IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Distributed signing protocol for IEEE P1363-compliant identity-based signature scheme

ISSN 1751-8709
Received on 21st October 2019
Revised 31st January 2020
Accepted on 10th February 2020
doi: 10.1049/iet-ifs.2019.0559
www.ietdl.org

Qi Feng[1,2], Debiao He[1,3] ✉, Zhe Liu[4], Ding Wang[5], Kim-Kwang Raymond Choo[6]

[1]*School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, People's Republic of China*
[2]*State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, People's Republic of China*
[3]*Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, People's Republic of China*
[4]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, People's Republic of China*
[5]*School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, People's Republic of China*
[6]*Cyber Security and Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, USA*
✉ *E-mail: hedebiao@163.com*

Abstract: The identity-based signature (IBS) scheme, with the fast, extend of economic globalisation, is becoming the most promising secure primitive for electronic commerce applications. Due to its importance, the problem of ID-based signing in a multi-party setting, *without ever revealing any private and secret information*, has received considerable interest in distributed applications such as a global manufacturer. However, there is no practical solution for more than two parties. Therefore, in this study, the authors present the first distributed identity-based signing protocol for the global electronic commerce system. Specifically, the authors' designed protocol allows a group of parties to generate the signature in a decentralised and fair manner. They also prove that their proposed protocol is secure against a malicious adversary under the discrete logarithm and decisional Diffie–Hellman assumptions. Moreover, they implement the protocol using the MIRACL libraries on physical computing devices. Findings from the evaluations demonstrate the practical utility of their proposed protocol, in terms of achieving high level of security within a reasonable time framework (e.g. signing time (including communication latency and waiting delay) takes 311.86 ms for three parties, 558.2 ms for five parties, and 707.21 ms for seven parties, under a single-thread implementation).

Nomenclature

τ	number of participants
$\mathscr{P}_1, \ldots, \mathscr{P}_\tau$	participants of our protocol
ID	common identity of participants
q	large prime numbers
$\mathbb{G}_1, \mathbb{G}_2$	two additive cyclic groups with order q
Q_1, Q_2	generators of $\mathbb{G}_1, \mathbb{G}_2$
$\mathbb{G}_T$	multiplicative cyclic group with order q
$D_{\mathrm{ID}}^{(1)}, \ldots, D_{\mathrm{ID}}^{(\tau)}$	partial private keys for $\mathscr{P}_1, \ldots, \mathscr{P}_\tau$
s, P_{pub}	master secret key and public key of KGC satisfying $P_{\mathrm{pub}} = s \cdot Q_2$
h_1, h_2	two cryptographic hash functions
$e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$	bilinear pairing map
$\leftarrow_R$	randomly sampling operation

1 Introduction

The wide use of electronic commerce (e-commerce) has made it more and more indispensable in our digitalised society. For example, according to the industry survey [1], e-commerce revenue is projected to double to nearly 5 trillion USD in 2021, from the reported revenue of 2.3 trillion USD in 2017. The annual growth rate for e-commerce sales in the US is reportedly nearly 10% over the past fifteen years, and in China, the total e-commerce sales are expected to double between 2017 and 2019, adding one trillion USD in three years [2]. This is not surprising, partly due to advances in artificial intelligence – AI, big data analytics, mobile cloud networking, and other underpinning technologies, and the society's acceptance of technologies. For example, Amazon's recommendation engine reportedly drives 35% of total sales [3]; Alibaba sets up a smart warehouse where over 700 robots assemble and ship orders to customers in 2018, raked in over 25 billion USD in sales [4, 5]; and so on.

Intuitively, e-commerce is here to stay and will likely evolve in the near future, for example becoming 'smarter' and requiring less human involvement (e.g. leveraging industry 5.0, 5G and other AI advances). Generally, in a smart e-commerce system, such as the example presented in Fig. 1, manufacturers can better learn from the market (in real-time) and share their resources in an upstream–downstream connection to optimise business processes, such as purchasing and other supply chain-related matters [6]. However, an origin, as well as the message transmitted through the e-commerce system, must guarantee authenticity and undeniableness as finance is tied to, i.e. any identity impersonation or message tampering may cause huge financial losses. There has been a lot of researches to address these issues, such as [7–11]. Some of them are based on a hash function for user authentication, whereas for some scenarios with a large scale, the digital signature scheme is an important content that has a special role in these problems' settlement.

Identity-based signature (IBS), standardised in IEEE P1363.3 [12], is a widely used cryptographic primitive to achieve authenticity, non-repudiation and data integrity. Unlike conventional digital signature schemes built on public key infrastructure (PKI), IBS allows a publicly known string representing an individual or organisation to be used as a public key. Thus, users could verify digital signatures using publicly available information associated with the signer's identity (e.g. organisational email address or some other unique public user identifiers). With the development of globalisation, a key challenge now is '*How can different manufacturers sign a purchase order as a common instance (e.g. different suppliers contributing their intellectual property to a product) without violating their interests.*'. Intuitively speaking, the goal is to enable multiple parties to jointly produce a digital signature without revealing their secret inputs – also referred to as multi-party computation (MPC) in the literature.

MPC, introduced by Yao in the early 1980s [13], is generally defined as a protocol in which n parties want to jointly compute a

Fig. 1 *Simple example of smart e-commerce system*

function $f(x_1, x_2, \ldots, x_n)$, where x_i is the ith party's private input. Since the seminal work of Yao, a number of MPC-based digital signature protocols/schemes with different features have been proposed, for example to enable privacy in the signing phases. Some proposed protocols are based on the secret-sharing protocol, where a threshold t is specified such that only t or more shares can sign a message correctly, such as those designed for Schnorr [14, 15] and for DSA/ECDSA [16–20]. However, such systems are generally limited to reconstruction of the secret key and the reconstructing party clearly can be targeted by attackers seeking to compromise the process. There are other variants of threshold signing protocol for DSA/ECDSA, designed to minimise such a limitation [21–23].

Another alternative approach to build a multi-party signature is based on the GMW pattern presented in [24, 25]. In such protocols, the parties hold additive shares of the secret values, and multiplication operations on them are transformed into additive shares of outputs. For ECDSA, the first two-party setting was addressed by MacKenzie and Reiter [26]. Since then, a number of two-party protocols have also been presented in the literature [27, 28], and extended multi-party version [29, 30].

However, most MPC solutions are not practical in real-world identity-based signature schemes. Recently in 2018 and 2019, He *et al.* [31] and Wu *et al.* [32], respectively, showed that a two-party signing protocol for IBS could be practically implemented, but it is unclear how to generalise their approaches to the multi-party setting. In other words, the challenge of '*How to construct a secure and practical multi-party signing protocol for identity-based signature schemes?*' remains. This is the contribution we focus on in this paper.

In this paper, we present the first secure multi-party signing protocol for identity-based signature scheme standardised in IEEE P1363, as summarised below.

1.1 Our contributions

In this paper, we present the first secure multi-party signing protocol for an identity-based signature scheme standardised in IEEE P1363. The main contributions we make are described as below:

- Firstly, we design a simple multi-party signing protocol for the IBS scheme standardised in IEEE P1363, where the signing process is executed among multiple parties in a distributed manner. The highlight of our proposed protocol is secure under the security assumptions of the discrete logarithm (DL), decisional Diffie–Hellman (DDH) and the security basis of the original IBS scheme itself.
- In service of our main protocol, we design a novel approach for multi-party scalar multiplication that may be of independent interest. To be more specific, for given two secrets $s \in \mathbb{Z}_q$ and $D \in \mathbb{G}$ shared additively (which means $s = \sum_{\ell=1}^{\tau} s_\ell \bmod q$, $D = \sum_{\ell=1}^{\tau} D_\ell$), where each party $\mathcal{P}_i$ holds s_i and D_i, our approach finally generates τ additive shares $T_1, \ldots, T_\tau \in \mathbb{G}$ such that $\sum_{\ell=1}^{\tau} T_\ell = s \cdot D = \sum_{\ell=1}^{\tau} s_\ell \cdot \sum_{\ell=1}^{\tau} D_\ell$.
- We implement our protocol in C/C++ to show its feasibility and performance in the real-world. Since that our proposed protocol does not need heavy algorithms (e.g. Paillier homomorphic encryption or Oblivious Transfer), we achieve this breakthrough with low computation and communication costs, approximately a factor of 8.91 of local signatures in timing. Concretely, three parties can jointly sign a message in around 311.86 ms.

1.2 Paper organisation

The rest of the paper is organised as follows. In Section 2, we give a brief review of related works on multi-party computation for public-key cryptography. In Section 3, we present the notations and backgrounds used in this paper. In Section 4, we present details of the proposed distributed signing protocol for identity-based signature in the IEEE P1363 standard. In Sections 5 and 6, we perform the analysis on the security and the cost of the proposed protocol to demonstrate its innovation and performance in theory. We implement our protocol and test it with some more parties over LAN in Section 7. Finally, we conclude the paper in Section 8.

2 Related work

More than a decade ago, MacKenzie and Reiter [26] presented the first two-party signing protocol for DSA/ECDSA. In their protocol, the private key x and temporary secret value k are *multiplicatively shared* among parties, so as to easily compute $k^{-1}(H(m) + rx) \bmod q$ without knowing either x or k. Their solution requires the zero-knowledge proof protocol and Paillier homomorphic encryption as building blocks to achieve guaranteed privacy preservation. Gennaro *et al.* [21] and Boneh *et al.* [22] generalise the MacKenzie–Reiter paradigm to n parties and support any subset of t (the predefined threshold that is not greater than n) parties to sign while ensuring security in the presence of any $t-1$ subset. While this is a significant breakthrough, there is a drawback in performance due to the need for inefficient distributed Paillier key generation. Gennaro and Goldfeder [23] later optimised such an approach using a faster distributed key generation. According to their protocol, secret values are *additively shared* among parties, i.e. $x = x_1 + \cdots + x_n$ and $k = k_1 + \cdots + k_n$, where each party owns a pair of unique x_i and k_i, such that multiplication with the secret shares could be split into several online two-party multiplication gates and local computations.

More recently, in 2017, a two-party ECDSA was presented by Lindell [27]. Similarly, they applied multiplicative sharing of both x and k and relied on the simple Paillier homomorphic cryptosystem for combination operations on these secret shares. In their protocol, the key generation does not incur computationally expensive operations and the signature will be eventually assembled single-handedly by the second party. Doerner *et al.* [28] optimised the two-party ECDSA signing protocol via the Simplest OT [33] and KOS [34] OT-extension protocols in the core private multiplication protocol. While this design is more efficient than previous ones, it requires significantly higher transmission bandwidth. To generalise their approaches to the multi-party setting, Lindell and Nof [29] extended their prior work and constructed a full-threshold protocol for multi-party ECDSA, where the private key is additively shared instead of multiplicatively. Thus, their private multiplication protocol on secret shares could be based on both Paillier and Oblivious Transfer [34] for various application scenarios. They briefly described a way to extend their protocol for the threshold mode. Doerner *et al.* [30] extended their previous approach to arbitrary threshold via Shamir secret sharing and a specifically oblivious transfer for randomised multiplication.

More recently, in 2018, He *et al.* [31] implemented an identity-based signature scheme in the two-party computation setting, where two parties communicate with each other using a protocol. In their protocol, the outputs will be issued by the first party only and output fairness is not expected. Wu *et al.* [32] independently introduced another two-party implementation for identity-based collaborative authentication architecture, which is based on the

additive homomorphism of Paillier to harden the defence, at the expense of computational efficiency.

Based on this brief review, it is apparent that how to design a distributed protocol for an identity-based signature scheme in the multi-party setting, particularly for real-world deployment, remains an open research challenge. This is the gap we seek to contribute to this paper.

3 Preliminaries

For reader's convenience to fully understand our paper, we present some preliminaries in this section. The notations used in this paper are described in the Nomenclature section.

3.1 Number-theoretic assumptions

We will prove the security of our protocol in the subsequent chapter based on the following number-theoretic assumptions.

Assumption 1: (Discrete logarithms assumption – DL assumption): Define $\mathbb{G}$ as a finite cyclic group with the order of prime number $q = |\mathbb{G}|$ and generator of Q. For the unknown $x \in \mathbb{Z}_q$, the advantage to compute x from the tuple (Q, xQ) for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ is negligible.

Q3

Assumption 2: (Decisional Diffie–Hellman assumption – DDH assumption): Define $\mathbb{G}$ as a finite cyclic group with the order of prime number $q = |\mathbb{G}|$ and generator of Q. For unknown $x, y, z \in \mathbb{Z}_q$, the advantage to distinguish the following distributions:

- the DDH-tuple $D = (Q, xQ, yQ, xyQ)$;
- the 4-tuple $R = (Q, xQ, yQ, zQ)$;

for any PPT adversary $\mathcal{A}$ is negligible.

3.2 Bilinear pairing

Assume that q is a large prime number, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ denote three cyclic groups with the same order of q, $\psi: \mathbb{G}_2 \to \mathbb{G}_1$ denote a homomorphic mapping satisfying $\psi(Q_2) = Q_1$, where Q_1, Q_2 are fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. If DL assumption is satisfied over $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, then $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with below attributes can be seen as a bilinear pairing:

- ***Bilinearity:*** For any elements $U \in \mathbb{G}_1$, $V \in \mathbb{G}_2$ and any integers $s, t \in \mathbb{Z}_q$, the equation $e(s \cdot U, t \cdot V) = e(U, V)^{s \cdot t}$ holds.
- ***Non-degeneracy:*** For some elements $U \in \mathbb{G}_1$, $V \in \mathbb{G}_2$, the equation $e(U, V) \neq 1_{\mathbb{G}_T}$ holds.
- ***Computability:*** Given two elements $U \in \mathbb{G}_1$, $V \in \mathbb{G}_2$, there exist effective algorithms to compute $e(U, V)$.

3.3 IEEE standard for identity-based signature

The identity-based signature standardised in IEEE P1363 is described as follows. Let $\mathbb{G}_1$, $\mathbb{G}_2$ denote two additive cyclic groups with generators of Q_1, Q_2 and order of large prime number q, $\mathbb{G}_T$ be the multiplicative cyclic group with the same order q. Define a bilinear pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and $g = e(Q_1, Q_2)$. The master secret key for KGC is a random number $s \leftarrow_R \mathbb{Z}_q$ with the corresponding public key of an element $P_{\text{pub}} = s \cdot Q_2 \in \mathbb{G}_2$. Furthermore, KGC chooses two cryptographic hash functions $h_1: \{0,1\}^* \to \mathbb{Z}_q$, $h_2: \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_q$. The system parameters are $\texttt{params} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, Q_1, Q_2, q, g, e, h_1, h_2, P_{\text{pub}}\}$ which will be published eventually, and the master secret key s will be kept by KGC privately.

- ***Extract:*** KGC executes this algorithm according to the certified identity $ID \in \{0,1\}^*$. To be more specific, it calculates $d_{\text{ID}} = (s + h_1(ID))^{-1} \bmod q$, $D_{\text{ID}} = d_{\text{ID}} \cdot Q_1$ and, finally, outputs D_{ID} as the unique private key for the user.
- ***Sign***: The user associating with the identity ID execute the following steps to sign on the message M using his/her private key D_{ID}:
  1. Selects a number $r \leftarrow_R \mathbb{Z}_q$ randomly;
  2. Computes $u = g^r$;
  3. Computes $h = h_2(M \parallel u)$ and $S = (r + h) \cdot D_{\text{ID}}$;
  4. Outputs $\text{sig} = \{h, S\}$ as the final signature on the message M.
- ***Verify:*** For given signature $\text{sig} = \{h, S\}$ on M, the verifier computes

$$u' = \frac{e(S, h_1(ID) \cdot Q_2 + P_{\text{pub}})}{g^h}$$

and accepts it if and only if $h = h_2(M \parallel u')$ holds.

3.4 Non-malleable equivocable commitments

A trapdoor commitment scheme allows a sender to commit a message with information-theoretic security. Generally, it fulfils that (i) any malicious entities (e.g. the receivers) cannot extract the committed message from given commitment transcripts; (ii) it is effectively computable for the sender to open his/her commitments; (iii) there exists a *trapdoor* whose knowledge allows to open the commitments in any possible way whereas is difficult being computed. Based on the research of Gennaro and Goldfeder [23], a non-malleable equivocable commitment scheme generally subsumes four algorithms, i.e. KG, Com, Ver, Ext, where

- KG is the key generation process that takes a security parameter as input and outputs a public key pk with a unique trapdoor tk.
- Com is the commitment algorithm defined as $\{C, D\} \leftarrow \text{Com}(pk; m; r)$, which takes the public key pk, message m to be committed, mask value r as inputs and outputs the commitment value C and associating specific value D used to open it later.
- Ver is the verification algorithms defined as $m/\perp \leftarrow \text{Ver}(C; D; pk)$, which takes the commitment tuples $\{C, D\}$, public key pk as inputs and outputs the message m if $\{C, D\}$ is valid under pk, or $\perp$ otherwise.
- Ext is the extract algorithm that takes as inputs of $\{m, r, pk\}$ satisfying $\{C, D\} = \text{Com}(m; r; pk)$, a message $m' \neq m$ and a specific string T. If $T = tk$, it outputs D' (satisfying $\text{Ver}(C; D'; pk) = m'$), otherwise, $\perp$.

It is noted that we set $D \leftarrow \perp$ and $\text{Ver}(C; \perp; pk) \leftarrow \perp$ when the sender refuses to open the commitment. A trapdoor commitment scheme must meet the requirements below:

- *Correctness:* if $\{C, D\} \leftarrow \text{Com}(pk; m; r)$, then $\text{Ver}(C; D; pk) = m$ holds;
- *Information-theoretic security:* for any different messages m and m', the distributions of C and C' are statistically close, where $\{C, D\} \leftarrow \text{Com}(pk; m; r)$ and $\{C', D'\} \leftarrow \text{Com}(pk'; m'; r')$.
- *Secure binding:* the probability for any possibility polynomial time (PPT) adversary $\mathcal{A}$ to compute a tuple of $\{C, D, D'\}$ which satisfies $\text{Ver}(C; D; pk) = m$ and $\text{Ver}(C; D'; pk) = m'$ is negligible.

Such a commitment is non-malleable, i.e. given a commitment C on m, the adversary $\mathcal{A}$ cannot produce another commitment C' such that after seeing the opening of C to m, $\mathcal{A}$ can decommit it to a related message m'. Referring to the implementation in [23], our non-malleable equivocable commitment sup-protocol is also constructed based on a secure hash function H, thus $h = H(m, r)$ is defined as the commitment on m with a uniformly chosen r of length λ.

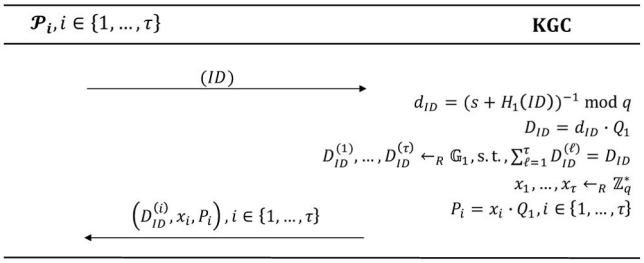

Fig. 2 *Key distribution phase*

4 Our scheme

In this section, we present a novel secure multi-party signature generation protocol for the IBS scheme standardised in IEEE P1363. To be more specific, our proposed protocol is mainly in an attempt to implement two algorithms, i.e. key distribution and distributed signature generation. We propose a new share conversion protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$, which goes over the additive cyclic group $\mathbb{G}_1$ as the sub-protocol for our multi-party signing protocol.

We assume that there are τ parties engaging in our protocol, denoted as $\mathcal{P}_1, \ldots, \mathcal{P}_\tau$, and each of them could launch a signing session, engage in the signing process, construct, as well as issue the final signature. Following the setting of the IBS scheme, key generation centre (KGC) is responsible for system setup and the distribution of partial private keys for the parties based on their common identity. It needs to note that these partial private keys are totally independent and it is impossible to reconstruct the full private key unless all parties are corrupted; therefore, the parties must cooperate with each other for a valid signature.

The details of our proposed protocol are described as follows.

4.1 Setup phase

Following the IEEE P1363 standard for the identity-based signature scheme, KGC will establish the signature system as follows:

1. Generates a bilinear map group $\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$ of the same order of large prime number q, supporting a bilinear pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.
2. Calculates $g = e(Q_1, Q_2)$, where Q_1, Q_2 are fixed generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
3. Chooses two cryptographic hash functions $h_1: \{0,1\}^* \to \mathbb{Z}_q$, $h_2: \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_q$.
4. Sets a randomly generated number $s \leftarrow_R \mathbb{Z}_q$ as its master secret key and the element $P_{\text{pub}} = s \cdot Q_2$ as the corresponding master public key.
5. Publishes the system parameters $\text{params} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, Q_1, Q_2, q, g, e, h_1, h_2, P_{\text{pub}}\}$ and stores s securely.

4.2 Key distribution phase

KGC distributes partial private keys for parties $\mathcal{P}_1, \ldots, \mathcal{P}_\tau$ in this phase. As shown in Fig. 2, when the parties register with the common identity ID, KGC will execute the following steps:

1. Computes $d_{\text{ID}} = (s + h_1(ID))^{-1} \bmod q$ and $D_{\text{ID}} = d_{\text{ID}} \cdot Q_1$;
2. Chooses τ independent random elements $D_{\text{ID}}^{(1)}, \ldots, D_{\text{ID}}^{(\tau)} \in \mathbb{G}_1$ satisfying $\sum_{\ell=1}^{\tau} D_{\text{ID}}^{(\ell)} = D_{\text{ID}}$;
3. Sets $D_{\text{ID}}^{(i)}$ as the partial private key for $\mathcal{P}_i$, for all $i \in \{1, \ldots, \tau\}$;
4. KGC further generates a pair of keys for $\mathcal{P}_i$ serving for the share conversion sub-protocol, i.e. sub-keys $\{x_i \in \mathbb{Z}_q, P_i \in \mathbb{G}_1\}$ (satisfying $P_i = x_i \cdot Q_1$) for all $i \in \{1, \ldots, \tau\}$;
5. Distributes $\{D_{\text{ID}}^{(i)}, x_i, P_i\}$ to the party $\mathcal{P}_i$ for all $i \in \{1, \ldots, \tau\}$.

We assume that all of the partial private keys are transmitted through a secure channel.

4.3 Multi-party signature generation phase

Parties generate an identity-based signature in a distributed way in this phase, with the inputs of M (the message to be signed) and the partial private keys obtained in the key distribution phase. According to Fig. 3, the multi-party signing protocol is built based on the share conversion protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$, as described in Section 4.4. To improve the readability, we describe the party who executes the following instructions as $\mathcal{P}_i$ for all $i \in \{1, \ldots, \tau\}$ and the other parties as $\mathcal{P}_j$ for all $j \in \{1, \ldots, \tau\} \setminus \{i\}$. Define ℓ as a running index from 1 to τ.

1. Each party $\mathcal{P}_i$ generates a random number $r_i \leftarrow_R \mathbb{Z}_q$, calculates $\{C_i, D_i\} = \text{Com}(g^{r_i})$ and broadcasts C_i.
2. Each party $\mathcal{P}_i$ broadcasts D_i. Assume that ξ_j be the value decommitted by $\mathcal{P}_j$ for all $j \in \{1, \ldots, \tau\} \setminus \{i\}$ who proves that he/she knows r_j s.t. $\xi_j = g^{r_j}$ using Schnorr's zero-knowledge proof scheme [14] (note that $\mathcal{P}_i$ will abort if any zero-knowledge proof fails). Each party $\mathcal{P}_i$ calculates

$$u = \prod_{\ell=1}^{\tau} u_\ell = \prod_{\ell=1}^{\tau} g^{r_\ell} = g^{\sum_{\ell=1}^{\tau} r_\ell \bmod q}$$

   and $h = h_2(M \parallel u)$, $\delta_i = (r_i + (h/\tau)) \bmod q$.
3. Each pair of parties $\mathcal{P}_i$ and $\mathcal{P}_j$ engage in two $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ sub-protocols:
   - $\mathcal{P}_i$ inputs $\{\delta_i\}$ and $\mathcal{P}_j$ inputs $\{D_{\text{ID}}^{(j)}\}$ to carry out the first $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ to get T_{ij}^1 and T_{ji}^1, respectively, s.t., $T_{ij}^1 + T_{ji}^1 = \delta_i \cdot D_{\text{ID}}^{(j)}$;
   - Similarly, $\mathcal{P}_i$ inputs $\{D_{\text{ID}}^{(i)}\}$ and $\mathcal{P}_j$ inputs $\{\delta_j\}$ to carry out the second $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ to get T_{ij}^2 and T_{ji}^2, respectively, s.t., $T_{ij}^2 + T_{ji}^2 = \delta_j \cdot D_{\text{ID}}^{(i)}$;
   - $\mathcal{P}_i$ adds up T_{ij}^1 and T_{ij}^2 to get $T_{ij} = T_{ij}^1 + T_{ij}^2$, similarly, $\mathcal{P}_j$ adds up T_{ji}^1 and T_{ji}^2 to get $T_{ji} = T_{ji}^1 + T_{ji}^2$. We can see that $T_{ij} + T_{ji} = \delta_i \cdot D_{\text{ID}}^{(j)} + \delta_j \cdot D_{\text{ID}}^{(i)}$ holds.
4. Each party $\mathcal{P}_i$ sets $T_i = \delta_i \cdot D_{\text{ID}}^{(i)} + \sum_{\ell \neq i} T_{i\ell}$ and broadcasts T_i to all other parties.
5. After $\mathcal{P}_i$ receives T_j, for all $j \in 1, \ldots, \tau \setminus \{i\}$, he/she constructs $S = \sum_{\ell=1}^{\tau} T_\ell$. If $\text{sig} = (h, S)$ is not a valid signature on the message M, $\mathcal{P}_i$ will abort this session; otherwise, he/she issues this signature and ends the protocol.

4.4 Share conversion protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$

Our multi-party signature generation protocol uses a special share conversion protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ that goes over the additive cyclic group $\mathbb{G}_1$. Assume that $\mathcal{P}_i$ owns a secret element $D_{\text{ID}}^{(i)} \in \mathbb{G}_1$ and $\mathcal{P}_j$ holds a secret value $\delta_j \in \mathbb{Z}_q$. Our $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ protocol aims to securely convert the two secrets into additive shares of $T_{ij}, T_{ji} \in \mathbb{G}_1$ for $\mathcal{P}_i$ and $\mathcal{P}_j$,

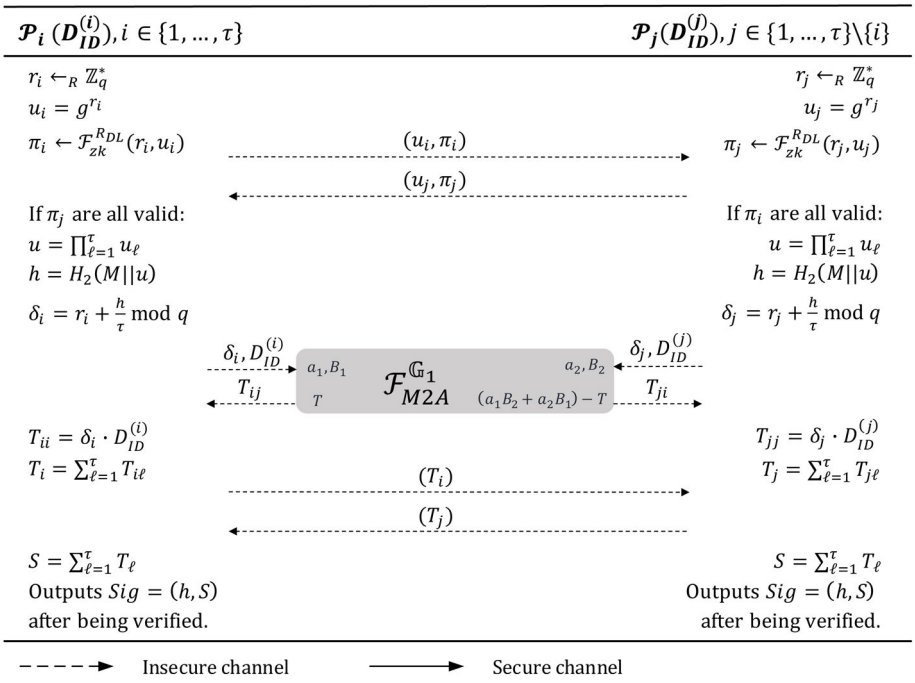

Fig. 3 *Distributed signature generation phase*

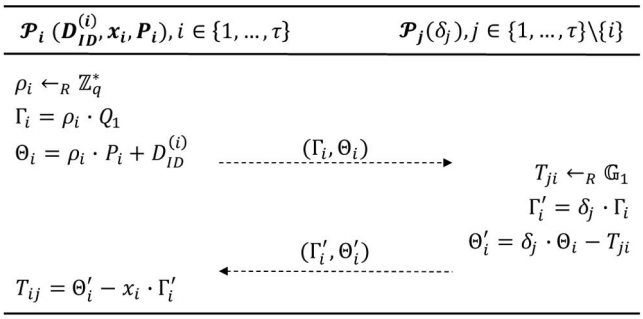

Fig. 4 *Share conversion protocol* $\mathcal{F}_{M2A}^{\mathbb{G}_1}$

respectively, s.t., $T_{ij} + T_{ji} = \delta_j \cdot D_{\mathrm{ID}}^{(i)}$. $\mathcal{P}_i$ further, we need to input his/her sub-keys $\{x_i, P_i\}$ obtained from KGC in advance. To be more specific, as shown in Fig. 4, the $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ sub-protocol can be done as follows:

1. $\mathcal{P}_i$ generates a random number $\rho_i \leftarrow_R \mathbb{Z}_q$, encrypts $D_{\mathrm{ID}}^{(i)}$ with $\Gamma_i = \rho_i \cdot Q_1$, $\Theta_i = \rho_i \cdot P_i + D_{\mathrm{ID}}^{(i)}$, and sends $\{\Gamma_i, \Theta_i\}$ to $\mathcal{P}_j$ as the initialisation request.
2. $\mathcal{P}_j$ randomly chooses an element $T_{ji} \leftarrow_R \mathbb{G}_1$ and re-randomises the ciphertexts as $\Gamma_i' = \delta_j \cdot \Gamma_i$, $\Theta_i' = \delta_j \cdot \Theta_i - T_{ji}$. Finally, $\mathcal{P}_j$ transmits $\{\Gamma_i', \Theta_i'\}$ to $\mathcal{P}_i$ as the response.
3. $\mathcal{P}_i$ decrypts $\{\Gamma_i', \Theta_i'\}$ to obtain $T_{ij} = \Theta_i' - x_i \cdot \Gamma_i'$. At this point, we can see that $T_{ij} + T_{ji} = \delta_j \cdot D_{\mathrm{ID}}^{(i)}$.

4.5 Correctness

It is easy to get $h = h_2(M \parallel u)$, where $u = g^{\sum_{\ell=1}^{\tau} r_\ell \bmod q}$. As for another part of the signature

$$
\begin{aligned}
S &= \sum_{\ell=1}^{\tau} T_\ell = \sum_{i=1}^{\tau} \sum_{j=1}^{\tau} \delta_i \cdot D_{\mathrm{ID}}^{(j)} \\
&= \left(\sum_{i=1}^{\tau} \delta_i \right) \cdot \left(\sum_{j=1}^{\tau} D_{\mathrm{ID}}^{(j)} \right) \\
&= \left(\sum_{i=1}^{\tau} \left(r_i + \frac{h}{\tau} \right) \right) \cdot D_{\mathrm{ID}} \\
&= \left(\sum_{i=1}^{\tau} r_i + h \right) \cdot D_{\mathrm{ID}}
\end{aligned}
$$

Therefore, if all of the parties behaviour honestly, sig $= \{h, S\}$ will be a valid signature.

5 Security analysis

In this section, we prove the security, especially the unforgeability of our proposed multi-party signature generation protocol for the IBS scheme standardised in IEEE P1363.

The proof is performed following the definitions in [23, 35], where $\mathscr{A}$ denotes an adversary who can forge a signature against the multi-party signing scheme (denoted as $\mathbb{P}$) with a non-negligible possibility ε. We write Corrupt($\mathscr{A}$) as the set of parties that are corrupted. Then we construct a simulator $\mathcal{S}$ who intends to forge in the original identity-based signature scheme using the information learned from $\mathscr{A}$. That is, for a protocol to be secure, it must be possible in the simulated setting to generate something indistinguishable from the adversary $\mathscr{A}$'s view. We write Sim as the simulator algorithm. Thus we have the following distributions:

- $\mathrm{MPIBS}_{\mathbb{P},\mathscr{A}}(\kappa; \{D_{\mathrm{ID}}^{(i)|i \notin \mathrm{Corrupt}(\mathscr{A})}\})$: execute the multi-party signing protocol on security parameter κ, where each honest party $\mathscr{P}_i, i \notin \mathrm{Corrupt}(\mathscr{A})$ engages the signing session using the partial private key $D_{\mathrm{ID}}^{(i)}$, and $\mathscr{A}$ sets the inputs for corrupted parties. Let sig_i denotes the output of $\mathscr{P}_i$ and let V denotes the view of corrupt parties (which consists of the private keys, random variants and lists of transcripts received during the protocol).
  Output$(\{V_i | i \in \mathrm{Corrupt}(\mathscr{A})\}, \{\mathrm{sig}_i | i \notin \mathrm{Corrupt}(\mathscr{A})\})$.
- $\mathrm{Sim}_{\mathscr{S},\mathscr{A}}(\kappa; \{D_{\mathrm{ID}}^{(i)|i \notin \mathrm{Corrupt}(\mathscr{A})}\})$: execute the simulated protocol after $\mathscr{S}$ learns a set of inputs $\{D_{\mathrm{ID}}^{(i)} | i \in \mathrm{Corrupt}(\mathscr{A})\}$ from $\mathscr{A}$ sets the inputs for corrupted parties. Then, it computes $\mathrm{sig}_1, \ldots, \mathrm{sig}_\tau$ and gives V^* as the simulated views.
  Output$(\{V^* | \{\mathrm{sig}_i | i \notin \mathrm{Corrupt}(\mathscr{A})\})$.

Definition 1: A multi-party signing protocol is secure if for any possibility polynomial time (PPT) adversary $\mathscr{A}$ there exists a simulator such that for all partial private keys of honest parties $\{D_{\mathrm{ID}}^{(i)} | i \notin \mathrm{Corrupt}(\mathscr{A})\}$, the distributions

$$\mathrm{MPIBS}_{\mathbb{P},\mathscr{A}}(\kappa; \{D_{\mathrm{ID}}^{(i)|i \notin \mathrm{Corrupt}(\mathscr{A})}\})$$

and

$$\mathrm{Sim}_{\mathscr{S},\mathscr{A}}(\kappa; \{D_{\mathrm{ID}}^{(i)|i \notin \mathrm{Corrupt}(\mathscr{A})}\})$$

are indistinguishable in κ.

Then our task now is to construct a simulator $\mathscr{S}$ to forge in the original identity-based signature scheme also with a non-negligible possibility.

Based on the security model defined above, we now attempt to build the simulator S to interact with $\mathscr{A}$ in an indistinguishable way and to forge in the original identity-based signature scheme with a non-negligible possibility. It's obvious that the simulation makes no sense if all the parties are corrupted. Thus, for simplicity, we assume that $\mathscr{P}_2, \ldots, \mathscr{P}_\tau$ are corrupt parties and $\mathscr{P}_1$ is the honest party. It should be pointed out that the proof also works if there remains more than one honest party because of the concurrently non-malleable commitment we used (where $\mathscr{A}$ can learn many commitments from the honest parties). So the simulation does not lose generality.

Our simulation will act on behalf of the honest party $\mathscr{P}_1$ and interact with the adversary (who controls $\mathscr{P}_2, \ldots, \mathscr{P}_\tau$). Since that $\mathscr{A}$ is assumed to be a rushing adversary, we remark that the simulator S (acting as $\mathscr{P}_1$) always speaks firstly at each round throughout the proof; furthermore, the corrupt parties receive the message in *this* round firstly from the honest party and only then from the adversary. We assume that KGC in our protocol is still trusted, such that the key generation phase is credible. So what the adversary $\mathscr{A}$ needs to do is to request a set of parties to sign a message $m_1, \ldots, m_k$ and to observe all the transcripts among the signing instances. At the end of the simulation, $\mathscr{A}$ is assumed to output a forgery signature (h, S) on $m' \notin \{m_1, \ldots, m_k\}$ that could pass the verification procedure under the parties' common identity ID^*.

We now turn to build $\mathscr{S}$ as a forger for the original identity-based signature scheme in the IEEE P1363 standard, who utilises the adversary $\mathscr{A}$ as the subroutine in a 'simulated' multi-party signature generation protocol. To be more specific, each time $\mathscr{A}$ asks for a signature on m_i, $\mathscr{S}$ will query its signature oracle for a real signature (h_i, S_i). It will then simulate the multi-party signing protocol with input m_i and reach to the signature (h_i, S_i). Due to the indistinguishability between the simulation and real protocol, the adversary $\mathscr{A}$ would still take out a forgery with the same possibility just as in the real world. Such a forged signature (h', S') is performed on the m' which $\mathscr{S}$ has never queried on, therefore is also a successful forgery for the original identity-based signature scheme in the IEEE P1363 standard. In the subsequent part of this section, we will turn to the details of the simulation.

- ***Setting up****:* Before the simulation, we assume that KGC has initialised the signature system and distributed the partial private keys to the parties, both corrupt and honest associating with the common identity ID^*. At this point, an adversary $\mathscr{A}$ could learn the private keys of all corrupt parties, including partial private keys $D_{ID}^{(j)}$ and subkeys $\{x_j, P_j\}$ used in $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ for $j \in \{2, \ldots, \tau\}$. However, neither the adversary $\mathscr{A}$ nor the simulator $\mathscr{S}$ knows the private keys related to $\mathscr{P}_1$.
- ***Signature generation simulation:*** In the following simulation, if some parties refuse to decommit in Step 2, or some ZK proofs fail in Step 2, or the final signature does not pass the verification procedure, $\mathscr{S}$ will then abort just as what the protocol is supposed to. Each time $\mathscr{A}$ submits a fresh message m_i to be signed:

  ○ Phase 1: $\mathscr{S}$ (acting as $\mathscr{P}_1$) selects a number $r_1 \leftarrow_R \mathbb{Z}_q$ randomly, computes $\{C_1, D_1\} = \mathrm{Com}(g^{r_1})$ and sends C_1 to all other parties. Similarly, $\mathscr{P}_j$ $(j > 2)$ broadcasts commitment C_j.
  ○ Phase 2:

  (2a) Each party broadcasts D_j to decommit the value ξ_j, which is later proven to satisfy $\xi_j = g^{r_j}$ through ZK proof. Since that $\mathscr{S}$ knows the random number r_1, it could perform the proof easily.
  (2b) $\mathscr{S}$ queries its signature oracle with $\{m_i, ID^*\}$, the later would output a signature $\{h_i, S_i\}$ for it to computes

$$u = g^{-h_i} \cdot e(S_i, h_1(ID^*) \cdot Q_2 + P_{\mathrm{pub}}))$$

  (where $h_i = h_2(m_i, u)$).
  (2c) Since that $\mathscr{S}$ needs to change the decommitted value of $\mathscr{P}_1$ as $\hat{u}_1 = u \cdot \prod_{j>1} u_j^{-1}$ so as to reach h_i and S, so it rewinds $\mathscr{A}$ to the decommitment step. Note that $\mathscr{S}$ knows the trapdoor of commitment scheme, it is able to run the Ext algorithm undetected. At this point, $\mathscr{A}$ knows the values $\delta_j = r_j + (h_i/\tau) \bmod q$ at corrupt parties (for $j > 2$).

  ○ Phase 3: Each pair of parties engage in all the $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ instances as expected. However, $\mathscr{S}$ (acting as $\mathscr{P}_1$) neither know $D_{\mathrm{ID}}^{(1)}$ nor compute the discrete logarithm of $\hat{u}_1$ (i.e., $\hat{r}_1 = \log_g \hat{u}_1$), so that it cannot re-randomise or decrypt its shares during the execution of $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ instances with $\mathscr{P}_j (j > 2)$. Thus,

  (3a) $\mathscr{S}$ returns $\mathscr{P}_j$ with two random elements in $\mathbb{G}_1$ when executing the first $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ that supposes the inputs of δ_1 and $D_{\mathrm{ID}}^{(j)}$;
  (3b) Similarly, $\mathscr{S}$ selects two random elements in $\mathbb{G}_1$ and sends to $\mathscr{P}_j$ when executing the second $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ that supposes the inputs of $D_{\mathrm{ID}}^{(1)}$ and δ_j;
  (3c) $\mathscr{S}$ sets T_{1j} as a random element in $\mathbb{G}_1$;

  After all the execution of $\mathscr{F}_{M2A}^{\mathbb{G}_1}$ instances, each $\mathscr{P}_j$ (for $j > 2$) computes

$$T_j = \delta_j \cdot D_{ID}^{(j)} + \sum_{\ell \neq j} T_{\ell j}$$

  ○ Phase 4: At this point, $\mathscr{S}$ could learn all the T_j of corrupt parties. So it could calculate the correct T_1 for $\mathscr{P}_1$ as

$$T_1 = S_i - \sum_{j>1} T_j$$

  ○ Phase 5: All the parties (including the corrupt parties and S) execute the remaining steps following the multi-party signature generation protocol. $\mathscr{S}$ uses T_1 as the additively share for $\mathscr{P}_1$.

Lemma 1: Assuming that DDH and DL assumptions hold and the commitment utilised in the protocol is a non-malleable equivoable commitment, then we can conclude that the simulation has the properties below:

- for given message m, it issues a correct signature (h, S) or abort;
- it meets fully indistinguishability compared with a real execution.

Proof: The differences between the simulated setting and the real setting are analysed as follows.

In the $\mathcal{F}_{M2A}^{\mathbb{G}_1}$ protocol, $\mathcal{S}$ simulates $\mathcal{P}_1$ to re-randomise the ciphertexts from $\mathcal{P}_j$. The instructions are $\Gamma'_j = \delta_1 \cdot \Gamma_j$, $\Theta'_j = \delta_1 \cdot \Theta_j - T_{1j}^{(1)}$ in the real protocol, but in the simulated protocol Γ'_j and Θ'_j are chosen randomly by $\mathcal{S}$ from $\mathbb{G}_1$. It should be noted that δ_1 and $T_{1j}^{(1)}$ are randomly generated so that according to the DDH assumption, the correct Γ'_j and Θ'_j are completely computationally indistinguishable from the randomly selected ones under the view of the adversary $\mathcal{A}$.

On the other hand, when $\mathcal{S}$ receives the signature (h, S) from its signature oracle, it could compute $u_1 = u \cdot \prod_{j>1} u_j^{-1}$ and rewinds $\mathcal{A}$ to reach the fixed (h, S) easily. By this execution, it implicitly contributes a fresh value $\hat{r}_1$ for $\mathcal{P}_1$ satisfying $u_1 = g^{\hat{r}_1}$ and $\hat{r}_1 = \hat{r} - \sum_{j>1} r_j \bmod q$. Other transcripts associating with the $\hat{r}_1$ is the re-randomised ciphertexts answered by $\mathcal{S}$ for $\mathcal{P}_1$ in step (3a), where $\Gamma'_j = (\hat{r}_1 + h\tau^{-1}) \cdot \Gamma_j$, $\Theta'_j = (\hat{r}_1 + h\tau^{-1}) \cdot \Theta_j - T_{1j}^{(1)}$. Under the assumptions of DL over $\mathbb{G}_T$ and $\mathbb{G}_1$, it obviously uncomputable for $\mathcal{A}$ to detect whether the exponent of u_1 and the scalar multiplier of Γ'_j or Θ'_j contain the same random value r_1 or $\hat{r}_1$.

The simulator S cloud always computes the correct share T_1 for $\mathcal{P}_1$ from the signature (h, S) at Phase 4. Since that (h, S) is a valid signature on m received from the original signature oracle in Step (2b), it could go to the conclusion that Phase 5 is totally indistinguishable from the real execution at the view of $\mathcal{A}$. The simulated protocol would come to an end with (h, S) or abort. □

Theorem 1: Protocol in the previous section securely computes the identity-based signature in the IEEE P1363 standard under the assumptions of the original identity-based signature scheme as well as the non-malleable equivocable commitment.

Proof: Based on Lemma 1, the simulation $\mathcal{S}$ described below produces an indistinguishable view for the adversary $\mathcal{A}$, and therefore, $\mathcal{A}$ will forge a signature with the same possibility as in real life, denoted by ϵ. In the end, $\mathcal{S}$ could output $\mathcal{A}$'s forgery as its fake signature to the single party identity-based signature scheme standardised in the IEEE P1363 with the same possibility.

However, under the security of the IBS scheme in the IEEE P1363 standard proven by decades, the probability of success for $\mathcal{S}$ must be negligible, which implies that ϵ must also be negligible. Therefore, $\mathcal{A}$ has no non-negligible probability against our proposed multi-party signature generation protocol. □

6 Complexity analysis

In this section, we analyse the computation and communication complexity of our proposed multi-party signing protocol for an identity-based signature scheme standardised in the IEEE P1363. As shown in Table 1, we provide a theoretical analysis of our multi-party signing protocol, comparing with the original signing algorithm.

6.1 Time complexity

The items for time complexity analysis are listed as follows:

- T_{exp}: The time cost for an exponentiation operation in $\mathbb{G}_T$.
- T_{mul}: The time cost for a multiplication operation in $\mathbb{G}_T$.
- $T_{\text{com-zk}}$: The time cost for the commitment and zero-knowledge proof on the knowledge of the discrete log of an element in $\mathbb{G}_T$.
- T_{sm}: The time cost for a scalar multiplication operation in $\mathbb{G}_1$.
- T_{pa}: The time cost for an addition operation in $\mathbb{G}_1$.
- T_{M2A}: The time cost for the share conversion sub-protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$.
- T_{Ver}: The time cost for the verify procedure of identity-based signature scheme.
- T_{h}: The time cost for the secure hash function.

Therefore, for each party, the signing protocol mainly contains one random number generation for r_i (which could be precomputed in advance), one exponentiation for u_i, one commitment and zero-knowledge proof for broadcasting u_i correctly, $\tau - 1$ multiplication over $\mathbb{G}_T$ for u, one secure hash function for h, $\tau - 1$ share conversion sub-protocols for T_{ij}, one scalar multiplication and $2 \times \tau$ addition in $\mathbb{G}_1$ for T_i and S, and finally one verify operation on the generated signature. The theoretical time complexity for each party will be

$$\begin{aligned} T_{\text{basic}} = {} & T_{\text{exp}} + T_{\text{com-zk}} + (\tau - 1) \times T_{\text{mul}} + T_{\text{h}} \\ & + (\tau - 1) \times T_{M2A} + T_{\text{sm}} + 2\tau \times T_{\text{pa}} + T_{\text{Ver}} \end{aligned}$$

in total, which is approximately $\Theta(\tau)$, i.e. be proportional to the number of parties.

In contract, signer in the original signing algorithm requires, similarly, one random number generation for r (which could be pre-computed in advance), one exponentiation for u, one secure hash function for h and finally one scalar multiplication in $\mathbb{G}_1$ for S. Thus the time complexity will be

$$T_{\text{IBS}} = T_{\text{exp}} + T_{\text{h}} + T_{\text{sm}}$$

in total.

6.2 Communication complexity

We assume that the elements in $\mathbb{Z}_q$ are presented in κ bits, s.t., $\kappa = |q|$, elements from $\mathbb{G}_1$ are presented in λ bits and elements from $\mathbb{G}_T$ are presented in σ bits. we further assume that a commitment consists of a single element from $\mathbb{Z}_q$, and its decommitment includes the committed value along with the zero-knowledge proof, which points to two elements in $\mathbb{G}_T$ and one element in $\mathbb{Z}_q$.

Recall that the number of parties is τ, thus in our multi-party signing protocol; each party needs to broadcast an element of $\mathbb{G}_T$, i.e. u_i with its corresponding commitment and zero-knowledge proof π_i to the others. In addition, each pair of parties need to execute the share conversion sub-protocol $\mathcal{F}_{M2A}^{\mathbb{G}_1}$, where involves the transmissions of two pairs of elements in $\mathbb{G}_1$. Finally, each party must broadcast its signature share $T_i \in \mathbb{G}_1$. Therefore a successful multi-party signature generation protocol requires

$$\begin{aligned} & \tau(\tau - 1) \cdot (3\sigma + \kappa) + \frac{\tau(\tau - 1)}{2} \cdot (4 \cdot 2) \cdot \lambda + \tau(\tau - 1) \cdot \lambda \\ & = \tau(\tau - 1)(3\sigma + \kappa + 5\lambda) \end{aligned}$$

Table 1 Cost analysis

	Time costs per party	Overall communication costs
multi-party signing	$T_{\text{basic}} = T_{\text{exp}} + T_{\text{com-zk}} + (\tau - 1) \times T_{\text{mul}} + T_{\text{h}} + (\tau - 1) \times T_{M2A} + T_{\text{sm}} + 2\tau \times T_{\text{pa}} + T_{\text{Ver}}$	$\tau(\tau - 1) \cdot (3\sigma + \kappa + 5\lambda)$
original signing	$T_{\text{IBS}} = T_{\text{exp}} + T_h + T_{sm}$	$\perp$

Table 2 Types of pairing curves

Types of pairing	Degree k	AES-like security level, bits
Miyaji–Nakabayashi–Takano (MNT) curve [38]	$k = 6$	80
Barreto–Naehrig (BN) curve [36]	$k = 12$	128
Kachisa–Schaefer–Scott (KSS) curve [39]	$k = 18$	192
Barreto–Lynn–Scott (BLS) curve [40]	$k = 24$	256

Table 3 Comparison of communication costs (in kilobytes)

Curve	MNT-80	BN-128	KSS-192	BLS-256
multi-party signing (in total)	3.4 KiB	8.81 KiB	24.3 KiB	41.63 KiB

Running among three parties with different security levels.

Table 4 Comparison of signing protocol (timing in milliseconds) among different security levels

Pairing curves	Multi-party signing (per party), ms	Original signing, ms
MNT-80	12.94	1.09
BN-128	30.07	3.66
KSS-192	566.38	75.46
BLS-256	18,269.17	2275.61

Table 5 Running times (in milliseconds) among different parties

Number of parties	Running times, ms
3	311.86
4	448.31
5	558.2
6	591.31
7	707.21

bits in total. Concretely, for example BN-pairing [36] with an extension degree of 12 satisfies $\lambda = 2 \cdot \kappa$, $\sigma = 12 \cdot \kappa$, thus the communication cost for multi-party signing is roughly $47 \cdot (\tau^2\kappa - \tau\kappa) \simeq \Theta(\tau^2\kappa)$, which is be proportional to the number of parties squared and the security strength.

7 Implementation

We implemented our protocol in C/C++, and ran it on a personal computer (Dell with Intel® Core™ CPU I7-6700 @ 3.4 GHz with 8 GB RAM and the Ubuntu 18.04 OS). Our implementation is based on the libraries of MIRACL [37]. Firstly, we implemented the protocol in a three-party setting and ran the experiments with different types of pairing curves (all are associated with an even degree of extension filed k and such that corresponding to different security levels, as listed in Table 2). Each execution was run 1000 times and we took the average. The executing time for *each party* can be seen in the graph in Fig. 5 and in Tables 3 and 4. We stress Q5 that the times include *local computation only* and no communication. As is clearly seen, the signing time is proportional to the security level and approximately a factor of 8.91 of local signatures.

Furthermore, we compared the concrete communication overheads for the curves in Table 2 in *three-party* setting and the results are how much the parties transmit during the execution of multi-party signing procedure, e.g. for MNT-80 curve, $\kappa = 160$, $\lambda = 320$, $\sigma = 960$ such that it reaches around 3.4 kilobytes in total; for BN-80 curve, $\kappa = 256$, $\lambda = 512$, $\sigma = 3072$ such that it reaches around 8.81 kilobytes in total; for KSS-80 curve, $\kappa = 512$, $\lambda = 1024$, $\sigma = 9216$ such that it reaches around 24.3 kilobytes in total; for BLS-80 curve, $\kappa = 512$, $\lambda = 2048$, $\sigma = 15360$ such that it reaches around 41.63 kilobytes in total. As we have analysed, our

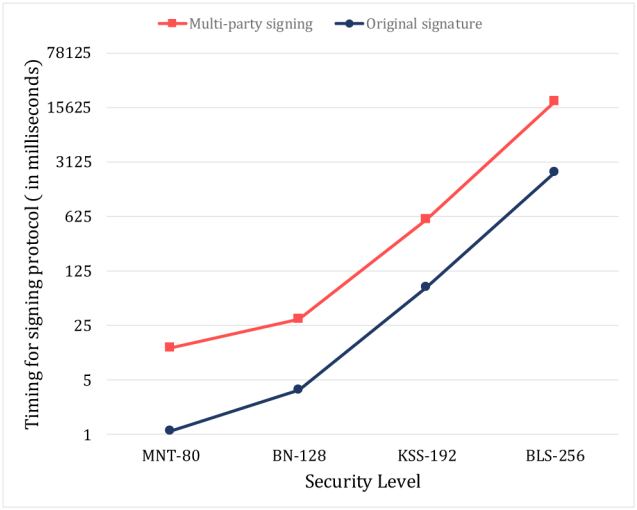

Fig. 5 *Running times in milliseconds for signing protocol for different security levels. The results are presented in logarithmic scale with the base of 5*

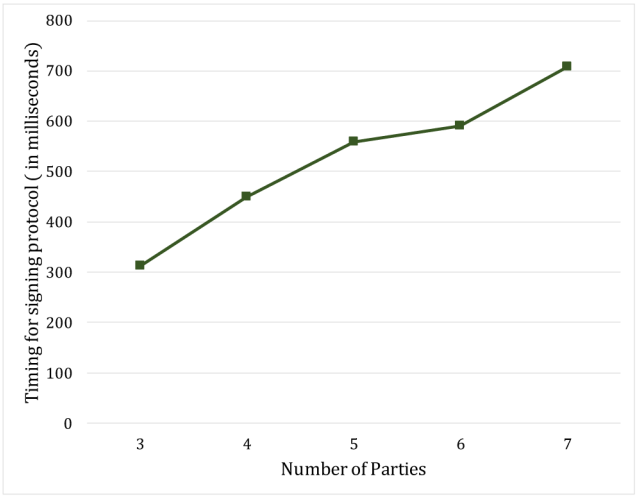

Fig. 6 *Running times in milliseconds for signing protocol for different parties*

multi-party identity-based signing protocol is free from OT-based operations or Paillier homomorphic cryptosystems; the communication/security trade-off is reasonably low.

For implementation in the network setting, we created several virtual machines connected over LAN with 100 Mbps network bandwidth to perform the protocol on behalf of the parties. We chose the BN-128 curve [36] and ran experiments sequentially among those parties (meaning that each party will delay and wait for the responses from all the other parties). The results are as shown in Table 5 and Fig. 6. We can see that the signing time is significantly practical: about 311.86 ms for three parties, about 558.2 ms for five parties and about 707.21 ms for seven parties, including the communication latency and waiting for the delay. We stress that the implementation is single-threaded and the running time can be significantly reduced by using multiple threads.

8 Conclusion

As the smart electrical commerce system tends to be more distributed, a secure multi-party identity-based signing protocol will be a promising setting for the distributed authentication and authorisation. Therefore, we designed the first multi-party signature generation protocol for the identity-based signature scheme standardised in IEEE P1363. The highlight of our protocol is to generate an identity-based signature jointly by multiple parties with practical computation/communication costs. Our proposed protocol is proven to be secure against more than one corrupt party

under weak assumptions. Furthermore, we implemented our protocol in an experimental environment, and the results demonstrate that it is really fast at signing time and has relatively low transmission overheads.

Future researches include testing it with more participants in both LAN and WAN setting, as well as on embedded devices for more extensive evaluation.

9 Acknowledgments

The authors greatly appreciate the anonymous reviewers and the associate editor for the valuable comments and suggestions, which helped us improve the content, organisation, and presentation of this paper. The work was supported in part by the National Natural Science Foundation of China (grant nos. 61972294, 61932016), and in part, the Science and Technology planning project of ShenZhen (grant no. JCYJ20170818112550194).

10 References

[1] Statista. Retail e-commerce sales worldwide from 2014 to 2021 (in billion U.S. dollars), 2018. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales

[2] Allen, R.: Top e-commerce trends to inform your 2017 marketing strategy, 2017. https://www.smartinsights.com/ecommerce/ecommerce-strategy/top-ecommerce-trends-inform-2017-marketing-strategy

[3] The Amazon recommendations secret to selling more online. http://rejoiner.com/resources/amazon-recommendations-secret-selling-online. Accessed: 2020-01-29

[4] Firm linked to Alibaba opens China's biggest robot warehouse to help deal with singles day demand. https://www.cnbc.com/2018/10/30/alibaba-cainiao-chinas-biggest-robot-warehouse-for-singles-day.html. Accessed: 2020-01-29

[5] Sennaar, K.: Artificial intelligence in eCommerce comparing the top 5 largest firms. https://emerj.com/ai-sector-overviews/artificial-intelligence-in-ecommerce-amazon-alibaba-jd-com. Updated: 2019-11

[6] Song, Z., Sun, Y., Wan, J., *et al.*: 'Smart e-commerce systems: current status and research challenges', *Electron. Mark.*, 2019, **29**, (2), pp. 221–238

[7] He, D., Kumar, N., Shen, H., *et al.*: 'One-to-many authentication for access control in mobile pay-tv systems', *Sci. China Inf. Sci.*, 2016, **59**, (5), p. 052108

[8] Rutherford, B., Dagher, A., Wiseman, M., *et al.*: 'Customer authentication in e-commerce transactions'. US Patent 9,514,458, December 6 2016

[9] Jiang, Q., Kumar, N., Ma, J., *et al.*: 'A privacy-aware two-factor authentication protocol based on elliptic curve cryptography for wireless sensor networks', *Int. J. Netw. Manage.*, 2017, **27**, (3), p. e1937

[10] Chaudhry, S.A., Farash, M.S., Naqvi, H., *et al.*: 'A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography', *Electron. Commer. Res.*, 2016, **16**, (1), pp. 113–139

[11] Feng, Q., He, D., Zeadally, S., *et al.*: 'Anonymous biometrics-based authentication scheme with key distribution for mobile multi-server environment'. *Future Gener. Comput. Syst.*, 2018, **84**, pp. 239–251

[12] IEEE standard for identity-based cryptographic techniques using pairings. IEEE Std 1363.3-2013, pages 1–151, Nov 2013

[13] Yao, A.C.: 'Protocols for secure computations'. 23rd Annual Symp. on Foundations of Computer Science (SFCS 1982), 1982, pp. 160–164

Q6 [14] Schnorr, C.-P.: 'Efficient identification and signatures for smart cards'. Conf. on the Theory and Application of Cryptology, 1989, pp. 239–252

[15] Stinson, D.R., Strobl, R.: 'Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates'. Australasian Conf. on Information Security and Privacy, 2001, pp. 417–434

[16] Langford, S.K.: 'Threshold DSS signatures without a trusted party'. Annual Int. Cryptology Conf., 1995, pp. 397–409

[17] Gennaro, R., Jarecki, S., Krawczyk, H., *et al.*: 'Robust threshold DSS signatures'. Int. Conf. on the Theory and Applications of Cryptographic Techniques, 1996, pp. 354–371

[18] Chandramowliswaran, N., Srinivasan, S., Muralikrishna, P.: 'Authenticated key distribution using given set of primes for secret sharing', *Syst. Sci. Control Eng.*, 2015, **3**, (1), pp. 106–112

[19] Jarecki, S., Kiayias, A., Krawczyk, H., *et al.*: 'Highly-efficient and composable password-protected secret sharing (or: how to protect your Bitcoin wallet online)'. 2016 IEEE European Symp. on Security and Privacy (EuroS&P), 2016, pp. 276–291

[20] Hu, L., Huang, Y., Yang, D., *et al.*: 'SSe-Cloud 'using secret sharing scheme to secure keys'. IOP Conf. Series: Earth and Environmental Science, 2017, vol. 81, p. 012207

[21] Gennaro, R., Goldfeder, S., Narayanan, A.: 'Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security'. Int. Conf. on Applied Cryptography and Network Security, 2016, pp. 156–174

[22] Boneh, D., Gennaro, R., Goldfeder, S.: 'Using level-1 homomorphic encryption to improve threshold DSA signatures for Bitcoin wallet security'. Int. Conf. on Cryptology and Information Security in Latin America, 2017, pp. 352–377

[23] Gennaro, R., Goldfeder, S.: 'Fast multiparty threshold ECDSA with fast trustless setup'. Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security, 2018, pp. 1179–1194

[24] Micali, S., Goldreich, O., Wigderson, A.: 'How to play any mental game'. Proc. of the Nineteenth ACM Symp. on Theory of Computing (STOC), 1987, pp. 218–229

[25] Goldreich, O.: *'Foundations of cryptography: volume 2, basic applications'* (Cambridge University Press, 2009)

Q7 [26] MacKenzie, P., Reiter, M.K.: 'Two-party generation of DSA signatures'. Annual Int. Cryptology Conf., 2001, pp. 137–154

[27] Lindell, Y.: 'Fast secure two-party ECDSA signing'. Annual Int. Cryptology Conf., 2017, pp. 613–644

[28] Doerner, J., Kondi, Y., Lee, E., *et al.*: 'Secure two-party threshold ECDSA from ECDSA assumptions'. 2018 IEEE Symp. on Security and Privacy (SP), 2018, pp. 980–997

[29] Lindell, Y., Nof, A.: 'Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody'. Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security, 2018, pp. 1837–1854

[30] Doerner, J., Kondi, Y., Lee, E., *et al.*: 'Threshold ECDSA from ECDSA assumptions: the multiparty case'. 2019 IEEE Symp. on Security and Privacy (SP), 2019, pp. 1051–1066

[31] He, D., Zhang, Y., Wang, D., *et al.*: 'Secure and efficient two-party signing protocol for the identity-based signature scheme in the IEEE P1363 standard for public key cryptography', *IEEE Trans. Dependable Secur. Comput.*, 2018

Q8 [32] Wu, L., Wang, J., Choo, K.-K.R., *et al.*: 'Secure key agreement and key protection for mobile device user authentication', *IEEE Trans. Inf. Forensics Secur.*, 2019, **14**, (2), pp. 319–330

[33] Chou, T., Orlandi, C.: 'The simplest protocol for oblivious transfer'. Int. Conf. on Cryptology and Information Security in Latin America, 2015, pp. 40–58

[34] Keller, M., Orsini, E., Scholl, P.: 'Actively secure OT extension with optimal overhead'. Annual Cryptology Conf., 2015, pp. 724–741

[35] Hazay, C., Lindell, Y.: '*Efficient secure two-party protocols: techniques and constructions*' (Springer Science & Business Media, 2010)

[36] Barreto, P.S., Naehrig, M.: 'Pairing-friendly elliptic curves of prime order'. Int. Workshop on Selected Areas in Cryptography, 2005, pp. 319–331

[37] MIRACL UK Ltd. Multiprecision integer and rational arithmetic C/C++ library (MIRACL). https://github.com/miracl/MIRACL, 2016

[38] Miyaji, A., Nakabayashi, M., Takano, S.: 'New explicit conditions of elliptic curve traces for FR-reduction', *IEICE Trans. Fund. Electron., Commun. Comput. Sci.*, 2001, **84**, (5), pp. 1234–1243

[39] Kachisa, E.J., Schaefer, E.F., Scott, M.: 'Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field'. Int. Conf. on Pairing-Based Cryptography, 2008, pp. 126–135

[40] Barreto, P.S., Lynn, B., Scott, M.: 'Constructing elliptic curves with prescribed embedding degrees'. Int. Conf. on Security in Communication Networks, 2002, pp. 257–267

Author Queries

Q Please make sure the supplied images are correct for both online (colour) and print (black and white). If changes are required please supply corrected source files along with any other corrections needed for the paper.
Q1 There is a mismatch of article title between doc file and metadata or doc file and pdf, followed pdf. Please check and confirm.
Q2 Please confirm the changes made in the article title. Please note that it is the IET's house style to remove words such as "Novel", "New" and "Study of" as well as "A", "An", "The", "On the", and "On".
Q3 Both "probabilistic polynomial time" and "possibility polynomial time " have the same abbreviation "PPT". Please clarify or provide an alternative.
Q4 Table 2 has been changed to Table 1 and all other tables in the text are renumbered sequentially as per IET style. Please check and confirm the changes made.
Q5 We have inserted citations for Table 4. Please approve or provide an alternative.
Q6 Please provide place of conference in Refs. [13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 33, 34, 36, 39, 40].
Q7 Please provide the location of the publisher (country) in Refs. [25, 35].
Q8 Please provide volume number, page range in Refs. [31].