A New Multi-Party Concurrent Signature Based on Discrete Logarithm

Wenwen Feng Xiangqian Liang† Hao Jiang Yanhong Li

Shandong University of Science and Technology, China

Abstract

The notion of concurrent signatures was introduced by Chen et al. to provide an efficient way to exchange fairly the digital signatures among parties. In the traditional initial-matcher scheme, the generation and publication of keystone is done by the initial user of session, while the matchers have no power to control the keystone and thus makes the signature not fair. In this paper, we present a scheme that makes sure that keystones are generated by the cooperation of all users, therefore, fairness will be guaranteed since users have the equal powers to control keystones. And, each user can independently run the verify algorithm without any information of keystones which provides a better privacy to the signers. Moreover, our scheme is practical and efficient according to computational costs, signature size and security assumptions.

Keywords:

Discrete logarithm; multi-party concurrent signature; fair exchange protocol; fairness; keystone encryption

1. Introduction

With the development of internet, more and more new application patterns of business conducted via the internet are emerged. For instance, fair exchange (sign the contract, e-mail authentication and electronic payment) among more than two users has become a practical problem recently. Many signatures were produced by scholars to handle this situation. In [1], a multi-signature scheme was proposed with proven security in the dedicated key generation model. Multi-signature schemes in [2] and [3] are proven secure in the knowledge of secret key model. And in [4], the multi-signature schemes of [2] and [3] have been showed to be secure in the proof of possession model. Although these schemes are proven secure, they are less practical for applications for their signature are proposed under the pairing-based assumptions.

After Chen, Kudla and Paterson [5] firstly introduced the concept of concurrent signature (CS) in 2004, people paid more attention to extend a two party concurrent signature scheme to the multi-party case. In a CS scheme, two signers use a piece of information called keystone to exchange their signatures in a fair way with no trust third party, and no third party can identify who signed which signature before keystone is released. If the keystone is released by one of the signers, both signatures are bound to their true signers concurrently. The research of multi-party fair exchange protocol in the domestic and abroad is few, because the protocol involves many collaborators with exchanging a lot of informations which makes a huge calculated quantity and a complicated protocol [6] [7]. In 2005, Nguyen [8] achieved the anonymity and the articulation of concurrent signature via using Schnorr signature scheme and Schnorr-like signature scheme. People call his scheme the dissymmetric signature system, for the initiator and the matcher in his protocol use different signature algorithms. In 2006, Susilo and Mu [9] proposed a three-party concurrent signature scheme from bilinear. This is the first signature that really pushes through the limitation of two party concurrent signatures. They laid a foundation for the developing of multi-party concurrent signature even their scheme seems difficult to be extended to a multi-party condition. In the same year, Tonien et al [10] proposed a multi-party concurrent signature scheme using techniques of ring signature and bilinear pairings, which is the first multi-party concurrent signature scheme. In 2008, Chow and Susilo [11] pointed out that Tonien et al's signature do not meet the concept of concurrent signature and proposed a new multi-party concurrent signature scheme based on two major paradigms of ID-based ring signature scheme. However, this scheme is also not good enough. Because before the keystone is released, the signature is both ambiguous to the inside signers and outside people, which makes the inside signer not know who signed the contract.

In our scheme, we present a new multi-party concurrent signature based on discrete logarithm. There are n users and we call these n users 'insiders' to distinguish them from any other parties as the 'outsiders'. A signer signs the message m , has her own public key and private key, and generates a public parameter K by running Algorithm Keystone Encryption with cooperation of all the other users, which makes the signature fairer than traditional signatures. After all the users release their keystones, any one no matter insider or outsider can run

Manuscript received September 5, 2026
Manuscript revised September 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.9.14

the binding verify algorithm. Otherwise, only the inside users can know who signed the signature, the outside verifier can only be convinced that the message was signed by one of the members in this group.

The rest of this paper is organized as follows. In the next section, we give some useful notions and preliminary knowledge that will be used in this paper. In section 3, we present our multi-party concurrent signature scheme based on discrete logarithm. Section 4 provides its security proof together with efficiency analysis. Section 5 concludes the paper.

2. Preliminaries

2.1 Cryptography Hypothesis

The concurrent signature system realized in this paper is based on difficult discrete logarithm assumption and computational Diffie-Hellman assumption. Since the former cryptography assumption is weaker than the latter, if the computational Diffie-Hellman hypothesis holds, the difficult discrete logarithm assumption is satisfied too. These two assumptions can be described as follows.

Difficult Discrete Logarithm Assumption

Given two big primes p and q whose length is L (L is the security parameter and $q|(p-1)$), the generator $g \in Z_p^*$ with an order of q, $g^x \in Z_p^*$, and x is unknown. The Difficult Discrete Logarithm Assumption is that for any polynomial computing ability attacker A, the following probability is negligible

$$\Pr\left[A(\mathrm{p},\mathrm{q},\mathrm{g},\mathrm{g}^x)=\mathrm{x}\right] \tag{1}$$

Computational Diffie-Hellman Assumption

Given two big primes p and q whose length is L (L is the security parameter and $q|(p-1)$), the generator $g \in Z_p^*$ with an order of q, $g^x \in Z_p^*$, and x is unknown. The Difficult Discrete Logarithm Assumption is that for any polynomial computing ability attacker A, the following probability is negligible

$$\Pr\left[A(\mathrm{p},\mathrm{q},\mathrm{g},\mathrm{g}^x,\mathrm{g}^y)=g^{xy}\right] \tag{2}$$

2.2 Security Requirement

W. Susilo [9] said a secure concurrent signature should satisfy four properties: correctness, ambiguity, fairness, and unforgeablility, so as the multi-party concurrent signature. Simply, the notions of these properties in our signature can be described as follows:

Correctness

A signature will output accept if the signature has been generated correctly by the signing algorithm when checking the two algorithms InsiderVerify and OutsiderVerify. Additionally, after all the keystones are released, the algorithm BindingVerify will also output accept.

Ambiguity

It prevents the public from distinguishing whether an ambiguous signature $\sigma_i(\sigma_j)$ was generated by $U_i(U_j)$ before the keystone is released. In addition, no one including $U_i(U_j)$ should be able to convince anyone else that $\sigma_i(\sigma_j)$ was generated by $U_i(U_j)$.

Fairness

Any signature that is generated with the same keystone will be bound concurrently when the keystone is released.

Unforgeability

It requires that when a party having received an ambiguous signature, can be sure that the signature is generated by its true signer.

3. Multi-party Concurrent Signature Based on Discrete Logarithm

3.1 Algorithm

1. Algorithm Setup

Let p and q be two large primes whose length is L (L is the security parameter) such that $q|(p-1)$. Let g be a generator with order q in Z_p^*. Every signer chooses a secret key $x_i \in Z_q (1 \le i \le n)$, and computes the public key $y_i = g^{x_i} \bmod p (1 \le i \le n)$, the message space $M = \{0,1\}^*$, keystone space $K \in Z_q$, all users' public keys tuple $Y = \{y_1,...,y_n\}$, three strong collision-resistant hash functions:

$$H_1 : \{0,1\}^* \times Z_q \to Z_q \tag{3}$$

$$H_2 : \{0,1\}^* \times \left(Z_p^*\right)^n \times Z_q^* \times Z_p^* \to Z_q \tag{4}$$

$$H_3 : \{0,1\}^* \times \left(Z_p^*\right)^n \times Z_q^* \times Z_q \to Z_q \tag{5}$$

2. Algorithm KeyGen

Every signer i chooses a random number $x_i \in Z_q$ as her private key, then computes $y_i = g^{x_i} \bmod p$, and publishes y_i to others.

3. Algorithm Keystone Encryption

Every signer i chooses $k_i \in K$ randomly as her own keystone, then she computes $k_i' = H_1(m_i \| k_i)$, $K_i = g^{k_i'} \bmod p$, and sends K_i to others. After all of them get the others' K_i, they compute

$$K = \sum_{j=1}^{n} K_j \bmod p \tag{6}$$

4. Algorithm Sign (m_i, y, K, y_i)

Taking as input of the common parameter $(p,q,g,\mathrm{L,y,M})$ and a message m_i, the signer i uses her public key and private key to participate in the signing protocol as follows.

For each $i = 1,2,...,\mathrm{n}$, $j \neq i$, compute

$$U_{ij} = H_2(m_i \| Y \| K \| y_j^{x_i} \bmod p) \tag{7}$$

$$r_i = K^{x_i} \bmod p \tag{8}$$

$$R_i = H_2(m_i \| Y \| K \| r_i) \tag{9}$$

$$U_{ii} = y_i^{-1} r \Big/ \sum_{j \neq i} U_{ij} y_j \tag{10}$$

$$V_i = H_3(m_i \| Y \| K \| R_i) \tag{11}$$

Then the user's signature is $\sigma_i = (V_i, U_{i1}, U_{i2}, ..., U_{in})$.

5. Algorithm OutsiderVerify (m_i, σ_i, Y, K)

For $j = 1,...,n$ compute $r_i = \sum_{i=1}^{n} U_{ij} y_j$ and $R_i = H_2(m_i \| Y \| K \| r_i)$. Then verify $V_i = H_3(m_i \| Y \| K \| R_i)$. If it is true, accept the signature.

6. Algorithm InsiderVerify $(m_i, \sigma_i, Y, K, y_j)$

After input the message m_i, a signature σ_i and the public key tuple $Y = \{y_1, ..., y_n\}$, the user j do as follows.

Firstly, verify whether U_{ij} equal to $H_2(m_i \| Y \| K \| y_j^{x_i} \bmod p)$, if not output reject, otherwise continue.

Secondly, for each $j = 1,...,n$, compute $r_i = \sum_{j=1}^{n} U_{ij} y_j$ and $R_i = H_2(m_i \| Y \| K \| r_i)$, output accept if $V_i = H_3(m_i \| Y \| K \| R_i)$.

Since this algorithm needs the user j 's secret key x_j, it can only be executed by user j, who has the secret key x_j.

7. Algorithm BindingVerify $(k_1, k_2, ..., k_n, Y, K, m_i, \sigma_i)$

Consequently, check the following conditions, outputs reject if any condition fails.

Firstly, check whether $K = g^{\sum_{j=1}^{n} k_j'} \bmod p$ holds.

Secondly, for each $j = 1,...,n$ in the signature, verify $U_{ij} = H_2(m_i \| Y \| K \| y_j^{x_i} \bmod \mathrm{p})$ and $V_i = H_3(m_i \| Y \| K \| R_i)$.

Actually, it can only be executed after all the users' keystones are released, since this algorithm needs to compute $g^{\sum_{j=1}^{n} k_j'} \bmod p$.

3.2 Scheme

Based on the concurrent signature above, we propose a scheme for multi-users.

a. All users run Setup and Keystone algorithm to obtain system public parameters and their own secret keys.

b. Each user chooses a $k_i \in K$ randomly as her own keystone and computes $K_i = g^{k_i'} \bmod p$, then send K_i to the others. After all of them got all the K_i from the others, they compute K by equation (6).

c. Each user computes her signature $\sigma_i = (V_i, U_{i1}, U_{i2}, ..., U_{in})$ and sends σ_i to others.

d. After one user receives all the signatures from others, she runs the insider verify algorithm to judge whether these signatures are valid or not. If they are all valid, then release her keystone. It is crucial to note that all the outsiders can only run the OutsideVerify algorithm. Therefore, the outsiders only have the permission to distinguish that the signature is signed by some one of the insiders.

e. After all the users release their keystones, any one no matter insiders or outsiders can run the Binding Verify algorithm. That is to say, the signature is bound to its true signer eventually after all the insiders release their keystones.

4. Security and Efficiency Analysis

4.1 Correctness

If the signature has been generated correctly by invoking Sign algorithm, then it should pass the OutsiderVerify, InsiderVerify and BindingVerify.

OutsiderVerify

Since (10) we have

$$\sum_{j=1}^{n} U_{ij} y_j = U_{ii} y_i \cdot \sum_{j \neq i} U_{ij} y_j = r_i \quad (12)$$

thus

$$H_2(m_i \| Y \| K \| \sum_{j=1}^{n} U_{ij} y_{ji}) = H_2(m_i \| Y \| K \| r_i) \quad (13)$$

The equation above shows that as long as the algorithm of Sign is run correctly, one can pass the OutsideVerify obviously.

InsiderVerify

Since

$$\begin{aligned} y_j^{x_i} \bmod p &= (g^{x_j})^{x_i} \bmod p = (g^{x_i})^{x_j} \bmod p \\ &= y_i^{x_j} \bmod p \end{aligned} \quad (14)$$

in a signature $\sigma_i = (r_i, U_{i1}, U_{i2}, ..., U_{in})$, for $j = 1, ..., n$, $j \neq i$, we have

$$\begin{aligned} U_{ij} &= H_2(m_i \| Y \| K \| y_j^{x_i} \bmod p) \\ &= H_2(m_i \| Y \| K \| y_i^{x_j} \bmod p) \end{aligned} \quad (15)$$

So if the signature is valid, one will also satisfy the OutsiderVerify as long as she can pass the InsiderVerify.

BindingVerify

Since

$$r_i = K^{x_i} \bmod p = g^{x_i \sum_{j=1}^{n} k'_j} \bmod p = y_i^{\sum_{j=1}^{n} k'_j} \bmod p \quad (16)$$

we obtain that

$$R_i = H_2(m_i \| Y \| K \| r_i) = H_2(m_i \| Y \| K \| y_i^{\sum_{j=1}^{n} k'_j} \bmod p) \quad (17)$$

So if the signature is valid, once a user can pass the BindingVerify, he will also satisfy the InsiderVerify and OutsideVerify algorithms.

4.2 Ambiguity

Since the outside users can not know the private keys of the inside users, it is difficult for them to compute $U_{ij} = H_2(m_i \| Y \| K \| y_j^{x_i} \bmod p)$ and $r_i = K^{x_i} \bmod p = y_i^{\sum_{j=1}^{n} k'_j} \bmod p$. An adversary cannot prove that user i is the actual signer of a signature before all the keystones k_i are released. So according to the ambiguity of concurrent signature, this signature has the quality of ambiguity for the outsiders. In this paper, the ambiguity dose not suit the insiders. A insider can judge who sign the signature before the keystone is released by computing whether $U_{ij} = H_2(m_i \| Y \| K \| y_j^{x_i} \bmod p)$ is valid or not. Because if the equation is true then $y_j^{x_i} \bmod p$ must equal to $y_i^{x_j} \bmod p$, the one who can compute $y_i^{x_j} \bmod p$ must know x_j and y_i.

4.3 Fairness

In our paper, if a multi-party concurrent signature scheme is fair, it should satisfy the following properties. Firstly, the processes of generating and releasing keystone for all users are same. Secondly, If there exists a user i has not released the keystone k_i, any attacker cannot produce the n random figures $(k_1, ..., k_n)$ and make them pass the BindingVerify algorithm. Finally, once all users publish their keystones and run the BindingVerify, the authorship of the signatures should be bounded to their signer's concurrently.

In Tonien's [10] signature, the binding of the signature only need the keystone of one who signs the signature which can makes someone not release his keystone on purpose or publish a wrong keystone to make the signature not bind to himself. It makes the unfair of the signature.

It is obvious that in this signature, the processes of generating and releasing keystone for all users are same, since we present a signature makes sure that the keystones are generated by the cooperation of all the users, and the binding of the signature realizes only after all the signers release their keystones. As long as there is someone not releases his keystone, no outsider can certify the source of a signature.

As illustrated in the proof of unforgeability, the scenarios that an attacker cannot pass the BindingVerify by producing the n random figures $(k_1, ..., k_n)$ and once all users public their keystones and run the BindingVerify, the authorship of the signatures should be bounded to their signer's concurrently due to the ability of the signer to execute the BindingVerify algorithm correctly.

4.4 Unforgeability

In document [6], the author proposed that there are two different cases of unforgeability in the concurrent signature need to be considered.

Outside Attacker: when an adversary doesn't have all users' secret keys, no valid signature will satisfy the verify algorithms.

Inside Attacker: when an adversary has all insider users' secret key x_k, but not has the user i's secret key x_i and the

user j 's secret key x_j , no valid signature will make the verify algorithms output accept.

It is obvious that an outside attacker has weaker authority than the inside adversary. Thus, if this signature can resist the latter, it can also counteract other adversary certainly. Moreover, from the notion of ambiguity we know that the unforgeability doesn't suit the outsiders, since an outsider cannot know the author of the signature until all the keystones are released. Therefore, the adversary cannot make an outsider believe that the signature is signed by whom concretely before keystones are released. That is to say, the adversary cannot forge an outsider and make him believe the signature is signed by its true signer before all keystones are released.

Theory 4.1 Based on the hardness of the Computational Diffie-Hellman Assumption, the proposed scheme is existential unforgeable against adaptive chosen message attack under the random oracle model.

Proof. Consider that an adversary A is playing forgery game as follows.

At first, he chooses two insiders u and v , letting $X_{uv} = \{1,2,...,\mathrm{n}\}/\{u,v\}$. Suppose that A has obtained the following three resources.

Public key and secret key: The adversary has got the public key tuple Y and $n-2$ secret keys $\{x_i, i \in X_{uv}\}$ as an initial input.

Hash queries: The adversary can require a hash value for any input at any time. What's more, the value of each hash function can be recorded in a hash list by the attacker.

Sign query: Having got the secret key of every user except u and v , the adversary can sign any message on behalf of user $i \in X_{uv}$. Additionally, a valid signature will be given to the adversary so that he can pass the InsiderVerify algorithm, which is executed against any secret key x_j of user $j \in X_{uv}$.

The adversary wants to forge a signature that could pass the InsiderVerify algorithm to makes user u believe that it is generated by user v , but the forgery signature is not equal to any signature that has been given to the adversary from the sign query.

Suppose that there exists an adversary who wins the attack game with non-negligible possibility. We are going to show that the Computational Diffie-Hellman Assumption can be solved by using such adversary.

We play as follows.

Step1 Each user $i \in X_{uv}$ chooses a random $x_i \in Z_q$, and sets $y_i = g^{x_i} \bmod p$. Randomly choose $s_u, \mathrm{s}_v \in Z_q$, then set $y_u = g^{s_u x} \bmod p$, and $y_v = g^{s_v y} \bmod p$ corresponding to $x_u = s_u x$ and $x_v = s_v y$. Note that we don't know the values of x_u and x_v , but we can calculate y_u , y_v . The public key tuple and $n-2$ secret keys $\{x_i, i \in X_{uv}\}$ are available to the adversary.

Step2 For each new hash query to H_1 , randomly choose a number t and output its corresponding hash value $g^t \bmod p$. If an old value is typed again by the adversary, then output the value on the hash list. Store the t values to a T-list.

Step3 For every query on the private key x_u , sign it by using our algorithm. According to our algorithm, the private key x_u is only used in $H_2(m_u \| Y \| K \| y_v^{x_u} \bmod p)$ and $r_u = K^{x_u} \bmod p$. But we need to calculate these value without x_u , since we don't know it. On one hand, for each $i \neq u, v$, $y_i^{x_u} \bmod p$ can be calculated by using y_u and x_i through $y_i^{x_u} \bmod p = y_u^{x_i} \bmod p$. On the other hand, after randomly choosing a number in Z_q to represent $y_v^{x_u} \bmod p$, we can pick a number t from the T-list such that $K = g^t \bmod p$,then the user v can obtain $U_{uv} = H_2(m_u \| Y \| K \| y_v^{x_u} \bmod p)$. Therefore, the signature can pass the algorithm InsiderVerify against the sender u for any secret key $\{x_i, i \in X_{uv}\}$. So we have proved that the sign requests on secret key x_i can be satisfied.

Suppose that the adversary can produce a forgery signature $(V_u, U_{u1}, U_{u2}, ..., U_{un})$. To convince user v that this signature is signed by user u , it must ensure that $H_2(m_u \| Y \| K \| y_u^{x_v} \bmod \mathrm{p}) = H_2(m_u \| Y \| K \| y_v^{x_u} \bmod \mathrm{p})$. Since the output of H_2 is totally random, the adversary must have queried H_2 with the exact input $m_u \| Y \| K \| y_v^{x_u} \bmod p$. We can look up in the hash list for this input, thus we can obtain $y_v^{x_u}$. However, $y_u^{x_v} \bmod p = y_v^{x_u} \bmod p = g^{x_u x_v} \bmod p$. Therefore, we can calculate the value $g^{x_u x_v} \bmod p = g^{s_u s_v xy} \bmod p$. This completes the proof. □

The same to the proof of inside unforgeability, if there exists a user i wants to release his real keystone k_i or an attacker wishes to produce a figure k_i'' to satisfy $K_i'' = K_i$

or $g^{k_i'} \bmod p = g^{k_i'} \bmod p$, they are also equal to solve the Difficult Discrete Logarithm Assumption.

4.5 Efficiency

Finally, we analyze the communication rounds, signature size and computation cost of our signature as the following table.

Table 1: Comparison between ours' and Tonien's

	Ours'	Tonien's
Communication Rounds	n^2+2n	n^2+n
Computation Cost	$\left(n^2+n+2\right)E$	$2\left(n^2+n\right)E$
Signature Size	n^2	n^2

Remark Here, E stands for an exponentiation in the group Z_p^*. We use E as the unit to measure the computational cost of signing and verification in order to make the comparisons clear. Since the compute of exponentiation in the group Z_p^* cost greater than operate multiplication in Z_p^*, addition operation and inverse operation in Z_p, we only consider the index operation cost in this paper.

By analysis we know that our signature not only fairer but also higher efficiency than the tradition's.

5. Conclusion

Multi-signature is a special kind of digital signature where a group of parties can sign on a common message to obtain a compact signature with fast verification. In this paper, we propose an efficient multi-party concurrent signature scheme based on discrete logarithm. The equal control power of keystone to each user is proposed in this paper by introducing a keystone encryption algorithm, together with a concrete scheme which can be used in more than three users under the new model. There are two interesting open problems along this research:

Give our signature a concrete certificate for its ambiguity.

Realize our signature with other cryptographic assumption.

Reference

[1] S. Micali, K. Ohta, L. Reyzin, Accountable-subgroup multisignatures[C], The 8th ACM Conference on Com-puter and Communications Security, 2001.

[2] A. Boldyreva, Efficient threshold signature, multi-signature and blind signature schemes based on the gap-Difiie-Hellman-group signature schemes[C], Public Key Cryptography, Lecture Notes in Computer Science, 2003: 1567.

[3] U. Feige, A. Shamir, Witness indistinguishable and witness hiding protocols[C], The 22nd Annual ACM Symposium on Theory of Computing, 1990.

[4] T. Ristenpart, S.Yilek, The power of proofs of possession, Securing multiparty signatures against rogue key attacks[C], Advances in Cryptology-EUROCRYPT, Lecture Notes in Computer Science, 2007: 4515 .

[5] L. Chen, C. Kudla, K. G. Paterson, Concurrent Signa-ture[C], Advances in Cryptology-EUROCRYPT 2004, Springer Berlin Heidelberg, 2004: 287-305.

[6] C. Shieh, Fair Multi-Party Concurrent Signatures [D]. Taiwan, National Central University, 2008.

[7] X. Tan, Q. Huang, S.Wong, Concurrent Signature without Random Oracles [J], IACR Cryptology ePrint Archive, 2012: 576.

[8] K. Nguyen, Asymmetric concurrent signatures[M], Information and Communica- tions Security2005, Springer Berlin Heide4lberg, 2005: 131-145.

[9] W. Susilo, Y. Mu, Tripartite Concurrent Signatures [M], IFIP/SEC 2005. LNCS, vol. 181, Springer, 2005: 425-441.

[10] D. Tonien, W. Susilo, R. Safavi-Naini, Multi-party concurrent signatures[C], Information Security. Springer Berlin Heidelberg, 2006: 131-145.

[11] S. Chow, S. Yiu, L. Hui, Efficient identity based ring signature[C], Applied Cryptography and Network Security, Springer Berlin Heidelberg, 2005: 499-512.