set to 10, the verification time of Oruta [6] and ours are linearly increasing with m. However, the verification time of our IPIC-DG is still the shortest. The reason is that we aggregate proofs on CSP side to minimize the computation costs for PIV on TPA side.

In general, when $n = 10000$ and $c = 460$, the verification time of our IPIC-DG on TPA side is less than 1 second while Knox's [9] is 4.15 second. Our scheme has great advantage in verification process on TPA side.

Communication costs during PIV. We simulate the communication costs during public integrity verification process. We set $l_s = 32, l_{name} = 128$, where l_{name} denotes the length of filename. We also set $n = 10000$ and $c = 460$ to keep detection rate 99 percent. The results are shown in Fig. 7.

In Fig. 7a, when d is fixed and set to 10, the communication costs of Oruta [6], Knox [9] and our IPIC-DG scheme are linearly increasing with m. Our IPIC-DG scheme has the least communication costs during to proof aggregation process on CSP side. The reason is that we aggregate proofs on CSP side to minimize t bhe communication costs during PIV.

In Fig. 7b, when m is fixed and set to 250, the communication costs on Oruta are linealy increasing with d. Communication costs on Knox and our IPIC-DG are irrevalent to d. Also, the communication costs of our IPIC-DG are far less than that of Knox. The reason is that the complexity of our signature algorithm is not increasing with number of group members. It is clear that our scheme is better than [6], [9] on communication overhead.

7 Conclusions

In this paper, we propose a concrete construction of Identity-preserving Public Integrity Checking scheme for Dynamic Groups (IPIC-DG). Our scheme provides identity-preserving, remote data integrity verification, and efficient user revocation, which can perfectly make up for the defects of the previous researches. Additionally, our scheme can ensure anonymity under chosen ciphertext attack which is rare in other researches. Finally, the simulation results show our scheme performs more efficient than state-of-art in signature length, public verification, and member revocation phase.

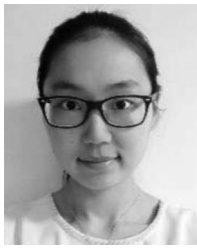
Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61372103), Key Project of Chinese National Programs for Fundamental Research and Development (973 program) (No. 2013CB338003), and the Key Laboratory Program of Information Network Security of Ministry of Public Security (No. C16612). Rui Jiang is a co-first author.

References

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptography Data Security*, 2010, pp. 136–149.
[2] K. Sakurai and S. Miyazaki, "An anonymous electronic bidding protocol based on a new convertible group signature scheme," in *Proc. Australasian Conf. Information Security Privacy*, 2000, pp. 385–399.
[3] K. Q. Nguyen and J. Traoré, "An online public auction protocol protecting bidder privacy," in *Proc. Australasian Conf. Information Security Privacy*, 2000, pp. 427–442.
[4] K. Omote and A. Miyaji, "A practical English auction with one-time registration," *Proc. Australasian Conf. Information Security Privacy*, 2001, pp. 221–234.
[5] X. Liu, Q. Xu, and J. Shang, "A public auction scheme based on gourp signature," in *Proc. 3rd Int. Conf. Information Security*, 2004, pp. 136–142.
[6] Boyang Wang, B. Li, and H. Li, "Oruta: privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan. 2014, doi: 10.1109/TCC.2014.2299807.
[7] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. 11th ACM Conf. Comput. Commun. Security*, 2004, pp. 168–177.
[8] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," *Proc. IEEE. INFOCOM*, 2010, pp. 1–9.
[9] B. Y. Wang, B. C. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. Int. Conf. Appl. Cryptography Netwo. Security*, 2012, pp. 507–525.
[10] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inform. Forensic Secur.*, vol. 10, no. 8, pp. 1717–1726, Aug. 2015, doi: 10.1109/TIFS.2015.2423264.
[11] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and R. Kotagiri, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2234–2244, Sep. 2014, doi: 10.1109/TPDS.2013.191.
[12] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inform. Forensic Secur.*, vol. 12, no. 4, pp. 767–778, Apr. 2017, doi: 10.1109/TIFS.2016.261583.
[13] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inform. Forensic Secur.*, vol. 12, no. 1, pp. 78–88, Jan. 2017, doi: 10.1109/TIFS.2016.2601070.
[14] D. Chaum and E. V. Heyst, "Group signatures," in *Proc. Workshop Theory Appl. Cryptographic Techn.*, 1991, pp. 257–265.
[15] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 416–432.
[16] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2003, pp. 614–629.
[17] M. Bellare, H. Shi, and C. Zang, "Foundations of group signatures: The case of dynamic groups," in *Proc. Cryptographers' Track RSA Conf.*, 2005, pp. 136–153.
[18] G. Atenises, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Proc. Annu. Int. Cryptology Conf.*, 2000, pp. 255–270.
[19] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. Cryptology Conf.*, 2004, pp. 41–55.
[20] C. Delerablée and D. Pointcheval, "Dynamic fully anonymous short group signatures," in *Proc. Int. Conf. Cryptology Vietnam*, 2006, pp. 193–210.
[21] J. Y. Hwang, L. Chen, H. S. Cho, and D. Nyang, "Short dynamic group signature scheme supporting controllable linkability," *IEEE Trans. Inform Forensic Secur.*, vol. 10, no. 6, pp. 1109–1124, Jun. 2015, doi: 10.1109/TIFS.2015.2390497.
[22] D. Song, "Practical forward secure group signature schemes," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2001, pp. 225–234.
[23] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. Annu. Int. Cryptology Conf.*, 2002, pp. 61–76.

[24] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. 11th ACM Conf. Comput. Commun. Security*, 2004, pp. 168–177, doi: 10.1109/TPDS.2015.2388446.
[25] Z. M. Zhu and R. Jiang, "A secure anti-collusion data sharing scheme for dynamic groups in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 40–50, Jan. 2016, doi: 10.1109/TPDS.2015.2388446.
[26] D. Boneh, B. Lynn, and H. Shacham, "Short signature from the Weil pairing," *Proc. ASIACRYPT*, Dec. 2001, pp. 514–532.
[27] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, pp. 56–73.
[28] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptology Information Security*, 2008, pp. 90–107.
[29] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attack," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1998.
[30] D. Boneh, "The decision diffie-Hellman problem," in *Proc. Int. Algorithmic Number Theory Symp.*, 1998, pp. 48–63.
[31] J. Camenish, S. Hohenberger, and A. Lysyanskaya, "Compact E-cash," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 302–321.
[32] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. 22nd Annu. ACM Symp. Theory Comput.*, 1990, pp. 427–437.
[33] L. Nguyen and R. Safavi-Naini, "Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings," in *Proc. Int. Conf. Theory Appl. Cryptology Information Security*, 2004, pp. 372–386.
[34] P. A. Fouque and D. Pointcheval, "Threshold cryptosystems secure against chosen-ciphertext attacks," in *Proc. Int. Conf. Theory Appl. Cryptology Information Security*, 2001, pp. 351–368.
[35] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, Jun. 2000.
[36] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan. 2015, doi: 10.1109/TSC.2013.2295611.
[37] 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. Security*, 2007, pp. 598–610.



Aoting Hu received the bachelor's degree in communication engeneering from the Anhui Universtiy of Technology. She is working toward the graduate degree in the School of Information Science and Engineering, Southeast University, Nanjing. Her research interests include data access control, data sharing protocol, and data integrity verification in cloud environments.

Rui Jiang received the PhD degree from Shanghai Jiaotong University, Shanghai, China, in 2005. He is now an associate professor at Southeast University, China. His current research interests include secure analysis and design of communication protocols, secure cloud computing and big data, secure network and systems communications, mobile voice end-to-end secure communications, and applied cryptography.

Bharat Bhargava is a professor of computer science at Purdue University. He is conducting research in security and privacy issues in distributed systems and sensor networks. This involves identity management, secure routing and dealing with malicious hosts, adaptability to attacks, and experimental studies. His recent work involves attack graphs for collaborative attacks. He has won five best paper awards in addition to the technical achievement award and golden core award from IEEE, and is a fellow of the IEEE.