A Trustless GQ Multi-Signature Scheme with Identifiable Abort

Handong Cui, Tsz Hon Yuen

Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong SAR
{hdcui,thyuen}@cs.hku.hk

Abstract. Guillou-Quisquater (GQ) signature is an efficient RSA-based digital signature scheme amongst the most famous Fiat-Shamir follow-ons owing to its good simplicity. However, there exist two bottlenecks for GQ hindering its application in industry or academia: the RSA trapdoor $n = pq$ in the key generation phase and its high bandwidth caused by the storage-consuming representation of RSA group elements (3072 bits per one element in 128-bit security).
In this paper, we first formalize the definition and security proof of class group based GQ signature (CL-GQ), which eliminates the trapdoor in key generation phase and improves the bandwidth efficiency from the RSA-based GQ signature. Then, we construct a trustless GQ multi-signature scheme by applying non-malleable equivocable commitments and our well-designed compact non-interactive zero-knowledge proofs (NIZK). Our scheme has a well-rounded performance compared to existing multiparty GQ, Schnorr and ECDSA schemes, in the aspects of bandwidth (no range proof or multiplication-to-addition protocol required), rather few interactions (only 4 rounds in signing), provable security in *dishonest majority model* and identifiable abort property. Another interesting finding is that, our NIZK is highly efficient (only one round required) by using the Bezout formula, and this trick can also optimize the ZK proof of Paillier ciphertext which greatly improves the speed of Yi's Blind ECDSA (AsiaCCS 2019).

Keywords: Guillou-Quisquater signature, multi-signature, zero-knowledge proof, remove trusted setup

1 Introduction

Guillou-Quisquater signature, also called GQ signature, was proposed by Guillou and Quisquater in 1988 [23]. Together with Schnorr signature [38], GQ signature scheme is amongst the most efficient and famous Fiat-Shamir [17] follow-ons. GQ has some applications in cryptographic protocols such as forward-secure signature [29], identity-based signature with bounded life-span [14], distributed certificate status protocol [44], distributed authentication algorithm for mobile ad-hoc

network [41], GQ1 (identity-based) and GQ2 schemes in ISO/IEC 14888-2 standard [27] and etc. GQ has already been used to construct distributed signing protocols, including multi-signature schemes [2,3,13,39] and threshold signature schemes [11,32,40]. Nevertheless, GQ's application scenarios and research discussions are still rather limited when compared with Schnorr and ECDSA which are the most widely used two digital signature schemes by virtue of Schnorr's great simplicity and ECDSA's application in blockchains like Bitcoin and Ethereum.

Drawbacks of RSA-based GQ. One obvious flaw of all the aforementioned GQ applications is that all these applications require a trusted setup to generate the public/private key pair through generating two large primes p and q secretly and setting $n = pq$ publicly as the group order. This is prohibitive for practical adoption of GQ in a trustless environment, like a public blockchain or a digital wallet where no trusted third party (TTP) is involved. In 2000, Hamdy and Möller [24] informally pointed out that class groups of imaginary of quadratic fields (IQC) proposed by Buchmann and Williams [7] can be applied in GQ signature, thus shedding light on how to remove the RSA trapdoor in GQ signature scheme, i.e., replacing the RSA group in GQ signature with a class group. Yet, such a class group based GQ signature lacks a formal definition and a rigorous security proof for EUF-CMA (Existential Unforgeability under Chosen Message Attack) along with a suitable hardness assumption. Another shortcoming for GQ protocols is that, since all the elements in RSA group of order n have to be represented by a 3072-bit string for 128-bit security, it is not bandwidth efficient, especially in a multi-user setting. On the class group side, to achieve 128-bit security, a group element only needs a tuple (a, b) which can be represented by a 1665-bit string, with a 1665-bit discriminant Δ which only needs to be declared for once. Thus, switching from RSA group to class group can save the bandwidth by 45.8% per each group element, which makes applying GQ in a trustless distributed setting more appealing.

Multi-signature and its applications. Multi-signature is firstly proposed in [28] which is a joint signing protocol that allows a group of signers to collaboratively generate a compact signature on a common message and requires that the verification time and signature size is constant. Two important applications of multi-signature are digital wallet and asset custody. Digital wallet usually requires its user to split his secret key into multiple devices and use all (or some) of them to transfer the currencies he holds. Asset custody is a bank service of protecting customer's currencies or real assets. For security consideration, any one single entity (bank, customer, or some third party institution) can not access the secret key directly, especially for some large amount of currencies protected, so the secret key should be also divided into multiple shares. Here are two major concerns: can we resist misbehaved devices/parties? And can we identify who is misbehaving?

Intuitions. In this work, we focus on constructing a *trustless* multi-signature scheme, allowing *key aggregation* and *identifiable abort* properties.

1. *trustless property* requires a non-trusted setup and security against the existence of any number of malicious participants during all phases (for both key setup or signing).
2. Although the dishonest majority model in [22] can well capture this security requirement, abort is not a violation of its security definition. Then, a malicious adversary can easily initiate DoS (Denial of Service) attack on the system. Thus we require an *identifiable abort property*, which is defined in [26], ensuring that the identities of the malicious participants leading to system abort are detectable to any participants or external entities, which is significant to detect broken or hacked devices or misbehaving banks or institutions which cause the failure the joint signing. In our proposed scheme, we achieve a stronger identifiable abort in a *timely manner*, which means the user can timely identify the incorrect messages sent from the malicious parties and avoid continuous useless computation.
3. Additionally, we hope our scheme supports *key aggregation*, which means that a signer, instead of using a full list of the public keys (or key shares), only needs an aggregated public key for everyone to verify a signature, thus saving computations and storage for devices with limited computing resources. In this work, we give a pretty nice solution with enough security and promising efficiency using GQ and class group.

1.1 Related Work

Now, we review the multiparty signature protocols built on top of GQ, Schnorr and ECDSA in the past few years.

The state-of-the-art GQ multi-signature (identity based) is proposed by Bellare and Neven (CT-RSA 2006 [3]). It is highly efficient in computation and proved secure using the forking lemma, although the bandwidth is heavier when compared to Schnorr-based multi-signatures which will be discussed later. But they adopted a fragile security model where all the signers are required honest, which is unrealistic to make it work in the presence of dishonest adversaries. We do not consider the key aggregation property since it is an identity based scheme, where there is only one secret key required to initialize the system by a trusted centre.

Bellare and Neven proposed an efficient Schnorr multi-signature scheme (ACM-CCS 2006 [2]) under a *plain public-key model* allowing the existence of dishonest signers. But it does not support key aggregation. In *plain public-key model*, the security against *rogue-key attack*[1] can be achieved without relying on KOSK (Knowledge of Secret Key) assumption like [5, 33] and accordingly reduce some burdensome computation[2]. Maxwell *et. al.* adopted the same *plain public-key*

[1] Rogue-key attack refers to that an adversary can forge multi-signature by arbitrarily choosing his public key, or using a function of the public keys of honest signers.

[2] KOSK well resists rogue-key attack but it requires the proof of knowledge of secret key when mounting attacks by submitting corresponding public keys, and thus incurs expensive computation.

model and proposed a variant of Bellare and Neven's Schnorr multi-signature, called MuSig, which adds the property of key aggregation [34] (DCC 19). Later on, MuSig2 [36] and MuSig-DN [35] are proposed both of which optimize the round complexity of MuSig from 3 rounds to 2 rounds. However, MuSig and MuSig2 have a considerable reduction loss led by *a double-forking technique* [34]. MuSig-DN achieves a deterministic signing at a cost of expensive zero-knowledge proofs. All of above schemes on GQ and Schnorr cannot achieve identifiable abort since there are no check on the correctness on either R_i or s_i. Their can achieve an identifiable abort by adding additional check $g_i^s = R_i X_i^c{}_i$ without any zero-knowledge proofs (not presented in their original paper). But we emphasize that it is not a "timely manner" since until the last round the honest parties know the malicious parties, which is different to the goal of our scheme.

Lindell *et. al.* proposed the first practical threshold ECDSA (ACM-CCS 2018 [31]) and Gennaro *et. al.* proposed a parallel work: the first efficient threshold ECDSA construction relying on game-based security proof (ACM-CCS 2018 [19]), there has been an abundance of follow-up work [8, 10, 16, 18, 20, 43] to improve these two schemes and made remarkable improvements on different aspects, like waiving expensive range proofs, lowering the signing rounds, adding the identifiable abort functionality. All the mentioned threshold ECDSA schemes operate in the *dishonest majority model*, which is much more secure than *plain public-key model*, especially for decentralized and trustless settings. Gennaro and Goldfeder's scheme [20] achieves the identifiable abort which attributes to a specific phase. Gagol et. al.'s scheme [18] achieves the identifiable abort only in the online signing phase, thus marked with ($\surd$) in the identifiable abort option in Table 1.

1.2 Contributions

We give a brief comparison between our proposed GQ multi-signature scheme and the above-mentioned multi-signature/threshold signature schemes in Table 1, which demonstrates that our protocol is well-rounded, with a competitive signing round complexity (4 rounds of interaction), supporting key aggregation and identifiable abort, secure in the dishonest majority model. Our construction can achieve a highly trustless digital wallet and asset custody. We summarize our contributions as follows.

(1) Formal definition and security proof for class group based GQ signature (CL-GQ). Applying class group to GQ signature can make GQ trapdoorless as mentioned in [24] but no formal discussion is given. We first formalize the definition of GQ signature over class group of imaginary quadratic fields, find the suitable hardness assumption *prime root assumption* for CL-GQ, and prove that the existential unforgeability under chosen message attack (EUF-CMA) in the random oracle model (ROM) under the *prime root assumption* implied by the *root assumption* in generic group in [12].

(2) Compact one-round NIZK proofs to resist malicious adversaries and achieve identifiable abort. In order to detect the malicious behaviour during the multi-party signing and the protocol can abort once misbehaving is

Table 1: Comparison with existing multiparty signing schemes. rds is the abbreviation of rounds; n denotes the number of sigining parties;each round allowing broadcasting and a point-to-point message sending is considered one round.

scheme	range proof	key aggregate	identifiable abort*	sign rds.
ECDSA (CCS 18) [19]	√	√	×	9
ECDSA (CCS 18) [31]	√	√	×	8
ECDSA (S&P 19) [16]	×	√	×	6+log(n)
ECDSA (PKC 20) [10]	×	√	×	8
ECDSA (PKC 21) [43]	×	√	×	8
ECDSA (G.G. 20) [20]	√	√	√	7
ECDSA (CCS 20) [8]	√	√	×	4
ECDSA (G.K.S.S. 20) [18]	×	√	(√)	13
Schnorr (CCS 06) [2]	×	×	×	3
Schnorr (DCC 19) [34]	×	√	×	3
Schnorr (CCS 20) [36]	×	√	×	2
Schnorr (N.R.S. 20) [35]	×	√	×	2
GQ (CT-RSA 06) [3]	×	-	×	3
GQ (This paper)	×	√	√	4

detected once the malicious message is recieved (a timely identifiable abort with attributability to the exact malicious message), we design two tailored ZK proofs including ZKPoKRoot and ZKPoKSig following the 3 moves in the traditional Σ-protocol. They promise any messages sent during interactions are verifiable. Our ZK proofs are highly efficient, since no repetition is required after adopting a Beout trick, although the ZK proofs work in an unknown order class group, unlike the binary challenge based ZK proofs in [9, 10]. This Bezout trick nicely solves the problem of how to accelerate the ZK proof of Paillier ciphertext used in Yi's blind ECDSA [42], which will be illustrated in detail in Appendix E in this paper.

(3) Provably secure trustless CL-GQ multi-signature in dishonest majority model. We generalize CL-GQ to a multi-user setting and combine non-malleable equivocable commitment used in [10, 19] and our ZK proofs to build up our trustless CL-GQ multi-signature scheme. Our scheme does not rely on any common reference string (CRS) produced by a trusted party. We reduce the unforgeability of our new multi-signature in *dishonest majority model* to the EUF-CMA of CL-GQ under ROM. Our proof enjoys smaller reduction loss than [34, 36] since we only require one time rewinding when reducing the CL-GQ to prime root assumption and no rewinding when reducing the CL-GQ multi-signature to CL-GQ, differing the *double-forking technique* which needs a two-layer rewinding framework, and it is much more concise than the ECDSA schemes [10, 19] since our simulator does not need to distinguish any non semi-correct executions.

(4) Implementation and efficiency analysis. We implement our protocol in Rust[3] to demonstrate the practical efficiency. One signer only needs 2.1/3.6 seconds to sign a document for 112/128-bit security level in a 5-user setting. We also analyze the concrete bandwidth needed in our scheme. In 128-bit security, our protocol only costs 6 kB (kilobytes) and 10 kB bandwidth for the interactive key generation and interactive signing phases respectively in a 5-signer setting. For signing, the bandwidth of our scheme is about one-third of the bandwidth in [19] since we do not have expensive range proofs led by Paillier encryption or tedious MtA (Multiplication-to-Addition) protocol led by the non-linear structure of ECDSA. Both running time and bandwidth are promising.

2 Preliminaries

2.1 Adversary Model and Security Definitions

Our proposed multi-signature scheme works in a dishonest majority model allowing static corruption which was used in [10,19,20,31]. Following [21], we present a game-based definition of security analogous to EUF-CMA: multi-signature unforgeability under chosen message attacks (MU-CMA). We review the definitions of digital signature, multi-signature, zero-knowledge proof system and non-malleable equivocable commitment in Appendix A.

Dishonest majority model with static corruption. In dishonest majority model, there can exist a majority of malicious adversaries who may arbitrarily deviate from the protocol and abort is not deemed as violating the security, assuming the existence of both broadcast channel and point-to-point channel among each participant, and assuming the static corruption that requires adversaries to select the participants to corrupt ahead of the start of the protocol.

Definition 1 (Multi-signature Unforgeability). *Consider a multi-signature scheme* $\mathcal{MS} = (\mathsf{MKeyGen}, \mathsf{MSign}, \mathsf{Verify})$ *with* N *parties and a PPT malicious adversary* $\mathcal{A}$ *who corrupts at most* $N-1$ *players, given the view of* $\mathsf{MKeyGen}$ *and* MSign *on inputs of adaptively chosen messages, denoted by* $\mathcal{M}$*, and the corresponding signatures on those messages. The multi-signature scheme* $\mathcal{MS}$ *is said to be existentially unforgeable (EUF-CMA) if there is no such a PPT adversary* $\mathcal{A}$ *that can produce, except with negligible probability, a valid signature on a message* $m \notin \mathcal{M}$*.*

2.2 Guillou-Quisquater Signature (GQ)

We review the original GQ signature scheme in [23].

- KeyGen. Choose randomly two large primes p and q and compute $n = pq$. Select an integer v s.t. $0 < v < \phi(n)$ and $gcd(v, \phi(n)) = 1$, where ϕ(n) is the Euler function. Select a hash function $H : \{0,1\}^* \to \mathbb{Z}_{v-1}$. Randomly select the secret key B from $\mathbb{Z}_n$ and compute $J = B^{-v} \bmod n$. Set $PK = (n, v, J, H)$ and $SK = (p, q, B)$.

[3] https://www.rust-lang.org/

- Sign. Randomly select r from $\mathbb{Z}_n$, then compute $T = r^v \bmod n$, $h = H(M, T)$ and $t = rB^h \bmod n$, where M is the message to be signed. Output signature $\sigma = (t, h)$.
- Verify. Upon receiving a signature $\sigma = (t, h)$ of message M, compute $T' = t^v J^h \bmod n$. If $h = H(M, T')$, output 1; otherwise, output 0.

The correctness is by $T' = t^v J^h = (rB^h)^v J^h = r^v (JB^v)^h = r^v = T \bmod n$. According to [4], GQ identification is secure under RSA-OMI (RSA one-more inversion) assumption and after applying Fiat-Shamir transformation, GQ signature is secure under RSA-OMI assumption in ROM (random oracle model).
RSA Trapdoor. If knowing the p and q, a malicious PKG can easily obtain the secret key B from public J through simply computing $d = v^{-1} \bmod (p-1)(q-1)$ and then $B = J^{-d}$. This RSA trapdoor makes the GQ signature infeasible to be used in trustless scenarios.

2.3 Class Group of Imaginary Quadratic Field

Let $-\Delta$ be a random (large) λ-bit prime such that $\Delta \equiv 1 \bmod 4$. The ring $\mathcal{O}_\Delta = \mathbb{Z} + \frac{\Delta+\sqrt{\Delta}}{2}\mathbb{Z}$ is an imaginary quadratic order of discriminant Δ. Its field of fractions is $\mathcal{Q}(\sqrt{\Delta})$. The fractional ideals of $\mathcal{O}_\Delta$ are of the form $q(a\mathbb{Z} + \frac{b+\sqrt{\Delta}}{2}\mathbb{Z})$ with $q \in \mathcal{Q}, \alpha \in \mathbb{Z}^+, b \in \mathbb{Z}$ and $4a|(b^2 - \Delta)$. An ideal is integral if $q = 1$, and it can be represented by a pair (a, b). Two factional ideals $\mathfrak{a}, \mathfrak{b} \in \mathcal{O}_\Delta$ are equivalent if for some non-zero $\alpha \in \mathcal{Q}(\sqrt{\Delta}), \mathfrak{a} = \alpha\mathfrak{b}$. The set of equivalence classes form an Abelian group under ideal multiplication, which is known as the class group of imaginary quadratic order $\mathrm{CL}(\Delta)$. Sometimes we denote the group as D_i, where $i = -\Delta$. One set of equivalence classes can be represented by a unique (a, b) form through a reduction algorithm satisfying that $gcd(a, b, c) = 1, -a < b \leq a \leq c$,and $b \geq 0$ if $a = c$. The class group of imaginary quadratic order D_i is an Abelian group with ideal multiplication. Meanwhile, class group is always finite and the group order is unknown. More description can be found in [24, 25].

3 GQ Signature Scheme without Trapdoor (CL-GQ)

When we replace the RSA group by class group of imaginary quadratic field $CL(\Delta)$, the group order and thus factoring of group order are unknown even to the authority or user who generates the group. Hence, this $n = pq$ trapdoor is perfectly removed. The GQ signature based on class group is portraited below. The main difference between GQ and CL-GQ is in the KeyGen phase, where v has to be a prime and the group is initialized by a prime Δ. Procedures in sign and verification are basically the same as GQ's. Group operations in class group and the necessity of computing modulo. We now describe the details.

- KeyGen. Given the security parameter λ, find a λ-bit prime $-\Delta$ s.t. $\Delta \equiv 1 \bmod 4$ and a λ-bit prime v. Randomly sample a generator B from class group of imaginary quadratic field $CL(\Delta)$. Compute $J = B^{-v}$. Notice that all the

multiplication and exponentiation in class group should be finalized to a reduced form. It is for the unity of representation and to lower computation cost. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_{v-1}$. Set $PK = (\Delta, v, J, H)$ and $SK = (B)$.
- Sign. On input the secret key B and a message M, randomly selects r from $CL(\Delta)$, then compute $T = r^v$, $h = H(M, T)$ and $t = rB^h$. Output signature $\sigma = (t, h)$.
- Verify. Upon receiving a signature $\sigma = (t, h)$ of message M, compute $T' = t^v J^h$ and $h' = H(M, T')$. If $h' = h$, output 1; otherwise, output 0.

Security. Damgård and Koprowski defined *root assumption* [12] working in generic group model, as a generalization of RSA assumption, by describing that given a group element $x \in G$ and a number e, finding a group element y s.t. $y^e = x$ is intractable, where G is a finite Abelian group in which the inverse and multiplication can be efficiently computed. Thus, we define a *prime root assumption* as below, working in class group, which rules out composite exponent and can be directly implied by *root assumption*. By Theorem 1, the EUF-CMA security of CL-GQ can be reduced to *prime root assumption* in ROM.

Definition 2 (Prime root assumption). *We say that a class group of imaginary quadratic fields satisfies prime root assumption for any efficient $\mathcal{A}$ if*

$$\Pr\left[u^v = g : u \leftarrow \mathcal{A}(\Delta, g, v), v \leftarrow \mathsf{Primes}(\lambda), g \xleftarrow{\$} CL(\Delta), \Delta \xleftarrow{\$} \mathsf{Primes}^*(\lambda)\right]$$

is negligible in λ, where $\mathsf{Primes}(\lambda)$ is the set of primes less than 2^λ and $\mathsf{Primes}^*(\lambda)$ is the set of λ-bit primes which are equal to 3 modulo 4.

Theorem 1. *If prime root assumption holds and H is a random oracle, the CL-GQ signature is provably secure in the EUF-CMA model.*

Proof. Suppose $\mathcal{B}$ is given a prime root problem instance (Δ, J^*, v), J^* is a group member in $CL(\Delta)$ and v is a prime. $\mathcal{B}$ tries to find a B^* from $CL(\Delta)$ s.t. $B^{*v} = J^*$ by using an EUF-CMA adversary $\mathcal{A}$ against the CL-GQ signature scheme.

Setup. $\mathcal{B}$ prepares an empty list $\mathcal{H}$, set p as the length of each element in $\mathcal{H}$. $\mathcal{B}$ sends (Δ, v, J^*, H) to adversary $\mathcal{A}$ as the public key.

Oracle Query. $\mathcal{B}$ answers the oracle queries as follows:

- Sign: On input a message M, $\mathcal{B}$ picks some random $t \in CL(\Delta), h \in \mathbb{Z}_p$ and computes $T = t^v J^h$. $\mathcal{B}$ puts (h, T, M) in the list $\mathcal{H}$. (If the value of h is already set in $\mathcal{H}$, $\mathcal{B}$ picks another h and repeats the previous step.) $\mathcal{B}$ returns $\sigma = (t, h)$.
- H: On input (T, M), if (h, T, M) is in the list $\mathcal{H}$, $\mathcal{B}$ returns h. Otherwise, $\mathcal{B}$ picks a random $h \in \mathbb{Z}_p$. $\mathcal{B}$ puts (h, T, M) in the list $\mathcal{H}$ and returns h.

Output. Finally $\mathcal{A}$ outputs an a message M^* and a forged signature $\sigma^* = (t^*, h^*)$. $\mathcal{B}$ can compute $h^* = H(T^*, M^*)$ s.t. $T^* = t^{*v} J^{*h^*}$.

$\mathcal{B}$ rewinds H to the point that (T^*, M^*) was queried, and returns a different $h' \neq h^*$. $\mathcal{B}$ eventually obtains another forgery (t', h') from $\mathcal{A}$. Therefore, we have $t^{*v} J^{*h^*} = t'^v J^{*h'}$ and it can be transformed into $J^{*h^*-h'} = (t'/t^*)^v$.

According to Bezout formula, there exists a unique pair of non-zero integers (k, m) where $0 \leq |k| \leq v-1$ and $0 \leq |m| \leq |h^*-h'|-1$ which is easily computed by Euclidean algorithm s.t.:

$$mv - k(h^* - h') = gcd(v, h^* - h') = 1.$$

Raise equation $J^{*h^*-h'} = (t'/t^*)^v$ to power k, we have:

$$\begin{aligned} J^{*k(h^*-h')} &= (t'/t^*)^{vk} \\ J^{*mv-1} &= (t'/t^*)^{vk} \\ J^* &= \{J^{*m}(t^*/t')^k\}^v \end{aligned}$$

Hence, $\mathcal{B}$ successfully extracts $B^* = J^{*m}(t^*/t')^k$ to solve the problem instance. □

4 Our Multi-Signature Scheme

In this section, we give the construction of our multi-signature scheme, which is a trustless GQ multi-signature with identifiable abort, secure in dishonest majority model. Both distributed key generation and distributed signing have six phases, they will either abort or output a CRS and a valid signature in each phase. We also utilize two zero-knowledge proofs ZKPoKRoot and ZKPoKSig in our protocol, which will be described in details in next section. Here we note that a plausible idea to achieve trustless setup is to use Boneh's distributed RSA key generation method [6] which will not compromise any secret information of each signer to others. The reason why we did not adopt this fashion to construct our GQ multi-signature is that this key generation is only secure assuming all the parties are honest. This contradicts our *dishonest majority* setting.

Parameters and notations. For the security level of 80/112/128-bit security, we set λ (the bit length of the discriminant Δ of class group) 958/1208/1665 according to Appendix C and set $\eta(\lambda)$=160/224/256 bits. Considering the requirement in [23] that h is smaller than v, h and v are set $\eta(\lambda)$ and $\eta(\lambda)+1$ bits respectively. $\mathsf{NextPrime}(x)$ (resp. $\mathsf{PrevPrime}(x)$) is a function using Miller-Rabin prime test to generate the next (resp. previous) nearest prime. $\mathsf{NextPrime}^*(x)$ (resp. $\mathsf{PrevPrime}^*(x)$) is a function using Miller-Rabin prime test to generate the next (resp. previous) nearest prime r such that $r \equiv 1 \mod 4$ after the input integer x. $\mathsf{Com}(x)$ is a non-malleable commitment for a committed value x and $\mathsf{Reveal}(c, d)$ opens the underlying committed value of the non-malleable equivocal commitment where c is a commitment and d is a decommitment.

Table 2: Interactive Key Generation Protocol $\mathsf{IKeyGen}$

	$\mathsf{IKeyGen}(\lambda)$	
P_i		All users $\{P_j\}, i \neq j$
$\delta_i \xleftarrow{\$} \{0,1\}^\lambda$		
$v_i \xleftarrow{\$} \{0,1\}^{\eta(\lambda)+1}$		
$[c_i, d_i] \leftarrow \mathsf{Com}(\delta_i)$		
$[\hat{c_i}, \hat{d_i}] \leftarrow \mathsf{Com}(v_i)$	$\xrightarrow{c_i, \hat{c_i}}$	
	$\xrightarrow{d_i, \hat{d_i}}$	$\delta_i \leftarrow \mathsf{Reveal}(c_i, d_i)$
		$v_i \leftarrow \mathsf{Reveal}(\hat{c_i}, \hat{d_i})$
$\Delta = \mathsf{NextPrime}^*(\oplus_{i=1}^n \delta_i)$		
$v = \mathsf{NextPrime}(\oplus_{i=1}^n v_i)$		
$B_i \xleftarrow{\$} CL(\Delta)$		
$J_i = B_i^{-v}$		
$[c_i^*, d_i^*] \leftarrow \mathsf{Com}(J_i)$	$\xrightarrow{c_i^*}$	
	$\xrightarrow{d_i^*}$	$J_i \leftarrow \mathsf{Reveal}(c_i^*, d_i^*)$
$\pi_i = \mathsf{ZKPoKRoot}((J_i, v) : B_i \vert J_i = B_i^{-v})$	$\xleftrightarrow{\pi_i}$	Abort if proof π fails
$J = \prod_{i=1}^n J_i$		
Set $CRS = (\Delta, v, J, H)$,		
and $PK_i = J_i; SK_i = B_i$		

4.1 Distributed Key Generation

Our distributed key generation algorithm (Table 2) will either abort or output a CRS. ZKPoKRoot is used to promise that public key J_i broadcasted by party P_i is correctly generated. We describe the details as follows.

Phase 1. Each party P_i picks $\delta_i \xleftarrow{\$} \{0,1\}^\lambda$ and $v_i \xleftarrow{\$} \{0,1\}^{\eta(\lambda)+1}$. P_i computes the commitment $[c_i, d_i] \leftarrow \mathsf{Com}(\delta_i)$ and $[\hat{c_i}, \hat{d_i}] \leftarrow \mathsf{Com}(v_i)$. Each P_i broadcasts to all other parties the commitment $(c_i, \hat{c_i})$.

Phase 2. Each P_i broadcasts the decommitment $(d_i, \hat{d_i})$ to all other parties.

Phase 3. After each P_i received all the (δ_j, v_j) generated by every $P_j (j \neq i)$, a collaboratively generated (Δ, v) is computed by $\Delta = \mathsf{NextPrime}^*(\oplus_{i=1}^n \delta_i)$ and $v = \mathsf{NextPrime}(\oplus_{i=1}^n v_i)$. Then, each P_i generate its key pair (B_i, J_i) by $B_i \xleftarrow{\$} CL(\Delta)$ and $J_i = B_i^{-v}$. P_i computes the commitment $[c_i^*, d_i^*] \leftarrow \mathsf{Com}(J_i)$ and broadcasts to all other parties the commitment c_i^*.

Phase 4. Each P_i broadcasts the decommitment d_i^* along with a non-interactive zero-knowledge proof π_i for the relation $\{(J_i, v) : B_i \vert J_i = B_i^{-v}\}$ to all other parties.

Phase 5. Upon receiving π_i from $P_j (j \neq i)$, each P_i checks the validity of π_j. If passing the check, P_i accepts π_j; otherwise, abort.

Phase 6. After each P_i received all the π_j generated by every $P_j (j \neq i)$ and every π_j's validity is proved, a common J is computed by $J = \prod_{i=1}^n J_i$. Each party P_i sets $CRS = (\Delta, v, J), PK_i = J_i; SK_i = B_i$.

Table 3: Interactive Signing Protocol ISign

	$\mathsf{ISign}(\lambda, SK, M)$	
P_i		All users $\{P_j\}, i \neq j$
$r_i \xleftarrow{\$} CL(\Delta)$		
$T_i = r_i^v$		
$[c_i, d_i] \leftarrow \mathsf{Com}(T_i)$	$\xrightarrow{c_i}$	
	$\xrightarrow{d_i}$	$T_i \leftarrow \mathsf{Reveal}(c_i, d_i)$
$\pi_i = \mathsf{ZKPoKRoot}((T_i, v) : r_i \vert T_i = r_i^v)$	$\xleftrightarrow{\pi_i}$	Abort if proof π fails
$T = \prod_{i=1}^n T_i$		
$h = \mathsf{H}(M, T)$		
$t_i = r_i B_i^h$		
$[\hat{c}_i, \hat{d}_i] \leftarrow \mathsf{Com}(t_i)$	$\xrightarrow{\hat{c}_i}$	
	$\xrightarrow{\hat{d}_i}$	$t_i \leftarrow \mathsf{Reveal}(\hat{c}_i, \hat{d}_i)$
$\hat{\pi}_i = \mathsf{ZKPoKSig}((T_i, J_i, t_i, h, v) : (r_i, B_i) \vert$		
$t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v})$	$\xleftrightarrow{\hat{\pi}_i}$	Abort if proof $\hat{\pi}$ fails
$t = \prod_{i=1}^n t_i$		
Output $\sigma = (t, h)$		

4.2 Distributed Signing

Our distributed signing algorithm (Table 3) will either abort or output a valid signature. We use ZKPoKRoot to ensure the well-formedness of commitment T_i and use ZKPoKSig to ensure the well-formedness of response t_i, thus preventing malicious behaviors during the signing phase. We describe the details as follows.

Phase 1. Each party P_i picks $r_i \xleftarrow{\$} CL(\Delta)$ and compute $T_i = r_i^v$. P_i computes the commitment $[c_i, d_i] \leftarrow \mathsf{Com}(T_i)$. Each P_i broadcasts to all other parties the commitment c_i.

Phase 2. Each P_i broadcasts the decommitment d_i along with a non-interactive zero-knowledge proof π_i for the relation $\{(T_i, v) : r_i \vert T_i = r_i^v\}$ to all other parties.

Phase 3. Upon receiving π_j from $P_j (j \neq i)$, P_i checks the validity of each π_j. If it is valid, P_i accepts π_j ; otherwise, abort.

Phase 4. After each P_i received all the T_j and π_j generated by every $P_j (j \neq i)$ and π_j is proved valid, a common $T = \prod_{i=1}^n T_i$ is computed. Then, calculate $h = \mathsf{H}(M, T)$. Each P_i computes $t_i = r_i B_i^h$ and the commitment $[\hat{c}_i, \hat{d}_i] \leftarrow \mathsf{Com}(t_i)$. Each P_i broadcasts to all other parties the commitment $\hat{c}_i$.

Phase 5. Each P_i broadcasts the decommitment $\hat{d}_i$ along with a non-interactive zero-knowledge proof $\hat{\pi}_i$ for the relation $\{(T_i, J_i, t_i, h, v) : (r_i, B_i) \vert t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v}\}$ to all other parties.

Phase 6. Upon receiving $\hat{\pi}_j$ from $P_j (j \neq i)$, each P_i checks the validity of $\hat{\pi}_i$. If it is valid, P_i accepts $\hat{\pi}_i$; otherwise, abort. Each party computes $t = \prod_{i=1}^n t_i$. Output the collaborative signature $\sigma = (t, h)$.

4.3 Verification

When receiving a signature $\sigma = (t, h)$ for the message M, the verification is similar to the original GQ signature scheme. Accept if and only h is equal to $\mathsf{H}(M, T')$ where $T' = t^v J^h$. The correctness follows by $T' = t^v J^h = (\prod_{i=1}^n t_i)^v (\prod_{i=1}^n J_i)^h = (\prod_{i=1}^n r_i B_i^h)^v (\prod_{i=1}^n B_i^{-v})^h = (\prod_{i=1}^n r_i)^v = r^v = T$. Since the operation is based on an unknown order class group and the results produced by class group multiplication and exponentiation is normalized when output, we do not need to modulo the result by any integer. Since the validity of the signature can be checked by any P_j, it is possible for P_i to send P_j the signature if it confirms the validity of this signature. This will not affect security at all. Moreover, non-malleable commitments and zero-knowledge proofs promise that each party cannot deny the message it broadcasts to the network and each message contributing to collaboratively generated signature is well-formed, and thus no malicious behaviors can affect the joint signing. Note that, the verification phase only needs the aggregated key $J = \prod_{i=1}^n J_i$, not the full list of signers' public keys $\{J_i\}_{i\in[1,n]}$.

4.4 Rogue-Key Attack Resistant

In the IKeyGen phase, an adversary, P_{j^*} for example, cannot choose its PK_{j^*} after seeing the public keys of other parties to initiate rogue-key attack. More specifically, he cannot set his public key as $J_{j^*} = B_{j^*}^{-v} (\prod_{i=1, i\neq j^*}^n J_i)^{-1}$ and thus make the aggregated key equal his arbitrarily selected public key $B_{j^*}^{-v}$, in which case he can forge valid multi-signature by himself easily, since he cannot prove the knowledge of the discrete logarithm of J_{j^*} by submitting valid ZKPoKRoot. This rules out the possibility of rogue-key attack following the KOSK assumption.

4.5 Identifiable Abort or Not

If we simply achieve dishonest majority security without identifiable abort, there is no need to generate and verify the well-formedness ZK Proof of t_i in ISign, namely, the ZKPoKSig. Instead, after obtaining t_i, each party directly computes $t = \prod_{i=1}^n t_i$, and verify the validity of $\sigma = (t, h)$, then output this σ if it is valid, abort if it is invalid. This does not violate the dishonest majority model we used. However, without using ZKPoKSig the identity of malicious party cannot be detected in the Phase 5, and thus our scheme cannot reach the property of identifiable abort.

5 Security Proof of Our Multi-Signature Scheme

The security proof of our multi-signature scheme is a reduction to the unforgeability of CL-GQ. If there is a PPT adversary $\mathcal{A}$ which breaks our multi-party CL-GQ, then we can construct a forger $\mathcal{F}$ to use $\mathcal{A}$ to break CL-GQ. $\mathcal{F}$ must simulate the environment of $\mathcal{A}$. Namely, when $\mathcal{A}$ corrupts $\{P_j\}$ where $j \neq 1$, we can construct a $\mathcal{F}$ to simulate honest party P_1 s.t. $\mathcal{A}$'s view of interaction with $\mathcal{F}$

is indistinguishable from $\mathcal{A}$'s view of interaction with P_1. Let $\mathcal{F}$ have the public key (Δ, v, J, H) of CL-GQ and owns the access to the signing oracle of its choice. After a series of queries from $\mathcal{F}$, it can output a forgery signature $\sigma = (t, h)$ for a message M chosen by itself which has never been queried. Different from the security proof of the multiparty ECDSA in [10], $\mathcal{F}$ does not need to distinguish a semi-correct or non semi-correct execution of $\mathcal{A}$ (δ_i in Phase 3, Fig 5 in [10] sent from adversary can be malicious) which makes our proof more concise.

Simulating P_1 in IKeyGen. $\mathcal{F}$ obtains a public key (Δ, v, J, H) from its CL-GQ challenger and he must set up in its simulation with $\mathcal{A}$ this same public key (Δ, v, J, H). This will allow $\mathcal{F}$ to subsequently simulate interactively signing messages with $\mathcal{A}$, using the output of its CL-GQ signing oracle. $\mathcal{F}$ repeats the following steps by rewinding $\mathcal{A}$ until $\mathcal{A}$ sends the correct decommitments for $P_2, ..., P_n$ on both iterations.

1. $\mathcal{F}$ randomly selects $\delta_1 \in \{0,1\}^\lambda$ and $v_1 \in \{0,1\}^{\eta(\lambda)+1}$, computes $[c_1, d_1] \leftarrow \mathsf{Com}(\delta_1)$ and $[\hat{c_1}, \hat{d_1}] \leftarrow \mathsf{Com}(v_1)$ and broadcasts $(c_1, \hat{c_1})$. $\mathcal{F}$ receives $\{c_j, \hat{c_j}\}_{j\in[n], j\neq 1}$.
2. $\mathcal{F}$ broadcasts $(d_1, \hat{d_1})$ and receives $\{d_j, \hat{d_j}\}_{j\in[n], j\neq 1}$. For $i \in [n]$, let $\delta_i \leftarrow \mathsf{Reveal}(c_i, d_i)$ and $v_i \leftarrow \mathsf{Reveal}(\hat{c_i}, \hat{d_i})$.
3. $\mathcal{F}$ randomly selects $\delta_1', v_1' \in \{0,1\}^\lambda$, subject to the condition $\Delta = \mathsf{NextPrime}^*(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v = \mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$. Then $\mathcal{F}$ computes equivocated decommitment $(d_1', \hat{d_1'})$ which reveal δ_1', v_1', rewinds $\mathcal{A}$ to step 2 and broadcasts $(d_1', \hat{d_1'})$.
4. All parties compute the common output $\Delta = \mathsf{NextPrime}^*(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v = \mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$.
5. $\mathcal{F}$ randomly selects $B_1 \in CL(\Delta)$ and computes $J_1 = B_1^{-v}$. Then $\mathcal{F}$ computes $[c_1^*, d_1^*] \leftarrow \mathsf{Com}(J_1)$ and broadcasts to all other parties the commitment c_1^*. $\mathcal{F}$ receives $\{c_j^*\}_{j\neq i}$.
6. $\mathcal{F}$ broadcasts d_1^* and performs a ZKPoKRoot for relation $\{(J_1, v) : B_1 : |J_1 = B_1^{-v}\}$. $\mathcal{F}$ then receives $\{d_j^*\}_{j\neq i}$. For $i \in [n]$, let $J_i \leftarrow \mathsf{Reveal}(c_i^*, d_i^*)$ be the opened commitment value of each party.
7. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 6 and equivocates P_1's commitment to $d_1^{*'}$ so that the revealed value now is $J_1' = J(\prod_{i=2}^n J_i)^{-1}$ and broadcasts $d_1^{*'}$. Then $\mathcal{F}$ simulates ZKPoKRoot.
8. If all the proofs and commitments are correct the protocol continues with $J' = J_1' \prod_{i=2}^n J_i = J$.

Theorem 2. *If the commitment scheme is non-malleable and equivocal and ZKPoKRoot is honest verifier zero-knowledge proof of knowledge, then the* IKeyGen *simulation above is indistinguishable from a real execution in the view of potentially corrupted parties $P_2, P_3, \ldots, P_n$. Moreover, when the simulation does not abort, all parties output Δ, v in step 4 and J in step 8.*

Simulating P_1 in ISign Phase.

1. As in a real execution, $\mathcal{F}$ randomly selects $r_1 \in CL(\Delta)$ and computes $T_1 = r_1^v$. Then $\mathcal{F}$ computes $[c_1, d_1] \leftarrow \mathsf{Com}(T_1)$ and broadcasts to all other parties the commitment c_1. $\mathcal{F}$ receives $\{c_j\}_{j \neq i}$.
2. $\mathcal{F}$ broadcasts d_1 and performs a ZKPoKRoot for relation $\{(T_1, v) : r_1 : |T_1 = r_1^v\}$. $\mathcal{F}$ then receives $\{d_j\}_{j \neq i}$. For $i \in [n]$, let $T_i \leftarrow \mathsf{Reveal}(c_i, d_i)$ be the opened commitment value of each party.
3. $\mathcal{F}$ requests a signature (t, h) for a message M from its CL-GQ signing oracle and computes $T = t^v J^h$ (note that $h = H(M, T)$).
4. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 2 and equivocates P_1's commitment to d_1' so that the revealed value now is $T_1' = T(\prod_{i=2}^n T_i)^{-1}$ and broadcasts d_1'. Then $\mathcal{F}$ simulates ZKPoKRoot.
5. If all the proofs and commitments are correct, all parties compute $T' = T_1' \prod_{i=2}^n T_i = T$, $h' = H(M, T) = h$. $\mathcal{F}$ computes $t_1 = r_1 B_1^{h'}$. and $[\hat{c}_1, \hat{d}_1] \leftarrow \mathsf{Com}(t_1)$. $\mathcal{F}$ broadcasts to all other parties the commitment $\hat{c}_1$. $\mathcal{F}$ receives $\{\hat{c}_j\}_{j \neq i}$.
6. $\mathcal{F}$ broadcasts $\hat{d}_1$ and performs a ZKPoKSig for relation $\{(T_1, J_1, t_1, h) : (r_1, B_1) | t_1 = r_1 B_1^h, T_1 = {r_1}^v, J_1 = {B_1}^{-v}\}$. $\mathcal{F}$ then receives $\{\hat{d}_j\}_{j \neq i}$. For $i \in [n]$, let $t_i \leftarrow \mathsf{Reveal}(\hat{c}_i, \hat{d}_i)$ be the opened commitment of each party.
7. $\mathcal{F}$ rewinds $\mathcal{A}$ to step 5 and equivocates P_1's commitment to $\hat{d'}_1$. The revealed value is $t_1' = t(\prod_{i=2}^n t_i)^{-1}$ and broadcasts $\hat{d'}_1$. Then $\mathcal{F}$ simulates ZKPoKSig.
8. If all the proofs and commitments are correct, all parties compute $t' = t_1' \prod_{i=2}^n t_i = t$ and output $\sigma = (t', h)$.

Theorem 3. *If the commitment scheme is non-malleable and equivocal and ZKPoKRoot and ZKPoKSig are honest verifier zero-knowledge proof of knowledge, then the* ISign *simulation above is indistinguishable from a real execution in the view of potentially corrupted parties $P_2, P_3, ..., P_n$ and on input M the simulation outputs a valid signature $\sigma = (t, h)$ or aborts.*

Finally, we capture the security of our protocol by Theorem 4.

Theorem 4. *Assuming standard CL-GQ is an existentially unforgeable signature scheme; the ZKPoKRoot and ZKPoKSig are honest verifier zero-knowledge proof of knowledge; and the commitment scheme is non-malleable and equivocable, then our GQ multi-signature protocol (IKeyGen, ISign) is an existentially unforgeable multi-signature scheme.*

6 Zero-knowledge Proofs

In this section, we give the detailed construction of ZKPoKRoot and ZKPoKSig which are used in our multi-signature protocol. At the first glance, both ZK proofs seem easy to construct. But one problem of ZK proofs in an unknown order group is that it requires that the challenge is a binary string and thus should be repeated for many rounds to achieve an acceptable soundness error, like the one-bit challenge ZK proofs in [9, 42]. We observe an interesting thing that the Bezout formula utilized in the EUF-CMA of CL-GQ can also be adopted

Table 4: Zero-knowledge Proof ZKPoKRoot for relation $\mathcal{R}_{\mathsf{root}}$

ZKPoKRoot(X, v)		
P_i		$P_j (j \neq i)$
$r \xleftarrow{\$} CL(\Delta)$		
$t = r^v$	$\xrightarrow{t}$	
	$\xleftarrow{k}$	$k \xleftarrow{\$} \{0,1\}^\gamma$
$u = x^{-k} r$	$\xrightarrow{u}$	Check: $u^v = X^k t$

when proving the special soundness of our ZK proofs, which accordingly waive the repetition of our protocol, the additional constraint is that the length of the challenge space should be smaller than v. This trick also answers the open problem in Yi's blind ECDSA scheme [42], that how to speed up their ZK proof of Paillier ciphertext and in Appendix E we give a slightly modified version of the ZK proof they used, which waives any repetition.

6.1 Zero-knowledge Proof for the $-v$-th Root

We define a relation for the $-v$-th root of a class group element x where v is a prime:

$$\mathcal{R}_{\mathsf{root}} = \{(X, v) : x | X = x^{-v}\}.$$

We put forward a zero-knowledge proof of knowledge (ZKPoK) protocol named ZKPoKRoot (Table 4) which is needed in our multi-signature scheme. It should run for only one round to achieve a soundness error of $2^{-\gamma}$ where γ is the length of the challenge space we set in the ZKPoKRoot protocol, additionally required that $1 \leq \gamma \leq v - 1$. x and X are class group elements and v is a prime.

Theorem 5. *The protocol ZKPoKRoot is an honest verifier zero-knowledge proof of knowledge with soundness error* $2^{-\gamma}$ *where* $1 \leq \gamma \leq v - 1$.

6.2 Zero-knowledge Proof of a CL-GQ Signature

We need another one-round ZKPoK protocol named ZKPoKSig (Table 5) for the following relation, where T_i, J_i, B_i are class group elements, h is a positive integer and v is a prime. We set γ as the challenge space which can be used to adjust the soundness error of ZKPoKSig, additionally required that $1 \leq \gamma \leq v - 1$.

$$\mathcal{R}_{\mathsf{sig}} = \{(T_i, J_i, t_i, h, v) : (r_i, B_i) | t_i = r_i B_i^h, T_i = {r_i}^v, J_i = {B_i}^{-v}\}$$

Theorem 6. *The protocol ZKPoKSig is an honest verifier zero-knowledge proof of knowledge with soundness error* $2^{-\gamma}$ *where* $1 \leq \gamma \leq v - 1$.

Table 5: Zero-knowledge Proof ZKPoKSig for relation $\mathcal{R}_{\mathsf{sig}}$

$\mathsf{ZKPoKSig}(T_i, J_i, t_i, h, v)$		
P_i		$P_j (j \neq i)$
$\rho_1, \rho_2 \xleftarrow{\$} CL(\Delta)$		
$\tau_1 = \rho_1^v$		
$\tau_2 = \rho_2^v$		
$\tau_3 = \rho_1^{-h}\rho_2$	$\xrightarrow{\tau_1,\tau_2,\tau_3}$	
	$\xleftarrow{k}$	$k \xleftarrow{\$} \{0,1\}^\gamma$
$u_1 = B_i^{-k}\rho_1$		
$u_2 = r_i^k \rho_2$	$\xrightarrow{u_1,u_2}$	Check: $u_1^v = J_i^k \tau_1$
		Check: $u_2^v = T_i^k \tau_2$
		Check: $u_1^{-h} u_2 = t_i^k \tau_3$

Remarks. To reduce the unnecessary interactions, we adopt Fiat-Shamir transformation [17] to make both ZKPoKRoot and ZKPoKSig non-interactive by replacing the challenge k in each ZKPoK with $H(t)$ and $H(\tau_1, \tau_2, \tau_3)$ respectively where H is a secure hash function. Due to the security level concern, we will set v larger than 161 bits in the joint signing protocol while γ is usually required to be 40/60/80 bits in the industry. Hence, for either ZKPoKRoot or ZKPoKSig, the additional requirement of $1 \leq \gamma \leq v - 1$ is practical.

6.3 ZKPoK with Lower Soundness

Consider an extreme scenario that we want to achieve a strict soundness error, 2^{-1000} for example, Bezout trick can not be applied in the *soundness with extractor* proof since the additional requirement of $1 \leq \gamma \leq v - 1$ does not hold (v is smaller than 257 in our real use, as claimed in Section 4). The γ can only be set 1 to construct the successful extractor. Hence, ℓ repetitions of either ZKPoKRoot or ZKPoKSig are compulsory when we want to achieve a soundness $2^{-\ell}$ where ℓ is a positive integer. The massive running time undermines its practical application. In this case, if a low soundness error should be satisfied, with reasonable computational cost, the LCM *(lowest common multiple)* trick used in [10] can be used to reduce the repeating time and thus remarkably improve the efficiency. To adopt this LCM trick, we need to modify the original ZKPoK protocols in two places: i) change the challenge space of k from $\{0,1\}$ to $\{0,1\}^C$ for some positive integer C and ii) change the repeat time from ℓ to ℓ/C. Through the revisited ZKPoK protocols, the relations, where $y = \mathsf{lcm}(1, 2, 3, ..., 2^C)$, are proved.

$$\mathcal{R}'_{\mathsf{root}} = \{x : X^z = (x^y)^v\}$$

$$\mathcal{R}'_{\mathsf{sig}} = \{(T_i, J_i, t_i, h, v) : (r_i, B_i) | t_i^z = r_i^y (B_i^y)^h, T_i^z = (r_i^y)^v, J_i^z = (B_i{}^y)^v\}$$

Caveat. The major concern of such an LCM trick is that the modified relation is a loosed relation and thus it is questionable if we can initiate any potential attacks, more specifically, forge a witness which holds in the loosed relation but does not hold in the standard relation and this issue is not well discussed in [10].

Table 6: Running time of original GQ and CL-GQ in different security levels.

Level	GQ's $\|\sigma\|$	GQ KeyGen	GQ Sign	CL-GQ's $\|\sigma\|$	CL-GQ KeyGen	CL-GQ Sign
80-bit	1184 bits	30.375 ms	96.130 us	847 bits	221.77 ms	99.250 ms
112-bit	2272 bits	147.94 ms	472.44 us	1433 bits	2.0269 s	300.61 ms
128-bit	3328 bits	455.42 ms	1.1299 ms	1921 bits	6.9179 s	564.09 ms

7 Implementation and Evaluation

We implemented the original GQ signature, the CL-GQ signature, and our multi-party GQ signature without trusted setup in Rust language. We use the Rust library Class[4] to conduct the class group operations, including sampling, reduction, exponentiation and multiplication. It should be noted that this Rust library calls the C library Pari and thus it basically ensures the efficiency of the heavy arithmetic computations for class groups, but can still be improved. We benchmark the running times of both KeyGen and Sign for three schemes. All the programs are executed in a single thread on a MacBook Pro with Intel Core i5 1.4GHz and 16GB RAM.

7.1 Standard GQ v.s. CL-GQ

We compare the standard GQ and CL-GQ in three security levels: 80-bit, 112-bit, 128-bit security, where 80-bit security is insecure and over 112-bit is generally deemed as secure. We set v as $\eta(\lambda)+1$ bits for both GQ and CL-GQ schemes. We compare the signature sizes, running times of both schemes. As observed from results in Table 6, removing the RSA trapdoor is obviously a trade-off of computational efficiency. CL-GQ is much slower for both KeyGen and Sign due to the complicated arithmetic operations for class group in CL-GQ. For signature size, our CL-GQ is much shorter than GQ. Details of computing bandwidth are given in Appendix D.

7.2 Performance of Trustless GQ Multi-signature

We evaluate the running time and bandwidth of multi-party GQ without trusted setup in Tables 7 and 8. The running time is obtained from the median running time among 20 test samples each of which *sequentially executes* the computation of each signer (in fact the protocol can be executed in parallel but here we consider achieving a fair comparison). In a 5-user setting without considering the network constraint, each signer only needs around 2.1 and 3.6 seconds to sign a message in 112-bit and 128-bit security levels respectively. We computed the concrete Bytes needed for multi-party GQ in 112-bit and 128-bit asymmetric security levels, and gave the calculation formula (Notice that in the given formula λ means the length of Δ, instead of a security level 112 or 128). The details of

[4] It is a library for building cryptography based on class groups of imaginary quadratic orders. https://github.com/ZenGo-X/class.

Table 7: Benchmarks of trustless GQ multi-signature.

security level	# Party	Comp. IKeyGen	Comp. ISign	Comm. IKeyGen	Comm. ISign
112-bit security	2	10.908 s	3.139 s	1848 Bytes	2945 Bytes
	3	15.006 s	5.253 s	2771 Bytes	4417 Bytes
	4	19.947 s	7.663 s	3695 Bytes	5889 Bytes
	5	35.295 s	10.505 s	4619 Bytes	7361 Bytes
128-bit security	2	29.206 s	5.569 s	2466 Bytes	4003 Bytes
	3	36.594 s	9.298 s	3698 Bytes	6004 Bytes
	4	40.168 s	13.372 s	4931 Bytes	8005 Bytes
	5	47.825 s	17.991 s	6164 Bytes	10006 Bytes

computing the bandwidth are given in Appendix D. Both bandwidth and running time confirm that our trapdoorless GQ multi-signature is very practical in use. Our bandwidth is only about one-thirds of the bandwidth of joint signing in [19].

$$Comm.cost(\mathsf{IKeyGen}) = n \times \{10 \times \lceil \frac{\lambda - 1}{2} \rceil + 6 \times \eta(\lambda) + 5\} \quad (bits)$$

$$Comm.cost(\mathsf{ISign}) = n \times \{18 \times \lceil \frac{\lambda - 1}{2} \rceil + 4 \times \eta(\lambda) + 9\} \quad (bits)$$

Impacts from the number of users. Consider an N-party setting, since we assume the existence of broadcast channel, each party only computes their commitments and NIZK proofs once, and thus N computations in total are needed. On the receiver's side, however, each party should de-commit the commitments and verify the NIZK proofs received from all other parties, and thus $N(N-1)$ computations in total are needed. The accumulations of $\delta_i, v_i, J_i, T_i, t_i$ are also in $\mathcal{O}(N^2)$ complexity. Hence, the computational burden increases in a non-linear way when participants increase. Besides, as the increasing of the size of Δ and v, the uncertainty of computing $\mathsf{NextPrime}^*$ and $\mathsf{NextPrime}$ is larger, leading to a noticeable variance of running time of IKeyGen. For example, in our 20 test samples of 5-user experiment, the longest running time is up to 70 seconds. On the other hand, the variance of the running time of ISign is trivial.

8 Conclusion

In this paper, we first formalize the class group based GQ signature and then propose a trapdoorless GQ multi-signature scheme with identifiable abort property and only 4 rounds of interaction in the signing phase, secure in the dishonest majority model. We have concise security proof (no need for the simulator to detect a non semi-correct execution) and two compact one-round NIZKs (removing repetitions led by binary challenge). We give a detailed implementation and efficiency analysis which demonstrate that our scheme has promising running time and extraordinary bandwidth.

References

1. Barker, E., Burr, W., Jones, A., Polk, T., Rose, S., Smid, M., Dang, Q.: Recommendation for key management part 3: Application-specific key management guidance. NIST special publication **800**, 57 (2009)
2. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM conference on Computer and communications security - ACM CCS 2006. pp. 390–399 (2006)
3. Bellare, M., Neven, G.: Identity-based multi-signatures from rsa. In: Cryptographers' Track at the RSA Conference - CT-RSA 2007. pp. 145–162. Springer (2007)
4. Bellare, M., Palacio, A.: Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Annual International Cryptology Conference. pp. 162–177. Springer (2002)
5. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: International Workshop on Public Key Cryptography - PKC 2003. pp. 31–46. Springer (2003)
6. Boneh, D., Franklin, M.: Efficient generation of shared rsa keys. In: Annual international cryptology conference - CRYPTO '97. pp. 425–439. Springer (1997)
7. Buchmann, J.A., Williams, H.C.: A key exchange system based on real quadratic fields extended abstract. In: Conference on the Theory and Application of Cryptology - CRYPTO '89. pp. 335–343. Springer (1989)
8. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security - ACM CCS '20'. pp. 1769–1787 (2020)
9. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ecdsa from hash proof systems and efficient instantiations. In: Annual International Cryptology Conference - CRYPTO 2019. pp. 191–221. Springer (2019)
10. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold ec-dsa. In: IACR International Conference on Public-Key Cryptography - PKC 2020. pp. 266–296. Springer (2020)
11. Chu, C.K., Tzeng, W.G.: Optimal resilient threshold gq signatures. Information sciences **177**(8), 1834–1851 (2007)
12. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2002. pp. 256–271. Springer (2002)
13. Delos, O., Quisquater, J.J.: Efficient multi-signature schemes for cooperating entities. In: Workshop on Algebraic Coding. pp. 63–74. Springer (1993)
14. Delos, O., Quisquater, J.J.: An identity-based signature scheme with bounded life-span. In: Annual International Cryptology Conference - CRYPTO '94. pp. 83–94. Springer (1994)
15. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98. pp. 141–150 (1998)
16. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ecdsa from ecdsa assumptions: the multiparty case. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1051–1066. IEEE (2019)
17. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques - CRYPTO '86. pp. 186–194. Springer (1986)

18. Gagol, A., Straszak, D.: Threshold ecdsa for decentralized asset custody. Tech. rep., Cryptology ePrint Archive, Report 2020/498, 2020. https://eprint. iacr. org ... (2020)
19. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ecdsa with fast trustless setup. In: ACM Conference on Computer and Communications Security - ACM CCS 2018 (2018)
20. Gennaro, R., Goldfeder, S.: One round threshold ecdsa with identifiable abort. IACR Cryptol. ePrint Arch. **2020**, 540 (2020)
21. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold dss signatures. In: International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT '96'. pp. 354–371. Springer (1996)
22. Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version **78** (1998)
23. Guillou, L.C., Quisquater, J.J.: A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In: Conference on the Theory and Application of Cryptography - CRYPTO '88. pp. 216–231. Springer (1988)
24. Hamdy, S., Möller, B.: Security of cryptosystems based on class groups of imaginary quadratic orders. In: International Conference on the Theory and Application of Cryptology and Information Security - ASIACRYPT 2000. pp. 234–247. Springer (2000)
25. Hua, L.K.: Introduction to number theory. Springer Science & Business Media (2012)
26. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Annual Cryptology Conference. pp. 369–386. Springer (2014)
27. I.S.I.: Information technology—security techniques— digital signatures with appendix—part 2: Integer factorization based mechanisms. ISO/IEC **14888-2**(2008) (1999)
28. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development (71), 1–8 (1983)
29. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Annual International Cryptology Conference - CRYPTO 2001. pp. 332–354. Springer (2001)
30. Jacobson, M.J.: Subexponential Class Group Computation in Quadratic Orders. Ph.D. thesis, Technische Universit at Darmstadt, Fachbereich Informatik, Darmstadt, Germany (1999)
31. Lindell, Y., Nof, A.: Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In: ACM Conference on Computer and Communications Security - ACM CCS 2018 (2018)
32. Liu, L.S., Chu, C.K., Tzeng, W.G.: A threshold gq signature scheme. In: International Conference on Applied Cryptography and Network Security - ACNS 2003. pp. 137–150. Springer (2003)
33. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques - EUROCRYPT 2006. pp. 465–485. Springer (2006)
34. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Designs, Codes and Cryptography **87**(9), 2139–2164 (2019)
35. Nick, J., Ruffing, T., Seurin, Y.: Musig2: Simple two-round schnorr multisignatures. Tech. rep., Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint. iacr. org ... (2020)

36. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1717–1731 (2020)
37. te Riele, H.J., et al.: Factorization of a 512-bits rsa key using the number field sieve. Announcment on the Number Theory List (NMBRTHRY@ listserv. nodak. edu) (1999)
38. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**(3), 161–174 (1991)
39. Ting, P.y., Huang, D.M., Huang, X.W.: A proxy multi-signature scheme with anonymous vetoable delegation. International Journal of Computer and Information Engineering **3**(5), 1387–1392 (2009)
40. Wang, H., Zhang, Z.F., Feng, D.G.: Robust threshold guillou-quisquater signature scheme. Wuhan University Journal of Natural Sciences **10**(1), 207–210 (2005)
41. Yao, J., Zeng, G.H.: A distributed authentication algorithm based on gq signature for mobile ad hoc networks. Journal of Shanghai Jiaotong University (Science) **11**(3), 346–350 (2006)
42. Yi, X., Lam, K.Y.: A new blind ecdsa scheme for bitcoin transaction anonymity. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security - AsiaCCS '19. pp. 613–620 (2019)
43. Yuen, T.H., Cui, H., Xie, X.: Compact zero-knowledge proofs for threshold ecdsa with trustless setup. In: Public-Key Cryptography – PKC 2021. pp. 481–511. Springer (2021)
44. Yum, D.H., Lee, P.J.: A distributed online certificate status protocol based on gq signature scheme. In: International Conference on Computational Science and Its Applications- ICCSA 2004. pp. 471–480. Springer (2004)

A Background

A.1 Recall multi-signature

We recall the definitions of signature scheme and multi-signature scheme in [5].

Signature scheme. A signature scheme $\mathcal{S}$ consists of three algorithms $\{\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}\}$. The randomized key generation algorithm KeyGen takes a global information $\mathcal{I}$ and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of a secret and a public keys. The global information can contain, for example, a security parameter, a description of the group and its generator, and the description of the hash function. We do not focus on who generates these parameters and assume that they are publicly available. A (possibly) randomized signature generation algorithm $\mathcal{S}$ takes a message M to sign and global info $\mathcal{I}$ and a secret key sk and outputs M along with a signature σ. A deterministic verification algorithm Verify takes a public key pk, a message M and a signature σ and outputs 1 (accepts) if the signature is valid and 0 (rejects) otherwise. In the random oracle model both signing and verification algorithms have access to the random hash oracle. Usually $M \in \{0,1\}^*$. The common requirement is that $\mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathcal{I}, \mathsf{sk}, M)) = 1$ for all $M \in \{0,1\}^*$.

Definition 3 (Existential Unforgeability under Chosen Message Attack (EUF-CMA)). *Given a digital signature scheme* $\mathcal{S} = \{\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}\}$*, consider a PPT adversary* $\mathcal{A}$ *who is given a public key generated by* KeyGen *and the oracle access to the* Sign *which it can adaptively send query messages. Let* $\mathcal{M}$ *be the set of messages queried by* $\mathcal{A}$*. The digital signature scheme* $\mathcal{S}$ *is said to be existentially unforgeable under chosen message attack if there is no such a PPT adversary* $\mathcal{A}$ *that can produce, except with negligible probability, a valid signature on a message* $m \notin \mathcal{M}$*.*

Multi-signature scheme. Let $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ be a group of n players. Let $\mathcal{I}$ be the global information string. The algorithms of a multi-signature scheme $\mathcal{MS} = (\mathsf{MKeyGen}, \mathsf{MSign}, \mathsf{Verify})$ are defined as follows. A randomized key generation algorithm MKeyGen takes a global information $\mathcal{I}$ and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of a secret and a public keys. Each player $P_i \in \mathcal{P}$ runs MKeyGen and as a result obtains a pair of secret and public keys $(\mathsf{sk}_i, \mathsf{pk}_i)$. A possibly randomized multi-signature generation algorithm MSign is an interactive protocol run by an arbitrary subset of players $\mathcal{L} \subseteq \mathcal{P}$ The input of each $P_i \in \mathcal{L}$ is a message $M \in \{0,1\}^*$, the global info $\mathcal{I}$ and the player's secret key sk_i. The output of the algorithm is a triple $\mathcal{T} = (M, \mathcal{L}, \sigma)$ consisting of the message, description of the subgroup $\mathcal{L}$ and the multi-signature. A deterministic verification algorithm Verify takes $(M, \mathcal{L}, \sigma)$ and public keys of all players in $\mathcal{L}$ and $\mathcal{T}$ and outputs 1 (accepts) or 0 (rejects).

A.2 Zero Knowledge Proof System

A proof system for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is a triple of randomized polynomial time algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$, where Setup takes a security parameter 1^λ and outputs a common reference string (CRS) param. P takes as input the param, a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$. V takes as input the param and x and after interaction with P outputs 0 or 1. The transcript between the prover and the verifier is denoted as $\langle \mathsf{V}(\mathsf{param}, x), \mathsf{P}(\mathsf{param}, x, w) \rangle$, and it is equal to 1 if V accepted the transcript. The zero knowledge proof system we use is a canonical Σ Protocol with 3 moves between prover P and verifier V where the first message (from V to P) is a commitment, denoted by t, the second message (from V to P) is a random coin of V, denoted by c, and the third message (from P to V) is a response denoted by z. We require the following properties for our Σ protocol.

Completeness. If $(x, w) \in \mathcal{R}$, prover P with auxiliary input w convinces V with overwhelming probability.
Special soundness. Given two transcripts (t, c, z) and (t, c', z') where $c' \neq c$ and $z \neq z'$ for a statement x, there exists an extractor which produce w s.t. $(x, w) \in \mathcal{R}$ in polynomial time with non-negligible probability.
Honest verifier zero-knowledge (HVZK). Given x and c, there exists a simulator without knowing w which outputs (t, z) such that (t, c, z) is indistinguishable to the real transcript between P with auxiliary input w and V in polynomial time with non-negligible probability.

A.3 Non-Malleable Equivocable Commitment

A non-interactive trapdoor commitment scheme consists of four algorithms KG, Com, Ver, Equiv:

- KG is the key generation algorithm. Given the security parameter, it outputs (pk, tk), pk is the public key of the commitment scheme and tk is the trapdoor.
- Com is the commitment algorithm. Given pk and message M, it outputs $[C_M, D_M] = \mathsf{Com}(pk, M, R)$ where R is obtained by coin tossing. C_M is the commitment string and D_M is the decommitment string.
- Ver is the verification algorithm. On input pk, C and D, it either outputs M or $\perp$.
- Equiv is the algorithm revealing a commitment in any possible way given the trapdoor. It takes as input pk, strings M, R with $[C_M, D_M] = \mathsf{Com}(pk, M, R)$, a message $M' \neq M$ and a string T. If $T = tk$ then Equiv outputs D' such that $\mathsf{Ver}(pk, C_M, D') = M'$.

The correctness holds if $[C_M, D_M] = \mathsf{Com}(pk, M, R)$, then we have $\mathsf{Ver}(pk, C_M, D_M) = M$. Now we review the properties] of information theoretic security, secure binding and non-malleability.

Information theoretic security. For every message pair (M, M') the distributions C_M and $C_{M'}$ are statistically close.

Secure binding. We say that an adversary $\mathcal{A}$ wins if it outputs (C, D, D') s.t. $\mathsf{Ver}(pk, C, D) = M$, $\mathsf{Ver}(pk, C, D') = M'$, $M' \neq M$, $M \neq \perp$, $M' \neq \perp$. We require that for all efficient algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible.

Non-Malleability [15]. A commitment is non-malleable if no adversary $\mathcal{A}$, given a commitment C to message M, is able to produce another commitment C' s.t. after the revealing of C to M, $\mathcal{A}$ can successfully decommit to a related message M'.

Remarks. As in [19], we can use any secure hash function $H : \{0,1\}^* \to \{0,1\}^n$ and instantiate the commitment to x as $h = H(x, r)$, where r is the blind factor uniformly chosen from $\{0,1\}^n$, assuming that *H behaves as a random oracle*. The decommitment contains a blind factor and the committed value x. For a user receiving a commitment h, it can use the later received decommitment (x, r) to check the validity of the commitment x. The validation passes if and only if $h = H(x, r)$. We adopt this *efficient random oracle version* in our implementation.

B Security Proofs

B.1 Proof of Theorem 2

Proof. In simulation, $\mathcal{F}$ does not know the δ_1 and v_1 chosen in real execution, but it chooses a δ_1' and v_1' such that $\Delta = \mathsf{NextPrime}^*(\delta_1' \oplus (\oplus_2^n \delta_i))$ and $v =$

$\mathsf{NextPrime}(v_1' \oplus (\oplus_2^n v_i))$. Let $D = \oplus_2^n \delta_i$ and $V = \oplus_2^n v_i$. Let $S_\delta = \{x \in \{0,1\}^\lambda : \mathsf{PrevPrime}^*(\Delta) < x \oplus D < \Delta - 1\}$ be the set of all element x such that $\Delta = \mathsf{NextPrime}^*(x \oplus D)$ and $S_v = \{x \in \{0,1\}^\lambda : \mathsf{PrevPrime}(v) < x \oplus V < v - 1\}$ be the set of all element x such that $v = \mathsf{NextPrime}(x \oplus V)$. Since δ_1 and v_1 belong to S_δ and S_v respectively and they are chosen uniformly at random, and δ_1' and v_1' are chosen uniformly at random in the same sets, simulation and real execution are indistinguishable in simulation setp 1-4. The only difference in step 5-8 is that $\mathcal{F}$ computes J_1' instead of using J_1. J_1 and $J(\prod_{i=2}^n J_i)^{-1}$ follow the same distribution. Hence, simulation and real execution are indistinguishable.

Moreover, the simulation may fail due to that someone may refuse to decommit after rewinding in step 2 and 7 and that some π_i fails. Since the commitment scheme is non-malleable and equivocal, in step 2 $\mathcal{F}$ can rewind and equivocate the commitment to δ_1 and v_1, and if there are not aborts, all parties decommit to their correct values. As a consequence, all parties output δ and v at the end of step 4. In step 7, all parties compute the correct J using δ and v from the deterministic setup of CL, if not there is an abort caused by the soundness of the proof π_i corresponding to the corrupted P_i. Finally, if no abort has occurred, $\mathcal{F}$ can equivocate the decommitment to J_1 and all parties decommit to the correct values thanks to the non-malleability of the scheme. If no party refuses to decommit after rewinding, the protocol ends with $J' = J_1' \prod_{i=2}^n J_i = J$. □

B.2 Proof of Theorem 3

Proof. The only difference in this simulation is that $\mathcal{F}$ computes t_1' and T_1' instead of using t_1 and T_1. Since t_1 and $t(\prod_{i=2}^n t_i)^{-1}$ follow the same distribution; T_1 and $T(\prod_{i=2}^n T_i)^{-1}$ follow the same distribution. Hence, simulation and real execution are indistinguishable.

Let (t, h) be the signature that $\mathcal{F}$ receives from its signing oracle in step 5. This is a valid signature for message M. We prove that if the protocol terminates, it does so with output $(t' = t, h)$, which is due to the non-malleability property of commitment scheme. Indeed, the revealing should be the same with overwhelming probability if the adversary decommits correctly. □

B.3 Proof of Theorem 4

Proof. By Theorem 2 and 3, $\mathcal{F}$ always knows how to simulate $\mathcal{A}$'s view and all simulations are indistinguishable of real executions of the protocol. Moreover if A, having corrupted up to $n-1$ parties in the GQ multi-signining protocol, outputs a forgery, since $\mathcal{F}$ set up with $\mathcal{A}$ the same public key as it received from its' CL-GQ challenger, $\mathcal{F}$ can use this signature as its own forgery, thus breaking the existential unforgeability of standard CL-GQ.

Denoting $\mathsf{Adv}_{\Pi,\mathcal{A}}^{mu-cma}$, $\mathcal{A}$'s advantage in breaking the existential unforgeability of our multi-signature protocol, and $\mathsf{Adv}_{cl-gq,\mathcal{A}}^{euf-cma}$ the forger $\mathcal{F}$'s advantage in breaking the existential unforgeability of standard CL-GQ, from Theorem 2 and 3 it holds that if ZKPoKRoot and ZKPoKSig are zero-knowledge and

the commitment scheme is non-malleable and equivocable then $|\mathsf{Adv}^{mu-cma}_{\Pi,\mathcal{A}} - \mathsf{Adv}^{euf-cma}_{cl-gq,\mathcal{A}}|$ is negligible in λ. Under the security of the CL-GQ signature scheme proved in Theorem 1, $\mathsf{Adv}^{euf-cma}_{cl-gq,\mathcal{A}}$ is negligible, which implies that $\mathsf{Adv}^{mu-cma}_{\Pi,\mathcal{A}}$ is negligible as well, contradicting the assumption that $\mathcal{A}$ has non-negligible advantage of forging a signature for our protocol. Hence the theorem holds. □

B.4 Proof of Theorem 5 for ZKPoKRoot

Proof. We prove completeness, special soundness and honest verifier zero-knowledge of our ZKPoKRoot protocol.

Completeness. The equation $X^k t = x^{-vk} r^v = (x^{-k} r)^v = u^v$ always holds for any $((X, v) : x) \in \mathcal{R}_{\mathsf{root}}$.

Special soundness. Assuming that the extractor $\mathcal{E}$ can send two challenges k and k' respectively to the prover for a same commitment t, it receives two responses u and u'. We have $u^v = X^k t$ and $u'^v = X^{k'} t$. We have $X^{k'-k} = (u/u')^v$. According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq |\alpha| \leq |k' - k| - 1$ and $0 \leq |\beta| \leq v - 1$ which is easily computed by Euclidean algorithm s.t.:

$$\alpha v - \beta(k' - k) = gcd(v, k' - k) = 1$$

The rightmost equation holds since v is a prime much larger than either k or k'.

Raise equation $X^{k'-k} = (u/u')^v$ to power β, we extract the witness x by:

$$\begin{aligned} X^{\beta(k'-k)} &= (u/u')^{\beta v} \\ X^{\alpha v - 1} &= (u/u')^{\beta v} \\ X &= \{X^{\alpha}(u'/u)^{\beta}\}^v \\ x &= X^{\alpha}(u'/u)^{\beta} \end{aligned}$$

This has a soundness error of $1/2^{\gamma}$ for running one round. The γ is usually set to 40, 60, 80 for different soundness requirements, but in any case γ is smaller than the bit size of v.

Honest verifier zero-knowledge. Given $\mathcal{S}$ randomly chooses $\tilde{u} \in CL(\Delta), \tilde{k} \in \{0,1\}^{\gamma}$. the simulator $\mathcal{S}$ computes $\tilde{t} \leftarrow \tilde{u}^v / X^{\tilde{k}}$. Clearly, the distribution of t in a real execution is statistically close to $\tilde{t}$. □

B.5 Proof of Theorem 6 for ZKPoKSig

Proof. We prove completeness, special soundness and honest verifier zero-knowledge of ZKPoKSig protocol.

Completeness. For any $((T_i, J_i, t_i, h, v) : (r_i, B_i)) \in \mathcal{R}_{\mathsf{sig}}$. The following equations always hold:

$$\begin{aligned} J_i^k \tau_1 &= B_i^{-vk} {\rho_1}^v = (B_i^{-k} \rho_1)^v = u_1^v; \\ T_i^k \tau_2 &= {r_i}^{vk} {\rho_2}^v = (x^k \rho_2)^v = u_2^v; \\ t_i^k \tau_3 &= (r_i B_i^h)^k \rho_1^{-h} \rho_2 = (B_i^{-k} \rho_1)^{-h} r_i^k \rho_2 = u_1^{-h} u_2. \end{aligned}$$

Special soundness. Assuming that the extractor $\mathcal{E}$ can send two challenges k and k' respectively to the prover for a same commitment (τ_1, τ_2, τ_3), it receives two responses (u_1, u_2) and (u_1', u_2'). We have $u_1^v = J_i^k \tau_1, u_2^v = T_i^k \tau_2, u_1^{-h} u_2 = t_i^k \tau_3$ and ${u'}_1^v = J_i^{k'} \tau_1, {u_2'}^v = T_i^{k'} \tau_2, {u'}_1^{-h} u_2' = t_i^{k'} \tau_3$. Observe the τ_1, τ_2, τ_3 are the same in both groups of equations, we have:

$$J_i^{k'-k} = (u_1'/u_1)^v = (u_1/u_1')^{-v}; \tag{1}$$

$$T_i^{k'-k} = (u_2'/u_2)^v; \tag{2}$$

$$t_i^{k'-k} = [(u_1'/u_1)^{-h}(u_2'/u_2)] = (u_1/u_1')^h (u_2'/u_2). \tag{3}$$

According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq |\alpha| \leq |k'-k| - 1$ and $0 \leq |\beta| \leq v-1$ and (W.L.O.G. assuming $k \geq k'$) which is easily computed by Euclidean algorithm s.t.:

$$\alpha v - \beta(k'-k) = gcd(v, k'-k) = 1$$

Raise equations (1) and (2) to power β:

$$\begin{aligned} J_i^{\beta(k'-k)} &= (u_1/u_1')^{-\beta v} \\ J_i^{\alpha v - 1} &= (u_1'/u_1)^{\beta v} \\ J_i &= \{J_i^{-\alpha}(u_1'/u_1)^{\beta}\}^{-v} \end{aligned} \tag{4}$$

$$\begin{aligned} T_i^{\beta(k'-k)} &= (u_2'/u_2)^{\beta v} \\ T_i^{\alpha v - 1} &= (u_2'/u_2)^{\beta v} \\ T_i &= \{T_i^{\alpha}(u_2/u_2')^{\beta}\}^{v} \end{aligned} \tag{5}$$

Apply the $u_1/u_1' = J_i^{(k-k')/v}$ and $u_2'/u_2 = T_i^{(k'-k)/v}$ implied by (1) and (2) and the results of (4) and (5), we imply t_i by the following:

$$\begin{aligned} t_i^{(k'-k)} &= (u_1/u_1')^h (u_2'/u_2) \\ t_i^{(k'-k)} &= (J_i^{(k-k')/v})^h T_i^{(k'-k)/v} \\ t_i &= J_i^{\frac{-h}{v}} T_i^{\frac{1}{v}} \\ t_i &= \{J_i^{-\alpha}(u_1'/u_1)^{\beta}\}^{-v\frac{-h}{v}} \{T_i^{\alpha}(u_2/u_2')^{\beta}\}^{v\frac{1}{v}} \\ t_i &= T_i^{\alpha}(u_2/u_2')^{\beta} \{J_i^{-\alpha}(u_1'/u_1)^{\beta}\}^h \end{aligned}$$

Hence, we extract the witness (r_i, B_i):

$$r_i = T_i^{\alpha}(u_2/u_2')^{\beta}; \quad B_i = J_i^{-\alpha}(u_1'/u_1)^{\beta}.$$

The extraction has a soundness error of $1/2^{\gamma}$ for running one round.

Honest verifier zero-knowledge. Given $\tilde{u_1}, \tilde{u_2} \in CL(\Delta), \tilde{k} \in \{0,1\}^{\gamma}$, the simulator $\mathcal{S}$ computes $\tilde{\tau_1} \leftarrow \tilde{u_1}^v / J_i^{\tilde{k}}, \tilde{\tau_2} \leftarrow \tilde{u_2}^v / T_i^{\tilde{k}}, \tilde{\tau_3} \leftarrow (\tilde{u_1}^{-h} \tilde{u_2}) / t_i^{\tilde{k}}$. Clearly, the distribution of (τ_1, τ_2, τ_3) in a real execution is statistically close to $(\tilde{\tau_1}, \tilde{\tau_2}, \tilde{\tau_3})$.

□

C Description of Security Level

To depict the security level of class group-based cryptographic system, Safuat and Bodo in [24] gave the expected number of MIPS (Million Instruction Per Second) years using GFNS algorithm [37] to factor large integer and using Cl-MPQS algorithm [30] to compute discrete logarithms in class groups should use, as shown in Table 8. They provided an algorithm to compute the order of a class group element, after which both roots and discrete logarithms can be computed by Pohlig-Hellman algorithm, thus showing the equivalence of hardnesses of computing roots and discrete logarithms in class groups. According to [1], the difficulties of factoring $|n| = 2^{1024}, 2^{2048}, 2^{3072}$ are equivalent to 80, 112 and 128 bits asymmetric security level. Thus we obtain that the difficulties of computing the discrete logarithms in class group under $|\Delta| = 2^{687}, 2^{1208}, 2^{1665}$ are respectively 80, 112 and 128 bits secure.

Table 8: Expected computational cost of factoring integers and computing discrete logarithms in class groups

n	Δ	Expected number of MIPS-years
768 bits	540 bits	4.99×10^7
1024 bits	687 bits	6.01×10^{10}
1536 bits	958 bits	5.95×10^{15}
2048 bits	1208 bits	7.05×10^{19}
3072 bits	1665 bits	2.65×10^{26}
4096 bits	2084 bits	5.87×10^{31}

D Details of Computing Bandwidth

For computing class group size, according to [24], each reduced class group represented by (a, b, Δ) satisfies that $-a < b \leq a$ and $a < \sqrt{|\Delta|/3}$. Let λ denote the bit length of Δ. Then, a and b can be denoted by a $\lceil \frac{\lambda-1}{2} \rceil$-bit string and a $\lceil \frac{\lambda-1}{2} \rceil + 1$-bit string ($b$ needs one more bit to represent its sign). Since Δ is already stored in the common public key which is available for every party, we directly use $Cl = 2 \times \lceil \frac{\lambda-1}{2} \rceil + 1$ to represent the bit size of one class group element.

Recall that denote by $\eta(\lambda)$ the length of h under our security parameter λ, i.e., the bit size of discriminant Δ. More precisely, we have $\eta(1208) = 224$ and $\eta(1665) = 256$ corresponding to the 112-bit and 128-bit security respectively. We use the rightmost $\eta(\lambda)$ bits of SHA256 as our hash function H and as the commitment of which the blind factor is also a $\eta(\lambda)$-bit binary string. For zero-knowledge proofs, we require a soundness error of 2^{-40}. Fiat-Shamir transformation is used to make our ZKPoKRoot and ZKPoKSig non-interactive. For

bandwidth, each broadcast message or received message is counted as one transmission and we compute the total bandwidth for one participant. We set v as $\eta(\lambda)+1$ bits when running our protocol. The bit lengths of one ZKPoKRoot proof and one ZKPoKSig proof can be represented by $2 \times Cl$ bits and $5 \times Cl$ bits respectively, in the non-interactive setting where the challenge is computed by the verifier with the predefined hash function and the received commitments, and thus is not included in the proof size.

E Optimization of the ZK Proof in Yi's Blind ECDSA

E.1 Zero-knowledge proof for well-formedness of Paillier ciphertext

A modified Paillier encryption scheme is proposed in [42], where the modulo N is set pqt (p, q, t are primes) and g is set $(1+N)^{pt}$, different from the original Paillier. By this modification, the message space becomes q instead of N and thus the message space can be set the same of the key space of an EC group of order q. The ciphertext is the same as a standard Paillier, which is $C = g^m r^N \mod N^2$. We review the ZK proof which ensures the correctness of the ciphertext encrypted from a modified paillier encryption scheme proposed in [42], where (N, g) is the public key, C is a ciphertext, (m, r) is the witness. The relation and the ZK protocol is described as follows:

$$R = \{(N, g, C) : (m, r) | C = g^m r^N \mod N^2\}$$

1. Prover randomly chooses $m' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}^*_{N^2}$ and computes $C' = g^{m'} r'^N \mod N^2$ and sends C' to the verifier;
2. Verifier randomly chooses $c \in \{0, 1\}$ and sends c to prover.
3. Prover computes $u = m' + cm \mod q, v = r^c r' \mod N^2$ and sends (u, v) to verifier.
4. Verifier outputs 1 if $C^c C' = g^u v^N \mod N^2$; outputs 0 otherwise.

Necessity of repeatition. This is a classic Σ protocol, including three phases of commit, challenge and response. To achieve a soundness error 2^ℓ, this ZK should be repeated by ℓ times. We demenstrate the necessity of such a repetition here: when proving the special soundness, the extractor can manipulate the random tape of the verifier. After a commitment of randomness r' by prover and a challenge c_1 by verifier, extractor rewinds the protocol to commitment phase again where the prover still uses the same randomness r but the verifier uses another challenge c_2. Now we obtain two equations: $v_1 = r^{c_1} r'$ and $v_2 = r^{c_2} r'$. Then we have $r^{c_1 - c_2} = \frac{v_1}{v_2} \mod N^2$. If $c_{i=1,2} \in \{0, 1\}^k$ and k is an integer larger than 1, we have to solve the $(c_1 - c_2) - th$ root of $\frac{v_1}{v_2} \mod N^2$. It is intractable to compute such a root without knowing the factorazation of N according to the DCRA assumption. Thus, $c_{i=1,2}$ has to be sampled from $\{0, 1\}$ and ℓ repeatitions are required to achieve soundness error $2^{-\ell}$.

E.2 ZK waiving repeatition

We change the modified paillier in [42] a little bit by requiring that an $R = r^N \mod N^2$ should be published along with the ciphertext $C = g^m r^N \mod N$. In another word, we adjust the original ciphertext for a message m to a tuple (C, R) where $C = g^m r^N \mod N$ and $R = r^N \mod N^2$). The modified relation $\mathcal{R}^*$ and corresponding ZK as follows.

$$\mathcal{R}^* = \{(N, g, C, R) : (m, r) | C = g^m r^N, R = r^N \mod N^2\}$$

1. Prover randomly chooses $m' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}^*_{N^2}$ and computes $C' = g^{m'} r'^N \mod N^2, R' = r'^N \mod N^2$;
2. Verifier randomly chooses $c \in \{0,1\}^k$ and sends c to prover.
3. Prover computes $u = m' + cm \mod q, v = r^c r' \mod N^2$ and sends (u, v) to verifier.
4. Verifier outputs 1 if $C^c C' = g^u v^N \mod N^2$ and $R^c R' = v^N \mod N^2$; outputs 0 otherwise.

Theorem 7. *The ZK proof for for relation $\mathcal{R}^*$ with challenge c is sampled from $\{0,1\}^k$ where $1 < k < min\{|p|, |q|, |t|\}$ has special soundness with soundness error of $\frac{1}{2^k}$ when repeated for one round.*

Proof. By rewinding, we obtain

$$C^{c_1} C' = g^{u_1} {v_1}^N, C^{c_2} C' = g^{u_2} {v_2}^N \mod N^2$$
$$R^{c_1} R' = v_1^N, \quad R^{c_2} R' = v_2^N \mod N^2$$

After deviding

$$R^{c_1 - c_2} = (\frac{v_1}{v_2})^N \mod N^2$$
$$C^{c_1 - c_2} = g^{u_1 - u_2 \mod q} (\frac{v_1}{v_2})^N \mod N^2$$

and accordingly

$$(CR^{-1})^{c_1 - c_2} = g^{u_1 - u_2 \mod q}$$
$$CR^{-1} = g^{(u_1 - u_2)(c_1 - c_2)^{-1} \mod q}$$

Since $(c_1 - c_2) < min\{p, q, t\}$, then $gcd(N, c_1 - c_2) = 1$.

According to Bezout formula, there exists a unique pair of (α, β) where $0 \leq \alpha \leq c_1 - c_2 - 1$ and $0 \leq \beta \leq N - 1$ (W.L.O.G. assuming $c_2 \geq c_1$) which is easily computed from Enclidean algorithm s.t.:

$$\alpha N - \beta(c_1 - c_2) = gcd(N, c_1 - c_2) = 1.$$

Raise equation $R^{c_1-c_2} = (\frac{v_1}{v_2})^N \mod N^2$ to power β, we have:

$$\begin{aligned} R^{\beta(c_1-c_2)} &= (\frac{v_1}{v_2})^{\beta N} \mod N^2 \\ R^{\alpha N-1} &= (\frac{v_1}{v_2})^{\beta N} \mod N^2 \\ R &= \{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2 \end{aligned}$$

□

Raise equation $C^{c_1-c_2} = g^{u_1-u_2}(\frac{v_1}{v_2})^N \mod N^2$ to power β, we have:

$$\begin{aligned} C^{\beta(c_1-c_2)} &= g^{\beta(u_1-u_2) \mod q}(\frac{v_1}{v_2})^{\beta N} \mod N^2 \\ C^{\alpha N-1} &= g^{\beta(u_1-u_2) \mod q}(\frac{v_1}{v_2})^{\beta N} \mod N^2 \\ C &= C^{\alpha N} g^{-\beta(u_1-u_2) \mod q}(\frac{v_2}{v_1})^{\beta N} \mod N^2 \\ C &= (CR^{-1})^{\alpha N} g^{-\beta(u_1-u_2) \mod q}\{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2 \\ C &= g^{\alpha N(u_1-u_2)(c_1-c_2)^{-1}-\beta(u_1-u_2) \mod q}\{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2 \\ C &= g^{(u_1-u_2)(c_1-c_2)^{-1}(\alpha N-\beta(c_1-c_2)) \mod q}\{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2 \\ C &= g^{(u_1-u_2)(c_1-c_2)^{-1} \mod q}\{R^{\alpha}(\frac{v_2}{v_1})^{\beta}\}^N \mod N^2 \end{aligned}$$

□

Now we extract the witnesses $r = R^{\alpha}(\frac{v_2}{v_1})^{\beta}$ and $m = (u_1 - u_2)(c_1 - c_2)^{-1} \mod q$. Hence, with a single round, the ZK proof for Paillier ciphertext can achieve suitable soundness with an additional requiretment that the length of challenge space smaller than $min\{|p|, |q|, |t|\}$ which is easy to realize, thus grealy improving the efficiency of their blind ECDSA scheme useful to Bitcoin anonymous transactions.