# Efficient Multiplicative-to-Additive Function from Joye-Libert Cryptosystem and Its Application to Threshold ECDSA

Haiyang Xue
The Hong Kong Polytechnic University
haiyangxc@gmail.com

Man Ho Au
The Hong Kong Polytechnic University
man-ho-allen.au@polyu.edu.hk

Mengling Liu
The Hong Kong Polytechnic University
mengling.liu@connect.polyu.hk

Kwan Yin Chan
The University of Hong Kong
kychan@cs.hku.hk

Handong Cui
The University of Hong Kong
hdcui@cs.hku.hk

Xiang Xie
Shanghai Qizhi Institute
PADO Labs
xiexiangiscas@gmail.com

Tsz Hon Yuen
The University of Hong Kong
thyuen@cs.hku.hk

Chengru Zhang
The University of Hong Kong
u3008875@connect.hku.hk

## ABSTRACT

Threshold ECDSA receives interest lately due to its widespread adoption in blockchain applications. A common building block of all leading constructions involves a secure conversion of multiplicative shares into additive ones, which is called the multiplicative-to-additive (MtA) function. MtA dominates the overall complexity of all existing threshold ECDSA constructions. Specifically, $O(n^2)$ invocations of MtA are required in the case of $n$ active signers. Hence, improvement of MtA leads directly to significant improvements for all state-of-the-art threshold ECDSA schemes.

In this paper, we design a novel MtA by revisiting the Joye-Libert (JL) cryptosystem. Specifically, we revisit JL encryption and propose a JL-based commitment, then give efficient zero-knowledge proofs for JL cryptosystem which are the first to have standard soundness. Our new MtA offers the best time-space complexity trade-off among all existing MtA constructions. It outperforms state-of-the-art constructions from Paillier by a factor of 1.85 to 2 in bandwidth and 1.2 to 1.7 in computation. It is 7× faster than those based on Castagnos-Laguillaumie encryption only at the cost of 2× more bandwidth. While our MtA is slower than OT-based constructions, it saves 18.7× in bandwidth requirement. In addition, we also design a batch version of MtA to further reduce the amortised time and space cost by another 25%.

## KEYWORDS

Multiplicative-to-Additive function; Joye-Libert cryptosystem; Threshold ECDSA; Zero-knowledge proof

## 1 INTRODUCTION

**Threshold ECDSA.** Threshold signature [24] distributes signing power among $n$ participants in such a way that a message can be signed if and only if $t + 1$ or more participants agree to do so. Elliptic Curve Digital Signature Algorithm (ECDSA) [19] is a standardised digital signature scheme adopted widely in blockchain and cryptocurrency applications. Due to the urgent need of private key protection mechanism in blockchain applications, threshold version of ECDSA draws huge attention from not only the academia but also the industry. Its practical significance can also be seen from the recent initiation from the National Institute of Standards and Technology (NIST) [38], which calls for proposals of threshold ECDSA.

However, ECDSA is widely perceived as "threshold-unfriendly". Specifically, it involves computing

$$s = k^{-1}(H(m) + xr) \mod q,$$

where $k$ is the secret nonce, $x$ is the secret key, and $r$ is the public nonce. The challenge of designing threshold ECDSA lies in the computation of $s$ in a distributed way from shares of $k$ and $x$ among the participants. That is, computing (additive) shares of $k^{-1}$ and $k^{-1}x$ from secret shares of $k$ and $x$. In recent years, a number of new threshold ECDSA protocols have been proposed [10, 12, 25, 26, 29, 35, 44]. At a high level, they mostly involve constructing new variants of *multiplicative-to-additive* functionality (denoted as MtA hereafter). Roughly speaking, a MtA functionality is a secure two-party computation that takes as inputs $a$ and $b$ from two parties, and securely computes $\alpha$ and $\beta$ such that $\alpha + \beta = ab \mod q$. With the help of MtA, shares of $k$, $x$ could be transformed to additive shares of $k^{-1}$ and $k^{-1}x$, and further into the additive shares of $s$.

**MtA.** Existing MtAs can be classified into three categories according to the cryptographic tools they are based on. These tools are Paillier encryption (e.g. [10, 29, 35, 44]), Castagnos-Laguillaumie encryption (CL, e.g. [12]), and oblivious transfer (OT, e.g.,[25, 26]). Table 1 presents a summary of their performance. CL-based MtA [12, 23] has the lowest bandwidth (less than 2KB), while it is computationally heavy (>1000ms). Those from OT [25, 26] are excellent in terms of computation cost and enjoy the added advantage that no extra assumptions are needed. While their computational cost is very low, OT-based MtAs [26] require ≈ 90KB of bandwidth. Those based on Paillier [10, 29, 35, 44] are the most popular ones and are preferred by the industry [43], due to their better overall performance (i.e., they offer a better trade-off between computation and communication complexity).

We briefly review how a MtA can be realised using an additively homomorphic encryption, such as Paillier [40]. Ciphertexts of Paillier lies in $\mathbb{Z}_{N^2}$ (with RSA modulus $N$ and message space $\mathbb{Z}_N$)

such that $\mathsf{Enc}(x_1) \oplus \mathsf{Enc}(x_2) = \mathsf{Enc}(x_1 + x_2)$, and $a \odot \mathsf{Enc}(x) = \mathsf{Enc}(ax)$. Paillier-based MtA is roughly built as follows: participant Bob with private input $b$ computes $\mathsf{Enc}(b)$ under his public key, and sends it to participant Alice. Alice with private input $a$ picks a random $\alpha$, computes $a \odot \mathsf{Enc}(b) \oplus \mathsf{Enc}(-\alpha)$ and sends it back. Bob decrypts the ciphertext and sets it as $\beta$. The output of Alice and Bob are $\alpha$ and $\beta$ respectively. It is easy to see that $\alpha + \beta = ab$. There are additional subtleties involved. Notably, due to the mismatch between Paillier's message space and ECDSA's signature space, zero-knowledge proofs (e.g. range proof) should be added to prevent malicious behaviors (such as attacks presented in [42]).

Typically, threshold ECDSA involving $n$ parties to sign makes $O(n^2)$ calls to MtA, making MtA the most dominating factor of the overall complexity of these schemes. As such, it is highly desirable to develop efficient MtA since its improvements translate directly to performance gains in many threshold ECDSA schemes. This is also why NIST [39, page 25] intends to call for MtAs as important building block. Although Paillier-based MtA has the best trade-off among existing constructions, it is still relatively expensive (compared with the cost of signing). A single Paillier-based MtA requires a bandwidth of at least $20 \log N$ bits and computation of 23 full exponentiations modulo $N$ (refer to Table 1). We observe that some of the cost is "wasted": the message space ($\mathbb{Z}_N$, 3072-bit) of Paillier is typically much larger than that of the signature space of ECDSA (256-bit). Furthermore, some operations (e.g., zero-knowledge proofs) runs in $\mathbb{Z}_{N^2}$, which is relatively expensive.

**Our Idea.** A natural idea is to look for a more efficient additively homomorphic encryption. We identified the Joye-Libert (JL) cryptosystem [31] as a suitable candidate. With a message space of $k$-bit, JL works directly over $\mathbb{Z}_N$ for special RSA modulus $N = (2^k p' + 1)(2q' + 1)$. Its security relies on the $k$ quadratic residuosity ($k$-QR) assumption, which is the standard QR assumption under the special RSA modulus. The advantage of JL over Paillier in our quest for an efficient MtA and threshold ECDSA is clear: it is more efficient to work in $\mathbb{Z}_N$ over $\mathbb{Z}_{N^2}$. The trade-off of reducing message space from $\log N$-bit to $k$-bit is acceptable since the signature space of ECDSA is much smaller.

However, instantiating such an idea is challenging. For instance, very little is known about efficient zero-knowledge proofs for JL (e.g., correctness of encryption, range proof to deal with the mismatch between the plaintext space and the ECDSA signature space). There is not even a standard zero-knowledge proof of knowing the plaintext in a ciphertext. Current state-of-the-art [14] only provides non-standard soundness (i.e., it says nothing to the most significant bits of the plaintext). Details are given in Sec. 1.2.

Motivated by the need of improving efficiency of MtA (and threshold ECDSA), this paper investigates the following problems: *Could we design more efficient MtA (and threshold ECDSA) by replacing Paillier with Joye-Libert? What exactly could we gain from this replacement?*

## 1.1 Our contributions

We give a modified JL encryption and a JL commitment, and propose related zero-knowledge proofs. Built on these primitives, we design a novel MtA protocol, which outperforms state-of-the-art MtA based on Paillier. Further, we develop a batching technique which further improves the amortised cost by 25%, making it much more efficient than Paillier-based constructions when multiple MtAs are executed in batch. Applying our MtA to existing threshold ECDSAs gives similar improvement in efficiency.

(1) We revisit the JL encryption and propose a variant, modified JL, which is zero-knowledge friendly without affecting its security. We propose a JL-based commitment scheme satisfying the following properties. a) Its security relies on the strong JL assumption which we prove to hold under the standard $k$-QR assumption and the strong RSA assumption. b) JL commitment can be publicly computed from a modified JL ciphertext by raising to the power of $2^k$. c) JL commitment can be easily extended to commit a vector (while maintaining the size of the commitment). Properties b) and c) help to gain savings in our MtA protocol.

(2) We design the first zero-knowledge proofs with standard soundness for JL cryptosystem, including proof for JL (vector) commitment, proof for JL encryption and affine operation, proof of equality between the encrypted value in modified JL ciphertext and that committed in JL commitment. It is one of our main technical contributions to prove the standard soundness of these proofs.

(3) With all the above building block in place, we build a JL-based MtA and its batch version with less commitments and zero-knowledge proofs. We benchmark our MtA in Rust and compare it with those based on Paillier, CL encryption and OT. Bandwidth of our MtA improves that based on Paillier by a factor of 1.85 to 2. The running time of our MtA outperforms that from Paillier by a factor of 1.2 to 1.7 depending on security levels (i.e., security parameter $\lambda = 128, 192, 256$). When batching is applied (e.g. batching $> 10$ MtAs), our improvement in bandwidth goes up further to a factor of 2.46 to 2.7, and the computational complexity of our MtA outperforms that of Paillier by a factor of 1.62 to 2.26. Details are given in Table 1, 4, and Figure 4.

(4) We also apply our MtA to threshold ECDSA of LN18 [35] and XAX+21 [44] by replacing Paillier-based MtA and give an efficiency comparison with OT-based, CL-based and Paillier-based schemes. Since MtA dominates their overall complexity, new threshold ECDSA schemes outperform Paillier-based schemes by a similar factor. Details are shown in Table 6.

(5) We would like to remark that our zero-knowledge proofs for JL cryptosystem and JL-based MtA have many other applications. JL encryption with range proof could be used to replace Paillier encryption with range proof in several scenarios, e.g, voting schemes [20], Naor-Yung CCA secure encryption [37]. Our MtA could be used to building more efficient three-party TLS handshake [46].

## 1.2 Technical overview

Figure 1 depicts the construction of MtA between $P_1$ and $P_2$ from additively homomorphic encryption (with Enc, Dec as encryption and decryption algorithms), assuming the message space is much larger than the input/output space (i.e., $\mathbb{Z}_q$). We denote by $C_1 \oplus C_2$

| $P_1$ $(pk_2; a \in \mathbb{Z}_q)$ | | $P_2$ $(pk_2, sk_2; b \in \mathbb{Z}_q)$ |
|---|---|---|
| | | $C_2 = \mathsf{Enc}(pk_2, b)$, $c_2 = \mathsf{Commit}(b)$ |
| Check $\pi_2$ | $\xleftarrow{C_2,\ c_2,\ \pi_2}$ | $\pi_2 = \mathsf{ZK} : \{(C_2, c_2; b) \mid b \in [0, q]\}$ |
| $C_1 = a \odot C_2 \oplus \mathsf{Enc}(pk_2, \alpha) = \mathsf{Enc}(pk_2, ab + \alpha)$ | | |
| $c_{11} = \mathsf{Commit}(a), c_{12} = \mathsf{Commit}(\alpha)$ | | |
| $\pi_1 = \mathsf{ZK} : \{(C_1, C_2, c_{11}, c_{12}; a, \alpha) \mid a \in [0, q], \alpha \in [0, K]\}$ | $\xrightarrow{C_1,\ c_{11},\ c_{12},\ \pi_1}$ | Check $\pi_1$ |
| $\alpha = -\alpha \mod q$ | | $\beta = \mathsf{Dec}(sk_2, C_1) \mod q$ |

**Figure 1: Illustration of semi-honest MtA (without the gray boxes), Paillier-based MtA (including the gray boxes) against malicious adversary, and our JL-based MtA (including the gray boxes but without the red values) against malicious adversary.**

| MtA Schemes | Communication | Computation |
|---|---|---|
| OT [25, 26] | $32\lambda^2 + O(\lambda)$ | $2\lambda$M |
| CL [12] | $84\lambda$ | 11E |
| Paillier [10, 29, 35] | $20N_{\text{Pai}}$ | 23E |
| JL<br>Batch JL | $10N_{\text{JL}}$<br>$(7 + 3/l)N_{\text{JL}}$ | 11E<br>$(8 + 3/l)$E |

**Table 1: Cost comparison of Multiplication phase in MtA. $\lambda$ is the security parameter. E represents a full exponentiation operation over $\mathbb{Z}_{N_{\text{Pai}}}$ in Paillier (one Paillier operation $\approx 2$E), a full exponentiation operation over $\mathbb{Z}_{N_{\text{JL}}}$ in JL, or in CL an exponentiation over CL group. M refers to the elliptic curve point multiplication. The cost in "Batch JL" is the average cost per MtA when batching $l$ MtAs.**

the addition of the plaintext in ciphertexts $C_1, C_2$, and by $a \odot C$ the multiplication of the plaintext in ciphertext $C$ by a scalar $a$.

Correctness of the semi-honest MtA is obvious. Let $a, b$ be the private inputs of $P_1$ and $P_2$ respectively. With randomness $\alpha$ and ciphertext $\mathsf{Enc}(pk_2, b)$ given by $P_2$, $P_1$ could compute ciphertext $\mathsf{Enc}(pk_2, ab + \alpha)$ using additively homomorphic property. At last, $P_2$ decrypts the ciphertext as $\beta$. They set $-\alpha \mod q$ and $\beta \mod q$ as outputs respectively. It is easy to check that $-\alpha + \beta = ab \mod q$.

The main challenge lies in upgrading the semi-honest protocol against malicious $P_1$ or $P_2$ since a malicious participant may extract the secret value by maliciously crafting the ciphertext (as in Alpha-rays attack [42]). This can be prevented by introducing zero-knowledge proofs. Specifically, we require the participants to prove the correct generation of $C_2$, $C_1$ and, especially, that $b$, $a$ and $\alpha$ lie in a proper range.

However, designing zero-knowledge proof on the range of encrypted value is not an easy task, especially when the public key is generated by the prover itself. Initiated by [8, 20], and later used by [10, 35], a general approach is appending integer commitment and transferring proof on the ciphertext to the commitment. We first commit to the encrypted value, then, prove equality between the encrypted value and committed value. At last, an efficient range proof (e.g., [8]) is performed on the committed value. In the following we use Paillier-based MtA to discuss the required commitment and zero-knowledge proofs in details.

**Paillier-Based MtA [35].** As illustrated in Figure 1, to guarantee the range of $b$, $a$ and $\alpha$, participants additionally send their RSA commitments [28][1] i.e., $\mathsf{Commit}(b)$, $\mathsf{Commit}(a)$, and $\mathsf{Commit}(\alpha)$. Zero-knowledge proofs (i.e., $\pi_1, \pi_2$) are then applied to these commitments and ciphertexts to guarantee the equality of plaintext and commited value, and that $b, a, \alpha$ lie in a proper range. However, these commitments and proofs are costly and contribute to the cost of Paillier-based MtA as listed in Table 1.

**Our Idea: Replacing Paillier with Joye-Libert.** We start from a simple observation that we do not need the large message space of Paillier. Joye-Libert (JL) encryption [31], on the other hand, has a message space of $\{0, 1\}^k$ and operates on $\mathbb{Z}_N$ where $N = (2^k p' + 1)(2q' + 1)$, which appears to suit our needs. In more details, the JL cryptosystem operates as follows. In addition to $N$, the public key also includes $y \in \mathbb{Z}_N$, a quadratic non-residue. JL scheme encrypts any message $m \in \{0, 1\}^k$ using randomness $r \in \mathbb{Z}_N^*$ as

$$C = y^m r^{2^k} \mod N.$$

The remaining challenge is to design commitments and appropriate zero-knowledge proofs to ensure that the ciphertexts are well-formed. This is, however, non-trivial. Let us take the basic case of proving knowledge of plaintext $m$ in ciphertext $C$ using a Schnorr-like protocol as an example. The prover starts by sending a JL encryption $A$ of random plaintext $v$. On receiving challenge $e \in \{0, 1\}^t$, the prover returns $z_m = em + v \mod 2^k$ and $z_r$ as respond such that $C^e A = y^{z_m} z_r^{2^k} \mod N$. While this adapted Schnorr-like protocol possesses correctness and (honest verifier) zero-knowledgeness, it is unclear if soundness holds or not (i.e., it is unknown if an extractor exists or not). More concretely, given two accepting transcripts $(A; e; z_m, z_r)$, $(A; e'; z'_m, z'_r)$, one could compute $C^{e-e'} = y^{z_m - z'_m} \tilde{r}^{2^k} \mod N$ for some $\tilde{r}$. However, it only guarantees $(e - e')m = z_m - z'_m \mod 2^k$. We cannot always extract $m$ since $e - e'$ is very likely to be non-invertible over $\mathbb{Z}_{2^k}$.

State-of-the-art zero-knowledge proof for JL cyrptosystem [14] circumvents this by allowing non-standard soundness, i.e., the extractor only extract the least significant $k - t$ bits of $m$, where $t$ is the

[1] An integer commitment scheme which adapts Pedersen commitment [41] to an RSA group of unknown order. Its security relies on the strong RSA assumption. It is also known as Ring-Pedersen or Fujisaki-Okamoto commitment in the literature.

length of $e$, i.e., the soundness parameter, for their applications. Non-standard soundness, however, is insufficient in our case as it does not prevent an attacker from generating a malformed ciphertext.

**Our solution.** We solve this by proposing a JL commitment which could be publicly computed from a modified JL encryption, and designing zero-knowledge proofs for JL cryptosystem with standard soundness.

*Modified JL encryption and JL commitment.* We first modify the JL encryption, and give an equivalent scheme that is commitment and zero-knowledge friendly. Concretely, let $h$ be the generator of the $2^k$-th power residue, i.e., $h = x^{2^k}$ for a random quadratic non-residue $x$. The modified JL encrypts $m \in \{0,1\}^k$ as

$$C = y^m h^r \mod N,$$

where $r$ is the randomness from $\mathbb{Z}_N$.

Then, we give a JL commitment whose public parameter is exactly the public key of JL encryption. The JL commitment of any integer $m$, denoted by $\mathsf{JL.commit}(m)$, is

$$c = y^{2^k m} h^{2^k r} \mod N,$$

with $r$ as the opening. Note that $y^{2^k}$, $h^{2^k}$ and $h$ belong to the subgroup of $2^k$-th power residues modulo $N$, whose order $p'q'$ is unknown to the committer. The JL commitment can be easily extended to commit a vector of elements without increasing its size by including more $y$-elements in the public parameter. We call the resulting scheme JL vector commitment.

In this way, our JL (vector) commitment can be treated as an integer commitment. Obviously, it is statistical hiding. We showed that it is computationally binding under $k$-QR assumption and another assumption which we called the strong JL assumption. In Theorem 1, we prove that the strong JL assumption holds under the standard $k$-QR and strong RSA assumptions.

Our main observation is that, by raising the power of $2^k$, we could transfer a modified JL ciphertext to a JL commitment, and transfer any affine operation on a JL ciphertext to a JL vector commitment. These transformations may allow to omit the sending of commitment which contributes to the cost of Paillier-based MtA. Specifically, let $C_2$ be a JL ciphertext and $C_2^a y^\alpha \mod N$ be the affine operation for some $a$ and $\alpha$. Then, $(C_2^a y^\alpha)^{2^k} \mod N$ is a JL vector commitment of $a, \alpha$. In other words, it can replace the role of the additional RSA commitments in the Paillier-based MtA (i.e, $c_{11}, c_{12}$ of Figure 1). Participant may simply sends the proof, and the counterparty can locally generate the commitment by converting the ciphertext. In addition, it offers better generality: the problem of designing a zero-knowledge proof on a JL ciphertext could be reduced to that of a JL commitment.

*Standard zero-knowledge proofs for JL cryptosystem.* We design zero-knowledge proofs for our JL commitment / encryption with standard soundness. The main barrier of designing zero-knowledge proof for JL schemes is the small factor of $2^k$ in the order of $\mathbb{Z}_N^*$. By raising the power of $2^k$, all the operations are computed over the group of $2^k$-th power residue, whose order $p'q'$ is unknown and has no small factors. In this way, we could borrow the technique for RSA commitment (which also works on unknown-order group) to argue the standard soundness of proofs for JL schemes, although we need to handle a different mathematical structure. Finally, we give zero-knowledge proofs for JL commitment, JL encryption, and proof of equality between a JL commitment and a JL ciphertext. These proofs also guarantee the range of the committed value or plaintext (with slack).

*JL-based MtA.* With the above tools, i.e., modified JL encryption, JL (vector) commitment and relevant zero-knowledge proofs, we construct JL-based MtA as illustrated in Figure 1. Compared with Paillier-based MtA, our scheme utilizes compact encryption and zero-knowledge proofs, and does not need to send the commitments of $a$ and $\alpha$.

**Extension to Batch MtA.** We further apply batching technique to improve the overall performance of JL-based MtA. Specifically, in $l$ MtAs, $P_2$ could commits the $l$ $b$-elements into one JL vector commitment. This reduce the number of commitments needed. In addition, the corresponding zero-knowledge proofs and verification could be batched as well. Details are shown in Sec. 5.2.

## 1.3 Discussion

**On Efficiency.** Prior to our work, homomorphic encryption based MtA uses either Paillier or CL. An advantage of CL encryption is that its message space is exactly the same as the signature space of ECDSA ($\mathbb{Z}_q$) and thus no additional integer commitment and range proof are needed. However, CL encryption is built on class groups whose operation is very expensive. On the other hand, there are "wasted" message space in Paillier encryption. Furthermore, due to a mismatch between the signature space of ECDSA and the message space of Paillier ($\mathbb{Z}_N$), an additional integer commitment (i.e., RSA commitments) and the expensive range proofs are needed. Finally, while more efficient than operations in class groups, Paillier still requires some operations in $\mathbb{Z}_{N^2}$ which is also quite heavy.

The efficiency gain of our JL-based constructions (over Pailler-based constructions) can be explained as follows:

- There is less "wasted" message space. Also, all operations are in $\mathbb{Z}_N$.
- The JL ciphertext can be converted to JL commitment on demand. We can reduce the number of integer commitments and range proofs.

As such, it is natural to see our construction leads to the best tradeoffs between time and space complexity.

- Space Complexity: CL-based MtA < Our MtA < Paillier-based MtA $\ll$ OT-based MtA
- Time Complexity: OT-based MtA $\ll$ Our MtA < Paillier-based MtA < CL-based MtA

Please refer to Table 4 and Figure 4 for concrete comparison.

**Other Applications.** Our zero-knowledge proofs can be applied to two-party computation to provide standard soundness and improve performance, such as Mon$\mathbb{Z}_{2^k}$a [14]. Our JL-based MtA can be used as a replacement of Paillier-based MtA in several multiparty computations, e.g., SPDZ [21, 22], SPD$\mathbb{Z}_{2^k}$ [18]. In addition, MPC usually runs over a much smaller field, e.g., $< 2^{64}$. Thus, a smaller $k$ and a tight modulus $N$ could be used, which further enlarge the overall improvement. We leave this as a further work.

Our JL-based MtA could be used to build more efficient three-party TLS handshake [46, Section 4.1] by replacing their Paillier-based MtA. Since 4 MtAs are required in their protocol, our batch technique can be applied to further enlarge the improvement.

We would like to remark that the technique of transforming modified JL ciphertext to a JL commitment and related range proof are of independent interest. It has potential applications to replace Paillier encryption with range proof in several protocols, e.g., voting scheme in [20], and Naor-Yung CCA secure encryption [37].

**Limitations.** Key generation phase of JL-based MtA is costly. We need to prove that the key is generated properly. However, correctness proof of the JL modulus is still not well-studied. The proof could be extended from that of Camenisch et al. [9]. Appendix E.1 gives a discussion. We leave efficient proof of correctness of JL modulus as an open problem and future work. Fortunately, since key generation is one-time only, an expensive key generation phase is usually acceptable. Furthermore, as discussed in [1], we may assume there exists a trusted dealer in the setup process in some applications. This is reasonable, for instance, in cryptocurrency applications when a client generates its own key and distributed them to a number of servers to protect the security of his own key.

In addition, there are some subtle issues in the choice of parameters. Security of JL cryptosystem requires $k \leq 1/4 \log N - \lambda$, where $\lambda$ is a security parameter. In other words, for a small security parameter, the message space of JL (i.e., $k$) maybe too small for encryption of the shares in threshold ECDSA. Then we have to increase $N$ to increase the message space. This happens when the security parameter $\lambda$ is 128. In this case, JL needs a 3360-bits modulus. In contrast, a 3072-bit modulus would be secure enough for Paillier. For 192-bit and 256-bit security, JL can use the same modulus as Paillier.

Finally, when $k$ is large (e.g. $\approx$ 1000-bit), current decryption of JL scheme is slower than that of Paillier. We propose a faster decryption algorithm by adding lookup tables to secret key. Details are given in Sec. 2.2. In case $\lambda = 128$, our decryption runs in 15 ms at a cost of 1168.1 KB secret key.

## 1.4 Related works

This section presents related works on zero-knowledge proof for JL encryption, constructions of MtA, and threshold ECDSAs.

**Proofs on JL encryption.** With the aim to design two-party computation over $\mathbb{Z}_{2^k}$, Catalano et al. [14] proposed several zero-knowledge proofs on the original JL encryption. Their proofs include proving knowledge of a JL plaintext, and proof of multiplication of two JL encrypted values. However, these proofs only provide non-standard soundness, i.e., only the least significant $k - t$ bits are extracted, and say nothing to the maximum significant $t$ bits.

**Constructions of MtA.** Aiming to design threshold ECDSA, several MtAs have been proposed from Paillier encryption, CL encryption, OT, etc. Paillier-based MtA can be traced back to [36] and is subsequent improved in [10, 29, 35]. Due to the mismatch of Paillier's message space and ECDSA's signature space, these schemes require relatively expensive zero-knowledge range proofs. Several techniques could be applied to simplify the proofs, such as range proof with slack [35]. Castagnos et al. [11, 12] replaced Paillier encryption with Castagnos and Laguillaumie (CL) [13] encryption from the observation that CL's message space matches the signature space of ECDSA. While there are subsequent works to improve efficiency [23, 45], CL-based MtAs are still computationally expensive. Doerner et al. [25, 26] built OT-based MtA from simplest OT [15] and OT extensions [32]. It is computationally efficient at a cost of relatively high bandwidth requirement (e.g., 90 KB for 128 bits security). There are instantiations from other tools, such as pseudorandom correlation generators [1], Ring-LWE [5], and noisy Reed-Solomon encodings (RS) [30].

**Threshold ECDSA.** We give a brief account of threshold ECDSA here. Interested readers may refer to a survey in [3].

Lindell et al. [35], and Gennaro and Goldfeder [29] proposed a full threshold ECDSA protocol. They both require at least $n(n-1) + n/2$ MtAs when $n$ parties are involved. Later, Canetti et al. [10] proposed an UC secure four-pass threshold ECDSA. We would like to remark that Tymokhanov et al. [42] discovered a weakness in the implementation of Gennaro and Goldfeder's scheme [29] when zero-knowledge proofs are eliminated. Following the blueprint of Gennaro and Goldfeder [29], several works [12, 23, 45] designed threshold ECDSA by replacing Paillier-based MtA with that from CL encryption. Doerner et al. [26] proposed a full threshold scheme from oblivious transfer.

Recently, Abram et al. [1] built threshold ECDSA with low-bandwidth from pseudorandom correlation generator (PCG). Their bandwidth complexity is $1 \sim 2$ orders of magnitude smaller than those based on Paillier encryption [10, 29, 35] or CL encryption [12], however, their amortized computational cost is expensive (i.e, $1 \sim 2$ seconds per ECDSA signature).

There are schemes that only focus on the two-party case, and some of them do not rely on MtA. Lindell [34] presented a competitive two-party ECDSA from Paillier, which is subsequently improved by Castagnos et al. [11]. Doerner et al. [25] achieved two-party ECDSA supporting fast online computation with the help of two MtAs from oblivious transfer. Xue et al. [44] further construct a general online-friendly two-party ECDSA from a single MtA.

## 1.5 Paper Organization.

We review preliminaries in Section 2, and propose JL commitment and zero-knowledge proofs for JL cryptosystem in Section 3 and 4. Section 5 presents the JL-based MtA and batching technique. We compare the complexity between our MtA and those from OT, Paillier and CL in Section 6. Finally, Section 7 gives benchmarks of JL-based MtA in threshold ECDSA and comparisons.

# 2 PRELIMINARY

In this paper, we denote by $\lambda$ the security parameter Given a finite set $D$, $a \leftarrow D$ means sampling a uniformly random $a$ from set $D$.

## 2.1 Mathematics and Assumptions

Let $p$ be an odd prime and let $n \geq 2$ such that $n | p-1$. The $n$-th power residue symbol modulo $p$ is defined as

$$\left(\frac{a}{p}\right)_n = a^{\frac{p-1}{n}} \mod p.$$

We abuse the notion $J_p(a) = \left(\frac{a}{p}\right)_2$. If $N = p\bar{q}$ is RSA modulus, $J_N(a)$ is defined as $J_p(a)J_{\bar{q}}(a)$.

Let $N = p\bar{q} = (2^k p' + 1)(2q' + 1)$ where $p, \bar{q}$ are primes, $k > 1$,and $p', q'$ are odd number. In the following, we denote such special RSA modulus as JL modulus. Define

$$\mathbb{J}_N = \left\{a \in \mathbb{Z}_N^* \mid J_N(a) = 1\right\}, \bar{\mathbb{J}}_N = \left\{a \in \mathbb{Z}_N^* \mid J_N(a) = -1\right\}$$
$$\mathbb{QR} = \left\{a \in \mathbb{Z}_N^* \mid \exists x \in \mathbb{Z}_N^*, a = x^2 \mod N\right\}, \mathbb{QNR} = \mathbb{J}_N \setminus \mathbb{QR}$$
$$\mathbb{QR}_{2^k} = \{a \in \mathbb{Z}_N^* \mid \exists x \in \mathbb{Z}_N^*, a = x^{2^k} \mod N\}.$$

Fact 1. *Let $N = p\bar{q}$ be a JL modulus. We have*

(1) $-1 \in \bar{\mathbb{J}}_N$ *since* $J_p(-1) = 1$ *and* $J_{\bar{q}}(-1) = -1$.
(2) $\mathbb{QR}_{2^k}$ *is the cyclic subgroup of* $\mathbb{Z}_N^*$ *of order* $p'q'$.
(3) *A random element from* $\mathbb{QR}_{2^k}$ *is its generator with probability* $(1 - 1/p')(1 - 1/q')$.
(4) *Finding a non-trivial square root (i.e.* $\neq \pm 1$*) of 1 is equivalent to factoring the modulus* $N$.

Definition 1 ($k$-QR Assumption [7]). *Let $N = p\bar{q}$ be a JL modulus. The k-QR assumption asserts that $Adv_{\mathcal{A}}^{kQR}$, defined as:*

$$|Pr[\mathcal{A}(x, k) = 1|x \leftarrow \mathbb{QR}] - Pr[\mathcal{A}(x, k) = 1|x \leftarrow \mathbb{QNR}]|,$$

*is negligible for any PPT distinguisher $\mathcal{A}$. The probability is taken over the randomness generating $N$ and choosing $x$.*

Lemma 1 (Theorem 3 in [7] for $\bar{q} = 3 \mod 4$). *Let $N = p\bar{q}$ be a JL modulus. For any PPT $\mathcal{D}$, define $Adv_{\mathcal{D}}^{Gap\text{-}2^k}$ as*

$$|Pr[\mathcal{D}(x, k) = 1|x \leftarrow \mathbb{QR}_{2^k}] - Pr[\mathcal{D}(x, k) = 1|x \leftarrow \mathbb{QNR}]|.$$

*We have, for any PPT algorithm $\mathcal{D}$, there exits a k-QR solver $C$ such that $Adv_{\mathcal{D}}^{Gap\text{-}2^k} \leq 3/2(k - 1/3)Adv_{C}^{kQR}$.*

We introduced a new assumption, namely, *strong JL assumption*, which is needed in the security analysis of our proposed primitives.

Definition 2 (Strong JL assumption). *Let $N$ be the JL modulus. The strong JL assumption states that, for a random element $x \in \mathbb{QR}_{2^k}$, it is hard to find the e-th root a modulo N, i.e., $a^e = x \mod N$, for any PPT algorithm and an exponent $e > 1$ of its choice.*

Theorem 1. *The strong JL assumption holds under k-QR assumption and strong RSA assumption with JL modulus.*

Proof is given in Appendix A.2.

## 2.2 Joye-Libert Encryption (Revisited)

The Joye-Libert encryption scheme [7] (an extension of [31]) contains the tuple (JL.kgen, JL.enc, JL.dec) as follows.

(1) JL.kgen$(1^\lambda)$. It defines a proper integer $k$, randomly generates primes $p = 2^k p' + 1$ and $\bar{q} = 2q' + 1$ where $p', q'$ are odd numbers (see below for a discussion on the choice of parameters), and set $N = p\bar{q}$. It also picks a random $y \in \mathbb{QNR}$. Let $pk = (N, y, k)$ and $sk = p$ be the public and secret key pair.
(2) JL.enc$(pk, m)$. Choose a random $r \in \mathbb{Z}_N^*$ and compute $C = y^m r^{2^k} \mod N$ as the ciphertext of $m \in \{0, 1\}^k$ (which is taken as an integer in $[0, \cdots, 2^k - 1]$).
(3) JL.dec$(sk, C)$. Given secret key $sk = p$, compute the $2^k$-th power residue symbol $z = \left(\frac{C}{p}\right)_{2^k}$. Find $m \in \{0, 1, \cdots, 2^k - 1\}$ such that the relation $z = \left[\left(\frac{y}{p}\right)_{2^k}\right]^m \mod p$ holds. A decryption algorithm is given in [31].

Under the $k$-QR assumption, the Joye-Libert encryption is IND-CPA secure according to [7, Theorem 2]. We give a modified Joye-Libert (JL) scheme which retains its security guarantee.

(1) In JL.kgen, choose $y \leftarrow \mathbb{QNR}$ and $h \leftarrow \mathbb{QR}_{2^k}$. Let $pk = (N, h, y, k)$ and $sk = p$.
(2) In JL.enc, choose a random $r \in \mathbb{Z}_N$, and compute ciphertext
$$C = y^m h^r \mod N.$$

We can choose $h \leftarrow \mathbb{QR}_{2^k}$ by $x \leftarrow \mathbb{QNR}$ and computing $h = x^{2^k} \mod N$. According to Fact 1, the order of $h$ is $p'q'$ with overwhelming probability $(1 - 1/p')(1 - 1/q')$.

**Fast Decryption.** We further give a fast decryption algorithm by adding lookup tables to the secret key. Assume $k = ln$ for some integers $l$ and $n$. We view the message as $m = \sum_{i=1}^{n} 2^{(i-1)l} m_i$ where $m_i \in [0, 2^l - 1]$. Then, given $C = y^m h^r \mod N$, we have for $i = 1, 2, \cdots, n$,

$$\left(\frac{C}{p}\right)_{2^{il}} = \left(\frac{y}{p}\right)_{2^{il}}^{m_1} \left(\frac{y}{p}\right)_{2^{(i-1)l}}^{m_2} \cdots \left(\frac{y}{p}\right)_{2^l}^{m_i} \mod p.$$

Thus, we could find $m_1, \cdots, m_n$ step by step using the following lookup tables,

$$T_1 = \left[\left(\frac{y}{p}\right)_{2^l}^0, \left(\frac{y}{p}\right)_{2^l}^1, \cdots, \left(\frac{y}{p}\right)_{2^l}^{2^l - 1}\right]$$
$$T_2 = \left[\left(\frac{y}{p}\right)_{2^{2l}}^0, \left(\frac{y}{p}\right)_{2^{2l}}^1, \cdots, \left(\frac{y}{p}\right)_{2^{2l}}^{2^l - 1}\right]$$
$$\cdots$$
$$T_n = \left[\left(\frac{y}{p}\right)_{2^{nl}}^0, \left(\frac{y}{p}\right)_{2^{nl}}^1, \cdots, \left(\frac{y}{p}\right)_{2^{nl}}^{2^l - 1}\right].$$

Let $T = (T_1, T_2, \cdots, T_n)$ and $sk = (p, T)$ in JL.kgen. The fast decryption algorithm runs as follows.

- For $i = n, \cdots, 1$, compute $z_i = \left(\frac{C}{p}\right)_{2^{il}}$.
- Set $m_1 = j$ if there exists $j$ s.t. $T_1[j] = z_1$, otherwise abort.
- For $i = 2, 3, \cdots, n$, do
  - tempz $= z_i \times (T_i[m_1] \times \cdots \times T_2[m_{i-1}])^{-1} \mod p$
  - Set $m_i = j$ if $\exists j$ s.t. $T_1[j] =$ tempz, otherwise abort.
- Output $m = \sum_{i=1}^{n} 2^{(i-1)l} m_i$.

**Choice of Parameters.** The security analysis of Joye-Libert [7, 31] requires that $k \leq 1/4 \log N - \lambda$, and that $p'$ and $q'$ are odd numbers, each of which has a large prime factor. Looking ahead, our zero-knowledge proofs (e.g. knowledge soundness in Appendix C.1) further require all factors of $p', q'$ are not less than $2^t$ where $t$ is the soundness parameter. This could be guaranteed by setting $p'$ and $q'$ to be primes (in this paper).

## 2.3 Commitment

A commitment Com is a 3-tuple (setup, commit, verify) with message space $\mathcal{M}_{\text{com}}$, commitment space $C_{\text{com}}$, and opening space $\mathcal{R}_{\text{com}}$.

(1) Com.setup$(1^\lambda)$. Generate public parameters pp.

**Setup**: On receiving (setup) from $P_1$ and $P_2$

- Store and send (setup-complete) to $P_1$ and $P_2$.

**Multiplication**: On receiving (input, sid, $a \in \mathbb{Z}_q$) from $P_1$, (input, sid, $b \in \mathbb{Z}_q$) from $P_2$ where sid has not been used, if (setup-complete) exists:

- Sample $\alpha \in \mathbb{Z}_q$ and compute $\beta = ab - \alpha \mod q$.
- Send (output-1, sid, $\alpha$) to $P_1$
- Send (output-2, sid, $\beta$) to $P_2$.

**Figure 2: Multiplicative-to-additive functionality $\mathcal{F}_{\text{MtA}}$.**

(2) $\mathsf{Com.commit}(\mathsf{pp}, m)$. Compute a commitment $c$ to $m \in \mathcal{M}_{\mathsf{com}}$ with its opening $d \in \mathcal{R}_{\mathsf{com}}$, and output pair $(c, d)$ as the commitment and its opening.

(3) $\mathsf{Com.verify}(\mathsf{pp}, c, m, d)$. Output a bit to indicate the validation of $(m, d)$ with respect to commitment $c$.

The correctness requires that for any $\mathsf{pp} \leftarrow \mathsf{Com.setup}(1^\lambda)$, any $m \in \mathcal{M}_{\mathsf{com}}$, it holds that $\mathsf{Com.verify}(\mathsf{pp}, c, m, d) = 1$, if $(c, d) \leftarrow \mathsf{Com.commit}(\mathsf{pp}, m)$.

A commitment could be statistical hiding and computational binding, or computational hiding and statistical hiding. We focus on the first one.

- *Hiding*: For any $m, m' \in \mathcal{M}_{\mathsf{com}}$, their commitments are statistical indistinguishable.
- *Binding*: No probability polynomial time (PPT) adversary could open a commitment $c$ on two different messages.

## 2.4 The Multiplicative-to-Additive Functionality

Functionality $\mathcal{F}_{\text{MtA}}$ runs between two parties, namely, $P_1$ and $P_2$. The functionality, illustrated in Figure 2, is parameterized by a number $q$ (In this paper, we focus on the prime group order $q$ in ECDSA). $P_1$ and $P_2$ participate in the Setup phase once, and run the Multiplication phases as many times as they wish. $\mathcal{F}_{\text{MtA}}$ outputs $\alpha$, $\beta$ on input $a$ and $b$ from $P_1$ and $P_2$ respectively, under the constraint that $\alpha + \beta = ab \mod q$.

## 2.5 Zero-Knowledge Proof

An interactive proof for a language $L$ is an interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Assume $\mathcal{R}$ is the associated relation of $L$. We call $(\mathcal{P}, \mathcal{V})$ an interactive proof for $\mathcal{R}$ or $L$ if it satisfies: completeness which says for every $x \in L$, $\langle \mathcal{P}, \mathcal{V} \rangle(x)$ always accepts; and soundness which says for every $x \notin L$ and every prover $\mathcal{P}^*$, $\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = 1]$ is negligible. When the soundness holds for computationally bounded provers, the system is usually called an "argument". We abuse the notion and use proof to represent both proof and argument.

An interactive proof is zero-knowledge if for every PPT $V^*$ there exists a PPT simulator Sim s. t. $\{View^{\mathcal{P}}_{\mathcal{V}^*}(x)\}_{x \in L}$ and $\{\mathsf{Sim}(x)\}_{x \in L}$ are statistically indistinguishable. It is said to be honest-verifier zero-knowledge if zero-knowledge holds for any PPT honest verifier. $\Sigma$-protocol [17] is a special honest-verifier zero-knowledge proof.

Definition 3 ($\Sigma$-protocol). *$\Sigma$-protocol is a special 3-move zero-knowledge proof $(\mathcal{P}, \mathcal{V})$ and proceeds as follows: $\mathcal{P}$ with inputs $(x, w) \in \mathcal{R}$, computes $(a, st)$ and sends $a$ to $\mathcal{V}$, who sends back a random challenge $e$; $\mathcal{P}$ sends a response $z = \mathcal{P}(x, w, a, e, st)$ to $\mathcal{V}$; On input of $(a, e, z)$, $\mathcal{V}$ outputs 0 or 1.*

- *Completeness: If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on input $(x, w)$ to $\mathcal{P}$ where $(x, w) \in \mathcal{R}$, $\mathcal{V}$ always outputs 1.*
- *Honest-verifier zero-knowledge: There exists a PPT simulator Sim that on input $x \in L$ and a challenge $e$, outputs $(a, z)$ such that $(a, e, z)$ is indistinguishable from a real transcript with challenge $e$.*
- *Special Soundness (proof of knowledge): There exists a PPT knowledge extractor Ext that for any statement $x$, on input of two accepting transcripts $(a, e, z)$ and $(a, e', z')$ with $e \neq e'$, outputs a witness $w'$ such that $(x, w') \in \mathcal{R}$.*

All zero-knowledge proofs in this paper are $\Sigma$-protocols and could be converted into non-interactive form via Fiat-Shamir transformation [27].

# 3 JL (VECTOR) COMMITMENT

In this section, we describe our JL (vector) commitment scheme and its relation with modified JL encryption. The JL commitment contains the tuple ($\mathsf{JL.setup}, \mathsf{JL.commit}, \mathsf{JL.verify}$).

(1) $\mathsf{JL.setup}(1^\lambda)$. Run $\mathsf{JL.kgen}$ of the modified JL encryption to obtain public key $pk = (N, h, y, k)$, and set it as pp.

(2) $\mathsf{JL.commit}(\mathsf{pp}, m)$. For $m \in \mathbb{Z}$, randomly choose $r \leftarrow \mathbb{Z}_N$, compute

$$c = y^{2^k m} h^{2^k r} \mod N,$$

and return $(c, r)$ as the commitment and its opening.

(3) $\mathsf{JL.verify}(\mathsf{pp}, c, m, d)$. If $c = y^{2^k m} h^{2^k d} \mod N$ and $J_N(c) = 1$, output 1, otherwise 0. Note that checking the Jacobi symbol is crucial for security.

Theorem 2. *If strong JL and $k$-QR assumptions hold, JL commitment ($\mathsf{JL.setup}, \mathsf{JL.commit}, \mathsf{JL.verify}$) is a statistical hiding and computational binding commitment.*

Proof. The correctness is obvious. Hiding property comes from the facts that $h^{2^k}$ is also a generator of $\mathbb{QR}_{2^k}$ (whose order is $p'q'$), and $y^{2^k} \in \mathbb{QR}_{2^k}$. There exists a $\alpha$ such that $y^{2^k} = h^{2^k \alpha} \mod N$. Thus, $c = y^{2^k m} h^{2^k r} \mod N$ could also be taken as the commitment of $m'$ with opening $r + \alpha(m - m')$.

Binding relies on the strong JL and $k$-QR assumptions. Given an instance $(N, h, k)$ for strong JL problem, the strong JL solver generates $y = h^\alpha \mod N$ for a random $\alpha \in \mathbb{Z}_N$ and sets $(N, h, y, k)$ as public parameter of JL commitment. The only difference is the generation of $y$ (i.e., $y \in \mathbb{QNR}$ or $y \in \mathbb{QR}_{2^k}$). The committer could not find this difference due to the $k$-QR assumption (according to Lemma 1). Then, from two different openings $(m, d)$ and $(m', d')$ of $c$, the verification check guarantees that $y^{2^k m} h^d = y^{2^k m'} h^{2^k d'} \mod N$. Denote $\Delta m = m - m'$, $\Delta d = d - d'$, then $y^{2^k \Delta m} h^{2^k \Delta d} = 1 \mod N$.

Recall the generation of $y$, we have $y^{2^k} = h^{2^k \alpha} \mod N$. Thus, $h^{2^k(\alpha \Delta m + \Delta d)} = 1 \mod N$. Let $e > 1$ be any number that is co-prime to $E = 2^k(\alpha \Delta m + \Delta d)$. We have

$$\left(h^{e^{-1} \mod E}\right)^e = h \mod N,$$

which finds $h^{e^{-1} \mod E}$ as the $e$-th root of $h \in \mathbb{QR}_{2^k}$, i.e., a solution of the strong JL problem. □

**Extension to vector commitment.** We generalize our scheme to commit a vector. Looking ahead, vector commitment (JLv.setup, JLv.commit, JL.verify) helps batching MtA (details presented in the next section).

(1) JLv.setup($1^\lambda$). As in JL.kgen of the modified JL encryption, choose $h \in \mathbb{QR}_{2^k}$ and generate many $y$-elements $y_1, y_2, \cdots, y_l \in \mathbb{QNR}$. Then output $(N, h, y_1, \cdots, y_l, k)$ as pp.
(2) JLv.commit(pp, $\vec{m}$). For vector $\vec{m} = (m_1, \cdots, m_l) \in \mathbb{Z}^l$, randomly choose $r \leftarrow \mathbb{Z}_N$, and compute

$$c = \prod_{i=1}^{l} y_i^{2^k m_i} \cdot h^{2^k r} \mod N.$$

Return $(c, r)$ as the commitment and its opening.
(3) JLv.verify(pp, $c$, $\vec{m}$, $d$). If $c = \prod_{i=1}^{l} y_i^{2^k m_i} \cdot h^{2^k d} \mod N$ and $J_N(c) = 1$, output 1, otherwise 0.

Theorem 3. *Under the strong JL and k-QR assumptions, vector commitment* (JLv.setup, JLv.commit, JLv.verify) *is a statistical hiding and computational binding commitment.*

Please refer to Appendix B for the proof.

**Converting a JL Ciphertext to a JL (vector) commitment.** Our MtA in Section 5 builds on the following observations.

(1) One could convert a JL ciphertext $C = y^m h^r \mod N$ into a JL commitment (under the same public key) of $m$ with opening $r$ by computing $c = C^{2^k} \mod N$. The conversion does not require any private knowledge.
(2) Furthermore, affine operation on a JL ciphertext can also be converted to our JL vector commitment with $l = 2$. This will be used in our MtA (refer to Section 5). Specifically, let $C$ be the JL ciphertext under public key $(N, h, y, k)$, let $C_{\mathsf{aff}} = C^a y^\alpha h^r \mod N$ be an affine operation on $C$, then $c_{\mathsf{aff}} = C^{2^k} \mod N$ could be taken as a JL vector commitment to $(a, \alpha)$ with bases $(C, y, h)$ and opening $r$.

# 4 ZERO-KNOWLEDGE PROOFS FOR JL CRYPTOSYSTEM

## 4.1 Proof for JL (vector) Commitment $\mathsf{ZK}_{\mathsf{JL\text{-}com}}$ / $\mathsf{ZK}_{\mathsf{JLv\text{-}com}}$

The public parameter pp and commitment $c = y^{2^k m} h^{2^k r} \mod N$ are the common input. The prover would like to prove the knowledge of a $m$ in range $[0, B]$ such that the following relation holds:

$$\mathcal{R}_{\mathsf{JL\text{-}com}} = \{(c; m, r) \mid c = y^{2^k m} h^{2^k r} \mod N, m \in [0, B]\}.$$

We define $\Sigma$-protocol $\mathsf{ZK}_{\mathsf{JL\text{-}com}}$ between $\mathcal{P}$ and $\mathcal{V}$, where $s$ and $t$ are the statistical and soundness parameters respectively.

- $\mathcal{P}$ randomly chooses $v$ from $[0, 2^{s+t}B]$, and $w$ from $[0, 2^{s+t}N]$. $\mathcal{P}$ computes and sends $d = y^{2^k v} h^{2^k w} \mod N$ to $\mathcal{V}$.
- $\mathcal{V}$ chooses and sends $e \leftarrow \{0, 1\}^t$ to $\mathcal{P}$.
- $\mathcal{P}$ computes and sends $z_m = em + v$ and $z_r = er + w$ (as integers) to $\mathcal{V}$.
- $\mathcal{V}$ accepts the proof only if
  - $J_N(c) = J_N(d) = 1$, $c^e d = y^{2^k z_m} h^{2^k z_r} \mod N$,
  - $z_m \in [0, 2^{s+t}B]$.

The completeness is trivial. The protocol is honest-verifier zero-knowledge since simulator Sim can be constructed using standard techniques: Sim chooses random responds $z_m \leftarrow [0, 2^{s+t}B]$, $z_r \leftarrow [0, 2^{s+t}N]$, together with $e \leftarrow \{0, 1\}^t$, and sets $d = y^{2^k z_m} h^{2^k z_r} c^{-e} \mod N$.

Showing that the above protocol has special soundness (proof-of-knowledge) is more involved and requires the strong JL and $k$-QR assumptions. Briefly, we show that there exists a probabilistic oracle machine to either extract witness $m, r$ or solve the strong JL or $k$-QR problems. Please refer to Appendix C.1 for a detailed proof.

The proof guarantees the range with slack, i.e., $m \in [-2^{s+t}B, 2^{s+t}B]$, since for any $m$ satisfying $|m| > 2^{s+t}B$, the probability of guessing the right $v$ such that $em + v \in [0, 2^{s+t}B]$ is less then $1/2^t$.

**Opening proof of JL vector commitment.** The $\mathsf{ZK}_{\mathsf{JL\text{-}com}}$ protocol can be extended to prove opening of JL vector commitment, i.e., the following relation

$$\mathcal{R}_{\mathsf{JLv\text{-}com}} = \{(c; \vec{m}, r) | c = y_1^{2^k m_1} \cdots y_l^{2^k m_l} h^{2^k r} \mod N, m_i \in [0, B_i]\},$$

where $N, h, y_1, \cdots, y_l, k$ are public parameters, and $\vec{m} = (m_1, \cdots, m_l)$. We define $\Sigma$-protocol $\mathsf{ZK}_{\mathsf{JLv\text{-}com}}$ as follows, where $s$ and $t$ are the statistical and soundness parameters respectively.

- $\mathcal{P}$ chooses random $v_i$ from $[0, 2^{s+t}B_i]$ (for $1 \le i \le l$), and random $w$ from $[0, 2^{s+t}N]$. $\mathcal{P}$ computes and sends $d = y_1^{2^k v_1} \cdots y_l^{2^k v_l} h^{2^k w} \mod N$ to $\mathcal{V}$.
- $\mathcal{V}$ chooses and sends $e \leftarrow \{0, 1\}^t$ to $\mathcal{P}$.
- $\mathcal{P}$ computes and sends $z_i = em_i + v_i$ for $1 \le i \le l$, and $z_r = er + w$ (as integers) to $\mathcal{V}$.
- $\mathcal{V}$ accepts the proof only if the following holds
  - $J_N(c) = J_N(d) = 1$, $c^e d = y_1^{2^k z_1} \cdots y_l^{2^k z_l} h^{2^k z_r} \mod N$,
  - $z_i \in [0, 2^{s+t}B_i]$ for $1 \le i \le l$.

The proof guarantees that every $m_i \in [-2^{s+t}B_i, 2^{s+t}B_i]$. The security analysis is give in Appendix C.2.

## 4.2 Range Proof for JL Encryption / Affine Operation $\mathsf{ZK}_{\mathsf{JL\text{-}enc}}$ / $\mathsf{ZK}_{\mathsf{JL\text{-}aff}}$

Given a JL ciphertext $C$ satisfying $J_N(C) = 1$, we can use the proof for JL commitment $c = C^{2^k} \mod N$ to prove relations for JL encryption. Concretely, define the following relation,

$$\mathcal{R}_{\mathsf{JL\text{-}enc}} = \{(C; m, r) \mid J_N(C) = 1, C = y^m h^r \mod N, m \in [0, B]\},$$

where $C$ is the common input.

Lemma 2. *Under the factoring assumption, when $J_N(C) = 1$, by setting $c = C^{2^k} \mod N$,* $\mathsf{ZK}_{\mathsf{JL\text{-}com}}$ *for relation $\mathcal{R}_{JL\text{-}com}$ is exactly a $\Sigma$-protocol, denoted by* $\mathsf{ZK}_{\mathsf{JL\text{-}enc}}$*, for relation $\mathcal{R}_{JL\text{-}enc}$.*

Proof. Completeness and honest-versifier zero-knowledge are the same. We only need to analyse the soundness. The soundness of $\mathsf{ZK}_{\mathsf{JL\text{-}com}}$ provides an extractor Ext to extract $m$ and $r$ such that $c = C^{2^k} = y^{2^k m} h^{2^k r} \mod N$. Now we claim that this is also the witness of relation $\mathcal{R}_{\mathsf{JL\text{-}enc}}$. According to Fact 1 (1) and (4), $(C^{-1} y^m h^r)^2 = 1 \mod N$, otherwise the prover provides a non-trivial square-root of

1 which violates the factoring assumption. Applying the factoring assumption again, $C^{-1}y^m h^r = \pm 1 \mod N$. Since $J_N(C) = 1$, we have $C^{-1}y^m h^r = 1 \mod N$. □

**Range proof for JL affine operation.** As mentioned at the end of Section 3, let $C_{\mathsf{aff}} = C^a y^\alpha h^r \mod N$ be an affine operation on JL ciphertext $C$ under public key $(N, h, y, k)$, we have $c = C_{\mathsf{aff}}^{2^k} \mod N$ is a JL vector commitment to $(a, \alpha)$ with bases $(C, y, h)$. Thus, proof for JL vector commitment could be used to prove affine operation on a JL ciphertext. The $\Sigma$-protocol, denoted by $\mathsf{ZK}_{\mathsf{JL\text{-}aff}}$, for relation

$$\mathcal{R}_{\mathsf{JL\text{-}aff}} = \{(C_{\mathsf{aff}}, C; a, \alpha, r) \mid J_N(C) = 1, C_{\mathsf{aff}} = C^a y^\alpha h^r \mod N \\ a \in [0, B_1], \alpha \in [0, B_2]\},$$

is exactly $\mathsf{ZK}_{\mathsf{JLv\text{-}com}}$ for relation $\mathcal{R}_{\mathsf{JLv\text{-}com}}$ by setting $l = 2$, $c = C_{\mathsf{aff}}^{2^k} \mod N$, $(m_1, m_2) = (a, \alpha)$, and $(y_1, y_2) = (C, y)$.

Similarly, we should analyse the soundness, i.e., the extractor for $\mathsf{ZK}_{\mathsf{JLv\text{-}com}}$ extracts the witness of relation $\mathcal{R}_{\mathsf{JL\text{-}aff}}$, i.e., $(a, \alpha, r)$. We only have $C_{\mathsf{aff}}^{2^k} = C^{2^k a} y^{2^k \alpha} h^{2^k r} \mod N$ from $\mathsf{ZK}_{\mathsf{JLv\text{-}com}}$'s soundness. As in Lemma 2, under the factoring assumption and condition $J_N(C_{\mathsf{aff}}) = 1$, $C_{\mathsf{aff}} = C^a y^\alpha h^r \mod N$.

## 4.3 Proof of Equality $\mathsf{ZK}_{\mathsf{JL\text{-}equ}}$ / $\mathsf{ZK}_{\mathsf{JLv\text{-}equ}}$

We propose the proof of equality which allows the prover to demonstrate that the plaintext of a JL ciphertext corresponds to the opening of a JL commitment. Note that the JL encryption and the JL commitment could be using a different modulus.

Let $(N_0, h_0, y_0, k)$ be the public key of the modified JL encryption, $(N, h, y, k)$ be the public parameter of the JL commitment. The prover would like to prove knowing $m \in [0, B]$ in ciphertext $C$ is equal to the value committed in a JL commitment $c$, i.e.,

$$\mathcal{R}_{\mathsf{JL\text{-}equ}} = \{(C, c; m) \mid c = y^{2^k m} h^{2^k r} \mod N, \\ C = y_0^m h_0^{r_0} \mod N_0, m \in [0, B]\}.$$

The protocol $\mathsf{ZK}_{\mathsf{JL\text{-}equ}}$ works as follows.

- $\mathcal{P}$ chooses random $v$ from $[0, 2^{s+t}B]$, and random $w_0$, $w$ from $[0, 2^{s+t}N]$. $\mathcal{P}$ computes and sends $D = y_0^v h_0^{w_0} \mod N_0$, $d = y^{2^k v} h^{2^k w} \mod N$ to $\mathcal{V}$.
- $\mathcal{V}$ chooses and sends $e \leftarrow \{0, 1\}^t$ to $\mathcal{P}$.
- $\mathcal{P}$ computes and sends $z_m = em + v$, $z_R = er_0 + w_0$, and $z_r = er + w$ (as integers) to $\mathcal{V}$.
- Verification: $\mathcal{V}$ accepts the proof only if
  - $C^e D = y_0^{z_m} h_0^{z_R} \mod N_0$,
  - $c^e d = y^{2^k z_m} h^{2^k z_r} \mod N$, $J_N(c) = J_N(d) = 1$.

Similar to proof of opening, both completeness and honest verifier zero-knowledge are trivial. Special soundness holds under the $k$-QR and strong JL assumptions, which is discussed in Appendix D.1.

**Batch proof of equality to a JL vector commitment.** Protocol $\mathsf{ZK}_{\mathsf{JL\text{-}equ}}$ could be extended to prove equality of many ciphertext to one JL vector commitment, i.e., proving the relation

$$\mathcal{R}_{\mathsf{JLv\text{-}equ}} = \{(\vec{C}, c; \vec{m}) \mid c = y_1^{2^k m_1} \cdots y_l^{2^k m_l} h^{2^k r} \mod N, \\ C_i = y_0^{m_i} h_0^{r_{0,i}} \mod N_0, m_i \in [0, B_i], 1 \le i \le l\}.$$

The extended protocol $\mathsf{ZK}_{\mathsf{JLv\text{-}equ}}$ is given in Appendix D.2.

| $P_1$ $(\mathsf{pp}_1; a \in \mathbb{Z}_q)$ | | $P_2$ $(pk_2, sk_2; b \in \mathbb{Z}_q)$ |
|---|---|---|
| | | $C_2 = \mathsf{JL.enc}(pk_2, b)$ |
| | $\xleftarrow{C_2,\ c,\ \pi_2}$ | $c_2 = \mathsf{JL.commit}(\mathsf{pp}_1, b)$ |
| $C_1 = \left(C_2 \cdot y_2^{2^{s+t}q}\right)^a y_2^{\alpha'} h_2^r \mod N_2$ | $\xrightarrow{C_1,\ \pi_1}$ | $\beta' = \mathsf{JL.dec}(sk_2, C_1)$ |
| $\alpha = -\alpha' \mod q$ | | $\beta = \beta' \mod q$ |

**Figure 3: Multiplication of Single JL-based MtA.**

# 5 MTA VIA JL CRYPTOSYSTEM

We first give a single MtA scheme then present a batched one.

## 5.1 Single JL-based MtA

The single JL-based MtA, illustrated in Figure 3, runs between $P_1$ and $P_2$. They run Setup once and Multiplication phase as many times as they want.

**Setup.** $P_i$ runs $\mathsf{JL.setup}$ to generate its public parameter $\mathsf{pp}_i$, i.e., JL public key $pk_i = (N_i, h_i, y_i, k)$. In additional, each party $P_i$ generates zero-knowledge proof on the correctness of JL modulus $N_i$ (i.e., $\mathsf{ZK}_{\mathsf{JLmod}}$ from Appendix E), and zero-knowledge proofs on $h \in \mathbb{QR}_{2^k}$ and $\exists \alpha \in \mathbb{Z}_N$ s.t. $y^{2^k} = h^{2^k \alpha} \mod N$ (i.e., $\mathsf{ZK}_{\mathbb{QR}_{2^k}}$ and $\mathsf{ZK}_{\mathbb{QR}\mathsf{2kdl}}$ given in Appendix E). $P_i$ takes its own secret key $sk_i$ corresponding to $pk_i$ as private.

**Multiplication.** $P_1$ and $P_2$ invoke the following protocol with their inputs $a \in \mathbb{Z}_q$ and $b \in \mathbb{Z}_q$, and receives $\alpha$, $\beta$ respectively, such that $\alpha + \beta = ab \mod q$.

(1) $P_2$'s message
  - (a) Compute $C_2 = \mathsf{JL.enc}(pk_2, b)$ under its public key.
  - (b) Compute $c = \mathsf{JL.commit}(\mathsf{pp}_1, b)$ under $P_1$'s public parameter.
  - (c) Generate $\pi_2$ as the equality proof of $C_2$ and $c_2$ such that $b \in [0, q]$ using $\mathsf{ZK}_{\mathsf{JL\text{-}equ}}$ of section 4.3.
  - (d) Send $(C_2, c_2, \pi_2)$ to $P_1$.

(2) $P_1$'s message
  - (a) Check the validation of $\pi_2$. Then, $\alpha' \leftarrow [0, q^2 2^{2s+t-1}]$, $r \leftarrow [0, N_2]$ and compute the affine operation
$$C_1 = \left(C_2 \cdot y_2^{2^{s+t}q}\right)^a y_2^{\alpha'} h_2^r \mod N_2.$$
  - (b) Generate range proof $\pi_1$ on the affine operation such that $a \in [0, q]$ and $\alpha' \in [0, q^2 2^{2s+t-1}]$ using $\mathsf{ZK}_{\mathsf{JL\text{-}aff}}$ in section 4.2.
  - (c) Send $(C_1, \pi_1)$ to $P_2$, and output $\alpha = -\alpha' \mod q$.

(3) $P_2$ checks the correctness of $\pi_1$, computes and outputs $\beta = \mathsf{JL.dec}(sk_2, C_1) \mod q$.

*Correctness.* By range proof, $b \in [-2^{s+t}q, 2^{s+t}q]$, and the plaintext of $C_1$ is upper bounded by

$$(b + 2^{s+t}q)a + \alpha' \le 2^{2s+2t+1}q^2 + 2^{3s+2t-1}q^2 < 2^{3s+2t}q^2.$$

Protocol is correct if there is no reduction modulo $2^k$, which means $k \ge 2 \log q + 3s + 2t$. At the same time, the strong JL assumption requires that $k \le 1/4 \log N - \lambda$. We suggest choosing $k = 2 \log q + 3s + 2t$ to give a good security margin.

*Remark.* Proof $\pi_2$ only guarantees $b \in [-2^{s+t}q, 2^{s+t}q]$. Thus, $P_1$ uses $C_2 \cdot y_2^{2^{s+t}q}$ rather than $C_2$ to ensure the plaintext is positive when generating $C_1$.

Theorem 4. *Under $k$-QR and strong JL assumptions, our MtA, illustrated in Figure 3, securely computes $\mathcal{F}_{\mathsf{MtA}}$ in the presence of a malicious static adversary under the ideal / real definition.*

Proof. In Setup phase, on or before receiving public parameters from the adversary indicating one participant, simulator $\mathcal{S}$ could simulates adversary's view via sampling a JL public key from public key space and appending zero-knowledge proofs via zero-knowledge simulators Sim.

We now need to handle the Multiplication phase. In ideal world, simulator $\mathcal{S}$ could only learn the public parameters and make queries to the ideal function $\mathcal{F}_{\mathsf{MtA}}$. In the real world, the adversary, having corrupted $P_1$ or $P_2$, will also see the interactions with non-corrupted party. Thus, $\mathcal{S}$ must simulate adversary's view of these interactions.

The proof proceeds in two cases: adversary $\mathcal{A}$ corrupts $P_2$, and $\mathcal{A}$ corrupts $P_1$.

**$\mathcal{S}$ simulates $P_1$-when $P_2$ is corrupted.** Simulator $\mathcal{S}$ receives the encryption and commitment pair $(C_2, c_2)$ with equality and range proofs $\pi_2$, that adversary $\mathcal{A}$ instructs $P_2$ to send with sid.

If the equality and range proofs are accepted, $\mathcal{S}$ could extracts $b$ from $\pi_2$ via the knowledge extractor of $\pi_2$. Then, $\mathcal{S}$ queries $\mathcal{F}_{\mathsf{MtA}}$ with $(\mathsf{sid}, b)$ and receives $(\mathsf{sid}, \beta)$ as the output of $P_2$. Then, $\mathcal{S}$ samples $r' \leftarrow [0, q2^{2s+t}]$, and sends $C_1$ as the encryption of $\beta + r'q$ to $\mathcal{A}$. A simulated proof $\pi_1$ is also appended via zero-knowledge simulator Sim.

The main difference between the simulation and a real execution is the generation of $C_1$ and zero-knowledge proof. Note that the distribution of $a(b + 2^{s+t}q) + \alpha'$ are $1/2^s$-statistical close when $\alpha' \leftarrow [0, q^2 2^{2s+t}]$. This implies that the view of a corrupted $P_2$ is the real execution is indistinguishable with that of the simulation.

**$\mathcal{S}$ simulates $P_2$-when $P_1$ is corrupted.** In session sid, $\mathcal{S}$ computes $C_2, c_2$ by encrypting and committing a random number, respectively. It also appends the equality and range proof $\pi_2$ via zero-knowledge simulator. Then, $\mathcal{S}$ receives $C_1$ with a range proof $\pi_1$ that $\mathcal{A}$ invokes $P_1$ to send out. $\mathcal{S}$ invokes the extractor of $\pi_1$ to extract $\mathcal{A}$'s input $a$ and $\alpha'$ (if the proof is accepted). The output of $\mathcal{A}$ could be computed as $-\alpha' \mod q$.

The difference is the generation of $C_2, c_2$ and proof $\pi_2$. By the security of JL encryption (which holds under $k$-QR assumption) and statistical hiding of commitment, any PPT adversary could not distinguish simulated $C_2, c_2$ from the real ciphertext and commitment. □

## 5.2 Batch JL-based MtAs

Suppose $P_1$ and $P_2$ would like to invoke $l$ MtAs with input vectors $\vec{a} = (a_i)_{1\le i\le l}$ and $\vec{b} = (b_i)_{1\le i\le l}$, and receives $\vec{\alpha} = (\alpha_i)_{1\le i\le l}$ and $\vec{\beta} = (\beta_i)_{1\le i\le l}$ respectively, such that $\alpha_i + \beta_i = a_i b_i \mod q$. Our batch technique mainly combines proving equality of $P_2$'s several ciphertexts to a single JL vector commitment.

**Setup.** $P_i$ runs JLv.setup to generate its public parameter $\mathsf{pp}_i$, i.e., $pk_i = (N_i, h_i, y_{i,1}, k)$ and $y_{i,2}, \cdots, y_{i,l} \leftarrow \mathbb{QNR}$. Furthermore, $P_i$ generates necessary zero-knowledge proof on the correctness of JL modulus $N_i$, $h_i \in \mathbb{QR}_{2^k}$, and $y_{i,j}^{2^k} \in < h_i^{2^k} >$ for $1 \le j \le l$. Each party $P_i$ takes its secret key $sk_i$ corresponding to $pk_i$ as private.

**Multiplication.**

(1) $P_2$'s message
   (a) Compute a vector of $l$ ciphertexts under its public key, i.e., $\{C_{2,i} = \mathsf{JL.enc}(pk_2, b_i)\}_{1\le i\le l}$.
   (b) Compute $c = \mathsf{JLv.commit}(\mathsf{pp}_1, \vec{b})$ under $P_1$'s public parameter.
   (c) Compute $\pi_2$ on equality of $b_i$s in $C_{2,i}$ and the vector commitment $c$ using $\mathsf{ZK}_{\mathsf{JLv\text{-}equ}}$ of section 4.3.
   (d) Send $(\{C_{2,i}\}_{1\le i\le l}, c, \pi_2)$ to $P_1$.

(2) $P_1$'s message
   (a) Check $\pi_2$. Choose $\alpha_i' \leftarrow [0, q^2 2^{2s+t-1}]$, $r_i \leftarrow [0, N_2]$ and compute affine function on every $C_{2,i}$ and get
   $$C_{1,i} = (C_{2,i} \cdot y_{2,i}^{2^{s+t}q})^{a_i} y_{2,i}^{\alpha_i'} h_2^{r_i} \mod N_2.$$
   (b) Compute proof $\pi_1$ consisting of $(\pi_{1,1}, \cdots, \pi_{1,l})$, where $\pi_{1,i}$ is the proof on affine operation of $C_{1,i}$ such that $a_i \in [0, q]$ and $\alpha_i' \in [0, q^2 2^{2s+t-1}]$.
   (c) Send $\{C_{1,i}\}_{1\le i\le l}$, $\pi_1$ to $P_2$.
   (d) Output $\{-\alpha_i' \mod q\}_{1\le i\le l}$.

(3) $P_2$ checks $\pi_1$, decrypts and outputs
   $$\{\beta_i = \mathsf{JL.dec}(sk_2, C_{1,i}) \mod q\}_{1\le i\le l}.$$

| MtAs | Communication | | Computation | |
|---|---|---|---|---|
| | $P_1$ | $P_2$ | $P_1$ | $P_2$ |
| Paillier [35] | $11.5N$ | $8.5N$ | 12E | 11E |
| JL | $3.5N$ | $6.5N$ | 5E | 6E |
| Batch JL | $3.5N$ | $(3.5 + 3/l)N$ | $(4 + 1/l)$E | $(4 + 2/l)$E |

**Table 2: Cost comparison of Multiplication in each MtA.**

# 6 COMPARISON

In this section, we benchmark and compare our JL-based MtA with previous OT-based, Paillier-based, and OT-based MtAs.

## 6.1 Theoretical Complexity

We analyse the theoretical complexity of our (batch) MtA and compare them with Paillier-based scheme in Table 2 and 3. In both tables, E represents a full exponentiation operation over $\mathbb{Z}_N$ (one full Paillier operation $\approx$ 2E). We note that our MtA needs more cost in Setup phase. Fortunately, it is a one-time shot.

**Comment on the difference with the published version.** Considering that not all exponentiation operations can be categorized as complete exponentiations, in order to establish a more accurate basis for comparison, it is imperative to precisely quantify the number of full exponentiation operations. For example, during the computation of $C_2 = \mathsf{JL.enc}(pk_2, b)$ as in Section 5.1, the total count of full exponentiation operations can be defined as $1 + \log p / \log N$. We also recalculate the communication cost.

**Single JL-based MtA.** In the multiplication phase, the message sent out by $P_1$ and $P_2$ are 3.5 $\mathbb{Z}_N$ and 6.5 $\mathbb{Z}_N$ respectively, and they

| MtA | Communication | Computation |
|---|---|---|
| Paillier [10, 29, 35] | $(6t+11)N$ | $(4t+15)\mathsf{E}$ |
| JL | $(12t+4)N$ | $(8t+6)\mathsf{E}$ |
| Batch JL | $(6(l+1)t+2l+2)N$ | $((4l+4)t+2l+4)\mathsf{E}$ |

**Table 3: Cost comparison of Setup phase in MtA. We do not count the cost of proving correctness of $N$ in all three cases.**

need to compute 5 and 6 full exponentiations, respectively. Cost in setup phase includes public key, and proofs of $\mathsf{ZK}_{\mathbb{QR}_{2^k}}$, $\mathsf{ZK}_{\mathbb{QR}\mathsf{2kdl}}$ and $\mathsf{ZK}_{\mathsf{JLmod}}$. When $\mathsf{ZK}_{\mathsf{JLmod}}$ is not taken into account, $(12t+4)$ $\mathbb{Z}_N$ bandwidth and $8t+6$ exponentiation are required, where $t$ is the sound parameter for zero-knowledge proof.

**Batch JL-based MtA.** In the multiplication phase, we could batch $P_2$'s proof of equality and range proofs to a JL vector commitment, which will reduce the total message sent by $P_2$ from $6.5l$ $\mathbb{Z}_N$ to $(3.5l+3)$ $\mathbb{Z}_N$, and the total computation from $5l$ (for $P_1$) + $6l$ (for $P_2$) fully exponential computation to $(4l+1)$ + $(4l+2)$. The batch technique increases the cost in the setup phase, since more $y_i$s are needed. When $\mathsf{ZK}_{\mathsf{JLmod}}$ is not taken into account, $(6(l+1)t+2l+2)$ $\mathbb{Z}_N$ elements and $(4l+4)t+2l+4$ exponentiation are required.

**Paillier-based MtA.** We evaluate the Paillier-based MtA abstracted by [44] from [10, 29, 35]. Paillier operation (resp. ciphertext) is counted to be approximately two JL exponentiation (resp. $\mathbb{Z}_N$ elements).

## 6.2 Benchmarking Results

We give a comprehensive implementation and comparison of JL, Paillier, CL and and OT based MtAs in Table 4, Figure 4 when computing 1, 10, 50, and 100 MtAs. We benchmark the implementation under three parameter settings, i.e, (security parameter $\lambda$, statistical parameter $s$, soundness parameter $t$, $k$) are $(128, 40, 40, 712)$, $(192, 80, 80, 1168)$, and $(256, 128, 128, 1682)$ respectively.

Our benchmark is done using Rust on a MacBook Pro 13-inch 2017 with Intel Core i5 @ 3.1 GHz CPU and 8 GB 2133 MHz LPDDR3 RAM running macOS Monterey v12.0.1. We evaluate the protocols on a laptop and do not take network latency into account, since these protocols have the same communication rounds. All the benchmark were taken over curves secp256k1, secp384r1, secp512r1 (as recommended by NIST [33]), and SHA256, SHA384, SHA512 are used to instantiate hash functions, to achieve $\lambda = 128, 192, 256$ security respectively. We implement Paillier-based, and CL-based MtAs based on the elementary codes of [43] and [45], respectively. In OT-based MtA, we implement the curve operations using OpenSSL 3.0.2 15.

When $\lambda = 192, 256$, Paillier / JL based schemes use $N$ and ECDSA group order $q$ of same size, i.e., $\log N = 7680, \log q = 384$, and $\log N = 15360, \log q = 521$ as recommended by NIST [4]. When $\lambda = 128$, $\log q = 256$, and Paillier uses 3072-bits modulus $N$, while (Batch) JL needs 3360-bits $N$ (due to the requirement $2\log q + 3s + 2t < k \leq 1/4 \log N - \lambda$). Nevertheless, when $\lambda = 128$, our scheme also outperforms Paillier-based MtA. Parameters for CL scheme are chosen according to [11, Sec. 5]

When $\lambda = 128$, our JL-based MtA improves Paillier-based MtA in bandwidth by a factor of 1.85, in computation by a factor of $> 1.2$. When $\lambda = 128$ and the batch number $l > 10$, our scheme improves Paillier-based MtA in bandwidth by a factor of 2.4 to 2.7, in computation by a factor of $> 1.62$.

When $\lambda = 192$ or 256, our JL based MtA outperforms the Paillier-based MtA by a factor of roughly 2 in communication, and by a factor of $\approx 1.7$ in computation. When $\lambda = 192$ or 256, and the batch number $l > 10$, batch JL improves Paillier-based MtA by a factor of $> 2.68$ in communication, and by a factor of $> 2.26$ in computation.

Table 5 presents the cost comparison of Setup phase between OT-based CL-based, Paillier-based and our (batch) JL-based MtAs. When proving correctness of $N$ is taken into account, Setup of JL scheme needs $\times 2 \sim 3$ cost than that of Paillier-based MtA. When preparing for batch JL (e.g. batching 10 JL MtA), the setup requires more complexity.

# 7 APPLICATION IN THRESHOLD ECDSA

Our JL-based MtA could be directly plugged in many threshold ECDSAs from Paillier, e.g., LN18 [35], CGG+20 [10], and XAX+21 [44]. The definitions of ECDSA signature and the functionality of threshold ECDSA are given in Appendix F.

We test the JL-based MtA in LN18 [35] and XAX+21 [44] for 128, 192, and 256 bits security respectively, and compare their performance with OT-based, CL-based, and Paillier-based schemes in Table 6. The benchmarks were taken over curves scep256k1, scep384r1 and secp521r1 respectively. Elliptic curve operations in OT-based schemes DKLs18 [25], DKLs19 [26], and XAX+21 [44] utilize OpenSSL 3.0.2 15. We implement CL-based schemes CCL+19 [12], XAX+21 [44] based on the code of [45], and Paillier-based schemes LN18 [35], CGG+20 [10], XAX+21 [44] based on ZenGo [43] respectively.

We benchmark threshold ECDSAs in the special two-party case. Since threshold ECDSA involving $n$ parties needs $n(n-1)+n/2$ computation of MtA (e.g., LN18 [35]), results would favor ours when more parties are involved. According to their costruction, LN18 [35] needs approximately 3 MtAs and XAX+21 [44] requires a single MtA. We evaluate the protocols by only considering the computation time. Latency is not taken into account.

# 8 CONCLUSION

We propose an efficient MtA from Joye-Libert cryptosystem. Benchmark shows that multiplication of our scheme outperforms state-of-the-art constructions. Our MtA can be applied to improve existing threshold ECDSAs.

The building blocks that we develop, namely, a JL-based commitment scheme and its companion zero-knowledge proofs, may be of independent interest. They can be used to improve multi-party computation over $\mathbb{Z}_{2^k}$, and build more efficient three-party TLS handshake.

# 9 ACKNOWLEDGEMENT

Haiyang Xue is supported by the National Natural Science Foundation of China (No. 62172412), and would like to thank LatticeX Fundation and Huawei for their support. Man Ho Au is supported by the Research Grant Council of Hong Kong (No. 17201421, 15211120,

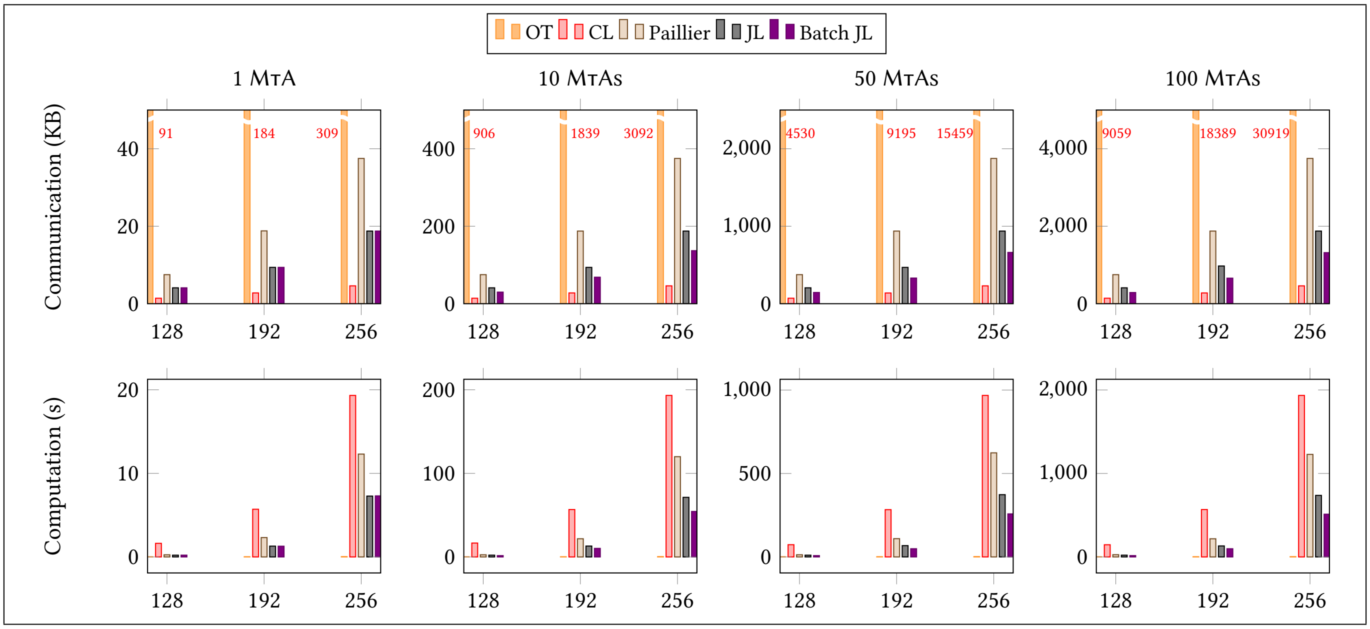


**Figure 4: Bandwidth and computational comparison of Multiplications between CL, Paillier, (batch) JL based MtAs. The communication is counted in KB. The computational cost is counted in seconds. $x$-axis indicates the security parameter $\lambda$ which fits the settings $(\lambda, s, t, k) = (128, 40, 40, 712)$, $(192, 80, 80, 1168)$, or $(256, 128, 128, 1682)$ respectively.**

| MtA Schemes | | Communication (KB) | | | | Computation (ms) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $l = 1$ | $l = 10$ | $l = 50$ | $l = 100$ | $l = 1$ | $l = 10$ | $l = 50$ | $l = 100$ |
| $\lambda = 128$ | OT [25] | 90.6 | 905.9 | 4529.7 | 9059.4 | 13.6 | 149.1 | 652.4 | 1326.6 |
| | CL [12] | 1.41 | 14.1 | 70.6 | 141.2 | 1625 | 16610 | 73230 | 145902 |
| | Paillier [10, 35, 44] | 7.5 | 75 | 375 | 750 | 255 | 2587 | 12849 | 26750 |
| | JL | 4.10 | 41.0 | 205.1 | 410.1 | 209 | 2136 | 10392 | 21047 |
| | Batch JL | 4.10 | 29.9 | 144.8 | 288.3 | 212 | 1574 | 7782 | 15872 |
| $\lambda = 192$ | OT [25] | 183.9 | 1838.9 | 9194.5 | 18389.1 | 16.8 | 190.6 | 916.8 | 1908.2 |
| | CL [12] | 2.78 | 27.8 | 138.3 | 277.6 | 5705 | 56715 | 283261 | 567725 |
| | Paillier [10, 35, 44] | 18.8 | 187.5 | 937.5 | 1875.0 | 2317 | 21766 | 109448 | 217429 |
| | JL | 9.38 | 93.75 | 468.8 | 973.5 | 1292 | 13036 | 67694 | 131791 |
| | Batch JL | 9.38 | 68.4 | 330.9 | 659.1 | 1283 | 10239 | 49069 | 98047 |
| $\lambda = 256$ | OT [25] | 309.2 | 3091.9 | 15459.4 | 30918.8 | 31.6 | 357.7 | 1578.0 | 2703.6 |
| | CL [12] | 4.61 | 46.1 | 230.4 | 460.8 | 19324 | 193233 | 967834 | 1934574 |
| | Paillier [10, 35, 44] | 37.5 | 375 | 1875 | 3750 | 12305 | 119950 | 623747 | 1228064 |
| | JL | 18.75 | 187.5 | 937.5 | 1875 | 7279 | 71303 | 373088 | 737083 |
| | Batch JL | 18.75 | 136.9 | 661.9 | 1318.1 | 7304 | 54414 | 257869 | 511789 |

**Table 4: Comparison of Multiplication in MtAs. $l$ is the number of MtAs.**

| MtAs | | Communication (KB) | Computation (ms) |
|---|---|---|---|
| OT | $\lambda = 128$ | 40 | 268.9 |
| | $\lambda = 192$ | 90.5 | 837.8 |
| | $\lambda = 256$ | 160.7 | 836.5 |
| CL | $\lambda = 128$ | 5.5 | 22941 |
| | $\lambda = 192$ | 10.5 | 168322 |
| | $\lambda = 256$ | 17.2 | 1105557 |
| Paillier | $\lambda = 128$ | 94.1 | 952 |
| | $\lambda = 192$ | 460.3 | 13324 |
| | $\lambda = 256$ | 1460.6 | 142103 |
| JL | $\lambda = 128$ | 198.5 | 2432 |
| | $\lambda = 192$ | 937.5 | 25936 |
| | $\lambda = 256$ | 2887.5 | 283227 |
| Batch 10 JL | $\lambda = 128$ | 1020.8 | 11169 |
| | $\lambda = 192$ | 4970.6 | 95041 |
| | $\lambda = 256$ | 15881.2 | 708508 |

**Table 5: Cost comparison of Setup phase in MtA. Setup of Batch 10 JL supports 10 batches.**

R1012-21), the National Natural Science Foundation of China (No. 61972332). Tsz Hon Yuen is supported by Huawei Shield Lab.

## REFERENCES


[1] Damiano Abram, Ariel Nof, Claudio Orlandi, Peter Scholl, and Omer Shlomovits. 2022. Low-bandwidth threshold ECDSA via pseudorandom correlation generators. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2554–2572.

[2] Benedikt Auerbach and Bertram Poettering. 2018. Hashing solutions instead of generating problems: On the interactive certification of RSA moduli. In *IACR International Workshop on Public Key Cryptography*. Springer, 403–430.

[3] Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. 2020. A Survey of ECDSA Threshold Signing. (2020). https://eprint.iacr.org/2020/1390.pdf.

[4] Elaine Barker, Elaine Barker, William Burr, William Polk, Miles Smid, et al. 2006. *Recommendation for key management: Part 1: General.* National Institute of

| Threshold ECDSA | | $\lambda = 128$ | | | | $\lambda = 192$ | | | | $\lambda = 256$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Communication (KB) | | Computation (ms) | | Communication (KB) | | Computation (ms) | | Communication (KB) | | Computation (ms) | |
| | | $l = 1$ | $l = 10$ | $l = 1$ | $l = 10$ | $l = 1$ | $l = 10$ | $l = 1$ | $l = 10$ | $l = 1$ | $l = 10$ | $l = 1$ | $l = 10$ |
| DKLs18 [25] | OT | 232.8 | 2328.4 | 36.4 | 249.1 | 481.3 | 4812.6 | 45.7 | 390.7 | 817.7 | 817.69 | 60.8 | 577.5 |
| DKLs19 [26] | OT | 176.5 | 1765.6 | 39.8 | 325.2 | 342.8 | 3428.4 | 61.8 | 473.6 | 561.1 | 5611.2 | 76.8 | 788.9 |
| CCL+20 [12] | CL | 6.3 | 6.34 | 4658 | 46723 | 11.3 | 113.1 | 14224 | 143650 | 17.6 | 176.9 | 39223 | 394364 |
| CGG+20 [10] | Paillier | 44.2 | 442.4 | 2384 | 23734 | 111.3 | 1112.5 | 19855 | 198348 | 222.5 | 2225 | 102299 | 1039845 |
| LN18 [35] | Paillier | 26.4 | 264.1 | 880 | 8821 | 59.8 | 598.4 | 7054 | 70493 | 126.2 | 1262.2 | 37510 | 374593 |
| LN18 by using our MtA | JL | 15.7 | 156.7 | 735 | 7374 | 32.4 | 323.6 | 4407 | 44327 | 61.4 | 614.0 | 22876 | 228047 |
| | Batch JL | 15.7 | 134.7 | 731 | 5988 | 32.4 | 250.9 | 4452 | 34855 | 61.4 | 445.1 | 22784 | 175593 |
| XAX+21 [44] | OT | 90.9 | 909.4 | 14.8 | 141.6 | 184.4 | 1844.1 | 19.1 | 201.0 | 309.9 | 3098.7 | 35.5 | 378.5 |
| | CL | 1.8 | 186.5 | 1767 | 17541 | 3.45 | 34.6 | 5794 | 58045 | 5.5 | 55.3 | 19805 | 198560 |
| | Paillier | 7.8 | 78.1 | 258 | 2563 | 18.7 | 186.6 | 2486 | 24604 | 38.4 | 384.4 | 12763 | 127394 |
| XAX+21 by using our MtA | JL | 4.49 | 44.9 | 214 | 2146 | 10.1 | 100.5 | 1313 | 13117 | 21.6 | 216.9 | 6858 | 68493 |
| | Batch JL | 4.49 | 33.8 | 218 | 1598 | 10.1 | 75.1 | 1310 | 10036 | 21.6 | 164.4 | 6891 | 49061 |

**Table 6: Comparison of threshold ($2$-out-of-$n$) ECDSAs.**


Standards and Technology, Technology Administration.

[5] Carsten Baum, Daniel Escudero, Alberto Pedrouzo-Ulloa, Peter Scholl, and Juan Ramón Troncoso-Pastoriza. 2020. Efficient Protocols for Oblivious Linear Function Evaluation from Ring-LWE. In *SCN*. Springer, 130–149.

[6] Fabrice Benhamouda, Houda Ferradi, Rémi Géraud, and David Naccache. 2017. Non-interactive provably secure attestations for arbitrary RSA prime generation algorithms. In *European Symposium on Research in Computer Security*. Springer, 206–223.

[7] Fabrice Benhamouda, Javier Herranz Sotoca, Marc Joye, and Benoit Libert. 2017. Efficient cryptosystems from $2^k$-th power residue symbols. *Journal of cryptology* 30, 2 (2017), 519–549.

[8] Fabrice Boudot. 2000. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*. Springer, 431–444.

[9] Jan Camenisch and Markus Michels. 1999. Proving in zero-knowledge that a number is the product of two safe primes. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 107–122.

[10] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. 2020. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1769–1787.

[11] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2019. Two-party ECDSA from hash proof systems and efficient instantiations. In *Annual International Cryptology Conference*. Springer, 191–221.

[12] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. 2020. Bandwidth-efficient threshold EC-DSA. In *IACR International Conference on Public-Key Cryptography*. Springer, 266–296.

[13] Guilhem Castagnos and Fabien Laguillaumie. 2015. Linearly homomorphic encryption from DDH. In *CT-RSA*. Springer, 487–505.

[14] Dario Catalano, Mario Di Raimondo, Dario Fiore, and Irene Giacomelli. 2020. Mon$\mathbb{Z}_{2^k}$a: Fast Maliciously Secure Two Party Computation on $\mathbb{Z}_{2^k}$. In *IACR International Conference on Public-Key Cryptography*. Springer, 357–386.

[15] Tung Chou and Claudio Orlandi. 2015. The simplest protocol for oblivious transfer. In *International Conference on Cryptology and Information Security in Latin America*. Springer, 40–58.

[16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. 2017. Removing the strong RSA assumption from arguments over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 321–350.

[17] Ronald Cramer. 1996. Modular design of secure yet practical cryptographic protocols. *Ph. D. Thesis, CWI and University of Amsterdam* (1996).

[18] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. 2018. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod $2^k$ for Dishonest Majority. In *Advances in Cryptology–CRYPTO*.

[19] William M Daley and Raymond G Kammer. 2000. *Digital signature standard (DSS)*. Technical Report. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

[20] Ivan Damgård and Mads Jurik. 2002. Client/server tradeoffs for online elections. In *International Workshop on Public Key Cryptography*. Springer, 125–140.

[21] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. 2013. Practical covertly secure MPC for dishonest majority–or: breaking the SPDZ limits. In *European Symposium on Research in Computer Security*. Springer, 1–18.

[22] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*. Springer, 643–662.

[23] Yi Deng, Shunli Ma, Xinxuan Zhang, Hailong Wang, Xuyang Song, and Xiang Xie. 2021. Promise Sigma-Protocol: How to Construct Efficient Threshold ECDSA from Encryptions Based on Class Groups. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 557–586.

[24] Yvo Desmedt and Yair Frankel. 1989. Threshold cryptosystems. In *Conference on the Theory and Application of Cryptology*. Springer, 307–315.

[25] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. 2018. Secure two-party threshold ECDSA from ECDSA assumptions. In *IEEE Symposium on Security and Privacy*. IEEE, 980–997.

[26] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. 2019. Threshold ECDSA from ECDSA assumptions: the multiparty case. In *IEEE Symposium on Security and Privacy*. IEEE, 1051–1066.

[27] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*. Springer, 186–194.

[28] Eiichiro Fujisaki and Tatsuaki Okamoto. 1997. Statistical zero knowledge protocols to prove modular polynomial relations. In *Annual International Cryptology Conference*. Springer, 16–30.

[29] Rosario Gennaro and Steven Goldfeder. 2018. Fast multiparty threshold ECDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1179–1194.

[30] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. 2017. Maliciously secure oblivious linear function evaluation with constant overhead. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 629–659.

[31] Marc Joye and Benoît Libert. 2013. Efficient cryptosystems from $2^k$-th power residue symbols. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 76–92.

[32] Marcel Keller, Emmanuela Orsini, and Peter Scholl. 2015. Actively secure OT extension with optimal overhead. In *Annual Cryptology Conference*. Springer, 724–741.

[33] C Kerry and P Gallagher. 2013. FIPS PUB 186-4: Digital Signature Standard (DSS). *Federal Information Processing Standards Publication. National Institute of Standards und Technology* (2013).

[34] Yehuda Lindell. 2017. Fast secure two-party ECDSA signing. In *Annual International Cryptology Conference*. Springer, 613–644.

[35] Yehuda Lindell and Ariel Nof. 2018. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1837–1854. Refer https://eprint.iacr.org/2018/987.pdf for the full version..

[36] Philip MacKenzie and Michael K Reiter. 2001. Two-party generation of DSA signatures. In *Annual International Cryptology Conference*. Springer, 137–154.

[37] Moni Naor and Moti Yung. 1990. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 427–437.

[38] NIST. 2022. Multi-Party Threshold Cryptography. https://csrc.nist.gov/Projects/threshold-cryptography.

[39] NIST. 2023. IR 8214C (Initial Public Draft), First Call for Multi-Party Threshold Schemes. https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8214C.ipd.pdf.

[40] Pascal Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of*

*cryptographic techniques*. Springer, 223–238.
[41] Torben Pryds Pedersen. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*. Springer, 129–140.
[42] Dmytro Tymokhanov and Omer Shlomovits. 2021. Alpha-rays: Key extraction attacks on threshold ecdsa implementations. *Cryptology ePrint Archive* (2021).
[43] ZenGo X. 2021. multi-party-ecdsa. https://github.com/ZenGo-X/multi-party-ecdsa.
[44] Haiyang Xue, Man Ho Au, Xiang Xie, Tsz Hon Yuen, and Handong Cui. 2021. Efficient Online-friendly Two-Party ECDSA Signature. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 558–573.
[45] Tsz Hon Yuen, Handong Cui, and Xiang Xie. 2021. Compact zero-knowledge proofs for threshold ECDSA with trustless setup. In *IACR International Conference on Public-Key Cryptography*. Springer, 481–511.
[46] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. 2020. Deco: Liberating web data using decentralized oracles for tls. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1919–1938.

# A STRONG RSA / JL ASSUMPTIONS

## A.1 Strong RSA assumption

DEFINITION 4 (STRONG RSA ASSUMPTION [28]). *Let $N$ be the RSA modulus. The strong RSA assumption states that, for a random element $x \in \mathbb{Z}_N^*$, it is hard to find the e-th root $y$ modulo $N$, i.e., $y^e = x \mod N$, for any PPT algorithm and an exponent $e > 1$ of its choice.*

The following assumption is useful for proving Theorem 1.

DEFINITION 5 (STRONG RSA* ASSUMPTION). *Let $N$ be the RSA modulus, $\mathbb{QNR}$ be the set of quadratic non-residuosity. The strong RSA* assumption states that, for a random element $x \in \mathbb{QNR}$, it is hard to find the e-th root $y$ modulo $N$, i.e., $y^e = x \mod N$, for any PPT algorithm and an exponent $e > 1$ of its choice.*

Let $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}}$ (resp. $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}^*}$) be algorithm $\mathcal{A}$'s advantage of solving Strong RSA problem (resp. Strong RSA* problem). We have $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}^*} \leq 4\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}}$ since that: $|\mathbb{Z}_N^*| = 4|\mathbb{QNR}|$ and an algorithm finding $e$-th roof of $x \in \mathbb{QNR}$ with probability $\epsilon$ could be trivially transferred to find $e$-th roof of $x \in \mathbb{Z}_N^*$ with probability at least $\epsilon/4$.

## A.2 Theorem 1: $k$-QR+strong RSA $\Rightarrow$ strong JL

Given an algorithm $\mathcal{A}$ solving the strong JL problem, we could construct an algorithm $\mathcal{B}$ distinguishing element of $\mathbb{QNR}$ from that of $\mathbb{QR}_{2^k}$, which conflict the $k$-QR assumption (according to Lemma 1). Specifically, given an input $(N, x, k)$, $\mathcal{B}$ just feeds it to $\mathcal{A}$. If an $e$-th roof of $x$ is outputted, returns 1 as the guess that $x \in \mathbb{QR}_{2^k}$, otherwise, return 0.

We have

$$\Pr[\mathcal{B}(x,k) = 1 | x \leftarrow \mathbb{QR}_{2^k}] - \Pr[\mathcal{B}(x,k) = 1 | x \leftarrow \mathbb{QNR}]$$
$$= \mathsf{Adv}_{\mathcal{A}}^{\mathrm{sJL}} - \mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}^*}.$$

Thus, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sJL}} = \mathsf{Adv}_{\mathcal{B}}^{\text{Gap-}2^k} + \mathsf{Adv}_{\mathcal{A}}^{\mathrm{sRSA}^*}$. By Lemma 1 and Appendix A.1, we conclude that given any algorithm $\mathcal{A}$, there exists algorithms $C$ and $\mathcal{E}$ such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sJL}} \leq 3/2(k - 1/3)\mathsf{Adv}_{C}^{\mathrm{kQR}} + 4\mathsf{Adv}_{\mathcal{E}}^{\mathrm{sRSA}}.$$

# B PROOF OF THEOREM 3

The correctness is obvious. Hiding property comes from the facts that $h^{2^k}$ is the generators of $\mathbb{QR}_{2^k}$ (whose order is $p'q'$), and $y_i^{2^k} \in \mathbb{QR}_{2^k}$ $(1 \leq i \leq l)$. For $1 \leq i \leq l$, there exists $\alpha_i$ s.t. $y_i^{2^k} = h^{2^k \alpha_i} \mod N$. Thus,

$$y_1^{2^k m_1} \cdots y_l^{2^k m_l} h^{2^k r} = h^{2^k (\sum_{i=1}^{l} \alpha_i m_i + r)} \mod N.$$

Computational binding relies on the strong JL and $k$-QR assumptions. Given an instance $(N, h, k)$ of strong JL problem, the strong JL solver generates $y_i = h^{\alpha_i} \mod N$ for $\alpha_i \leftarrow \mathbb{Z}_N$ and sets $(N, h, y_1, \cdots, y_l, k)$ as public parameter of JL commitment. The only difference is the generation of $y_i$ elements (i.e., $y_i \in \mathbb{QNR}$ or $y_i \in \mathbb{QR}_{2^k}$). The committer could not find this difference due to the $k$-QR assumption (according to Lemma 1). From two different openings $(m_1, \cdots, m_l, d)$ and $(m'_1, \cdots, m'_l, d')$ of $c$, the verification check guarantees that

$$y_1^{2^k m_1} \cdots y_l^{2^k m_l} h^{2^k d} = y_1^{2^k m'_1} \cdots y_l^{2^k m'_l} h^{2^k d'} \mod N.$$

Denote $\Delta m_i = m_i - m'_i$ (for every $1 \leq i \leq l$), $\Delta d = d - d'$. We have $h^{2^k (\sum_{i=1}^{l} \alpha_i \Delta m_i + \Delta d)} = 1 \mod N$ since $y_i^{2^k} = h^{2^k \alpha_i} \mod N$. Let $e > 1$ be any number co-prime to $E = 2^k (\sum_{i=1}^{l} \alpha_i \Delta m_i + \Delta d)$. We have

$$\left(h^{t^{-1} \mod E}\right)^e = h \mod N,$$

which gives a solution of the strong JL problem.

# C PROOF OF OPENING

## C.1 Proof of knowledge for $\mathsf{ZK}_{\text{JL-com}}$.

This could be taken as an extension of the proof of knowledge for RSA commitment from [16]. Here, we should handle a different structure, and we do not aim to eliminate the strong JL assumption.

We assume the knowledge extractor Ext is given $(N, h, k)$ as an instance of strong JL problem. In setup for JL commitment, Ext sets $y = h^{\alpha} \mod N$ for a random $\alpha \leftarrow \mathbb{Z}_N$, rather than $y \leftarrow \mathbb{QNR}$ in the real execution. Ext sets $(N, h, y, k)$ as the public parameter of JL commitment. Any PPT adversary could not figure out this difference due to the $k$-QR assumption (according to Lemma 1). We consider an adversary $P^*$ who provides a convincing proof of knowledge of opening for commitment $c$ with probability $\epsilon$ under the parameter $(N, h, y, k)$.

The standard rewind technique on Sigma protocol would allow us to obtain two accepted challenge-respond pairs $(e; z_m, z_r)$, $(e'; z'_m, z'_r)$, for a given committed $d$, with probability grater than $\epsilon^2/4$ in expected polynomial time. Assume $e > e'$ (w.l.o.g) and denote $\Delta e = e - e'$, $\Delta m = z_m - z'_m$, and $\Delta r = z_r - z'_r$. We would have

$$c^{\Delta e} = y^{2^k \Delta m} h^{2^k \Delta r} \mod N. \quad (1)$$

Applying the rewind technique, one of two cases in the following would happen.

- Case 1: $\Delta e \mid \Delta m \wedge \Delta e \mid \Delta r$ with probability greater than $\epsilon^2/8$.
- Case 2: $\Delta e \nmid \Delta m \vee \Delta e \nmid \Delta r$ with probability greater than $\epsilon^2/8$.

**In Case 1**, if $\Delta e$ is odd, there is $c = (y^{2^k})^{\Delta m/\Delta e} h^{2^k \Delta r/\Delta e} \mod N$ (since $\Delta e \leq 2^t$ is co-prime to $p'q'$). Thus, $(\Delta m/\Delta e, \Delta r/\Delta e)$ is a valid opening; if $\Delta e = 2^v \rho$ for an odd $\rho$ and $v \geq 1$, it means $2^v \rho \mid \Delta m$ and $2^v \rho \mid \Delta r$. Let $\Delta m = 2^v \Delta m'$, $\Delta r = 2^v \Delta r'$. Then, we have that $c^{2^v} = (y^{2^k})^{2^v \Delta m'/\rho} (h^{2^k})^{2^v \Delta r'/\rho} \mod N$ (since $\rho$ is co-prime to

$p'q'$). Define $u = (y^{2^k})^{\Delta m'/\rho}(h^{2^k})^{\Delta r'/\rho}c^{-1} \mod N$, then $u^{2^v} = 1 \mod N$. It falls into the following sub-cases.

(1) $v = 1$.
  (a) $u = -1$. This would never happen since $J_N(u) = J_N(c) = 1$ and $J_N(-1) = -1$.
  (b) $u = 1$. This indicates a valid opening.
  (c) $u \neq 1$. From 4) of Fact 1, this leads to factoring $N$.

(2) $v \geq 2$.
  (a) $u^{2^{v-1}} = -1$. This would never happen since $J_N(u^{2^{v-1}}) = 1$ and $J_N(-1) = -1$.
  (b) $u^{2^{v-1}} = 1$. This reduces to the case $u^{2^{v-1}} = 1 \mod N$, which could be analysed via recursion.
  (c) $u^{2^{v-1}} \neq \pm 1$. From 4) of Fact 1, it leads to factoring $N$.

Thus, Case 1 would give valid opening of $c$ or solution to the strong JL problem (by factoring $N$).

**For Case 2**, by the generation of $y$, (1) could be rewritten as

$$c^{\Delta e} = h^{2^k \alpha \Delta m + 2^k \Delta r} \mod N \tag{2}$$

If $\Delta e = 2^v \rho$ for $v \geq 1$ and some odd number $\rho$,

$$[c^{-\rho} h^{2^{k-v}(\alpha\Delta m + \Delta r)}]^{2^v} = 1 \mod N.$$

As the analysis in Case 1 (for the sub-cases), by recursion and under the factoring assumption, we have

$$c^{\rho} = h^{2^{k-v}(\alpha \Delta m + \Delta r)} \mod N. \tag{3}$$

If $\Delta e = \rho$ for some odd number $\rho$, i.e., $v = 0$, (3) still hold.

Since odd number $\rho \mid \Delta e$, we have $\rho \nmid \Delta m \vee \rho \nmid \Delta r$ with probability greater than $\epsilon^2/8$.

We divide Case 2 into two sub-cases, accordingly.

- Case 2.1: $\rho \nmid \alpha\Delta m + \Delta r$.
- Case 2.2: $\rho \mid \alpha\Delta m + \Delta r$.

We first argue that, in Case 2.1, we could solve the strong JL problem. Then we prove the fact that, for any unbounded adversary, Case 2.2 happens with probability less than 1/3. Thus, Case 2.1 happens with probability $> 2/3$, which will help us to solve the strong JL problem.

**In Case 2.1**, let $\beta = gcd(\rho, 2^{k-v}(\alpha\Delta m + \Delta r))$. There exists efficient algorithm to find $f, g$ such that $\beta = f\rho + g(2^{k-v}(\alpha\Delta m + \Delta r))$. Then, according to (3),

$$h^{\beta} = h^{f\rho + g(2^{k-v}(\alpha\Delta m + \Delta r))} = (h^f c^g)^{\rho} \mod N.$$

From the fact that $\beta$ is co-prime to $p'q'$,

$$h = (h^f c^g)^{\rho/\beta} \mod N \tag{4}$$

Due to odd $\rho \nmid \alpha\Delta m + \Delta r$, we have $\rho > \beta$ and $\rho/\beta > 1$. Thus, (4) gives us a solution of the strong JL problem.

**We now handle Case 2.2.** Let $\rho_i$ be the prime factor of $\rho$ and $j$ be the integer such that $(\rho_i)^j \mid \rho$ and $(\rho_i)^{j+1} \nmid \rho$. If $(\rho_i)^j \mid \Delta m$, then $(\rho_i)^j \mid \Delta r$, since $\rho \mid \alpha\Delta m + \Delta r$. Thus,

$$(\rho_i)^j \nmid \Delta m. \tag{5}$$

Note that $\alpha$ could be written as $\alpha \mod p'q' + \gamma p'q'$ for some totally random $\gamma \in [0, 2^{k+1}]$ (from the view of the adversary). For the prime factor $\rho_i$ in (5), we have $(\rho_i)^j \mid \alpha\Delta m + \Delta r$, i.e.,

$$(\rho_i)^j \mid \gamma p'q'\Delta m + (\alpha \mod p'q')\Delta m + \Delta r. \tag{6}$$

For this $\rho_i$ and totally random $\gamma$, (6) satisfies with probability less than $1/\rho_i \leq 1/3$ (due to $\rho_i > 2$).

## C.2 Opening proof of JL vector commitment: $\mathsf{ZK}_{\text{JLv-com}}$

The completeness is trivial. The protocol is honest-verifier zero knowledge since simulator Sim just does as follows: Sim chooses random responds $z_{m_i} \leftarrow [0, 2^{s+t}B_i]$ (for $1 \leq i \leq l$), $z_r \leftarrow [0, 2^{s+t}N]$, together with $e \leftarrow \{0,1\}^t$. Sim sets $d = \prod_{i=1}^{l} y_i^{2^k z_{m_i}} h^{2^k z_r} c^{-e} \mod N$.

Proof of knowledge is an extension of that for $\mathsf{ZK}_{\text{JL-com}}$. We could get $c^{\Delta e} = y_1^{2^k \Delta z_1} \cdots y_l^{2^k \Delta z_l} h^{2^k \Delta r} \mod N$ from two accepted transcripts. Assume $y_i^{2^k} = y^{2^k s_i} h^{2^k t_i} \mod N$ for random $s_i$, $t_i$ (and $y = h^{\alpha} \mod N$ for random $\alpha \in \mathbb{Z}_N$). Then, we have $c^{\Delta e} = y^{2^k(\sum s_i \Delta z_i)} h^{2^k(\Delta r + \sum t_i \Delta z_i)} \mod N$. Thus, under strong JL assumption, $\Delta e$ must divide $\sum s_i \Delta z_i$ and $\Delta r + \sum t_i \Delta z_i$ from analysis for knowledge soundness of $\mathsf{ZK}_{\text{JL-com}}$ (refer to Appendix C.1). Furthermore, due to the randomness of $s_i, t_i$, $\Delta e$ must divide all $\Delta z_i$ and $\Delta r$, which gives a valid opening.

The proof guarantees that every $m_i \in [-2^{s+t}B_i, 2^{s+t}B_i]$, since for any $m_i$ satisfying $|m_i| > 2^{s+t}B_i$ and a random chosen $e$, the probability of guessing the right $v_i$ such that $em_i + v_i \in [0, 2^{s+t}B_i]$ is less then $1/2^t$.

# D PROOF OF EQUALITY.

## D.1 Knowledge soundness of $\mathsf{ZK}_{\text{JL-equ}}$

The completeness and honest verifier zero-knowledge are obvious. We only need to argue proof of knowledge. As proof of opening, let $(e; z_m, z_R, z_r)$, $(e'; z'_m, z'_R, z'_r)$ be two accepting transcripts. Let $\Delta e = e - e'$, $\Delta m = z_m - z'_m$, $\Delta R = z_R - z'_R$, and $\Delta r = z_r - z'_r$. Under strong JL and $k$-QR assumptions, as in the proof-of-knowledge analysis for $\mathsf{ZK}_{\text{JL-com}}$ (refer to Appendix C.1), we have $\Delta e \mid \Delta m$ and $\Delta e \mid 2^k \Delta r$, and could extract $m = \Delta m/\Delta e$ and $2^k \Delta r/\Delta e$ such that $c = y^{2^k m} h^{2^k \Delta r/\Delta e} \mod N$.

We also have $C^{\Delta e} = y_0^{\Delta m} h_0^{\Delta R} \mod N_0$. Since $\Delta e \mid \Delta m$, we could extract an $m = \Delta m/\Delta e$ such that $C = y_0^m h_0^x$ for some $x$. (Actually, since $\Delta e \leq 2^t$ is co-prime to $p'_0 q'_0$, $x$ is some value satisfying $x = \Delta R \Delta e^{-1} \mod p'_0 q'_0$.)

## D.2 Batch proof of equality to a JL vector commitment $\mathsf{ZK}_{\text{JLv-equ}}$

The prover would like to prove the relation

$$\begin{aligned}\mathcal{R}_{\text{JLv-equ}} = \{(\vec{C}, c; \vec{m}) \mid c = y_1^{2^k m_1} \cdots y_l^{2^k m_l} h^{2^k r} \mod N,\\ C_i = y_0^{m_i} h_0^{r_{0,i}} \mod N_0, m_i \in [0, B_i], 1 \leq i \leq l\}.\end{aligned}$$

The protocol $\mathsf{ZK}_{\text{JLv-equ}}$ runs as follows.

- $\mathcal{P}$ chooses random $v_1, \cdots, v_l$ from $[0, 2^{s+t}B_i]$, and random $w_{0,1}, \cdots, w_{0,l}, w$ from $[0, 2^{s+t}N]$. $\mathcal{P}$ computes and sends $\{D_i = y_0^{v_i} h_0^{w_{0,i}} \mod N_0\}_{1 \leq i \leq l}, d = \prod_{i=1}^{l} y_i^{2^k v_i} \cdot h^{2^k w} \mod N$ to $\mathcal{V}$.
- $\mathcal{V}$ chooses and sends $e \leftarrow \{0,1\}^t$ to $\mathcal{P}$.
- $\mathcal{P}$ computes and sends $z_{m,i} = em_i + v_i$, $z_{R,i} = er_{0,i} + w_{0,i}$ for $1 \leq i \leq l$, and $z_r = er + w$ (as integers) to $\mathcal{V}$.

KGen : On receiving KGen($1^\kappa$) from all parties $P_1, \cdots, P_n$

- Choose a random $x \leftarrow \mathbb{Z}_q$ and compute $Q = x \cdot P$. Set $(Q, x)$ as an ECDSA key pair and store $(\mathbb{G}, H, P, Q, x)$.
- Send $(H, Q)$ to all $P_1, \cdots, P_n$.
- Ignore any further call to KGen.

Sign : On receiving Sign(sid, $m$) from all parties, if KGen was already called and sid has not been used before:

- Choose a random $k \leftarrow \mathbb{Z}_q$, compute $R = (r_x, r_y) = k \cdot P$.
- Compute $r = r_x \mod q$ and $s = k^{-1}(H(m) + rx) \mod q$.
- Send $(r, s)$ to all parties $P_1, \cdots, P_n$.
- Store (Complete, sid) in the memory.

**Figure 5: Threshold ECDSA functionality $\mathcal{F}_{\mathsf{ECDSA}}$.**

- $\mathcal{V}$ accepts the proof only if
  - $C_i^e D_i = y_0^{z_{m,i}} h_0^{z_{R,i}} \mod N_0$ for every $1 \le i \le l$,
  - $c^e d = \prod_{i=1}^{l} y_i^{2^k z_{m,i}} \cdot h^{2^k z_r} \mod N$,
  - $J_N(c) = J_N(d) = 1$.

Completeness and honest verifier zero-knowledge are simple extensions of $\mathsf{ZK}_{\mathsf{JL\text{-}equ}}$. Proof of knowledge could also be similarly argued. Roughly, under $k$-QR and strong JL assumptions, proof of opening of JL vector commitment gives an extraction of $m_1, \cdots, m_l$, which also helps to give an extraction of plaintext in $(C_1, \cdots, C_l)$.

# E ON GENERATING PUBLIC PARAMETERS

## E.1 Proof on the correct JL modulus: $\mathsf{ZK}_{\mathsf{JLmod}}$

In $\mathsf{ZK}_{\mathsf{JLmod}}$, the prover would like to prove the following relation:

$$\begin{aligned} R_{\mathsf{JLmod}} = \{(N, k; p, p', \bar{q}, q') \mid N = p\bar{q}, \\ p = 2^k p' + 1, \bar{q} = 2q' + 1 \\ p, \bar{q}, p', q' \text{ are primes.}\}. \end{aligned}$$

In 1999, Camenisch et al. [9] proposed proofs on that a number is the multiplication of two safe primes. Their protocol could be easily extended to get $\mathsf{ZK}_{\mathsf{JLmod}}$. The resulting protocols is rather costly. Let $t$ be the soundness parameter. The proof needs $4\lambda t^2 \log N$ bits. Prover needs to computes $2t \log N$ exponentiation and the verifier needs to computes $6t \log N$ exponentiation.

This is mainly due to proving wellness of JL modulus is not well-studied. We believe that studies could be done to significantly improve the efficiency, such as a recent work [2] proposed very efficient proof for a wide of algebraic properties the RSA modulus have. We also note that there is a compact proof [6] for arbitrary RSA prime generation. Of course it could be applied here. However, it seems to be computational expensive.

## E.2 Proof for language $\mathbb{QR}_{2^k}$: $\mathsf{ZK}_{\mathbb{QR}_{2^k}}$

Let $N$ be the JL modulus. The following protocol $\mathsf{ZK}_{\mathbb{QR}_{2^k}}$ is a perfect zero-knowledge proof for language $\mathbb{QR}_{2^k}$ with soundness error 1/2. Repeating it $t$ times would achieve a soundness error of $2^{-t}$. We could transfer it into non-interactive via Fiat-Shamir [27].

Prover $P$, holding witness $x$ s.t. $h = x^{2^k}$, runs the following to prove that $h \in \mathbb{QR}_{2^k}$.

- Prover's commitment: $P$ chooses $r \leftarrow \mathbb{Z}_N$ and computes $a = r^{2^k} \mod N$. Then, $P$ sends $a$ to $V$.
- Challenge: $V$ chooses and sends $e \leftarrow \{0, 1\}$ to $P$.
- Prover's respond: $P$ computes and sends $z = x^e r$ to $V$.
- Verification: $V$ accepts if $z^{2^k} = h^e a \mod N$.

## E.3 Proof of discrete-log over $\mathbb{QR}_{2^k}$: $\mathsf{ZK}_{\mathbb{QR}\mathbf{2kdl}}$

Let $(N, h, y, k)$ be the public key of modified JL encryption. The following protocol $\mathsf{ZK}_{\mathbb{QR}\mathsf{2kdl}}$ is a $2^s$ statistical zero-knowledge proof for relation

$$\mathcal{R}_{\mathbb{QR}\mathsf{2kdl}} = \{(N, h, y, k; \alpha \in \mathbb{Z}_N) \mid y^{2^k} = h^{2^k \alpha} \mod N\}$$

with soundness error 1/2. Repeating it $t$ times would achieve a soundness error of $2^{-t}$. We could transfer it into non-interactive via Fiat-Shamir [27].

Prove $P$, holding $\alpha$, runs as the following.

- Prover's commitment: $P$ chooses a random even number $\beta$ from $[1, 2^s N]$ and computes $a = h^{2^k \beta} \mod N$. Then, $P$ sends $a$ to $V$.
- Challenge: $V$ chooses and sends $e \leftarrow \{0, 1\}$ to $P$.
- Prover's respond: $P$ computes and sends $z = e\alpha + \beta$ (as integer) to $V$.
- Verification: $V$ accepts if $h^{2^k z} = y^{2^k e} a \mod N$.

# F THE ECDSA SIGNATURE AND THRESHOLD ECDSA

Let $\mathbb{G}$ be an elliptic curve group of prime order $q$ with base point (generator) $P$. ECDSA scheme [19] makes use of the hash function $H$, and works as follows.

(1) KGen($1^\lambda$): on input $1^\lambda$
  (a) Choose $x \leftarrow \mathbb{Z}_q$, set $x$ as the private key.
  (b) Compute $Q = x \cdot P$, and set $Q$ as the public key.
(2) Sign$(x, m)$: on input sign key $x$ and message $m$
  (a) Choose $k \leftarrow \mathbb{Z}_q$, compute $R = (r_x, r_y) = k \cdot P$.
  (b) Compute $r = r_x \mod q$ and $s = k^{-1}(H(m) + rx) \mod q$.
  (c) Output $(r, s)$ as the signature.
(3) Verify$(m; (r, s))$ calculates

$$(r_x, r_y) = R = s^{-1} H(m) \cdot P + s^{-1} r \cdot Q,$$

and outputs 1 if and only if $r = r_x \mod q$.

It is well known that for every valid signature $(r, s)$, the pair $(r, -s)$ is also a valid signature. To make $(r, s)$ unique, in this paper, we mandate that the "smaller" of $\{s, -s\}$ is the output.

Figure 5 presents the ideal functionality $\mathcal{F}_{\mathsf{ECDSA}}$ for threshold ECDSA. It consists of two functions, namely, a key generation function KGen, called once, and a signing function Sign, called an arbitrary number of times under the generated key.