OXFORD

The Computer Journal, 2024, **67**, 1265–1278
https://doi.org/10.1093/comjnl/bxad057
Advance access publication date 2 June 2023
Original Article

Bandwidth-Efficient Zero-Knowledge Proofs For Threshold ECDSA

Handong Cui[1,*], Kwan Yin Chan[1], Tsz Hon Yuen[1], Xin Kang[2] and Cheng-Kang Chu[2]

[1]Department of Computer Science, The University of Hong Kong, Hong Kong SAR, China
[2]Digital Identity and Trustworthiness Laboratory, Huawei Singapore Research Center, Singapore

*Corresponding author: Email: hdcui@cs.hku.hk

In most threshold Elliptic Curve Digital Signature Algorithm (ECDSA) signatures using additively homomorphic encryption, the zero-knowledge (ZK) proofs related to the ciphertext or the message space are the bottleneck in terms of bandwidth as well as computation time. In this paper, we propose a compact ZK proof for relations related to the Castagnos–Laguillaumie (CL) encryption, which is 33% shorter and 29% faster than the existing work in PKC 2021. We also give new ZK proofs for relations related to homomorphic operations over the CL ciphertext. These new ZK proofs are useful to construct a bandwidth-efficient universal composable-secure threshold ECDSA without compromising the proactive security and the non-interactivity. In particular, we lowered the communication and computation cost of the key refresh algorithm in the Paillier-based counterpart from $O(n^3)$ to $O(n^2)$. Considering a 5-signer setting, the bandwidth is better than the Paillier-based counterpart for up to 99, 95 and 35% for key generation, key refreshment and pre-signing, respectively.

Keywords: ECDSA; threshold signature; zero-knowledge proof; bandwidth-efficient

1. INTRODUCTION

Threshold signatures [1] allow multiple parties to sign a message jointly and the resulting signature can be verified with a single public key. Threshold Elliptic Curve Digital Signature Algorithm (ECDSA) signatures can be easily integrated into existing blockchain systems using ECDSA signatures. In Bitcoin, users can create a *multisig* address of n public keys to support the functionality of threshold signing using a naive way: t ECDSA signatures are generated and the signatures are valid only if they can be verified by any t out of n public keys in the address. However, the signature size is large ($O(t)$) in this naive approach and it results in a higher transaction cost. Therefore, threshold ECDSA signatures are useful to replace the existing *multisig* address in Bitcoin.

Many existing threshold ECDSA signatures [2–7] use an additive homomorphic encryption with a zero-knowledge (ZK) proof as a building block. Some schemes [2, 4, 6] use the Paillier encryption [8]. However, the size of the message space for the Paillier encryption does not match the order of the elliptic curve group used by ECDSA. Therefore, a range proof is needed to ensure that the encrypted message is still within a suitable range. This range proof is expensive in terms of bandwidth. A new additive homomorphic encryption is proposed by Castagnos and Laguillaumie [9] (CL encryption) and it can be implemented in a class group of imaginary quadratic order. The advantage of the CL encryption is that the size of the message space can be set as the order of the elliptic curve group. Some recent threshold ECDSA schemes [3, 5] used the CL encryption to lower the communication bandwidth, without the need of the additional range proof. It comes with a price of a higher computation cost for the signers.

1.1. Motivation

ZK proofs related to the class group and the CL encryption are complicated since they are mostly related to relations in an unknown order group G with a known order subgroup F. Earlier scheme [10] used a ZK proof with a single bit challenge and repeated it for λ_s times for a soundness error of $2^{-\lambda_s}$ due to the inability to compute the inverse in $\mathcal{Z}$ (instead of $\mathcal{Z}_{\mathrm{II}}$, q is the unknown group order) when building up soundness extractor. For the discrete logarithm (DL) relation in G, a *lowest common multiple* trick [3] was proposed to reduce the proof size of [10] by 10 times, at the cost of relaxing the relation to a more loosed one. Yuen *et al.* [5] proposed a compact ZK proof for the (generalized) DL relation which further reduced the proof size of [3] by around four times.

A ZK proof for the well-formedness of a CL ciphertext is also used in the signing phase of two-party ECDSA [10] and threshold ECDSA [3]. Castagnos *et al.* [3] proposed a ZK proof which is around eight times shorter than the single bit challenge ZK proof in [10], at a cost of running n extra ZK proofs for the DL relation in G during the key generation phase by the n participating parties. Yuen *et al.* [5] proposed an alternative solution which is only four times shorter than [10], but instead requires t extra ZK proofs for the DL relation by the t signing parties.

Recently, a threshold ECDSA using the CL encryption was proposed by Castagnos *et al.* [7], with the new properties of non-interactive signing and identifiable abort. However, it cannot achieve universal composable (UC) security. For the theoretical

Received: April 13, 2022. **Revised:** March 13, 2023. **Accepted:** April 29, 2023

complexity, the scheme from [7] reduces the bandwidth consumption of the Paillier-based threshold ECDSA [11] by up to a factor of 10. However, [7] requires an expensive interactive setup as [3]. Another threshold ECDSA using CL encryption was proposed by Deng *et al.* [12], which introduces a weaker soundness called *promise extractability* and waives the interactive setup. However, they cannot achieve the non-interactive signing and proactive security.

1.2. Our Contributions

In this paper, we focus on three research questions. Firstly, we would like to further improve the bandwidth as well as the running time of the above ZK proofs. Secondly, we would like to propose ZK proofs that are useful in other threshold ECDSA signatures, for example, threshold ECDSA with identifiable abort [4], or UC-secure threshold ECDSA [13]. In particular, one commonly used ZK proof is to prove the knowledge of (A, B) such that the plaintext m in a CL ciphertext is transformed to a ciphertext of $Am + B$. This type of relation is called the *affine transformation*, in which the prover does not have the knowledge of m. Thirdly, we evaluate the improvement of using the CL encryption with our new ZK proof for affine transformation in the Paillier-based UC-secure threshold ECDSA signatures [13].

The main contributions are as follows.

- We improve the ZK proofs for DL (and its generalization) and the well-formedness of a CL ciphertext in [5]. The proof size is reduced by at least 33% and the running time for the prover is reduced by at least 29%. The novel ZK proofs can be directly adopted in the two threshold ECDSA settings in [5] achieving more efficient performance.
- We give a new ZK proof for the affine transformation over the CL ciphertext. It can be used for threshold ECDSA with identifiable abort [4], or UC-secure threshold ECDSA [13].
- When applied to UC-secure threshold ECDSA, we have the most bandwidth-efficient UC-secure threshold ECDSA. We lowered the communication and computation cost of the key refresh algorithm in [13] from $O(n^3)$ to $O(n^2)$.

1.3. Overview of Our Construction

Following the settings in the CL encryption [9], we use an unknown order group G, which contains a subgroup F in which the DL problem is tractable. Consider the argument of knowledge for a simple DL relation $\mathcal{R}$ in G for some group elements $g, w \in G \setminus F$:

$$\mathcal{R} = \{x \in \mathbb{Z} : w = g^x\}.$$

For the DL relation in an unknown order group, Boneh *et al.* [14] proposed to use a random prime ℓ as a challenge, and the prover computes d and $e \in [0, \ell - 1]$ such that $x = d\ell + e$. The prover sends $D = g^d$ and e to the verifier to check if $D^\ell g^e = w$. Yuen *et al.* [5] showed that this method does not work if G has a known order subgroup F with a generator f. One possible attack is that $w = g^x f^y$ for some $x, y \in \mathbb{Z}$. The adversary can set $D' = g^d f^{y/\ell}$. The value (D', e) can also pass the verification. Therefore, they proposed another round of challenges using the prime q, the order of F. The prover additionally needs to compute r and $s \in [0, q - 1]$ such that $x = rq + s$. The prover also sends $R = g^r$ and s to the verifier to check if $R^q g^s = w$. This checking eliminates the possibility of having some order q element in w, at a cost of almost doubling the proof size and the running time in the original scheme [14].

Single factorization technique. In this paper, we propose to use a single factorization to replace the two-round factorization used in the argument of knowledge in [5]. Roughly speaking, the verifier picks a random prime $\ell < q$ as a challenge, and the prover computes d' and $e' \in [0, \ell - 1]$ such that $x = d'q\ell + e'$. The prover now sends $D' = g^{d'}$ and e' to the verifier to check if $D'^{q\ell} g^{e'} = w$. This method can also prevent the attack mentioned above, without the need of doubling the proof size and the running time of [14]. This argument of knowledge can be extended to a ZK proof.

One technical obstacle is that we need to handle the soundness proof differently. In previous works [5, 14], the witness x is extracted by rewinding multiple times to obtain (x_i, e_i, ℓ_i) such that $x_i \equiv e_i \bmod \ell_i$. By the Chinese Remainder Theorem (CRT), we can extract a unique witness $x < \prod_i \ell_i$ such that $x \equiv e_i \bmod \ell_i$. Using our single factorization proof, we get $x_i \equiv e'_i \bmod q\ell_i$. They cannot be combined by the CRT directly. We further split the equations into mod q and mod ℓ_i. Since q and ℓ_i are set as distinct primes, these split equations can be combined by the CRT to extract the required witness x. Hence we can achieve soundness in our ZK proof of DL relation in G. We also apply this technique in our ZK proof for the relation $\mathcal{R}_{\text{enc}}$ (knowing the plaintext encrypted in a CL ciphertext) and its variant $\mathcal{R}_{\log}$ (knowing the plaintext of a CL ciphertext is equal to the DL of an element in the ECC group).

ZK proofs for affine transformation. In this paper, we give new ZK proofs related to the affine transformation on a CL ciphertext. Denote C as a CL ciphertext for a plaintext m. Since the CL encryption is additive homomorphic, anyone can turn C into a ciphertext C' which is the encryption of $Am + B$, where A, B is an integer. The prover who perform this transformation can generate a ZK proof for the witness (A, B). We give a compact ZK proof for the affine transformation relation $\mathcal{R}_{\text{aff}}$, as well as two related relations $\mathcal{R}_{\text{aff-p}}$ and $\mathcal{R}_{\text{aff-g}}$ defined in [13].

Applications in UC, non-interactive threshold ECDSA. We demonstrate the advantage of our bandwidth-efficient ZK proofs by transforming the Paillier-based threshold ECDSA [13] into CL-encryption-based[1]. The resulting scheme is UC-secure (composable security), non-interactive (one round online signing phase) and proactive (periodic key refresh).

Using the Paillier encryption [8], the threshold ECDSA in [13] uses a lot of ZK proofs related to the Paillier encryption in the pre-signing phase and the key refresh phase. In the pre-signing phase involving n parties, there are $n(n-1)$ ZK proofs for $\mathcal{R}_{\text{enc}}$, and $2n(n-1)$ ZK proofs for the DL in the Paillier group, i.e. $\mathcal{R}_{\log}$ in [13]. They can be directly changed to $3n$ ZK proofs in our CL-based counterpart if the CL encryption is employed. The bandwidth and computation complexity for these ZK proofs are lowered from $O(n^2)$ to $O(n)$. There are still $2n(n-1)$ ZK proofs for the affine transformation of the Paillier/CL ciphertext in the pre-signing phase. In the key refresh phase involving n parties, [13] used $2n$ ZK proofs for showing a modulus is Paillier–Blum denoted by $\mathcal{R}_{\text{mod}}$, n ZK proofs that s belongs to the multiplicative group generated by t in the Paillier group denoted by $\mathcal{R}_{\text{prm}}$ and $n(n-1)^2$ ZK proofs for the Paillier version of $\mathcal{R}_{\log}$. The overall complexity is $O(n^3)$. When adopting the CL encryption, they can be replaced by only $n(n-1)$ ZK proofs for $\mathcal{R}_{\log}$. The overall complexity is lowered to $O(n^2)$ for both the bandwidth and computation. Similarly, in the key generation phase, we replace the two ZK proofs for $\mathcal{R}_{\text{mod}}$ and one ZK proof for $\mathcal{R}_{\text{prm}}$ with a single ZK proof for $\mathcal{R}_{\text{RepS}}$ described in Section 3.1.

The higher complexity of the number of ZK proofs in [13] is led by that their ZK proofs (ZK proofs other than those for $\mathcal{R}_{\text{mod}}$ and

[1] We note that [11] is the combination of UC-secure threshold ECDSA [13] and threshold ECDSA with identifiable abort [4]. Since the main goal of this paper is about bandwidth efficiency, we do not include identifiable abort for optimal performance.

$\mathcal{R}_{\text{prm}}$) should use the verifier's public Ring–Pedersen parameters. Thus, one relation proved to n different receivers results in n different ZK proofs, while in CL setting there is no such concern, which further reduces the bandwidth.

Another bottleneck of [13] is the tedious ZK proofs for $\mathcal{R}_{\text{mod}}$ and $\mathcal{R}_{\text{prm}}$, denoted by π_{mod} and π_{prm} respectively, used in both key generation and key refreshment. They require $m = 80$ times repetitions in proof generation and verification to achieve a soundness error of 2^{-80}. The repetitions are expensive and require large computation and communication costs, which are unavoidable. Moreover, π_{mod} should be executed twice due to their extraction needs, while the CL setting does not have this concern.

In terms of computation time, we can reduce the running time of the key generation and the key refresh algorithms by significantly reducing the total number of ZK proofs. However, we note that the computation time related to the group used by the CL encryption is higher than the computation time related to the Paillier group. Hence, the running time of our PreSign algorithm is 10+ times more expensive than the Paillier-based counterpart [13]. Nevertheless, the online signing time is still the same as [13]. The running time in the online phase is usually considered to be more important for the online/offline signature.

2. PRELIMINARIES

Notations. We express d being sampled from a distribution $\mathcal{D}$ and b is sampled uniformly in the set B by $d \leftarrow \mathcal{D}$ and $b \xleftarrow{\$} B$, respectively. A negligible (resp. exponential) function is denoted as $\text{negl}(\lambda)$ (resp. $\exp(\lambda)$). We denote $\text{ord}_{\mathbb{G}}(g)$, ϵ_s and ϵ_d as the order of $g \in \mathbb{G}$, the parameter for soundness error and statistical distance, respectively. In particular, we adopt a group where the *hard subgroup membership* assumption [10] holds.

2.1. Groups

We remark some group generation algorithms as in [10]:

- The GGen_{ECC} algorithm generates a cyclic group $\hat{G}$ with prime order q, and returns $\mathcal{G}_{\text{ECC}} = (\hat{G}, q, \hat{P})$, where $\hat{P}$ is a generator of $\hat{G}$. Note that the security parameter input is 1^λ.
- The GGen_{HSM} algorithm returns $\mathcal{G}_{\text{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$ with the inputs of a prime number q and a security parameter 1^λ. The finite abelian group $(\tilde{G}, \cdot)$ is of order $q \cdot \hat{s}$, where the length of $\hat{s}$ is a function of λ and $\gcd(q, \hat{s}) = 1$. The value of the upper bound of $\hat{s}$ is $\tilde{s}$; any element can be decided in polynomial time if it is in $\tilde{G}$. The set $(F, \cdot)$ is the unique cyclic subgroup of $\tilde{G}$ of order q, generated by f. The group generated by g_q is a subgroup of G of order s is denoted as $G^q := \{x^q, x \in G\}$. The cyclic subgroup of $\tilde{G}$ of order $q \cdot s$ is denoted as $(G, \cdot)$, where s divides $\hat{s}$. With the construction of $F \subset G$, it holds that $G = G^q \times F$ and $g := f \cdot g_q$ is the generator of G. A polynomial time algorithm **Solve** solves the DL problem in F:

$$x \leftarrow \text{Solve}_{\mathcal{G}_{\text{HSM},\,q}}(f^x), \forall x \xleftarrow{\$} \mathbb{Z}_q.$$

  We call this *HSM group*, and drop the subscript for Solve in the following paragraphs for simplicity.

Class groups of imaginary quadratic order. The HSM group can be instantiated by class groups of imaginary quadratic order.

The GGen_{HSM} algorithm computes $\Delta_K = -q\tilde{q}$ and $\Delta_q = q^2\Delta_K$, where $\tilde{q}$ is a random prime such that $q\tilde{q} \equiv 1 \pmod 4$ and $(q/\tilde{q}) = -1$.

Denote $\tilde{G}$ as the class group $Cl(\Delta_q)$ with order of $h(\Delta_q) = q \cdot h(\Delta_K)$. It computes $\tilde{s} := \left\lceil \frac{1}{\pi} \log|\Delta_K|\sqrt{|\Delta_K|} \right\rceil$ and thus $h(\Delta_K) < \tilde{s}$.

GGen_{HSM} requires $F = \langle f \rangle$, where $f = [(q, q^2)] \in Cl(\Delta_q)$. It takes a small prime r, where $r \neq q$ and $(\frac{\Delta_K}{r}) = 1$, and sets an ideal lying above r as I. The algorithm computes $g_q = [\varphi_q^{-1}(I^2)]^q \in Cl(\Delta_q)$ and sets $G^q = \langle g_q \rangle$, where φ^{-1} is the surjection defined in the Algorithm 1 of [15]. It finally computes $g = f \cdot g_q$, sets $G = \langle g \rangle$, and outputs $\mathcal{G}_{\text{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$.

2.2. Elliptic Curve Digital Signature Algorithm

The algorithm of ECDSA can be divided into four components; we review them as follows: **Setup**. With the security parameter 1^λ, it runs $\mathcal{G}_{\text{ECC}} \leftarrow \text{GGen}_{\text{ECC}}(1^\lambda)$ and the outputs $\text{param} = \mathcal{G}_{\text{ECC}}$. The input **param** is omitted for simplicity. **KeyGen**. It returns $(\hat{Q}, x)$, where $x \xleftarrow{\$} \mathbb{Z}_q$ is a random secret key, and $\hat{Q} = \hat{P}^x$ is the public key. **Sign**. It computes $\hat{R} = (r_x, r_y) = \hat{P}^k$, $r = r_x \bmod q$, and $s = k^{-1}(xr + H(m)) \bmod q$, where $k \xleftarrow{\$} \mathbb{Z}_q$ and m is an input message. It outputs (r, s) as the signature. **Verify**. It computes $\hat{R} = (r_x, r_y) = (\hat{Q}^r \hat{P}^{H(m)})^{1/s}$ after getting the public key $\hat{Q}$, a message m and a signature (r, s). It outputs 1 if $r = r_x$ and 0 otherwise.

2.3. CL Encryption from HSM Group

A framework of a group with HSM was introduced in [9] for an easy DL subgroup. The algorithm of CL encryption from class group of quadratic fields [10] is divided into six components; we review them as follows:

Setup. Given a prime p and a security parameter 1^λ, it runs $\mathcal{G}_{\text{HSM}} \leftarrow \text{GGen}_{\text{HSM},\,q}(1^\lambda)$ and parses $\mathcal{G}_{\text{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$, then outputs $\text{param} = \mathcal{G}_{\text{HSM}}$. Set $S = \tilde{s} \cdot 2^{\epsilon_d}$, where ϵ_d are some statistical distance. The input **param** is omitted for simplicity.

KeyGen. It returns (**sk**, **pk**), where $\text{sk} \xleftarrow{\$} [0, S]$ is randomly picked and $\text{pk} = g_q^{\text{sk}}$.

Encrypt. With input a message m and a public key **pk**, it returns the ciphertext $C = (C_1, C_2)$ with random $\rho \xleftarrow{\$} [0, S]$, where $C_1 = f^m \text{pk}^\rho$, $C_2 = g_q^\rho$.

Decrypt. With input a ciphertext $C = (C_1, C_2)$ and a secret key **sk**, it returns $m \leftarrow \text{Solve}(M)$, where $M = C_1/C_2^{\text{sk}}$.

EvalScal. With input a scalar s, a ciphertext $C = (C_1, C_2)$ and a public key **pk**, it returns $C' = (C_1' = C_1^s, C_2' = C_2^s)$.

EvalSum. With input two ciphertexts $C = (C_1, C_2)$, $C' = (C_1', C_2')$ and a public key **pk**, it returns $\hat{C} = (\hat{C_1} = C_1C_1', \hat{C_2} = C_2C_2')$.

2.4. Generic Group Model for HSM Group

The HSM group is modeled by the generic group model for groups of unknown order [16] together with groups of known order. Yuen *et al.* [5] generalized the generic group model for HSM Group, and they are reviewed as follows:

A group $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2$ is parameterized by three integer public parameters q, A, B. $\mathbb{G}$ is defined by a random injective function σ: $\mathbb{Z}_{|\mathbb{G}_1| \times q} \to \{0,1\}^\ell$ for some ℓ, where $2^\ell \gg |\mathbb{G}_1| \times q$. The order of $\mathbb{G}_1$ is sampled uniformly from $[A, B]$ and the order of $\mathbb{G}_2$ is q. The group elements are $\sigma(0), \sigma(1), \ldots, \sigma(|\mathbb{G}_1| \times q - 1)$.

A generic algorithm $\mathcal{A}$ is a probabilistic algorithm that takes $(q, \mathcal{L})$ as input, where $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$ is a list that is initialized with the encodings. We further defined a function $\pi(a, b) = qa + b$ for $a \in \mathbb{Z}_{|\mathbb{G}_1|}$ and $b \in \mathbb{Z}_q$. $\mathcal{A}$ queries two generic group oracles:

- $\mathcal{O}_1$ takes $b' \leftarrow \{0, 1\}$. If $b' = 0$, it samples a random $a \in \mathbb{Z}_{|\mathbb{G}_1|}$, $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(a, b))$, which is appended to $\mathcal{L}_0$. If $b' = 1$, it samples a random $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(0, b))$, which is appended to $\mathcal{L}_1$.

- When $\mathcal{L}$ has size $\tilde{q}$, $\mathcal{O}_2(i, j, \pm)$ takes $i, j \in [1, \tilde{q}]$ as indices and a sign bit, and returns $\sigma(\pi(a_i \pm a_j \bmod |\mathbb{G}_1|, b_i \pm b_j \bmod q))$, which is appended to $\mathcal{L}_0$ if $a_i \pm a_j \neq 0 \bmod |\mathbb{G}_1|$. Otherwise, it is appended to $\mathcal{L}_1$.

This model treats the output of $\mathcal{O}_1(1)$ as the elements in F and the output of $\mathcal{O}_1(0)$ as the element in G for the group $\mathcal{G}_{\text{HSM}}$. For some random a, the generator g_q in G^q is initialized as $\sigma(\pi(a, 0))$. It is difficult to distinguish whether it is in G^q, given the output of $\mathcal{O}_1(0)$. Suppose that for some $b^* \in \mathbb{Z}_q$, f is initialized as $\sigma(\pi(0, b^*))$. For input $\tilde{f} \in F$, the Solve algorithm can be modeled by finding the encoding of $\tilde{f}$ in $\mathcal{L}_1$ as $\sigma(\pi(0, \tilde{b}))$ for some $\tilde{b} \in \mathbb{Z}_q$ and returning $\tilde{b}/b^* \bmod q$. We recap two related lemmas from [5].

Lemma 2.1. (Subgroup Element Representation) Let $\mathbb{G}$ be a generic group and $\mathcal{A}$ be a generic algorithm making q_1 queries to $\mathcal{O}_1$ and q_2 queries to $\mathcal{O}_2$. Let $\{g_1, \ldots, g_{m_0}\}$ be the outputs of $\mathcal{O}_1(0)$. There is an efficient algorithm Ext that, given as input the transcript of $\mathcal{A}$'s interaction with the generic group oracles, produces, for every element $u \in \mathbb{G}$ that $\mathcal{A}$ outputs, a tuple $(\alpha_1, \ldots, \alpha_{m_0}) \in \mathbb{Z}^m$ and $\gamma \in \mathbb{Z}_q$ such that $u = f^{\gamma} \cdot \prod_{i=1}^{m_0} g_i^{\alpha_i}$ and $\alpha_i \leq 2^{q_2}$.

Lemma 2.2. (Subgroup Discrete Logarithm) Let $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2$ be a generic group where $|\mathbb{G}_1|$ is a uniformly chosen integer in $[A, B]$ and $1/A$ and $1/|B - A|$ are negligible in λ. Let $\mathcal{A}$ be a polynomial time generic algorithm and let $g_1, \ldots, g_{m_0}$ be the outputs of $\mathcal{O}_1(0)$. The probability that $\mathcal{A}$ succeeds in outputting $\alpha_1, \ldots, \alpha_{m_0}, \beta_1, \ldots, \beta_{m_0} \in \mathbb{Z}$ and $\gamma, \delta \in \mathbb{Z}_q$, such that $f^{\gamma} \cdot \prod_{i=1}^{m_0} g_i^{\alpha_i} = f^{\delta} \cdot \prod_{i=1}^{m_0} g_i^{\beta_i} \in \mathbb{G}$, $\alpha_i \neq \beta_i$ and $\gamma \neq \delta \bmod q$, is negligible.

2.4.1. *Assumptions*

Let $\mathcal{D}$ (resp. $\mathcal{D}_q$) be a distribution over the integers. From the uniform distribution in G (resp. G^q), the distribution over $\{g^x, x \xleftarrow{\$} \mathcal{D}\}$ (resp. $\{g_q^x, x \xleftarrow{\$} \mathcal{D}_q\}$) is at a distance less than 2^{λ}.

Hard Subgroup Membership Assumption. The hard subgroup membership assumption for the group $\mathcal{G}_{\text{HSM}}$ means that is difficult to identify the elements of G^q in G. For every polynomial time algorithm $\mathcal{A}$:

$$\left| Pr\left[b = b^* \;\middle|\; \begin{array}{l} \mathcal{G}_{\text{HSM}} \leftarrow \text{GGen}_{\text{HSM},q}(1^{\lambda}), \\ x \hookleftarrow \mathcal{D}, x' \hookleftarrow \mathcal{D}_q, b \xleftarrow{\$} \{0,1\}, \\ Z_0 = g^x, Z_1 = g_q^{x'}, \\ b^* \leftarrow \mathcal{A}(\mathcal{G}_{\text{HSM}}, Z_b, Solve(\cdot)) \end{array} \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Adaptive Root Subgroup Assumption. The adaptive root subgroup assumption is the modification of the adaptive root assumption [14] in the group $\mathcal{G}_{\text{HSM}}$. Denote Primes(λ) as the set of odd primes less than 2^{λ}.

The adaptive root subgroup assumption holds for the group $\mathcal{G}_{\text{HSM}}$ if for all polynomial time algorithms $(\mathcal{A}_0, \mathcal{A}_1)$:

$$Pr\left[\begin{array}{l} u^{\ell} = w, \\ w^q \neq 1 \end{array} \;\middle|\; \begin{array}{l} q > 2^{\lambda}, \mathcal{G}_{\text{HSM}} \leftarrow \text{GGen}_{\text{HSM},q}(1^{\lambda}), \\ (w, \text{state}) \leftarrow \mathcal{A}_0(\mathcal{G}_{\text{HSM}}), \\ \ell \xleftarrow{\$} \text{Primes}(\lambda), u \leftarrow \mathcal{A}_1(\ell, \text{state}) \end{array} \right] \leq \text{negl}(\lambda).$$

Yuen *et al.* [5] show the intractability of the adaptive root subgroup problem and the non-trivial order element problem in a generic group model.

Corollary 2.1. (Adaptive Root Subgroup Hardness). Let $G \in \mathcal{G}_{\text{HSM}}$ be a generic group where $|\mathcal{G}^q|$ is a uniformly chosen integer in $[A, B]$ such that $1/A$ and $1/|B - A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in breaking the adaptive root subgroup assumption on $\mathcal{G}_{\text{HSM}}$ with at most negligible probability in λ.

Corollary 2.2. (Non-trivial order hardness). Let $G \in \mathcal{G}_{\text{HSM}}$ be a generic group where $|G^q|$ is a uniformly chosen integer in $[A, B]$ such that $1/A$ and $1/|B - A|$ are negligible in λ. Any generic adversary $\mathcal{A}$ that performs a polynomial number of queries to oracle $\mathcal{O}_2$ succeeds in finding an element $h \neq 1 \in G$ and a positive integer d such that $h^d = 1$ and $d < q$ with at most negligible probability in λ.

3. ZK PROOFS FOR HSM GROUP WITH TRUSTLESS SETUP

In this section, we will give a number of ZK proofs for relations that are useful in constructing threshold ECDSA. They are mostly related to the DL and the ciphertext of the CL encryption.

3.1. ZK Proof for Multi-exponentiation

We now construct an argument of knowledge for the following *representation* relation:

$$\mathcal{R}_{\text{RepS}} = \{w \in G; \vec{x} \in \mathbb{Z}^n : w = \prod_{i=1}^{n} g_i^{x_i}\},$$

where $g_1, \ldots, g_n \in G \setminus F$ are in the common reference string $\mathcal{G}_{\text{HSM}}$. This is the generalization of the DL relation (in which $n = 1$). The ZK proof is given in Algorithm 1.

ALGORITHM 1 (ZKPoKRepS).
Param: $\mathcal{G}_{\text{HSM}} \leftarrow \text{GGen}_{\text{HSM},q}(1^{\lambda})$, $g_1, \ldots, g_n \in G \setminus F$, $B = 2^{\epsilon_d + 2\lambda} n q^2 \tilde{s}$ where $\epsilon_d = 80$.
Input: $w \in G$.
Witness: $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$.

1. Prover chooses $k_1, \ldots, k_n \xleftarrow{\$} [-B, B]$ and sends $R = \prod_{i=1}^{n} g_i^{k_i}$ to the verifier.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.
3. Verifier sends $\ell \xleftarrow{\$} \text{Primes}(\lambda)$ to the prover.[2]
4. Prover computes $s_i = k_i + cx_i$ for $i \in [1, n]$. Prover finds $d_i \in \mathbb{Z}$ and $e_i \in [0, q\ell - 1]$ s.t. $s_i = d_i q\ell + e_i$ for $i \in [1, n]$. Prover sends $D = \prod_{i=1}^{n} g_i^{d_i}$ and $\vec{e} = (e_1, \ldots, e_n)$ to the verifier.
5. Verifier accepts if $e_1, \ldots, e_n \in [0, q\ell - 1]$ and $D^{q\ell} \prod_{i=1}^{n} g_i^{e_i} = Rw^c$.

Theorem 3.1. Protocol ZKPoKRepS is an argument of knowledge for $\mathcal{R}_{\text{RepS}}$ in the generic group model.

[2] To reduce the round complexity, we can set $c = H_1(R)$, $\ell = H_2(c)$, where H_1 and H_2 are hash functions which output a number in $[0, q-1]$ and a suitable prime number, respectively.

Proof. Let $(\ell_1, D_1, \vec{e}_1)$ and $(\ell_2, D_2, \vec{e}_2)$ be two accepting transcripts for ZKPoKRepS. We first prove a claim.

Claim. With overwhelming probability for fixed (q, R, c) there exists $\vec{\alpha}$, $\vec{\beta}$ and $\vec{x} = (x_1, ..., x_n)$ such that $\vec{x} = \vec{\alpha}q\ell_1 + \vec{e}_1 = \vec{\beta}q\ell_2 + \vec{e}_2$ and $Rw^c = Rep(\vec{x}) = \prod_{i=1}^n g_i^{x_i}$, and each x_i for $i \in [1, n]$ is bounded by 2^{q_2}.

proof. By the verification equation of this protocol, we have $D_1^{q\ell_1} Rep(\vec{e}_1) = D_2^{q\ell_2} Rep(\vec{e}_2) = Rw^c$. With overwhelming probability the generic group adversary knows $\alpha_1, ..., \alpha_m$ and $\beta_1, ..., \beta_m$ and $m > n$ such that $D_1 = \prod_{i=1}^m g_i^{\alpha_i}$ and $D_2 = \prod_{i=1}^m g_i^{\beta_i}$. By lemma 2.2 with overwhelming probability $\alpha_i q\ell_1 + \vec{e}_1[i] = \beta_i q\ell_2 + \vec{e}_2[i]$ for each $i \le n$ and $\alpha_i\ell_1 = \beta_i\ell_2$ for each $i \in [n+1, m]$ and thus $\ell_1 | \beta_i\ell_2$ for each $i \in [n+1, m]$. We note that ℓ_1 and ℓ_2 are co-prime unless $\ell_1 = \ell_2$ which happens with probability $\frac{\lambda \ln 2}{2^\lambda}$, and that $\alpha_i \le 2^{q_2}$ and α_i is chosen before ℓ_2 is sampled. Hence, with overwhelming probability $\alpha_i = \beta_i = 0$ for each $i \in [n+1, m]$ in which case $Rw^c = \prod_{i=1}^n g_i^{\alpha_i q\ell_1 + \vec{e}_1[i]} = \prod_{i=1}^n g_i^{\beta_i q\ell_2 + \vec{e}_2[i]}$. Setting $\vec{\alpha} = (\alpha_1, ..., \alpha_n)$ and $\vec{\beta} = (\beta_1, ..., \beta_n)$ we conclude with overwhelming probability $Rw^c = Rep(\vec{\alpha}q\ell_1 + \vec{e}_1) = Rep(\vec{\beta}q\ell_2 + \vec{e}_2)$. Finally, if $\mathcal{A}$ has made at most q_2 queries to $\mathcal{O}_2$ then $\alpha_i < 2^{q_2}$ and $\beta_i < 2^{q_2}$ for each i. The claim holds. ■

Consider any third transcript, w.l.o.g. $(\ell_3, D_3, \vec{e}_3)$. Involving this claim again, there exists $\vec{\alpha}\prime, \vec{\beta}\prime, \vec{x}\prime$ in $\mathbb{Z}^n$ such that $\vec{x}\prime = \vec{\alpha}\prime q\ell_2 + \vec{e}_2 = \vec{\beta}\prime q\ell_3 + \vec{e}_3$. Thus with overwhelming probability $\vec{x}\prime - \vec{x} = (\vec{\alpha}\prime - \vec{\beta})q\ell_2$. However, since ℓ_2 is sampled randomly from an exponentially large set of primes independent of $\vec{e}_1, \vec{e}_3, \ell_1$ and ℓ_3 (which fix the value of $\vec{x}\prime - \vec{x}$), there is a negligible probability that $\vec{x}\prime - \vec{x} = 0 \bmod q\ell_2$, unless $\vec{x}\prime - \vec{x} = 0$. We draw a conclusion that by a simple union bound over the poly(λ) number of transcripts, for any polynomial number of accepting transcripts $\{(\ell_i, D_i, \vec{e}_i)\}_{i=1}^{\text{poly}(\lambda)}$, there exists a single $\vec{x}$ such that $\vec{x} = \vec{e}_i \bmod q\ell_i$ for all i.

Now we describe the extractor Ext:

1. Run $\mathcal{A}_0$ to get output (w, state).
2. Let $\mathcal{L} \leftarrow \{\}$. Run Step 1 of Protocol ZKPoKRepS with $\mathcal{A}_1$ on input (w, state).
3. Run Steps 2–4 of Protocol ZKPoKRepS with $\mathcal{A}_1$, sampling fresh randomness c and ℓ for the verifier. If the transcript $(R, c, \ell, D, \vec{e})$ is accepting, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\vec{e}, \ell)\}$, and otherwise repeat this step.
4. Compute $\vec{e}\prime_0 = \vec{e}_1 \bmod q$ and $\vec{e}\prime_i = \vec{e}_i \bmod \ell_i$ for each $(\vec{e}_i, \ell_i) \in \mathcal{L}$. Compute by CRT $\vec{s} = (s_1, \ldots, s_n)$ such that $\vec{s} = \vec{e}\prime_0 \bmod q$ and $\vec{s} = \vec{e}\prime_i \bmod \ell_i$ for each $(\vec{e}_i, \ell_i) \in \mathcal{L}$. If $\prod_{i=1}^n g_i^{s_i} \neq Rw^c$, return to Step 4.
5. Consider the intermediate transcript as $(R, c, \vec{s})$. Run from step 2 for the second time and obtain $(R, c\prime, \vec{s}\prime)$.
6. Compute $\Delta_{s_i} = s_i - s\prime_i$ for $i \in [1, n]$ and $\Delta_c = c - c\prime$. Output $\vec{x} = (x_1, \ldots, x_n)$ for $x_i = \Delta_{s_i}/\Delta_c$.

Analysis for Step 4. Suppose that after some polynomial number of rounds the extractor has obtained M accepting transcripts $(R, c, \ell_i, D_i, \vec{e}_i)$ for independent values of $\ell_i \xleftarrow{\$} \text{Primes}(\lambda)$, by the claim above, with overwhelming probability there exists $\vec{s} = (s_1, \ldots, s_n) \in \mathbb{Z}^n$ such that $\vec{s} = \vec{e}_i \bmod q\ell_i$, and $\prod_{i=1}^n g_i^{s_i} = Rw^c$, $s_j < 2^{q_2}$ for $j \in [n]$. Hence, the CRT algorithm used in Step 4 will recover the required vector $\vec{s}$ once $|\mathcal{L}| > q_2$. Since a single round of interaction with $\mathcal{A}_1$ results in an accepting transcript with probability $\epsilon \ge 1/\text{poly}(\lambda)$, in expectation the extractor obtains $|\mathcal{L}| > q_2$ accepting transcripts for independent primes ℓ_i after $q_2 \cdot \text{poly}(\lambda)$ rounds. Hence, Ext output $\vec{s}$ such that $\prod_{i=1}^n g_i^{s_i} = Rw^c$ in expected polynomial time.

Analysis for Step 6. It remains to argue that Ext succeeds with overwhelming probability in Step 6. W.l.o.g., assume that $c > c'$, by Step 5, we have $\prod_{i=1}^n g_i^{s_i} \cdot w^{-c} = \prod_{i=1}^n g_i^{s_i'} \cdot w^{-c'} = R$. Denote $\Delta_c = c' - c$, $\Delta_s = s_i - s_i'$ for $i \in [0, n]$. We have $\prod_{i=1}^n g_i^{\Delta_{s_i}} = w^{\Delta_c} = (\prod_{i=1}^m g_i^{\alpha_i'} \cdot f^{\gamma'})^{\Delta_c}$ for some $\alpha_i' \in \mathbb{Z}$ and $\gamma' \in \mathbb{Z}_q$ by Lemma 1. By Lemma 2, $\Delta_{s_i} = \alpha_i'\Delta_c$ for $i \in [1, n]$, $\alpha_i' = 0$ for $i \in [n+1, m]$ and $\gamma' = 0 \bmod q$ with overwhelming probability. If $\mu = \prod_{i=1}^n g_i^{\Delta_{s_i}/\Delta_c} \neq w$, then $\mu^{\Delta_c} = w^{\Delta_c}$. It follows that μ/w is an element of order $1 < \Delta_c < q$. By Corollary 2.2, the probability of finding a non-trivial order of $\mu/w \neq 1$ is negligible. Hence, $\mu = w$ with overwhelming probability. It implies that $\Delta_{s_i}/\Delta_c \in \mathbb{Z}$ for all i. Hence, the witness $\vec{x} = (x_1, \ldots, x_n)$ can be extracted as in Step 6. □

Theorem 3.2. The protocol ZKPoKRepS is an honest-verifier statistically zero-knowledge argument of knowledge for relation $\mathcal{R}_{\text{RepS}}$ in the generic group model.

Proof. The simulator Sim picks a random challenge $c' \xleftarrow{\$} [0, q-1]$ and $\ell \xleftarrow{\$} \text{Primes}(\lambda)$. It picks random $d_1', \ldots, d_n' \xleftarrow{\$} [0, B-1]$, $e\prime_1, \ldots, e\prime_n \xleftarrow{\$} [0, q\ell - 1]$. It computes: $D' = \prod_{i=1}^n g_i^{d_i'}$, $R' = D'^{q\ell} \prod_{i=1}^n g_i^{e_i'} \cdot w^{-c'}$.

We argue that the transcript $(R', c', (D', \vec{e}' = (e_1', \ldots, e_n')), \ell')$ is indistinguishable from a real transcript between a prover and a verifier. Sim chooses ℓ', c' identically to the honest verifier. R' is uniquely determined by the other values such that the verification holds.

We must show that in the real protocol, independent of ℓ and c, the values in $\vec{e}$ have a negligible statistical distance from the uniform distribution over $[0, q\ell - 1]$ and each $g_i^{d_i}$ has a negligible statistical distance from uniform over G. In addition, we must argue that D and $\vec{e}$ are independent. For this we use the following facts, which are easy to verify:

1. Fact 1: If Z is a uniform random variable over N consecutive integers and $m < N$, then $Z \bmod m$ has a statistical distance at most m/N from the uniform distribution over $[0, m-1]$.
2. Fact 2: For independent random variables X_1, X_2, Y_1, Y_2, the distance between the joint distributions (X_1, X_2) and (Y_1, Y_2) is at most the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2. Similarly, if these variables are group elements in G, the statistical distance between $X_1 \cdot X_2$ and $Y_1 \cdot Y_2$ is no greater than the sum of statistical distances of X_1 from Y_1 and X_2 from Y_2.
3. Fact 3: Consider random variables X_1, X_2, Y_1, Y_2 with statistical distances $s_1 = \Delta(X_1, Y_1)$ and $s_2 = \Delta(X_2, Y_2)$, where $Pr(X_1 = a | X_2 = b) < Pr(X_1 = a) + \epsilon_1$ and $Pr(Y_1 = a | Y_2 = b) < Pr(Y_1 = a) + \epsilon_2$ for all values a, b. Then the joint distributions (X_1, X_2) and (Y_1, Y_2) have a statistical distance at most $s_1 + s_2 + \epsilon_1 |\text{supp}(X_2)| + \epsilon_2 |\text{supp}(Y_2)|$, where supp is the support.

Consider fixed values of $c, x_1, \ldots, x_n$ and ℓ. In the real protocol, the prover computes $s_i = k_i + cx_i$, where k is uniform in $[-B, B]$ and t is uniform in $\mathbb{Z}_q$, and sets $e_i = s_i \bmod q\ell$. By Fact 1, the value of s_i is distributed uniformly over a range of $2B+1$ consecutive integers, thus e_i has a statistical distance at most $q\ell/(2B+1)$ from uniform over $[0, q\ell - 1]$. This bounds the distance between the real e_i and the simulated e_i' which is uniform over $[0, q\ell - 1]$.

Next, we show that each $g_i^{d_i}$ is statistically indistinguishable from uniform in the subgroup generated by g_i (denoted as G_i). The distribution of $g_i^{d_i}$ over G_i is determined by the distribution of $d_i \bmod |G_i|$. Consider the distribution of $d_i = \lfloor \frac{s_i}{q\ell} \rfloor$ over the consecutive integers in $[\lfloor \frac{cx_i - B}{q\ell} \rfloor, \lfloor \frac{cx_i + B}{q\ell} \rfloor]$. Denote this by the random variable Z. The probability that $d_i = z$ is the probability that s_i falls in the interval $[zq\ell, (z+1)q\ell - 1]$. Hence, $Pr[d_i = z] = q\ell/(2B+1)$ for all $z \in Z$ if $zq\ell \ge cx_i - B$ and $(z+1)q\ell - 1 \le cx_i + B$. This probability

may or may not hold for the two endpoints $E_1 = \lfloor \frac{cx_i - B}{q\ell} \rfloor$ and $E_2 = \lfloor \frac{cx_i + B}{q\ell} \rfloor$. Denote Y as the set of points with $Pr[d_i = z] = q\ell/(2B+1)$ only. The distance of d_i from a uniform random variable U_Y over Y is largest when the number of possible s_i mapping to E_1 and E_2 are both $q\ell - 1$, i.e. $cx_i - B = 1 \bmod q\ell$ and $cx_i + B = q\ell - 2 \bmod q\ell$. In this case, d_i is one of the two endpoints outside Y with probability $\frac{2(q\ell-1)}{2B+1}$. As $|Y| = \frac{2B+3}{q\ell} - 3$, the statistical distance of d_i from U_Y is at most $\frac{1}{2}(|Y|(\frac{1}{|Y|} - \frac{q\ell}{2B+1}) + \frac{2(q\ell-1)}{2B+1}) = \frac{5q\ell-4}{2(2B+1)} < \frac{2q\ell}{B} \le \frac{2^{\lambda+1}q}{B}$. Moreover, the statistical distance of $d_i \bmod |G_i|$ from $U_Y \bmod |G_i|$ is no larger. By Fact 1, $U_Y \bmod |G_i|$ has a statistical distance at most $\frac{|G_i|}{|Y|} \le \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda}$. By the triangle inequality, the statistical distance of $d_i \bmod |G_i|$ from uniform is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda}$. This also bounds the distance of $g_i^{q_i}$ from uniform in G_i. The simulated value q_i' is uniformly chosen from a set of size B. Again by Fact 1, if $|G_i| < B$, then $d_i' \bmod |G_i|$ has a distance $|G_i|/B \le |G|/B$ from uniform. The simulated value $g_i^{d_i'}$ has a distance at most $|G|/B$ from uniform in G_i. By the triangle inequality, the statistical distance of $g_i^{d_i}$ and $g_i^{d_i'}$ is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda} + \frac{|G|}{B} < \frac{(2^\lambda q+2)|G|+2^{\lambda+2}q}{2B+3-3q\cdot 2^\lambda} \le \frac{1}{n2^{\epsilon_d+1}}$, if $B \ge n2^{\epsilon_d}(2^\lambda q+2)|G| + nq\cdot 2^{\epsilon_d+\lambda+2} + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d.

Finally, we consider the joint distribution of $g_i^{d_i}$ and e_i. Consider the conditional distribution of $d_i|e_i$. Note that $d_i = z$ if $(s_i - e_i)/(q\ell) = z$. We repeat a similar argument as above for bounding the distribution of d_i from uniform. For each possible value of z, there always exists a unique value of s_i such that $\lfloor \frac{s_i}{q\ell} \rfloor = z$ and $s_i = 0 \bmod q\ell$, except possibly at the two endpoints E_1, E_2 of the range of d_i. When e_i disqualifies the two points E_1 and E_2, then each of the remaining points $z \notin \{E_1, E_2\}$ still has an equal probability mass, and thus the probability $Pr(d_i = z|e_i)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $d_i|e_i \bmod |G_i|$ and hence the variable $g^{d_i}|e_i$.

We can compare the joint distribution $X_i = (g_i^{d_i}, e_i)$ with the simulated distribution $Y_i = (g_i^{d_i'}, e_i')$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{n2^{\epsilon_d+1}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{n2^{\epsilon_d+1}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1}$. Moreover, as each X_i is independent from X_j for $i \ne j$, we use Fact 2 to bound the distance between joint distributions $(g_1^{d_1}, \ldots, g_n^{d_n}, e_1, \ldots, e_n)$ and $(g_1^{d_1'}, \ldots, g_n^{d_n'}, e_1', \ldots, e_n')$ by the sum of individual distances between each X_i and Y_i, which is at most $\frac{1}{2^{\epsilon_d+1}} + \frac{nq^2\ell^2}{2B+3-3q\ell} + \frac{nq\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+1}} + \frac{nq^2\ell^2}{2B+3-3q\ell} < \frac{1}{2^{\epsilon_d}}$, where the last equality holds if $B \ge 2^{\epsilon_d+2\lambda}nq^2 + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. Finally, this also bounds the distance between $(D, \vec{r})$ and $(D', \vec{r'})$, where $D = \prod_{i=1}^n g_i^{d_i}$ and $D' = \prod_{i=1}^n g_i^{d_i'}$. Combining the two requirements on B (recall the first requirement is $B \ge n2^{\epsilon_d}(2^\lambda q + 2)|G| + nq\cdot 2^{\epsilon_d+\lambda+2} + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$, which can be further required by $B \ge nq\cdot 2^{\epsilon_d+\lambda+1}|G|$; and $|G| = q\cdot s > q$), we obtain a simplified requirement $B \ge nq\cdot 2^{\epsilon_d+2\lambda}|G| = 2^{\epsilon_d+2\lambda}nq^2\tilde{s}$. ■

3.2. ZK Proof for the Well-formedness of a CL Ciphertext

Consider a prover encrypted a message $m \in \mathbb{Z}_q$ using a randomness $\rho \in [0, S]$, for some honestly generated public key $\mathsf{pk} \in G^q$[3]. We present a ZK proof of knowledge of the following relation:

$$\mathcal{R}_{\text{Enc}} = \{(\mathsf{pk}, C_1, C_2); (m, \rho) | C_1 \in G \setminus F; C_2, \mathsf{pk} \in G^q; \rho \in [0, S]; m \in \mathbb{Z}_q : C_1 = f^m \mathsf{pk}^\rho \wedge C_2 = g_q^\rho\}.$$

[3] This condition holds if there is another explicit ZK proof of knowing $\log_{g_q} \mathsf{pk}$ generated by the owner of the decryption key, or pk is honestly generated by the verifier himself.

For the relation $\mathcal{R}_{\text{Enc}}$, we cannot apply the protocol ZKPoKRepS directly since $f \in F$. We propose a new ZK proof ZKPoKEnc for $\mathcal{R}_{\text{Enc}}$ in Algorithm 2.

ALGORITHM 2 (ZKPoKEnc).
Param: $\mathcal{G}_{\text{HSM}} \leftarrow \mathsf{GGen}_{\text{HSM},q}(1^\lambda)$, $B = 2^{\epsilon_d+\lambda+2}q\tilde{s}$, where $\epsilon_d = 80$.
Input: $C_1 \in G \setminus F; C_2, \mathsf{pk} \in G^q$.
Witness: $\rho \in [0, S], m \in \mathbb{Z}_q$, where $S = \tilde{s}\cdot 2^{\epsilon_d}$.

1. Prover chooses $s_\rho \xleftarrow{\$} [-B, B]$, $s_m \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, S_2 = g_q^{s_\rho}$. Prover sends (S_1, S_2) to the verifier.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.
3. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.
4. Prover computes: $u_\rho = s_\rho + c\rho, u_m = s_m + cm \bmod q$. Prover finds $d_\rho \in \mathbb{Z}$ and $e_\rho \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$. Prover computes: $D_1 = \mathsf{pk}^{d_\rho}, D_2 = g_q^{d_\rho}$. Prover sends (u_m, D_1, D_2, e_ρ) to the verifier.
5. Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and: $D_1^{q\ell}\mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c, \quad D_2^{q\ell} g_q^{e_\rho} = S_2 C_2^c$.

Theorem 3.3. The protocol ZKPoKEnc is an argument of knowledge in the generic group model.

Proof. We rewind the adversary on fresh challenges ℓ so that each accepting transcript outputs an (e_ρ, ℓ), where $\rho^* = e_\rho \bmod q\ell$ by ZKPoKRepS with overwhelming probability. We consider the following two cases:
Case 1. If $\mathsf{pk}^{\rho^*} \ne S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{\rho^*})^q \ne (S_1 C_1^c f^{-u_m})^q$, then we have $\mathsf{pk}^{\rho^*} \ne S_1 C_1^c f^{-u_m} = D_1^{q\ell}\mathsf{pk}^{e_\rho}$. Let $\gamma_\rho = \frac{e_\rho - \rho^*}{\ell}$. Then $D_1^q \mathsf{pk}^{\gamma_\rho}$ is an ℓ-th root of $(S_1 C_1^c f^{-u_m})/\mathsf{pk}^{\rho^*} \ne 1$. This breaks the adaptive root subgroup assumption since $(S_1 C_1^c f^{-u_m})^q/(\mathsf{pk}^{\rho^*})^q \ne 1$.
Case 2. If $\mathsf{pk}^{\rho^*} \ne S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{\rho^*})^q = (S_1 C_1^c f^{-u_m})^q$, then $S_1 C_1^c = \mathsf{pk}^{\rho^*} f^{\delta'}$ for some $\delta' \ne u_m \in \mathbb{Z}_q$. Then we have $S_1 C_1^c f^{-u_m} = \mathsf{pk}^{\rho^*} f^{\delta' - u_m} = D_1^{q\ell}\mathsf{pk}^{e_\rho}$. Observe that $\mathsf{pk}^{\rho^*} f^{\delta' - u_m}$ cannot cancel element f because $|\delta' - u_m| < q$. Instead, $D_1^{q\ell}\mathsf{pk}^{e_\rho}$ cancel out f and thus lies in G^q, which leads to contradiction. Hence, by Corollary 2.1, $\mathsf{pk}^{\rho^*} f^{u_m} = S_1 C_1^c$ with overwhelming probability.

By rewinding, the extractor obtains a pair of accepting transcripts with (ρ^*, u_m, c) and (s_ρ', u_m', c'). The extractor can compute $\Delta_{\rho^*} = \rho^* - s_\rho'$ and $\Delta_{u_m} = u_m - u_m' \bmod q$. We denote $\rho = \frac{\Delta_{\rho^*}}{\Delta_c}$ and $m = \frac{\Delta_{u_m}}{\Delta_c} \bmod q$. Hence we have $C_1^{\Delta_c} = (\mathsf{pk}^\rho f^m)^{\Delta_c}$. If $C_1 \ne \mathsf{pk}^\rho f^m$, then $\frac{\mathsf{pk}^\rho f^m}{C_1}$ is a non-trivial element of order $\Delta_c < q$ which contradicts Corollary 2.2.

Note that our scheme includes a sub-protocol ZKPoKRepS on input C_2 w.r.t. bases $g_q \in G \setminus F$. Since ZKPoKRepS is an argument of knowledge, there exists an extractor to extract the same ρ such that $C_2 = g_q^\rho$.

Hence the extractor can output (m, ρ) such that $C_1 = \mathsf{pk}^\rho f^m$, $C_2 = g_q^\rho$. ■

Theorem 3.4. The protocol ZKPoKEnc is an honest-verifier statistically zero- knowledge argument of knowledge for relation $\mathcal{R}_{\text{Enc}}$ in the generic group model.

Proof. The simulator Sim randomly picks a challenge $c' \in [0, q-1]$ and a prime $\ell' \in \text{Prime}(\lambda)$. It picks randomly $u_m' \in \mathbb{Z}_q$, $d_\rho' \in [0, B-1]$ and $e_\rho' \in [0, q\ell' - 1]$.

It computes $D_1' = \mathrm{pk}^{d_\rho'}, D_2' = g_q^{d_\rho'}, S_1' = D_1'^{q\ell'}\mathrm{pk}^{e_\rho'} f^{u_m'} C_1^{-c'}, S_2' = D_2'^{q\ell'} g_q^{e_\rho'} C_2^{-c'}$.

We argue that the simulated transcript $(S_1', S_2', c', u_m', D_1', D_2', e_\rho', \ell')$ is indistinguishable from a real transcript $(S_1, S_2, c, u_m, D_1, D_2, e_\rho, \ell)$ between a prover and a verifier. Sim chooses (ℓ', c') identically to the honest verifier. Both u_m and u_m' are uniformly distributed in $\mathbb{Z}_q$. (S_1', S_2') is uniquely defined by the other values such that the verification holds.

For simulated transcript (D_1', D_2', e_ρ') and real transcript (D_1, D_2, e_ρ), we want to prove that the simulator produces statistically indistinguishable transcripts in that independent of ℓ and c, the values e_ρ have a negligible statistical distance from the uniform distribution over $[0, q\ell - 1]$ and each one of $\mathrm{pk}^{d_\rho}, g_q^{d_\rho}$ has negligible statistical from uniform over $G_k = \langle \mathrm{pk} \rangle, G^q$, respectively; D_1, D_2 and e_ρ are independent.

Consider fixed values of c, ρ and ℓ. In the real protocol, the prover computes $u_\rho = c\rho + s_\rho$, where s_ρ is uniform in $[-B, B]$ and sets $e_\rho = u_\rho \bmod q\ell$. By Fact 1, the value of u_ρ is distributed uniformly over a range of $2B+1$ consecutive integers, thus e_ρ has a statistical distance at most $q\ell/(2B+1)$ from uniform over $[0, q\ell-1]$. This bounds the distance between the real e_ρ and the simulated e_ρ', which is uniform over $[0, q\ell - 1]$.

Next, we show that $g_q^{d_\rho}$ is statistically indistinguishable from uniform in G^q. The distribution of $g_q^{d_\rho}$ over G^q is determined by the distribution of $d_\rho \bmod |G^q|$.

Similar to *Theorem 2*, we conclude that the statistical distance of d_ρ from U_Y is at most $\frac{1}{2}[Y(\frac{1}{Y} - \frac{q\ell}{2B+1}) + \frac{2(q\ell-1)}{2B+1}] = \frac{5q\ell-4}{2(2B+1)} < \frac{2q\ell}{B} \le \frac{2^{\lambda+1}q}{B}$ and the statistical distance of $d_\rho \bmod |G^q|$ from $U_Y \bmod |G^q|$ will not exceed. By Fact 1, $U_Y \bmod |G^q|$ has a statistical distance at most $\frac{|G^q|}{|Y|} \le \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda}$. Thus, by triangle inequality, we have that the statistical distance of $d_\rho \bmod |G^q|$ from uniform is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda}$. This also bounds the distance of $g_q^{d_\rho}$ from uniform in G^q. The simulated value d_ρ' is uniformly chosen from a set of size B. Again by Fact 1, if $|G^q| < B$, then $d_\rho' \bmod |G^q|$ has a distance $|G^q|/B$ from uniform. The simulated value $g_q^{d_\rho'}$ has a distance at most $|G^q|/B$ from uniform in G^q. By the triangle inequality again, the statistical distance of $g_q^{d_\rho}$ and $g_q^{d_\rho'}$ is at most $\frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda} + \frac{|G^q|}{B} < \frac{(2^\lambda q+2)|G^q|+2^{\lambda+2}q}{2B+3-3q\cdot 2^\lambda} \le \frac{1}{2^{\epsilon_d+2}}$ if $B \ge 2^{\epsilon_d+1}(2^\lambda q + 2)|G^q| + 2^{\epsilon_d+\lambda+3}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d. Similarly, the same argument holds for the distances of pk^{d_ρ} and $\mathrm{pk}^{d_\rho'}$. Using Fact 3, the distance between the joint distribution $X_\rho = (\mathrm{pk}^{d_\rho}, g_q^{d_\rho})$ and the simulated distribution $Y_\rho = (\mathrm{pk}^{d_\rho'}, g_q^{d_\rho'})$ is at most $\frac{1}{2^{\epsilon_d+1}}$.

Finally, we consider the joint distribution of $(\mathrm{pk}^{d_\rho}, g_q^{d_\rho})$ and e_ρ. Consider the conditional distribution of $d_\rho | e_\rho$. Note that $r_\rho = z$ if $(s_\rho - e_\rho)/q\ell = z$. We repeat a similar argument as above for bounding the distribution of d_ρ from uniform. For each possible value of z, there always exists a unique value of s_ρ such that $\lfloor \frac{s_\rho}{q\ell} \rfloor = z$ and $s_\rho = 0 \bmod q\ell$, expected possibly at the two endpoints E_1 and E_2 of the range of d_ρ. When e_ρ disqualifies the two points E_1 and E_2, then each of the remaining points $z \notin \{E_1, E_2\}$ still have equal probability mass, and thus the probability $Pr(d_\rho = z | e_\rho)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $(\mathrm{pk}^{d_\rho}, g_q^{d_\rho}) | e_\rho$.

We can compare the joint distribution $X_\rho = (\mathrm{pk}^{d_\rho}, g_q^{d_\rho}, e_\rho)$ with the simulated distribution $Y_\rho = (\mathrm{pk}^{d_\rho'}, g_q^{d_\rho'}, e_\rho')$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{2^{\epsilon_d+1}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{2^{\epsilon_d+1}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+1}} + \frac{q\ell}{2B+3-3q\ell} < \frac{1}{2^{\epsilon_d}}$, where the last equality holds if $B > 2^{\epsilon_d+\lambda}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. This bounds the distance between (D_1, D_2, e_ρ) and (D_1', D_2', e_ρ').

Combining the two requirements on B (recall the first requirement is $B \ge 2^{\epsilon_d+1}(2^\lambda q + 2)|G^q| + 2^{\epsilon_d+\lambda+3}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d), we can simplify the requirement as $B \ge 2^{\epsilon_d+\lambda+2}q\tilde{s}$. ■

3.3. ZK Proof for Affine Transformation for CL Ciphertext

We want to prove a relation between two CL ciphertext c and $\tilde{c}$. Here c is the encryption of k using randomness r, but (k, r) are not treated as witness of the ZK proof. The ciphertext c is transformed to a ciphertext $\tilde{c}$ for a message $k\gamma + \beta$ using the additive homomorphic property of the CL encryption. Define ρ as the randomness of encrypting β.

$$
\begin{aligned}
c &= \mathrm{Encrypt}_{\mathrm{pk}}(k; r) = (C_1 = f^k \mathrm{pk}^r, C_2 = g_q^r) \\
\tilde{c} &= \mathrm{Encrypt}_{\mathrm{pk}}(k\gamma + \beta) \\
&= \mathrm{EvalAdd}(\mathrm{EvalScal}(c, \gamma), \mathrm{Encrypt}_{\mathrm{pk}}(\beta; \rho)) \\
&= \mathrm{EvalAdd}((C_1^\gamma, C_2^\gamma), (f^\beta \mathrm{pk}^\rho, g_q^\rho)) \\
&= (\tilde{C}_1 = C_1^\gamma f^\beta \mathrm{pk}^\rho, \tilde{C}_2 = C_2^\gamma g_q^\rho)
\end{aligned}
$$

So we obtain the following relation:

$$
\begin{aligned}
\mathcal{R}_{\mathrm{Aff}} = \{&(\mathrm{pk}, C_1, C_2, \tilde{C}_1, \tilde{C}_2); (\gamma, \beta, \rho) | \mathrm{pk}, C_2 \in G^q; \\
&C_1 \in G \setminus F; \gamma, \beta \in \mathbb{Z}_q; \rho \in [0, S] : \\
&\tilde{C}_1 = C_1^\gamma f^\beta \mathrm{pk}^\rho \wedge \tilde{C}_2 = C_2^\gamma g_q^\rho\}.
\end{aligned}
$$

ALGORITHM 3 (ZKPoKAff).

Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+3}q^2\tilde{s}$ where $\epsilon_d = 80$.

Input: $C_1, \tilde{C}_1 \in G \setminus F; C_2, \tilde{C}_2, \mathsf{pk} \in G^q$.

Witness: $\rho \in [0, S], \gamma, \beta \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1. Prover chooses $s_\rho, s_\gamma \xleftarrow{\$} [-B, B]$, $s_\beta \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = C_1^{s_\gamma} f^{s_\beta} \mathsf{pk}^{s_\rho}, S_2 = C_2^{s_\gamma} g_q^{s_\rho}$. Prover sends (S_1, S_2) to the verifier.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.
3. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.
4. Prover computes: $u_\beta = s_\beta + c\beta \bmod q, u_\rho = s_\rho + c\rho, u_\gamma = s_\gamma + c\gamma$. Prover finds $d_\rho, d_\gamma \in \mathbb{Z}$ and $e_\rho, e_\gamma \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$ and $u_\gamma = d_\gamma q\ell + e_\gamma$. Prover computes: $D_1 = \mathsf{pk}^{d_\rho}, D_2 = g_q^{d_\rho}, E_1 = C_1^{d_\gamma}, E_2 = C_2^{d_\gamma}$. Prover sends $(u_\beta, D_1, D_2, E_1, E_2, e_\rho, e_\gamma)$ to the verifier.
5. Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:
   $(D_1 E_1)^{q\ell} \mathsf{pk}^{e_\rho} C_1^{e_\gamma} f^{u_\beta} = S_1 \tilde{C}_1^c$,
   $(D_2 E_2)^{q\ell} g_q^{e_\rho} C_2^{e_\gamma} = S_2 \tilde{C}_2^c$.

Theorem 3.5. The protocol ZKPoKAff is an argument of knowledge for $\mathcal{R}_{\mathrm{Aff}}$ in the generic group model.

Proof. We rewind the adversary on fresh challenges ℓ so that each accepting transcript outputs an (e_ρ, e_γ, ℓ), where $\rho^* = e_\rho \bmod q\ell$

and $\gamma^* = e_\gamma \bmod q\ell$ with overwhelming probability. We consider the following two cases:

Case 1. If $\mathrm{pk}^{\rho^*}C_1^{\gamma^*} \neq S_1\tilde{C}_1^c f^{-u_\beta}$ and $(\mathrm{pk}^{\rho^*}C_1^{\gamma^*})^q \neq (S_1\tilde{C}_1^c f^{-u_\beta})^q$, then we have $\mathrm{pk}^{\rho^*}C_1^{\gamma^*} \neq S_1\tilde{C}_1^c f^{-u_\beta} = D_1^{q\ell}E_1^{q\ell}\mathrm{pk}^{e_\rho}C_1^{e_\gamma}$. Let $\chi_\rho = \frac{e_\rho-\rho^*}{\ell}$ and $\chi_\gamma = \frac{e_\gamma-\gamma^*}{\ell}$. Then $D_1^qE_1^q\mathrm{pk}^{\chi_\rho}C_1^{\chi_\gamma}$ is an ℓ-th root of $(S_1\tilde{C}_1^c f^{-u_\beta})/(\mathrm{pk}^{\rho^*}C_1^{\gamma^*}) \neq 1$. This breaks the adaptive root subgroup assumption since $(S_1\tilde{C}_1^c f^{-u_\beta})^q/(\mathrm{pk}^{\rho^*}C_1^{\gamma^*})^q \neq 1$.

Case 2. If $\mathrm{pk}^{\rho^*}C_1^{\gamma^*} \neq S_1\tilde{C}_1^c f^{-u_\beta}$ and $(\mathrm{pk}^{\rho^*}C_1^{\gamma^*})^q = (S_1\tilde{C}_1^c f^{-u_\beta})^q$, then $S_1\tilde{C}_1^c = \mathrm{pk}^{\rho^*}C_1^{\gamma^*}f^{\delta'}$ for some $\delta' \neq u_\beta \in \mathbb{Z}_q$. Then we have $S_1\tilde{C}_1^c f^{-u_\beta} = \mathrm{pk}^{\rho^*}C_1^{\gamma^*}f^{\delta'-u_\beta} = D_1^{q\ell}E_1^{q\ell}\mathrm{pk}^{e_\rho}C_1^{e_\gamma}$. By Lemma 1, write $C_1 = f^\upsilon\prod_{i=1}^{m_0} g_i^{\alpha_i}$. Consider the f parts in both sides, we have $f^{\upsilon\cdot\gamma^*+\delta'-u_\beta} = f^{\upsilon\cdot e_\gamma}$, namely $\upsilon\cdot(\gamma^*-e_\gamma) = u_\beta - \delta' \bmod q$. Since $0 < |u_\beta - \delta'| < q$ and $\gamma^* - e_\gamma = 0 \bmod q$, contradiction happens. Hence, by Corollary 2.1, $\mathrm{pk}^{\rho^*}C_1^{\gamma^*}f^{u_\beta} = S_1\tilde{C}_1^c$ with overwhelming probability.

By rewinding, the extractor obtains a pair of accepting transcripts with $(\rho^*, \gamma^*, u_\beta, c)$ and $(s'_\rho, s'_\gamma, u'_\beta, c')$. The extractor can compute $\Delta_{\rho^*} = \rho^* - s'_\rho$, $\Delta_{\gamma^*} = \gamma^* - s'_\gamma$ and $\Delta_{u_\beta} = u_\beta - u'_\beta \bmod q$. We denote $\rho = \frac{\Delta_{\rho^*}}{\Delta_c}, \gamma = \frac{\Delta_{\gamma^*}}{\Delta c}$ and $\beta = \frac{\Delta_{u_\beta}}{\Delta_c} \bmod q$. Hence we have $\tilde{C}_1^{\Delta_c} = (\mathrm{pk}^\rho C_1^\gamma f^\beta)^{\Delta_c}$. If $\tilde{C}_1 \neq \mathrm{pk}^\rho C_1^\gamma f^\beta$, then $\frac{\mathrm{pk}^\rho f^\beta C_1^\gamma}{\tilde{C}_1}$ is a non-trivial element order $\Delta_c < q$ which contradicts Corollary 2.2.

Note that our scheme includes a sub-protocol **ZKPoKRepS** on input $\tilde{C}_2$ w.r.t. bases $g_q \in G \setminus F$. Since **ZKPoKRepS** is an argument of knowledge, there exists an extractor to extract the same (γ, ρ) such that $\tilde{C}_2 = C_2^\gamma g_q^\rho$.

Hence the extractor can output (β, γ, ρ) such that $\tilde{C}_1 = C_1^\gamma f^\beta \mathrm{pk}^\rho$ and $\tilde{C}_2 = C_2^\gamma g_q^\rho$. ■

Theorem 3.6. The protocol **ZKPoKAff** is an honest-verifier statistically zero-knowledge argument of knowledge for relation $\mathcal{R}_{\mathrm{Aff}}$ in the generic group model.

Proof. The simulator **Sim** randomly picks a challenge $c' \in [0, q-1]$ and a prime $\ell' \in \mathrm{Prime}(\lambda)$. It picks randomly $u'_\beta \in \mathbb{Z}_q, d'_\rho, d'_\gamma \in [0, B-1]$ and $e'_\rho, e'_\gamma \in [0, q\ell'-1]$.

It computes

$$D'_1 = \mathrm{pk}^{d'_\rho}, \quad D'_2 = g_q^{d'_\rho}, \quad E'_1 = C_2^{d'_\gamma}, \quad E'_2 = C_2^{d'_\gamma}$$
$$S'_1 = {D'_1}^{q\ell'}\mathrm{pk}^{e'_\rho}f^{u'_\beta}C_1^{e'_\gamma}\tilde{C}_1^{-c'}, \quad S'_2 = {D'_2}^{q\ell'}g_q^{e'_\rho}C_2^{e'_\gamma}\tilde{C}_2^{-c'}$$

We argue that the simulated transcript $(S'_1, S'_2, c', u'_\beta, D'_1, D'_2, e'_\rho, e'_\gamma, \ell')$ is indistinguishable from a real transcript $(S_1, S_2, c, u_\beta, D_1, D_2, e_\rho, e_\gamma, \ell)$ between a prover and a verifier. **Sim** chooses (ℓ', c') identically to the honest verifier. Both u_β and u'_β are uniformly distributed in $\mathbb{Z}_q$. (S'_1, S'_2) is uniquely defined by the other values such that the verification holds. Next, we compare the simulated transcript $(D'_1, D'_2, E'_1, E'_2, e'_\rho, e'_\gamma)$ and the real transcript $(D_1, D_2, E_1, E_2, e_\rho, e_\gamma)$.

Similar to the proof of Theorem 4, we know that e_ρ or e_γ has a statistical distance at most $q\ell/(2B+1)$ from uniform over $[0, q\ell-1]$. This bounds the distance between the real e_ρ (resp. e_γ) and the simulated e'_ρ (resp. e'_γ), which is uniform over $[0, q\ell-1]$.

By similar argument in Theorem 2, the statistical distances of $(g_q^{d_\rho}, g_q^{d'_\rho})$,$(\mathrm{pk}^{d_\rho}, \mathrm{pk}^{d'_\rho})$ and $(C_2^{d_\gamma}, C_2^{d'_\gamma})$ are all at most $\mathrm{dist}_1 = \frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G^q|}{2B+3-3q\cdot 2^\lambda} + \frac{|G^q|}{B}$; the statistical distance of $(C_1^{d_\gamma}, C_1^{d'_\gamma})$ is at most $\mathrm{dist}_2 = \frac{2^{\lambda+1}q}{B} + \frac{2^\lambda q|G|}{2B+3-3q\cdot 2^\lambda} + \frac{|G|}{B}$. We have $\mathrm{dist}_1 \leq \frac{1}{2^{\epsilon_d+3}}$ if $B \geq 2^{\epsilon_d+2}(2^\lambda q+2)|G^q| + 2^{\epsilon_d+\lambda+4}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d and $\mathrm{dist}_2 \leq \frac{1}{2^{\epsilon_d+3}}$ if $B \geq 2^{\epsilon_d+2}(2^\lambda q+2)|G| + 2^{\epsilon_d+\lambda+4}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$ for some distance parameter ϵ_d. By Fact 3, the distance between the joint distribution $X_\rho = (\mathrm{pk}^{d_\rho}, g_q^{d_\rho})$ and the simulated distribution $Y_\rho = (\mathrm{pk}^{d'_\rho}, g_q^{d'_\rho})$ is at most $\frac{1}{2^{\epsilon_d+2}}$; the distance between the joint distribution $X_\gamma = (C_1^{d_\gamma}, C_2^{d_\gamma})$ and the simulated distribution $Y_\gamma = (C_1^{d'_\gamma}, C_2^{d'_\gamma})$ is at most $\frac{1}{2^{\epsilon_d+2}}$. Moreover, we can obtain that the probability $Pr(d_\rho = z|e_\rho)$ or $Pr(d_\gamma = z|e_\gamma)$ increases by at most $\frac{1}{|Y|} - \frac{q\ell}{2B+1}$. The same applies to the variable $(\mathrm{pk}^{d_\rho}, g_q^{d_\rho})|e_\rho$ (resp. $(C_1^{d_\gamma}, C_2^{d_\gamma})|e_\gamma$).

We can compare the joint distributions $X'_\rho = (\mathrm{pk}^{d_\rho}, g_q^{d_\rho}, e_\rho)$ with the simulated distribution $Y'_\rho = (\mathrm{pk}^{d'_\rho}, g_q^{d'_\rho}, e'_\rho)$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{q\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{2^{\epsilon_d+2}} + \frac{q\ell}{2B+1} + \epsilon_1 q\ell = \frac{1}{2^{\epsilon_d+2}} + \frac{q^2\ell^2}{2B+3-3q\ell} + \frac{q\ell(1-q\ell)}{2B+1} < \frac{1}{2^{\epsilon_d+2}} + \frac{q\ell}{2B-3q\ell+3} \leq \frac{1}{2^{\epsilon_d+1}}$, where the last equality holds if $B \geq 2^{\epsilon_d+\lambda+1}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. Similarly, we have the distance between the joint distributions $X'_\gamma = (C_1^{d_\gamma}, C_2^{d_\gamma}, e_\gamma)$ and $Y'_\gamma = (C_1^{d'_\gamma}, C_2^{d'_\gamma}, e'_\gamma)$ at most $\frac{1}{2^{\epsilon_d+1}}$ if $B \geq 2^{\epsilon_d+\lambda+1}q + 3q\cdot 2^{\lambda-1} - \frac{3}{2}$. Using Fact 2, we have the statistical distance between $X = (\mathrm{pk}^{d_\rho}, g_q^{d_\rho}, C_1^{d_\gamma}, C_2^{d_\gamma}, e_\gamma, e_\gamma)$ and $Y = (\mathrm{pk}^{d'_\rho}, g_q{}^{d'_\rho}, C_1^{d'_\gamma}, C_2^{d'_\gamma}, e'_\rho, e'_\gamma)$ at most $\frac{1}{2^{\epsilon_d}}$. Combining the above requirements on B, we have a simplified requirement that $B \geq 2^{\epsilon_d+\lambda+3}q^2\tilde{s}$. ■

Remarks. The analyses for argument of knowledge for ZKPoKEnc and ZKPoKAff are similar but they achieve contradiction in Case 2 in different ways. The former utilizes the fact that in its first verification equation of the last step $D_1^{q\ell}\mathrm{pk}^{e_\rho}$ cancels f with the assumption $\mathrm{pk} \in G^q$. The latter cannot follow the same method since there is one C_1 lying in $G \setminus F$ involved, in which case we lead to contraction through analyzing the exponent of f parts in the verification equation.

3.3.1. *ZK Proofs Related to Affine Transformation.*

In our proposed threshold ECDSA scheme, we also use some ZK proofs for relations closely related to the affine transformation of a CL ciphertext. Details of these ZK proofs are given in A.

3.4. Comparison with ZK Proofs in PKC-2021

Yuen *et al.* [5] proposed two compact ZK proofs for **ZKPoKRepS** and **ZKPoKEnc**. In the previous section, we further improve these two protocols, which can be directly plugged into the two threshold schemes proposed in [5]. We also give a ZK Proof for **ZKPoKAff**. We now show the improvement in terms of bandwidth and running time.

Parameter setting. For computing class group size, according to [17], each reduced class group represented by (a, b, Δ) satisfies that $-a < b \leq a$ and $a < \sqrt{|\Delta|/3}$. Let $||\Delta||$ denote the bit length of Δ. Then, a and b can be denoted by a $\lceil\frac{||\Delta||-1}{2}\rceil$-bit string and a $\lceil\frac{||\Delta||-1}{2}\rceil + 1$-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 $2\times\lceil\frac{||\Delta||-1}{2}\rceil+1$ to represent the bit size of one class group element (1827 bits per one class group element). According to [3], we require $||\Delta|| = 1827$ in 128-bit security level.

Result. We implement our ZK proofs in Rust language using a MacBook Pro laptop with 16G RAM and Intel Core i5. We use the Class[4] library in Rust. The running time and bandwidth are shown in Tables 1 and 2. The proof generation is much faster than the counterparts in [5] for ZKPoKRepS and ZKPoKEnc. The verification time does not vary much. The ZKPoKAff is relatively costly in proving but efficient in verification.

4. APPLICATION TO UC NON-INTERACTIVE, PROACTIVE THRESHOLD ECDSA

Canetti *et al.* [13] proposed a UC non-interactive, proactive threshold ECDSA which introduced a global random oracle and an

[4] https://github.com/ZenGo-X/class

Table 1. Comparison on proof size in **bits** where $\lambda = 128, ||q|| = 256, ||\Delta|| = 1827$.

Protocol	[5]	This work	Change
ZKPoKRepS (n=1)	5865	4038	⇑ 31.2%
ZKPoKEnc	11 062	7948	⇑ 33.0%
ZKPoKAff	×	11 986	-

enhanced ECDSA assumption. Their implementation involves ZK proofs in order to validate the information among the interaction. In this section, we generalize the use of ZK proofs in [13] and proposed the improvement with our newly implemented ZK proofs. Moreover, we demonstrate the UC property remained in our modification and show the performance analysis between the implementation from [13] and ours.

4.1. ZK Proofs in Threshold ECDSA

We generalize the ZK proofs from [13] at a high level as follows. We transform them into CL instantiations in A and name them ZKPoKEnc (the same with Algorithm 2), ZKPoKLog, ZKPoKAff-g and ZKPoKAff-p, which will be used to construct CL-based bandwidth-efficient threshold ECDSA scheme.

$$\mathcal{R}_{\text{enc}} = \{(\text{pk} \in G^q, K \in \mathbb{C}); (k, \rho \in \mathbb{Z}) : K = \text{Enc}_{pk}(k; \rho)\}$$

$$\mathcal{R}_{\text{log}} = \{(\text{pk} \in G^q, \hat{P}, \hat{X} \in \hat{G}, C \in \mathbb{C}); (x, \rho \in \mathbb{Z}) : \hat{X} = \hat{P}^x, C = \text{Enc}_{pk}(x; \rho)\}$$

$$\mathcal{R}_{\text{aff-g}} = \{(Y, C, D \in \mathbb{C}, \text{pk}_1, \text{pk}_2 \in G^q, \hat{X} \in \hat{G}); (x, y, \rho_y, \rho \in \mathbb{Z}) : \hat{X} = \hat{P}^x, Y = \text{Enc}_{\text{pk}_1}(y; \rho_y), D = \text{EvalAdd}(\text{EvalScal}(C, x), \text{Enc}_{\text{pk}_2}(y; \rho)\}$$

$$\mathcal{R}_{\text{aff-p}} = \{(X, Y, C, D \in \mathbb{C}, \text{pk}_1, \text{pk}_2 \in G^q); (x, y, \rho_x, \rho_y, \rho \in \mathbb{Z}) : X = \text{Enc}_{\text{pk}_1}(x; \rho_x), Y = \text{Enc}_{\text{pk}_1}(y; \rho_y), D = \text{EvalAdd}(\text{EvalScal}(C, x), \text{Enc}_{\text{pk}_2}(y; \rho)\}$$

4.2. Construct CL-based Bandwidth-efficient Threshold ECDSA

We present our CL-based bandwidth-efficient threshold ECDSA, including key generation, key refreshment, pre-signing and online signing, respectively, in Tables 3, 4, 5 and 6, where communication through point-to-point channel and synchronized broadcast channel are denoted by $\rightarrow$ and $\Rightarrow$, respectively. We summarize the major changes from Canetti's Paillier-based threshold ECDSA to our CL-based threshold ECDSA as follows.

- For KeyGen and KeyRefresh, we replace the two proofs for $\mathcal{R}_{\text{mod}}$ (denoted by ZKPoKMod) and one proof $\mathcal{R}_{\text{prm}}$ (denoted by ZKPoKPrm), with one single ZKPoKRepS ($n = 1$) for each party.
- For Paillier-related algorithms including key generation Enc, Dec, EvalAdd and EvalScal are updated with CL encryption alternatives; all ZK proofs are transformed from Paillier version to corresponding CL version, as the blue parts shown in Tables 3, 4 and 5.
- Further, we optimize the complexity of generating and verifying ZK proofs, We emphasize that, the operations in the orange box in Key Refresh is required $(n-1)^2$ times and the orange box in Pre-signing is required $(n-1)$ times for each party. Instead, in our CL setting, they are, respectively, required by only $(n-1)$ and 1 times.

4.3. Security Claim

Recap security proof in [13]. We recall first the security analysis in the Paillier-version non-interactive threshold ECDSA from [13] following the UC framework [18]. In traditional non-UC (standalone) security framework, it allows the existence of rewinding technique to extract adversary's secrets. On the contrary, UC framework introduces one more role called *environment* and the analyzed protocol is called safe if the *environment* cannot tell the differences between an ideal execution and a real execution. However, this augmented UC framework brings the technical obstacle of disabling the rewinding technique. But the security proof in [13] bypasses this obstacle by defining a *generic* ideal threshold signature functionality $\mathcal{F}_{\text{tsig}}$ (c.f. fig. 14 of [13]), instead of an ECDSA functionality. In this way, they well capture the required proactive security and successfully reintroduce the rewinding technique which greatly simplifies the security analysis. They also define another *global random oracle* functionality $\mathcal{H}$ which is accessible to both real and ideal systems. Moreover, they formalize an *enhanced unforgeability* of ECDSA which is used to prove the non-interactive threshold ECDSA protocol UC-realizes the functionality $\mathcal{F}_{\text{tsig}}$. We note that this *enhanced unforgeability* unconditionally holds in generic group model (c.f. sec 1.2.5, [13]). Equipped with the above arsenals, they proved by reduction that if their non-interactive threshold ECDSA protocol cannot UC-realize functionality $\mathcal{F}_{\text{tsig}}$, there exists a PPT distinguisher $\mathcal{R}_1$ who can break Paillier's semantic security or a PPT forger $\mathcal{R}_2$ who can win the enhanced ECDSA experiment (c.f. E.1, [13]).

Security of our proposed scheme. The UC security analysis in [13] is transferrable to our CL setting except for some changes during simulation. We list below the updates of the five simulators, two UC-simulators $\mathcal{R}_{\{1,2\}}$ and three non-UC simulators $\mathcal{S}^{\{1,2,3\}}$, and omit the full proof here. Then, we demonstrate that our modifications toward [13] do not affect the UC security. Note that in the following descriptions we omit some indices for simplicity.

- **Cancel out ring pederson parameters and its ZKs.** Across the five simulators, we unifiably cancel out every (s, t) parameters and its ZK simulator $\mathcal{S}^{\text{prm}}$ since ring pederson commitment is not required in the CL setting.
- **Convert encryption from Paillier to CL** First, the $\mathcal{R}_1$ is renamed to CL distinguisher. Accordingly, the CL distinguisher is parameterized with CL public keys and ciphertexts instead of the Paillier ones. Second, all the encryption/decryption keys, encryption/decryption algorithms, ciphertexts across the three non-UC simulators $\mathcal{S}^{\{1,2,3\}}$ are converted from Paillier setting to CL setting. For example, each (p, q) is transformed to sk (CL secret key).
- **Shift ZK-simulators from Paillier to CL**. Across the five simulators, first, redefine the proofs π^{enc}, π^{log} and π^{aff} in CL setting instead of Paillier setting, equivalently for their ZK simulators $\mathcal{S}^{\text{enc}}$, $\mathcal{S}^{\text{log}}$ and $\mathcal{S}^{\text{aff}}$; second, since we have already replaced all the Paillier secret key (p, q) to CL secret key sk, the ZK proof π^{mod} and ZK simulator $\mathcal{S}^{\text{mod}}$ should be updated to π^{RepS} and $\mathcal{S}^{\text{RepS}}$, respectively.

We emphasize here that our above updates in security analysis has no affections violating the requirements in analysis of [13], as

Table 2. Comparison on the running time for both prover and verifier (128-bit security level).

	Prover			Verifier		
	[5]	This work	change	[5]	This work	change
ZKPoKRepS (n=1)	488.7 ms	412.5 ms	⇑ 15.6%	168.6 ms	164.2 ms	⇑ 2.6%
ZKPoKEnc	969.2 ms	684.5 ms	⇑ 29.4%	333.2 ms	324.6 ms	⇑ 2.6%
ZKPoKAff	×	1715.2 ms	-	×	441.9 ms	-

Table 3. Key Generation

	KeyGen (param)	
$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{CL.KeyGen}(1^\lambda, \mathsf{crs})$ $\pi_i^{\mathsf{RepS}} := \mathsf{ZKPoKRepS}(\mathsf{pk}_i; \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i})$ $x_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ and $Q_i := x_i P$ $\mathsf{srid}_i \leftarrow \{0,1\}^\kappa; \tau_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$and $A_i := \tau_i P$ $u_i \leftarrow \{0,1\}^\kappa; V_i = \mathcal{H}(\mathsf{srid}_i, Q_i, A_i, u_i)$	$\xrightarrow{\mathsf{pk}_i, \pi_i^{\mathsf{Reps}}, V_i}$	Abort if π_i^{Reps} fails.
$\mathcal{P}_i$	Round 2 $\xrightarrow{\mathsf{srid}_i, Q_i, A_i, u_i}$	All players $\{\mathcal{P}_j\}_{j\neq i}$ Abort if $\mathcal{H}(\mathsf{srid}_i, Q_i, A_i, u_i) \neq V_i$.
$\mathcal{P}_i$	Round 3	All players $\{\mathcal{P}_j\}_{j\neq i}$
Perform (t-n)-VSS share of x_i: $p_i(X) = u_i + \sum_{k=1}^t a_{i,k} X^k \bmod q$ Denote $\{\sigma_{i,j} := p_i(j)\}_{j\in[0,n]}$ and $\{V_{i,k} := a_{i,k} P\}_{k\in[t]}$ $c_i := \mathcal{H}(V_i, \mathsf{srid})$ where $\mathsf{srid} = \oplus_j \mathsf{srid}_j$ $z_i := \tau_i + c_i x_i \bmod q$	$\xrightarrow{\sigma_{i,j}}$ $\xrightarrow{\{V_{i,k}\}_{k\in[t]}, z_i}$	Abort if $\sigma_{i,j} \cdot P \neq \sum_{k=0}^t (j^k \cdot V_{i,j})$ or $\mathcal{H}(\mathsf{srid}_i, X_i, \tilde{A}_i, u_i) \neq V_i$ where $\tilde{A}_i = z_i P - \mathcal{H}(i, \mathsf{srid}) \cdot Q_i$.
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\{\sigma_{k,i}\}_k$ are additive shares of $x_i := \sum_{k\in[n]} p_k(i)$ where $\{x_i\}_{i\in[n]}$ are (t, n) Shamir shares of x. Output $X = \prod_j X_j$		

Table 4. Key Refresh and Auxiliary Info

	Key Refresh (param)	
$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{CL.KeyGen}(1^\lambda)$ $x_{i,1}, ..., x_{i,i-1}, x_{i,i+1}, ...x_{i,n} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ s.t. $\sum_{j\neq i} x_{i,j} = 0$ $\mathbf{Y}_i = \{Q_{i,j}\}_{j\in[n], j\neq i}$, where $Q_{i,j} := x_{i,j} \cdot P$ $u_i \leftarrow \{0,1\}^\kappa$ and $Vi \leftarrow \mathcal{H}(sid; i; \mathbf{Y}_i; u_i)$ $\pi_i^{\mathsf{RepS}} := \mathsf{ZKPoKRepS}(\mathsf{pk}_i; \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i})$	$\xrightarrow{\mathsf{pk}_i, V_i, \pi_i^{\mathsf{RepS}}}$	Abort if π_i^{RepS} fails.
$\mathcal{P}_i$	Round 2	All players $\{\mathcal{P}_j\}_{j\neq i}$
$C_{i,j,j\neq i} := \mathsf{Enc}(\mathsf{pk}_j, x_{i,j}; \rho_j)$, where $\rho_j \xleftarrow{\$} [0, S]$	$\xrightarrow{\{C_{i,j}\}_{j\in[n],j\neq i}, \mathbf{Y}_i, u_i}$	Abort if $\sum_{k\in[n],k\neq i} Q_{i,k} \neq 0_{\mathbb{G}}$ or $V_i \neq \mathcal{H}(sid; i; \mathbf{Y}_i; u_i)$
$\pi_{i,j,j\neq i}^{\mathsf{log}} := \mathsf{ZKPoKLog}((x_{i,j}, \rho_j) : ((\mathsf{pk}_j, g, Q_{i,j}, C_{i,j}); (v_{i,j}, \rho_j)) \in \mathcal{R}_{\mathsf{log}})$	$\xrightarrow{\{\pi_{i,j}^{\mathsf{log}}\}_{j\in[n],j\neq i}}$	Abort if $\pi_{i,j}^{\mathsf{log}}$ fails
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}_{j\neq i}$
$u_i^{\mathsf{new}} := u_i + \sum_{j\in[n],j\neq i} \mathsf{Dec}(\mathsf{sk}_i, C_{i,j}) \mod q$ $Q_i^{\mathsf{new}} := Q_i + \sum_{j\in[n],j\neq i} Q_{j,i}$ Erase previously computed pre-signatures and all Key Refresh data except $u_i^{\mathsf{new}}, Q, \mathsf{pk}_i, \mathsf{sk}_i$		

Table 5. Modifications to the Pre-Signing protocol. Note that $S = \tilde{s} \cdot 2^{\epsilon_d}$.

Pre-Signing (param)		
$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$k_i, \gamma_i \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$; $\rho_i, \nu_i \xleftarrow{\$} [0, S]$ $K_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, k_i; \rho_i)$; $G_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \gamma_i; \nu_i)$ $\pi_i^{\mathsf{Enc}} := \mathsf{ZKPoKEnc}((k_i, \rho_i), ((\mathsf{pk}_i, K_i); (k_i, \rho_i)) \in \mathcal{R}_{\mathsf{Enc}})$	$\xrightarrow{K_i, G_i, \pi_i^{\mathsf{Enc}}}$	Abort if the π_i^{Enc} fails.
$\mathcal{P}_i$	Round 2	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\Gamma_i = g^{\gamma_i}$ $r_{i,j}, s_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j} \xleftarrow{\$} [0, S]$; $\beta_{i,j}, \hat{\beta}_{i,j} \xleftarrow{\$} \mathbb{Z}/q\mathbb{Z}$ $D_{j,i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(\gamma_i, K_j), \mathsf{Enc}(\mathsf{pk}_j, -\beta_{i,j}; s_{i,j}))$ $\hat{D}_{j,i} \leftarrow \mathsf{EvalAdd}(\mathsf{EvalScal}(x_i, K_j), \mathsf{Enc}(\mathsf{pk}_j, -\hat{\beta}_{i,j}; \hat{s}_{i,j}))$ $F_{j,i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \beta_{i,j}; r_{i,j})$; $\hat{F}_{j,i} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \hat{\beta}_{i,j}; \hat{r}_{i,j})$ $\pi_i^{\mathsf{log}} := \mathsf{ZKPoKLog}((\gamma_i, \nu_i) : ((\mathsf{pk}_i, g, \Gamma_i, G_i); (\gamma_i, \nu_i)) \in \mathcal{R}_{\mathsf{log}})$ $\pi_{j,i}^{\mathsf{Aff\text{-}p}} := \mathsf{ZKPoKAff\text{-}p}((\gamma_i, \beta_{i,j}, \nu_i, r_{i,j}, s_{i,j}) :$ $((G_i, F_{j,i}, K_j, D_{j,i}, \mathsf{pk}_i, \mathsf{pk}_j); (\gamma_i, \beta_{i,j}, \nu_i, r_{i,j}, s_{i,j})) \in \mathcal{R}_{\mathsf{Aff\text{-}p}})$ $\pi_{j,i}^{\mathsf{Aff\text{-}g}} := \mathsf{ZKPoKAff\text{-}g}((x_i, \hat{\beta}_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j}) :$ $((G_i, F_{j,i}, K_j, D_{j,i}, \mathsf{pk}_i, \mathsf{pk}_j, g^{x_i}); (x_i, \hat{\beta}_{i,j}, \hat{r}_{i,j}, \hat{s}_{i,j})) \in \mathcal{R}_{\mathsf{Aff\text{-}g}})$	$\xrightarrow{\pi_i^{\mathsf{log}}, \Gamma_i}$ $\xrightarrow{D_{j,i}, \hat{D}_{j,i}, F_{j,i}, \hat{F}_{j,i}, \pi_{j,i}^{\mathsf{Aff\text{-}g}}, \pi_{j,i}^{\mathsf{Aff\text{-}p}}}$	Abort if any proof fails.
$\mathcal{P}_i$	Round 3	All players $\{\mathcal{P}_j\}_{j\neq i}$
$\Gamma := \prod_j \Gamma_j$; $\Delta_i = \Gamma^{k_i}$ $\alpha_{i,j} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, D_{i,j})$; $\hat{\alpha}_{i,j} \leftarrow \mathsf{Dec}(\mathsf{sk}_i, \hat{D}_{i,j})$ $\delta_i := \gamma_i k_i + \sum_{j\neq i}(\alpha_{i,j} + \beta_{i,j}) \mod q$ $\chi_i := x_i k_i + \sum_{j\neq i}(\hat{\alpha}_{i,j} + \hat{\beta}_{i,j}) \mod q$ $\pi_i^{\mathsf{log}'} := \mathsf{ZKPoKLog}((k_i, \rho_i) : ((\mathsf{pk}_i, \Gamma, \Delta_i, K_i); (k_i, \rho_i))) \in \mathcal{R}_{\mathsf{log}})$ Erase all items in memory except for the stored state.	$\xrightarrow{\delta_i, \Delta_i, \pi_i^{\mathsf{log}'}}$	Abort if $\pi_i^{\mathsf{log}'}$ fails.
$\mathcal{P}_i$	Output	All players $\{\mathcal{P}_j\}$
$\delta := \sum_j \Delta_j$ Verify $g^{\delta} = \prod_j \Delta_j$, abort otherwise Set $R := \Gamma^{\delta^{-1}}$ Output (R, k_i, χ_i). Erase all items except the store state.		

Table 6. Signing protocol.

Signing (param)		
$\mathcal{P}_i$	Round 1	All players $\{\mathcal{P}_j\}_{j\neq i}$
$r = R\|_{\text{x-axis}}$ $\sigma_i = km + r\chi$ Erase (R, k, χ).	$\xrightarrow{\sigma_i}$	$\sigma = \sum_j \sigma_j$ Verify (r, σ) is a valid signature, return (m, r, σ) if valid, abort otherwise.

Table 7. Message Sizes in bits under soundness error of 2^{-80}.

Message Type	[13]	This work
Public key	9216	1827
Ciphertext	6144	3654
ZKPoKRepS (n=1)	-	4038
ZKPoKMod	494 752	-
ZKPoKPrm	491 520	-
ZKPoKEnc	19 971	7948
ZKPoKLog	20 227	8204
ZKPoKAff-g	42 244	20 190
ZKPoKAff-p	51 204	29 197

shown in (i) and (ii); and that some assumptions should be tuned in claims 5.5 and 5.6 in [13] as shown in (iii).

(i) The construction of stand-alone (non-UC) simulators $\mathcal{S}^{\{1,2,3\}}$ treat each ZK as a module without specifying its inner setting as in section 5.5 of [13]. Thus, our updates in ZK proofs, simulators and also encryption schemes do not have material influence during the simulation.

(ii) The transition from π^{mod} to π^{RepS} keeps the *extractability requirement* in the UC simulators $\mathcal{R}^{\{1,2\}}$; further, generic group model is required in the extractability analysis of π^{RepS} but still the enhanced ECDSA assumption unconditionally holds in generic group model as aforementioned;

(iii) In the output phase of simulator $\mathcal{S}^1$, it should fulfill an environment secrets extraction and the secrets are the encrypted plaintexts. This kind of extractability is promised by the special soundness of their encryption well-formedness

Table 8. Bandwidth Analysis in bits under n-party setting in 128-bit security level.

Protocol	[13]	This work
Key generation	$256 \times n^2 + (1497\,920 + 256t) \times n$	$256 \times n^2 + (7401 + 256t) \times n$
Key refreshment	$20\,227 \times n^3 - 34\,054 \times n^2 + 1009\,955 \times n$	$12\,114 \times n^2 - 5865 \times n$
Pre-signing	$118\,024 \times n^2 - 44\,543 \times n$	$64\,003 \times n^2 - 31\,571 \times n$

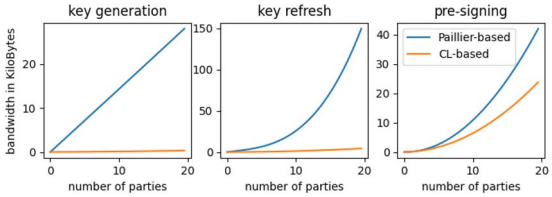

Figure 1. Bandwidth in 128-bit security level where t is set $n - 1$.

ZK proof. More specifically, it relies on the *strong RSA assumption*. Switching to our setting, the special soundness of our ZKPoKEnc is assured by three assumptions: adaptive root subgroup assumption, Corollary 2.1 and Corollary 2.2. Hence, without changing much of the proof sketch of section 5.3.1 of [13], we simply adapt these three assumptions to replace the original *strong RSA assumption*, more specifically, for Lemma 5.4 and Claim 5.5 in [13]. Also, it is trivial to replace the semantic security of Paillier encryption with the semantic security with CL encryption, which produces the noticeable winning probability of a CL distinguisher $\mathcal{R}^1$.

Hence, we conclude the security claim of our new threshold ECDSA scheme in Theorem 4.1, where we also follow the definition of proactive, ideal threshold signature functionality $\mathcal{F}_{\text{tsig}}$ which follows the same definition of fig. 14 of [13].

Theorem 4.1. Assuming semantic security of the CL cryptosystem, adaptive root subgroup assumption, Corollary 2.1 and Corollary 2.2 and enhanced existential unforgeability of ECDSA, it holds that the non-interactive threshold ECDSA modified in this paper UC-realizes the functionality $\mathcal{F}_{\text{tsig}}$, in the presence of the global random oracle functionality H.

4.4. Bandwidth Analysis

In this subsection, we analyze the theoretical complexity of our ZK proofs and modified ECDSA. We compare the communication bandwidth of our protocol with the one from Canetti *et al.* [13] corresponding to the level of security 128, and under soundness error 2^{-80}.

Zero-knowledge Proofs. We compare the communication bandwidth of the ZK proofs proposed in [13] and this paper. We observe that, for the public key and ciphertext, the communication bandwidth is reduced by around 80.2% and 40.5%, respectively. For the ZK proofs, the communication bandwidth is improved in the range of 43.0–60.2%. A detailed communication bandwidth analysis is shown in Table 7.

Components in threshold ECDSA. We compare the communication bandwidth of the key generation, key refreshment and pre-signing components in threshold ECDSA. The results are shown in Table 8 and Fig. 1. As the number of parties increased, the communication bandwidth of the threshold ECDSA proposed by Canetti *et al.* [13] grows much higher than that of ours, thus the modification from ours saves a large amount of communication bandwidth. Note that in the picture for key generation, the two curves seem linear and like a mismatch with Table 8. This is due to that when n is not large enough, the trend will be dominated by the coefficient of n instead of the coefficient of n^2.

Table 9. Running time.

(t,n)	Protocol	KeyGen	KeyRefresh	PreSign	OnlineSign
(1,3)	[13]	36.5 s	39.4 s	6.2 s	1.2 ms
	our scheme	3.7 s	13.9 s	80.5 s	1.2 ms
(2,4)	[13]	54.1 s	62.2 s	11.9 s	1.6 ms
	our scheme	5.9 s	26.9 s	155.2 s	1.6 ms
(2,5)	[13]	72.1 s	89.9 s	19.7 s	2.1 ms
	our scheme	8.4 s	42.7 s	253.8 s	2.1 ms

4.5. Implementation

We implemented both the Paillier-based threshold ECDSA [13] and our CL-based threshold ECDSA in Rust using a MacBook Pro laptop with 16G RAM and Intel Core i5. We use the Bitcoin secp256k1 curve[5] and number theory library[6] for large integer operations. Our program is yet to be parallelized and instead, we linearly execute each party's operations in each round without considering the network issue. We choose the (t,n) settings with (1,3), (2,4) and (2,5) which are the most popular settings in Bitcoin's P2SH transactions[7] .

We observe that, from Table 9, for the running time, the key generation is reduced by up to 10 factors and KeyRefresh also improved by around 50%. But the running time of pre-signing is 10+ times than the Paillier counterpart (parallel computation will reduce the running time by dividing the party number), which is the only cost of improving the bandwidth for all phases and improving the running time in key generation and key refreshment, although it is less important in a non-interactive signing scenario where we pursue the optimal online signing while can loosen the computation assumption in pre-signing. We leave further optimization of the inefficient CL-based pre-signing as our future work. For online signing, our scheme is the same with [13].

5. CONCLUSIONS

In this paper, we optimize the existing ZK proofs in CL-based threshold ECDSA in both proof size and proving time; we propose a new ZK proof for the affine transformation in CL ciphertext; finally, we extend these three to devise tailored ZK proofs to construct the bandwidth-optimal UC secure, non-interactive and proactive threshold ECDSA, which outperforms its Paillier-based counterpart in bandwidth of all phases and in running time in key generation and refreshment, at a cost of a slower pre-signing.

[5] https://github.com/rust-bitcoin/rust-secp256k1
[6] https://docs.rs/rust-gmp/0.5.0/gmp/mpz/struct.Mpz.html
[7] https://txstats.com/dashboard/db/p2sh-repartition-bytype?orgId=1

FUNDING

The first three authors are supported by Huawei Singapore Research Centre project.

DATA AVAILABILITY STATEMENTS

No new data were generated or analyzed in support of this research.

REFERENCES

1. Itakura, K. and Nakamura, K. (1983) A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, **71**, 1–8.
2. Gennaro, R. and Goldfeder, S. (2018) Fast multiparty threshold ECDSA with fast trustless setup. *Proceedings of CCS '18, Toronto, 15-19 October*, pp. 1179–1194. ACM, New York City.
3. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., and Tucker, I. (2020) Bandwidth-efficient threshold EC-DSA. *Proceedings of PKC '20, Edinburgh, UK, 4-7 May*, pp. 266–296. Springer, Berlin.
4. Gennaro, R. and Goldfeder, S. (2020) One round threshold ECDSA with identifiable abort. *Cryptology*, ePrint Archive, Report 2020/540.
5. Yuen, T. H., Cui, H., and Xie, X. (2021) Compact zero-knowledge proofs for threshold ECDSA with trustless setup. *Proceedings of PKC '21, Virtual Event, 10-13 May*, pp. 481–511. Springer, Berlin.
6. Lindell, Y. and Nof, A. (2018) Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. *Proceedings of CCS '18, Toronto, 15-19, October*, pp. 1837–1854. ACM, New York City.
7. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F. and Tucker, I. (2021) Bandwidth-efficient threshold EC-DSA revisited: online/offline extensions, identifiable aborts, proactivity and adaptive security. *Cryptology*, ePrint Archive, Report 2020/492.
8. Paillier, P. (1999) Public-key cryptosystems based on composite degree residuosity classes. *Proceedings of EUROCRYPT 99, Prague, Czech Republic, 2-6 May*, pp. 223–238. Springer, Berlin.
9. Castagnos, G. and Laguillaumie, F. (2015) Linearly homomorphic encryption from DDH. *Proceedings of CT-RSA '15, San Francisco, CA, 20-24 April*, pp. 487–505. Springer, Berlin.
10. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., and Tucker, I. (2019) Two-party ECDSA from hash proof systems and efficient instantiations. *Proceedings of CRYPTO '19, Santa Barbara, CA, 18-22 August*, pp. 191–221. Springer, Berlin.
11. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., and Peled, U. (2020) UC non-interactive, proactive, threshold ECDSA with identifiable aborts. *Proceedings of CCS '20, Virtual Event, 9-13 November*, pp. 1769–1787. ACM, New York City.
12. Deng, Y., Ma, S., Zhang, X., Wang, H., Song, X., and Xie, X. (2021) Promise sigma-protocol: How to construct efficient threshold ecdsa from encryptions based on class groups. *Proceedings of ASIACRYPT '21, Singapore, 6-10 December*, pp. 557–586. Springer, Berlin.
13. Canetti, R., Makriyannis, N. and Peled, U. (2020) UC non-interactive, proactive, threshold ECDSA. *Cryptology*, ePrint Archive, Report 2020/492.
14. Boneh, D., Bünz, B., and Fisch, B. (2019) Batching techniques for accumulators with applications to iops and stateless blockchains. *Proceedings of CRYPTO '19*, Santa Barbara, CA, pp. 561–586. Springer, Berlin.
15. Castagnos, G. and Laguillaumie, F. (2009) On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. *Proceedings of EUROCRYPT '09, Cologne, 26-30 April*, pp. 260–277. Springer, Berlin.
16. Damgård, I. and Koprowski, M. (2002) Generic lower bounds for root extraction and signature schemes in general groups. *Proceedings of EUROCRYPT '02, Amsterdam, The Netherlands, 28 April-2 May*, pp. 256–271. Springer, Berlin.
17. Hamdy, S. and Möller, B. (2000) Security of cryptosystems based on class groups of imaginary quadratic orders. *Proceedings of ASIACRYPT '00, Kyoto, Japan, 3-7 December*, pp. 234–247. Springer, Berlin.
18. Canetti, R. (2001) Universally composable security: A new paradigm for cryptographic protocols. *Proceedings of FOCS '01, Las Vegas, Nevada, USA, 8-11 October*, pp. 136–145. IEEE.

ZK Proofs compatible with UC Non-interactive Proactive Threshold ECDSA

We give our tailored ZK proofs, extended from our new ZK proofs, for the Canetti's UC non-interactive proactive's threshold ECDSA in a class group setting, which achieves the *bandwidth-optimal* counterpart without compromising its UC security, non-interacivity and proactive security. For simplicity, we omit the HVZK analysis for ZKPoKAff-p and ZKPoKAff-g. We emphasize that the sampling Bound B's of ZKPoKAff-p and ZKPoKAff-g are different from other ZK proofs. ZKPoKLog shares the same B with ZKPoKEnc. We summarize the sampling ranges for each ZK in Table A1.

Table A1. Sample ranges in different settings.

ZKP	B
ZKPoKRepS	$2^{\epsilon_d+2\lambda}nq^2\tilde{s}$
ZKPoKEnc	$2^{\epsilon_d+\lambda+2}q\tilde{s}$
ZKPoKAff	$2^{\epsilon_d+\lambda+3}q^2\tilde{s}$
ZKPoKAff-p	$2^{\epsilon_d+\lambda+4}q(5+q\tilde{s})$
ZKPoKAff-g	$2^{\epsilon_d+\lambda+2}q(5+q\tilde{s})$
ZKPoKLog	$2^{\epsilon_d+\lambda+2}q\tilde{s}$

ZK for Aff-p

$$\mathcal{R}_{\text{aff-p}} = \{(X_1, X_2, Y_1, Y_2, C_1, C_2, D_1, D_2, \text{pk}_1, \text{pk}_2 \in G^q);$$
$$(x, y \in \mathbb{Z}_q, \rho_x, \rho_y, \rho \in [0, S]) : X_1 = f^x \text{pk}_1^{\rho_x}, X_2 = g_q^{\rho_x},$$
$$Y_1 = f^y \text{pk}_1^{\rho_y}, Y_2 = g_q^{\rho_y}, D_1 = C_1^x f^y \text{pk}_2^{\rho}, D_2 = C_2^x g_q^{\rho}\}$$

ZK for Aff-g

$$\mathcal{R}_{\text{aff-g}} = \{(Y_1, Y_2, C_1, C_2, D_1, D_2, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q, \hat{X} \in \hat{G});$$
$$(x, y \in \mathbb{Z}_q, \rho_y, \rho \in [0, S]) : \hat{X} = \hat{P}^x, Y_1 = f^y \mathsf{pk}_1^{\rho_y},$$
$$Y_2 = g_q^{\rho_y}, D_1 = C_1^x f^y \mathsf{pk}_2^{\rho}, D_2 = C_2^x g_q^{\rho}\}$$

ZK for log

$$\mathcal{R}_{\log} = \{(\hat{P}, \hat{X} \in \hat{G}, C_1, C_2, \mathsf{pk} \in G^q); (m \in \mathbb{Z}_q, \rho \in [0, S]) :$$
$$\hat{X} = \hat{P}^m, C_1 = f^m \mathsf{pk}^{\rho}, C_2 = g_q^{\rho}\}$$

Algorithm 4 (ZKPoKAff-p).
Param: $\mathcal{G}_{\text{HSM}} \leftarrow \mathsf{GGen}_{\text{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+4} q(5+q\tilde{s})$ where $\epsilon_d = 80$.
Input: $X_1, X_2, Y_1, Y_2, C_1, C_2, D_1 D_2, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q$.
Witness: $\rho, \rho_x, \rho_y \in [0, S], x, y \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1. Prover chooses $s_\rho, s_x, s'_x, s'_y \xleftarrow{\$} [-B, B]$, $s_y \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = f^{s_x} \mathsf{pk}_1^{s'_x}, S_2 = g_q^{s'_x}, S_3 = f^{s_y} \mathsf{pk}_1^{s'_y}, S_4 = g_q^{s'_y}, S_5 = C_1^{s_x} f^{s_y} \mathsf{pk}_2^{s_\rho}, S_6 = C_2^{s_x} g_q^{s_\rho}$ Prover sends $(S_1, S_2, S_3, S_4, S_5, S_6)$ to the verifier.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.
3. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.
4. Prover computes: $u_y = s_y + cy \mod q, u_x = s_x + cx, u_\rho = s_\rho + c\rho, u'_x = s'_x + c\rho_x, u'_y = s'_y + c\rho_y$. Prover finds $d_x, d_\rho, d'_x, d'_y \in \mathbb{Z}$ and $e_x, e_\rho, e'_x, e'_y \in [0, q\ell - 1]$ s.t. $u_x = d_x q\ell + e_x, u_\rho = d_\rho q\ell + e_\rho, u'_x = d'_x q\ell + e'_x, u'_y = d'_y q\ell + e'_y$, Prover computes: $E_1 = \mathsf{pk}_1^{d'_x}, E_2 = g_q^{d'_x}, E_3 = \mathsf{pk}_1^{d'_y}, E_4 = g_q^{d'_y}, E_5 = \mathsf{pk}_2^{d_\rho}, E_6 = g_q^{d_\rho}, H_1 = C_1^{d_x}, H_2 = C_2^{d_x}, F = f^{d_x}$. Prover sends $(u_y, E_1, E_2, E_3, E_4, E_5, E_6, H_1, H_2, F, e_\rho, e_x, e'_x, e'_y)$ to the verifier.
5. Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:
$(E_1 F)^{q\ell} \mathsf{pk}_1^{e'_x} f^{e_x} = S_1 X_1^c, \quad E_2^{q\ell} g_q^{e'_x} = S_2 X_2^c,$
$E_3^{q\ell} \mathsf{pk}_1^{e'_y} f^{u_y} = S_3 Y_1^c, \quad E_4^{q\ell} g_q^{e'_y} = S_4 Y_2^c,$
$(E_5 H_1)^{q\ell} \mathsf{pk}_2^{e_\rho} C_1^{e_x} f^{u_y} = S_5 D_1^c,$
$(E_6 H_2)^{q\ell} g_q^{e_\rho} C_2^{e_x} = S_6 D_2^c.$

Algorithm 5 (ZKPoKAff-g).
Param: $\mathcal{G}_{\text{HSM}} \leftarrow \mathsf{GGen}_{\text{HSM},q}(1^\lambda)$; $B = 2^{\epsilon_d+\lambda+2} q(5 + q\tilde{s})$ where $\epsilon_d = 80$.
Input: $Y_1, Y_2, C_1, C_2, D_1, D_2, \mathsf{pk}_1, \mathsf{pk}_2 \in G^q, \hat{X} \in \hat{G}$.
Witness: $\rho, \rho_x, \rho_y \in [0, S], x, y \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1. Prover chooses $s_\rho, s_x, s'_y \xleftarrow{\$} [-B, B]$, $s_y \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = f^{s_y} \mathsf{pk}_1^{s'_y}, S_2 = g_q^{s'_y}, R_1 = C_1^{s_x} f^{s_y} \mathsf{pk}_2^{s_\rho}, R_2 = C_2^{s_x} g_q^{s_\rho}, \hat{S} = \hat{P}^{s_x}$ Prover sends $(S_1, S_2, R_1, R_2, \hat{S})$ to the verifier.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover. Prover aborts if $c \notin [0, q-1]$.
3. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.
4. Prover computes: $u_y = s_y + cy \mod q, u_x = s_x + cx, u_\rho = s_\rho + c\rho, u'_y = s'_y + c\rho_y$. Prover finds $d_\rho, d_x, d'_y \in \mathbb{Z}$ and $e_x, e_\rho, e'_y \in [0, q\ell - 1]$ s.t. $u_x = d_x q\ell + e_x, u_\rho = d_\rho q\ell + e_\rho, u'_y = d'_y q\ell + e'_y$, Prover computes: $H_1 = C_1^{d_x}, H_2 = C_2^{d_x}, E_1 = \mathsf{pk}_1^{d'_y}, E_2 = g_q^{d'_y}, F_1 = \mathsf{pk}_2^{d_\rho}, F_2 = g_q^{d_\rho}, \hat{D} = \hat{P}^{d_x}$. Prover sends $(u_y, H_1, H_2, E_1, E_2, F_1, F_2, \hat{D}, e_\rho, e_x, e'_y)$ to the verifier.
5. Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:
$\hat{D}^{q\ell} \hat{P}^{e_x} = \hat{S} \hat{X}^c, \quad E_2^{q\ell} g_q^{e'_y} = S_2 Y_2^c,$
$E_1^{q\ell} \mathsf{pk}_1^{e'_y} f^{u_y} = S_1 Y_1^c,$
$(H_1 F_1)^{q\ell} \mathsf{pk}_2^{e_\rho} C_1^{e_x} f^{u_y} = R_1 D_1^c,$
$(H_2 F_2)^{q\ell} g_q^{e_\rho} C_2^{e_x} = R_2 D_2^c.$

Algorithm 6 (ZKPoKLog).
Param: $\mathcal{G}_{\text{HSM}} \leftarrow \mathsf{GGen}_{\text{HSM},q}(1^\lambda)$, $B = 2^{\epsilon_d+\lambda+2} q\tilde{s}$, where $\epsilon_d = 80$.
Input: $\hat{P}, \hat{X}, C_1, C_2, \mathsf{pk} \in G^q$.
Witness: $\rho \in [0, S], m \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.

1. Prover chooses $s_\rho \xleftarrow{\$} [-B, B]$, $s_m \xleftarrow{\$} \mathbb{Z}_q$ and computes: $S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, S_2 = g_q^{s_\rho}, \hat{S} = \hat{P}^{s_m}$. Prover sends $(S_1, S_2, \hat{S})$ to the verifier. Prover aborts if $c \notin [0, q-1]$.
2. Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover.
3. Verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$ to the prover.
4. Prover computes: $u_\rho = s_\rho + c\rho, u_m = s_m + cm \mod q$. Prover finds $d_\rho \in \mathbb{Z}$ and $e_\rho \in [0, q\ell - 1]$ s.t. $u_\rho = d_\rho q\ell + e_\rho$. Prover computes: $D_1 = \mathsf{pk}^{d_\rho}, D_2 = g_q^{d_\rho}$. Prover sends (u_m, D_1, D_2, e_ρ) to the verifier.
5. Verifier accepts if $e_\rho \in [0, q\ell - 1]$ and:
$\hat{P}^{u_m} = \hat{S} \hat{X}^c, \quad D_1^{q\ell} \mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c,$
$D_2^{q\ell} g_q^{e_\rho} = S_2 C_2^c.$

Corrigendum

In this edition, we have several improper places to update (computation mistake in bandwidth, typos in proof and graph). Please refer to the Chapter 4 of the thesis **Trustless Digital Signatures in Blockchain** for an updated correct version of this work. The thesis is accessible in https://hub.hku.hk/handle/10722/336617.

We highlight the changes as follows.

1. There is a mistake when measuring the class group element in section 3.4. The size of Δ should be 2339 bits instead of 1827 bits since it follows that Δ = q^2 * Δ_K, where size of Δ_K is 1827 bits. Accordingly, results in table 1, 7, 8 and figure 1 for bandwidth analysis should be updated. For the updates, please refer to the parameter setting in section 4.3.4, table 4.1, table 4.9, table 4.10, figure 4.1 of the thesis.
2. There is misuse between x_i and u_i, X_i and Q_i in KeyGen algorithm chart (table 3). Refer to section 4.4.2 in thesis for the updated charts.
3. In the proof of Theorem 3.1, the representation of D1/2 is not complete, omitting f^\gamma and f_\delta respectively. Refer to Theorem 4.1 in thesis for the updates.