Journal of Information Security and Applications 61 (2021) 102805

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Dynamic threshold ECDSA signature and application to asset custody in blockchain

Huili Wang [a,b], Wenping Ma [a], Fuyang Deng [c], Haibin Zheng [c,*], Qianhong Wu [d]

[a] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China
[b] China Electronics Standardization Institute, Beijing, China
[c] Hangzhou Innovation Institute, Beihang University, Hangzhou, China
[d] School of Cyber Science and Technology, Beihang University, Beijing, China

ARTICLE INFO

Keywords:
Blockchain
Threshold ECDSA signature
Dynamic cryptosystem
Distributed key generation
Asset custody

ABSTRACT

The centralized exchange is one of the hottest DeFi applications based on blockchain transaction systems. However, depositing user assets to the exchanges brings the security risks of assets misappropriation. Threshold cryptosystem can effectively solve the drawbacks of centralized hosting by assigning the assets authorization to multiple trustees, but the collusion attack generated by malicious trustees is still unavoidable. In this paper, we propose a new dynamic threshold ECDSA signature scheme which is compatible with current blockchain transaction system. It realizes distributed custody of assets in exchanges, and further achieves a dynamic mechanism allowing user join and drop out to resist collusion attacks. Specifically, we formalize the definition of this system architecture and give its construction based on basic cryptography modules such as ECDSA signature, distributed key generation, and distributed computation. Analysis and experiment results show that our scheme holds protocol security and is more efficient than other threshold ECDSA signature schemes when threshold is less than 200, which makes it applicable to the assets custody scenarios of exchanges.

1. Introduction

The centralized/decentralized exchanges (CEX/DEX) allow cryptocurrency trading among peers in the blockchain system. Compared to the conventional exchanges, the blockchain-based exchanges are decentralized, tamper-proof, and transparent which reduce the cost of trust in transactions. However, the emerging exchanges also come with security risks. Firstly, the unregulated exchanges could misappropriate funds without the authorization of users. In general, users transfer their digital assets to a specified account which is completely under the control of the CEX. Then the CEX will utilize the assets for investment on behalf of users. One of the problems of the mechanism is that the exchanges could embezzle assets for other purposes, even run away. Moreover, with considerable amount of funds kept in the CEX, they are highly vulnerable to the hacker attacks. One of the largest exchanges, Bitfinex, suffered a loss of 72 million US dollars due to the security breach [1]. Thus, the CEX requires complex access structure for spending the funds. The account should be controlled by multiple trustees instead of a single trustee. Only with the sufficient authorizations of trustees that meet a certain threshold, the account can be operated. In addition, the collusion of trustees should also be considered. In order to meet these requirements, the dynamic threshold scheme is adopted.

The threshold schemes (secret sharing) [2,3] are designed to share a secret among n individuals in cooperation. The secret is divided into n pieces and will be revealed only if gathering more than $k-1$ pieces (where k is less than n). The secret will not be reconstructed even if collecting $k-1$ pieces. In later work, the researchers combine the cryptosystem with these threshold schemes. The newly proposed techniques include threshold signature, threshold decryption, etc. [4–8]. The threshold signature schemes (TSS) generate a signature for a message instead of k signatures. Moreover, the verification of the threshold signature is compatible with that in conventional cryptosystem. Therefore, the threshold ECDSA signature schemes [9–13] have re-entered into the view of the public due to the advantages inherited from threshold schemes and compatibility with the blockchain transaction system. Instead of a single private key, the threshold ECDSA signature schemes require at least k private key pieces to unlock the transactions, providing a joint control on the cryptocurrency.

To address the issue of collusion attack, a better way is to select a group of trustees in a random manner. The corresponding protocol

* Corresponding author.
E-mail addresses: wanghuili@cesi.cn (H. Wang), wpma@mail.xidian.edu.cn (W. Ma), dengfy@buaa.edu.cn (F. Deng), zhenghaibin29@buaa.edu.cn (H. Zheng), qianhong.wu@buaa.edu.cn (Q. Wu).

https://doi.org/10.1016/j.jisa.2021.102805

Available online 13 August 2021

could be run in a smart contract as follow. Firstly, the participants register in the contract, collateralize certain amount of assets to the contract and become potential trustees. Then, the contract will run a random selection algorithm to determine a group of trustees. The trustees in the group will be replaced periodically. Once the newly trustees are determined, the previous group of trustees will be revoked. In order to meet the requirements, we apply the dynamic property to the threshold ECDSA signature schemes.

We proposed a dynamic threshold ECDSA signature scheme that is compatible with the blockchain transaction system to best ensure the security of the digital assets. In our scheme, the trustees could join or drop out the trustee group dynamically. The secret shares of revoked trustees are no longer available to sign transactions, which are replaced by the shares of newly joined participates.

1.1. Related work

The blockchain-based transaction system is the foundation of most blockchain applications. The original blockchain technology such as Bitcoin [14] is designed for transfer of value among distributed peers. The transfer is initiated by transactions and will be approved on chain only if the signatures of transactions are verified valid. Thus, the signature is an authorization of spending the assets of an account. Then, Ethereum [15] improves the functionality of Bitcoin by enabling the transactions to execute smart contracts. As a result, blockchain can be further applied to more complex application scenarios such as Internet of Things (IoT), smart cities, and decentralized finance (DeFi) [16–18]. Moreover, due to a large amount of value stored on chains, the considerations of security and privacy issues of blockchain applications based on transaction systems are required [19].

In order to apply threshold ECDSA signature schemes to the blockchain system, several researchers have spent efforts on solving the efficiency and scalability challenges. Mackenzie and Reiter [9] built a two-party signature scheme. However, the 2-out-of-2 scheme is not practical to blockchain applications. Motivated by the applications of bitcoin, Gennaro et al. [10–12] devise an optimal threshold signature scheme for ECDSA which is best applicable for the world of bitcoin. Firstly, it generalized the Mackenzie and Reiter's 2-of-2 signature case, providing a t-of-n case. Secondly, the Gennaro et al.'s scheme is efficient in computation time and rounds of interactions. Moreover, inspired by Gennaro et al. Lindell and Nof [13] improved the distributed key generation and signing phase of the threshold ECDSA signature scheme by replacing the Paillier key generation.

Dikshit and Singh [20,21] ensure the security of bitcoin wallets by introducing weight property to threshold schemes. The original idea of [20] is that the number of secret share pieces hold by the players is based on their weightage. In later work [21], the authors improve their scheme by introducing groups with different weightage. However, the above weighted threshold schemes increase the complexity of the system, especially in the setup phase. Moreover, with more secret shares generated, the shares are more prone to be exposed to the attackers. Furthermore, the distributed negotiation of the weightage of the participants may also decrease the efficiency of the system.

Introducing dynamic attributes into threshold signature schemes is another challenging topic for researchers. The dynamic properties may vary depending on the specific applications. One of the dynamic definitions is that the threshold could be dynamically selected to generate different signatures. Lee [22] proposed a threshold signature scheme that has multiple signing policies. The group members could reconstruct various group secrets with different thresholds of secret shares provided. Li et al. [23] proposed a dynamic threshold signature scheme based on Lee's scheme. The authors devise a verification protocol to validate the group signatures which avoids the impersonation attack. However, the setup is based on finding multiple large primes for groups. The efficiency of the scheme may be low when the group members increase. Luo and Zhang [24] further improve the efficiency issue of [23] by introducing the bilinear pairing. To sum up, the above schemes are applicable to the applications that require different levels of security. The more signers that sign the message, the more credible the message is. However, it does not apply to the exchanges scenarios. Secondly, the verification of multiple group signatures is not compatible with blockchain systems. Moreover, these methods are all rely on secure trusted setups which are not applicable for the decentralized situations.

Noack and Spitz [25] first proposed a threshold scheme that has the ability to add and remove members without modifying the overall secret key. The proposed scheme without the setup of trusted third party (TTP) is best suitable for distributed and dynamic networks. Later, Kubilay et al. [26] applied the dynamic scheme to improve the block proposal of PBFT consensus in their novel PKI architecture. However, the ElGamal-based scheme of Noack and Spitz is not compatible with the ECDSA-based verification mechanisms of the current blockchain transaction system. Thus, the dynamic ElGamal-based threshold scheme is not applicable to the transaction-based applications such as exchanges.

1.2. Our results

We propose a dynamic threshold ECDSA signature scheme which is compatible to the verification mechanisms of current blockchain transaction systems. As best to our knowledge, we are the first to apply the dynamic threshold ECDSA signature scheme to ensure the security of assets custody in the exchanges. The main features of our scheme are as follow.

- The overall signing key of the trustee group remains unchanged when trustees dynamically join or drop off. The fixed signing key guarantees the operation on a same account regardless of the rotation of trustees which is best applicable to the custody scenario of exchanges.
- Our solution could further avoid the collusion attack of trustees by cooperating with smart contracts that run random trustee election algorithms.
- Compared to other ECDSA threshold signature schemes, our scheme is more efficient in signature generation of low thresholds by utilizing the BGW protocol.
- The security of our scheme is the same as the other ECDSA threshold signature schemes while holding the dynamic feature.

2. Preliminaries

2.1. ECDSA signature

Setup. Select an elliptic curve E over $\mathbb{F}_q$, where $q = 2^m$. Thus, We get a set of ECDSA domain parameters $D = (E, \mathbb{F}_q, G, n)$, where G is the generator on the curve and n is a large prime that divides q. For security reason, the selection and verification of ECDSA domain parameters must follow the standard [27] strictly.

KeyGen. Generate a public key based on the domain parameters chosen in the setup phase.

1. Select a random integer $x \in [1, n-1]$ as secret key
2. Compute the public key $Y = xG$

SigGen. Sign a message m by the key generated above.

1. Select a random integer $k \in [1, n-1]$
2. Compute $kG = (x_1, y_1)$
3. Compute $r = x_1 \mod n$
4. Compute $h = H(m)$, where H is a one-way hash function
5. Compute $s = k^{-1}(h + xr) \mod n$
6. The signature (r, s) is a tuple of r and s

Verification. The individuals receive the payload $(r, s) \parallel m$ and verify the signature as follows:

1. Check integers r and $s \in [1, n-1]$
2. Compute $h = H(m)$
3. Verify the signature by computing

$$X = hs^{-1}G + rs^{-1}Y \pmod{n}$$

4. The signature is accepted, iff $x_1 \equiv r \mod n$, where x_1 is the x coordinate of X

2.2. Distributed Key Generation

The Distributed Key Generation (DKG) is proposed in [28] for describing the key generation phase of the protocol designed by Pedersen [7,29]

Setup. G_q is the subgroup of $\mathbb{F}_p$ of order q, where p, q are both large primes, q divides $p-1$ and g is the generator of $\mathbb{F}_p$. We denote a group of n members as $(P_1, P_2, \dots, P_n)$.

KeyGen. The threshold public key h is constructed by the share public keys of all the members.

1. P_i computes its share public key $h_i = g^{x_i}$ and generates a random string r_i, where $x_i \in \mathbb{Z}_q$ at random
2. P_i broadcasts a commitment $C_i = C(h_i, r_i)$ to all members, where $C(m, r)$ is a commitment tuple of a share public key and random string.
3. Every member computes the threshold public key $h = \prod_{i=1}^{n} h_i$, thus the threshold secret key $x = \sum_{i=1}^{n} x_i$, where $x = \log_g h$.

Interaction. P_i shares its polynomial $f_i(z)$ to all the members without revealing the coefficients.

1. Construct a random polynomial $f_i(z) \in \mathbb{Z}_q$ of degree $k-1$ such that free coefficient $f_i(0) = x_i$. Let

$$f_i(z) = f_{i0} + f_{i1}z + \cdots + f_{i,k-1}z^{k-1}$$

where $f_{i0} = x_i$.
2. Compute $F_{ij} = g^{f_{ij}}$, where $j = 0, 1, \dots, k-1$
3. Broadcast $(F_{ij})_{j=0,1,\dots,k-1}$ to all members $(F_{i0} = h_i)$

Verification. P_j verifies the correctness of $(F_{ij})_{j=0,1,\dots,k-1}$ and s_{ij} sent from P_i.

1. P_i computes $s_{ij} = f_i(j) = f_{i0} + f_{i1}j + \cdots + f_{i,k-1}j^{k-1}$
2. P_i sends s_{ij} with corresponding signature to P_j through a secure channel
3. P_j verifies the signature first, then compares $(F_{ij})_{j=0,1,\dots,k-1}$ sent from P_i by

$$g^{s_{ij}} = \prod_{l=0}^{k-1} (F_{il})^{j^l}$$

If the condition is not satisfied, the interaction will end. And P_j will broadcast the error to all members.

Secret Computation. By defining $f(z) = f_1(z) + f_2(z) + \cdots + f_n(z)$, P_i could compute $s_i = \sum_{j=1}^{n} s_{ij} = f(i)$. Thus s_i is a share of $f(0) = x$

1. P_i computes his share $s_i = \sum_{j=1}^{n} s_{ij}$
2. P_i collects over $k-1$ shares in total and computes the secret through Lagrange interpolation

$$x = \sum_{i=1}^{k} s_i \prod_{j=1}^{k} \frac{j}{j-i}$$

Therefore, collecting over $k-1$ shares s_i, we could reconstruct $f(z)$ and compute the secret $x = f(0)$

2.3. Distributed computation

We apply the theorem 1 of Ben-Or, Goldwasser, and Wigderson (BGW) [30] to solve the multiplying issue of multiple secrets. Take the following case as an example, the secret x is defined as the multiplication of two secrets $x = pq$. We denote a group of n members as $P = (P_1, P_2, \dots, P_n)$. The secrets p, q are shared among group P and can be constructed through the protocol in Section 2.2 independently. Thus, we have

- $h(z), g(z)$ are defined as random polynomials of degree $k-1$ over $\mathbb{Z}_q$ such that $h(0) = p$, $g(0) = q$
- $p = \sum_{i=1}^{n} p_i$ and $q = \sum_{i=1}^{n} q_i$
- (p_i, q_i) are the share tuple of member P_i, where $h(\alpha_i) = p_i$, $g(\alpha_i) = q_i$, and α_i is the x coordinate selected by P_i
- Define $f(z) = h(z)g(z) = f_{i0} + f_{i1}z + \cdots + f_{i,2(k-1)}z^{2(k-1)}$, where $f(0) = x$.

Randomization. The coefficients of the polynomial should follow the uniform distribution [2]. Therefore, the randomization of the coefficients of $f(z)$ is required.

1. To meet the requirement of Randomization, the number of the members should satisfy $n > 2k-1$.
2. P_i constructs a random polynomial $\theta_i(z) \in \mathbb{Z}_q$ of degree $2(k-1)$. Let $\theta_i(z) = \theta_{i1}z + \cdots + \theta_{i,2(k-1)}z^{2(k-1)}$ with a zero free coefficient.
3. Follow the interaction procedure described in Section 2.2. Then P_i could compute the share $d_i = \theta(i)$, where $\theta(z)$ is defined as $\theta(z) = \theta_1(z) + \theta_2(z) + \cdots + \theta_{2(k-1)}(z)$.
4. Define the random polynomial $\widetilde{f}(x) = f(x) + \theta(x)$, where $\widetilde{f}(0) = x$. Thus, $\widetilde{f}(x)$ follows the uniform distribution.

Degree Reduction. By reducing the degree of $\widetilde{f}(z)$ from $2(k-1)$ to $k-1$, the $k-threshold$ requirement is met.

1. We define $\widetilde{k}(z)$, $k(z)$, and $\Theta(z)$ of degree $k-1$ are the truncation of $\widetilde{f}(z)$, $f(z)$, and $\theta(z)$ respectively.
2. Take $\widetilde{f}(z)$ as an example, we denote P_i's shares of $\widetilde{f}(0)$ and $\widetilde{k}(0)$ as $\widetilde{s}_i = \widetilde{f}(\alpha_i)$ and $\widetilde{r}_i = \widetilde{k}(\alpha_i)$. Let $S = (\widetilde{s}_0, \widetilde{s}_1, \dots, \widetilde{s}_{2(k-1)})$ and $R = (\widetilde{r}_0, \widetilde{r}_1, \dots, \widetilde{r}_{2(k-1)})$, then S and R have the following relationship

$$R = S \cdot A$$

where A is a constant $(2k-1) \times (2k-1)$ matrix.

Secret Computation. With all the above prerequisites, every member could compute the secret x through Lagrange interpolation as follows:

1. Aggregate over k shares of s_i and d_i, where $s_i = p_i q_i$
2. Compute $(b_i)_{i \in 0,1,\dots,n}$ by

$$(b_0, b_1, \dots, b_n) = (d_0, d_1, \dots, d_n) \cdot B'$$

where b_i is a share of $\Theta(0)$ and B' is a constant matrix related to the polynomial $\theta(z)$. If the corresponding share d_i is not received, its value is set to 0. And the indexes of d_i (where $d_i \neq 0$) construct the set $\boldsymbol{V}$.
3. Compute $(r_i)_{i \in 0,1,\dots,n}$ by

$$(r_0, r_1, \dots, r_n) = (s_0, s_1, \dots, s_n) \cdot A'$$

where r_i is a share of $k(0)$ and A' is a constant matrix related to the polynomial $f(z)$. If the corresponding share s_i is not received, its value is set to 0.
4. Compute secret $x = \widetilde{k}(0)$, where

$$\widetilde{k}(0) = k(0) + \Theta(0)$$

$$\widetilde{k}(0) = \sum_{i \in \boldsymbol{V}} r_i \prod_{j \in \boldsymbol{V}, j \neq i} \frac{\alpha_j}{\alpha_j - \alpha_i} + \sum_{i \in \boldsymbol{V}} b_i \prod_{j \in \boldsymbol{V}, j \neq i} \frac{\alpha_j}{\alpha_j - \alpha_i}$$

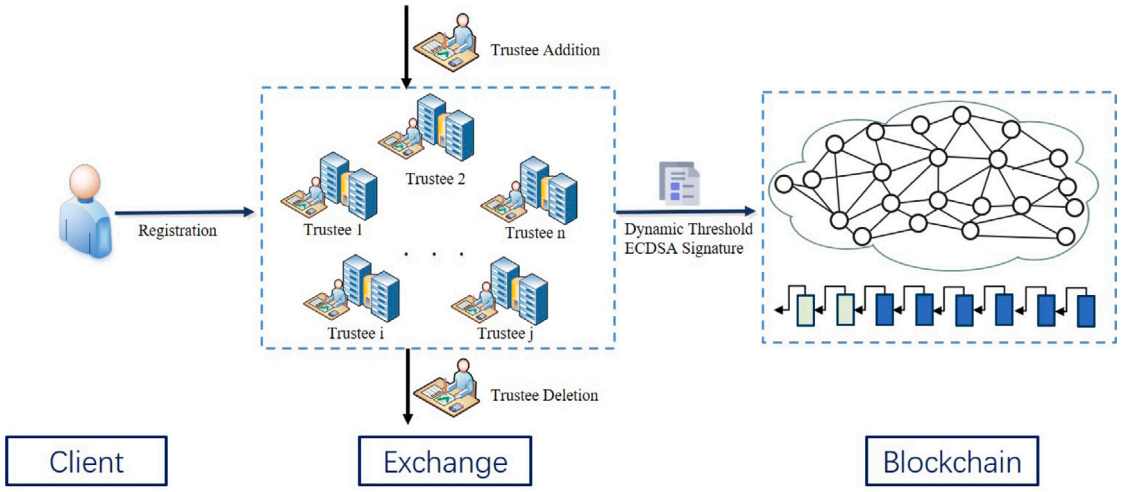

Fig. 1. System Architecture of Blockchain System with Dynamic Threshold ECDSA Signature.

3. Blockchain system with dynamic threshold ECDSA signature

In this section, we give an overview of blockchain-based transaction system with dynamic threshold ECDSA signature, including motivating scenario, system architecture, and property requirements.

3.1. Motivating scenario

Digital currency exchanges based on blockchain technology are divided into centralized exchange and decentralized exchange. In a centralized exchange environment, users usually transfer assets to an account designated and controlled by the exchange, and then enter the exchange for trading. The exchange conducts matching transactions, but the transaction is taking place only when the transaction is over and the coin is withdrawn. This brings many problems. The exchange can arbitrarily control the user's assets or even run away with the money. Based on the feature of blockchain that does not rely on third-party, decentralized exchange provides a new credit mechanism and value exchange model. In the decentralized exchange environment, although the user's encrypted assets really belong to himself and the user can truly control it, it also needs to rely on the exchange for matching transactions. They both have certain asset management problems.

One solution to this problem is that, the account is controlled by multiple trustees, and only the consent that meets a certain threshold can be operated on the account. This can be achieved by using the threshold ECDSA protocol. But at the same time, new problems will arise, that is, trustees may collude. A better way is to generate the trustee of the account in a random way. In this way, the threshold ECDSA is required to support the dynamic addition and deletion of trustees. The deleted trustee no longer has the ability to participate in the control of the account, and his rights are replaced by the new added trustee. The dynamic threshold ECDSA scheme can efficiently support the above functions. This is also the original intention of proposing the concept of dynamic threshold ECDSA in this paper.

Considering the two scenarios of centralized exchange and decentralized exchange, in centralized exchange scenarios, dynamic threshold ECDSA scheme can perfectly fit and apply to solve the problem of randomly generated trustees in asset custody. In decentralized exchange environment, randomly generated trustees can register in a smart contract because of the distributed transactions are implemented based on smart contracts. They pledge certain assets to become potential trustees, and then run a random selection algorithm every certain time to choose a new trustee and revoke the original trustee. In this paper, we focus on the dynamic threshold ECDSA scheme under the centralized trading exchange.

3.2. System architecture

The blockchain system with dynamic threshold ECDSA signature consists of three sub-modules: asset client $Client$, centralized exchange $Exchange$, and blockchain system $Blockchain$. The system architecture is depicted in Fig. 1.

- Asset Client $Client$, is the initiator of the system. It first registers to the centralized exchange and obtains the address assigned by the exchange. When he recharges to his own exchange address, all assets will be automatically transferred to the exchange address. When he submits a trading instruction to the exchange, the exchange executes the transaction matching operation.
- Centralized Exchange $Exchange$, is the core of the system. In addition to the traditional centralized exchange functions that issues transaction addresses to clients and matches transactions, the exchange has multiple distributed trustees, which are implementation nodes of dynamic threshold ECDSA signatures. Among the n trustee nodes, the exchange can perform transaction operations on the account only if more than t nodes complete the signature at the same time, for ensuring the security of client assets. Moreover, the n trustees can add and delete dynamically, which avoids the risk of possible trustee collusion.
- Blockchain System $Blockchain$, is the execution of the system. Similar to the operation of traditional blockchain transaction system, after receiving the transaction jointly signed by t trustees, the miner nodes in the system verify the validity of the transaction that broadcasted to the blockchain network, and package it to generate new block.

Compared to traditional centralized exchange, the blockchain system with dynamic threshold ECDSA signature offers a new way to protect the asset security of transaction users. It translates the transaction signed by one trustee of centralized exchange into transaction signed by several distributed trustees, making client accounts controlled by multiple trustees and avoiding assets dominated by the centralized exchange arbitrarily.

This system architecture could achieve a transaction mechanism allowing trustee user to join and drop out dynamically to resist collusion attacks. This system could prevent the exchange from arbitrarily control user's assets or even run away with the money. It focuses more on the privacy protection of transaction users, and cannot achieve stronger improvements in the throughput and security of the blockchain system.

3.3. Property requirements

The blockchain system with dynamic threshold ECDSA signature we proposed is designed to achieve the following four property requirements: distribution, dynamic, security and robustness.

Distribution. Our system aims to achieve distributed delegation of assets. The client's asset is controlled by multiple distributed trustees simultaneously, and the account can be operated only when a certain threshold achieved. No trustee knows the shared private key and they only holds their own secret shares. This feature can effectively avoid the single-node threat of the exchange, prohibiting it from arbitrarily controlling clients' assets, or even running out of cash.

Dynamic. Our system aims to achieve dynamic stability of trustees. Fixed trustee structure may have the risk of trustees collusion. A better way is to generate these trustees in a random manner, which satisfies the dynamic addition and deletion. The revoked trustee will no longer owns the ability to participate in the control of assets, and his some rights are replaced by the new added trustee. But at the same time, original shared public key and private key remain unchanged.

Security. Our system aims to achieve protocol security. The proposed dynamic threshold ECDSA signature protocol mainly focus on the privacy protection of transaction users through the joint signature of multiple trustees. According to the security model of signature scheme, the security service here mainly refers to the unforgeability of threshold ECDSA signature. The attackers of this scheme include the third-party attackers from outside the system and the collusion attackers from dishonest trustees inside the system.

Robustness. Our system aims to achieve protocol robustness. The system reliability depends on (t, n) threshold characteristics. Only more than t trustees can construct a valid signature on the transaction to realize the verification of the miner nodes. This system can at most tolerate $t-1$ corruption trustee nodes.

4. Dynamic threshold ECDSA signature scheme

In this section, we give an introduction of dynamic threshold ECDSA signature, including system definition and concrete implementation. Then we give the correct analysis, security analysis and performance analysis of the concrete scheme.

4.1. System definition

Definition 1 (*Dynamic Threshold ECDSA (DT-ECDSA)*)**.** A dynamic threshold ECDSA mainly consists of the following six algorithms: system initialization $Setup$, distributed key generation $DKeyGen$, threshold signature $TSign$, public verification $Verify$, user dynamic addition $UserAdd$ and user dynamic deletion $UserDelete$.

$\boldsymbol{params} \leftarrow \boldsymbol{Setup}(\mathbf{1}^{\boldsymbol{k}})$: Runs to initialize system. It takes as input a security parameter 1^k, and outputs the system public parameter $params$.

$\{(\boldsymbol{upk_i}, \boldsymbol{usk_i}), \boldsymbol{d_i}, \boldsymbol{mpk}\} \leftarrow \boldsymbol{DKeyGen}(\boldsymbol{params})$: Runs by trustee user node $U_i, i = 1, \ldots, n$ to generate their respective public–private key pairs $(upk_i, usk_i, i = 1, \ldots, n)$, shared signature key share $(d_i, i = 1, \ldots, n)$ and public signature verify key mpk. It takes as input system public parameter $params$, and outputs $(upk_i, usk_i, i = 1, \ldots, n)$, $(d_i, i = 1, \ldots, n)$, mpk.

$\boldsymbol{\sigma} \leftarrow \boldsymbol{TSign}(\boldsymbol{params}, \boldsymbol{m}, (\boldsymbol{d_i}, \boldsymbol{i} = \mathbf{1}, \ldots, \boldsymbol{t}))$: Runs by trustee user node $U_i, i = 1, \ldots, t$ among $U_i, i = 1, \ldots, n$ to generate an ECDSA signature σ. It takes as input system public parameter $params$, message m and shared signature key share $(d_i, i = 1, \ldots, n)$, and outputs ECDSA signature σ.

$\boldsymbol{(0,1)} \leftarrow \boldsymbol{Verify}(\boldsymbol{params}, \boldsymbol{m}, \boldsymbol{\sigma}, \boldsymbol{mpk})$: Runs by verify user node $VerUser$ to verify the validity of the signature σ. It takes as input system public parameter $params$, message m, signature σ and shared signature verify public key mpk, and outputs 1 if and only if the signature is valid, otherwise outputs 0.

$\boldsymbol{d_r} \leftarrow \boldsymbol{UserAdd}(\boldsymbol{params}, \langle \boldsymbol{U_r}(\boldsymbol{ID_r}), \boldsymbol{U_i}(\boldsymbol{ID_i}) \rangle)$: Runs by new added trustee user node U_r and primite trusteee user node $U_i, i = 1, \ldots, n$ to interactively generate new added trustee user node's shared signature key share d_r. It takes as input their respective identifier $ID_r, ID_i, i = 1, \ldots, n$, and outputs d_r.

$\tilde{\boldsymbol{d}}_{\boldsymbol{i}} \leftarrow \boldsymbol{UserDelete}(\boldsymbol{params}, \langle \boldsymbol{U_w}(\boldsymbol{ID_w}), \boldsymbol{U_i}(\boldsymbol{ID_i}) \rangle)$: Runs by deleted trustee user node U_w and primite trustee user node $U_i, i = 1, \ldots, n, i \neq w$ to interactively generate new trustee user node's shared signature key share $\tilde{d}_i, i = 1, \ldots, n, i \neq w$. It takes as input their respective identifier $ID_w, ID_i, i = 1, \ldots, n, i \neq w$, and outputs $\tilde{d}_i$.

4.2. DT-ECDSA construction

According to the system definition and related building blocks introduced above, we can construct the specific dynamic threshold ECDSA signature scheme.

Define $\Sigma = (Setup, DKeyGen, TSign, Verify, UserAdd, UserDelete)$ represents the DT-ECDSA signature scheme.

Setup. $\mathbb{F}_q$ is a finite field, E is an elliptic curve over $\mathbb{F}_q$. G is a generator on the curve E and its order is prime q. a, b are the coefficients of the elliptic curve, and $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is the one-way hash function. The system parameter is $params = (\mathbb{F}_q, E, G, q, a, b, H)$.

DKeyGen. Suppose the mutually independent trustee user nodes are $U_i, i = 1, \ldots, n$, each node discloses its own unique identifier ID_i.

Here, trustee user nodes $U_i, i = 1, \ldots, n$ generate their respective public–private key pairs $(upk_i, usk_i, i = 1, \ldots, n)$, and generate shared signature key share $(d_i, i = 1, \ldots, n)$, shared secret signature private key d and public signature verify key mpk using distributed key generation algorithm. It means, at least t users $U_i, i = 1, \ldots, t$ cooperate to calculate the shared secret signature private key d and public signature verify key mpk.

1. Every trustee user node $U_i, i = 1, \ldots, n$ randomly chooses $x_i \in [1, q-1]$ as its private key usk_i, computes $y_i = x_i G$ as its public key upk_i, and broadcasts the public key y_i.
2. Every trustee user node $U_i, i = 1, \ldots, n$ randomly chooses $a_{i,1}, a_{i,2}, \ldots, a_{i,t-1} \leftarrow \mathbb{Z}_q^*$, generates a polynomial $f_i(x)$ of degree $t-1$.

   $$f_i(x) = x_i + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1} \mod q$$

   $$f_i(0) = x_i$$

   Trustee user node U_i computes the corresponding commit $C_{i,k}$ and broadcasts it.

   $$C_{i,k} = a_{i,k}G \mod q, k = 1, \ldots, t-1$$

   At the same time, U_i computes sub-secret share

   $$s_{i,j} = f_i(ID_j)$$

   for other $n-1$ members $U_j, j = 1, \ldots, n, j \neq i$, and sends it to U_j via secure channel.
3. After received the sub-secret share $s_{i,j}$ from U_i, trustee user node U_j verifies whether the sub-secret share is valid by the following equation.

   $$s_{i,j}G = \sum_{k=1}^{t-1} (ID_j)^k C_{i,k} + y_i$$

   If the above equation succeeds, then accepts the message of U_i. Otherwise, U_j rejects and broadcasts a complaint against U_i.
4. After verifying the sub-secret shares generated by all other nodes, trustee user node U_j uses the following equation to generate its own secret share.

   $$d_j = \sum_{i=1}^{n} s_{i,j} = \sum_{i=1}^{n} f_i(ID_j)$$

   And broadcasts its shadow value $d_j G$.

5. Define $F(x) = \sum_{i=1}^{n} f_i(x)$, then

$$d_j = F(ID_j)$$

$$d = F(0) = \sum_{i=1}^{n} x_i$$

For any t trustee user nodes $U_i, i = 1, \ldots, t$, they can recover the shared secret signature private key d by using Lagrange interpolation algorithm and calculate the public signature verify key mpk.

$$d = F(0) = \sum_{i=1}^{t} d_i \prod_{j=1, j\neq i}^{t} \frac{ID_j}{ID_j - ID_i} \mod q$$

$$mpk = Q = dG = \sum_{i=1}^{t} d_i \prod_{j=1, j\neq i}^{t} \frac{ID_j}{ID_j - ID_i} G \quad (1)$$

TSign. Trustee user node $U_i, i = 1, \ldots, t$ among $U_i, i = 1, \ldots, n$ generate an ECDSA signature σ by using shared signature key share $(d_i, i = 1, \ldots, t)$ to message m. It is equivalent to the signature calculated by using shared secret private key d.

1. Trustee user node $U_i, i = 1, \ldots, t$ randomly chooses $k_i \in [1, q-1], i = 1, \ldots, t$, computes $k_i G$ and sends $k_i G$ to $U_j, j = 1, \ldots, t; j \neq i$.
2. After received $k_j G, j = 1, \ldots, t; j \neq i$, U_i lets $k = \sum_{i=1}^{t} k_i$, computes

$$\sum_{i=1}^{t} k_i G = kG = (\bar{x}_1, \bar{y}_1)$$

$$r = \bar{x}_1 \mod q$$

If $r = 0$, then returns to Step 1.
3. $U_i, i = 1, \ldots, t$ computes $e = H(m)$.
4. $U_i, i = 1, \ldots, t$ randomly chooses b_i, uses BGW distributed computing algorithm to compute

$$\varphi = (\sum_{i=1}^{t} k_i)(\sum_{i=1}^{t} b_i) \mod q$$

5. $U_i, i = 1, \ldots, t$ then performs BGW distributed computing algorithm to generate a signature. It computes

$$e_i = (et^{-1}) \mod q$$

$$c_i = (\prod_{j=1, j\neq i}^{t} \frac{ID_j}{ID_j - ID_i}) \mod q$$

$$w_i = (e_i + d_i c_i r) \mod q$$

then computes

$$s = \varphi^{-1}(\sum_{i=1}^{t} b_i)(\sum_{i=1}^{t} e_i + \sum_{i=1}^{t} d_i c_i r) \mod q$$

$$= \varphi^{-1}(\sum_{i=1}^{t} b_i)(\sum_{i=1}^{t} w_i) \mod q$$

If $s = 0$, then returns to Step 1.
Finally, returns $\sigma = (r, s)$ as the signature.

Verify. After given $\sigma = (r, s)$, verify user node $VerUser$ could verifies the validity of the signature σ.

1. Verify user node $VerUser$ computes $e = H(m)$, $w = s^{-1} \mod q$.
2. $VerUser$ computes $u_1 = ew \mod q$, $u_2 = rw \mod q$, $u_1 G + u_2 Q = (\bar{x}_0, \bar{y}_0)$.
3. $VerUser$ computes $v = \bar{x}_0 \mod q$.
4. If $v = r$, the verification succeeds, otherwise the verification fails.

UserAdd. New added trustee user node U_r and primite trustee user node $U_i, i = 1, \ldots, n$ interactively generate new added trustee user node's shared signature key share d_r. When new trustee user node added, the following conditions must be satisfied: (1) The shared signature key share of existing nodes $(d_i, i = 1, \ldots, n)$ will not be changed or disclosed. (2) The joining of new node will not change the public key Q. (3) New node can generate its own shared signature key share d_r and cooperate to generate the public key.

Assuming the identifier of the new added trustee user node U_r is ID_r, the specific steps are as follows.

1. Primite trustee user node $U_i, i = 1, \ldots, n$ randomly chooses $b_{i,0}, b_{i,1}, b_{i,2}, \ldots, b_{i,t-1} \leftarrow \mathbb{Z}_q^*$, generates a polynomial $g_i(x)$ of degree $t-1$ which satisfies $g_i(ID_r) = 0$.

$$g_i(x) = b_{i,0} + b_{i,1}x + b_{i,2}x^2 + \cdots + b_{i,t-1}x^{t-1} \mod q$$

$$g_i(ID_r) = 0$$

Trustee user node U_i computes the corresponding commit $C'_{i,k}$ and broadcasts it.

$$C'_{i,k} = b_{i,k} G \mod q, k = 0, 1, \ldots, t-1$$

At the same time, U_i computes sub-secret share

$$s'_{i,j} = g_i(ID_j)$$

for other $n-1$ members $U_j, j = 1, \ldots, n, j \neq i$, and sends it to U_j via secure channel.
2. After received the sub-secret share $s'_{i,j}$ from U_i, trustee user node U_j verifies whether the sub-secret share is valid by the following equation.

$$s'_{i,j} G = \sum_{k=0}^{t-1} (ID_j)^k C'_{i,k}$$

If the above equation succeeds, then accepts the message of U_i. Otherwise, U_j rejects and broadcasts a complaint against U_i.
3. After verifying the sub-secret shares generated by all other nodes, trustee user node U_j uses the following equation to generate its temporary secret share.

$$d'_j = d_j + \sum_{i=1}^{n} g_i(ID_j) \mod q$$

$$= F(ID_j) + \sum_{i=1}^{n} g_i(ID_j) \mod q \quad (2)$$

And sends this temporary secret share d'_j to new added trustee user node U_r via secure channel.
4. After received the temporary secret share d'_j from at least t members $U_i, i = 1, \ldots, t$, node U_r can calculate its own secret share d_r by using Lagrange interpolation algorithm and calculate the public signature verify key mpk.

$$d_r = F(ID_r) = \sum_{i=1}^{t} d'_i \prod_{j=1, j\neq i}^{t} \frac{ID_j - ID_r}{ID_j - ID_i} \mod q \quad (3)$$

Then, the new added trustee user node U_r obtains its own secret share d_r without knowing the secret share of other members, and the public signature verify key mpk and the shared secret signature private key d have not been changed or exposed.

UserDelete. Primite trustee user nodes $U_i, i = 1, \ldots, n-1$ interactively generate new trustee user nodes' shared signature key share $(\tilde{d}_i, i = 1, \ldots, n-1)$. Deletion operation often occurs in the following two cases: 1) Dishonest members need to be eliminated. 2) Members leave the system by themselves. When new trustee user node deleted, the following features must be satisfied: 1) Reconstruct the secret shares of other reserved members $(\tilde{d}_i, i = 1, \ldots, n-1)$ to invalidate the secret share

of the deleted member. 2) Keep the public key $mpk = Q$ and private key $d = F(0)$ unchanged.

Assuming the deleted trustee user node is U_w whose identifier is ID_w, the specific steps are as follows.

1. Primite trustee user node $U_i, i = 1, \dots, n, i \neq w$ randomly chooses $c_{i,1}, c_{i,2}, \dots, c_{i,t-1} \leftarrow \mathbb{Z}_q^*$, generates a polynomial $h_i(x)$ of degree $t-1$.

   $$h_i(x) = c_{i,1}x + c_{i,2}x^2 + \cdots + c_{i,t-1}x^{t-1} \mod q$$

   Trustee user node U_i computes the corresponding commit $\tilde{C}_{i,k}$ and broadcasts it.

   $$\tilde{C}_{i,k} = c_{i,k}G \mod q, k = 1, 2, \dots, t-1$$

   At the same time, U_i computes sub-secret share

   $$\tilde{s}_{i,j} = h_i(ID_j)$$

   for other $n-2$ members $U_j, j = 1, \dots, n, j \neq i, w$, and sends it to U_j via secure channel.
2. After received the sub-secret share $\tilde{s}_{i,j}$ from U_i, trustee user node U_j verifies whether the sub-secret share is valid by the following equation.

   $$\tilde{s}_{i,j}G = \sum_{k=1}^{t-1}(ID_j)^k\tilde{C}_{i,k}$$

   If the above equation succeeds, then accepts the message of U_i. Otherwise, U_j rejects and broadcasts a complaint against U_i.
3. After verifying the sub-secret shares generated by all other nodes, trustee user node U_j uses the following equation to generate its new secret share.

   $$\begin{aligned}\tilde{d_j} &= d_j + \sum_{i=1,i\neq w}^{n} h_i(ID_j) \mod q \\ &= F(ID_j) + \sum_{i=1,i\neq w}^{n} h_i(ID_j) \mod q \end{aligned} \quad (4)$$

Then, the secret shares of other members will be reconstructed and updated, so the secret share of the deleted node U_w is naturally invalid. And the public signature verify key $mpk = Q$ and the shared secret signature private key $d = F(0)$ have not been changed or exposed.

Here, our system requires that after a member is deleted, other members update their secret shares and start to use the new secret shares to prevent an extreme case of collusion attacks, that is, all previous signers are dishonest nodes.

4.3. Correct analysis

The correctness of this dynamic threshold ECDSA scheme includes three aspects: valid signature, dynamic user addition and dynamic user deletion. They respectively depend on:

(1) the generation of valid ECDSA signature $\sigma = (r, s)$ that could be verified.

(2) new added trustee node U_r can compute valid secret share d_r and calculate public key mpk cooperated with primitive member.

(3) shared secret key $F(0)$ and public key mpk remains unchanged after trustee node U_w deleted.

Specifically, the correctness of this scheme is indicated by the following branches.

- Threshold Signature Correctness

$$\sum_{i=1}^{t} e_i \mod q = \sum_{i=1}^{t} et^{-1} \mod q = e$$

$$\sum_{i=1}^{t} c_i d_i \mod q = \sum_{i=1}^{t}\Big(\prod_{j=1,j\neq i}^{t} \frac{ID_j}{ID_j - ID_i}\Big)d_i \mod q = d$$

$$\varphi^{-1}\Big(\sum_{i=1}^{t} b_i\Big) \mod q = \Big(\Big(\sum_{i=1}^{t} k_i\Big)\Big(\sum_{i=1}^{t} b_i\Big)\Big)^{-1}\Big(\sum_{i=1}^{t} b_i\Big) \mod q$$

$$= \Big(\sum_{i=1}^{t} k_i\Big)^{-1} \mod q = k^{-1} \mod q$$

then,

$$\begin{aligned} s &= \varphi^{-1}\Big(\sum_{i=1}^{t} b_i\Big)\Big(\sum_{i=1}^{t} e_i + \sum_{i=1}^{t} d_i c_i r\Big) \mod q \\ &= k^{-1}(e + dr) \mod q \end{aligned}$$

Thus, $\sigma = (r, s)$ is a valid ECDSA signature.

- User Addition Correctness

$$\begin{aligned} d_j' &= F(ID_j) + \sum_{i=1}^{n} g_i(ID_j) \mod q \\ &= \sum_{i=1}^{n} f_i(ID_j) + \sum_{i=1}^{n} g_i(ID_j) \mod q \\ &= \sum_{i=1}^{n}(f_i(ID_j) + g_i(ID_j)) \mod q \end{aligned}$$

Define

$$\begin{aligned} \eta_i(ID_j) &= f_i(ID_j) + g_i(ID_j) \\ &= (x_i + b_{i,0}) + (a_{i,1} + b_{i,1})x + \cdots + (a_{i,t-1} + b_{i,t-1})x^{t-1} \\ &\quad \mod q \end{aligned}$$

then

$$d_j' = \sum_{i=1}^{n} \eta_i(ID_j) \mod q$$

It can be seen from the above formula that it conforms to the Lagrange interpolation polynomial model, then the new trustee user node U_r can compute its secret share d_r after receiving the temporary secret share d_j' of other t members by formula (3).

Because $g_i(ID_r) = 0$, according to formula (2), we can get

$$\begin{aligned} d_r' &= F(ID_r) + \sum_{i=1}^{n} g_i(ID_r) \mod q \\ &= F(ID_r) + 0 \\ &= F(ID_r) \end{aligned}$$

It can be seen that the temporary secret share d_r' of new trustee user node U_r is equal to secret share d_r. This indicates the function value d_r with ID_r exists in the Lagrange interpolation polynomial determined by the original secret share $d_j, j = 1, 2, \dots, n$. Then the new added node U_r can cooperate with the primitive member to calculate the public key mpk through formula (1) without revealing the secret share of the existing members.

- User Deletion Correctness

Because $h_i(0) = 0$, then according to formula (4), we can get

$$\tilde{d} = \Big(F(0) + \sum_{i=1,i\neq w}^{n} h_i(0)\Big) \mod q = F(0) \mod q = d$$

It can be seen that the shared secret key $F(0)$ remains unchanged after the secret share of the primitive undeleted member reconstructed and updated. At the same time, according to formula (1), it can be seen that the public key mpk has not changed, which is still valid and can be used for signature verification.

4.4. Security analysis

Attack Model. Unforgeability is the most important security requirement for dynamic threshold ECDSA signature. The unforgeable attack model is defined as follows.

Assume probabilistic polynomial-time adversary $\mathcal{A}$ executes the following experiment, where $\mathcal{A}$ may corrupt the signature group member

but not the simulator S, and tries to get one dynamic threshold ECDSA signature.

- Setup phase. Simulator S first runs $Setup$ algorithm and outputs public parameters $params$.
- DKeyGen Query phase. Simulator S calculates $\mathcal{A}$' public–private key pairs $(upk_{\mathcal{A}}, usk_{\mathcal{A}})$, shared signature key share $d_{\mathcal{A}}$ and public signature verify key mpk, then returns these values to the adversary $\mathcal{A}$.
- TSign Query Phase. Simulator S generate ECDSA signature $\sigma = (r, s)$ on message m and shared signature key share $d_{\mathcal{A}}$, then returns these values to the adversary $\mathcal{A}$.
- Forgery Phase. Adversary $\mathcal{A}$ makes signature forgery. $\mathcal{A}$ chooses challenge message m^*, generate forged ECDSA signature σ^* on message m^* based on shared signature key share $d_{\mathcal{A}}$.
- Finally, adversary $\mathcal{A}$ outputs the forged dynamic threshold ECDSA signature σ^*.

If the maximum probability of $\mathcal{A}$ wins in the above attack game is negligible, we say this dynamic threshold ECDSA signature scheme satisfies unforgeability.

Theorem 1. *The blockchain system with dynamic threshold ECDSA signature holds security based on existing blockchain, satisfying signature unforgeability under the Elliptic Curve Discrete Logarithm Problem (ECDLP) assumption and Lagrange Interpolation theorem.*

Proof. Assuming $\mathcal{A}$ is an adversary to attack the new scheme, there are two types of adversary according to different attack targets in the system: third-party attacker and dishonest member inside the trustee group.

Type I Adversary. This type of adversary is a third-party attacker and faked as a legal signer. It intends to forge the value of shared secret share variable needed in the process of threshold signature algorithm $TSign$ to generate a valid signature with other participants. Written as $\mathcal{A}_1$.

Type II Adversary. This type of adversary is a dishonest member inside the trustee group and corrupted some other trustee nodes to generate valid shared private key to forge signature. It intends to corrupt the value of shared secret share variable needed in the process of threshold signature algorithm $TSign$ to generate a valid signature with other participants. Written as $\mathcal{A}_2$.

We utilize the query of adversary $\mathcal{A}$ and the responses of simulator S in the system attack experiment based on provable security theory to build the relationship between them. Based on the security assumptions of basic cryptography theories, we can get the success advantage of adversary $\mathcal{A}$ is negligible.

First, we give the two assumptions of Elliptic Curve Discrete Logarithm Problem (ECDLP) and Lagrange Interpolation theorem, and then give the adversary attack simulation game.

Assumption 1. Suppose $\mathbb{F}_q$ is a finite field, E is an elliptic curve over $\mathbb{F}_q$. G is a generator on the curve E and its order is prime n, $Q \in E(\mathbb{F}_q)$. The Elliptic Curve Discrete Logarithm Problem (ECDLP) assumption states that, given G, Q, no polynomial-time adversary can compute l to satisfy $Q = lG$.

Assumption 2. The Lagrange Interpolation theorem states that, Given a $d-1$ degree polynomial $\zeta(x)$, $\zeta(1), \zeta(2), \ldots, \zeta(d)$ are d points on this polynomial, using Lagrange Interpolation we could compute $\zeta(i)$ for any $i \in \mathbb{Z}_p$.

- For adversary of Type I, $\mathcal{A}_1$, the details of attack–response process are as follows.
  Setup phase. Simulator S first runs $Setup$ algorithm and outputs public parameters $params = (\mathbb{F}_q, E, G, q, a, b, H)$ for the dynamic threshold ECDSA signature unforgeability attack experiment, then sends $params$ to the adversary $\mathcal{A}_1$.
  DKeyGen Query Phase. Adversary $\mathcal{A}_1$ makes queries of DKeyGen oracle, Simulator S calculates $\mathcal{A}_1$' public–private key pairs $(upk_{\mathcal{A}_1}, usk_{\mathcal{A}_1})$, shared signature key share $d_{\mathcal{A}_1}$ and public signature verify key mpk, then returns these values to the adversary $\mathcal{A}_1$.
  TSign Query Phase. Adversary $\mathcal{A}_1$ makes queries of TSign oracle, Simulator S generate ECDSA signature $\sigma = (r, s)$ on message m and shared signature key share $d_{\mathcal{A}_1}$, then returns these values to the adversary $\mathcal{A}_1$.
  Forgery Phase. Adversary $\mathcal{A}_1$ makes signature forgery. $\mathcal{A}_1$ chooses challenge message m^*, generate forged ECDSA signature σ^* on message m^* based on shared signature key share $d_{\mathcal{A}_1}$.
  Finally, adversary $\mathcal{A}_1$ outputs the forged ECDSA signature σ^*.
  For the adversary $\mathcal{A}_1$, in order to succeed in the challenge attack experiment, the random parameter k must be forged, so that the forged signature can be verified. Since k is the sum of parameters k_i randomly selected by t signers, k_i is not public to obtain. Moreover, during the implementation of the protocol, the random parameter k_i has never been reconstructed, so it is very difficult for $\mathcal{A}_1$ to intercept k_iG from somewhere to construct k_i. Because this is Elliptic Curve Discrete Logarithm Problem (ECDLP), and this problem is the basis of the security theory of elliptic curve cryptography and belongs to Non-deterministic Polynomial Complete (NPC) problem.
  Therefore, we can conclude that the probability of a successful signature forgery for $\mathcal{A}_1$ in the signature unforgeability attack experiment is equivalent to solving the ECDLP. Adversary $\mathcal{A}_1$ cannot forge the signature successfully.
- For adversary of Type II, $\mathcal{A}_2$, the details of attack–response process are as follows.
  Setup phase. Simulator S first runs $Setup$ algorithm and outputs public parameters $params = (\mathbb{F}_q, E, G, q, a, b, H)$ for the dynamic threshold ECDSA signature unforgeability attack experiment, then sends $params$ to the adversary $\mathcal{A}_2$.
  Corruption phase. Adversary $\mathcal{A}_2$ can corrupt the members of the trustee group. Here, we assume $\mathcal{A}_2$ is a strong admissible, static attacker. Strong admissible means that $\mathcal{A}_2$ cannot collude with exceed $t-1$ members. Static means that $\mathcal{A}_2$ has determined whom to collude with before the protocol begins. $\mathcal{A}_2$ can get any secret information including shared secret share from the dishonest trustee.
  DKeyGen Query Phase. Adversary $\mathcal{A}_2$ makes queries of DKeyGen oracle, Simulator S calculates $\mathcal{A}_2$' public–private key pairs $(upk_{\mathcal{A}_2}, usk_{\mathcal{A}_2})$, shared signature key share $d_{\mathcal{A}_2}$ and public signature verify key mpk, then returns these values to the adversary $\mathcal{A}_2$.
  TSign Query Phase. Adversary $\mathcal{A}_2$ makes queries of TSign oracle, Simulator S generate ECDSA signature $\sigma = (r, s)$ on message m and shared signature key share $d_{\mathcal{A}_2}$, then returns these values to the adversary $\mathcal{A}_2$. Here, $\mathcal{A}_2$ can make corresponding queries based on the obtained corrupted member information.
  Forgery Phase. Adversary $\mathcal{A}_2$ makes signature forgery. $\mathcal{A}_2$ chooses challenge message $\hat{m}^*$, generate forged ECDSA signature $\hat{\sigma}^*$ on message $\hat{m}^*$ based on shared signature key share $d_{\mathcal{A}_2}$.
  Finally, adversary $\mathcal{A}_2$ outputs the forged ECDSA signature $\hat{\sigma}^*$.
  For the adversary $\mathcal{A}_2$, suppose $T = T_1, T_2, \ldots, T_t$ is a qualified trustee subset. Since the attacker cannot get all the shares of all members in T, he cannot determine $F(x)$ from these points according to Lagrange Interpolation theorem, nor any other point on $F(x)$. Therefore, the attacker cannot recover the secret, and cannot obtain any other information about the shared secret d at the same time.
  Therefore, we can conclude that the probability of a successful signature forgery for $\mathcal{A}_2$ in the signature unforgeability attack experiment is almost impossible. Adversary $\mathcal{A}_2$ cannot forge the signature successfully.

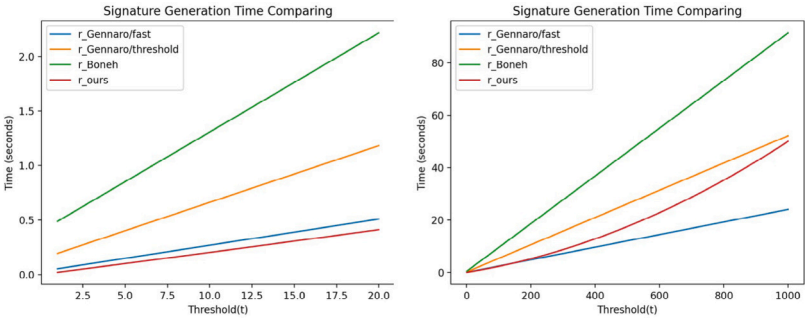

Fig. 2. Signature Runtime Compared with Related Work.

Table 1
Performance analysis of computation time.

Scheme algorithm	Computation time
$DKeyGen$	$(t-1)(2n-1)T_d + \frac{(n-1)t(t-1)}{2}T_m$
$TSign$	$t \cdot T_d + (t^2 - t + 6)T_m + t \cdot T_h$
$Verify$	$2T_d + 3T_m + T_h$
$UserAdd$	$t(2n-1)T_d + \frac{(n-1)t(t-1)}{2}T_m$
$UserDelete$	$(t-1)(2n-3)T_d + \frac{(n-2)t(t-1)}{2}T_m$

Table 2
Functionality comparison with related works.

Scheme	Sig.Gen	Security	Dynamic
[11]	Paillier encryption	unforgeable	×
[31]	Paillier encryption	unforgeable	×
[12]	MtA protocol	unforgeable	×
Ours	BGW protocol	unforgeable	✓

Therefore, the adversary $\mathcal{A}$ can attack the unforgeability of new scheme if the adversary $\mathcal{A}_1$ wins the ECDLP attack, adversary $\mathcal{A}_2$ wins the Lagrange Interpolation attack. The relationship of adversaries' success probability is as follows.

$$
\begin{aligned}
&Adv^{unforge}_{\mathcal{A},DF-\mathcal{ECDSA}}(\lambda)\\
&= Pr[Exp^{unforge}_{\mathcal{A},DF-\mathcal{ECDSA}}(\lambda) = 1]\\
&= Pr[Exp^{unforge}_{\mathcal{A}_1,DF-\mathcal{ECDSA}}(\lambda) = 1]\\
&+ Pr[Exp^{unforge}_{\mathcal{A}_2,DF-\mathcal{ECDSA}}(\lambda) = 1]\\
&= Pr[Exp_{\mathcal{A}_1,DF-\mathcal{ECDSA}}(guess, k_iG) \rightarrow k_i]\\
&+ Pr[Exp_{\mathcal{A}_2,DF-\mathcal{ECDSA}}(guess, d_1, \ldots, d_l) \rightarrow d|l \leq t-1]\\
&= Pr[Exp^{ECDLP}_{\mathcal{A}}(\lambda) = 1] + Pr[Exp^{Lag.Int}_{\mathcal{A}}(\lambda) = 1]\\
&\leq \epsilon_1 + \epsilon_2
\end{aligned}
$$

From the above, regardless of which type adversary $\mathcal{A}$ is, the success advantage is all negligible based on the security assumptions of basic hard problems. □

4.5. Performance analysis

In this section, we analyze the performance of dynamic threshold ECDSA signature scheme in the view of computation time of algorithm process, including distributed key generation $DKeyGen$, threshold signature $TSign$, public verification $Verify$, user dynamic addition $UserAdd$ and user dynamic deletion $UserDelete$. Then we compare the signature runtime with existing related work of Gennaro et al. [11,12] and Boneh et al. [31]. Finally, we give an analysis of signature size and security properties. The results are given in Table 1, Fig. 2 and Table 2 .

As shown in Table 1, we give the runtime consumption of simulation system from each phase. Here, let T_d denotes the time performing a scalar multiplication operation on the elliptic curve E in ECDSA signature algorithm, T_m denotes the time performing a multiplication operation on the group G, and T_h denotes the time performing hash function SHA256. The total number of trustee nodes in the dynamic threshold ECDSA signature scheme is n, the number of valid signature nodes is threshold t. Table 1 shows the time-consuming requirements in each phase increases as t and n increase.

As depicted in Fig. 2, We compare the performance of our scheme to the runtimes of Gennaro et al. [11,12] and Boneh et al. [31]. Here, we mainly focus on the signature generation time, since that the signing time only depends on the number of active nodes t, does not depend on the total number n. We simulate the signature algorithm based on formula $t \cdot T_d + (t^2 - t + 6)T_m + t \cdot T_h$ in Table 1. The simulation was implemented in Java by using a computer of 64 bits Windows 10 with 1 core Intel i7 2.60 GHz and 16 GB RAM. To further illustrate the system performance, we run the system to obtain the time of perform a scalar multiplication on elliptic curve is about 10 ms, a multiplication in group is about 0.03 ms, and the time to perform an hash function is about 0.01 ms.

Then, we get our scheme signature running time is:

$$r_{ours}(t) = 0.03t^2 + 19.98t + 0.18 \quad \text{milliseconds}$$

According to [12], the running time of it is given by:

$$r_{Gennaro/fast}(t) = 29 + 24t \quad \text{milliseconds}$$

The running time of [11] is given by:

$$r_{Gennaro/threshold}(t) = 142 + 52t \quad \text{milliseconds}$$

The running time of [31] is given by:

$$r_{Boneh}(t) = 397 + 91t \quad \text{milliseconds}$$

From Fig. 2, we can see that, although our runtime function is quadratic, its performance is efficient when t is less than 1000, and even better than other schemes when t is less than 200.

As shown in Table 2, considering the signature length, although the signature running time is different between related works [11,12,31] with ours, the signature length of each scheme is the same. The difference of running time is just due to the different signature generation methods used by each scheme. For example, Gennaro et al. [11] and Boneh et al. [31] use the additively homomorphism of Paillier encryption scheme to realize the generation of parameter s in signature σ. Gennaro et al. [12] use the Multiplicative to Additive (MtA) share conversion protocol. We use the distributed computation BGW protocol. Considering the security properties, the security of all above threshold ECDSA signature scheme is unforgeable, but our solution also holds the dynamics and the application scenarios are more diverse.

Moreover, compared with the threshold ECDSA signature scheme based on discrete logarithm, our scheme that using elliptic curve has better efficiency. Let $|l_d|$ represent the key length, and n represents the number of members, each member in the keygen phase based on discrete logarithm needs to broadcast $2n|l_d|$ bits, and each member

in the keygen phase based on elliptic curve needs to broadcast $3n|l_d|$ bits. However, the current cryptosystem based on discrete logarithm requires a key length of at least 1024 bits to ensure security, and the elliptic curve system only needs 160 bits to achieve the same security. According to this extent, each member in the discrete logarithm keygen phase broadcasts 2048n bits, while the elliptic curve keygen phase only needs 480n bits. The communication data of each member is greatly reduced, which improves efficiency and practicability.

5. Conclusion

We have presented a dynamic threshold ECDSA scheme that allows the addition or deletion of a user. The signing key (secret) remains unchanged when users join or leave. Moreover, it is proofed that our scheme is able to avoid the fake signature generated by the adversaries outside ($\mathcal{A}_1$) or inside ($\mathcal{A}_2$) the system. As shown in the performance analysis, our solution is practical and efficient. To sum up, due to the dynamic and high efficiency, our dynamic threshold ECDSA scheme is applicable to secure the digitals assets managed by the exchanges.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper is supported by the National Key R&D Program of China through project 2018YFB0803905, by the Natural Science Foundation of China through projects 61972019, 61932011, 61772538, by the Zhejiang Soft Science Research Program, China 2021C35044.

References

[1] Baldwin C. Bitcoin worth $72 million stolen from bitfinex exchange in hong kong. 2016, https://www.reuters.com/article/us-bitfinex-hacked-hongkong-idUSKCN10E0KP.
[2] Shamir A. How to share a secret. Commun ACM 1979;22(11):612–3. http://dx.doi.org/10.1145/359168.359176.
[3] Blakley GR. Safeguarding cryptographic keys. In: 1979 international workshop on managing requirements knowledge. New York, NY, USA: IEEE; 1979, p. 313–8. http://dx.doi.org/10.1109/MARK.1979.8817296.
[4] Croft R, Harris S. Public-key cryptography and re-usable shared secrets. Cryptogr Coding 1989;189–201.
[5] De Soete M, Quisquater J-J, Vedder K. A signature with shared verification scheme. In: Brassard G, editor. Advances in cryptology — CRYPTO' 89 proceedings. Lecture notes in computer science, New York, NY: Springer; 1990, p. 253–62. http://dx.doi.org/10.1007/0-387-34805-0_23.
[6] Desmedt Y, Frankel Y. Threshold cryptosystems. In: Brassard G, editor. Advances in cryptology — CRYPTO' 89 proceedings. Lecture notes in computer science, New York, NY: Springer; 1990, p. 307–15. http://dx.doi.org/10.1007/0-387-34805-0_28.
[7] Pedersen TP. A threshold cryptosystem without a trusted party. In: Davies DW, editor. Advances in cryptology — EUROCRYPT '91, vol. 547. Berlin, Heidelberg: Springer Berlin Heidelberg; 1991, p. 522–6. http://dx.doi.org/10.1007/3-540-46416-6_47.
[8] Desmedt Y, Frankel Y. Shared generation of authenticators and signatures. In: Feigenbaum J, editor. Advances in cryptology — CRYPTO '91. Lecture notes in computer science, Berlin, Heidelberg: Springer; 1992, p. 457–69. http://dx.doi.org/10.1007/3-540-46766-1_37.
[9] MacKenzie P, Reiter MK. Two-party generation of dsa signatures. In: Annual international cryptology conference. Springer; 2001, p. 137–54.
[10] Goldfeder S, Gennaro R, Kalodner H. Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme. 2015, p. 26.
[11] Gennaro R, Goldfeder S, Narayanan A. Threshold-optimal DSA/ECDSA signatures and an application To Bitcoin wallet security. Tech. Rep. 013, Springer; 2016.
[12] Gennaro R, Goldfeder S. Fast multiparty threshold ECDSA with fast trustless setup. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. Toronto Canada: ACM; 2018, p. 1179–94. http://dx.doi.org/10.1145/3243734.3243859.
[13] Lindell Y, Nof A. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. Toronto Canada: ACM; 2018, p. 1837–54. http://dx.doi.org/10.1145/3243734.3243788.
[14] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008, p. 9.
[15] Buterin V. A next generation smart contract & decentralized application platform. 2013, p. 36.
[16] Ferrag MA, Derdour M, Mukherjee M, Derhab A, Maglaras L, Janicke H. Blockchain technologies for the internet of things: Research issues and challenges. IEEE Internet Things J 2019;6(2):2188–204. http://dx.doi.org/10.1109/JIOT.2018.2882794.
[17] Xie J, Tang H, Huang T, Yu FR, Xie R, Liu J, Liu Y. A survey of blockchain technology applied to smart cities: Research issues and challenges, IEEE Commun Surv Tutor 21(3); 2794–2830. http://dx.doi.org/10.1109/COMST.2019.2899617.
[18] Singh S, Sharma PK, Yoon B, Shojafar M, Cho GH, Ra I-H. Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city. Sustainable Cities Soc 2020;63:102364. http://dx.doi.org/10.1016/j.scs.2020.102364.
[19] Ferrag MA, Shu L, Yang X, Derhab A, Maglaras L. Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges. IEEE Access 2020;8:32031–53. http://dx.doi.org/10.1109/ACCESS.2020.2973178.
[20] Dikshit P, Singh K. Weighted threshold ECDSA for securing bitcoin wallet. TIS 2016;2(6):43–51. http://dx.doi.org/10.19101/TIS.2017.26003.
[21] Dikshit P, Singh K. Efficient weighted threshold ECDSA for securing bitcoin wallet. In: 2017 ISEA Asia security and privacy. Surat, India: IEEE; 2017, p. 1–9. http://dx.doi.org/10.1109/ISEASP.2017.7976994.
[22] Lee N-Y. Threshold signature scheme with multiple signing policies. IEE Proc - Comput Digit Tech 2001;148(2):95–9. http://dx.doi.org/10.1049/ip-cdt:20010206.
[23] Li H, Cai W, Pang L. A Secure dynamic threshold signature scheme, J Comput Res Dev (9).
[24] L-xLuo, Zhang J. Dynamic threshold signature scheme based on bilinear pairing, J Comput Appl (3).
[25] Noack A, Spitz S. Dynamic threshold cryptosystem without group manager. Netw Protoc Algorithms 2009;1(1):108–21. http://dx.doi.org/10.5296/npa.v1i1.161.
[26] Kubilay MY, Kiraz MS, Mantar HA. KORGAN: an efficient PKI architecture based on PBFT through dynamic threshold signatures. Tech. Rep. 1141, 2019.
[27] Johnson D, Menezes A, Vanstone S. The elliptic curve digital signature algorithm (ECDSA). Int J Inf Secur 2001;1(1):36–63. http://dx.doi.org/10.1007/s102070100002.
[28] Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure distributed key generation for discrete-log based cryptosystems. In: Stern J, editor. Advances in cryptology — EUROCRYPT '99. Lecture notes in computer science, Berlin, Heidelberg: Springer; 1999, p. 295–310. http://dx.doi.org/10.1007/3-540-48910-X_21.
[29] Pedersen TP. Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum J, editor. Advances in cryptology — CRYPTO '91, 576. Berlin, Heidelberg: Springer Berlin Heidelberg; 1992, p. 129–40. http://dx.doi.org/10.1007/3-540-46766-1_9.
[30] Ben-Or M, Goldwasser S, Wigderson A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the twentieth annual ACM symposium on theory of computing. New York, NY, USA: Association for Computing Machinery; 1988, p. 1–10. http://dx.doi.org/10.1145/62212.62213.
[31] Boneh D, Gennaro R, Goldfeder S. Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In: Progress in cryptology - LATINCRYPT 2017-5th international conference on cryptology and information security in Latin America, Havana, Cuba, September 20-22, 2017, Revised selected papers. Lecture notes in computer science, vol. 11368, Springer; 2017, p. 352–77. http://dx.doi.org/10.1007/978-3-030-25283-0_19.