Identity-Based Threshold Group Signature Scheme of Blockchain Verification

Lipeng Wang[1,2(✉)], Mingsheng Hu[1], Zhijuan Jia[1], Yage Cheng[1], Junjun Fu[1], Yubo Wang[1], and Bei Gong[1]

[1] Zhengzhou Normal University, Zhengzhou 450044, China
wlpscu@126.com
[2] Peking University, Beijing 100871, China

Abstract. In the e-commerce scenario, the signature schemes generally have to meet four requirements: public verification, integrity, traceable and efficiency. To achieve the above goals, the paper proposes a identity-based threshold group signature scheme which can not only simplify the process of key management, but also allow to trace the user identities. To protect the user privacy, the scheme blinds the user identities and stores them on the blockchain to prevent the malicious members from tampering with the content. Security analysis shows that the proposed signature, whose difficulty is equivalent to solve the discrete logarithm problem, achieves a high level of anonymity and can resist impersonation attacks. Computational complexity analysis shows that the new method with low computation overhead and high communication efficiency can be effectively adapted to the electronic commerce scene.

Keywords: Blockchain · Confidential computation · Threshold group signature · Traceability · E-commerce

1 Introduction

Electronic commerce is a business activity, which focuses on commodity exchange technology. However the security situation of e-commerce has been getting worse. Billions of accounts have been stolen or controlled by the hackers, and millions of the user identities are leaked, even publicly traded. Therefore, it is necessary to study the signature schemes applicable to the e-commerce scenarios, which can protect the user privacy and information behaviors, prevent message forgery and repudiation, and guarantee integrity of the trading contents.

The identity-based signature scheme can verify the message content to ensure that the message is not tampered with during transmission. In the e-commerce scenario, the available signature schemes generally have to meet four requirements: public verification, integrity, traceable and efficiency. Previously, the user identities were the pseudonym information used by the user to participate in the signing process and the temporary identify labels were usually generated through the public key encryption algorithm with anonymity. For the identity-based signature scheme, the user can select his ID number, E-mail, mobile phone number as his identity for tracing after the event, which simplifies the key management process.

W. Han et al. (Eds.): CTCIS 2019, CCIS 1149, pp. 144–158, 2020.
https://doi.org/10.1007/978-981-15-3418-8_11

Blockchain is a distributed database technology which can record the transactions. The characteristics of blockchain are decentralization, anonymization and non-tampering. It handles distrust among nodes and has been widely applied in the fields of e-money, financial investment and IoT. As described above, the signatures and the blinded user identities can be stored in the blockchain, which guarantees the public verification of the signatures, and prevents the third party from maliciously tampering with the public information.

2 Related Work

In 1994, Marsh firstly proposed the concept of trusted computing and elaborated on the canonical representation of trust metrics. Based on the typical mathematical problems, existing threshold group signature schemes can be divided into three categories: 1. Based on prime factorization of large numbers. 2. Based on the discrete logarithm problem. 3. Based on the elliptic curve discrete logarithm problem. According to the way of key distribution, the threshold group signature schemes are mainly divided into two categories: 1. The ones with the trusted center. 2. The ones without the trusted center.

In 2004, Chen [1] proposed a signature scheme based on the elliptic curve encryption algorithm, and the length of the private key was short. However, it lacked the revoking operation and could not trace the user identities. In 2018, Wang [2] proposed a threshold group signature scheme which was introduced through collaboration between the blockchain and the trusted center. Dong [3] proposed an ECC threshold signature scheme with the elliptic curve discrete logarithm difficulty, which could trace the user identities with the help of the trusted center and effectively resist the malicious participants forging the final signature. The above threshold group signature schemes are based on the Shamir secret sharing method. Some other secret sharing methods will be discussed in the following part.

Yage [4] proposed a threshold signature scheme with strong forward security based on the Chinese Remainder theorem. The scheme updated the private key regularly to improve the security level and excluded the trusted center to increase the availability. Wang [5] proposed a scheme based on Asmuth-Bloom scheme applying on the blockchain voting scene. The method without the participation of the trust center was able to synthesize the final signature with the signature shares, during which the group private key was not exposed and the sensitive data would be validated to protect the user privacy during data transmitting in the blockchain unsafe transmission channel.

The paper [6] proposed an identity-based threshold group signature scheme based on the bilinear mapping. The master key of the scheme was distributed into the whole group and the user's signature key was stored in a threshold way. In [7], the threshold group signature scheme with the elliptic curve discrete logarithm difficulty was proposed. The method allowed signature verification and thresholds modification. The paper [8] proposed a threshold group signature scheme based on Elliptic Curve Cryptosystem with characteristics of easy verification, unforgeability and undeniability. The scheme had a shorter key length, lower computation and bandwidth requirement. [9] and [10] proposed two ECDSA threshold signature schemes. Both schemes allowed

participants to reconstruct keys. Gennaro et al. [11] proposed a threshold signature scheme which realized the multi-party control on the Bitcoin scene.

The concept of secret sharing was first proposed by Shamir [12] and Blackey [13]. The idea is to split the target secret S into N copies and each share will be sent to its corresponding participant. When need to reconstruct the secret, we should involve a certain number of participants, the number of whom should be greater than or equal to a specified threshold t. In the e-commerce scenario, the signature schemes generally have to meet four requirements: public verification, integrity, traceable and efficiency. To achieve the above goals, the paper proposes an identity-based threshold group signature scheme, which can not only simplify the process of key management, but also allow us to trace the user identities. To protect the user privacy, the scheme blinds the user identities and stores them on the blockchain to prevent the malicious members from tampering with the content.

3 Proposed Scheme

3.1 Architecture

The participants in the proposed scheme include user (U), trust center (TC), signature compositor (SC) and signature verifier (V). The proposed threshold group signature scheme includes seven parts: setup, registration, signature shares generation, combining signature shares, signature verification, signature opening and revocation.

The following section describes the details of the proposed threshold group signature scheme. For convenience, the symbols are defined as Table 1.

Table 1. Symbols defined for the proposed scheme

Symbol	Description	Symbol	Description
TC	Trusted center	ID_i	The identity of i th node
U_i	i th user	d_i	Private key for i th user
g_s	Private group key	D_i	Public key for i th user
g_p	Public group key	P	Set which comprises t members
T_s	Private key of trusted center	ID	Set which comprises t member identities
T_p	Public key of trusted center	ID_{i1}	Blinded member identity
SC	Signature compositor	ID_{i2}	Secondary blinded member identity
V	Signature verifier	UL	User information list

1. *System initialization*
   TC initializes the system parameters, and then mainly accomplishes two tasks: the first is to set the parameters for the proposed threshold group signature scheme and build the system template parameters; the second is to generate the key information and hash function for TC. The process is as follows:

1.1 Determine the participant number N and its corresponding threshold t for the proposed method, where $t \leq N$. F_p denotes the finite field, wherein p is a large prime number and the generator is g.

1.2 TC generates its private key and other related information depending on the template parameters generated in the previous step. First, TC selects its private key $T_s = s$, $s \in Z_p^*$, and its corresponding public key is $T_p = g^s \bmod p$. Then TC selects a $(t-1)$ degree polynomial $f(x) = \sum_{i=1}^{t-1} a_i x^i + a_0 \bmod p$, $a_j \in [1, p-1]$, $j = 0, 1, \ldots, t-1$, where $a_0 \in Z_p^*$ is the secret to be shared. The variable $a_0 = g_s = f(0)$ represents the group private key, and its group public key is $g_p = g^{g_s} \bmod p$.

1.3 Select a one-way hash function $h : \{0,1\}^* \rightarrow F_p$.

1.4 $(s, g_s, f(x))$ is the private information of TC, and (T_p, g_p, h, g, p) is the public information.

2. *User Registration*

When the user U_i wants to join the group, the registration process is performed. First, U_i sends its identity to TC, and TC will verify it. After the verification is passed, the identity will be blinded and sent to U_i. The blinded identity is verified and blinded for the second time. The secondary blinded identity along with the partial key which is generated by U_i is sent to TC. After TC receives the above information, it performs the verification operation, and stores the result on the blockchain. After that, TC generates another part of the private key for the user and sends it to U_i. After U_i verifies the information, the user synthesizes the key generated by himself and the partial key generated by TC to generate his own private key and the public key. The details are as follows:

2.1. The user U_i sends his identity ID_i to TC.

2.2. After TC receives ID_i, it checks whether the user has already registered or not. If the user does, TC rejects his request. Otherwise, TC generates $u \in Z_p^*$ randomly and calculates: $U = g^u \bmod p$, $ID_{i1} = s \times h(ID_i) + u$. After that, TC sends (U, ID_{i1}) to the user U_i.

2.3. After the user U_i receives (U, ID_{i1}), the data pair will be verified with $g^{ID_{i1}} \bmod p = \left(T_p^{h(ID_i)} \times U\right) \bmod p$. If the verification fails, it means that the data is tampered with during the transmission process, and the user U_i requests TC to resend the data. If the verification succeeds, the user U_i selects his partial private key $x_i \in Z_p^*$ and calculates $X_i = g^{x_i} \bmod p$. The user U_i needs to perform the secondary blindness operation on his identity to increase the security of the solution. First, the user U_i randomly selects $v_i \in Z_p^*$ and calculates $V_i = g^{v_i} \bmod p$. Then he performs the secondary blinding operation on his identity with $ID_{i2} = x_i \times h(ID_{i1}) + v_i$. After that, U_i sends $(X_i, V_i, ID_{i1}, ID_{i2})$ to TC.

2.4. After TC receives $(X_i, V_i, ID_{i1}, ID_{i2})$, it first checks the data pair with $g^{ID_{i2}} \bmod p = (X_i^{h(ID_{i1})} \times V_i) \bmod p$. If the verification succeeds, it denotes that the user has successfully generated his blinded identity, and TC will store

(X_i, ID_i, ID_{i2}) in the user information list UL, which will be used to trace the user identity afterwards. Then TC assigns another private key y_i to the user where $y_i = f(ID_{i2}) = \sum_{j=1}^{t-1} a_j (ID_{i2})^j + a_0 \bmod p$. In order to prevent the malicious third party from intercepting y_i, y_i needs to be encrypted to $E_{PK_{U_i}}(y_i)$ with the public key of U_i.

5. 2.5 After the user receives $E_{PK_{U_i}}(y_i)$, the user decrypts it with his private key to obtain the plaintext y_i. The user U_i can then generate his own private key with $d_i = x_i + y_i$. The corresponding public key for U_i is $D_i = g^{d_i} \bmod p$. At this point, the user registration process ends.

3. *Generating Signature Shares*
For the proposed threshold group signature scheme, the set of participants is denoted as $U' = \{U_1, U_2, \ldots, U_N\}$, and the final legal signature can be generated with at least t $(t \leq N)$ signature shares. For convenience, we will only consider t signature shares to combine the final signature, and the user set is denoted as $U = \{U_1, U_2, \ldots, U_t\}$. After registration, the user in the set firstly generates his signature share corresponding to the message, and then delivers it to SC for synthesizing. In the end, V checks the final signature and stores it in the blockchain. For the user U_i, when to generate his signature share, he first generates a random number $k_i \in Z_p^*$, and then obtains $r_i = g^{k_i} \bmod p$. We can obtain the corresponding hash value $z = h(m)$ from the message m, and then calculates the signature share with $s_i = k_i - z d_i I_i$, where $I_i = \prod_{j=1, i \neq j}^{t} \frac{ID_{j2}}{ID_{j2} - ID_{i2}} \bmod p$, $1 \leq i \leq t$. Since I_i is public, it can be pre-computed and published to reduce the computational complexity. Finally, the user U_i sends the signature share (r_i, s_i), the message m and its corresponding identity ID_{i2} to SC.

4. *Combine signature shares*
After SC receives greater than or equal to t signature shares, the final signature can be synthesized. Before that, we need to verify the received signature shares. In order to trace the user identify afterwards and prevent the third party from tampering with the content, the final signature needs to be stored into the blockchain. When SC verifies (r_i, s_i) and ID_{i2}, he first calculates the corresponding hash value $z = h(m)$ from the message m, and then performs the verification with $g^{s_i} D_i^{z I_i} = r_i \bmod p$. If the above equation holds, we can perform the subsequent signature synthesis operation, otherwise the signature compositor will reject the signature share. When performing signature synthesis, we first calculate $R = \prod_{i=1}^{t} r_i$, $S = \sum_{i=1}^{t} s_i$, and (R, S) is the final synthesized signature. In order to verify the synthesized signature, we should calculate $W = \prod_{i=1}^{t} X_i^{I_i} \bmod p$. SC sends (R, S, W) and the message m to V for verification.

5. *Signature verification*
   After V receives the above information, it needs to verify the signature. First, V performs a hash function $z = h(m)$ on the received message m, and then verifies the final signature with $R = g^S \times (g_p W)^z$. When the verification holds, V stores (ID_{i2}, r_i, s_i) and the synthesized signature (R, S, W) in the blockchain for traceability.

6. *Signature Opening*
   When dispute occurs, we need to access the blockchain and TC to trace the target identities from the given signature. During the process, the participation of SC and V is not required, and the feature of the blockchain guarantees the data against tampering, which not only increases the security level of the system, but also reduces the number of interactions to increase the efficiency. For the synthesized signature (R, S), we need to access the blockchain when to trace its corresponding identity. When accessing the blockchain, we need to search for (ID_{i2}, r_i, s_i) corresponding to (R, S). Among them, ID_{i2} is the secondary blinding identity, and the third party cannot infer the user identity. Then we need to access TC and obtain (X_i, ID_i, ID_{i2}) through ID_{i2}, among which ID_i is the user identity.

7. *Members Revoking*
   When a member withdraws from the group, only TC needs to perform most of the calculation tasks. The revocation message along with attached information needs to be broadcast to other users. Other users only need to perform a small number of operations to revoke the target member. First, TC needs to reselect a $t-1$ order polynomial, then recalculate the new partial key $y_i = f(ID_{i2})$ for each member ID_{i2}, and send the revocation message together with the encrypted message $PK_{U_i}(y_i)$ to other users. Other users only need to update their private keys with $d_i = x_i + y_i$, and their corresponding public keys with $D_i = g^{d_i} \bmod p$. After that, TC can delete the specified member ID_{i2} from the group.

4 Security Analysis

4.1 Correctness Analysis

Theorem 1. Two verification equations $g^{ID_{i1}} \bmod p = (T_p^{h(ID_i)} \times U) \bmod p$ and $g^{ID_{i2}} \bmod p = (X_i^{h(ID_{i1})} \times V_i) \bmod p$ are established when any user registers.

Proof: When the user U_i begins to register, the user U_i and TC are required to perform the two-way authentication. The corresponding verification formula is $g^{ID_{i1}} \bmod p = (T_p^{h(ID_i)} \times U) \bmod p$, $g^{ID_{i2}} \bmod p = (X_i^{h(ID_{i1})} \times V_i) \bmod p$. Because of $ID_{i1} = s \times h(ID_i) + u$, we can obtain $g^{ID_{i1}} \bmod p = g^{(s \times h(ID_i) + u)} \bmod p = (g^s)^{h(ID_i)} g^u \bmod p$. From

$U = g^u \bmod p$ and $T_p = g^s \bmod p$, we can get $g^{ID_{i1}} \bmod p = (g^s)^{h(ID_i)} g^u \bmod p = (T_p^{h(ID_i)} \times U) \bmod p$. Because of $ID_{i2} = x_i \times h(ID_{i1}) + v_i$, we can obtain $g^{ID_{i2}} \bmod p = g^{(x_i \times h(ID_{i1}) + v_i)} \bmod p = (g^{x_i})^{h(ID_{i1})} g^{v_i} \bmod p$. From $X_i = g^{x_i} \bmod p$ and $V_i = g^{v_i} \bmod p$, we can calculate $g^{ID_{i2}} \bmod p = (g^{x_i})^{h(ID_{i1})} g^{v_i} \bmod p = (X_i^{h(ID_{i1})} \times V_i) \bmod p$.

Theorem 2. When to generate the synthesized signature, the verification equation $r_i = g^{s_i} D_i^{zI_i} \bmod p$ of SC holds.

Proof: The user U_i begins to sign the message m and generate the signature share which will be sent to SC for signature synthesis. After receiving the information, SC should verify the message to ensure that the message is not maliciously tampered by a third party with $r_i = g^{s_i} D_i^{zI_i} \bmod p$. Because of $s_i = k_i - zd_iI_i$, we can obtain $g^{s_i} = g^{k_i - zd_iI_i} = g^{k_i}(g^{d_i})^{-zI_i} \bmod p$. From $r_i = g^{k_i} \bmod p$ and $D_i = g^{d_i} \bmod p$, we can get $g^{s_i} = r_i (D_i)^{-zI_i} \bmod p$, and then we can obtain $g^{s_i} D_i^{zI_i} = r_i \bmod p$.

Theorem 3. After generating the final signature, the verification equation $R = g^S \times (g_p W)^z$ of the signature verifier holds.

***Proof*:** After the signature compositor synthesizes the signature shares, the final signature is sent to the signature verifier for verification with $R = g^S \times (g_p W)^z$. Because of $s_i = k_i - zd_iI_i$, $S = \sum_{i=1}^{t} s_i$, we can obtain $g^S = g^{\sum_{i=1}^{t} s_i} = g^{\sum_{i=1}^{t} (k_i - zd_iI_i)} = g^{\sum_{i=1}^{t} k_i} g^{-\sum_{i=1}^{t} (zd_iI_i)} = \prod_{i=1}^{t} g^{k_i} \prod_{i=1}^{t} g^{-zd_iI_i}$. From $d_i = x_i + y_i$, $r_i = g^{k_i} \bmod p$ and $R = \prod_{i=1}^{t} r_i$, we can get $g^S = \prod_{i=1}^{t} g^{k_i} \prod_{i=1}^{t} g^{-zd_iI_i} = \prod_{i=1}^{t} r_i \prod_{i=1}^{t} g^{-z(x_i + y_i)I_i} = R \prod_{i=1}^{t} g^{-z(x_i + y_i)I_i}$. Then we can calculate $g^S \prod_{i=1}^{t} g^{z(x_i + y_i)I_i} = R = g^S \prod_{i=1}^{t} (g^{zx_iI_i} g^{zy_iI_i}) = g^S g^{z \sum_{i=1}^{t} f(ID_{i2})I_i} \prod_{i=1}^{t} g^{zx_iI_i}$. According to Lagrange interpolation formula, we can obtain $z \sum_{i=1}^{t} f(ID_{i2})I_i = za_0 = zg_s$. Then we can get $R = g^S g^{zg_s} \prod_{i=1}^{t} g^{zx_iI_i}$. Because of $g_p = g^{g_s} \bmod p$ and $X_i = g^{x_i} \bmod p$, we can obtain $R = g^S g^{zg_s} \prod_{i=1}^{t} g^{zx_iI_i} = g^S g_p^z \prod_{i=1}^{t} g^{zx_iI_i} = g^S (g_p \prod_{i=1}^{t} X_i^{I_i})^z$. Because of $W = \prod_{i=1}^{t} X_i^{I_i} \bmod p$, we can obtain $R = g^S \times (g_p W)^z$.

4.2 Threshold Safety Analysis

The threshold feature of the proposed scheme means that for a (t, n) threshold signature scheme, the secret will be dispersed into a group comprising of n members, and the subset of no less than t members can use their respective possessions to produce the final signature. On the other hand any subset of less than t members cannot recover the secret or obtain the correct result.

The process to obtain the threshold group signature includes generation of signature shares and synthesis of signature shares. For n participants, when to generate the signature shares, each user utilizes its own private key $k_i \in Z_p^*$ to sign the message and generates the signature share with the formula $s_i = k_i - zd_iI_i$. After the trusted center receives the signature shares $\{(r_i, s_i)|t \le i \le n\}$ from participants, the trusted centers will perform the synthesis operations with $R = \prod_{i=1}^{t} r_i$ and $S = \sum_{i=1}^{t} s_i$. (R, S) is the final synthesized signature which needs to be verified with the formula $R = g^S \times (g_pW)^z$.

At least t nodes are required to synthesize the signature shares. When to combine less than t signature shares, the synthesis will fail. When obtain the final signature, the content must be verified with $R = g^S \times (g_pW)^z$. According to Lagrange interpolation formula, we can obtain $z\sum_{i=1}^{t} f(ID_{i2})I_i = za_0 = zg_s$. If a malicious third party wants to obtain the private group key g_s, he needs at least t participants cooperating to succeed.

If the attacker has obtained the public key g_p and g, he wants to calculate the private group key g_s through $g_p = g^{g_s} \bmod p$, and the difficulty of the operation is reduced to the discrete logarithm problem which is computationally infeasible. In the end, the above analysis shows that the proposed scheme is safe.

4.3 Anonymity Analysis

The anonymity of the proposed threshold signature scheme means that for a given group signature, no one else can know the true identities of the participants except for TC or themselves.

When a user U_i wants to join the group, he first sends his identity ID_i to TC. TC will check its repeatability, and then blinds the user identity with $ID_{i1} = s \times h(ID_i) + u$, which takes $u \in Z_p^*$ and his private key s as input. ID_{i1} will be sent to the user who will check its integrity. After that, the user U_i will use his partial private key $x_i \in Z_p^*$ and a random value $v_i \in Z_p^*$ to perform the secondary blinding operation on his identity with $ID_{i2} = x_i \times h(ID_{i1}) + v_i$.

The two items ID_{i1} and ID_{i2} which are respectively generated by the user and TC are public, while the user id ID_i is secret. Other unauthorized nodes are not able to obtain the user true identity without the knowledge of the values of s, u, x_i and v_i. Even the attacker obtains those values, it is still difficult for him to retrieve the identities through the one-way hash function $h(x)$. The two items ID_{i1} and ID_{i2} are stored into the blockchain which cannot be tampered with. Based on the above analysis, the user identity can achieve a high level of anonymity which could prevent the malicious third party from tampering with the user sensitive information.

4.4 Unforgeability Analysis

Unforgeability means that any participant cannot impersonate another one to generate the legal signature. In real life, TC and the user may impersonate each other to sign the

given message with another identity. The attack scenes can be divided into two cases: Case 1: TC masquerades as member U_i, and signs the message m with the identity of U_i. Case 2: The user U_j masquerades as member U_i, and signs the message m with the identity of U_i.

For the first case, TC spoofs the identity of U_i. The blinded identity set $\{ID_{i1}, ID_{i2}, , ID_{it}\}$ is public, and TC can know the $(t-1)$ degree polynomial $f(x) = \sum_{i=1}^{t-1} a_i x^i + a_0 \bmod p$. The trusted center randomly selects $u_i' \in Z_p^*$ to generate $\left(U', ID_{i1}'\right)$ and selects $x_i^{'} \in \mathrm{Z}_p^*$ as the user U_i partial private key to calculate $X_i^{'} = g^{x_i^{'}} \bmod p$. TC performs secondary blindness on the identity of the user U_i by randomly selecting $v_i^{'} \in \mathrm{Z}_p^*$ to calculate $V_i^{'} = g^{v_i^{'}} \bmod p$. Then he performs the secondary blinding operation with $ID_{i2}^{'} = x_i^{'} \times h\left(ID_{i1}^{'}\right) + v_i^{'}$. After that, TC stores $\left(X_i^{'}, ID_i^{'}, ID_{i2}^{'}\right)$ in the information list, and then generates the final signature. In the end, TC stores $\left(ID_{i2}^{'}, r_i^{'}, s_i^{'}\right)$ and the synthesized signature (R', S') in the blockchain. However, the user U_i can obtain the data pair $\left(ID_{i2}^{'}, r_i^{'}, s_i^{'}, R', S'\right)$ from blockchain and $\left(X_i^{'}, ID_i^{'}, ID_{i2}^{'}\right)$ from TC with his own X_i. If TC wants to pass the verification, he must generate the value $X_i^{'}$ guaranteeing $X_i^{'} = X_i$, which means that TC has to obtain $x_i \in \mathrm{Z}_p^*$ from $X_i = X_i^{'} = g^{x_i} \bmod p$. The question is reduced to discrete logarithm problem, and the operation is computationally infeasible.

For the second scenario, the user U_j poses as the node U_i to sign the message m. At this time, the user U_j only knows ID_{i2} of the user U_i. The user U_j randomly selects $u_i' \in Z_p^*$, and selects $x_i^{'} \in \mathrm{Z}_p^*$ as the user U_i partial private key to calculate $X_i^{'} = g^{x_i^{'}} \bmod p$. Then the user U_j will perform the secondary identity blind operation to generate $\left(X_i^{'}, ID_i^{'}, ID_{i2}^{'}\right)$. In the end, TC will combine the signature shares to generate the final signature (R', S'), which will be sent to SC for verification with $R' = g^{S'} \times (g_p W)^z$. If the user U_j wants to pass the verification, he must guarantee the following equation holds $z\sum_{i=1}^{t} f(ID_{i2}) I_i = z a_0 = z g_s = z \sum_{i=1}^{t} f'(ID_{i2}) I_i$. I_i is public, and the user U_j can obtain $z = h(m)$ from the message m, so the user U_j should obtain the function $f'(ID_{i2})$ which meets $f(ID_{i2}) = f'(ID_{i2})$. Because of $f(x) = \sum_{i=1}^{t-1} a_i x^i + a_0 \bmod p$, the user U_j needs to guess the values of a_i, $0 \le i \le t-1$, which probability is $\mathrm{P}r = \frac{1}{(p-1)^t}$. Because p is a large prime number, the adversary can only succeed with a negligible probability.

5 Performance Analysis

5.1 Functionality Comparison

In this part, we give the functionality comparison of the proposed scheme with other related ones as shown in the following Table 2. We mainly focus on the five properties: public verifiability, sensitive information hiding, members revocable, traceability and anti-collusion.

Table 2. Functionality comparison

Schemes	Public verifiability	Sensitive information hiding	Members revocable	Traceability	Anti collusion
Dahshan et al. [14]	Yes	Yes	No	Yes	Yes
Yage et al. [4]	Yes	Yes	Yes	No	No
Lipeng et al. [15]	Yes	Yes	Yes	No	No
Shacham et al. [16]	Yes	No	Yes	No	No
Wang et al. [17]	Yes	No	Yes	No	No
Yannan et al. [18]	Yes	No	Yes	No	No
Wang et al. [19]	No	No	Yes	No	Yes
Shen et al. [20]	Yes	No	Yes	No	Yes
Shen et al. [21]	Yes	Yes	No	Yes	Yes
Ours	Yes	Yes	Yes	Yes	Yes

Public verifiability means that the final signature can be verified by the others apart from SV. Sensitive information hiding denotes that anyone else can not fetch the user identity from the public information advertised in the method. This property allows participants to opt out of the group, which is especially important in mobile Internet scenarios. Traceability represents that the supervisor can perform post hoc analysis to trace the identity information of the signer from the synthesized signature, mainly used in the auditing scenario. Anti-collusion means that any participant cannot impersonate another one to generate the legal signature. Note that the proposed scheme supports all the five properties compared with other related algorithms.

5.2 Performance Comparison

We define several notations to denote the operations in the proposed scheme as Table 3.

Table 3. Defined symbols for computational complexity

Symbols	Descriptions
T_{mul}	Modular multiplication operation
T_{exp}	Modular exponentiation operation
T_{inv}	Modular inverse operation
T_h	Hashing operation
T_{add}	Modular addition operation
T_{sub}	Modular subtraction operation

We mainly consider the three steps in the scheme, which are signature shares generation, signature shares combing and the final signature verification. We do not consider the step of user registration. This is because registration only needs to be performed only once when a user attempts to join the group. The time-consuming operations of the system are mainly signature synthesis and verification. The computation overhead comparison is described as the Table 4.

Table 4. Comparison of computation overhead

Schemes	Shares generation	Shares combining	Signature verification
Yage et al. [4]	$tT_{\exp} + (6t-1)T_{mul} + tT_{add} + tT_{inv}$	$(t-1)T_{add}$	$2T_{\exp} + T_{mul}$
Shen et al. [21]	$32tT_{mul} + tT_h + tT_{sub}$	$34tT_{mul} + (3t-2)T_{add}$	$30T_{mul} + T_h + 2T_{add}$
Lipeng et al. [15]	$(2t-1)T_{add} + (6t-1)T_{mul} + tT_{inv}$	$(t-1)T_{add}$	$2T_{\exp} + 2T_{mul}$
Ours	$tT_{\exp} + tT_h + tT_{sub} + 2tT_{mul}$	$tT_h + (4t-2)T_{mul} + 3tT_{\exp} + (t-1)T_{add}$	$2T_{\exp} + 2T_{mul} + T_h$

Generally speaking, $T_{\exp}$ is the most time consuming operation, followed by T_{inv}. The operation execution time rankings are usually as follows:

$$T_{\exp} > T_{inv} > T_{mul} > T_{add} > T_{sub}.$$

Although in the two stages of signature combining and signature verification, Yage et al. [4] takes less time. For the signature shares generation, the proposed scheme consumes less time than Yage et al. [4], which includes the inversion operation. Shen et al. [21] mainly contains the modular multiplication and modular addition operations without modular exponentiation operation, which seems more efficient than the proposed one. But during the process of user registration and the following steps, the method involves bilinear pairing operations which will take more time. That is because a bilinear pair operation is slightly equal to 50 exponentiation operations in general. The performance of the proposed scheme is inferior to Lipeng et al. [15] in the signature shares combining and the final signature verification, but the proposed one can

take the user identities as input during registration, which simplifies the key management process for users.

In real-world scenarios, the effectiveness of the proposed scheme depends on the expression execution time and the communication overhead. According to the description of the proposed algorithm, we can know that the communication overhead mainly comes from user registration, signature shares generating, signature shares combing and signature verification. For simplicity, the two steps of signature shares generating and signature shares combing are combined in one step which is denoted as signatures generating. Assuming $|\varsigma|$ is the size of an element in $\mathbf{Z}_p^*$, $|\Gamma|$ is the size of the message m, $|\eta|$ is the size of user identification and the length of $E_{PK_{U_i}}(y_i)$ is $|\varsigma|$ during the process of user registration, communication overhead of the proposed scheme is shown as Table 5.

Table 5. Communication overhead of the proposed scheme

Signature length	Registration	Signatures generating	Signature verification
$3\|\varsigma\|$	$\|\eta\|+7\|\varsigma\|$	$t(3\|\varsigma\|+\|\Gamma\|)$	$8\|\varsigma\|+\|\Gamma\|$

5.3 Experimental Results

In this subsection, we will evaluate the performance of the proposed scheme with several experiments. The details of the running environment are as following:

Hardware
CPU: Intel(R) Core(TM) i5-7500 CPU @ 3.40 GHz 3.40 GHz; *RAM: 8.00 GB.*

Software
OS: Windows 7 64; MyEclipse: 2015 CI; JAVA: 1.7.

Specifically, we will run the test 10 times and calculate the average value to eliminate errors. The running time of each step varies with N and t which will be recorded. The details are shown in Table 6, and the time measurement unit is milliseconds (ms).

Table 6. Execution time between different phases varying from N and t.

N	t	Setup	Register	Generate Shares	Combine Shares	Verify
5	2	1.84	17.96	0.78	2.38	0.8
10	5	0.77	31.21	1.87	5.66	0.76
15	7	0.78	46.02	2.61	7.76	0.75
20	10	0.74	61.43	3.67	11.1	0.76
25	12	0.73	75.09	4.39	13.16	0.85
30	15	0.74	90.4	5.56	16.42	0.74
35	17	0.74	105.4	6.33	18.64	0.74
40	20	0.73	121.8	7.34	21.71	0.73

(*continued*)

Table 6. (*continued*)

N	t	Setup	Register	Generate Shares	Combine Shares	Verify
45	22	0.75	137.61	8.14	24.22	0.75
50	25	0.74	154.19	9.26	27.47	0.71
55	27	0.73	170.23	9.82	29.43	0.74
60	30	0.74	188.98	11.1	33.25	0.74
65	32	0.77	207.47	11.78	35.37	0.75
70	35	0.74	220.4	12.87	38.37	0.73
75	37	0.75	238	13.5	40.52	0.74
80	40	0.74	255.53	14.57	43.6	0.74
85	42	0.77	273.31	15.41	45.65	0.74
90	45	0.74	290.62	16.46	49.19	0.73
95	47	0.75	308.36	17.36	51.44	0.73
100	50	0.75	328.68	18.3	54.79	0.79

From the data in the table, as N and t increase, the running time of the setup phase and the signature verification phase remains almost unchanged. The experimental results show that these two steps have no direct correlation with N and t, which is consistent with the implementation. We can also know the registration phase consumes most of the time, followed by the signature shares combing phase. That is because the two steps consist a lot of modular exponentiation operations and modular inverse operations which is demonstrated in Table 4.

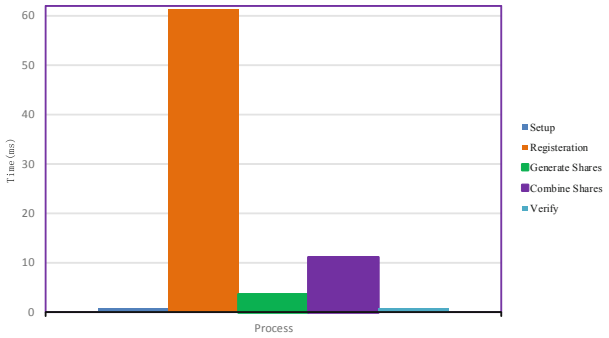

Fig. 1. Execution time of different steps

Figure 1 shows the execution time of the five steps when N is 20 and t is 10. We can know that the registration phase takes most of the overall running time of the system, almost 90%. It also shows that the step is the bottleneck of the overall performance, which denotes that we should focus on this step if we want to optimize the performance of the proposed algorithm.

6 Conclusion

The paper proposes an identity-based threshold group signature scheme, which can not only simplify the process of key management, but also allow tracing the user identities. To protect the user privacy, the scheme blinds the user identities and stores them on the blockchain to prevent the malicious members from tampering with the content. Security analysis shows that the proposed signature, whose difficulty is equivalent to solve the discrete logarithm problem, achieves a high level of anonymity and can resist impersonation attacks. Computational complexity analysis shows that the new method with low computation overhead and high communication efficiency can be effectively adapted to the e-commerce scenario.

Acknowledgments. This work was supported by Henan Province Higher Education Key Research Project (20B520040).

References

1. Chen, T.S., Hsiao, T.C., Chen, T.L.: An efficient threshold group signature scheme. In: TENCON 2004, pp. 13–16 (2004)
2. Wang, L., et al.: A voting scheme in blockchain based on threshold group signature. In: Zhang, H., Zhao, B., Yan, F. (eds.) CTCIS 2018. CCIS, vol. 960, pp. 184–202. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-5913-2_12
3. Dong, X., Jiajia, L., Zhonghua, S.: A new threshold signature scheme based on elliptic curve cryptosystem. J. Hangzhou Normal Univ. (Nat. Sci. Ed.) **12**(1), 57–60 (2013)
4. Yage, C., Mingsheng, H., Bei, G., Lipeng, W., Erfeng, X.: Dynamic threshold signature scheme with strong forward security. Comput. Eng. Appl. **1**(23), 1–12 (2019)
5. Wang, L., Hu, M., Jia, Z., Gong, B., Lei, Y.: A signature scheme applying on blockchain voting scene based on asmuth-bloom algorithm. In: IEEE 4th International Conference on Computer and Communications (2018)
6. Hongwei, L., Weixin, X., Jianping, Y., Peng, Z.: Efficiency identity-based threshold group signature scheme. J. Commun. **30**(5), 122–127 (2009)
7. Jie, Y., Xuri, Y., Wujun, Z.: Research on group signature with threshold value based on elliptic curve. J. Southeast Univ. (Nat. Sci. Ed.) **38**(1), 43–46 (2008)
8. Yufang, C., Tzerlong, C., Tzer-Shyong, C., Chihsheng, C.: A study on efficient group-oriented signature schemes for realistic application environment. Int. J. Innov. Comput. Inf. Control **8**(4), 2713–2727 (2012)
9. 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, pp. 1179–1194 (2018)
10. Yehuda, L., 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, pp. 1837–1854 (2018)
11. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 156–174. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_9

12. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
13. Blakley, G.R.: Safeguarding cryptographic keys. In: AFIPS Conference Proceedings, pp. 313–317 (1979)
14. Dahshan, H., Kamal, A., Rohiem, A.: A threshold blind digital signature scheme using elliptic curve dlog-based cryptosystem. In: IEEE Vehicular Technology Conference, pp. 1–5 (2015)
15. Lipeng, W., Mingsheng, H., Zhijuan, J., Bei, G., Jialei, Z.: A signature scheme applying on blockchain voting scene based on chinese remainder theorem. Appl. Res. Comput. **29**(1), 1–8 (2018)
16. Shacham, H., Waters, B.: Compact proofs of retrievability. J. Cryptol. **26**(3), 442–483 (2013)
17. Wang, B., Li, B., Li, H.: Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans. Serv. Comput. **8**(1), 92–106 (2015)
18. Yannan, L., Yong, Y., Geyong, M., et al.: Fuzzy identity-based data integrity auditing for reliable cloud storage systems. IEEE Trans. Dependable Secure Comput. **14**(8), 72–83 (2017)
19. Wang, H.: Proxy provable data possession in public clouds. IEEE Trans. Serv. Comput. **6**(4), 551–559 (2013)
20. Shen, J., Shen, J., Chen, X., Huang, X., Susilo, W.: An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans. Inf. Forensics Secur. **12**(10), 2402–2415 (2017)
21. Wenting, S., Jing, Q., Jia, Y., et al.: Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **14**(2), 331–346 (2019)