Cryptography with Weights: MPC, Encryption and Signatures

Sanjam Garg[1,2], Abhishek Jain[3], Pratyay Mukherjee[4],
Rohit Sinha[5], Mingyuan Wang[1], and Yinuo Zhang[1]

[1] UC Berkeley
[2] NTT Research
[3] John Hopkins University
[4] SupraOracles
[5] Meta

Abstract. The security of several cryptosystems rests on the trust assumption that a certain fraction of the parties are honest. This trust assumption has enabled a diverse of cryptographic applications such as secure multiparty computation, threshold encryption, and threshold signatures. However, current and emerging practical use cases suggest that this paradigm of one-person-one-vote is outdated.

In this work, we consider *weighted* cryptosystems where every party is assigned a certain weight and the trust assumption is that a certain fraction of the total weight is honest. This setting can be translated to the standard setting (where each party has a unit weight) via virtualization. However, this method is quite expensive, incurring a multiplicative overhead in the weight.

We present new weighted cryptosystems with significantly better efficiency. Specifically, our proposed schemes incur only an *additive* overhead in weights.

- We first present a weighted ramp secret-sharing scheme where the size of the secret share is as short as $O(w)$ (where w corresponds to the weight). In comparison, Shamir's secret sharing with virtualization requires secret shares of size $w \cdot \lambda$, where $\lambda = \log |\mathbb{F}|$ is the security parameter.
- Next, we use our weighted secret-sharing scheme to construct weighted versions of (semi-honest) secure multiparty computation (MPC), threshold encryption, and threshold signatures. All these schemes inherit the efficiency of our secret sharing scheme and incur only an additive overhead in the weights.

Our weighted secret-sharing scheme is based on the Chinese remainder theorem. Interestingly, this secret-sharing scheme is *non-linear* and only achieves statistical privacy. These distinct features introduce several technical hurdles in applications to MPC and threshold cryptosystems. We resolve these challenges by developing several new ideas.

1 Introduction

Cryptography allows mutually distrusting parties to accomplish various tasks as long as a certain subset of the parties are honest. For example, a threshold signature (resp., encryption) [14,15] scheme allow for distributing a secret key among multiple parties such that it is possible to sign a message (resp., decrypt a ciphertext) if and only if a threshold number of parties participate honestly. More generally, a secure multiparty computation protocol (MPC) [30,20] allows a group of parties to jointly compute a public function over their private inputs such that nothing beyond the function output is revealed if a subset of the participants are honest.

This paradigm of trust has been immensely successful over the years. Threshold cryptosystems have seen widespread use in recent years, especially within the blockchain ecosystem [29]. Furthermore, efforts to standardize threshold cryptosystems have already begun [27]. MPC protocols have also started seeing increased adoption due to recent dramatic improvements in their efficiency.

Traditionally, in such systems, parties are treated equally. For instance, it is assumed that all parties are equally motivated to participate in the protocol actively; or that it is equally hard for an adversary to corrupt any party. However, the assumption that "each party is equal" does not suffice for many current and emerging applications. For instance, in stake-based blockchains [23], parties are associated with stakes that are not necessarily binary. Similarly, in oracle networks [16,9], parties have reputation scores with high variance. In such scenarios, parties in the system are naturally asymmetrical. Hence, it is natural to consider a *weighted* setting, where every party is assigned some weight, and one assumes that a certain fraction of the weights is honest.

Despite being a natural problem, essentially, the only general way in the literature to realize weighted cryptography is through a *virtualization* approach. That is, a party with assigned weight w is treated as w virtual parties, and then a standard unweighted system is used for all the virtual parties. This naïve solution, however, is extremely inefficient: a party with weight w has to bear w times the amount of computation and/or communication cost that one does in the unweighted setting. When the weights are large, this multiplicative overhead in the degradation of efficiency can be prohibitive.

In this work, we ask the following natural question:

Can we realize weighted cryptography with better efficiency?
Specifically, could the efficiency degradation depend additively *on the weights?*

Summary of this work. Our work answers this question positively. We first construct an efficient *weighted* secret-sharing scheme based on the Chinese remainder theorem. We show that if there is a sufficient gap between the privacy threshold and the reconstruction threshold (a.k.a ramp secret-sharing), our weighted ramp secret-sharing scheme (WRSS) can efficiently realize the weighted access structure, where the size of the secret share of a party with weight w is $O(w)$. In comparison, Shamir's secret sharing requires a secret share of size $w \cdot |\mathbb{F}|$, where the $\mathbb{F}$ is the field where the secret lives.

Next, we demonstrate the utility of our WRSS scheme by applying it to secure multiparty computation, threshold encryption, and threshold signature. In all applications, we show that in the weighted setting where a sufficiently large gap exists between the privacy threshold and the reconstruction threshold, the computation/communication cost of the parties only degrades additively in the weight w. Interestingly, as our WRSS scheme is both *non-linear* and *imperfect* (i.e., it only achieves statistical privacy in contrast with Shamir's which achieves perfect privacy), several new technical ideas are required for each application.

1.1 Our Contribution

Secret Sharing. Our first contribution is a construction of a weighted ramp secret-sharing with succinct share sizes. Recall that a ramp secret sharing scheme is parameterized by two thresholds: a reconstruction threshold T and a privacy threshold t. Any collection of parties with cumulative weights $\geqslant T$ should be able to reconstruct the secret; any collection of parties with cumulative weights $\leqslant t$ should not learn anything about the secret. In particular, we prove the following theorem.

Theorem 1 (Efficient WRSS). *Let $(w_1, \ldots, w_n, T, t)$ define a weighted access structure, where w_i are weights and T and t are reconstruction and privacy threshold, respectively. Assume $T - t = \Theta(\lambda)$. There exists a weighted ramp secret sharing scheme realizing $(w_1, \ldots, w_n, T, t)$ such that*

- *The share size of a party with weight w is $O(w)$.*
- *It has perfect correctness.*
- *It is $2^{-\lambda}$-statistically private.*

Our WRSS scheme builds upon the CRT-based secret sharing scheme previously studied by [26,4,21]. Interestingly, the CRT-based secret sharing (and, henceforth, our WRSS) is *non-linear.*

Weighted Secure Multiparty Computation. Next, we consider weighted MPC. In a weighted MPC, every party is assigned a weight. It is assumed that at most a certain fraction of the weights is corrupted. In this work, we consider the honest majority setting with information-theoretic security. That is, the cumulative weight of the malicious party is less than half of the total weight. Following the BGW framework [8], we construct such a weighted MPC protocol based on our WRSS scheme. In particular, our result is summarized as the following theorem.

Theorem 2 (Efficient Weighted MPC). *Let C be an arithmetic circuit over a field $\mathbb{F}$ with depth d. There exists a weighted MPC protocol for n parties with weights $w_1, \ldots, w_n$ and total weight W for computing C satisfying the following properties:*

- *The round complexity is $d + O(1)$.*

– *In the pre-processing phase, the communication cost per party per gate is $O(W)$.*
– *In the online phase, the communication cost per gate for party P_i with weight w_i is $O(w_i)$.*
– *For any semi-honest adversary who may corrupt a total weight of t, this protocol is $\exp(-\lambda)$-secure given $W - 2t = \Theta(\lambda)$.*

In comparison, the BGW protocol based on Shamir's secret sharing with virtualization would require a communication cost $W \cdot |\mathbb{F}|$ and $w_i \cdot |\mathbb{F}|$ in the preprocessing and online phase, respectively.

While MPC protocols could generically realize threshold encryption and threshold signature schemes, it will incur a large overhead if one needs to transform group operations into an arithmetic circuit over $\mathbb{F}$. Therefore, our next objectives are to construct specific efficient weighted threshold encryption and signature schemes.

Weighted Threshold Encryption. As typical in a threshold encryption scheme, we aim for a weighted threshold encryption scheme with one-round threshold decryption. In particular, we construct our weighted threshold encryption scheme based on the ElGamal cryptosystem, where the partial decryption computation cost is $O(w) + \mathsf{poly}(\lambda)$.

Shamir's secret sharing with virtualization approach would require a computation cost of $O(w)$ group operations (in contrast to bit operations).

The communication cost is only λ as partial decryption only consists of one group element. This is identical to the Shamir-based approach (See Remark 1).

Weighted Threshold Signature. Finally, we construct a threshold signature scheme based on the ECDSA signature. In particular, we construct a special weighted MPC protocol for ECDSA signing functionality summarized as follows.

Theorem 3 (Weighted Threshold Signing for ECDSA). *For any privacy threshold t, reconstruction threshold T, and total weight W such that $T - t = \Theta(\lambda)$ and $W - 2t = \Theta(\lambda)$. There is a weighted MPC protocol realizing ECDSA signing functionality such that:*

– *It has a semi-honestly secure two-round pre-signing protocol in which all the parties participate. The communication/computation cost per party is $O(W + \lambda)$.*
– *It has a non-interactive signing phase where each party i broadcasts a partial signature. The communication/computation cost per party is $O(w_i)$. As long as the cumulative weight of parties who send their partial signature is $\geqslant T$, one could correctly aggregate the signature.*

1.2 Related Work

Weighted secret sharing scheme has been previously studied by Beimel and Weinreb [7], where they showed that any weighted secret sharing scheme could be realized with share size $n^{\Theta(\log n)}$. Although the share size is independent of the weights, the share size is super-polynomial in n. In addition, Zou et al. [31] also proposed the idea of more efficiently realizing weighted secret sharing using CRT-based secret sharing. However, their work does not have a formal proof of security and only presents some experimental results on the security of the CRT-based weighted secret-sharing scheme.

We note that another way of reducing the dependence on the number of parties is through the committee-based approach. In this approach, a small number of parties are selected as committee members to perform the task on half of all parties. This approach has been considered both in the MPC setting [19,12] and the threshold signature scheme [10].

It is imaginable that this approach, combined with virtualization, could yield an efficient weighted scheme. However, we note that this approach is not generally preferable because it incurs high costs for specific parties, and is typically vulnerable to adaptive corruption attacks.

2 Technical Overview

The secret-sharing scheme is essential to any threshold cryptosystem. To build any efficient weighted threshold primitive, an efficient weighted secret-sharing scheme is usually the first objective. Hence, we start our discussion with weighted secret-sharing.

Linear Secret-sharing. We first observe that a linear weighted secret-sharing scheme is unlikely to be efficient. For a particular set of weights (for instance, if all the weights are the same), one might be able to construct a linear secret with a small overhead. However, to construct a general linear scheme that works an arbitrary set of weights, it seems inevitable that the secret share of a party with weight w contains at least $\Omega(w)$ field elements.[6] Therefore, in order to obtain a more efficient weighted secret-sharing scheme, we have to resort to non-linear schemes.

Non-linear secret-sharing. Compared to linear secret-sharing schemes, non-linear secret-sharing schemes are much less well-understood. Most of the non-linear secret-sharing schemes that have been studied are either for specialized access structures [6] or for general access structures [25,1,2]. These schemes either cannot realize the weighted threshold structure or have an exponential-size secret share. The only exception of a non-linear secret sharing scheme for threshold structure is the Chinese remainder theorem-based secret sharing scheme [26,4,21]. Indeed, as we explain later, CRT-based secret-sharing can help construct efficient weighted secret-sharing schemes.

CRT-based Secret-sharing. Let us first recall the (unweighted) CRT-based secret-sharing. Let p_0 be the order of the field $\mathbb{F}$. In CRT-based secret-sharing, parties are associated with distinct integers $p_1, \ldots, p_n$, where $p_0, p_1, \ldots, p_n$ are required to be coprime. To share a secret $s \in \mathbb{F}_{p_0}$, one picks a random integer

$$S = s + u \cdot p_0,$$

where the operations are over the integer and u is uniform over some range $\{1, 2, \ldots, L\}$. The choice of L will become clear as we proceed to discuss the correctness and security. Now, the i^{th} party shall get

$$s_i = S \mod p_i$$

as its secret share. For an authorized set A of parties, one may reconstruct the field element s by finding the unique integer S such that

$$0 \leqslant S \leqslant P_A - 1 \qquad \text{and} \qquad \forall i \in A,\ S = s_i \mod p_i,$$

where $P_A = \prod_{i \in A} p_i$. Once one finds S, s can be reconstructed by computing $s = S \mod p_0$. Therefore, for correctness, it must hold that $(p_0 + 1) \cdot L \leqslant P_A - 1$ for all authorized set A. On the other hand, for privacy, consider an unauthorized set $\overline{A}$. The adversary's view is equivalent to

$$\{S \mod p_i\}_{i \in \overline{A}} \qquad \Longleftrightarrow \qquad S \mod P_{\overline{A}}.$$

Hence, it suffices to prove that $S \mod P_{\overline{A}}$ is statistically close to the uniform distribution. This is indeed the case as long as $P_{\overline{A}}/L$ is exponentially small (see our Claim 1). To summarize, we can construct a CRT-based secret sharing as long as we can pick L such that

$$\max_{\overline{A}} P_{\overline{A}} \ll L \leqslant \min_{A} P_A/2^{\lambda}.$$

For example, for a threshold secret sharing with reconstruction threshold T. One may pick p_i as n distinct primes with length 2λ. Then, $\max_{\overline{A}} P_{\overline{A}}$ and $\min_A P_A$ are $2^{2\lambda(T-1)}$ and $2^{2\lambda \cdot T}$, respectively. Consequently, letting L to be $2^{2\lambda \cdot T - \lambda}$ satisfies the constraint above.

Note that, one could again use virtualization to realize weighted secret-sharing through (unweighted) CRT-based secret sharing. This approach will result in a secret share of length $\Theta(w \cdot \lambda)$ for a party with weight w, similar to Shamir's secret sharing.

Main Idea: Weighted Ramp Secret-sharing can be efficient. In this work, we observe that in the *ramp* setting, where there is a gap between the privacy and reconstruction threshold, one could construct an extremely efficient weighted secret sharing based on CRT secret sharing. Let w_i be the weight of the i^{th} party. One may pick the associated number p_i to be of length $c \cdot w_i$ (as opposed to aforementioned share size of $\Theta(w \cdot \lambda)$). Here, the same c is picked for all parties. Then, the constraint naturally transforms into

$$\max_{\overline{A}} 2^{\sum_{i \in \overline{A}} c \cdot w_i} \ll L \leqslant \min_{A} 2^{\sum_{i \in A} c \cdot w_i}/2^{\lambda}.$$

[6] Unless one could generically transform a set of weight $\{w_i\}$ to another set of weights $\{w'_i\}$ that are significantly smaller (i.e., $w'_i = o(w_i)$), but define the same access structure. However, this seems extremely challenging, if at all possible.

In a threshold setting, where $\max_{\overline{A}}(\sum_{i\in\overline{A}} w_i)$ can be as high as $T-1$ and $\min_A(\sum_{i\in A} w_i)$ can be as low as T, one has to pick c to be $\Theta(\lambda)$. However, if we consider a ramp secret-sharing with a privacy threshold t and reconstruction threshold T, it suffices to pick c such that $c\cdot(T-t)=\Theta(\lambda)$. In particular, in the case where $T-t=\Theta(\lambda)$, one may pick $c=1$. In other words, we observe

There is a natural trade-off between the gap *of privacy and reconstruction threshold and the* efficiency *for CRT-based secret sharing.*

This is in contrast to the linear secret sharing schemes, where it is unclear how the relaxation from threshold to ramp secret sharing could help improve efficiency.

Indeed, for the applications that we envision, it is often reasonable to assume a large gap between the privacy and reconstruction threshold. For instance, one may assume that $\leqslant 1/3$ fraction of the weights are corrupted and $\geqslant 1/2$ fraction of the weights will come online during reconstruction. In this scenario, as long as the total weight $\sum_{i=1}^n w_i$ is $\Theta(\lambda)$, the large gap is guaranteed.

2.1 Challenges in using the WRSS scheme

Our ultimate goal is to use the efficient WRSS to realize weighted cryptosystems with efficient communication/computation costs. Now, although the WRSS is well-suited for efficient weighted secret-sharing, it comes with several critical challenges. We shall discuss them next.

1. **Non-linearity.** One prominent feature of the WRSS is its non-linearity. Given secret shares $s_1,\dots,s_n$, one needs to reconstruct the secret through a *non-linear function* as
$$\Big((\lambda_1\cdot s_1+\lambda_2\cdot s_2+\cdots+\lambda_n\cdot s_n) \mod P\Big) \mod p_0,$$
where $P=p_1p_2\cdots p_n$. Similar to Lagrange coefficients, here, λ_i is the integer satisfying[7]
$$\lambda_i \mod p_i = 1 \qquad \text{and} \qquad \forall j\neq i,\ \lambda_i \mod p_j = 0.$$
Now, imagine we want to reconstruct g^s for some generator g from the group $\mathbb{G}$ of order p_0. In Shamir's secret sharing, parties may simply broadcast g^{s_i}, and later one can use Lagrange interpolation to find g^s. In WRSS scheme, however, interpolation using group elements g^{s_i} will only give $g^{\lambda_1\cdot s_1+\cdots+\lambda_n\cdot s_n}$, whose exponent is effectively equal to
$$\Big(\lambda_1\cdot s_1+\lambda_2\cdot s_2+\cdots+\lambda_n\cdot s_n\Big) \mod p_0,$$
which is not necessarily equal to s. Evidently, the non-linearity poses a challenge to correctness.
2. **Integer Growing Problem.** Although the reconstruction procedure of the WRSS is non-linear, it does support local computations similar to Shamir's secret sharing. For instance, suppose x and y are secret shared. Intuitively, parties can locally compute $x_i+y_i \mod p_i$ as a secret share of the secret $x+y$. This, however, is not always correct. The issue is that the associated integer might grow out of range. Recall that x is re-randomized as some integer $X=x+u\cdot p_0$ and y as $Y=y+u'\cdot p_0$. For any authorized set A and the product P_A, the correctness guarantees that both X and Y is $<P_A$. Nonetheless, it is not guaranteed that $X+Y<P_A$. Therefore, when parties use secret shares $x_i+y_i \mod p_i$ to reconstruct $x+y$, they are trying to reconstruct the secret integer $X+Y$ first. And they can only correctly reconstruct $X+Y$ when $X+Y<P_A$.
Similar issues arise when one wants to locally compute the secret shares of $-x$, $x\cdot y$, and scalar multiplication $c\cdot x$ for some constant c. Therefore, one must be careful with correctness when trying to do local computations.
3. **Challenges for Simulation.** Consider a secret-sharing-based MPC protocol. At the end of the protocol, parties typically broadcast the secret share s_i of the output wire to allow reconstruction of the output s. A simulator, given the output s, needs to simulate all the secret shares of the honest parties. This is usually not an issue for linear secret sharing schemes as, at each wire s, it is maintained that s_i's are identically distributed as freshly sampled secret sharing of s (and, hence, simulatable). However, consider a WRSS secret sharing of x and y. Observe that the secret shares of $x_i+y_i \mod p_i$ is *not* identically distributed as a fresh secret sharing of $x+y$.[8] Therefore, given

[7] We note that λ_i could be efficiently computed. Refer to Remark 2.

[8] In fact, their statistical distance is quite far. In particular, the distribution of the integer $X+Y$, where $X=x+u\cdot p_0$ and $Y=y+u'\cdot p_0$ is very different from the integer $(x+y)+u''\cdot p_0$.

the output $x + y$, it is not clear how to simulate the broadcast secret shares. One may hope to resolve this issue by masking the secret shares with a fresh secret sharing of 0. However, note that we are essentially trying to mask an integer $X + Y$ over integer operations (instead of over a field). Consequently, extra care is required for this to go through.

Next, we discuss how we address these issues in different settings.

2.2 Weighted Threshold Encryption

For expository purposes, we start with a threshold encryption scheme. Recall that we aim for a scheme with one-round threshold decryption. Typically, this is done by combining a PKE scheme with a secret sharing scheme, where the secret key is shared among parties. In this work, we consider the ElGamal encryption scheme for the underlying PKE scheme. Let us recall it first. In the ElGamal encryption scheme, a group $\mathbb{G}$ with order p_0 and generator g is sampled. The secret key sk and public key pk are sampled as s and g^s where $s \leftarrow \mathbb{F}_p$. To encrypt a message msg, one sample a random $r \leftarrow \mathbb{F}_p$, and the ciphertext is defined as $(\mathsf{msg} \cdot \mathsf{pk}^r, g^r)$. Given a ciphertext (c_1, c_2), one could decrypt it as $c_1 \cdot c_2^{-\mathsf{sk}}$. This encryption scheme is semantically secure as long as DDH is hard.

Now, suppose we sample a public key and secret key (g^s, s) from ElGamal and secret share s using our WRSS scheme. Given a ciphertext $(\mathsf{msg} \cdot g^{r \cdot s}, g^r)$, what should parties send as a partial decryption? As we discussed earlier, if parties simply send $g^{r \cdot s_i}$, one cannot correctly aggregate it to obtain $g^{r \cdot s}$.

Towards resolving this challenge, we first observe that the reconstruction of CRT-based secret sharing can be rewritten as

$$\Big(\Big((\lambda_1 \cdot s_1 \mod P) + (\lambda_2 \cdot s_2 \mod P) + \cdots + (\lambda_n \cdot s_n \mod P)\Big) \mod P\Big) \mod p_0.$$

For simplicity, let us write $\lambda_i \cdot s_i \mod P$ as α_i. There are several benefits to writing the reconstruction as above. First, parties can locally compute α_i. Second, given $\alpha_1, \alpha_2, \ldots, \alpha_n$, we know that the secret s is of the form

$$s = (\alpha_1 + \alpha_2 + \cdots + \alpha_n - \varDelta \cdot P) \mod p_0, \qquad \text{where } \varDelta \in \{0, 1, \ldots, n-1\}.$$

Crucially, the overflow number $\varDelta$ has only polynomial many possibilities. Therefore, given the partial decryption $g^{r \cdot \alpha_i}$, one knows that the one-time pad g^{rs} is one of the following

$$g^{r \cdot \sum_i \alpha_i}, g^{r \cdot \sum_i \alpha_i - r \cdot P}, \ldots, g^{r \cdot \sum_i \alpha_i - r \cdot (n-1)P}.$$

To get statistical correctness, we shall ask the encryptor to include a hash of the encapsulated key $H(g^{r \cdot s})$ (using, for example, a universal hash function). Consequently, the decryptor could check all possibilities of the encapsulated key against the hash $H(g^{r \cdot s})$ to find $g^{r \cdot s}$. Finally, since $H(g^{r \cdot s})$ leaks information about $g^{r \cdot s}$, we shall add a randomness extractor Ext to extract uniform randomness from $g^{r \cdot s}$. Overall, the ciphertext would be

$$\mathsf{msg} \cdot \mathsf{Ext}(\mathsf{seed}, g^{r \cdot s}),\ \mathsf{seed},\ g^r,\ H(g^{r \cdot s}).$$

This presents the main ideas behind our efficient weight threshold decryption scheme. To prove the security, we need the additional guarantee that the weights cannot be too small (for example, a constant). Indeed, if the weight w_i is too small, one could use an exhaustive search to find party P_i's secret share using its partial decryption output. We refer the readers to Section 6 for more details.

Remark 1 (Raise hand setting.). We note that our scheme is in the "raise hand" setting. That is, parties need to know what authorized set will participate in the partial decryption process. This is because the Lagrange coefficient λ_i depends on this information. In contrast, Shamir's secret sharing-based scheme does not need this information for partial decryption. Indeed, parties could directly send g^{s_i} and the aggregator could do Lagrange interpolation over the group elements.

However, we note that, even for Shamir's secret sharing, "raise hand" might be preferable in the weighted setting as the communication cost is much lower compared to the non-raise-hand setting. Indeed, a party with weight w would need to broadcast w many group elements in the non-raise-hand setting; while in the "raise hand" setting, parties aggregate the partial decryption locally first and only need to broadcast one group element.

2.3 Weighted MPC

Next, we consider weighted MPC. In a weighted MPC protocol, every party is assigned some weights. And it is assumed that the cumulative weight of the corrupted parties is upper-bounded by a certain fraction. In this work, we restrict to the information-theoretic honest majority setting and semi-honest adversaries. Crucially, the communication/computation cost (per party i and gate) should be $O(w_i) + \mathsf{poly}(\lambda)$.

On a high level, our protocol adopts the secret-sharing-based MPC framework (e.g., BGW protocol [8]), where we shall use the WRSS scheme as the underlying secret sharing scheme. Consequently, the efficiency of the WRSS scheme will determine the efficiency of the weighted MPC protocol. As we have mentioned, this approach involves several issues. We discuss how to address these issues next.

Multiplication. We consider the multiplication gate first. Let $W = w_1 + \cdots + w_n$ be the total weight and assume that the adversary may corrupt parties with weight at most t. The security of the WRSS requires that: if the value x of a wire is secret shared, it must be the case that the random integer $X = x + u \cdot p_0$ is sampled from $u \leftarrow \{1, \ldots, L\}$ with $L \gg 2^t$ (e.g., $L = 2^{t+\lambda}$). Therefore, the integer X associated with every wire x is (approximately) of size $2^{t+2\lambda}$. Now, suppose we want to compute the product $x \cdot y$. The corresponding integer $X \cdot Y$ may be as large as $2^{2t+4\lambda}$. This integer XY (henceforth, the secret xy) could only be reconstructed if the total weight W satisfies $2^W \geqslant 2^{2t+4\lambda}$. Therefore, our protocol only works in the setting where there is an honest majority and a large enough gap (i.e., $\Theta(\lambda)$) between the corruption threshold t and half of the total weight $W/2$.

Although the secret could be reconstructed after one multiplication gate, one cannot let the integer grow indefinitely. Therefore, after every multiplication gate, one has to use a "degree reduction" protocol[9] to reduce the integer Z associated with $z = xy$ to a smaller range. Our degree reduction protocol follows the standard approach in the MPC literature. In particular, in the preprocessing phase, we ask parties to generate two secret shares $[r]^0$ and $[r]^1$ of a random value r. Here, in the share $[r]^0$, r is re-randomized as some integer over the small range $L = 2^{t+\lambda}$; while in the share $[r]^1$, r is re-randomized as some integer over the large range $L = 2^{2t+4\lambda}$. The idea is that parties will use the secret shares of $[r]^1$ to reconstruct $r + xy$ in the clear. Afterward, they may locally subtract $r + xy$ from $[r]^0$ to obtain a secret share of xy with a small integer range.

However, there is one crucial issue here. One has to guarantee that the reconstruction of $r + xy$ leaks only the value $r + xy$ and nothing else. While this comes for free in Shamir's secret sharing, it is not the case here. Indeed, the secret shares of $r + xy$ reveal its associated integer, whose distribution may not be close to a fresh secret sharing of the secret $r + xy$. We defer the discussion of this issue to the discussion on the output reconstruction procedure.

Addition. Similarly to the multiplication gate, the addition gate also has integer growing issues. One might think if we can handle the multiplication gate, we can certainly handle the addition gate in the same way. While this is true, we do not want to invoke a degree reduction protocol for addition gates, which incurs additional interactions and consumes correlations.

Instead, we observe that the growth of the integer for addition gates is very slow. In particular, if a circuit has size $\mathsf{poly}(\lambda)$, the integer associated with a wire, which is a sum of several other wires, is upper-bounded by $\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda} \lll 2^W$. Hence, reconstructing the sum of wires is not an issue. However, it becomes problematic when we want to reconstruct $x \cdot y$, where x and y are the sums of several wires. Indeed, both X and Y are now upper-bounded by $\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda}$ and $X \cdot Y$ might be $\geqslant 2^W$ if $W \approx 2t + 4\lambda$. However, this is not an issue as long as W is large enough (e.g., $W \geqslant 2t + 5\lambda$). In other words, if the total weight is large enough, the integer growing for the addition gates is not an issue.

Output Reconstruction. As we have mentioned, unlike Shamir's secret sharing, it is not clear if parties could broadcast the secret shares of the output wire for reconstruction. To resolve this issue, we shall use a freshly sampled secret share $[0]$ to mask the secret shares $[\mathsf{out}]$ of the output wire. Parties will reconstruct $\mathsf{out} + 0$ as the output of the protocol. Again, here, we need to argue that the secret shares of $\mathsf{out} + 0$ leak only $\mathsf{out} + 0$. In particular, the integer associated with the secret $\mathsf{out} + 0$ should only depend on $\mathsf{out} + 0$. We observe that if the integer associated with out is (arbitrarily) distributed over some range

[9] We call this a degree reduction protocol as it is reminiscent of the degree reduction protocol in the BGW protocol based on Shamir's secret sharing. In Shamir's secret sharing, the product of two secrets shared by a degree-t polynomial is shared by a degree-$2t$ polynomial. A degree reduction protocol in this case brings down the degree of the polynomial back to t.

$\{1,\ldots,L\}$, it suffices to sample the integer associated with 0 uniformly randomly from an exponentially large range $\{1,\ldots,L\cdot 2^{\lambda}\}$. The sum of these two integers will be exponentially close to a freshly sampled secret share of $0+\mathsf{out}$ from the range $\{1,\ldots,L\cdot 2^{\lambda}\}$. We formally prove this by our integer masking lemma (Lemma 1).

2.4 Weighted Threshold Signature

Lastly, we consider the threshold signature protocol. In particular, we consider a weighted multiparty signing protocol based on the ECDSA signatures.

Let us first recall the signing functionality of the ECDSA signature. Let sk be the signing key, G be the curve base point, H be a cryptographic hash function and m be the message. To sign message m with sk, one computes the following:

1. **(Pre-signing Phase)** Generate a secret random value $k \leftarrow \mathbb{F}_q$, and then compute (public) group element $k \times G$.
2. **(Signing Phase)** Parse $k \times G$ as curve point (r_x, r_y). Then compute $\sigma = k^{-1}\cdot(H(m)+r_x\cdot\mathsf{sk})$.
3. Output the signature (r_x,σ).

Note that we could generically use our MPC protocol to compute all the field operations. However, parties do need to construct the group element $k \times G$ in the clear. We further note that parties need to agree on the exact value of $k \times G$ in order to proceed in the signing phase (i.e., step 2). Hence, our ideas from the threshold encryption section, where parties agree that $k \times G$ is one of n possibilities, are not applicable.

However, note that our task at hand is significantly simpler compared to the threshold encryption setting. In the pre-signing phase, we simply need all parties to collectively sample a random group element $k \times G$ while also obtaining a secret sharing of k. This is different from the threshold encryption setting, where parties start with a secret share of k. And later in the online phase, they need to reconstruct $(g')^k$ for some random group element g'.

To collectively sample $k \times G$ and the secret shares $[k]$, we simply ask party P_i sample a random k_i and (i) secret share $[k_i]$ among all the parties; (ii) broadcast the group element $k_i \times G$. Afterward, parties could locally reconstruct $k \times G$ as $\sum_i (k_i \times G)$. Party P_i locally computes the secret share of k by computing $\sum_j [k_j]_i$. This is secure simply because $k_i \times G$ forms an additive secret share of $k \times G$ and could be simulated given only $k \times G$.

Finally, by standard techniques in ECDSA, one could prepare the secret shares of the correlated values $[k^{-1}]$ and $[r_x \cdot \mathsf{sk}]$ in the pre-signing protocol, which leads to a one-round signing phase. We refer the readers to Section 7 for details.

2.5 Open Problems

Our work initializes the use of CRT-based secret-sharing techniques in the weighted threshold cryptosystem. It leaves open several exciting problems. We discuss some of them below.

Maliciously Secure MPC. For the weighted MPC protocol, a natural question is achieving malicious security. In the literature, several approaches have been developed for achieving information-theoretic malicious security such as verifiable secret-sharing schemes [11]. Could we construct an efficient WRSS with verifiability while still being efficient?

Scalable MPC. A large body of the MPC literature works on removing the dependency of efficiency on the number of parties. For instance, to generate one instance of secret shares of a random value, our MPC protocol requires $O(n^2)$ amount of total communication. In the Shamir secret sharing-based MPC, the Vandemonde matrix randomness extraction [13] is a standard technique for reducing this dependency to linear. Is there a similar technique for CRT-based secret sharing? Note that this problem is challenging as we are essentially trying to extract randomness over the integers (instead of over a field).

Another standard efficiency-saving technique is the packed secret sharing scheme [17]. Could one pack secrets similarly in the CRT-based secret sharing? We note that CRT-based secret sharing naturally supports packing two secrets $s \in \mathbb{F}_{p_0}$ and $s \in \mathbb{F}_{p'_0}$ from fields with *different* characteristics (as they could be treated as one secret modulo $p_0 \cdot p_0'$). It is, however, surprisingly challenging to pack two secrets from the same field for CRT secret sharing.

CCA Threshold Encryption. Currently, our weighted threshold encryption scheme only achieves CPA security. It is a natural problem whether one could construct CCA-secure efficient weighted threshold encryption using similar techniques.

3 Preliminaries

We use λ for the security parameter. Let $\mathsf{negl}(\lambda)$ denote a negligible function. That is, for all polynomial $p(\lambda)$, it holds that $\mathsf{negl}(\lambda) < 1/p(\lambda)$ for large enough λ. For any two distributions A, B over the finite universe Ω, the statistical distance between A and B is defined as $\mathsf{SD}(A, B) = \frac{1}{2}\sum_{\omega\in\Omega}|\Pr[A=\omega]-\Pr[B=\omega]|$. For an integer n, we shall use $[n]$ for the set $\{1, 2, \ldots, n\}$. For an integer M, we also use U_M for the uniform distribution over $\{0, 1, \ldots, M-1\}$.

The security of the CRT-based secret sharing relies heavily on the following claim.

Claim 1 ([21]) *Let $M < L$ be arbitrary integers. Let p be an arbitrary integer that is coprime with M. Let s be any integer. We have*

$$\mathsf{SD}\left(\ (s + p \cdot U_L) \mod M \quad , \quad U_M\ \right) < M/L.$$

Intuitively, this claim states the following. Suppose we have a secret $s \in \mathbb{F}$, where the order of $\mathbb{F}$ is p. If we pick a sufficiently random[10] integer $S = s + p \cdot U_L$, it is guaranteed that $S \mod M$ is statistically close to uniformly random. This claim is crucial in proving the security of the CRT-based secret-sharing scheme. We include a formal proof in Section A.

4 Efficient Weighted Ramp Secret-sharing Scheme

In this section, we show how to construct an efficient weighted ramp secret-sharing (WRSS) scheme. Our scheme is based on the Chinese Remainder Theorem-based (CRT-based) secret-sharing scheme. This scheme is introduced by [26,4,21] in the unweighted setting. Let us recall their scheme and formally present its security. Next, we show how to transform this scheme to the weighted setting, where the size of the secret share is small.

4.1 Unweighted CRT-based Secret-sharing

Let $\mathbb{F}_{p_0}$ be a field, where $p_0 \approx 2^\lambda$. Suppose we want to secret share a secret $s \in \mathbb{F}_{p_0}$. Unlike Shamir's secret-sharing scheme, the CRT-based scheme is non-linear. In particular, the secret shares are not elements from $\mathbb{F}_{p_0}$. Instead, for all $i \in [n]$, the i^{th} party is associated with an integer p_i and its secret share shall be an integer s_i such that $0 \leqslant s_i < p_i - 1$. Formally, the CRT-based secret-sharing scheme among n parties is constructed as follows.

- **Access Structure.** Let $\mathcal{A}$ be the set of authorized subsets and $\overline{\mathcal{A}}$ be the set of unauthorized subsets.
- **Parameters.** The scheme is parametrized by a set of integers $p_1, p_2, \ldots, p_n$ and an additional integer L. It is required that all the p_i's (including p_0) are *coprime* with each other. These parameters implicitly define the following two products. (Note that $P_{\mathsf{max}} < P_{\mathsf{min}}$.)

$$P_{\mathsf{max}} = \max_{\overline{A}\in\overline{\mathcal{A}}}\left(\prod_{i\in\overline{A}} p_i\right) \qquad \text{and} \qquad P_{\mathsf{min}} = \min_{A\in\mathcal{A}}\left(\prod_{i\in A} p_i\right).$$

- **Share the secret.** To share a secret s, one picks a *random integer*

$$S = s + p_0 \cdot U_L.$$

Recall that U_L is uniformly distributed over $[L]$. We will refer to the integer S as the lifting of s and write the above step as $S = \mathsf{Lift}(s, U_L)$. When it is clear from the text, we also write $S = \mathsf{Lift}(s)$. The

[10] Measured by the parameter L.

secret share of the i^{th} party shall be

$$s_i = S \mod p_i.$$

- **Reconstruct the secret.** For an authorized set $A \in \mathcal{A}$, parties in A reconstruct the secret as follows. Using Chinese remainder theorem, they can find a set of Lagrange coefficients $\{\lambda_i\}_{i\in A}$ such that $S = \sum_{i\in A} \lambda_i \cdot s_i \mod P$. Then they can reconstruct the secret s as

$$s = S \mod p_0.$$

Fig. 1: A generic CRT-based Secret-sharing Scheme

Remark 2. The Lagrange coefficient λ_i here are integers such that

$$\lambda_i \mod p_i = 1 \qquad \text{and} \qquad \forall j \neq i,\ \lambda_i \mod p_j = 0.$$

We note that the Lagrange coefficients λ_i could be efficiently computed as follows. Let $Q = \prod_{j\neq i} P_j$ be the product of p_j's except for p_i. Then,

$$\lambda_i = Q \cdot Q^{-1},$$

where Q^{-1} is the inverse of Q modulo p_i. That is, $Q \cdot Q^{-1} \mod p_i = 1$.

Theorem 4. *The secret-sharing scheme in Figure 1 satisfies the following.*

- ***Correctness.*** *The scheme is perfectly correct if* $(L+1)\cdot p_0 < P_{\mathsf{min}}$.
- ***Security.*** *The insecurity of scheme is* $\leqslant P_{\mathsf{max}}/L$. *That is, for any unauthorized set, the statistical distance between the distributions of its secret shares for any two distinct secrets is at most* P_{max}/L.

Proof. Suppose $(L+1)\cdot p_0 < P_{\mathsf{min}}$. For any authorized set A and secret s, observe the following. The random integer $S = s + p_0 \cdot U_L$ always satisfies

$$S \leqslant (L+1)\cdot p_0 < P_{\mathsf{min}} \leqslant \prod_{i\in A} p_i.$$

Consequently, given the secret shares s_i for $i \in A$, parties can always correctly recover the integer S and, consequently, correctly reconstruct the secret $s = S \mod p_0$.
Next, we argue the security. For any unauthorized set $\overline{A}$ and any secret s, observe the following. Let $P = \prod_{i\in\overline{A}} p_i$. By the Chinese remainder theorem, there is a bijection between the secret shares $\{s_i\}_{i\in\overline{A}}$ and the integer in $\{0, 1, \ldots, P-1\}$. Therefore, instead of considering the distribution of the secret shares, i.e.,

$$\{s + p_0 \cdot U_L \mod p_i\}_{i\in\overline{A}},$$

we shall equivalently consider the distribution of the following integer

$$s + p_0 \cdot U_L \mod P.$$

By Claim 1, for any secret s, this distribution is (P/L)-close to the uniform distribution over U_P. Therefore, for any unauthorized set $\overline{A}$, the insecurity is $\leqslant (\prod_{i\in\overline{A}} p_i)/L$ and, by definition, the insecurity of the whole scheme is $\leqslant P_{\mathsf{max}}/L$.

Threshold secret-sharing. As a representative example, we illustrate how one can implement a t-threshold secret-sharing using the CRT-based scheme. The parameters can be set up as follows. Pick $p_1, \ldots, p_n$ as n distinct prime numbers of length 2λ. By definition, P_{max} is the maximum product of $t-1$ integers, which is approximately $P_{\mathsf{max}} \approx 2^{(2\lambda)\cdot(t-1)}$; P_{min} is the minimum product of t integers, which is approximately $P_{\mathsf{min}} \approx 2^{(2\lambda)\cdot t}$. Then, if one picks L to be $L \approx 2^{2t\lambda-\lambda}$, one can verify by Theorem 4 that the scheme is a threshold secret-sharing with perfect correctness and $2^{-\lambda}$-insecurity.

4.2 Realizing Efficient WRSS using CRT-based Secret-sharing

Weighted Secret-sharing. In a weighted secret-sharing among n parties. Every party i is associated with a weight $w_i \in \mathbb{N}$. We consider *the ramp secret-sharing* setting. That is, there is a reconstruction threshold T and also a privacy threshold t. A set of parties is authorized if their collective weight is $\geqslant T$; a set of parties is unauthorized if their collective weight is $\leqslant t$. In a ramp scheme, a set of parties with collective weight $\in (t, T)$ may learn partial information about the secret.

– Reconstruction threshold T. A set $A \in \mathcal{A}$ is authorized if $\sum_{i\in A} w_i \geqslant T$.
– Privacy threshold t. A set $\overline{A} \in \overline{\mathcal{A}}$ is unauthorized if $\sum_{i\in B} w_i \leqslant t$.

Fig. 2: The access structure of the weighted ramp secret-sharing scheme.

Naïve Construction with Large Share Size: Shamir's Secret-sharing with Virtual Parties. It is not hard to see that one can construct the (threshold) weighted secret-sharing scheme using Shamir's secret-sharing scheme. In particular, one thinks of the i^{th} party with weight w_i as w_i *virtual parties*. That is, one can use the standard Shamir's secret-sharing scheme with $w_1 + w_2 + \cdots + w_n$ parties. Afterward, the i^{th} party shall get w_i secret shares as its secret share. In words, the i^{th} party represents w_i virtual parties in this secret-sharing scheme with $w_1 + \cdots + w_n$ virtual parties.
However, the size of the secret share in this naïve construction is quite large. In particular, party with weight w_i shall get w_i field elements $\in \mathbb{F}_{p_0}$ as its secret share. Therefore, the total length of the secret share is $w_i \cdot \lambda$.

CRT-based Construction with Small Share Size. To realize the access structure of the weighted secret-sharing scheme, we shall pick each p_i to be an integer of w_i length.[11] In particular, we shall pick p_i in the range $2^{w_i}/(1+1/n) \leqslant p_i < 2^{w_i}$.[12] By definition,

$$P_{\mathsf{max}} = \max_{\overline{A}\in\overline{\mathcal{A}}} \left(\prod_{i\in\overline{A}} p_i \right) < \max_{\overline{A}\in\overline{\mathcal{A}}} \left(\prod_{i\in\overline{A}} 2^{w_i} \right) \leqslant 2^t.$$

On the other hand,

$$P_{\mathsf{min}} = \min_{A\in\mathcal{A}} \left(\prod_{i\in A} p_i \right) \geqslant \max_{A\in\mathcal{A}} \left(\prod_{i\in A} 2^{w_i}/(1+1/n) \right) \geqslant 2^T/(1+1/n)^n = 2^{T-O(1)}.$$

Therefore, if one picks the parameter L to be $2^{t+\lambda}$. One may verify by Theorem 4 that this secret-sharing scheme is $O(2^{-\lambda})$-insecure and is perfectly correct as long as $T \geqslant t + 2\lambda + \Theta(1)$. Furthermore, observe that the secret shares of the i^{th} party is simply an integer between 0 and p_i. Therefore, the total length of the i^{th} secret share is w_i. In conclusion, this construction gives rise to the following theorem.

Theorem 5. *Assume $T \geqslant t + 2\lambda + \Theta(1)$, the CRT-based secret-sharing scheme described above realizes the access structure in Figure 2 with perfect correctness and $2^{-\lambda}$ insecurity. Furthermore, the length of the secret share with weight w_i is w_i.*

Observe that, if the gap $T - t$ could always be amplified at the cost of efficiency. In particular, for any integer c, the access structure of parties with weights $c \cdot w_1, c \cdot w_2, \ldots, c \cdot w_n$ and reconstruction (resp. privacy) threshold $c \cdot T$ (resp. $c \cdot t$) is identical to the original access structure. Hence, this gives us the following corollary.

Corollary 1 (Efficient WRSS). *For any integer c such that $c \cdot (T - t) \geqslant 2\lambda + \Theta(1)$, the weighted ramp secret-sharing scheme described above realizes the access structure in Figure 2 with perfect correctness and $2^{-\lambda}$ insecurity. Furthermore, the length of the secret share with weight w_i is $c \cdot w_i$.*

In particular, as long as $T - t = \Omega(\lambda)$, we can construct a weighted ramp secret sharing scheme with share size $O(w_i)$.

[11] To ensure they are coprime, we may pick p_i to be a distinct prime of length w_i.

[12] There are $2^{w_i}/(n+1)$ many integers between $2^{w_i}/(1+1/n)$ and 2^{w_i}, among which, there are asymptotically $2^{w_i}/((n+1) \cdot w_i)$ many primes numbers. Therefore, as long as w_i is large enough, e.g., $\mathsf{polylog}(\lambda)$, one could always pick a p_i for all parties. Even if the smallest w_i is a small constant, one could always multiply every weight by some small factor to enable this.

5 Efficient Weighted MPC

In this section, we shall present a weighted MPC protocol against semi-honest adversaries. Moreover, we consider an honest majority in the weighted setting[13] and information-theoretic security. Let us first define security. We follow the definition in [3] with appropriate adaptation to the weighted setting.

Definition 1 (Semi-honestly Security). *Let $\mathbf{W} = (w_1, \ldots, w_n)$ be the weights of a total of n parties. Let $C : \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$ be an arithmetic circuit over $\mathbb{F}_{p_0}$. We say that a protocol π ε-securely realized C with corruption threshold t in the weighted setting if the following holds. For any input $\vec{x}$ and any subset $I \subset [n]$ such that $\sum_{i \in I} w_i \leqslant t$, there exists an efficient simulator $\mathcal{S}$ such that*

$$\mathsf{SD}\left(\left(\mathcal{S}(I, \vec{x}_I, C(\vec{x})_I), C(\vec{x})\right)\ ,\ \left(\mathsf{View}_I^{\pi}(\vec{x}), \mathsf{Output}^{\pi}(\vec{x})\right)\right) \leqslant \varepsilon.$$

Notations. We use the following notations for the WRSS in our weighted MPC protocol. Let $\mathbf{W} = (w_1, \ldots, w_n)$ be the weights of a total of n parties, $\mathbf{P} = (p_0, p_1, \ldots, p_n)$ be the corresponding bases and let (T, t) be the reconstruction and privacy threshold. In the MPC case, the reconstruction threshold $T = W$ is the total weight of all parties. We denote the weighted ramp secret sharing of some secret s by $\{[s]_i\}_{i \in [n]} \leftarrow \mathsf{Share}(\mathbf{P}, T, t, s)$, where $[s]_i$ is party P_i's share of the secret s. Furthermore, we express the associated lifting of s as $S = \mathsf{Lift}(s)$ where the randomness U_L is implicit. Correspondingly we let $S = \mathsf{Reconstruct}(\{[s]_i\}_{i \in [n]})$ be the reconstructed integer $\mathsf{Lift}(s)$ value. For every secret s, we have $s = S \bmod p_0$.

Overview of the protocol. For every input wire s, we secret share the value s using our WRSS where the parameter L is $2^{t+\lambda}$. Therefore, $\mathsf{Lift}(s)$ is of size at most $(2^{t+\lambda}+1) \cdot p_0 \leqslant 2^{t+2\lambda}$. Throughout the MPC protocol, we shall maintain the invariant that the for every wire s, the secret integer $S = \mathsf{Lift}(s)$ associated with the secret share of s is upper-bounded by some $\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda}$. Intuitively, this invariant is maintained for each addition gate. However, after each multiplication gate (including scalar multiplication where the scalar is superpolynomial in λ), this invariant is broken. Hence, we shall employ a degree reduction protocol to re-establish this invariant. For degree reduction, in the preprocessing phase, every party shall generate two secret shares of a random value r, denoted by $[r]^0$ and $[r]^1$. The instance $[r]^0$ is sampled where the corresponding parameter L is $2^{t+\lambda}$; while the instance $[r]^1$ is sampled where the corresponding parameter L is $2^{2t+5\lambda}$. Parties shall use $[r]^1$ as a mask to reconstruct the value $r+s$ in the clear and then deduct it from the secret share $[r]^0$ locally. To successfully reconstruct the value $r+s$, which corresponds to an integer of size at most $2^{2t+5\lambda} \cdot p_0$, we need the total weights to satisfy $W \geqslant 2t + 6\lambda + \Theta(1)$. Therefore, as long as $W - 2t = \Theta(\lambda)$, we have the following theorem.

Theorem 6. *Let C be an arithmetic circuit over $\mathbb{F}$ with depth d. There is a weighted MPC protocol realizing C with the following property.*

- *The round complexity is $d + O(1)$.*
- *In the online phase, the communication/computation cost per party per gate is $O(w_i)$.*
- *In the preprocessing phase, the communication/computation cost per party per gate is $O(W)$.*
- *For any semi-honest adversary who may corrupt a total weight of t, this protocol is $\exp(-\lambda)$-secure given $W - 2t = \Theta(\lambda)$.*

We next describe our protocol in detail.

5.1 Generating shares of random value F_{Random}

In this sub-protocol, parties generate a secret sharing of a random value. Observe that the communication cost per party is $O(W)$ as it sends $O(w_i)$ bits to the i^{th} party.

- For all $i \in [n]$, the i^{th} party samples a random value $r_i \in \mathbb{F}$. It secret shares r_i : $\{[r_i]_j\}_{j \in [n]} \leftarrow \mathsf{Share}(\mathbf{P}, W, t, r_i)$ and sends the shares to each party.
- For all $i \in [n]$, the i^{th} party locally computes $[r]_i = \left([r_1]_i + [r_2]_i + \cdots + [r_n]_i\right) \bmod p_i$ as its secret share of the random field element $r = r_1 + \cdots + r_n \in \mathbb{F}$.

[13] I.e., the cumulative weight of the corrupted party is less than half of the total weight.

We note that the threshold parameter L in generating the WRSS secret shares is either $2^{t+\lambda}$ or $2^{2t+5\lambda}$. Furthermore, we also use this protocol for generating secret shares of the value 0 among all the parties. The only difference is that parties sample a fresh secret share of 0 instead of a random r_i. The threshold L is generating the secret sharing of 0 is $2^{t+3\lambda}$.
We will sometimes refer to the above protocol as $F_{\mathsf{Random}}(r = \sum_{i\in[n]} r_i)$ to specify the individual randomness r_i from each party.

5.2 Degree reduction protocol F_{deg}

In this sub-protocol, parties re-sample the secret share of some wire x such that the corresponding integer $\mathsf{Lift}(x)$ is small enough. Observe that the communication cost per party is $O(w_i)$.

- **Input.** Parties hold the secret shares $[x]$ of some wire x. Additionally, parties hold two secret shares (i.e., $\{[r]_i^0\}_{i\in[n]}$ and $\{[r]_i^1\}_{i\in[n]}$) of a random r. Both $[r]^0$ and $[r]^1$ are sampled using the F_{Random} protocol, where the threshold parameters are $2^{t+\lambda}$ and $2^{2t+5\lambda}$, respectively.
- Party P_i locally computes and broadcasts $\left([x]_i + [r]_i^1\right) \mod p_i$ as the secret shares of $x + r$.
- Given all the secret shares, parties locally reconstruct $x + r \in \mathbb{F}$ and subtract $(x + r) \mod p_i$ from the secret shares $\{[r]_i^0\}_{i\in[n]}$.

5.3 Opening secret shares F_{open}

In this sub-protocol, parties open the value of the output wire. Observe that the communication cost per party is $O(w_i)$

- **Input.** Parties hold a secret share $[\mathsf{out}]$ of the output wire. Parties also hold a secret sharing of $[0]$ generated similarly as in the F_{Random} sub-protocol.
- Party P_i locally computes and broadcasts $\left([0]_i + [\mathsf{out}]_i\right) \mod p_i$ as the secret shares of $0 + \mathsf{out}$.
- Parties locally reconstruct $0 + \mathsf{out}$ as the value of out.

5.4 Realizing negation gate F_{neg}

In this (non-interactive) sub-protocol, parties switch the secret shares $[x]$ of x to the secret shares of $[-x]$. Negation gate usually comes for free in the Shamir secret share-based MPC. However, in our scheme, it requires some special care.
Observe that if parties simply invert their secret share from $[x]_i$ to $p_i - [x]_i$. The lifted integer goes from $\mathsf{Lift}(x)$ to $P - \mathsf{Lift}(x)$, where P is the product of p_i. Crucially, note that

$$\mathsf{Lift}(x) = x \mod p_0 \quad \not\Longrightarrow \quad P - \mathsf{Lift}(x) = -x \mod p_0$$

as P is not a multiple of p_0. Therefore, this approach has a correctness issue. We realize negation by the following protocol.

- **Input.** Parties hold WRSS of some secret x.
- Parties (locally) identify a bound $B \cdot p_0$ on the integer $\mathsf{Lift}(x)$. For example, if x is an input wire, $\mathsf{Lift}(x)$ is at most $(2^{t+\lambda} + 1) \cdot p_0$. Hence, one set $B = 2^{t+\lambda} + 1$. If x is the secret share of the sum of two input wires, the corresponding bound B will be $2 \cdot (2^{t+\lambda} + 1)$. If x is the output of a degree reduction protocol, the maximum value of $\mathsf{Lift}(x)$ is reset to be $(2^{t+\lambda} + 1) \cdot p_0$. Hence, one could again pick $B = 2^{t+\lambda} + 1$. Consequently, this bound B only depends on the topology of the circuit, and parties could identify the same bound B without interaction.
- Party P_i locally computes $[-x]_i = (B \cdot p_0 - [x]_i) \mod p_i$.

Observe that the lifting integer of the secret shares $[-x]_i$ is now the integer $B \cdot p_0 - \mathsf{Lift}(x)$ and we have $(B \cdot p_0 - \mathsf{Lift}(x)) = -x \mod p_0$. Therefore, this sub-protocol correctly realizes the negation gate.

- **Preprocessing Phase.**
  - Parties generate $|C|$ fresh samples of $[r]^0, [r]^1$ (as described in F_{Random}).
  - Parties generate samples of the secret sharing $[0]$ of 0 (as described in F_{Random}). The number of instances equals to the number of output wires of C.
- **Online Phase.**
  - Parties sample a WRSS of their inputs and send it to all parties. The threshold parameter L in generating the secret shares is $2^{t+\lambda}$.
  - Addition Gate $x+y$: Parties locally compute $([x]_i + [y]_i) \mod p_i$ as the secret share of $x+y$.
  - Multiplication Gate $x \cdot y$: Parties locally compute $([x]_i \cdot [y]_i) \mod p_i$ as the secret share of $x \cdot y$. They then employ a degree reduction protocol F_{deg} and obtain $[z]_i$ as the new share where $z = x \cdot y$. In subsequent sections we will refer to this as F_{Mult}.
  - Negation Gate $-x$: Parties use the sub-protocol F_{neg}.
  - Scalar Multiplication Gate $c \cdot x$: Parties locally compute $(c \cdot [x]_i) \mod p_i$ as the secret share of $c \cdot x$. They then employ a degree reduction protocol F_{deg} and obtain $[z]_i$ as the new share where $z = c \cdot x$. In subsequent sections we will refer to this as F_{sMult}.
- **Reconstruct the Output.** For each output wire out, parties use the F_{Open} with input $[\mathsf{out}]$ to reconstruct the value out.

Fig. 3: Our Efficient Weighted MPC Protocol

5.5 Our Protocol

We are now ready to state our protocol in Figure 3. The correctness is straightforward as the reconstruction of the secret is correct for each sub-protocol.
For security, the following lemma is helpful. The proof is included in Section A.

Lemma 1 (Integer Masking Lemma). *Let p and $0 \leqslant r_1, r_2 < p$ be any integers. Let $M < N$ also be arbitrary integers. Let D be an arbitrary distribution over the universe $\{r_1, p + r_1, 2p + r_1, \ldots\} \cap [M]$. Then,*

$$\mathsf{SD}\left(\left(D + U_N \,\middle|\, U_N \mod p = r_2\right),\ \left(U_N \,\middle|\, U_N \mod p = r_1 + r_2\right)\right) \leqslant M/N + 2p/N,$$

where the addition is over the integers.

We provide some intuition about this lemma and why it is relevant to the security of the MPC protocol. Take the multiplication sub-protocol as an example. We need to argue that the reconstructed integer $[x] \cdot [y] + [r]^1$ could be simulated. Here, the integer corresponds to $[x] \cdot [y]$ is the distribution D and the integer corresponds to $[r]^1$ is the distribution U_N. The conditioning on $U_N \mod p$ is because of the adversary's secret share of $[r]^1$. That is, it knows that the remainder of U_N modulo some product of p_i. Now, this lemma states that as long as the range of the integer $[r]^1$ is sampled from a much larger domain (measured by N) compared to the maximum value of $[x] \cdot [y]$ (measured by M), one may simply sample the integer corresponds to $[x] \cdot [y] + [r]^1$ as a uniformly random one (given that it is consistent with the adversary's secret share).[14]

Security. The security proof essentially follows from the security of the WRSS and Lemma 1. Due to space constraints, we defer the security proof to the appendix section A.3.

6 Efficient Weighted Threshold Encryption Scheme

In this section, we will demonstrate the utility of our secret-sharing scheme by constructing a weighted threshold encryption scheme, where the size of the secret-key shares is small. Let us first define the primitive.

[14] The term p/N will always be small since p is the product of the adversary's p_i, which is at most 2^t. The WRSS scheme requires that whenever we pick a random lift integer, we shall always pick a domain much larger than 2^t.

Definition 2. *A public-key encryption scheme with weighted threshold decryption consists of a tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{PartialDec}, \mathsf{Reconstruct})$.

- $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Gen}(1^\lambda, \{w_1\}_{i=1}^n, T, t)$*: The* Gen *algorithm takes the security parameter* 1^λ *as input and a weighted access structure with privacy threshold* t *and reconstruction threshold* T *as inputs. It outputs a public key* pk *and a set of secret-key shares* $\{sk_i\}_{i=1}^n$*, where* sk_i *is given to the* i^{th} *party with weight* w_i.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: The* Enc *algorithm takes as input the public key* pk*, a message* m*, and outputs a ciphertext* c.
- $\mu \leftarrow \mathsf{PartialDec}(S, \mathsf{sk}', c)$*: The* $\mathsf{PartialDec}$ *algorithm takes as input a subset of* $S \subseteq [n]$*, a secret-key share* sk'*, a ciphertext* c*, and outputs partial decryption* μ.
- $m \leftarrow \mathsf{Reconstruct}(\{\mu_i\}_{i\in S}, c)$*: The* $\mathsf{Reconstruct}$ *is a deterministic algorithm that takes as input a set of partial decryptions* $\{\mu_i\}_{i\in S}$ *from a subset* S *of parties, a ciphertext* c*, and outputs a message* m*. In case of failure, it outputs* $\perp$.

It shall satisfy the following guarantees.

- ***Statistical Correctness.*** *For any weighted access structure* $(\{w_i\}_{i=1}^n, T, t)$*, authorized subset* $S \subseteq [n]$*, and message* m*, it holds that*

$$\Pr\left[m^* = m \ : \ \begin{array}{c} (\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Gen}(1^\lambda, \{w_1\}_{i=1}^N, T, t) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ \forall i \in S \ : \ \mu_i \leftarrow \mathsf{PartialDec}(S, \mathsf{sk}_i, c) \\ m^* \leftarrow \mathsf{Reconstruct}(\{\mu_i\}_{i\in S}, c) \end{array} \right] \geqslant 1 - \mathsf{negl}(\lambda).$$

- ε***-Strong CPA Security.*** *For any PPT adversary* A*, any weighted access structure* $(\{w_i\}_{i=1}^n, T, t)$*, and any unauthorized subset* $S \subseteq [n]$*, it holds that*

$$\Pr\left[b^* = b \ : \ \begin{array}{c} (\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Gen}(1^\lambda, \{w_1\}_{i=1}^N, T, t) \\ (m_0, m_1) \leftarrow A^{\mathcal{O}(\cdot)}(\mathsf{pk}, \{\mathsf{sk}\}_{i\in S}) \\ b \leftarrow \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b^* \leftarrow A^{\mathcal{O}(\cdot)}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in S}, m_0, m_1, c) \end{array} \right] \leqslant 1/2 + \varepsilon.$$

  Here, the oracle $\mathcal{O}(A, B, m)$ *takes as input an authorized set* A *and a subset* B *such that* $B \cup S$ *is unauthorized, and a message* m*. Its outputs are sampled from the following distribution*

$$\left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(\mathsf{pk}, m), \quad \forall i \in B,\ \mu_i = \mathsf{PartialDec}(A, \mathsf{sk}_i, c) \\ \textit{Output}\ \ (c, \{\mu_i\}_{i\in B}) \end{array} \right\}.$$

  In other words, the adversary is given access to the partial decryption oracle on honestly sampled *ciphertexts.*

Remark 3. We notice that, in the threshold setting, the plain CPA security (where the adversary does not have any access to partial decryption) is trivial to achieve. For instance, consider the following trivial scheme. Take any CPA-secure PKE scheme and any secret-sharing scheme. Sample the public key and secret key pair from the underlying PKE scheme and secret share the secret key with all parties. Now, the partial decryption algorithm simply outputs the secret share. Observe that even this scheme satisfies the plain CPA security.
Due to this observation, we consider a stronger definition, where the adversary has access to partial decryption on ciphertexts that are honestly sampled. This stronger CPA-security definition excludes the trivial construction above.

6.1 Building Blocks

ElGamal Encryption. Our construction is based on the ElGamal encryption system. Let us recall it. In the ElGamal encryption scheme, a group $\mathbb{G}$ with order p and generator g is sampled as $(\mathbb{G}, g) \leftarrow \mathsf{Setup}(1^\lambda)$.

The secret key sk and public key pk are sampled as s and g^s where $s \leftarrow \mathbb{F}_p$. To encrypt a message m, one sample a random $r \leftarrow \mathbb{F}_p$, and the ciphertext is defined as $(m \cdot \mathsf{pk}^r, g^r)$. Given a ciphertext (c_1, c_2), one could decrypt it as $c_1 \cdot c_2^{-\mathsf{sk}}$. This encryption scheme is semantically secure as long as the Decisional Diffie-Hellman (DDH) problem is hard in $\mathbb{G}$, which states that the following two distributions are computationally indistinguishable

$$(g, g^a, g^b, g^{ab}) \approx (g, g^a, g^b, g^c),$$

where $a, b, c \leftarrow \mathbb{F}_p$.
We need the following definitions regarding min-entropy and randomness extractor. For a distribution X, its min-entropy is defined as

$$H_\infty(X) = -\log\left(\max_x \ \Pr[X = x]\right).$$

Definition 3 (Randomness Extractor). *A function* $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is called a* (k, ε)-strong randomness extractor *if, for all distributions* X *over* $\{0,1\}^n$ *such that* $H_\infty(X) \geqslant k$*, we have*

$$\mathsf{SD}\left(\ \left(s, \mathsf{Ext}(X, s)\right)\ ;\ \left(U_{\{0,1\}^d}, U_{\{0,1\}^m}\right)\ \right) \leqslant \varepsilon,$$

where the seed s *is chosen uniformly at random from* $\{0,1\}^d$.

For our purpose, we may use the *leftover hash lemma* [22] as a concrete instantiation of the randomness extractor.

Definition 4 (Universal Hashing). *A family of hash function* $\{h_k : \{0,1\}^\lambda \to \{0,1\}^\alpha\}_k$*, where* $k \in \{0,1\}^\beta$ *is called a universal hashing function family if, for any two distinct inputs* $x, y \in \{0,1\}^\lambda$*, it holds that*

$$\Pr_{k \leftarrow \{0,1\}^\beta}[h_k(x) = h_k(y)] = 1/2^\alpha.$$

Instantiation. We provide a simple instantiation as follows. Given a message space $\{0,1\}^\lambda$, one picks $\alpha = \lambda/2$ and $\beta = \lambda$. A message $x \in \{0,1\}^\lambda$ is treated as a vector $(x_1, x_2) \in \mathbb{F}_{2^\alpha} \times \mathbb{F}_{2^\alpha}$. Similarly, the index of the hash function $k \in \{0,1\}^\lambda$ is also treated as $(k_1, k_2) \in \mathbb{F}_{2^\alpha} \times \mathbb{F}_{2^\alpha}$. Define the hash function output as

$$h_{k_1,k_2}(x_1, x_2) = k_1 \cdot x_1 + k_2 \cdot x_2,$$

where the operations are over $\mathbb{F}_{2^\alpha}$. One may verify that it is indeed a universal hash function.
For our purpose, observe that for any key $(k_1, k_2) \neq (0, 0)$ and any hash output $\sigma \in \{0,1\}^{\lambda/2}$, it holds that

$$H_\infty\left(U_{\{0,1\}^\lambda} \ \middle|\ (k_1, k_2),\ h_{k_1,k_2}\left(U_{\{0,1\}^\lambda}\right) = \sigma\right) = \lambda/2.$$

That is, a uniformly sampled message has at least $\lambda/2$ bits of entropy after being conditioned on the hash function output.

6.2 Our Construction

Our construction based on the ElGamal encryption scheme is in Figure 4.

Efficiency. The efficiency of our threshold encryption scheme inherits the efficiency of the WRSS scheme as the size of the secret key share is $O(w_i)$. Moreover, the partial decryption and reconstruction time is $O(W) + \mathsf{poly}(\lambda)$, where W is the total weights $\sum_{i \in S} w_i$. This is because every party is computing an $O(W)$-bit integer, i.e., $\mathsf{sk}_i \cdot \lambda_i \mod P_S$, which takes $O(W)$ time and the rest of the reconstruction time is independent of the weight and takes $\mathsf{poly}(\lambda)$ time.
In comparison, if one uses Shamir's secret sharing with the virtualization approach, every party needs to interpolate a degree-$(W-1)$ polynomial and evaluate it at 0. This needs at least $W \log W$ field operations based on fast Fourier transform techniques, which takes at least $O(W \cdot \lambda)$ time.

Correctness. Observe that the decryption is correct as long as it finds the correct index j^*. Furthermore, it might not find the correct j^* if and only if there is a collision for the universal hash function h_k. By the property of the universal hash function, for any $j \neq j^*$, the probability of the collision between j and j^* is $\exp(-\lambda)$. Therefore, by union bound, the probability of incorrectness is upper-bounded by $n \cdot \exp(-\lambda)$.

$\underline{\mathsf{Gen}(1^\lambda, \{w_1\}_{i=1}^n, T, t)}$. The public key and secret keys are set up as follows.

- Sample $(\mathbb{G}, g) \leftarrow \mathsf{Setup}(1^\lambda)$ and $s \leftarrow \mathbb{F}_p$.
- Set $\mathsf{pk} = s$. Use the WRSS scheme with access structure $(\{w_1\}_{i=1}^n, T, t)$ to secret share s as $s_1, \dots, s_n$. Set $\mathsf{sk}_i = s_i$.

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$. To encrypt a message m, one computes:

- Sample a random exponent $r \leftarrow \mathbb{F}_p$, a hash function $k \leftarrow \{0,1\}^\beta$, and a seed for the randomness extractor $\mathsf{sd} \leftarrow \{0,1\}^d$.
- The ciphertext is defined as

$$m \oplus \mathsf{Ext}(\mathsf{sd}, \mathsf{pk}^r),\ \mathsf{sd},\ g^r,\ k,\ h_k(\mathsf{pk}^r).$$

$\underline{\mu \leftarrow \mathsf{PartialDec}(S, \mathsf{sk}', c)}$. The partial decryption is defined as follows. Note that the authorized set S implicitly defined $P_S = \prod_{i \in S} p_i$ and also the Lagrange coefficients λ_i. That is, the unique integer λ_i that satisfies

$$\lambda_i = 1 \mod p_i \qquad \text{and} \qquad \forall j \in S \setminus \{i\},\ \lambda_i = 0 \mod p_j.$$

Parse the ciphertext c as above and the partial decryption outputs

$$(g^r)^{\left(\mathsf{sk}' \cdot \lambda_i \mod P_S\right)}.$$

$\underline{m \leftarrow \mathsf{Reconstruct}(\{\mu_i\}_{i \in S}, c)}$. Given all the partial decryptions $\{\mu_i\}_{i \in S}$, the reconstruction does the following. It set $\mu = \prod_{i \in S} \mu_i$ and computes

$$\mu,\ \mu \cdot (g^r)^{-P_S},\ \dots,\ \mu \cdot (g^r)^{-(|S|-1) \cdot P_S}.$$

It checks if there exists an j such that

$$h_k\left(\mu \cdot (g^r)^{-j \cdot P_S}\right) = h_k(\mathsf{pk}^r).$$

If such an j does not exist, it output $\perp$; otherwise, it finds any such j^* and outputs

$$c \oplus \mathsf{Ext}\left(\mathsf{sd}, \mu \cdot (g^r)^{-j^* \cdot P_S}\right).$$

Fig. 4: Our Efficient Threshold Encryption Scheme

Security. We now show the CPA security of our weighted public-key threshold encryption scheme. In particular, in the generic group model [28], we shall prove that our scheme satisfies ε-strong CPA-security where $\varepsilon = \mathsf{poly}(\lambda)/p_{\mathsf{min}}$ where $p_{\mathsf{min}} = \min_i p_i$. Therefore, as long as the minimum weight is large enough, e.g., $w_{\mathsf{min}} \geqslant \log^2 \lambda$, our threshold encryption scheme satisfies the $\mathsf{negl}(\lambda)$-strong CPA security.
We briefly explain why p_{min} needs to be large and we need the generic group model (instead of DDH). Note that if w_i is small, the total possibility of the secret share of party P_i is also small. Therefore, given the partial decryption output of P_i, one could use an exhaustive search (in time p_i) to find the exact s_i. Therefore, it is inevitable that the security depends on the minimum w_i.
Next, our proof relies on the generic group model as our WRSS is non-linear. In particular, for a linear partial decryption, given g^{s_i}, one could easily simulate $(g^r, (g^{s_i})^r)$, where $r \leftarrow \mathbb{F}_p$. However, in our case, given g^{s_i}, it is not clear how to simulate $(g^r, (g^r)^{(s_i \cdot \lambda_i \mod N)})$. Therefore, we have to rely on generic group to argue that this distribution is indistinguishable from two random group elements.

Proof. In Shoup's generic group model [28], there is an injective labeling oracle $\xi : \mathbb{F}_p \to \{0,1\}^\alpha$ that maps an exponent r to the label of the group element g^r. Additionally, there is an oracle $F\colon \{0,1\}^\alpha \times \{0,1\}^\alpha \to \{0,1\}^\alpha$ that maps (u, v) to $\xi(g^{a+b})$ if $u = \xi(g^a)$ and $v = \xi(g^b)$. That is, F is the oracle that helps compute the group operations on the labels. Now, consider a generic adversary $\mathcal{A}$ that plays the CPA security game. Without loss of generality, we may assume that all the queries that $\mathcal{A}$ asks to F are on labels that it receives from ξ, since we may assume that the range $\{0,1\}^\alpha$ is large enough such that it is exponentially hard to guess a valid label.
Our simulation strategy is as follows. For all the group elements in the security game, instead of querying ξ, we shall give a random string from $\{0,1\}^\alpha$ as its label. This includes the following queries. For every

partial decryption query to $\mathcal{O}$, it receives

$$g^r, \left\{ (g^r)^{(s_i \cdot \lambda_i \mod P_A)} \right\}_{i \in B},$$

where r is freshly sampled for each query. In the setup and challenge ciphertext, we also sample $g^s, g^{r^*}, g^{r^* \cdot s}$. Therefore, all the queries that $\mathcal{A}$ makes to F will be a linear combination of these group elements. Now if $\mathcal{A}$ asks a linear combination that has not been queried before, we answer this query with a random label. Otherwise, we answer it with the same label, consistent with $\mathcal{A}$'s view. This simulated view is identically distributed as the real view except when there is a collision on the exponent. For instance, if it happens that $s = r^*$, then g^s and g^{r^*} should have the same label in the real view, but in our simulated view, they have different labeling. We next argue that the collision probability is small. Consider any linear combination of these group elements. Note that if there exists an r such that the coefficient on r is non-zero, then the probability of collision is $1/p_0$. This is because r is a random exponent. Therefore, it suffices to argue that the probability that the adversary may find a set of β_i such that

$$\beta_0 \cdot 1 + \sum_{i \in B} \beta_i \cdot (s_i \cdot \lambda_i \mod P_A) = 0$$

is small. Note that since $B \cup S$ is unauthorized, $\{s_i\}_{i \in B}$ are uniformly random by the security of the WRSS. Therefore, for any such linear combination β_i, the probability that s_i satisfies the above equation is at most $1/p_{\mathsf{min}}$. Since the adversary makes some $q = \mathsf{poly}(\lambda)$ queries, by a union bound, the probability that any two of those queries cause a collision is at most $q^2/p_{\mathsf{min}} = \mathsf{poly}(\lambda)/p_{\mathsf{min}}$. In conclusion, if we simulate all the group elements in the security game as a random group element, this simulated view is at most $\mathsf{poly}(\lambda)/p_{\mathsf{min}}$ far from the real view.
Finally, in this simulated view, the choices bit b is hidden from the adversary because $\mathsf{Ext}(\mathsf{sd}, g^{r^* \cdot s})$ is statistically close to uniformly random by the property of the randomness extractor. This completes the proof that our threshold encryption scheme is $\mathsf{poly}(\lambda)/p_{\mathsf{min}}$-strong CPA secure.

7 Efficient Weighted Threshold Signature

We show how to apply our weighted MPC protocol in the context of threshold signatures. More specifically, we show how to construct a weighted multiparty signing protocol for ECDSA signatures. Such protocol is also known as weighted threshold signature.

7.1 ECDSA Signatures

We first briefly recall the ECDSA signature scheme below:

ECDSA Signature Scheme
Let G be the elliptic curve base point which generates a subgroup of some prime order q. Let $H(\cdot)$ be a cryptographic hash function. We use $a \times G$ to denote the multiplication of curve point G by a scalar a.

– $\mathsf{Gen}(1^\lambda)$: Sample signing key as $\mathsf{sk} \leftarrow \mathbb{F}_q$ and then set verification key as $\mathsf{vk} = \mathsf{sk} \times G$.
– $\mathsf{Sign}(\mathsf{sk}, m)$: Sample random element $k \leftarrow \mathbb{F}_q$. Compute curve point $(r_x, r_y) = k \times G$ and let $r = r_x$. Then set $\sigma = k^{-1} \cdot (H(m) + r \cdot \mathsf{sk})$. Output (r, σ).
– $\mathsf{Verify}(\mathsf{vk}, m, (r, \sigma))$: Compute $(r_x, r_y) = \sigma^{-1} \cdot H(m) \times G + \sigma^{-1} \cdot r \times \mathsf{vk}$. Then output 1 if and only if $r_x = r$.

Following the same general framework as previous approaches [24,18], our weighted threshold ECDSA signature scheme starts with a WRSS of the secret signing key sk among all parties. We described this step next.

Weighted Threshold ECDSA Key Generation Functionality $\mathcal{F}_{\mathsf{Gen}}(1^\lambda, T, t)$:
$\mathcal{F}_{\mathsf{Gen}}$ takes as input the security parameter 1^λ and CRT-based weighted (Ramp) secret-sharing scheme with respect to reconstruction threshold T and privacy threshold t. Then it does the following:

1. Sample a secret signing key $\mathsf{sk} \leftarrow \mathbb{F}_q$. Then it sets verification key as $\mathsf{vk} = \mathsf{sk} \times G$.
2. Generate a WRSS of sk : $\{[\mathsf{sk}]_i\} \leftarrow \mathsf{Share}(\mathbf{P}, T, t, \mathsf{sk})$. Then send $(\mathsf{vk}, \{[\mathsf{sk}]_i\})$ to each party i.

In order to build a weighted multiparty signing protocol for ECDSA signing functionality, we begin by describing the ideal signing functionality, step by step as follows:

ECDSA Signing Functionality F_{Sign}

1. Generate a secret random value $k \leftarrow \mathbb{F}_q$, and then compute (public) group element $k \times G$.
2. Parse $k \times G$ as curve point (r_x, r_y).
3. Compute multiplication between inverse of secret random value k and secret signing key sk. We denote this by $s = k^{-1} \cdot \mathsf{sk}$.
4. Compute two scalar multiplications: $k^{-1} \cdot H(m)$ and $r_x \cdot s$.
5. Add up the above two values and obtain σ.

Our weighted MPC protocol will proceed as follows: For the first step of signing, the secret random value k shall be contributed by all parties. More specifically, each party i will sample its own secret random value k_i, broadcast the group element $k_i \times G$, and then distribute the WRSS of k_i among all parties. This allows each party to obtain a share of the combined random value $k = \sum_{i\in[n]} k_i$ as well as the group element $k \times G = \sum_{i\in[n]} k_i \times G$. The subsequent steps naturally fit into our MPC protocol: step 2 only incurs public operations, and step 5 only incurs addition, both of which can be computed locally by every party. Step 3 involves first computing the inversion k^{-1} and then multiplying it with sk. Using the inversion protocol as suggested in [5], these operations can be handled via F_{Mult} and F_{Open}. Finally, step 4 involves scalar multiplications. While in our weighted MPC protocol parties need to run degree reduction to keep the integer value of share small for subsequent multiplications; here each party can perform scalar multiplication locally since there are no multiplications afterward.
We describe our weighted multiparty ECDSA signing protocol which realizes the ideal ECDSA signing functionality in Figure 5. We split our signing protocol into two phases: a pre-signing protocol which only depends on the shares of the signing key, followed by a non-interactive signing protocol which depends on the actual message.

Correctness and Security Both correctness and security of our weighted multiparty ECDSA signing protocol follow from these of weighted MPC protocol. The only catch is that we also need to simulate the value $k_i \times G$ sent by each honest party. However, since those values form an additive sharing of $k \times G$, they can be simulated given only $k \times G$.

Efficiency. The aforementioned pre-signing phase involves three rounds. However, instead of having the parties perform a multiplication protocol on $[\gamma]_i \cdot [k]_i$ and then open the result, we can directly let the parties open the multiplication of their local shares, thus bringing the pre-signing phase to two rounds. The communication cost per party in the pre-signing phase is $O(W + \lambda)$.
The online signing phase is non-interactive. Each party i broadcasts a share of final signature $[\sigma]_i$ which has size $O(w_i)$.

Weighted Threshold ECDSA Signing Protocol
Let there be a total of n parties where each party i has base p_i, its secret input $[\mathsf{sk}]_i$, public input vk and m. Let $S \subseteq [n]$ be the subset of parties participating in the weighted threshold ECDSA signing protocol and let W be the total weight of these parties. We will rely on the following protocols: $(F_{\mathsf{Random}}, F_{\mathsf{Mult}}, F_{\mathsf{Open}})$.

Pre-signing Phase

1. Parties generate CRT shares of random values $\{[\gamma]_i\}_{i\in S} \leftarrow F_{\mathsf{Random}}(\gamma = \sum_{i\in S} \gamma_i)$, and $\{[k]_i\}_{i\in S} \leftarrow F_{\mathsf{Random}}(k = \sum_{i\in S} k_i)$. Each party i also broadcasts $k_i \times G$.
2. Parties compute $\{[\delta]_i\}_{i\in S} = F_{\mathsf{Mult}}(\{[\gamma]_i\}_{i\in S}, \{[k]_i\}_{i\in S})$, and $\{[\theta]_i\}_{i\in S} = F_{\mathsf{Mult}}(\{[\gamma]_i\}_{i\in S}, \{[\mathsf{sk}]_i\}_{i\in S})$.
3. Parties compute $\delta = F_{\mathsf{Open}}(\{[\delta]_i\}_{i\in S}))$. Then they compute $R = \sum_{i\in S} k_i \times G$ and set curve point $R = (r_x, r_y)$
4. Each party i computes $[\sigma^0]_i = \delta^{-1} \cdot [\gamma]_i$ and $[\sigma^1]_i = r_x \cdot \delta^{-1} \cdot [\theta]$.
   Note that $[\sigma^0]_i$ is a share of k^{-1} and $[\sigma^1]_i$ is a share of $k^{-1} \cdot \mathsf{sk}$.
5. Each party i saves the values $(r_x, [\sigma^0]_i, [\sigma^1]_i)$.

Signing Phase

1. Each party i locally computes $[\sigma]_i = H(m) \cdot [\sigma^0]_i + [\sigma^1]_i$.
2. Parties compute $\sigma = F_{\mathsf{Open}}(\{[\sigma]_i\}_{i\in S}))$. The signature of m is (σ, r_x).

Fig. 5: Weighted Threshold ECDSA Signing

Acknowledgements

The first author was supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. The second author was supported in part by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, AFOSR Award FA9550-19-1-0200, Office of Naval Research Grant N00014-19-1-2294, JP Morgan Faculty Award, and research gifts from Ethereum, Stellar and Cisco. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA.

References

1. Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 441–471, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. 4
2. Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 280–293, Chicago, IL, USA, June 22–26, 2020. ACM Press. 4
3. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, January 2017. 12
4. Charles Asmuth and John Bloom. A modular approach to key safeguarding. *IEEE Trans. Inf. Theory*, 29(2):208–210, 1983. 2, 4, 9
5. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. 19
6. Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. *IACR Cryptol. ePrint Arch.*, page 30, 2001. 4
7. Amos Beimel and Enav Weinreb. Monotone circuits for monotone weighted threshold functions. *Inf. Process. Lett.*, 97(1):12–18, 2006. 3
8. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press. 2, 7
9. Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs*, 2021. 1
10. Pyrros Chaidos and Aggelos Kiayias. Mithril: Stake-based threshold multisignatures. Cryptology ePrint Archive, Report 2021/916, 2021. https://eprint.iacr.org/2021/916. 3
11. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. 8
12. Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: Secure multiparty computation with dynamic participants. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 94–123, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 3
13. Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. 8
14. Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany. 1
15. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 1

16. Steve Ellis, Ari Juels, and Sergey Nazarov. Chainlink: A decentralized oracle network. *Retrieved March*, 11:2018, 2017. 1
17. Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 699–710, Victoria, BC, Canada, May 4–6, 1992. ACM Press. 8
18. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1179–1194, Toronto, ON, Canada, October 15–19, 2018. ACM Press. 18
19. Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 64–93, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 3
20. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 1
21. Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. In *31st Annual ACM Symposium on Theory of Computing*, pages 225–234, Atlanta, GA, USA, May 1–4, 1999. ACM Press. 2, 4, 9
22. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 16
23. Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 1
24. Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1837–1854, Toronto, ON, Canada, October 15–19, 2018. ACM Press. 18
25. Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 699–708, Los Angeles, CA, USA, June 25–29, 2018. ACM Press. 4
26. Maurice Mignotte. How to share a secret? In Thomas Beth, editor, *Advances in Cryptology – EUROCRYPT'82*, volume 149 of *Lecture Notes in Computer Science*, pages 371–375, Burg Feuerstein, Germany, March 29 – April 2, 1983. Springer, Heidelberg, Germany. 2, 4, 9
27. National Institute of Standards and Technology. Multi-party threshold cryptography, 2018–present. 1
28. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. 17
29. C Stathakopoulous and Christian Cachin. Threshold signatures for blockchain systems. *Swiss Federal Institute of Technology*, 30, 2017. 1
30. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 1
31. Xukai Zou, Fabio Maino, Elisa Bertino, Yan Sui, Kai Wang, and Feng Li. A new approach to weighted multi-secret sharing. In Haohong Wang, Jin Li, George N. Rouskas, and Xiaobo Zhou, editors, *Proceedings of 20th International Conference on Computer Communications and Networks, ICCCN 2011, Maui, Hawaii, USA, July 31 - August 4, 2011*, pages 1–6. IEEE, 2011. 3

A Missing Proofs

A.1 Proof of Claim 1

Proof. Suppose $L = M \cdot q + r$ for some $0 \leqslant r < M$. Let $L' = M \cdot q$. We have

$$\begin{aligned}
&\mathsf{SD}\left((s + p \cdot U_L) \mod M \ , \ U_M\right) \\
\leqslant&\mathsf{SD}\left((s + p \cdot U_L) \mod M \ , \ (s + p \cdot U_{L'}) \mod M\right) + \mathsf{SD}\left((s + p \cdot U_{L'}) \mod M \ , \ U_M\right) \\
&\hfill \text{(Triangle inequality)} \\
\leqslant&\mathsf{SD}\left(U_L, U_{L'}\right) + \mathsf{SD}\left((s + p \cdot U_{L'}) \mod M \ , \ U_M\right). \qquad \text{(Data processing inequality)}
\end{aligned}$$

Observe that

$$\mathsf{SD}\left((s + p \cdot U_{L'}) \mod M \ , \ U_M\right) = 0$$

since p is coprime with M and L' is a multiple of M. On the other hand,

$$\mathsf{SD}\left(U_L, U_{L'}\right) = r/N < M/N.$$

This completes the proof.

A.2 Proof of Lemma 1

Proof. We first truncate N to be a multiple of p. Suppose $N = N' + r'$, where $N' = m \cdot p$ and $0 \leqslant r' < p$. Note that for any r,

$$\mathsf{SD}\left((U_N | U_N \mod p = r) \ , \ (U_{N'} | U_{N'} \mod p = r)\right) \leqslant p/N.$$

Hence, it suffices to prove

$$\mathsf{SD}\left(\left(D + U_{N'} \;\middle|\; U_{N'} \mod p = r_2\right) \ , \ \left(U_{N'} \;\middle|\; U_{N'} \mod p = r_1 + r_2\right)\right) \leqslant M/N.$$

Define the set S as

$$S = \{x \mid M + 1 \leqslant x < N', \ x \mod p = r_1 + r_2\}$$

Observe that for all $x \in S$, conditioned on $U_{N'} \mod p = r_2$, we have

$$\Pr[D + U_{N'} = x] = \sum_{y \in \mathsf{supp}} \Pr[D = y] \cdot \Pr[U_{N'} = x - y] = \sum_{y=1}^{M} \Pr[D = y] \cdot \frac{p}{N'} = \frac{p}{N'}.$$

Moreover, conditioned on $U_{N'} \mod p = r_1 + r_2$, $\Pr[U_{N'} = x] = \frac{p}{N'}$. Therefore, the statistical distance is strictly bounded by the maximum probability that the distribution $D + U_{N'}$ and $U_{N'}$ falls outside the set S. Since the probability mass of the set S is already

$$\frac{p}{N'} \cdot |S| = \frac{p}{N'} \cdot \frac{N' - M}{p} = 1 - M/N'.$$

This concludes that the statistical distance is at most M/N.

A.3 Security Proof for Weighted MPC Protocol

Now, we argue the security of our weighted MPC protocol. In particular, the simulator $\mathcal{S}$ does the following.

- Preprocessing: The simulator simulates the protocol honestly. There is no simulation error for this.
- Input Sharing: For any malicious party $i \in I$, the simulator samples a fresh secret sharing of its input x_i. For any honest party $j \in \overline{I}$, the simulator samples a fresh secret sharing of 0 as the secret share that the adversary receives. The simulation error in this step is $2^{-\lambda}$ since the adversary corrupts at most t weights and the WRSS is sampled with threshold $L = 2^{t+\lambda}$. By the security of the WRSS, the statistical distance between the simulated view and the real view is at most $2^{-\lambda}$.

- Addition: The simulator locally computes the addition of the secret shares of the adversary. There is no simulation error in this step.
- Negation: The simulator locally computes the negation of the secret shares of the adversary. There is no simulation error in this step.
- Multiplication $x \cdot y$: The simulator locally computes the secret share $[x]_i \cdot [y]_i + [r]_i^1$. Recall that $r = r_1 + \cdots + r_n$ is the summation of randomness from every party. For the malicious party r_i, the simulator knows the corresponding integer R_i. For all honest parties r_i, it samples the integer R_i honestly except for one honest party, say party P_1. It samples R_1 as follows. It picks a random element $r' \leftarrow \mathbb{F}$ and picks a random integer $R_1' = r' + u \cdot p_0$ where $u \leftarrow \{1, \ldots, 2^{2t+5\lambda}\}$ conditioned on that
$$\forall i \in I, \quad R_1' \mod p_i = [x]_i \cdot [y]_i + [r_1]_i^1.$$
Finally, it computes
$$R' = R_1' + R_2 + \cdots + R_n.$$
It proceeds to simulate the view of the adversary by using $R' \mod p_i$ as party P_i's message in the degree reduction protocol. It then subtracts r' from the secret shares $[r]^0$ as the sharing of $x \cdot y$.
Next, we argue that the simulation error is $\exp(-\lambda)$. Note that the adversary's view in the degree reduction protocol is uniquely determined by the reconstructed integer R'. Hence, we directly argue that the distribution of R' is $\exp(-\lambda)$-close to the distribution of the integer associated with $[x] \cdot [y] + [r]$. In particular, it suffices to argue that the distribution of R_1' is $\exp(-\lambda)$-close to the distribution of the integer associated with $[x] \cdot [y] + [r_1]^1$.
Now, this is exactly what Lemma 1 proves. That is, the distribution D describes the distribution of the integer associated with $[x] \cdot [y]$. Due to our invariant, both integers X and Y are at most $\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda}$. Hence, the integer associated with $[x] \cdot [y]$ is some distribution over the range $\mathsf{poly}(\lambda) \cdot 2^{2t+4\lambda}$. The distribution U_N describes the distribution of $R_1 = \mathsf{Lift}([r]^1)$ and R_1', which is uniformly distributed over the range $2^{2t+5\lambda}$. Lemma 1 implies that the simulation error here is bounded by
$$\frac{\mathsf{poly}(\lambda) \cdot 2^{2t+4\lambda}}{2^{2t+5\lambda}} = \exp(-\lambda).$$
- Output Reconstruction: The simulator locally computes $[\mathsf{out}]_i + [0]_i \mod p_i$. It proceeds to simulate the reconstruction message by simulating the reconstructed integer. This is entirely analogous to the multiplication gate case. Recall that the sharing $[0]$ is the summation of all parties' share of $[0]$. The simulator picks one honest party, say P_1, and proceeds to simulate the integer associated with $[0_1] + [\mathsf{out}]$. This is again by our integer masking lemma (Lemma 1). Note that the integer associated with $[\mathsf{out}]$ is of range $\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda}$ and $[0_1]$ is sampled by picking an integer over range $2^{t+3\lambda}$. Hence, the simulation error in this step is at most
$$\frac{\mathsf{poly}(\lambda) \cdot 2^{t+2\lambda}}{2^{t+3\lambda}} = \exp(-\lambda).$$

Overall, the simulation error of the simulator is at most $\exp(-\lambda)$, which completes the proof.