Open-Pub: A Transparent yet Privacy-Preserving Academic Publication System based on Blockchain

Yan Zhou, Zhiguo Wan and Zhangshuang Guan

Abstract—Academic publication of latest research results are crucial to advance development of all disciplines. However, there are a number of severe disadvantages in current academic publication systems. The first problem is the misconduct during the publication process due to the opaque paper review process. An anonymous reviewer may give biased comments to a paper without being noticed or punished, because the comments are seldom published for evaluation. Secondly, the author anonymity during the paper review process is easily compromised since this information is simply open to the conference chair or the journal editor. Last but not least, access to research papers is restricted to only subscribers, and even the authors cannot access their own papers. In this paper, we propose Open-Pub, a decentralized, transparent yet privacy-preserving academic publication scheme using the blockchain technology, aiming to reduce academic misconducts and promote free sharing of research results. To this end, we design a threshold group signature to achieve anonymity for reviewers and authors. With this group signature, authors can choose to submit papers anonymously and validators take turns to distribute papers anonymously to reviewers on the blockchain according to their research interests. After the reviewers submit their review comments, the identities of reviewers and anonymous authors will be disclosed. These processes will be recorded on the blockchain so that everyone can trace the entire process. To evaluate its efficiency, we implement Open-Pub based on Ethereum source code and conduct comprehensive experiments to evaluate its performance, including computation cost and processing delay. The experiment results show that Open-Pub is highly efficient in computation and processing anonymous transactions.

Index Terms—Publication, Blockchain, Privacy, Anonymity, Threshold Group Signature

I. INTRODUCTION

In academia, publishing latest research achievements on academic publications can significantly promote advances of sciences and technologies. In current publication systems, the publication procedure roughly includes paper submission, assignment, review and the final publication. Most mainstream academic publishers work in this way, such as Elsevier, Springer, Institute of Electrical and Electronics Engineers (IEEE) and Association for Computing Machinery (ACM). Meanwhile, there are some online systems like EDAS [1] and Easychair [2] to manage submitted manuscripts for conferences and journals.

Current publication systems have a number of serious problems, which should be addressed for the benefit of the whole community. The first problem is academic misconducts due to the opaque reviewing process. Normally, a paper is given to several reviewers to examine its contributions and novelties objectively. Unfortunately, reviewers are likely to give biased comments that are not solely based on research merits for multiple reasons, e.g. intense competitions among researchers, conflict of interest and personal preferences.

The current review process also lacks mechanisms to motivate reviewers to provide constructive and unbiased comments. A reviewer is seldom rewarded for his/her valuable comments that help to improve the reviewed manuscript, so he/she will not try to provide the most constructive comments for the authors. On the other hand, a reviewer’s comments are normally not open to the public, so the reviewer is not accused even if he/she provide totally wrong comments. Clearly, making the paper review process transparent can certainly alleviate this problem.

The second problem is anonymity during the review phase. In the current paper review process, many journals and conferences adopt the double-blind or single-blind approach. For the double-blind review, the reviewers and the author do not know each other, while in the single-blind review the reviewers know the authors. Unfortunately, the anonymity of the review process can be easily compromised, causing biased results. It will be beneficial if the review process can effectively preserve anonymity for reviewers and authors.

In addition, it is important to share latest research achievements in academia. Currently, many preprint systems publish and share research results in different disciplines without peer review, e.g. arXiv [3], bioRxiv [4] and IACR eprint [5]. However, research results on these preprint systems may be problematic because they are not reviewed by peers. On the other hand, access to peer-reviewed papers published by traditional publishers is usually restricted to registered users.

The emerging blockchain [6] technology can be utilized to solve the problems in current academic publication systems. The blockchain technology is originally designed as an open, distributed ledger without any trusted party. Due to its advantages of decentralization, transparency, fault-tolerance and credibility, blockchain has been applied in many fields such as finance, insurance, notarization, healthcare, logistics, internet of things and social network.

In this paper, we propose Open-Pub, a transparent and privacy-preserving decentralized academic publication system based on the blockchain technology. To balance openness and limitation, Open-Pub is based on a consortium blockchain operated by multiple validators in a decentralized way, and the entire review process will be recorded on the blockchain.

Corresponding author: Zhiguo Wan.
Y. Zhou, Z. Wan and Z. Guan are with the School of Computer Science and Technology, Shandong University, Qingdao, 266237, China (e-mail: {yanzhoudsu@mail,xwanzhiguo}@.guanzs@.mail.isdu.edu.cn).
This work is supported by National Natural Science Foundation of China undert Grant No. 61972229.
Everyone can trace the entire process from the submission of the paper to the final publication.

Open-Pub utilizes the underlying consortium blockchain to realize transparent paper review, making review comments open to everyone on the blockchain. The reviewers are rewarded according to the quality of their comments, and hence they are motivated to provide unbiased and constructive comments for authors. To achieve anonymity during the review process in Open-Pub, we develop a threshold group signature scheme TIBGS from the identity-based group signature in [7].

The contributions of this paper can be summarized as follows:

- We propose Open-Pub, a transparent and decentralized academic publication system that is based on the blockchain technology. Open-Pub is an effective solution that integrates Verifiable Secret Sharing [8]. Identity-based Group Signature, threshold signature [9] and the blockchain technique to realize anonymous paper review and publication. To the best of our knowledge, this is the first decentralized privacy-preserving academic publication system based on blockchain.
- We design TIBGS, a Threshold Identity-based Group Signature scheme used to manage keys for Open-Pub. The master private key is shared among a group of managers, instead of a single manager as [7]. By using TIBGS in Open-Pub, we can achieve anonymous peer review.
- We implement Open-Pub by modifying Ethereum [10] source code and conduct comprehensive experiments to evaluate its performance. We test the computation and communication costs for each type of operations in Open-Pub, and the result shows that Open-Pub is efficient in terms of both computation and communication.

The remainder of the paper is structured as follows. We first review research work related to blockchain privacy protection and application of blockchain in academic publication in Section II. Then we provide preliminaries on our proposal, including cryptographic building blocks in Section III. Next, we describe a threshold identity-based group signature algorithm TIBGS and analyze its security in section IV. We then present Open-Pub in detail in Section V. After that, we give a comprehensive discussion and analysis of Open-Pub in Section VI. We describe details on the implementation of Open-Pub and evaluate its performance in Section VII. Finally, concluding remarks are given in Section VIII.

II. RELATED WORK

In this section, we introduce the work related to privacy-preserving blockchain and the application of blockchain in paper publishing.

A. Privacy-Preserving Blockchains

The first blockchain system Bitcoin [6] was invented by Satoshi Nakamoto in 2008. In public blockchains, all transactions are public and can be verified by every participant. Transaction amounts and the links between transactions are publicly visible. However, privacy issues emerge as a serious problem for blockchains. As a result, a number of privacy-preserving solutions for blockchains have been proposed recently.

Monero [11] was one of the most successful privacy-preserving cryptocurrencies using ring signature [12]. The ring signature allows a member of a set to sign on behalf of the set. Unlike the group signature, there is no way to revoke the anonymity of a ring signature. Although the ring signature provides strong anonymity, there are some limitations. First, the size of a ring signature is directly proportional to the number of participants. Thus in Monero, there are 4 outputs per transaction by default. Second, its transactions (especially RingCT transactions) are very large in size, with almost thousands of bytes per transaction. It will increase the storage space for the entire blockchain records. Monero is an untraceable digital currency, with transaction details completely invisible to the public.

Another widely used privacy-preserving cryptocurrency is Zerocash [13], an anonymous cryptocurrency based on Bitcoin. Zerocash makes use of zk-SNARKs (zero-knowledge succinct non-interactive arguments of knowledge) [14] proofs and a commitment scheme to hide transaction amounts and participants. Zerocash implements the highest level of anonymity and transaction privacy protection for blockchain, but it is computationally expensive in generating transaction proofs. In addition, zk-SNARKs require a trusted setup. If the adversary is aware of the secret randomness used in the setup, the adversary can generate deceptive proofs for false statements, and the false statements are indistinguishable from true statements.

Ring signature and zk-SNARKs can provide strong privacy protection for blockchains, but they do not have the identity disclosure functionality as group signature, which is an important feature employed by Open-Pub.

B. Blockchain for Publication

A number of attempts have been made to utilize the blockchain to promote scientific publication. Novotny et al. [15] highlight the transparency of the blockchain system for academic publishing. Janowicz et al. [16] present an outline that aims to combine distributed ledger technologies and academic publishing. Leible et al. [17] introduce the adaptability, challenges and research potential between blockchain and open science. Heaven et al. [18] introduce the advantages and challenges of blockchain to scientific publishing. Duh et al. [19] present some social dilemmas occurring in academic publishing through a strategic game setting and show that building a trusted scientific community is the key to promote a publish-and-flourish culture. Mohan et al. [20] emphasize the use of blockchain to tackle academic misconduct.

Eureka [21], [22] is a blockchain-based scientific publishing platform, developed to address traditional inefficient processes, long delays, and lack of fair financial incentives in the current academic publishing industry. Eureka maps the review process to the blockchain through smart contracts and designs a token-based incentive mechanism.

PubChain [23] uses blockchain, smart contract and IPFS peer-to-peer file-sharing system to implement a decentralized open-access publication platform. PubChain utilizes the
blockchain technology to incentivize participation of authors, readers and reviewers, and carries out a simulation to study the proposed decentralized scoring system.

TIM et al. \cite{24} propose a governance framework for scientific publishing, aiming to enhance transparency, accountability, and trust in the publishing process. The goal of the framework is to create an ecosystem allowing participants to eventual self-govern and agree on how to fairly enforce the rules and norms.

Coelho et al. \cite{25} propose a system to solve incentive problems of traditional systems in science communication and publishing, and present a minimal working model to define roles, processes, and expected results of the novel system.

Unfortunately, all these works did not solve the privacy problem during the paper review process, while Open-Pub aims to tackle this challenge for blockchain-based academic publication.

### III. Preliminaries

In this section, we will introduce some cryptographic techniques used in Open-Pub, including Bilinear Map, Verifiable Secret Sharing (VSS), an asymmetric encryption algorithm and two signature algorithms.

| Notation | Meaning |
|----------|---------|
| λ       | security parameter |
| \( H() \) | a hash function |
| \( G() \) | a function that maps a string to a integer |
| \( \text{mpk}, \text{msk} \) | master public key and private key |
| \( \text{gpiID} \) | the group identifier |
| \( \text{S} \) | a set of group managers |
| \( \text{gsk}, \text{gvk} \) | group secret key and verify key |
| \( \text{userID} \) | the user identifier |
| \( \text{usk} \) | group private key |
| \( \sigma \) | signature |
| \( (k, n) \) | threshold |

#### A. Bilinear Map and Related Assumptions

Choose a security parameter \( \lambda \), a bilinear group description, \( (q, G, \hat{G}, \hat{G}_T, e) \), can be generated by an algorithm \( \mathcal{G} \). In this description, \( e \) is used as an efficiently computable bilinear map defined on cyclic bilinear groups \( G, \hat{G} \) and \( \hat{G}_T \) of order \( q > 2^\lambda \) as \( e: G \times \hat{G} \rightarrow G_T \), which satisfies the following properties:

1. Bilinearity: \( e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} \) for all \( g \in G, \hat{g} \in \hat{G} \) and \( a, b \in \mathbb{Z}_p^* \).
2. Non-degeneracy: \( e(g, \hat{g}) \neq 1 \).

#### B. Pedersen’s Verifiable Secret Sharing

A \( (k, n) \) Pedersen’s verifiable secret sharing scheme \( \mathcal{S} \) enables \( n \) participants to share a random value \( x \) without a trusted third party, and at least \( k \) participants \( (1 \leq k \leq n) \) can participants to restore \( x \). Before \( x \) is restored, the random secret value \( x \) is kept secret from all participants. Each participant obtains a share \( x_i \) known by participant \( i \) only. More importantly, each participant can verify validity of \( x_i \), so as to detect invalid messages sent by malicious participants. Let \( P_1, P_2, \cdots, P_n \) be the \( n \) participants. The protocol for \( P_1 \) is:

1. Choose a random number \( s_{1,0} \in \mathbb{Z}_q \);
2. Distribute \( s_{i,0} \) verifiably among \( P_1, P_2, \cdots, P_n \) and \( P_3 \) can get \( s_{i,j} \);
3. Verify \( n - 1 \) received shares;
4. After receiving \( n - 1 \) correct shares, \( P_1 \) compute the share \( s_i = s_{1,1} + s_{2,1} + s_{3,1} + \cdots + s_{n,1} \). The complete secret \( s = s_{1,0} + s_{2,0} + \cdots + s_{n,0} \) is shared among \( n \) participants.

Later we will use Pedersen’s VSS scheme in our protocol, and use \( (k, n) \)-VSS to denote a Pedersen’s VSS scheme for \( (k, n) \) secret sharing.

#### C. Cryptographic Building Blocks

An asymmetric encryption \( \mathcal{E} \) scheme can be represented by a tuple of polynomial-time algorithms \( \Pi_{\mathcal{E}} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}) \).

- **Setup**\( (\lambda) \rightarrow \text{pp} \). On input a security parameter \( \lambda \), this algorithm generates a list of public parameters \( \text{pp} \).
- **KeyGen**\( (\text{pp}) \rightarrow (\text{pk}, \text{sk}) \). On input a list of public parameters \( \text{pp} \), this algorithm generates a public/private key pair \( (\text{pk}, \text{sk}) \).
- **Enc**\( (m, \text{pk}) \rightarrow c \). With a public key \( \text{pk} \), this algorithm encrypts an input plaintext \( m \) to output a ciphertext \( c \).
- **Dec**\( (c, \text{sk}) \rightarrow m \). With a secret key \( \text{sk} \), this algorithm decrypts an input ciphertext \( c \) to output a plaintext \( m \).

A signature scheme can be represented by a tuple of polynomial-time algorithms \( \Pi_s = (\text{Setup}, \text{KeyGen}, \text{Sign}, \text{Verify}) \).

- **Setup**\( (\lambda) \rightarrow \text{pp} \). On input a security parameter \( \lambda \), this algorithm generates a list of public parameters \( \text{pp} \).
- **KeyGen**\( (\text{pp}) \rightarrow (\text{pk}, \text{sk}) \). On input a list of public parameters \( \text{pp} \), this algorithm generates a public/private key pair \( (\text{pk}, \text{sk}) \).
- **Sign**\( (m, \text{sk}) \rightarrow \text{sig} \). With a private key \( \text{sk} \), this algorithm generate a signature \( \text{sig} \) corresponding to the message \( m \).
- **Verify**\( (\text{pk}, m, \text{sig}) \rightarrow \{0, 1\} \). This algorithm can verify whether the signature \( \text{sig} \) is generated by private key \( \text{sk} \) corresponding to public key \( \text{pk} \).

A \( (k, n) \) threshold signature on a message \( m \) is a single, constant-sized aggregate signature that passes verification if and only if at least \( k \) out of the \( n \) participants sign \( m \). Note that the verifier does not need to know the identities of the \( k \) signers. A \( (k, n) \) threshold signature scheme involves \( n \) participants, which can be represented by a tuple of polynomial-time algorithms \( \Pi_{ts} = \)
(Setup, ThresKeyGen, ThresSign, SigShareVer, SigShareComb, Verify).

- **Setup**(1^λ) → pp. On input a security parameter λ, this algorithm generates a list of public parameters pp.
- **ThresKeyGen**(pp, k, n) → (PK, sk_i, v_ki). On input a list of public parameters pp, this algorithm generates a public key PK, a set of n secret key shares \{sk_1, sk_2, ..., sk_n\} and a set of verification keys \{vk_1, vk_2, ..., vk_n\}.
- **ThresSign**(m, sk_i) → sig_i. Each participant signs the message m with a secret key share sk_i and outputs a signature share sig_i.
- **SigShareVer**(PK, vk_i, m, sig_i) → {0, 1}. The algorithm can verify the correctness of the signature share sig_i by PK and the corresponding vk_i.
- **SigShareComb**(sig_s, k) → sig. With at least k valid signature shares sig_i’s, this algorithm calculates the complete signature sig.
- **Verify**(PK, m, sig) → {0, 1}. The algorithm can verify the correctness of the complete signature sig by PK.

Π enc used in our scheme needs to satisfy key indistinguishability and ciphertext indistinguishability under chosen-ciphertext attack [28]. Π_s and Π_t should satisfy unforgeability and robustness against adaptive identity [29] and chosen message attacks [30].

IV. TIBGS: THRESHOLD IDENTITY-BASED GROUP SIGNATURE

In this section, we describe TIBGS, a threshold identity-based group signature algorithm. A group signature [31] scheme allows a member of a group to sign a message anonymously without leaking identity information. A group manager can open the group signature to disclose the true identity of the signer.

TIBGS involves 3 different participants: n group managers, group users and verifier. In contrast to normal group signature schemes, the number of group managers has increased from 1 to n.

In TIBGS, we use \(k, n\)-VSS to decentralize the master private key msk, the group secret key gsk and the group verify key gvk to n managers, with each manager holding only a secret shadow. Therefore, each manager can only produce a portion of the group private key usk_i for the group user through gsk and userID. With k usk_i’s corresponding to userID, a user can computes complete group private key usk, which is used to sign anonymous on behalf of the group. With a group signature σ from the group, others can not find the signer for this signature and verifier can use grpID and master public key mpk to verify the correctness of the signature. Finally, anonymous signers can be exposed by at least k group managers. As a result, TIBGS realizes decentralize cryptographic operations including key generation, signature and opening. It enables Open-Pub to manage identity-based keys and carry out cryptographic operations in a decentralized way.

A. Framework and Security of TIBGS

TIBGS is composed of 8 polynomial-time algorithms TIBGS = (Setup, GrpSetUp, ExtShare, ReconstKey, Sign, Verify, OpenPart, Open) as defined below:

- **Setup**(1^λ, k, n) → (mpk, msk_i). Each manager can run this algorithm to generate master public key mpk and master private key share msk_i.
- **GrpSetUp**(grpID, i, msk_i, k, n) → (gsk_i, gvk_i). This algorithm on input of grpID and msk_i and outputs a group secret/verify key pair (gsk_i, gvk_i) corresponding to ith manager.
- **ExtShare**(userID, gsk_i) → usk_i. The group manager executes the algorithm and outputs the group private key share usk_i, which is sent to the user.
- **ReconstKey**(userID, \{usk_i\}_{i∈S}, \{gvk_i\}_{i∈S}) → usk. The user executes the algorithm to reconstruct its full private key from the secret shares obtained from managers.
- **Sign**(m, usk) → σ. Each user can execute the algorithm and generate a signature σ corresponding to the message m.
- **Verify**(m, σ, mpk, grpID) → {0, 1}. This algorithm can verify whether the signature is generated by user in the group grpID.
- **OpenPart**(gsk_i, σ, m) → ok_i. The group manager can execute the algorithm and obtain an intermediate result ok_i.
- **Open**(k, \{ok_i\}_{i∈S}) → userID. The group manager can execute the algorithm and reveal the identifier userID of the user who produced the signature σ corresponding to the message m.

The concrete construction of TIBGS with two security experiments, the full-anonymity experiment in Fig. 11 and the full-traceability experiment in Fig. 12 in Appendix.

**Definition 1** (Full-anonymity). Let Π = (Setup, GrpSetUp, ExtShare, ReconstKey, Sign, Verify, OpenPart, Open) be a threshold identity-based group signature scheme. We say that Π is fully anonymous if for all sufficiently large security parameter k \(∈\mathbb{N}\) and any proper probabilistic polynomial time (PPT) adversary A, its advantage \(\text{Adv}_{\Pi,\mathcal{A}}^{\text{anon}}(1^λ) = |\Pr[\text{Exp}_{\Pi,\mathcal{A}}^{\text{anon}}(1^λ) = 1] - \frac{1}{2}|\) is negligible.

**Definition 2** (Full-traceability). Let Π = (Setup, GrpSetUp, ExtShare, ReconstKey, Sign, Verify, OpenPart, Open) be a threshold identity-based group signature scheme. We say that Π is fully traceable if for all sufficiently large security parameter k \(∈\mathbb{N}\) and any proper probabilistic polynomial time (PPT) adversary A, its advantage \(\text{Adv}_{\Pi,\mathcal{A}}^{\text{trace}}(1^λ) = |\Pr[\text{Exp}_{\Pi,\mathcal{A}}^{\text{trace}}(1^λ) = 1] - \frac{1}{2}|\) is negligible.

For a threshold scheme, the following robustness property is defined as follows:

**Definition 3** (Robustness). A TIBGS scheme is said to be robust if it computes a correct output even in the presence
proof. assuming that theorem 3.

theorem 2.

anonymous, the above tibgs scheme is also fully anonymous. assuming that the ibgs scheme in [7] is fully deviate from the normal execution.

the following theorem states the robustness of the proposed tibgs scheme:

**theorem 3. assuming that** \( n \geq 2k - 1 \) **where** \((k, n)\) **is the threshold of the proposed tibgs scheme, then it is robust in the presence of up to** \( k - 1 \) **corrupted managers.**

**proof.** it is easy to see that \( k \) honest group managers are required to generate a valid group private key usk, and at most \( k - 1 \) managers can be corrupted. in addition, each group manager obtains a group verification key gvk, corresponding to its secret group key share gsk.

v. open-pub: the transparent yet privacy-preserving academic publication system

in this section we first provide the system model of open-pub, and then present the threat model. after that, we describe the design of open-pub, including its transaction management and threshold identity management.
A. System Model

The system architecture of Open-Pub and the workflow are depicted in Fig. 2. The Open-Pub system involves 4 different participants with the following roles and responsibilities:

- **Validator.** Validators validate transactions and broadcast transactions in Open-Pub. In addition, validators are responsible for the distribution of user secret key, papers and rewards. After the paper review, validators collaborate to reveal the anonymous author and make the final result public. Validators are also responsible for sending rewards to authors and reviewers. All validators maintain Open-Pub to work properly.

- **Author.** In the system, author needs to request group private key shares \(\text{usk}_i\)'s from validators and computes complete group private key \(\text{usk}\). With \(\text{usk}\), author can submit papers anonymously or under real names. During the review process, the author does not know who the reviewer is until receiving reviewer comments. If the paper is accepted, the author will receive a reward from the validators.

- **Reviewer.** Before the review process, reviewers register in the blockchain based on different research areas, which helps to find suitable reviewers. After receiving a paper from the validator, the reviewer puts forward his own review opinions and scores, which are conducted in the form of sending transactions. The reviewer cannot know the real identity of the anonymous author until the validators reveal the author. After the review process, reviewers will receive the review fees.

- **Reader.** Any registered member of Open-Pub can be considered as a reader, and readers can read and comment on papers in Open-Pub.

At registration, an author can get a group private key share \(\text{usk}_i\) from each validator and calculate the complete the group private key \(\text{usk}\) with a sufficient number of shares (Step 1). Authors can choose to submit papers under their real names or anonymously, where anonymous submissions are signed using the group private key \(\text{usk}\) (Step 2).

After the validators receive the submitted papers, they take turns as distributors to select reviewers for the paper based on the research area (Step 3). In order to hide the identities of reviewers, the distributor will use the public keys of the reviewers to encrypt the identity and paper information, and send the ciphertext and deadline to the blockchain. Only the corresponding reviewer can use his/her private key to decrypt the ciphertext to get the paper information. Reviewers review and grade papers using their real names, and they do not know the real identity of the anonymous authors in the process (Step 4). The identities of the unreviewed reviewers remain confidential.

When the deadline is reached, the distributor will publish the result of the paper based on existing reviews, and multiple validators cooperate to expose the anonymous author (Step 5-6). To motivate reviewers and authors, validators will pay review fees to reward reviewers, and if the papers are accepted, the validators will pay rewards to encourage the authors (Step 7). The blockchain makes the entire review process visible to all members, including registration, submission, distribution, review, opening and reward, making the review process open and transparent.

We emphasize that how much the authors and the reviewers
should be rewarded is an independent research problem, and we leave it as the future work. These rewards are created by validators following a pre-determined rule, like the incentive mechanism in Bitcoin. Any appropriate rewarding mechanism can be used in Open-Pub, but we assume these rewards are fixed in the following description.

Open-Pub relies on threshold identity-based group signature scheme TIBGS to manage identity-based keys, where the validators perform the duties of group managers and the authors can be viewed as group members.

B. Threat Model

We assume a Byzantine threat model in which the adversary can compromise no more than 1/3 validators of blockchain. Instead of following the specified protocol, the compromised validators will act arbitrarily and may collude with each other to coordinate attacks, including injecting, modifying and dropping messages during participating in the protocol. However, the adversary is assumed to have limited computational resources and cannot break the cryptosystem used in our proposal.

Open-Pub aims to achieve the following design goals:

- **Accountability.** Whenever there is a misconduct or abuse in the system, the system should be able to identify the author or the reviewer according to the corresponding transaction.
- **Anonymity.** The identity of the anonymous author is kept secret during the review process and multiple validators collaborate to reveal the identity of the anonymous author after the review is complete. During the review process, no one except the distributor and the reviewers themselves can know which papers the reviewers have been assigned.
- **Recoverability.** When the group private key of an author is lost, it should be recoverable.

C. Open-Pub

We will describe in detail how Open-Pub utilizes the threshold identity-based group signature to implement a decentralized privacy-preserving academic publication system on blockchain. The system consists of the following 7 steps: system initialization, registration, submission, distribution, review, open and reward. To implement these steps, we create five types of transactions including txtransfer, txsend, txdistribute, txreview and txopen, and we introduce the processing logic of these transactions. We specify that the first element of the transaction structure is the public key of the sender pk_{sender} and the second is the public key of the receiver pk_{receiver}. For anonymous transactions, we uniformly set the public key of the sender to pk_{anonymity}.

1) System Initialization. To initialize the system, validators that perform the duties of group managers run TIBGS.Setup and TIBGS.Grpsetup with grpID to generate master public key mpk, master private key share msk, group secret key share gsk, and group verify key share gvk. Each validator initializes to create key pair (pk, sk), and we use pk to represent the account. All validators maintain a (k, n) threshold signature account whose public key is acc_{pub} for storing deposits, review fees and layout fees. In this step, we will stipulate the amount of deposit $s_{deposit}$, review fee $s_{review}$ and incentive fee $s_{incentive}$.

**Algorithm 1: - System Initialization**

| Input: | λ, k, n, grpID |
|-------|---------------|
| Output: | mpk, msk, gsk, gvk, sk, pk, acc_{pub}, tsk, tvk, λ |

2) Registration. The system accepts the input of the userID and creates key pair (pk, sk) for various types of users. Users can sign up for three types of accounts: reader, reviewer, and author. A type identifier is used to distinguish these accounts, with a type 0 for reader, a type 1 for reviewer, and a type 2 for author. At the same time, author accounts need to pay $s_{deposit}$ to acc_{pub} through txtransfer to prevent author from sabotaging the blockchain through anonymous transactions. After the deposit is confirmed, the validator will run TIBGS.ExtShare to generate usk, and send it to account, and the author can calculate the complete usk by TIBGS.ReconstKey. For the reviewer account, they need to register professional directions, such as machine learning, cloud computing, blockchain, etc.

**Algorithm 2: - AuthorRegistration**

| Input: | λ, userID, acc_{pub}, gsk_{i}, gvk_{i}, s_{deposit}, k |
|-------|-------------------------------------------------|
| Output: | pk, sk, txtransfer, usk |

3) Submission. In the system, only the author account can submit the paper body to the database maintained by the validators and return the hash h_{paper}. The database does not record the identity of the author. Paper submission can be divided into real-name submission and anonymous submission. A tag is used to distinguish between two types of submission, with a tag of real indicating real-name submission and a tag of anonymity indicating
anonymous submission. A real-name submit transaction includes \((pk, \text{acc}_{pub}, \text{real}, \text{field}, \text{userID}, \text{title}, \text{h}_{\text{paper}}, \text{sig})\) and an anonymous submit transaction includes \((pk_{\text{anonymity}}, \text{acc}_{pub}, \text{anonymity}, \text{field}, \text{title}, \text{h}_{\text{paper}}, \text{ssig})\). To generate an anonymous transaction, the author can run \text{TIBGS.Sign} with \text{usk} to generate a group signature \text{ssig}. Others, including validators, can verify the accuracy of \text{ssig} through \text{TIBGS.verify}. But unlike ordinary signatures, no one can discover the true identity of the signer, except that the signer is a member of the group. The recipient of \(tx_{\text{submit}}\) is \text{acc}_{pub}.

### Algorithm 3: Submission

**Input:** tag, field, userID, title, h\(_{\text{paper}}\), pk, sk, pk\(_{\text{anonymity}}\), acc\(_{pub}\), usk  
**Output:** tx\(_{\text{submit}}\)

```plaintext
1) if tag = real then
   2) tx\(_{\text{origin}}\) = (pk, acc\(_{pub}\), tag, field, userID, title, h\(_{\text{paper}}\));
   3) sig = \Pi\(_{1}\).	ext{Sign}(tx\(_{\text{origin}}\), sk);
   4) tx\(_{\text{submit}}\) = (tx\(_{\text{origin}}\), sig);

5) else if tag = anonymity then
   6) tx\(_{\text{origin}}\) = (pk\(_{\text{anonymity}}\), acc\(_{pub}\), tag, field, title, h\(_{\text{paper}}\));
   7) ssig = TIBGS.Sign(tx\(_{\text{origin}}\), usk);
   8) tx\(_{\text{submit}}\) = (tx\(_{\text{origin}}\), ssig);

9) return tx\(_{\text{submit}}\);
```

### Algorithm 4: Distribution

**Input:** h\(_{\text{submit}}\), reviewerIDs, endtime, pk, sk, acc\(_{pub}\)  
**Output:** tx\(_{\text{distribute}}\)

```plaintext
for each reviewerID in reviewerIDs do
   2) pk\(_{i}\) = GetPK(reviewerID);
   3) //GetPK() is a function to get the pk of the ID*;  
   4) sig\(_{dis}\) = \Pi\(_{1}\).	ext{Sign}(h\(_{\text{submit}}\), reviewerID, sk);
   5) c = \Pi\(_{\text{enc}}\).	ext{Enc}((sig\(_{dis}\), h\(_{\text{submit}}\), reviewerID), pk\(_{i}\));
   6) store c in set S\(_{\text{ciphertext}}\);
   7) tx\(_{\text{origin}}\) = (pk, acc\(_{pub}\), h\(_{\text{submit}}\), S\(_{\text{ciphertext}}\), endtime);
   8) sig = \Pi\(_{1}\).	ext{Sign}(tx\(_{\text{origin}}\), sk);
   9) tx\(_{\text{distribute}}\) = (tx\(_{\text{origin}}\), sig);

return tx\(_{\text{distribute}}\);
```

### Algorithm 5: Review

**Input:** reviewerID, tx\(_{\text{distribute}}\), pk, sk  
**Output:** tx\(_{\text{review}}\)

```plaintext
for each c in tx\(_{\text{distribute}}\).S\(_{\text{ciphertext}}\) do
   2) if c = \Pi\(_{\text{enc}}\).	ext{Dec}(c, sk) then
      3) sig\(_{dis}\), h\(_{\text{submit}}\), reviewerID = p;
      4) if \Pi\(_{\text{1}}\).	ext{Verify}(tx\(_{\text{distribute}}\), pk, (h\(_{\text{submit}}\), reviewerID),
         sig\(_{dis}\)) = true then
         5) break;
      6) else
         7) return;

8) find the paper through h\(_{\text{submit}}\), and review it to get comment and score;
9) pk\(_{\text{dis}}\) = tx\(_{\text{distribute}}\).pk;
10) tx\(_{\text{origin}}\) = (pk, pk\(_{\text{dis}}\), reviewerID, h\(_{\text{submit}}\), comment, score, sig\(_{dis}\), c);
11) sig = \Pi\(_{1}\).	ext{Sign}(tx\(_{\text{origin}}\), sk);
12) tx\(_{\text{review}}\) = (tx\(_{\text{origin}}\), sig);
13) return tx\(_{\text{review}}\);
```

### 5) Review

After \(tx_{\text{distribute}}\) is confirmed, reviewer can retrieve \(tx_{\text{distribute}}\) to find the corresponding ciphertext \(c\). By decrypting the ciphertext, reviewer can obtain plaintext including a signature, \(h_{\text{submit}}\) and reviewerID. Through \(h_{\text{submit}}\), the reviewer finds the paper in the database and reviews it. The reviewer will post comment and score through a review transaction including \((pk, pk_{\text{dis}}, \text{reviewerID}, h_{\text{submit}}, \text{comment}, \text{score}, \text{sig}_{\text{dis}}, c, \text{sig})\), which will be sent to the distributor. Until now, the author can know the true identity of the reviewer and the identities of reviewers who have not reviewed remain unknown. Readers can find and read the paper through \(h_{\text{submit}}\), and they can comment on the paper through the review transaction, which will be sent to \text{acc}_{pub}.

### 6) Open

After reaching the endtime of the paper, validator who distributes the paper will publish the author, the reviewers and the review result. Until then, the identity of the anonymous author has not been revealed. The distributor sends an open request and \(tx_{\text{submit}}\) to all validators, all of whom run TIBGS.OpenPart to generate \(ok_{i}\) and return it to the distributor. With at least \(k\) \(ok_{i}\)'s, the distributor runs TIBGS.Open to find the identity userID of the anonymous author. Finally, validator publishes the final result of the paper through the open transaction including \((pk, \text{acc}_{pub}, h_{\text{submit}}, \text{userID}, \text{reviewerID}, \text{score}, \text{sig}_{\text{dis}}, \text{c})\).

### 7) Reward

After the open operation, validators shall pay the reward fee \$\text{review} to reviewers, and validators shall pay the incentive fee \$\text{incentive} to author if the result is accepted. These rewards will be paid out of account \text{acc}_{pub}. If the deposit submitted by the author still exists, the deposit will be returned to the author. To decentralize power, a \(tx_{\text{transfer}}\) transaction transferred from \text{acc}_{pub} requires a threshold signature. The transaction passes verification only after at least \(k\) validators have signed the transaction.

### 8) VerTx

Validators call this algorithm to check the validity of all types of transactions and then update the state of related accounts. The algorithm outputs \(b = 1\) if \(tx\) is valid, otherwise it outputs \(b = 0\).

VI. DISCUSSION AND ANALYSIS

In this section, we first discuss details of Open-Pub, then analyze its anonymity, accountability and recoverability.
Hierarchical blockchains. In Open-Pub, we design a single blockchain for one subject, however, there are many scientific subjects, which are difficult to be implemented in the single blockchain. To solve this issue, we can create a multilevel hierarchy of blockchains for Open-Pub. For example, a 3-layer hierarchical blockchain structure for Open-Pub is as follows: first layer is a root blockchain maintained by publishers (denoted as publisher-chain), second layer contains multiple blockchains supporting different subjects respectively (denoted as subject-chain), and third layer runs multiple single blockchains designed in Open-Pub (denoted as paper-chain). These three types of blockchains form a hierarchical tree structure which each node of the tree structure is a blockchain, and all paper-chains are independent and have no influence on each other. Note that the lower-level blockchain should upload periodically blocks to the higher-level blockchain to achieve the integrity and immutability of all the blocks of the lower-level blockchain.

Malicious participants. In Open-Pub, the participants of the single blockchain contains validators (or publishers), authors, reviewers and readers, which cannot attack Open-Pub without loss of personal assets. There are some reasons to explain the above occasion: (i) the single blockchain in Open-Pub is a consortium blockchain with a secure consensus algorithm such as PBFT allowing that f validators fail at most; (ii) we require that each author should pay a deposit when registering an account, which means that a malicious author cannot withdraw his payment; (iii) if a reviewer forgers a review, he/she will lose his reputation; (iv) a reader only have the ability to read and comment papers, which does not affect the review procedure of papers. Moreover, all operations of these malicious participants are recorded on the blockchain, everyone can trace the related transactions to check the malicious operations.

A. Discussion

Public account. A public account accPub, which is maintained by all validators, is required to process deposits from authors and reward fees for reviewers and authors. To reduce the risk of accPub, we utilize threshold signature to manage this account when any funds are transferred from this account, i.e., before submitting a paper, the author should pay funds as a deposit to accPub; after reviewing a valid paper, all validators control accPub to refund the deposit to the author and distribute rewards to author and related reviewers. Note that we set the threshold for the threshold signature scheme to be the same as that of the TIBGS scheme.

Dynamic update. The group signature used in Open-Pub needs a large setup cost to create and exchange keys as described in Fig. [I]. However, the group signature does not support dynamic update of participant group, which means the participant group cannot be updated too frequently. Moreover, when the participant group changed, members (i.e., validators) of this group must update their corresponding keys to ensure...


Algorithm 8: - VerTx

\[
\text{Input: } tx, mpk, grpID, acc\text{pub}
\]
\[
\text{Output: } b
\]
\[
\begin{align*}
\text{if } tx = tx_{\text{getBalance}} \text{ then} \\
& \quad \text{if } tx.tag = \text{real} \text{ then} \\
& \quad \quad b = \Pi_s, \text{Verify}(tx.pk_{\text{sender}}, tx, tx, \text{sig}); \\
& \quad \text{else if } tx.tag = \text{anonymity} \text{ then} \\
& \quad \quad b = \text{TIBGS.Verify}(tx, tx, \text{sig}, mpk, grpID); \\
& \text{else if } tx = tx_{\text{Transfer}} \text{ then} \\
& \quad \text{if GetBalance}(tx.pk_{\text{sender}}) < tx.v \text{ then} \\
& \quad \quad \text{return } 0; \\
& \quad \text{else if } tx.pk_{\text{sender}} = \text{acc\text{pub}} \text{ then} \\
& \quad \quad b = \Pi_s, \text{Verify}(tx.pk_{\text{sender}}, tx, tx, \text{sig}); \\
& \quad \text{else} \\
& \quad \quad b = \Pi_s, \text{Verify}(tx.pk_{\text{sender}}, tx, tx, \text{sig}); \\
& \quad \text{if } b = 1 \text{ then} \\
& \quad \quad \text{set the balance of } tx.pk_{\text{sender}} \text{ to } \text{GetBalance}(tx.pk_{\text{sender}}) - tx.v; \\
& \quad \quad \text{set the balance of } tx.pk_{\text{receiver}} \text{ to } \text{GetBalance}(tx.pk_{\text{receiver}}) + tx.v; \\
\text{return } b;
\end{align*}
\]

the validity of group signature. To achieve dynamic update of the participant group, we can use epoch scheme as follows: at the beginning of each epoch, validators can vote to append new members into this group or remove malicious validators from this group. To reduce the communication cost of the distributed key generation (DKG), readers can refer [32] to obtain the DKG solution.

**Reward.** In Open-Pub, validators reward reviewers and authors for their contributions, and these rewards are generated by the blockchain. In terms of rewards, we mainly refer to the existing publication systems for improvement, but we do not differentiate reviewers in detail. In order to better motivate reviewers, it is important to distinguish between reviewers, such as more rewards for serious reviewers. The anonymity is the main goal of our system, we will investigate how to better motivate authors and reviewers using game theory as a future work.

**B. Analysis**

**Anonymity.** Open-Pub implements a double-blind review through TIBGS and asymmetric encryption. Open-Pub implements anonymous transactions through TIBGS and authors can hide their identities by publishing papers through anonymous transactions. Only when the number of validators reaches the threshold can they collectively reveal the sender of the anonymous transaction. In order to hide the identity of the reviewer, we do not send the transaction directly to the reviewer. We will first sign the identity of the reviewer and the paper information to obtain the signature, and then use the key of the reviewer to encrypt the signature, the identity of the reviewer and the paper information. Only the corresponding reviewer can decrypt the ciphertext. The identity of the reviewer and the information of the paper will be known to the author only after the reviewer has reviewed the paper.

**Accountability.** In Open-Pub, only author accounts can generate anonymous transactions. While the identity of the anonymous author will be disclosed later, anonymous transactions during this time can cause the author to send spam transactions without being detected. In addition to verifying the identity of the author, we require the author to pay a deposit upon registration. Doing evil will cause the deposit to be locked up completely.

**Recoverability.** The group private key is associated with the identity of the user, which has the advantage that the key can be recovered by reexecuting TIBGS.ExtShare and TIBGS.ReconstrKey algorithms. That is, validators whose quantity exceeds the threshold number can regenerate the group private key for the user in case of key loss.

**VII. IMPLEMENTATION AND PERFORMANCE EVALUATION**

In this section, we describe the implementation of Open-Pub, and then we present comprehensive experiment results to demonstrate its performance.

**A. Implementation**

We implement the Open-Pub system based on Ethereum source code in Golang language. Since Open-Pub is based on TIBGS, threshold signature and blockchain, our implementation mainly includes the following components:

**TIBGS.** To implement TIBGS, we use the PBC (Pairing-Based Cryptography) library which implements pairing-based cryptosystems in C language and a Go wrapper to use PBC. Based on PBC, we implement 8 algorithms of TIBGS including Setup, GrpSetUp, ExtShare, ReconstrKey, Sign, Verify, OpenPart and Open.

**Threshold signature.** We choose the threshold BLS [33] signature scheme as our threshold signature, and we implement it based on a go library [33] on github.com/dfinity-side-projects/go-dfinity-crypto.

**PBFT and Threshold.** In Open-Pub, we adopt the PBFT algorithm as the consensus mechanism and PBFT algorithm requires 3f + 1 replicas to ensure security and activity in the case of f failed nodes. We also need to set the threshold parameter of TIBGS and threshold signature, and we can set the threshold parameter as (2f + 1, 3f + 1) to match PBFT algorithm.

**Transactions.** In order to realize the function of paper review, we extend Ethereum by defining five types of transactions: tx_transfer, tx_submit, tx_distribute, tx_review, tx_open. These transactions involve three signature algorithms including ECDSA signature [34], TIBGS signature and threshold signature. tx_submit will select the TIBGS signature or ECDSA signature based on the different submission methods, tx_transfer will select the threshold signature or ECDSA signature based on the different accounts and other types of transactions will use the ECDSA signature algorithm.
B. Experiments and Performance

Table II shows the computation and communication costs of TIBGS. With the exception of some constants, the computation and communication costs are determined by the threshold and the number of validators.

TABLE II
Computation and Communication Costs of TIBGS

| Algorithm      | Computation           | Communication |
|----------------|-----------------------|---------------|
| Setup          | $2 + k \text{ Exp}$  | $O(n^2)$      |
| GrpSetUp       | $6 \text{ Exp}$      | $O(n^2)$      |
| ExtShare       | $3 \text{ Exp}$      | $O(k)$        |
| ReconstKey     | $(4k + 6) \text{ Exp} + 9 \text{ Pairing}$ | $O(k)$ |
| Sign           | $1 \text{ Pairing} + 21 \text{ Exp}$ | $-$           |
| Verify         | $3 \text{ Pairing} + 10 \text{ Exp}$ | $-$           |
| OpenPart       | $2 \text{ Pairing}$  | $-$           |
| Open           | $(2k + 1) \text{ Exp}$ | $O(k)$ |

Note: Exp denotes exponentiation. Pairing denotes bilinear pairings, $k$ denotes threshold value and $n$ is the number of validators of Open-Pub.

To evaluate the performance of Open-Pub and underlying TIBGS scheme, we deploy our system on Aliyun ecs.g6.xlarge virtual machines, each of which has 4 vCPU and 16GB memory. We run 6 docker containers on each virtual machine used to run the blockchain nodes independently and we use Python language to develop a set of automated testing tools to aid our experiments. With these tools, we can measure the performance of TIBGS scheme and assess the impact of different thresholds on block size, block consensus time, transaction latency and throughput.

We first test the performance of the TIBGS scheme. We present the performance of each algorithm of TIBGS under different $(t, n)$-threshold. We configure the threshold of Open-Pub to be $(11, 16)$, $(15, 22)$, $(21, 31)$, $(25, 37)$, $(31, 46)$, $(35, 52)$ respectively and we set Open-Pub as grpID. As showed in Fig. 3, the computation time for Setup, GrpSetUp, ReconstKey and Open increases steadily as the threshold increases, and ExtShare, Sign, Verify and OpenPart have almost fixed time costs. This is consistent with our analytical results summarized in Table II.

In the test, we divide all transactions into TXETH, TXGS and TXTS according to the type of signature, where TXETH represents ECDSA signature transaction, TXGS represents TIBGS signature transaction, and TXTS represents threshold signature transaction. The signature size and verification time of the three signature algorithms are showed in Table III. We deploy some independent nodes to simulate the user node, which can generate TXETH, TXGS or TXTS in different thresholds settings.

TABLE III
Signature Size and Verification Time

| Signature     | Signature size | Verification time |
|---------------|----------------|-------------------|
| ECDSA signature | 65 bytes     | 0.2 ms            |
| TIBGS Signature | 533 bytes   | 20.1 ms           |
| Threshold signature | 32 bytes | 0.7 ms            |

At different thresholds, we measure the block size and the block consensus time when only one type of transaction is sent. Fig. 4 shows that the block size does not change much under different thresholds. The block size of all TXGS transactions is about 90KB, for all TXTS transactions it is about 48KB, and for all TXETH transactions it is about 51KB. The difference of block size under the same threshold mainly comes from the different size of ECDSA signature, threshold signature and TIBGS signature. This indicates that the threshold does not affect the packaging process of the transaction.

Fig. 4: Block size in different thresholds settings.

Fig. 5: Block consensus time in different thresholds settings.
Fig. 5 shows that the block consensus time increases with the increase of the threshold. The block consensus time of TXGS transactions is the largest, and the block consistency time of TXTS and TXETH is close. As the threshold increases, the PBFT algorithm needs more time to consensus. The difference of the block consensus time under the same threshold is related to the verification time of TXGS, TXTS and TXETH. In order to ensure the anonymity of TXGS, the verification process of TXGS is much more complicated than that of TXTS and TXETH.

We measure the transaction confirming latency, which is the time between the transaction being issued by user and being confirmed by Open-Pub. Fig. 6 shows that the transaction confirming latency of TXGS is greater than that of TXTS and TXETH, and both increase with the increase of threshold. The increase of threshold will lead to the increase of consensus time, and naturally the transaction confirming latency will increase. The difference in transaction confirmation latency under the same threshold is due to the different validation times for the three types of signatures. Under the condition of satisfying the PBFT algorithm, we set $t$ as 21, 23, 25, 27, 29, 31 for a $(t, 31)$-threshold Open-Pub system. Fig. 7 shows that the transaction confirmation latency is growing steadily as $t$ increases. The larger $t$ increases the block consensus time.

Fig. 6: Transaction confirming latency in different thresholds settings.

Fig. 7: How $t$ with a $(t, 31)$-threshold setting influences the transaction confirming latency.

Fig. 8 shows that the TPS (transactions per second) of the three types of transactions decreases as the threshold increases and the TPS of TXETH is the maximum and TPS of TXGS is the minimum. As the block consensus time increases with the threshold and the number of transactions per block remains roughly the same, TPS naturally declines. The block consensus time of TXGS is the largest, so TPS of TXGS is the smallest.

Overall, the Open-Pub system has better performance for three types of transactions. But the system has a slight performance degradation when handling TXGS, which is the price of anonymity.

VIII. CONCLUSION

In this paper, we have presented Open-Pub, a transparent privacy-preserving academic publication system on blockchain. In Open-Pub, we design a threshold group signature TIBGS, and we use TIBGS and asymmetric encryption to protect the privacy of authors and reviewers. In addition, we improve transparency and fairness of the entire review process through blockchain. We have analyzed the performance and security of Open-Pub, and implemented Open-Pub based on Ethereum source code. Experimental results show that Open-Pub is highly efficient in dealing with anonymous transactions.

Future work can study appropriate incentive mechanisms to encourage participation of authors, readers and reviewers. Meanwhile, it may be also interesting to expand Open-Pub with more accurate metrics like impact factors for authors, reviewers, conferences and journals.

REFERENCES

[1] EDAS. https://edas.info/
[2] EasyChair. https://easychair.org/
[3] arXiv. https://arXiv.org/
[4] bioRxiv. https://biorxiv.org/
[5] IACR eprint. https://eprint.iacr.org/
[6] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system,” 2008.
[7] N. P. Smart and B. Warinschi, “Identity based group signatures from hierarchical identity-based encryption,” in International Conference on Pairing-Based Cryptography, pp. 150–170, Springer, 2009.
[8] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in Annual International Cryptology Conference, pp. 129–140, Springer, 1991.
[9] C. Cachin, “Threshold signatures for blockchain systems,” 2017.
A. Identity-based Group Signature

An ID-based group signature IBGS scheme $\Lambda$ consists of six polynomial time algorithms $(\text{Setup}, \text{GrpSetUp}, \text{Extract}, \text{Sign}, \text{Verify}, \text{Open})$:

- $\text{Setup}(1^\lambda) \rightarrow (mpk, msk)$. This algorithm generates a master public/private key pair $(mpk, msk)$.  
- $\text{GrpSetUp}(\text{grpID}, msk) \rightarrow (gsk, gvk)$. GrpID is a string that identifies the group. This algorithm on input of $\text{grpID}$ and $msk$ and outputs a group secret key $gsk$. This $gsk$ belongs to the group manager.  
- $\text{Extract}(\text{userID}, gsk) \rightarrow \text{usk}$. The group manager executes the algorithm and outputs the group private key $\text{usk}$, which is sent to the user.  
- $\text{Sign}(m, \text{usk}) \rightarrow (\sigma, m)$. Each user can execute the algorithm and generate a signature $\sigma$ corresponding to the message $m$.  
- $\text{Verify}(m, \sigma, mpk, \text{grpID}) \rightarrow \{0, 1\}$. This algorithm can verify whether the signature is generated by user in the group.  
- $\text{Open}(gsk, \sigma, m) \rightarrow \text{userID}$. The group manager can execute the algorithm and reveal the identifier userID of the user who produced the signature $\sigma$ corresponding to the message $m$.

Security Model. We recall the security model defined by Smart and Warinschi [7] for the identity-based group signature case. The security model defines two security notions, namely full-anonymity and full-traceability. Full-anonymity captures the anonymity property of the TIBGS scheme by an indistinguishability experiment between an adversary and the group signature scheme, while full-traceability captures the traceability property by an traceability experiment between an adversary and the group signature scheme.

The full-anonymity experiment for the IBGS scheme defined in [7] is defined in Fig. 9.

Definition 4 (Full-anonymity). Let $\Lambda = (\text{Setup}, \text{GrpSetUp}, \text{Extract}, \text{Sign}, \text{Verify}, \text{Open})$ be an identity-based group signature scheme. We say that $\Lambda$ is fully-anonymous if for all sufficiently large security parameter $k \in \mathbb{N}$ and any proper probabilistic polynomial time (PPT) adversary $A$, its advantage $\text{Adv}_{\Lambda, A}^{\text{anon}}(1^\lambda) = \Pr[\text{Exp}_{\Lambda, A}^{\text{anon}}(1^\lambda) = 1] - \frac{1}{2}$ is negligible.

It has been proved in [7] that the IBGS in [7] is fully anonymous. We omit the full-traceability experiment and the corresponding theorem for conciseness.

B. POK: Proof of Knowledge

The proof of knowledge and its verification [7] are used in the TIBGS.Sign and TIBGS.Verify algorithms. We set:
\[ POK_\ell = O \cdot \text{Prover} \cdot \text{Verifier} \]

The resulting proof is \( \sum \), and the rest of the parameters are public. The variable is named to help the reader understand how the proof matches the variable in TIBGS. We show the process of \( POK \) algorithm in Fig. 10.

**Prover:**
Prover generates random numbers \( k_1, k_2, k_3 \in Z_q \) and calculates:
\[
\begin{align*}
\hat{r}_1 &\gets u_{0,1} \cdot u_{1,0} \\ r_2 &\gets g^{k_2} \\ \hat{r}_3 &\gets \hat{r}_3 \cdot e_{3} \\ \tau_4 &\gets n^{k_1} \cdot g^{k_3} \\ c &\gets H(\text{grpID}|u_0|u_1|u_2|u_3|g|\hat{f}|n|g|d_0|e_1|e_2|\hat{r}_1|\hat{r}_2|\hat{r}_3|\tau_4) \\ s_1 &\gets k_1 + c \cdot x \\ s_2 &\gets k_2 + c \cdot y \\ s_3 &\gets k_3 + c \cdot z
\end{align*}
\]

The resulting proof is \( \sum \left( c, s_1, s_2, s_3 \right) \).

**Verifier:**
To verify the proof \( \sum \), verifier first computes:
\[
\begin{align*}
\hat{r}_1 &\gets u_{0,1} \cdot u_{1,0} \\ r_2 &\gets g^{k_2} \\ \hat{r}_3 &\gets \hat{r}_3 \cdot e_{3} \\ \tau_4 &\gets n^{k_1} \cdot g^{k_3} \\ c &\gets H(\text{grpID}|u_0|u_1|u_2|u_3|g|\hat{f}|n|g|d_0|e_1|e_2|\hat{r}_1|\hat{r}_2|\hat{r}_3|\tau_4)
\end{align*}
\]

and then check:
\[
c = H(\text{grpID}|u_0|u_1|u_2|u_3|g|\hat{f}|n|g|d_0|e_1|e_2|\hat{r}_1|\hat{r}_2|\hat{r}_3|\tau_4)
\]

where \( x, y \) and \( z \) are hidden, and the rest of the parameters are public. The variable is named to help the reader understand how the proof matches the variable in TIBGS. We show the process of \( POK \) algorithm in Fig. 10.

**C. Threshold Identity-based Group Signature**

The full-anonymity experiment is defined in Fig. 11. The adversary is allowed to query several oracles, \( \text{GrpSetUp}, \text{ExtShare}, \text{Sign} \) and \( \text{OpenPart} \). The adversary generates a group identity and two user identities for which it will be challenged with a signature signed by one of the users. TIBGS achieves full anonymity if the adversary fails to guess the correct user identity with non-negligible probability.

\[ \text{Exp}_{\Pi}^\text{full-anonymity}(1^\lambda) : \]
\[
\begin{align*}
&\text{(mpk, msk)} \leftarrow \text{Setup}(1^\lambda) \\
&\text{(grpID}, \text{userID}_0, \text{userID}_1, m, \text{state}) \leftarrow \text{GrpSetUp}(\text{gsk}, \text{msk}, \text{grpID}) \\
&b \xleftarrow{\$} \{0, 1\} \\
&\sigma^* \leftarrow \text{Sign}(m, \text{usk}), \text{where } ((\text{grpID}^{*}, \text{userID}_0), \text{usks}) \in \text{userIDs} \\
&b' \leftarrow \text{GrpSetUp}(\text{gsk}, \text{msk}, \text{grpID})(\sigma^*, \text{state}) \\
&\text{if } b' = b \text{ return } 1 \\
&\text{else return } 0
\end{align*}
\]

where \( \text{msk} \) is generated by:
\[
\begin{align*}
\text{msk} &\leftarrow \text{ExtShare}(\text{gsk}, \text{grpID}) \\
\text{return msk}
\end{align*}
\]

Fig. 11: The full-anonymity experiment for TIBGS. It maintains two lists: \( \text{grpIDs} \) contains all group identities with their private keys, and \( \text{userIDs} \) contains all user identities with their private keys. \( S_{\text{grpID}} \) represents the index set of the group managers.

The full-traceability experiment is defined in Fig. 12. Similar to the full-anonymity experiment, the adversary is also allowed to query several oracles, \( \text{GrpSetUp, ExtShare, Sign} \) and \( \text{OpenPart} \). The adversary generates a group identity and a signature of a message \( m \). TIBGS achieves full traceability if the signature produced by the adversary cannot be traced to one of the corrupted users with negligible probability.

**Proof:**
We reduce the full anonymity of our TIBGS scheme (denoted as \( \Pi \)) to that of the IBGS scheme (denoted as \( \lambda \)) in [7]. Suppose there is a polynomial-time adversary \( A \) can break the full anonymity of \( \Pi \), we construct another adversary \( B \) that uses \( A \) as a subroutine to break the full anonymity of \( \lambda \).

The challenger \( C \) of \( \Lambda \) executes \( \text{Setup} \) to output \( \text{mpk} \) and gives it to \( B \) as in Fig. 9. Then \( B \) passes msk to \( A \). As per
the full-anonymity experiment, \( \mathcal{A} \) makes the following oracle queries, which are answered by \( \mathcal{B} \) as follows:

- \( O_{\text{GrpSetUp}}(\text{grpID}, i) \): If \( \text{grpID} \neq \text{grpID}^* \), \( \mathcal{B} \) obtains \( gsk_i \) by querying the oracle \( O_{\text{GrpSetUp}}(\text{grpID}) \) in Fig. 9. Then \( \mathcal{B} \) computes \( gsk_i \) as well as \( gvk_i \) from \( gsk_i \) and \( i \) for \( \mathcal{A} \). Otherwise, \( \mathcal{B} \) randomly generates \( gsk_i \) for \( \mathcal{A} \).

- \( O_{\text{ExtShare}}(\text{grpID}, i, \text{userID}, \text{type}) \): If \( \text{type} = \text{corrupt} \), \( \mathcal{B} \) computes \( usk_i \) from \( gsk_i \) and \( i \). If \( \text{type} = \text{ok} \), \( \mathcal{B} \) returns \( usk_i \) to \( \mathcal{A} \) and \( \text{corrgrpIDs} \) for at most \( t - 1 \) different \( i \) for \( \mathcal{A} \). Otherwise, \( \mathcal{B} \) randomly generates \( ok_i \) for \( \mathcal{A} \).

- \( O_{\text{OpenPart}}(\text{grpID}, i, \text{sigma}, \text{m}) \): If \( \text{grpID} \neq \text{grpID}^* \), \( \mathcal{B} \) obtains \( gsk_i \) by querying the oracle \( O_{\text{GrpSetUp}}(\text{grpID}) \). Then \( \mathcal{B} \) computes \( gsk_i \) as well as \( gvk_i \) from \( gsk_i \) and \( i \). After that, \( \mathcal{B} \) executes \( \text{OpenPart}(gsk_i, \text{sigma}, \text{m}) \) and return the result to \( \mathcal{A} \). Otherwise, \( \mathcal{B} \) randomly generates \( ok_i \) for \( \mathcal{A} \).

After \( \mathcal{A} \) has made enough oracle queries, \( \mathcal{A} \) outputs \( \text{grpID}^*, \text{userID}_0, \text{userID}_1, m, \text{state} \) to \( \mathcal{B} \), who will forward the output to the challenger \( \mathcal{C} \). Then \( \mathcal{C} \) outputs a signature \( \sigma^* \) to \( \mathcal{B} \) who forwards \( \sigma^* \) to \( \mathcal{A} \) and obtains the output \( b' \) from \( \mathcal{A} \). Finally, \( \mathcal{B} \) outputs \( b' \) as its guess for \( b \) chosen by \( \mathcal{C} \).

Clearly, the adversary \( \mathcal{B} \) has the same advantage of the experiment as \( \mathcal{A} \), i.e.,

\[
\text{Adv}_{\mathcal{A},\mathcal{B} }^\text{anon}(1^\lambda) = \text{Adv}_{\mathcal{A},\mathcal{C} }^\text{anon}(1^\lambda).
\]

Since no such adversary \( \mathcal{B} \) can break full anonymity of \( \Lambda \), we conclude that \( \mathcal{A} \) cannot break full anonymity of \( \Pi \).