Compact Zero-Knowledge Proofs for Threshold ECDSA with Trustless Setup

Tsz Hon Yuen[1], Handong Cui[1], and Xiang Xie[2]

[1] The University of Hong Kong, Hong Kong
{thyuen, hdcui}@cs.hku.hk
[2] Shanghai Key Laboratory of Privacy-Preserving Computation,
MatrixElements Technologies
xiexiang@matrixelements.com

Abstract. Threshold ECDSA signatures provide a higher level of security to a crypto wallet since it requires more than t parties out of n parties to sign a transaction. The state-of-the-art bandwidth efficient threshold ECDSA used the additive homomorphic Castagnos and Laguillaumie (CL) encryption based on an unknown order group G, together with a number of zero-knowledge proofs in G. In this paper, we propose compact zero-knowledge proofs for threshold ECDSA to lower the communication bandwidth, as well as the computation cost. The proposed zero-knowledge proofs include the discrete-logarithm relation in G and the well-formedness of a CL ciphertext.

When applied to two-party ECDSA, we can lower the bandwidth of the key generation algorithm by 47%, and the running time for the key generation and signing algorithms are boosted by about 35% and 104% respectively. When applied to threshold ECDSA, our first scheme is more optimized for the key generation algorithm (about 70% lower bandwidth and 85% faster computation in key generation, at a cost of 20% larger bandwidth in signing), while our second scheme has an all-rounded performance improvement (about 60% lower bandwidth, 46% faster computation in key generation without additional cost in signing).

Keywords: Threshold signature · ECDSA · Zero-knowledge Proof.

1 Introduction

Threshold signature allows n parties to share the message signing ability without trusting each other, such that no coalition of $t < n$ or fewer users can generate a valid signature. Threshold ECDSA signatures become a popular research topic recently since ECDSA is adopted in Bitcoin and other cryptocurrencies. Threshold ECDSA signatures are useful for managing keys in crypto wallet. For example, two-party ECDSA [14, 4] (with $t = 1, n = 2$) is useful for smart contract building blocks such as Coinswap and Lightning Network. A threshold signature with $t = 1, n = 3$ is useful for a hot wallet of a crypto exchange: the exchange holds a private key for online transaction and a private key for paper backup, and a separate security firm holds the third key to validate transactions. In this case, losing one key from the exchange or the security firm does not compromise the hot wallet. General threshold ECDSA signatures were proposed in [15, 11, 5].

1.1 Additive Homomorphic CL Encryption in Threshold ECDSA

Using additive homomorphic encryption is one of the most popular techniques for generating efficient two-party or threshold ECDSA. Some earlier papers [14, 15, 11] used Paillier encryption. Recently, Castagnos *et al.* [4] used the additive homomorphic Castagnos and Laguillaumie (CL) encryption [7] based on an unknown order group G, which contains a subgroup F in which the discrete logarithm (DL) problem is tractable. We call the group G as the HSM group since we require that the *hard subgroup membership* assumption holds in G. It was shown in [1] that the HSM group G can be constructed from class groups of quadratic fields. The advantage of using CL encryption over Paillier encryption is that the generation of the class group is trustless, and the size of a class group element is smaller than that of a Paillier group element (for the same security level).

Zero-knowledge Proofs for CL Encryption. One of the technical difficulties for using the CL encryption for threshold ECDSA is the design of zero-knowledge (ZK) proofs in the HSM group. In particular, we need the ZK proofs related to (1) the discrete-logarithm (DL) of an unknown order group element, and (2) the well-formedness of a CL ciphertext. In [4], the authors used a ZK proof with a single bit challenge. In order to achieve soundness error of $2^{-\epsilon_s}$, the protocol has to be repeated for ϵ_s-times and hence the resulting algorithm is inefficient. In [5], the authors tackled the first DL problem by using a *lowest common multiple* (*lcm*) tricks, which reduces the repetition of the ZK proof to about $\epsilon_s/10$-times. The authors tackled the second CL ciphertext well-formedness problem based on a strong root assumption in the HSM group.

Although the ZK proof for a CL ciphertext in [5] is highly efficient, it does not allow a fast, trustless setup. The strong root assumption used in [5] assumes that when given a random group element $w \in G \setminus F$, it is difficult to output a group element u and a positive integer $e \neq 2^k$ such that $u^e = w$.[3] In their security proof, it requires that w is a random group generator, which can be obtained from a standardized group, or jointly generated by all participating parties during the interactive key generation algorithm. In the former case, all users have to trust the standardizing authority and it is not desirable for decentralized applications such as public blockchain. In the latter case, it greatly increases the round complexity and the bandwidth used for the interactive key generation algorithm.

[3] This special requirement on e is needed since computing square roots in class groups of quadratic fields is easy [4]. The assumptions used in this paper do not require such a special arrangement.

Algorithm 1: Insecure ZK Proof for the relation $\mathcal{R}$

```
Verifier sends a random $\lambda$-bit prime $\ell$.
Prover finds $q' \in \mathbb{Z}$ and $r \in [0, \ell - 1]$ s.t. $x = q'\ell + r$. Prover sends $Q = g^{q'}$ and
  $r$ to the verifier.
Verifier accepts if $r \in [0, \ell - 1]$ and $Q^{\ell} g^{r} = w$.
```

1.2 Compact Zero-Knowledge Proof with Fast Trustless Setup

In this paper, we propose compact ZK proofs for the DL relation of HSM group element, and the well-formedness of CL ciphertext with a fast trustless setup. We first consider a ZK proof for a simple DL relation $\mathcal{R}$ in an unknown order group G for some group elements $g, w \in G \setminus F$:[4]

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

The subgroup F makes the ZK proof on the relation $\mathcal{R}$ much more complicated.

First Attempt. We start by adopting the adaptive root assumption [1] in the group G with order q subgroup F. In short, the adversary first outputs a group element $w \in G \setminus F$ (which can be verified by $w^{q} \neq 1$). When given a random prime ℓ, we assume that no polynomial time adversary can output a group element u such that $u^{\ell} = w$ with non-negligible probability. Given such an assumption, we can construct a simple ZK proof for $\mathcal{R}$ based on [1] in Algorithm 1.

However, this trivial construction is not secure since there exists a known order subgroup $F \subset G$. Suppose that the prover knows x and y such that $w = g^{x} f^{y}$ for some $f \in F$. The prover can compute $Q' = g^{q'} f^{\frac{y}{\ell}}$ since the order of f is known. It can pass the verification since:

$$Q'^{\ell} g^{r} = (g^{q'} f^{\frac{y}{\ell}})^{\ell} g^{r} = g^{x} f^{y} = w.$$

Our Solution. We propose the use of an extra round of challenge to eliminate the elements of order q in w. This extra round simply uses q instead of using the prime number ℓ. We give a simplified ZK proof for the relation $\mathcal{R}$ in Algorithm 2. (It is the simplified version of Algorithm 4 by setting $n = 1$.)

Note that our protocol only runs for one time only for a soundness error of $2^{-\epsilon_s} \approx 2^{\log \lambda - \lambda}$, as compared to ϵ_s-times for [4] and $\epsilon_s/10$-times for [5] for a soundness error of $2^{-\epsilon_s}$. Based on our efficient ZK proof for DL relation of a class group element, we can later formulate an efficient ZK proof for the well-formedness of a CL ciphertext. The major technical difficulty of this paper lies in the security proof and the security model.

[4] Since it is easy to compute $\log_g w$ if $g \in F$, it is impossible to construct a ZK proof for $\mathcal{R}$ if $g \in F$. Hence, we restrict that $g \in G \setminus F$.

Algorithm 2: Simplified ZK Proof for the relation $\mathcal{R}$

```
Param: A security parameter $B$.
Prover chooses $k \xleftarrow{\$} [-B, B]$ and sends $R = g^k$ to the verifier.
Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover.
Prover computes $s = k + cx$. Prover finds $d \in \mathbb{Z}$, $e \in [0, q-1]$ s.t. $s = dq + e$
  and sends $D = g^d$ and $e$ to the verifier.
If $e \in [0, q-1]$ and $D^q g^e = Rw^c$, verifier sends a random $\lambda$-bit prime $\ell$.
Prover finds $q' \in \mathbb{Z}$ and $r \in [0, \ell-1]$ s.t. $s = q'\ell + r$. Prover sends $Q = g^{q'}$ and
  $r$ to the verifier.
Verifier accepts if $r \in [0, \ell-1]$ and $Q^\ell g^r = Rw^c$.
```

1.3 Our Contribution

Our contribution is twofold: (1) In theoretical aspect, we give compact ZK proofs for the well-formedness of a CL ciphertext and for the DL relation in the HSM group with a fast and trustless setup. (2) In practical aspect, we improve the performance of two-party ECDSA and threshold ECDSA with trustless setup by using our ZK proofs.

We observe that by using the generic group model, we can build a more compact ZK proof for two-party/threshold ECDSA. Since ECDSA is known to be secure in the generic group model [2], the security of two-party/threshold ECDSA also indirectly relies on the generic group model. Using our compact ZK proof for two-party/threshold ECDSA still relies on the generic group model.

ZK Proofs and the Generic Group Model. We propose the *first* generic group model for the HSM group (including the class group of imaginary quadratic group order), by defining group operations with the main group G, as well as the subgroup F. Equipped with the new generic group model, we are able to analyse the security of the hard subgroup membership assumption and the adaptive root subgroup assumption in the generic group model. The technical difficulty for the generic group mode is how to maintain the correctness of group operations among elements in G and F, where the discrete logarithm of elements in F is known. Denote $G = G^q \times F$ where G^q is a subgroup of G. For all $g \in G$, we represent g by an element in G^q and an element in F. We handle the group operations in G^q and F separately in order to ensure the correctness of DL computation in F.

Afterwards, we propose some building blocks, such as the proof of knowledge of an exponent for a group element in G, and a zero-knowledge proof of knowledge for a group element representation in G (a generalization of the DL relation), and then a zero-knowledge proof of knowledge for the well-formedness of a CL ciphertext. As shown in Figure 1, our ZK proof for DL in the HSM group is around 97% shorter than CCL+19 [4] and around 74% shorter than CCL+20 ([5], §5.1) with the same level of soundness error and statistical distance of 2^{-80}.

As compared with ZK proofs in [5], their strong root assumption is similar to the strong RSA assumption, while our adaptive root assumption is more similar

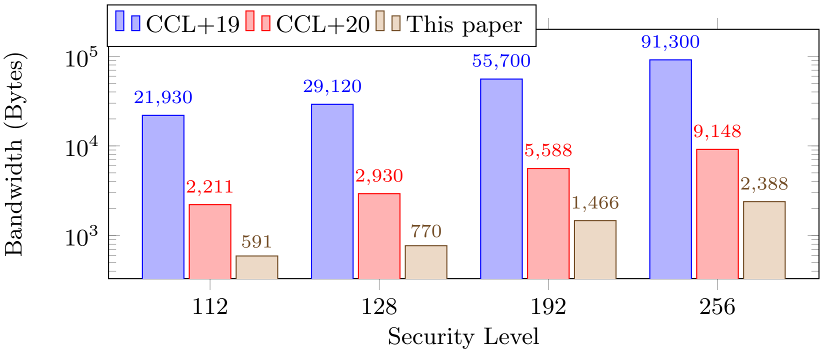


Fig. 1: Comparison of ZK Proof of DL relation in HSM group.

to the RSA assumption. On the other hand, the security of our ZK proofs requires the use of generic group model while the security of the ZK proofs in [5] does not.

Two-party ECDSA. The two-party ECDSA scheme CCL+19 [4] has an efficient ISign algorithm, with the drawback of running the IKeyGen algorithm with a communication size of $>$ 50kB and a running time of $>$ 60s for 128-bit security level. Recently, [5] improved the IKeyGen algorithm in [4] by adding an *lcm* trick upon it (we denote it as CCL+19-*lcm*). In this paper, we implement these schemes and find out that CCL+19-*lcm* has a non-obvious cost of doubling the running of ISign as compared with CCL+19.

We propose a new two-party ECDSA (§5.1) by modifying the ZK proof of the well-formedness of CL ciphertext, such that the plaintext encrypted is related to an ECC group element. As compared with CCL+19 [4] or CCL+19-*lcm* ([5] §5.1), we use a single round ZK proof to replace the multiple rounds of ZK proofs. Our new two-party ECDSA outperforms the state-of-the-art CCL+19-*lcm* in most aspects. Our scheme uses 47% less bandwidth in IKeyGen than the CCL+19-*lcm* for 128-bit security. The running time of our IKeyGen is 35% faster, and the running time of our ISign is 104% faster. Detailed comparison in terms of security assumptions and security models are discussed in §5.1.

Threshold ECDSA. For threshold ECDSA, the major bottleneck for the IKeyGen algorithm for the threshold ECDSA in CCL+20 [5] is that the ZK proof for the well-formedness of a CL ciphertext requires a random group generator g_q as discussed above (from the strong root assumption). As a result, their IKeyGen algorithm requires an additional interactive ISetup algorithm to generate such g_q. This ISetup algorithm requires a ZK proof of DL relation in the class group for n parties.

By using our bandwidth efficient ZK proof of DL relation, we can remove this complicated IKeyGen algorithm. It is because our underlying adaptive root assumption does not require a random group generator g_q. We can build a bandwidth efficient threshold ECDSA (Scheme 2 in §5.2) with about 60% smaller bandwidth than [5] for $(t, n) = (1,3)$, $(2,4)$ and $(2,5)$[5]. The running time of our $\mathsf{IKeyGen}$ is 46-65% faster. Our scheme 1 is even more optimized for the key generation algorithm (about 70% lower bandwidth and 85-90% faster computation in key generation than CCL+20), at a cost of 20% larger bandwidth in signing. Detailed comparison in terms of security assumptions and security models are discussed in §5.2.

2 Backgrounds

We review some definitions of groups and introduce some intractability assumptions in these groups. In particular, we will use a group where the *hard subgroup membership* assumption [4] holds.

For a distribution $\mathcal{D}$, we write $d \hookleftarrow \mathcal{D}$ to refer to d being sampled from $\mathcal{D}$ and $b \overset{\$}{\leftarrow} B$ if b is sampled uniformly in the set B. We use $\mathsf{negl}(\lambda)$ (resp. $\exp(\lambda)$) to represent a negligible (resp. exponential) function in λ. We denote $\mathrm{ord}_{\mathbb{G}}(g)$ as the order of $g \in \mathbb{G}$. We denote ϵ_s and ϵ_d as the parameter for soundness error and statistical distance respectively.

2.1 Groups

We define some group generation algorithms as in [4]:

- On input a security parameter 1^λ, the $\mathsf{GGen}_{\mathrm{ECC}}$ algorithm generates a cyclic group $\hat{G}$ with prime order q and $\hat{P}$ is a generator of $\hat{G}$. It outputs $\mathcal{G}_{\mathrm{ECC}} = (\hat{G}, q, \hat{P})$.
- On input a security parameter 1^λ and a prime number q, the $\mathsf{GGen}_{\mathrm{HSM}}$ algorithm outputs $\mathcal{G}_{\mathrm{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$.

  The set $(\tilde{G}, \cdot)$ is a finite abelian group 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 $\tilde{s}$ is the upper bound of $\hat{s}$. One can decide if an element is in $\tilde{G}$ in polynomial time. The set $(F, \cdot)$ is the unique cyclic subgroup of $\tilde{G}$ of order q, generated by f. The group $G^q := \{x^q, x \in G\}$ is the subgroup of order s of G, generated by g_q. The set $(G, \cdot)$ is a cyclic subgroup of $\tilde{G}$ of order $q \cdot s$, where s divides $\hat{s}$. By construction $F \subset G$, it holds that $G = G^q \times F$ and $g := f \cdot g_q$ is the generator of G. The discrete logarithm problem in F can be solved by a polynomial time algorithm Solve:

  $$x \leftarrow \mathsf{Solve}_{\mathcal{G}_{\mathrm{HSM}},q}(f^x), \quad \forall x \overset{\$}{\leftarrow} \mathbb{Z}_q.$$

  We drop the subscript for Solve when the context is clear. For simplicity, we will call this group the *HSM group*.

[5] These are the most popular types of threshold signatures in Bitcoin's P2SH transactions as shown in https://txstats.com/dashboard/db/p2sh-repartition-by-type?orgId=1. Hence we use these 3 settings for comparison in this paper.

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

The $\mathsf{GGen}_{\mathrm{HSM}}$ algorithm picks a random prime $\tilde{q}$ such that $q\tilde{q} \equiv 1 \pmod 4$ and $(q/\tilde{q}) = -1$. It computes $\Delta_K = -q\tilde{q}, \Delta_q = q^2\Delta_K$. Denote $\tilde{G}$ as the class group $Cl(\Delta_q)$, whose order is $h(\Delta_q) = q \cdot h(\Delta_K)$. It computes $\tilde{s} := \left\lceil \frac{1}{\pi} \log |\Delta_K| \sqrt{|\Delta_K|} \right\rceil$ such that $h(\Delta_K) < \tilde{s}$.

It sets $f = [(q^2, q)] \in Cl(\Delta_q)$ and $F = \langle f \rangle$. Let r be a small prime, with $r \neq q$ and $(\frac{\Delta_K}{r}) = 1$. It sets I as an ideal lying above r. Denote φ_q^{-1} as the surjection defined in the Algorithm 1 of [6]. It computes $g_q = [\varphi_q^{-1}(I^2)]^q \in Cl(\Delta_q)$ and sets $G^q = \langle g_q \rangle$. It computes $g = f \cdot g_q$ and sets $G = \langle g \rangle$. It outputs $\mathcal{G}_{\mathrm{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$.

2.2 ECDSA

We review the ECDSA below.

Setup. On input a security parameter 1^λ, it runs $\mathcal{G}_{\mathrm{ECC}} \leftarrow \mathsf{GGen}_{\mathrm{ECC}}(1^\lambda)$. It outputs $\mathsf{param} = \mathcal{G}_{\mathrm{ECC}}$. The input param is omitted for other algorithms for simplicity.

KeyGen. It picks a random secret key $x \xleftarrow{\$} \mathbb{Z}_q$ and computes a public key $\hat{Q} = \hat{P}^x$. It returns $(\hat{Q}, x)$.

Sign. On input a message m, it picks $k \xleftarrow{\$} \mathbb{Z}_q$. It computes $\hat{R} = (r_x, r_y) = \hat{P}^k$, $r = r_x \bmod q$ and $s = k^{-1}(xr + H(m)) \bmod q$. It outputs the signature (r, s).

Verify. On input a public key $\hat{Q}$, a message m and a signature (r, s), it computes $\hat{R} = (r_x, r_y) = (\hat{Q}^r \hat{P}^{H(m)})^{1/s}$. It outputs 1 if $r = r_x \bmod q$. Otherwise, it outputs 0.

2.3 CL Encryption from HSM Group

Castagnos and Laguillaumie [7] introduced a framework of a group with an easy DL subgroup. We review the additive homomorphic CL encryption algorithm instantiated from class groups of quadratic fields [4].

Setup. On input a security parameter 1^λ and a prime q, it runs $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$. It parses $\mathcal{G}_{\mathrm{HSM}} = (\tilde{s}, g, f, g_q, \tilde{G}, G, F, G^q)$. Define $S = \tilde{s} \cdot 2^{\epsilon_d}$ for some statistical distance ϵ_d. It outputs $\mathsf{param} = \mathcal{G}_{\mathrm{HSM}}$. The input param is omitted for other algorithms for simplicity.

KeyGen. It picks a random $\mathsf{sk} \xleftarrow{\$} [0, S]$ and computes $\mathsf{pk} = g_q^{\mathsf{sk}}$. It returns $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Encrypt}$. On input a public key pk and a message m, it picks a random $\rho \xleftarrow{\$} [0, S]$ and outputs the ciphertext $C = (C_1, C_2)$, where:

$$C_1 = f^m \mathsf{pk}^\rho, \quad C_2 = g_q^\rho.$$

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

$\mathsf{EvalScal}$. On input a public key pk, a ciphertext $C = (C_1, C_2)$ and a scalar s, it outputs $C' = (C'_1 = C_1^s, C'_2 = C_2^s)$.

$\mathsf{EvalSum}$. On input a public key pk, two ciphertexts $C = (C_1, C_2)$ and $C' = (C'_1, C'_2)$, it outputs $\hat{C} = (\hat{C}_1 = C_1 C'_1, \hat{C}_2 = C_2 C'_2)$.

3 Generic Group Model for HSM Group

We use the generic group model for groups of unknown order [8] together with groups of known order to model the HSM group.

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

A generic group algorithm $\mathcal{A}$ is a probabilistic algorithm. Let $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_1$ be a list that is initialized with the encodings. $\mathcal{A}$ is given $(q, \mathcal{L})$ as input. The algorithm can query two generic group oracles:

- $\mathcal{O}_1$ takes a bit b'. If $b' = 0$, it samples a random $a \in \mathbb{Z}_{|\mathbb{G}_1|}$, $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(a, b))$. It is appended to the list of encodings $\mathcal{L}_0$. If $b' = 1$, it samples a random $b \in \mathbb{Z}_q$ and returns $\sigma(\pi(0, b))$[6]. It is appended to the list of encodings $\mathcal{L}_1$.
- When $\mathcal{L}$ has size $\tilde{q}$, the second oracle $\mathcal{O}_2(i, j, \pm)$ takes two indices $i, j \in [1, \tilde{q}]$ and a sign bit, and returns $\sigma(\pi(a_i \pm a_j \mod |\mathbb{G}_1|, b_i \pm b_j \mod q)$, which is appended to $\mathcal{L}_1$ if $a_i \pm a_j \neq 0 \bmod |\mathbb{G}_1|$. Otherwise, it is appended to $\mathcal{L}_0$.

For the group $\mathcal{G}_{\text{HSM}}$, 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 elements in G. The generator g_q in G^q is initialized as $\sigma(\pi(a, 0))$ for some random a. Given the output of $\mathcal{O}_1(0)$, it is difficult to distinguish if it is in G^q or not. Suppose that f is initialized as $\sigma(\pi(0, b^*))$ for some $b^* \in \mathbb{Z}_q$. The Solve algorithm for input $\tilde{f} \in F$ can be modelled 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 $\tilde{b}/b^* \bmod q$.

Lemma 1 (Element Representation [16]). *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 $\{g_1, \ldots, g_m\}$ be the outputs of $\mathcal{O}_1$. 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 $(\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}^m$ such that $u = \prod_{i=1}^m g_i^{\alpha_i}$ and $\alpha_i \leq 2^{q_2}$.*

[6] The random encoding for DL-easy subgroup is necessary, since the adversary may obtain some $g' = \sigma(\pi(a_1, b_1))$ and $f' = \sigma(\pi(0, b_2))$ from $\mathcal{O}_1$. The adversary can obtain $g' \cdot f'$ or $(g')^2/f'$ from $\mathcal{O}_2$. The encodings b_1 and b_2 ensure that the value in the DL-easy subgroup is always correct even when the computation involves elements in $\mathbb{G}_1$.

Lemma 2 (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 $\alpha_i \leq 2^{q_2}$.*

Proof. Suppose that there is an algorithm $\mathcal{A}$ of this lemma and we will show how to build the extractor Ext. Ext first runs as an algorithm $\mathcal{A}'$ in the Lemma 1. $\mathcal{A}'$ is given initial encodings from its challenger and forwards them to $\mathcal{A}$. When $\mathcal{A}$ makes an oracle query, $\mathcal{A}'$ forwards them to its challenger to get the answer. Finally, $\mathcal{A}$ outputs an element $u \in \mathbb{G}$. $\mathcal{A}'$ forwards u to its challenger. By Lemma 1, there exists an extractor Ext' that outputs, a tuple $(\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}^m$ such that $u = \prod_{i=1}^{m} g_i^{\alpha_i}$ and $\{g_1, \ldots, g_m\}$ are the outputs of $\mathcal{O}_1$. W.l.o.g., assume that $(g_1, \ldots, g_{m_0})$ are the outputs of $\mathcal{O}_1(0)$ and $(g_{m_0+1}, \ldots, g_m)$ are the outputs of $\mathcal{O}_1(1)$. Ext can compute $\beta_i = \log_f g_i \in \mathbb{Z}_q$ for $i \in [m_0 + 1, m]$ by running the Solve algorithm. Hence, Ext can compute $\gamma = \sum_{i=m_0+1}^{m} \beta_i \alpha_i \bmod q$ and can output $(\alpha_1, \ldots, \alpha_{m_0}, \gamma)$ such that $u = f^\gamma \cdot \prod_{i=1}^{m_0} g_i^{\alpha_i}$ and $\alpha_i \leq 2^{q_2}$. □

Lemma 3 (Subgroup Hidden Order). *Let $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2$ be a generic group where $|\mathbb{G}_1|$ is a uniformly chosen integer in $[A, B]$. Let $\mathcal{A}$ be a generic algorithm making q_1 queries to $\mathcal{O}_1(0)$ and q_2 queries to $\mathcal{O}_2$. The probability that $\mathcal{A}$ succeeds in computing $0 \neq k \in \mathbb{N}$ such that for a g which is a response to an $\mathcal{O}_1(0)$ query $g^k = 1$ is at most $\frac{(q_1+q_2)^3}{M}$, where $1/M$ is negligible whenever $|B - A| = \mathsf{exp}(\lambda)$.*

It follows from the Lemma 3 of [1]. If there is an $\mathcal{A}$ succeeds in this lemma, it is easy to build an algorithm $\mathcal{A}'$ which succeeds in the Lemma 3 of [1].

Lemma 4 (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 $f^\gamma \prod_{i=1}^{m_0} g_i^{\alpha_i} = f^\delta \prod_{i=1}^{m_0} g_i^{\beta_i} \in \mathbb{G}$, $\alpha_i \neq \beta_i$ and $\gamma \neq \delta$ mod q, is negligible.*

Proof. By Lemma 2, every group element u in $\mathbb{G}$ that the adversary obtains from the $\mathcal{O}_2$ query can be written as $u = f^\gamma \prod_{i=1}^{m_0} g_i^{\alpha_i}$ for some known $\alpha_i \in \mathbb{Z}$, $\gamma \in \mathbb{Z}_q$. Let $h = f^\delta \prod_{i=1}^{m_0} g_i^{\beta_i}$ be another such a group element.

If there is some $i \in [1, m_0]$ for which $\alpha_i \not\equiv \beta_i \bmod \mathrm{ord}_{\mathbb{G}}(g_i)$ or $\gamma \not\equiv \delta \bmod q$, then the probability that $u = h$ is at most $\frac{(q_1+q_2)^2}{A}$ as shown in [8]. Therefore when $f^\gamma \prod_{i=1}^{m_0} g_i^{\alpha_i} = f^\delta \prod_{i=1}^{m_0} g_i^{\beta_i}$, then $\alpha_i \equiv \beta_i \bmod \mathrm{ord}_{\mathbb{G}}(g_i)$ and $\gamma \equiv \delta \bmod q$ with non-negligible probability if $1/A$ is negligible.

If $\alpha_i \equiv \beta_i \bmod \mathrm{ord}_{\mathbb{G}}(g_i)$, we have either $\alpha_i = \beta_i$ or $\alpha_i = \beta_i + K{\cdot}\mathrm{ord}_{\mathbb{G}}(g_i)$ for some integer K. By Lemma 3, $\alpha_i = \beta_i$ with overwhelming probability $(1 - \frac{(q_1+q_2)^3}{M}$, where $1/M$ is negligible whenever $|B - A| = \mathsf{exp}(\lambda))$. □

3.1 Assumptions

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

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

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

Adaptive Root Subgroup Assumption. We define the adaptive root subgroup assumption, which is the modification of the adaptive root assumption [1] in the group $\mathcal{G}_{\mathrm{HSM}}$. We denote $\mathsf{Primes}(\lambda)$ as the set of odd primes less than 2^λ.

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

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

The next two corollaries show that the adaptive root subgroup problem and the non-trivial order element problem are intractable in a generic group model.

Corollary 1. *(Adaptive Root Subgroup Hardness). Let $G \in \mathcal{G}_{\mathsf{HSM}}$ 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 breaking the adaptive root subgroup assumption on $\mathcal{G}_{\mathsf{HSM}}$ with at most negligible probability in λ.*

Proof. Recall that the adversary outputs $u, w \in G$ for a challenge ℓ such that $u^\ell = w$ and $w^q \neq 1$. According to Lemma 2, we can write $u = f^\gamma \prod_{i=1}^m g_i^{\alpha_i}$ and $w = f^\delta \prod_{i=1}^m g_i^{\beta_i}$, where $\{g_1, \ldots, g_m\}$ is the outputs of $\mathcal{O}_1(0)$. Since $w^q \neq 1$, there exists some $i^* \in [1, m]$ such that $\beta_{i^*} \neq 0$.

According to Lemma 4, we know that $\alpha_{i^*}\ell = \beta_{i^*} \bmod \mathrm{ord}_{\mathbb{G}}(g_{i^*})$ with overwhelming probability $1 - \epsilon$. Hence, $\alpha_{i^*}\ell = \beta_{i^*} + k{\cdot}\mathrm{ord}_{\mathbb{G}}(g_{i^*})$ for some $k \in \mathbb{Z}$. According to Lemma 3, an efficient adversary can compute a multiple of the order of the group G^q with at most negligible probability ϵ'. It follows that $k = 0$ and $\alpha_{i^*}\ell = \beta_{i^*}$ with probability greater than $1 - \epsilon - \epsilon'$ Hence, ℓ must divides β_{i^*}. However, β_{i^*} is chosen before ℓ and if $\mathcal{A}$ makes q_2 generic group queries then $\beta_{i^*} \le 2^{q_2}$. The probability that ℓ divides β_{i^*} is bounded by the probability that a random prime in $\mathsf{Primes}(\lambda)$ divides a number less than 2^{q_2}. Any such a number has less than q_2 distinct prime factors and there are more than $2^\lambda/\lambda$ primes in $\mathsf{Primes}(\lambda)$. Therefore, the probability that ℓ divides β_{i^*} is at most $\frac{q_2\lambda}{2^\lambda}$.

Overall, we obtain that a generic adversary can break the adaptive root subgroup assumption with probability at most $\frac{(q_1+q_2)^2}{A} + \frac{2(q_1+q_2)^3}{M} + \frac{q_2\lambda}{2^\lambda}$, which is negligible if $1/A$ and $1/|B-A|$ are negligible in λ and q_1, q_2 are bounded by some polynomials in λ. □

Corollary 2. *(Non-trivial order hardness). Let $G \in \mathcal{G}_{\mathsf{HSM}}$ 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.*[7]

Proof. Suppose that $\mathcal{B}$ an adaptive root adversary that is given G from its challenger. $\mathcal{B}$ gives G to $\mathcal{A}$. When $\mathcal{A}$ makes an oracle query, $\mathcal{B}$ forwards it to its challenger. $\mathcal{A}$ returns h and d to $\mathcal{B}$.

We claim that $h^q \neq 1$. Assume that on the contrary $h^q = 1$. We have $0 < d < q$, $h^d = 1$. Denote that $q' = q \bmod d$. Then $h^{q'} = 1$ and $0 < q' < d$. Since q is prime and $0 < d < q$, $\gcd(d, q) = 1$. By the Euclidean algorithm, we can apply the same computation recursively until we get $h^1 = 1$, which is a contradiction. Hence $h^q \neq 1$.

Since $h^q \neq 1$, $\mathcal{B}$ sends h to its challenger and receives a prime ℓ. With non-negligible probability, ℓ is relative prime to d. If so, $\mathcal{B}$ computes $c = \ell^{-1} \bmod d$. $\mathcal{B}$ returns $h^c = h^{1/\ell}$ to its challenger. Since the adaptive root assumption holds in the generic group model, $\mathcal{A}$ succeeds with negligible probability. □

4 ZK Proofs for HSM Group with Trustless Setup

In this section, we will give two different ZK proofs for HSM groups. The definition of an argument system is given in the Appendix A.1.

4.1 Argument of Knowledge for Exponentiation

We first construct an argument of knowledge for the following relation about exponentiation within a group G with order q subgroup F:

$$\mathcal{R}_{\mathsf{ExpS}} = \{w \in G; x \in \mathbb{Z} : w = g^x \neq 1\},$$

where g and G are the parameters in the CRS $\mathcal{G}_{\mathsf{HSM}}$. The ZK proof is given in Algorithm 3.

Lemma 5. *Protocol* PoKES *is an argument of knowledge of* $\mathcal{R}_{\mathsf{ExpS}}$ *in the generic group model.*

Proof. We describe the extractor Ext:

1. W.l.o.g. let $g_1 = g$ be encoded in the CRS.

[7] Non-trivial order hardness is similar to the low order assumption in [5], except that their assumption did not rule out the trivial attack that $f^q = 1$.

Algorithm 3: Protocol PoKES for the relation $\mathcal{R}_{\mathsf{ExpS}}$

```
Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^{\lambda})$, $g \in G \setminus F$
Input: $w \in G$
Witness: $x \in \mathbb{Z}$
Prover finds $d \in \mathbb{Z}$ and $e \in [0, q-1]$ s.t. $x = dq + e$. Prover sends $D = g^{d}$ and $e$
  to the verifier.
If $e \in [0, q-1]$ and $D^{q} g^{e} = w$, verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
Prover finds $q' \in \mathbb{Z}$ and $r \in [0, \ell-1]$ s.t. $x = q'\ell + r$. Prover sends $Q = g^{q'}$ and
  $r$ to the verifier.
Verifier accepts if $r \in [0, \ell-1]$ and $Q^{\ell} g^{r} = w$.
```

2. Run $\mathcal{A}_0$ to get output (w, state).
3. Let $\mathcal{L} \leftarrow \{\}$.
4. Run Protocol PoKES with $\mathcal{A}_1$ on input (w, state), sampling fresh randomness for the verifier. If the transcript (D, e, ℓ, Q, r) is accepting set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(r, \ell)\}$, and otherwise repeat this step.
5. Use the CRT algorithm to compute x such that $x = r_i \bmod \ell_i$ for each $(r_i, \ell_i) \in \mathcal{L}$. If $g^{x} = w$, output x and stop. Otherwise, return to Step 4.

It remains to argue that Ext succeeds with overwhelming probability in a poly(λ) number of rounds. Suppose that after some polynomial number of rounds the extractor has obtained M accepting transcripts $\{D, e, \ell_i, Q_i, r_i\}$ for independent values of $\ell_i \in \mathsf{Primes}(\lambda)$.

Consider an accepting transcripts (D, e, ℓ_1, Q_1, r_1) such that $w = Q_1^{\ell_1} g^{r_1} = D^{q} g^{e}$. By Lemma 2, we can write $Q_1 = f^{\gamma} \prod_{i=1}^{m} g_i^{\alpha_i}$[8], and $D = f^{\nu} \prod_{i=1}^{m} g_i^{\mu_i}$. Hence:

$$
\begin{aligned}
Q_1^{\ell_1} g^{r_1} &= f^{\gamma \ell_1} g^{r_1} \prod_{i=1}^{m} g_i^{\alpha_i \ell_1} = f^{\gamma \ell_1} g^{\alpha_1 \ell_1 + r_1} \prod_{i=2}^{m} g_i^{\alpha_i \ell_1} \\
=& D^{q} g^{e} = f^{\nu q} g^{e} \prod_{i=1}^{m} g_i^{\mu_i q} = g^{\mu_1 q + e} \prod_{i=2}^{m} g_i^{\mu_i q}
\end{aligned}
$$

By Lemma 4, $\gamma \ell_1 = 0 \bmod q$. Also, $\alpha_i \ell_1 = \mu_i q$ for all $i \in [2, m]$ with probability 1 - ϵ. Therefore ℓ_1 divides $\mu_i q$. Since $\ell_1 \neq q$, then ℓ_1 divides μ_i since ℓ_1 and q are relatively prime. However, $\mu_i \leq 2^{q_2}$ and μ_i is chosen before ℓ_1 is sampled. Hence the probability that ℓ_1 divides some non-zero μ_i is at most $\frac{q_2 \lambda \ln 2}{2^{\lambda}}$. We conclude that $\alpha_i = \mu_i = 0$ for $i \in [2, m]$ with probability 1 - $\epsilon - \frac{q_2 \lambda \ln 2}{2^{\lambda}}$. Hence, we can express $w = g^{\alpha_1 \ell_1 + r_1}$ for some integers α_1, r_1.

By the argument above, with overwhelming probability there exists $x \in \mathbb{Z}$ such that $x = r_i \bmod \ell_i$ and $g^{x} = w$ and $x < 2^{q_2}$. Hence, the CRT algorithm used in Step 5 will recover the required x once $|\mathcal{L}| > q_2$.

[8] Since $g = g_1$, if Q_1 is computed from f, g_i and $w = g^{x} = g_1^{x}$, we can write $Q_1 = f^{\gamma} \prod_{i=1}^{m} g_i^{\alpha_i}$.

Algorithm 4: Protocol $\mathsf{ZKPoKRepS}$ for the relation $\mathcal{R}_{\mathsf{RepS}}$

```
Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $g_1, \ldots, g_n \in G \setminus F$, $B = n2^{\lambda+\epsilon_d+1}|G|$ where
    $\epsilon_d = 80$.
Input: $w \in G$.
Witness: $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$.
1 Prover chooses $k_1, \ldots, k_n \xleftarrow{\$} [-B, B]$, $t \xleftarrow{\$} \mathbb{Z}_q$ 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.
3 Prover computes $s_i = k_i + cx_i$ for $i \in [1, n]$. Prover finds $d_i \in \mathbb{Z}$, $e_i \in [0, q-1]$
    s.t. $s_i = d_i q + e_i$ and sends $D = \prod_{i=1}^n g_i^{d_i}$ and $\boldsymbol{e} = (e_1, \ldots, e_n)$ to the verifier.
4 If $e_1, \ldots, e_n \in [0, q-1]$ and $D^q \prod_{i=1}^n g_i^{e_i} = Rw^c$, verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
5 Prover finds $q_i \in \mathbb{Z}$ and $r_i \in [0, \ell-1]$ s.t. $s_i = q_i\ell + r_i$ for $i \in [1, n]$. Prover
    sends $Q = \prod_{i=1}^n g_i^{q_i}$ and $\boldsymbol{r} = (r_1, \ldots, r_n)$ to the verifier.
6 Verifier accepts if $r_1, \ldots, r_n \in [0, \ell-1]$ and $Q^\ell \prod_{i=1}^n g_i^{r_i} = Rw^c$.
```

Since a single round of interaction with $\mathcal{A}_1$ results in an accepting transcript with probability $\epsilon \geq 1/\mathsf{poly}(\lambda)$, in expectation the extractor obtains $|\mathcal{L}| > q_2$ accepting transcripts for independent primes ℓ_i after $q_2 \cdot \mathsf{poly}(\lambda)$ rounds. Hence, Ext outputs x such that $g^x = w$ in expected polynomial time, as required. □

Note that there are more than $2^\lambda/\lambda$ primes in $\mathsf{Primes}(\lambda)$ and it can be instantiated by a hash to prime function [1]. The soundness error is about $1/2^{\lambda - \log_2 \lambda}$ if ℓ is λ bits.

4.2 ZK Proof for Multi-exponentiation

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

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

where $g_1, \ldots, g_n \in G \backslash F$ are in the CRS $\mathcal{G}_{\mathsf{HSM}}$. The ZK proof is given in Algorithm 4.

Theorem 1. *Protocol* $\mathsf{ZKPoKRepS}$ *is an argument of knowledge for* $\mathcal{R}_{\mathsf{RepS}}$ *in the generic group model.*

Proof. 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 $\mathsf{ZKPoKRepS}$ with $\mathcal{A}_1$ on input (w, state).
3. Run Step 2-3 of Protocol $\mathsf{ZKPoKRepS}$ with $\mathcal{A}_1$, sampling fresh randomness c for the verifier.
4. Run Step 4-5 of Protocol $\mathsf{ZKPoKRepS}$ with $\mathcal{A}_1$, sampling fresh randomness ℓ for the verifier. If the transcript $(R, c, z, D, \boldsymbol{e}, \ell, Q, \boldsymbol{r})$ is accepting, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\boldsymbol{r}, \ell)\}$, and otherwise repeat this step.

5. Use the CRT algorithm to compute $\boldsymbol{s} = (s_1, \ldots, s_n)$ such that $\boldsymbol{s} = \boldsymbol{r}_i \bmod \ell_i$ for each $(\boldsymbol{r}_i, \ell_i) \in \mathcal{L}$. If $\prod_{i=1}^n g_i^{s_i} \neq Rw^c$, return to Step 4.
6. Consider the intermediate transcript as $(R, c, D, \boldsymbol{e}, \boldsymbol{s})$. Run from Step 4 for the second time and obtain $(R, c', D', \boldsymbol{e}', \boldsymbol{s}')$.
7. Compute $\Delta_{s_i} = s_i - s_i'$ for $i \in [1, n]$ and $\Delta_c = c - c'$. Output $\boldsymbol{x} = (x_1, \ldots, x_n)$ for $x_i = \Delta_{s_i} / \Delta_c$.

Analysis for Step 5. This is a generalization of the protocol PoKES. We first argue that $\prod_{i=1}^n g_i^{s_i} = Rw^c$ with overwhelming probability in a poly(λ) number of rounds. Suppose that after some polynomial number of rounds the extractor has obtained M accepting transcripts $\{R, c, D, \boldsymbol{e}, \ell_i, Q_i, \boldsymbol{r}_i\}$ for independent values of $\ell_i \in \mathsf{Primes}(\lambda)$.

Consider an accepting transcripts $(R, c, D, \boldsymbol{e} = (e_1, \ldots, e_n), \ell_1, Q_1, \boldsymbol{r}_1 = (r_{1,1}, \ldots, r_{1,n}))$ such that $Rw^c = Q_1^{\ell_1} \prod_{i=1}^n g_i^{r_{1,i}} = D^q \prod_{i=1}^n g_i^{e_i}$. By Lemma 2, we can write $Q_1 = \prod_{i=1}^m g_i^{\alpha_i} \cdot f^\gamma$ and $D = \prod_{i=1}^m g_i^{\beta_i} \cdot f^\delta$. Hence:

$$Q_1^{\ell_1} \prod_{i=1}^n g_i^{r_{1,i}} = \prod_{i=1}^n g_i^{\alpha_i \ell_1 + r_{1,i}} \prod_{i=n+1}^m g_i^{\alpha_i \ell_1} \cdot f^{\gamma \ell_1} = D^q \prod_{i=1}^n g_i^{e_i}$$
$$= \prod_{i=1}^n g_i^{\beta_i q + e_i} \prod_{i=n+1}^m g_i^{\beta_i q} \cdot f^{\delta q} = \prod_{i=1}^n g_i^{\beta_i q + e_i} \prod_{i=n+1}^m g_i^{\beta_i q}.$$

By Lemma 4, $\alpha_i \ell_1 = \beta_i q$ for all $i \in [n+1, m]$ with overwhelming probability. Therefore ℓ_1 divides $\beta_i q$. Since $\ell_1 \neq q$, ℓ_1 and q are relatively prime and ℓ_1 divides β_i. However, $\beta_i \leq 2^{q_2}$ and β_i are chosen before ℓ_1 is sampled. Hence the probability that ℓ_1 divides some non-zero β_i is at most $\frac{q_2 \lambda \ln 2}{2^\lambda}$. We conclude that with overwhelming probability $\alpha_i = \beta_i = 0$ for $i \in [n+1, m]$. Also by Lemma 4, $\gamma \ell_1 = 0 \bmod q$. Hence, we can express $Rw^c = \prod_{i=1}^n g_i^{\alpha_i \ell_1 + r_{1,i}}$ for some integers $\alpha_i, r_{1,i}$.

By the argument above, with overwhelming probability there exists $\boldsymbol{s} \in \mathbb{Z}^n$ such that $\boldsymbol{s} = \boldsymbol{r}_i \bmod \ell_i$, $s_i < 2^{q_2}$ for all $s_i \in \boldsymbol{s}$ and $\prod_{i=1}^n g_i^{s_i} = Rw^c$. Hence, the CRT algorithm used in Step 5 will recover the required vector $\boldsymbol{s}$ once $|\mathcal{L}| > q_2$. Since a single round of interaction with $\mathcal{A}_1$ results in an accepting transcript with probability $\epsilon \geq 1/\mathsf{poly}(\lambda)$, in expectation the extractor obtains $|\mathcal{L}| > q_2$ accepting transcripts for independent primes ℓ_i after $q_2 \cdot \mathsf{poly}(\lambda)$ rounds. Hence, Ext outputs $\boldsymbol{s}$ such that $\prod_{i=1}^n g_i^{s_i} = Rw^c$ in expected polynomial time.

Analysis for Step 7. It remains to argue that Ext succeeds with overwhelming probability in Step 7. W.l.o.g., assume that $c > c'$, by Step 6, we have $\prod_{i=1}^n g_i^{s_i} \cdot w^{-c} = \prod_{i=1}^n g_i^{s_i'} \cdot w^{-c'}$. Then $\prod_{i=1}^n g_i^{\Delta_{s_i}} = w^{\Delta_c} = (\prod_{i=1}^m g_i^{\alpha_i'} \cdot f^{\gamma'})^{\Delta_c}$ for some $\alpha_i' \in \mathbb{Z}$ and $\gamma' \in \mathbb{Z}_q$ by Lemma 2. By Lemma 4, $\Delta_{s_i} = \alpha_i' \Delta_c$ 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, 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 $\boldsymbol{x} = (x_1, \ldots, x_n)$ can be extracted as in Step 7. □

Theorem 2. *The protocol* $\mathsf{ZKPoKRepS}$ *is an honest-verifier statistically zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathsf{RepS}}$ *in the generic group model.*

Proof. The simulator Sim picks a random challenge $c' \xleftarrow{\$} [0, q-1]$ and $\ell' \xleftarrow{\$} \mathsf{Primes}(\lambda)$. It picks random $q'_1, \ldots, q'_n, \xleftarrow{\$} [0, B-1]$, $r'_1, \ldots, r'_n \xleftarrow{\$} [0, \ell-1]$. It finds $d'_i \in \mathbb{Z}$ and $e'_i \in [0, q-1]$ such that $d'_i q + e'_i = q'_i \ell' + r'_i$. It computes:

$$Q' = \prod_{i=1}^{n} g_i^{q'_i}, \quad D' = \prod_{i=1}^{n} g_i^{d'_i}, \quad R' = D'^q \prod_{i=1}^{n} g_i^{e'_i} \cdot w^{-c'}.$$

We argue that the transcript $(R', c', (D', \boldsymbol{e'} = (e'_1, \ldots, e'_n)), \ell', (Q', \boldsymbol{r'} = (r'_1, \ldots, r'_n)))$ is indistinguishable from a real transcript between a prover and a verifier. Sim chooses ℓ', c' identically to the honest verifier. It also solves $R', D', \boldsymbol{e'}$ uniquely from the other values such that the verification holds.

We must show that in the real protocol, independent of ℓ and c, the values in $\boldsymbol{r}$ have a negligible statistical distance from the uniform distribution over $[0, \ell-1]$ and each $g_i^{q_i}$ has a negligible statistical distance from uniform over G. In addition we must argue that Q and $\boldsymbol{r}$ 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 $s_1 + s_2 + \epsilon_1 |\mathrm{supp}(X_2)| + \epsilon_2 |\mathrm{supp}(Y_2)|$, 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_i is uniform in $[-B, B]$ and t is uniform in $\mathbb{Z}_q$, and sets $r_i = s_i \bmod \ell$. By Fact 1, the value of s_i is distributed uniformly over a range of $2B + 1$ consecutive integers, thus r_i has a statistical distance at most $\ell/(2B+1)$ from uniform over $[0, \ell-1]$. This bounds the distance between the real r_i and the simulated r'_i, which is uniform over $[0, \ell-1]$.

Next, we show that each $g_i^{q_i}$ is statistically indistinguishable from uniform in the subgroup generated by g_i (denoted as G_i). The distribution of $g_i^{q_i}$ over G_i is determined by the distribution of $q_i \bmod |G_i|$. Consider the distribution of $q_i = \left\lfloor \frac{s_i}{\ell} \right\rfloor$ over the consecutive integers in $[\left\lfloor \frac{cx_i - B}{\ell} \right\rfloor, \left\lfloor \frac{cx_i + B}{\ell} \right\rfloor]$. Denote this by the random variable Z. The probability that $q_i = z$ is the probability that s_i falls in the interval $[z\ell, (z+1)\ell - 1]$. Hence $\Pr[q_i = z] = \ell/(2B+1)$ for all $z \in Z$ if $z\ell \geq cx_i - B$ and $(z+1)\ell - 1 \leq cx_i + B$. This probability may or may not hold for the two endpoints $E_1 = \left\lfloor \frac{cx_i - B}{\ell} \right\rfloor$ and $E_2 = \left\lfloor \frac{cx_i + B}{\ell} \right\rfloor$. Denote Y as the set of points with $\Pr[q_i = z] = \ell/(2B+1)$ only. The distance of q_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 $\ell - 1$, i.e., $cx_i - B = 1 \bmod \ell$ and $cx_i + B = \ell - 2 \bmod \ell$. In this case, q_i is one of the two endpoints outside Y with probability $\frac{2(\ell-1)}{2B+1}$. As $|Y| = \frac{2B+3}{\ell} - 3$, the statistical distance of q_i from U_Y is at most $\frac{1}{2}(|Y|(\frac{1}{|Y|} - \frac{\ell}{2B+1}) + \frac{2(\ell-1)}{2B+1}) = \frac{5\ell-4}{2(2B+1)} \leq \frac{2^{\lambda+1}}{B}$. Moreover, the statistical distance of $q_i \bmod |G_i|$ from $U_Y \bmod |G_i|$ is no larger.

By Fact 1, $U_Y \bmod |G_i|$ has a statistical distance at most $\frac{|G_i|}{|Y|} \leq \frac{2^\lambda |G|}{2B+3-3\cdot 2^\lambda} < \frac{2^{\lambda-1}|G|}{B+1-2^\lambda}$. By the triangle inequality, the statistical distance of $q_i \bmod |G_i|$ from uniform is at most $\frac{2^{\lambda+1}}{B} + \frac{2^{\lambda-1}|G|}{B+1-2^\lambda}$. 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 $q_i' \bmod |G_i|$ has a distance $|G_i|/B \leq |G|/B$ from uniform. The simulated value $g_i^{q_i'}$ has a distance at most $|G|/B$ from uniform in G_i. By the triangle inequality, the statistical distance of $g_i^{q_i}$ and $g_i^{q_i'}$ is at most:

$$\begin{aligned} \frac{2^{\lambda+1}}{B} + \frac{2^{\lambda-1}|G|}{B+1-2^\lambda} + \frac{|G|}{B} &< \frac{2^{\lambda-1}|G| + |G| + 2^{\lambda+1}}{B+1-2^\lambda} \\ = \frac{(2^{\lambda-1}+1)|G| + 2^{\lambda+1}}{B+1-2^\lambda} &\leq \frac{1}{n2^{\epsilon_d+1}}, \end{aligned}$$

if $B \geq n2^{\epsilon_d+1}(2^{\lambda-1}+1)|G| + n2^{\epsilon_d+\lambda+2} + 2^\lambda - 1$ for some distance parameter ϵ_d.

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

We can compare the joint distribution $X_i = (g_i^{q_i}, r_i)$ to the simulated distribution $Y_i = (g_i^{q_i'}, r_i')$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{n2^{\epsilon_d+1}} + \frac{\ell}{2B+1} + \epsilon_1 \ell = \frac{1}{n2^{\epsilon_d+1}} + \frac{\ell^2}{2B+3-3\ell} + \frac{\ell(1-\ell)}{2B+1}$. Moreover, as each X_i is independent from X_j for $i \neq j$, we use Fact 2 to bound the distance between joint distributions $(g_1^{q_1}, \ldots, g_n^{q_n}, r_1, \ldots, r_n)$ and $(g_1^{q_1'}, \ldots, g_n^{q_n'}, r_1', \ldots, r_n')$ by the sum of individual distances between each X_i

Table 1: Comparison of ZK Proofs of DL relation in HSM group for $x \in [0, \tilde{s} \cdot 2^{80}]$.

	Communication Size (Bytes)				Remark
	$\lambda = 112$	$\lambda = 128$	$\lambda = 192$	$\lambda = 256$	
CCL+19 [4]	21930	29120	55700	91300	
CCL+20 [5] (*lcm* trick)	2211	2930	5588	9148	Modified relation
This paper	591	771	1467	2389	Generic group model

and Y_i, which is at most:

$$\frac{1}{2^{\epsilon_d+1}} + \frac{n\ell^2}{2B+3-3\ell} + \frac{n\ell(1-\ell)}{2B+1} = \frac{1}{2^{\epsilon_d+1}} + \frac{n\ell(2B+3-5\ell-3\ell^2)}{(2B+3-3\ell)(2B+1)}$$
$$< \frac{1}{2^{\epsilon_d+1}} + \frac{n\ell}{2B+1} < \frac{1}{2^{\epsilon_d}},$$

where the last equality holds if $B > n2^{\epsilon_d+\lambda} - 1$. Finally, this also bounds the distance between $(Q, \boldsymbol{r})$ and $(Q', \boldsymbol{r'})$, where $Q = \prod_i g_i^{q_i}$ and $Q' = \prod_i g_i^{q_i'}$. Combining the two requirements on B, we can simplify the requirement as $B \geq n2^{\lambda+\epsilon_d+1}|G|$. □

Comparison. We compare our scheme with the similar ZK proofs for DL relation in HSM group in [4] and [5]. However, there are some minor differences for the relation to be proven. In our case, we prove the knowledge of $x = \log_{g_1} w$ for some $g_1 \in G \setminus F$ and $x \in \mathbb{Z}$. In the other two schemes, $g_1 \in G^q$ and the range of x is restricted to $x \in [0, S]$ (where $S = \tilde{s} \cdot 2^{40}$ in [5] and $S = \tilde{s} \cdot 2^{\lambda-2}$ in [4]). More importantly, the ZK proof in [5] only proves the knowledge of x such that $h^y = g_1^x$ for some public value y. The relation proved is slightly modified. On the other hand, our proof uses the generic group model. We note that there are some ZK proofs for class group [1] using the generic group model as well.

We compare these schemes in Table 1 by setting $g_1 \in G^q \subset G \setminus F$ and fixing the range S as $\tilde{s} \cdot 2^{80}$. We use 2^{-80} for statistical distance and soundness error for fair comparison. In our scheme, we can set $B = 2^{\lambda+81}\tilde{s}$, where $\tilde{s} := \left\lceil \frac{1}{\pi} \log |\Delta_K| \sqrt{|\Delta_K|} \right\rceil$. Note that the communication size of our scheme does not change much for soundness error $\epsilon_s < \lambda - \log \lambda$ (only the size of ℓ and $\boldsymbol{r}$ are affected).

4.3 ZK Proof for the well-formedness of a CL ciphertext

Consider a prover honestly generated his public key pk and encrypted a message $m \in \mathbb{Z}_q$ using a randomness $\rho \in [0, S]$. We present a zero-knowledge proof of knowledge of the following relation:

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

For the relation $\mathcal{R}_{\mathsf{Enc}}$, we cannot apply the protocol $\mathsf{ZKPoKRepS}$ directly since $f \in F$. We propose a new ZK proof $\mathsf{ZKPoKEnc}$ for $\mathcal{R}_{\mathsf{Enc}}$ in Algorithm 5.

Algorithm 5: Protocol $\mathsf{ZKPoKEnc}$ for the relation $\mathcal{R}_{\mathsf{Enc}}$

```
Param: $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $B = 2^{\lambda+\epsilon_d+2}\tilde{s}$, where $\epsilon_d = 80$.
Input: $C_1, C_2, \mathsf{pk} \in G^q$.
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:
      $S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, \quad S_2 = g_q^{s_\rho}.$
  Prover sends $(S_1, S_2)$ to the verifier.
2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover.
3 Prover computes:
      $u_\rho = s_\rho + c\rho, \quad u_m = s_m + cm \mod q.$
  Prover finds $d_\rho \in \mathbb{Z}$ and $e_\rho \in [0, q-1]$ s.t. $u_\rho = d_\rho q + e_\rho$. Prover computes:
      $D_1 = \mathsf{pk}^{d_\rho}, \quad D_2 = g_q^{d_\rho}.$
  Prover sends $(u_m, D_1, D_2, e_\rho)$ to the verifier.
4 The verifier checks if $e_\rho \in [0, q-1]$ and:
      $D_1^q \mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c, \quad D_2^q g_q^{e_\rho} = S_2 C_2^c.$
  If so, the verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
5 Prover finds $q_\rho \in \mathbb{Z}$ and $r_\rho \in [0, \ell-1]$ s.t. $u_\rho = q_\rho \ell + r_\rho$. Prover computes:
      $Q_1 = \mathsf{pk}^{q_\rho}, \quad Q_2 = g_q^{q_\rho}.$
  Prover sends $(Q_1, Q_2, r_\rho)$ to the verifier.
6 Verifier accepts if $r_\rho \in [0, \ell-1]$ and:
      $Q_1^\ell \mathsf{pk}^{r_\rho} f^{u_m} = S_1 C_1^c, \quad Q_2^\ell g_q^{r_\rho} = S_2 C_2^c.$
```

Theorem 3. *The protocol* $\mathsf{ZKPoKEnc}$ *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 (r_ρ, ℓ), where $s_\rho = r_\rho \mod \ell$ with overwhelming probability.

If $\mathsf{pk}^{s_\rho} \neq S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{s_\rho})^q \neq (S_1 C_1^c f^{-u_m})^q$, then we have:

$$\mathsf{pk}^{s_\rho} \neq S_1 C_1^c f^{-u_m} = Q_1^\ell \mathsf{pk}^{r_\rho} = D_1^q \mathsf{pk}^{e_\rho}.$$

Let $\gamma_\rho = \frac{r_\rho - s_\rho}{\ell}$. Then $Q_1 \mathsf{pk}^{\gamma_\rho}$ is an ℓ-th root of $(S_1 C_1^c f^{-u_m})/\mathsf{pk}^{s_\rho} \neq 1$. This would break the adaptive root subgroup assumption since $(S_1 C_1^c f^{-u_m})^q / (\mathsf{pk}^{s_\rho})^q \neq 1$. If $\mathsf{pk}^{s_\rho} \neq S_1 C_1^c f^{-u_m}$ and $(\mathsf{pk}^{s_\rho})^q = (S_1 C_1^c f^{-u_m})^q$, then $S_1 C_1^c = \mathsf{pk}^{s_\rho} f^{\delta'}$ for some $\delta' \neq u_m \in \mathbb{Z}_q$. It is contradictory to $S_1 C_1^c f^{-u_m} = D_1^q \mathsf{pk}^{e_\rho}$ where $\mathsf{pk} \in G^q$.

Hence by Corollary 1 it follows that $\mathsf{pk}^{s_\rho} f^{u_m} = S_1 C_1^c$ with overwhelming probability.

The extractor obtains a pair of accepting transcripts with (s_ρ, u_m, c) and (s'_ρ, u'_m, c'). The extractor can compute $\Delta_{s_\rho} = s_\rho - s'_\rho$ and $\Delta_{u_m} = u_m - u'_m$ mod q. We denote $\rho = \frac{\Delta_{s_\rho}}{\Delta_c}$ and $m = \frac{\Delta_{u_m}}{\Delta_c}$ mod q. Hence we have:

$$C_1^{\Delta_c} = (\mathsf{pk}^\rho f^m)^{\Delta_c}.$$

If $C_1 \neq \mathsf{pk}^\rho f^m$, then $\frac{\mathsf{pk}^\rho f^m}{C_1}$ is a non-trivial element of order $\Delta c < q$. It contradicts the hardness of computing a non-trivial element and its order in the generic group model (Corollary 2).

Note that our scheme includes a sub-protocol $\mathsf{ZKPoKRepS}$ on input C_2 w.r.t. bases $g_q \in G \setminus F$. Since $\mathsf{ZKPoKRepS}$ 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, sk) such that $C_1 = \mathsf{pk}^\rho f^m$, $C_2 = g_q^\rho$. □

Theorem 4. *The protocol* $\mathsf{ZKPoKEnc}$ *is an honest-verifier statistically zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathsf{Enc}}$ *in the generic group model.*

Proof. The simulator Sim randomly picks a challenge $c' \in [0, q-1]$ and a prime $\ell' \in \mathsf{Prime}(\lambda)$. It picks a random $u'_m \in \mathbb{Z}_q$, $q'_\rho \in [0, B-1]$ and $r'_\rho \in [0, \ell' - 1]$.

It finds $d'_\rho \in \mathbb{Z}$ and $e'_\rho, \in [0, q-1]$ such that

$$d'_\rho q + e'_\rho = q'_\rho \ell' + r'_\rho.$$

It computes:

$$D'_1 = \mathsf{pk}^{d'_\rho}, \quad D'_2 = g_q^{d'_\rho}, \quad Q'_1 = \mathsf{pk}^{q'_\rho}, \quad Q'_2 = g_q^{q'_\rho},$$
$$S'_1 = {Q'_1}^{\ell'} \mathsf{pk}^{r'_\rho} f^{u'_m} C_1^{-c'}, \quad S'_2 = {Q'_2}^{\ell'} g_q^{r'_\rho} C_2^{-c'}.$$

We argue that the simulated transcript $(S'_1, S'_2, c', u'_m, D'_1, D'_2, e'_\rho, \ell', Q'_1, Q'_2, r'_\rho)$ is indistinguishable from a real transcript $(S_1, S_2, c, u_m, D_1, D_2, e_\rho, \ell, Q_1, Q_2, r_\rho)$ 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 $\mathbb{Z}_q$. $(S'_1, S'_2, D'_1, D'_2, e'_\rho)$ is uniquely defined by the other values such that the verification holds.

For simulated transcript (Q'_1, Q'_2, r'_ρ) and real transcript (Q_1, Q_2, r_ρ), the same arguments as in the *Theorem* 2 apply. Namely, in the real protocol, independent of ℓ and c, the values r_ρ has a negligible statistical distance from the uniform distribution over $[0, \ell - 1]$ and each one of $\mathsf{pk}^{q_\rho}, g_q^{q_\rho}$ has negligible statistical from uniform over $G_k = \langle \mathsf{pk} \rangle, G^q$ respectively. In addition, Q_1, Q_2 and r_ρ are independent. Thus, the simulator produces statistically indistinguishable transcripts. The complete proof is as follows.

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 $r_\rho = u_\rho$ mod ℓ. By Fact 1, the value of u_ρ is distributed uniformly over a range of $2B + 1$ consecutive integers, thus r_ρ has a statistical distance at most $\ell/(2B+1)$ from uniform over

$[0, \ell - 1]$. This bounds the distance between the real r_ρ and the simulated r'_ρ, which is uniform over $[0, \ell - 1]$.

Next, we show that $g_q^{q_\rho}$ is statistically indistinguishable from uniform in G^q. The distribution of $g_q^{q_\rho}$ over G^q is determined by the distribution of q_ρ mod $|G^q|$. Consider the distribution of $q_\rho = \left\lfloor \frac{u_\rho}{\ell} \right\rfloor$ over the consecutive integers in $[\left\lfloor \frac{c\rho - B}{\ell} \right\rfloor, \left\lfloor \frac{c\rho + B}{\ell} \right\rfloor]$ Denote this by the random variable Z. The probability that $q_\rho = z$ is the probability that u_ρ falls in the interval $[z\ell, (z+1)\ell - 1]$. This probability is $\ell/(2B+1)$ for all points where $z\ell \geq c\rho - B$ and $(z+1)\ell - 1 \leq c\rho + B$, which includes all points except possibly the two endpoints $\left\lfloor \frac{c\rho - B}{\ell} \right\rfloor$ and $\left\lfloor \frac{c\rho + B}{\ell} \right\rfloor$. Call this set of points Y. The distance of q_ρ from a uniform random variable U_Y over Y is largest when $c\rho - B = 1 \bmod \ell$ and $c\rho + B = \ell - 2 \bmod \ell$. In this case, q_ρ is one of the two endpoints outside Y with probability $\frac{2(\ell-2)}{2B+1}$. For each $z \in Y$, $\Pr[q_\rho = z] = \ell/(2B + 1)$. As $|Y| = \frac{2B+3}{\ell} - 3$, the statistical distance of q_ρ from U_Y is at most: $\frac{1}{2}[Y(\frac{1}{Y} - \frac{\ell}{2B+1}) + \frac{2(\ell-1)}{2B+1}] = \frac{5\ell-4}{2(2B+1)} \leq \frac{2^{\lambda+1}}{B}$. Moreover, the statistical distance of q_ρ mod $|G^q|$ from U_Y mod $|G^q|$ is no larger. By Fact 1, U_Y mod $|G^q|$ has a statistical distance at most $\frac{|G^q|}{|Y|} \leq \frac{2^\lambda |G^q|}{2B+3-3\cdot 2^\lambda} < \frac{2^{\lambda-1}|G^q|}{B+1-2^\lambda}$. By the triangle inequality, the statistical distance of q_ρ mod $|G^q|$ from uniform is at most $\frac{2^{\lambda+1}}{B} + \frac{2^{\lambda-1}|G^q|}{B+1-2^\lambda}$. This also bounds the distance of $g_q^{q_\rho}$ from uniform in G^q. The simulated value q'_ρ is uniformly chosen from a set of size B. Again by Fact 1, if $|G^q| < B$, then q'_ρ mod $|G^q|$ has a distance $|G^q|/B$ from uniform. The simulated value $g_q^{q'_\rho}$ has a distance at most $|G^q|/B$ from uniform in G^q. By the triangle inequality, the statistical distance of $g_q^{q_\rho}$ and $g_q^{q'_\rho}$ is at most $\frac{2^{\lambda+1}}{B} + \frac{2^{\lambda-1}|G^q|}{B+1-2^\lambda} + \frac{|G^q|}{B} < \frac{1}{2^{\epsilon_d+2}}$, if $B \geq 2^{\epsilon_d+2}(2^{\lambda-1} + 1)|G^q| + 2^{\lambda+\epsilon_d+3} + 2^\lambda - 1$. Similarly, the same argument holds for the distances of pk^{q_ρ} and $\mathsf{pk}^{q'_\rho}$. By using Fact 3, the distance between the joint distribution $X_\rho = (\mathsf{pk}^{q_\rho}, g_q^{q_\rho})$ and the simulated distribution $Y_\rho = (\mathsf{pk}^{q'_\rho}, g_q^{q'_\rho})$ is at most $\frac{1}{2^{\epsilon_d+1}}$.

Finally, we consider the joint distribution of $(\mathsf{pk}^{q_\rho}, g_q^{q_\rho})$ and r_ρ. Consider the conditional distribution of $q_\rho | r_\rho$. Note that $q_\rho = z$ if $(s_\rho - r_\rho)/\ell = z$. We repeat a similar argument as above for bounding the distribution of q_ρ from uniform. For each possible value of z, there always exists a unique value of s_ρ such that $\left\lfloor \frac{s_\rho}{\ell} \right\rfloor = z$ and $s_\rho = 0 \bmod \ell$, except possibly at the two endpoints E_1, E_2 of the range of q_ρ. When r_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 $\Pr(q_\rho = z | r_\rho)$ increases by at most $\frac{1}{|Y|} - \frac{\ell}{2B+1}$. The same applies to the variable $(\mathsf{pk}^{q_\rho}, g_q^{q_\rho}) | r_\rho$.

We can compare the joint distribution $X_\rho = (\mathsf{pk}^{q_\rho}, g_q^{q_\rho}, r_\rho)$ to the simulated distribution $Y_\rho = (\mathsf{pk}^{q'_\rho}, g_q^{q'_\rho}, r'_\rho)$ using Fact 3. Setting $\epsilon_1 = \frac{1}{|Y|} - \frac{\ell}{2B+1}$ and $\epsilon_2 = 0$, the distance between these joint distributions is at most $\frac{1}{2^{\epsilon_d+1}} + \frac{\ell}{2B+1} + \epsilon_1 \ell = \frac{1}{2^{\epsilon_d+1}} + \frac{\ell^2}{2B+3-3\ell} + \frac{\ell(1-\ell)}{2B+1} < \frac{1}{2^{\epsilon_d}}$, where the last equality holds if $B > 2^{\lambda+\epsilon_d} + 2^{\lambda+1} - 1$. This bounds the distance between (Q_1, Q_2, r_ρ) and (Q'_1, Q'_2, r'_ρ). Combining

Table 2: Comparison of communication size for ZK proof of the well-formedness of CL ciphertext.

	Communication Size (Bytes)				Additional Requirement
	$\lambda = 112$	$\lambda = 128$	$\lambda = 192$	$\lambda = 256$	
CCL+19 [4]	37970	49950	95520	156130	×
CCL+20 [5]	495	645	1214	1972	Pick random $g_q \in G^q$
This paper	1129	1488	2864	4692	$\mathsf{pk} \in G^q$, generic group model

the two requirements on B, we can simplify the requirement as $B \geq 2^{\lambda+\epsilon_d+2}\tilde{s}$. □

Comparison. We compare our scheme with the similar ZK proofs for the well-formedness of CL ciphertext in [4] and [5] in Table 2. We use the statistical distance 2^{-80} suggested for the CL encryption in [7]. We use the same statistical distance and soundness error of 2^{-80} for fair comparison.

We note that CCL+20 [5] required that the generator g_q is randomly chosen in G^q prior to running the zero knowledge proof. In order to achieve trustless setup, it should be jointly generated by all participating parties according to [5]. It introduces some overheads in bandwidth as well as a few more rounds of communication. In our scheme, we additionally require that $\mathsf{pk} \in G^q$. It can be proved by the owner of the secret key separately (e.g., §5.2 scheme 1), or can be embedded into this ZK proof if the prover himself is also the owner of the secret key (e.g., §5.1).

5 Applications to Threshold ECDSA and Two-Party ECDSA

5.1 Two-Party ECDSA

The two-party ECDSA in [4] used a ZK proof for the CL ciphertext with a slightly different relation. Suppose that $\hat{P}$ is a generator in $\mathcal{G}_{\mathrm{ECC}}$ included in the system parameter param. The two-party ECDSA in [4] used a ZK proof of plaintext and randomness used in the additive homomorphic encryption for the following relation:

$$\mathcal{R}_{\mathsf{EncECC}} = \{(m, \rho) : C_1 = f^m \mathsf{pk}^\rho \wedge C_2 = g_q^\rho \wedge \hat{Q} = \hat{P}^m\}.$$

This ZK proof is used in the interactive key generation $\mathsf{IKeyGen}$ phase of the two-party ECDSA.

For the relation $\mathcal{R}_{\mathsf{EncECC}}$, we cannot apply the protocol $\mathsf{ZKPoKRepS}$ directly since pk is not in the CRS. Moreover, pk may not be well-formed (e.g., $\mathsf{pk} = g_q^{\mathsf{sk}} f^\delta$ for some $\delta \in \mathbb{Z}_q$). Therefore, we change the relation to:

$$\mathcal{R}_{\mathsf{Enc}'} = \{(m, \rho, \mathsf{sk}) : C_1 = f^m \mathsf{pk}^\rho \wedge C_2 = g_q^\rho \wedge \hat{Q} = \hat{P}^m \wedge \mathsf{pk} = g_q^{\mathsf{sk}}\}.$$

It is because the knowledge of the secret key is known by the prover in the $\mathsf{IKeyGen}$ algorithm.

We propose a new ZK proof $\mathsf{ZKPoKEnc'}$ for $\mathcal{R}_{\mathsf{Enc'}}$ as shown in Algorithm 6. The security proofs are similar to the previous proof and are omitted due to the page limit.

Algorithm 6: Protocol $\mathsf{ZKPoKEnc'}$ for the relation $\mathcal{R}_{\mathsf{Enc'}}$

```
Param: $(\hat{G}, q, \hat{P}) \leftarrow \mathsf{GGen}_{\mathrm{ECC}}(1^\lambda)$, $\mathcal{G}_{\mathrm{HSM}} \leftarrow \mathsf{GGen}_{\mathrm{HSM},q}(1^\lambda)$, $B = 2^{\lambda+\epsilon_d+2}\tilde{s}$,
    $\epsilon_d = 80$.
Input: $C_1, C_2, \mathsf{pk} \in G, \hat{C} \in \hat{G}$.
Witness: $\rho, \mathsf{sk} \in [0, S], m \in \mathbb{Z}_q$, where $S = \tilde{s} \cdot 2^{\epsilon_d}$.
1 Prover chooses $s_\rho, s_k \xleftarrow{\$} [-B, B]$, $s_m \xleftarrow{\$} \mathbb{Z}_q$ and computes:
      $$S_1 = \mathsf{pk}^{s_\rho} f^{s_m}, \quad S_2 = g_q^{s_\rho}, \quad S_3 = g_q^{s_k}, \quad \hat{S} = \hat{P}^{s_m}.$$
  Prover sends $(S_1, S_2, S_3, \hat{S})$ to the verifier.
2 Verifier sends $c \xleftarrow{\$} [0, q-1]$ to the prover.
3 Prover computes:
      $$u_\rho = s_\rho + c\rho, \quad u_k = s_k + c \cdot \mathsf{sk}, \quad u_m = s_m + cm \mod q.$$
  Prover finds $d_\rho, d_k \in \mathbb{Z}$ and $e_\rho, e_k \in [0, q-1]$ s.t. $u_\rho = d_\rho q + e_\rho$ and
  $u_k = d_k q + e_k$. Prover computes:
      $$D_1 = \mathsf{pk}^{d_\rho}, \quad D_2 = g_q^{d_\rho}, \quad D_3 = g_q^{d_k}.$$
  Prover sends $(u_m, D_1, D_2, D_3, e_\rho, e_k)$ to the verifier.
4 The verifier checks if $e_\rho, e_k \in [0, q-1]$ and:
      $$\hat{S}\hat{C}^c = \hat{P}^{u_m}, \quad D_1^q \mathsf{pk}^{e_\rho} f^{u_m} = S_1 C_1^c, \quad D_2^q g_q^{e_\rho} = S_2 C_2^c, \quad D_3^q g_q^{e_k} = S_3 \mathsf{pk}^c.$$
  If so, the verifier sends $\ell \xleftarrow{\$} \mathsf{Primes}(\lambda)$.
5 Prover finds $q_\rho, q_k \in \mathbb{Z}$ and $r_\rho, r_k \in [0, \ell-1]$ s.t. $u_\rho = q_\rho \ell + r_\rho$ and
  $u_k = q_k \ell + r_k$. Prover computes:
      $$Q_1 = \mathsf{pk}^{q_\rho}, \quad Q_2 = g_q^{q_\rho}, \quad Q_3 = g_q^{q_k}.$$
  Prover sends $(Q_1, Q_2, Q_3, r_\rho, r_k)$ to the verifier.
6 Verifier accepts if $r_\rho, r_k \in [0, \ell-1]$ and:
      $$Q_1^\ell \mathsf{pk}^{r_\rho} f^{u_m} = S_1 C_1^c, \quad Q_2^\ell g_q^{r_\rho} = S_2 C_2^c, \quad Q_3^\ell g_q^{r_k} = S_3 \mathsf{pk}^c.$$
```

Evaluation. We follow the evaluation methodology in [5]. For each scheme, the bandwidth used is the sum of the total input and output transmission for a single party. A broadcast message is only counted as one transmission only. All interactive zero-knowledge proofs are turned into a non-interactive one using

Table 3: Comparison for two-party ECDSA with different security levels.

	Security Level	IKeyGen (Bytes)	ISign (Bytes)	Assumption
CCL+19 [4]	$\lambda = 112$	38714	575	Hard subgroup membership
	$\lambda = 128$	50876	697	
	$\lambda = 192$	97230	1260	
	$\lambda = 256$	158850	1973	
CCL+19-*lcm* [5]	$\lambda = 112$	4559	575	Hard subgroup membership
	$\lambda = 128$	5939	697	
	$\lambda = 192$	11280	1260	
	$\lambda = 256$	18351	1973	
Our two-party ECDSA	$\lambda = 112$	2453	575	Hard subgroup membership, adaptive root subgroup.
	$\lambda = 128$	3173	697	
	$\lambda = 192$	6030	1260	
	$\lambda = 256$	9789	1973	

the Fiat-Shamir transformation, such that the commit message can be omitted if possible.

We compare our scheme with the two-party ECDSA scheme in [4], which used a binary challenge in the ZK proof in the IKeyGen algorithm. As a result, the ZK proof has to be repeated for ϵ_s times for soundness error of $2^{-\epsilon_s}$. Recently, [5] proposed an *lcm* trick (CCL+19-*lcm*) to replace a binary challenge with a challenge of 10 bits. Hence, the ZK proof has to be repeated for $\epsilon_s/10$ times. However, the relationship proved by the *lcm* trick is changed slightly, and hence the prover and the verifier have to additionally compute exponentiation of $y = lcm(1, \ldots, 2^{10})$, which is a 1479 bits integer, in IKeyGen and ISign respectively.

In our scheme, we only need to run the ZK proof for one time only and no extra exponentiation is needed. The comparison of communication size is shown in Table 3. For a soundness error and statistical distance of 2^{-80}, CCL+19 [4] needs at least 10 times more bandwidth in IKeyGen than ours, while CCL+19-*lcm* [5] needs about twice the bandwidth in IKeyGen than ours. Our scheme additionally relies on the adaptive root subgroup assumption in the generic group model. Note that the security of ECDSA is based on the DL assumption in the generic group model [2].

5.2 Threshold ECDSA

Our proposed ZK proofs can be used to improve the state-of-the-art bandwidth efficient threshold ECDSA CCL+20 [5]. We give two threshold ECDSA schemes in this section. Our scheme 1 reduces the communication cost of IKeyGen in CCL+20 and also reduces the computation time in both IKeyGen and ISign, at the price of having a larger communication cost for ISign. Our scheme 2 outperforms CCL+20 [5] in the communication cost and computation time in IKeyGen, while having the same performance in ISign.

Our Scheme 1. We show how to use our protocols ZKPoKRepS and ZKPoKEnc to build a threshold ECDSA with fast trustless setup from the scheme in [5]. There are three main differences in the protocol (shown in Fig. 4):

1. IKeyGen:
   (a) We do not need to run the interactive ISetup algorithm in [5] to generate the generator g_q used in IKeyGen.
   (b) One of the main differences between our ZKPoKEnc protocol and with the argument of knowledge for CL ciphertext in [5] is that our ZKPoKEnc protocol requires that the public key pk is well-formed. This can be achieved by adding a zero-knowledge proof of the secret key sk with respect to pk in the key generation phase.
2. ISign: For the interactive signing phase, we only need to modify phase 1 of the signing protocol in [5]. All other phases remain the same.

The resulting scheme 1 is secure in the generic group model by assuming the hardness of the hard subgroup membership and the adaptive root subgroup assumption.

Our Scheme 2. If we make the extra adaptive root subgroup assumption, we can keep the ISign algorithm and the most of the IKeyGen algorithm in CCL+20 [5]. We only need to modify the interactive ISetup algorithm in [5], such that the proof of knowledge of t_i for $g_i = g_q^{t_i}$ is replaced by our ZKPoKRepS protocol. The resulting scheme is the most bandwidth efficient for the total bandwidth used in the IKeyGen and the ISign algorithms, at the price of using one more assumption.

Evaluation. We compare our schemes with the state-of-the-art bandwidth efficient threshold ECDSA scheme [5] in table 5. The total number of party is n and the threshold is t.

The most common threshold signature P2SH transaction of Bitcoin is the case of (t, n) = (1,3), (2,4) and (2,5). By using this parameter, our scheme 1 is the most bandwidth efficient for the IKeyGen algorithm and it is about 69-74% less than CCL+20. However, the bandwidth of the ISign algorithm of CCL+20 and our scheme 2 is 20-22% less than our scheme 1. Our scheme 2 uses 59-65% less bandwidth than [5] in IKeyGen (as shown in Fig. 2), with the same bandwidth in ISign. Our schemes are proved secure in the generic group model. Note that the security of ECDSA is based on the DL assumption in the generic group model [2].

6 Implementation

Choices of Parameters. Various security parameters are used for soundness error and statistical distance in different threshold ECDSA papers which makes it difficult to compare the efficiency of different schemes. Lindell17 [14] used 2^{-40} for soundness error and statistical distance. LN18 [15] used 2^{-80} for these parameters. CCL+19 [4] followed [14] to use 2^{-40} for comparison, but they

Table 4: Scheme 1: Modifications to the threshold ECDSA in [5] are shown in the box.

	$\mathsf{IKeyGen}(\mathsf{param})$	
P_i		All players $\{P_j\}_{j\neq i}$
$u_i \xleftarrow{\$} \mathbb{Z}_q$		
$(\mathsf{kgc}_i, \mathsf{kgd}_i) \leftarrow \mathsf{Com}(\hat{P}^{u_i})$		
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{CL.KeyGen}()$	$\xrightarrow{\mathsf{kgc}_i, \mathsf{pk}_i}$	
	$\xrightarrow{\mathsf{kgd}_i}$	
$\boxed{\pi_k := \mathsf{ZKPoKRepS}(\mathsf{pk}_i; \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i})}$	$\boxed{\xleftrightarrow{\pi_k}}$	$\boxed{\text{Abort if the proof fails.}}$
Follow from line 5 of Fig. 4 in in [5].		

	$\mathsf{ISign}(\mathsf{param}, m)$	
P_i	Phase 1	All players $\{P_j\}_{j\neq i}$
$k_i, \gamma_i \xleftarrow{\$} \mathbb{Z}_q$, $r_i \xleftarrow{\$} [0, S]$		
$(\mathsf{c}_i, \mathsf{d}_i) \leftarrow \mathsf{Com}(\hat{P}^{\gamma_i})$		
$C_{k_i} \leftarrow \mathsf{CL.Enc}(\mathsf{pk}_i, k_i; r_i)$	$\xrightarrow{C_{k_i}, \mathsf{c}_i}$	
$\boxed{\pi_C := \mathsf{ZKPoKEnc}((k_i, r_i) :}$		
$\boxed{((\mathsf{pk}_i, C_{k_i}); (k_i, r_i)) \in \mathcal{R}_{\mathsf{Enc}})}$	$\boxed{\xleftrightarrow{\pi_C}}$	$\boxed{\text{Abort if the proof fails.}}$

suggested to use 2^{-60} in practice. GG18 [11] and GG20 [12] used a soundness error of 2^{-q} and a statistical distance of $2^{-\lambda}$. CCL+20 [5] used $2^{-\lambda}$ for soundness error and 2^{-40} for statistical distance. For the two-party and threshold ECDSA based on oblivious transfer, DKLs18 [9] and DKLs19 [10] used 2^{-80} for statistical distance. In addition, the CL encryption [7] proposed to use 2^{-80} for statistical distance. In this paper, we take the middle ground of using 2^{-80} for soundness error and statistical distance for the ZK proofs as well as the CL encryption.

We only implement the schemes with 112-bit and 128-bit security due to the constraint in running time (it takes > 66 seconds to run the $\mathsf{IKeyGen}$ of CCL+19 for 128-bit security). We use the secp256k1 curve.

Testing Environment. We implemented our schemes, CCL+19 [4] and CCL+20 [5] using Rust. We tested the program in a MacBook with Intel Core i5 1.4GHz, 16GB RAM. The results are the median running time for running > 100 times. The program is implemented in one single thread for comparing different settings.

During the testing, we do not consider the network conditions. We may further outperform existing schemes in terms of running time since our schemes use a smaller bandwidth.

6.1 Two-party ECDSA

We show the running time for both $\mathsf{IKeyGen}$ and ISign for 112-bit and 128-bit security level in Figure 3. As compared with the CCL+19-*lcm* in [5], the running

Table 5: Comparison for threshold ECDSA with different security levels.

	Security Level	IKeyGen (Bytes)	ISign (Bytes)	Assumption
CCL+20 [5]	$\lambda = 112$	$32tn + 2692n - 64$	$2397t - 1412$	Hard subgroup membership Strong root subgroup
	$\lambda = 128$	$32tn + 3479n - 64$	$3100t - 1891$	
	$\lambda = 192$	$32tn + 6518n - 96$	$5862t - 3694$	
	$\lambda = 256$	$32tn + 10535n - 128$	$9489t - 6099$	
Our threshold ECDSA scheme 1	$\lambda = 112$	$32tn + 797n - 64$	$3031t - 1412$	Hard subgroup membership Adaptive root subgroup
	$\lambda = 128$	$32tn + 979n - 64$	$3944t - 1891$	
	$\lambda = 192$	$32tn + 1779n - 96$	$7512t - 3694$	
	$\lambda = 256$	$32tn + 2805n - 128$	$12210t - 6099$	
Our threshold ECDSA scheme 2	$\lambda = 112$	$32tn + 1072n - 64$	$2397t - 1412$	Hard subgroup membership Adaptive root subgroup Strong root subgroup
	$\lambda = 128$	$32tn + 1319n - 64$	$3100t - 1891$	
	$\lambda = 192$	$32tn + 2397n - 96$	$5862t - 3694$	
	$\lambda = 256$	$32tn + 3775n - 128$	$9489t - 6099$	

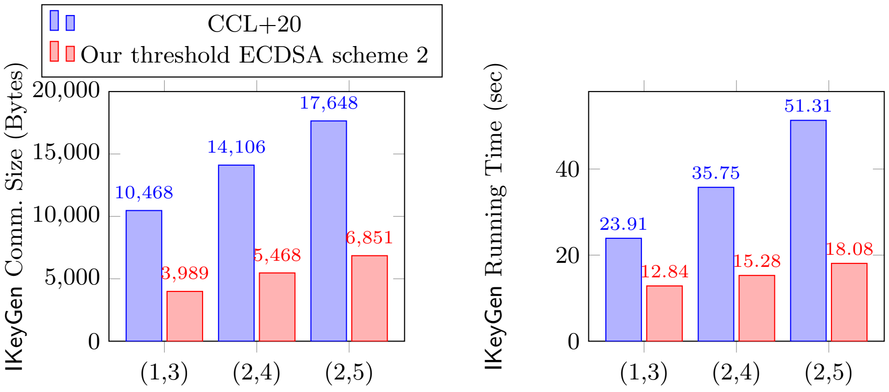


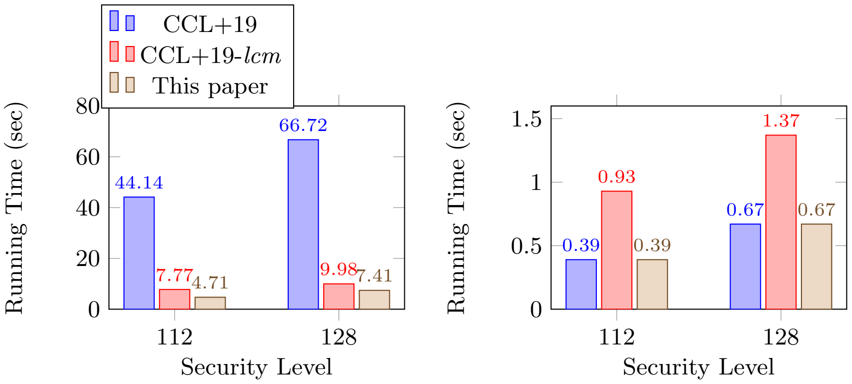
Fig. 2: (t, n)-Threshold ECDSA with 128-bit security.

time of our IKeyGen is 35-65% times faster, and the running time of our ISign is 104-138% times faster.

In particular, the *lcm* trick has a non-obvious cost of doubling the running of ISign as compared with CCL+19 [4] and our scheme. The prover and the verifier have to additionally compute exponentiation of $y = lcm(1, \ldots, 2^{10})$, which is a 1479 bits integer, in IKeyGen and ISign respectively. It takes about 0.6 seconds and significantly affects the performance in ISign.

6.2 Threshold ECDSA

We show the running time for both IKeyGen and ISign for 112-bit and 128-bit security level in Figure 4. As compared with CCL+20 [5], the running time of our scheme 1 is 85-90% faster in IKeyGen, with the price of a higher communication cost in ISign. If one wants to minimize the communication cost, our scheme 2

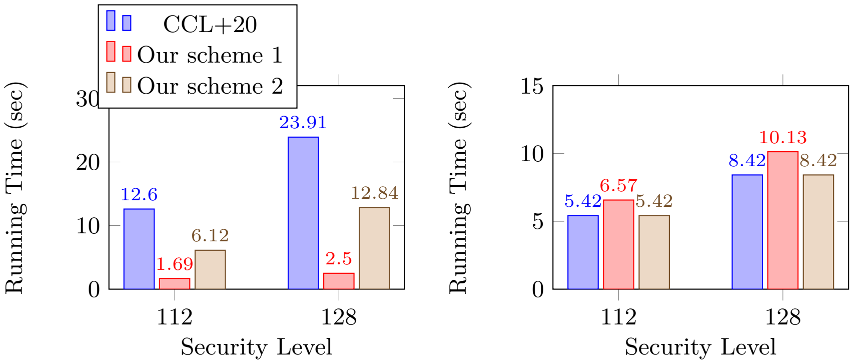


(a) IKeyGen

(b) ISign

Fig. 3: Running time of two-party ECDSA.

is still 46-65% faster in IKeyGen. For the running time in ISign, our scheme 1 is slightly slower than scheme 2/CCL+20.



(a) IKeyGen.

(b) ISign.

Fig. 4: Running time of threshold ECDSA with $t = 1, n = 3$.

7 Conclusion

In this paper, we propose a compact zero-knowledge proof for the DL relation in HSM groups and the CL ciphertext. When applied to two-party ECDSA and threshold ECDSA, it can significantly improve the performance in terms of bandwidth used in IKeyGen, and the running time of IKeyGen and ISign.

References

1. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to iops and stateless blockchains. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Lecture Notes in Computer Science, vol. 11692, pp. 561–586. Springer (2019)
2. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Des. Codes Cryptogr. **35**(1), 119–152 (2005)
3. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs. In: Joux [13], pp. 425–442
4. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Lecture Notes in Computer Science, vol. 11694, pp. 191–221. Springer (2019)
5. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DSA. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. Lecture Notes in Computer Science, vol. 12111, pp. 266–296. Springer (2020)
6. Castagnos, G., Laguillaumie, F.: On the security of cryptosystems with quadratic decryption: The nicest cryptanalysis. In: Joux [13], pp. 260–277
7. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. Lecture Notes in Computer Science, vol. 9048, pp. 487–505. Springer (2015)
8. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 256–271. Springer (2002)
9. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Secure two-party threshold ECDSA from ECDSA assumptions. In: IEEE SP 2018. pp. 980–997. IEEE Computer Society (2018)
10. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: IEEE SP 2019. pp. 1051–1066. IEEE (2019)
11. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) CCS 2018. pp. 1179–1194. ACM (2018)
12. Gennaro, R., Goldfeder, S.: One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive, Report 2020/540 (2020), https://eprint.iacr.org/2020/540
13. Joux, A. (ed.): EUROCRYPT 2009, Lecture Notes in Computer Science, vol. 5479. Springer (2009)
14. Lindell, Y.: Fast secure two-party ECDSA signing. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. Lecture Notes in Computer Science, vol. 10402, pp. 613–644. Springer (2017)
15. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) CCS 2018. pp. 1837–1854. ACM (2018)
16. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT '97. Lecture Notes in Computer Science, vol. 1233, pp. 256–266. Springer (1997)

A Background

A.1 Definitions for Argument Systems

An argument system for a relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ is a triple of randomized polynomial time algorithms $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$, where:

- Setup takes a security parameter 1^λ and outputs a common reference string (CRS) crs.
- P takes as input the crs, a statement $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$. V takes as input the crs and x and after interaction with P outputs 0 or 1.

The transcript between the prover and the verifier is denoted as $\langle \mathsf{V}(\mathsf{crs}, x), \mathsf{P}(\mathsf{crs}, x, w)\rangle$, and it is equal to 1 if the verifier accepted the transcript.

Definition 1 (Completeness). *An argument system* $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *for a relation* $\mathcal{R}$ *is complete if for all* $(x, w) \in \mathcal{R}$*:*

$$\Pr[\langle \mathsf{V}(\mathsf{crs}, x), \mathsf{P}(\mathsf{crs}, x, w)\rangle = 1 : \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)] = 1.$$

We follow the soundness definition for trapdoorless crs from [1].

Definition 2 (Soundness). *An argument system* $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *is sound if for all polynomial time adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*:*

$$\Pr\left[\begin{array}{l} \langle \mathsf{V}(\mathsf{crs}, x), \mathcal{A}_1(\mathsf{crs}, x, \mathsf{state})\rangle = 1 \\ \textit{and } \nexists w \textit{ s.t. } (x, w) \in \mathcal{R} \end{array} : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{state}) \leftarrow \mathcal{A}_0(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Additionally, the argument system is an argument of knowledge if for all polynomial time adversaries $\mathcal{A}_1$ *there exists a polynomial time extractor* Ext *such that for all polynomial time adversaries* $\mathcal{A}_0$*:*

$$\Pr\left[\begin{array}{l} \langle \mathsf{V}(\mathsf{crs}, x), \mathcal{A}_1(\mathsf{crs}, x, \mathsf{state})\rangle = 1 \\ \textit{and } (x, w') \notin \mathcal{R} \end{array} : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (x, \mathsf{state}) \leftarrow \mathcal{A}_0(\mathsf{crs}) \\ w' \xleftarrow{\$} \mathsf{Ext}(\mathsf{crs}, x, \mathsf{state}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Definition 3 (Zero Knowledge). *An argument system* $(\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *is statistical zero-knowledge if there exists a polynomial time simulator* Sim *such that for all* $(x, w) \in \mathcal{R}$*, the following two distributions are statistically indistinguishable:*

$$D_1 = \{\langle \mathsf{V}(\mathsf{crs}, x), \mathsf{P}(\mathsf{crs}, x, w)\rangle, \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)\},$$

$$D_2 = \{\langle \mathsf{V}(\mathsf{crs}, x), \mathsf{Sim}(\mathsf{crs}, x)\rangle, \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)\}.$$

A.2 Generalized Schnorr Proofs

The Sigma protocol based on Schnorr's proof can be generalized for proving DL in groups of unknown order [3]. It can be done by introducing appropriate range checking and using computations over the integers. The proof size is dominated by the response of size $(\epsilon_s + \epsilon_d) \cdot \mathrm{ord}(G)$, and hence it is not practical. By taking $\epsilon_s = \epsilon_d = 80$, the proof size is in the order of MBytes.

Table 6: $(2, 4)$-Threshold ECDSA for 112 bit security

	IKeyGen		ISign	
	Communication Size (bytes)	Running Time (sec)	Communication Size (bytes)	Running Time (sec)
CCL+20	10958	19.34	3382	10.23
Our Threshold Scheme 1	3380	2.82	4650	11.70
Our Threshold Scheme 2	4478	8.33	3382	10.23

Table 7: $(2, 4)$-Threshold ECDSA for 128 bit security

	IKeyGen		ISign	
	Communication Size (bytes)	Running Time (sec)	Communication Size (bytes)	Running Time (sec)
CCL+20	14106	35.75	4308	15.75
Our Threshold Scheme 1	4107	4.11	4650	17.54
Our Threshold Scheme 2	5468	15.28	4308	15.75

B More Implementation Results for Threshold ECDSA

We also implemented the threshold ECDSA schemes for the setting of (t, n) = (2,4) and (2,5) and the security level of 112-bit and 128-bit. The complete comparison tables for these cases are given in Table 6 to 9.

Table 8: $(2,5)$-Threshold ECDSA for 112 bit security

	IKeyGen		ISign	
	Communication Size (bytes)	Running Time (sec)	Communication Size (bytes)	Running Time (sec)
CCL+20	13714	28.70	3382	16.45
Our Threshold Scheme 1	4241	4.25	4650	18.28
Our Threshold Scheme 2	5614	11.00	3382	16.45

Table 9: $(2,5)$-Threshold ECDSA for 128 bit security

	IKeyGen		ISign	
	Communication Size (bytes)	Running Time (sec)	Communication Size (bytes)	Running Time (sec)
CCL+20	17468	51.31	4308	25.51
Our Threshold Scheme 1	5150	6.10	4650	27.23
Our Threshold Scheme 2	6851	18.08	4308	25.51