Garbled Circuits with 1 Bit per Gate

Hanlin Liu
Northwestern University
hanlin.liu@northwestern.edu

Xiao Wang
Northwestern University
wangxiao@northwestern.edu

Kang Yang
State Key Laboratory of Cryptology
yangk@sklc.org

Yu Yu
Shanghai Jiao Tong University
Shanghai Qi Zhi Institute
yuyu@yuyu.hk

Abstract

We present a garbling scheme for Boolean circuits with 1 bit per gate communication based on either ring learning with errors (RLWE) or NTRU assumption, with key-dependent message security. The garbling consists of 1) a homomorphically encrypted seed that can be expanded to encryption of many pseudo-random bits and 2) one-bit stitching information per gate to reconstruct garbled tables from the expanded ciphertexts. By using low-complexity PRGs, both the garbling and evaluation of each gate require only $O(1)$ homomorphic addition/multiplication operations without bootstrapping.

1 Introduction

Garbled circuits (GCs), first introduced by Yao [Yao86] in the 1980s, are of central importance in cryptography and have been used in secure two-party computation [LP09], zero-knowledge proofs [JKO13], identity-based encryption [DG17], etc. Optimizing the size of GCs has been an important task both concretely and asymptotically. For concrete optimizations, significant efforts [BMR90, NPS99, KS08, PSSW09, KMR14, ZRE15, GLNP15, RR21] have been made to reduce the size of GCs, leading to notable improvements. Among these garbling schemes, Rosulek and Roy [RR21] achieve the lowest communication, requiring only $1.5\lambda + 5$ bits per AND gate and no communication for XOR gates, where λ is the computational security parameter. On the other hand, GCs with asymptotically sublinear size can be achieved through reusable GCs [GKP+13, GGH+13b, BGG+14, Agr17] based on strong assumptions, such as the subexponential hardness of LWE or multilinear maps. However, concrete communication savings only emerge when the circuit is astronomically large due to the hidden constant in the asymptotic complexity. In summary, traditional GCs use symmetric-key techniques and need $O(\lambda|C|)$ bits of communication, while "fancy" garbled circuits can achieve $\mathsf{poly}(\log|C|, \lambda)$ bits of communication with unclear concrete efficiency, where $|C|$ is the circuit size. There is a huge gap between practically efficient $O(\lambda|C|)$-sized solutions and completely impractical $\mathsf{poly}(\log|C|, \lambda)$-sized solutions. In this paper,

we present concretely efficient GCs from the super-polynomial hardness of lattice assumptions. We aim to achieve concrete communication savings compared to traditional GCs without using heavy mechanisms such as attribute-based encryption, functional encryption, or fully homomorphic encryption.

1.1 Our Contributions

We introduce a garbling scheme based on lattice assumption, achieving just 1 bit of communication per gate. From a high-level view, our garbling scheme uses a specialized somewhat homomorphic encryption (SWHE) scheme to assemble garbled circuits, with the garbled tables divided into two parts:

- **Ciphertext that encrypts a random seed.** The garbler sends an encrypted random bit for each gate to the evaluator using our special SWHE scheme. Sending encrypted bits directly leads to a large communication cost. To optimize this, we first generate an encrypted random seed and then expand it using homomorphic evaluation of a low-depth pseudo-random generator (PRG).

- **"Stitching information", 1 bit per gate.** The second part includes "stitching information" that converts pseudo-random encrypted bits into a garbled circuit. Our scheme uses this "stitching information" to flip the encrypted bit for each gate with only one bit of communication per gate.

We prove the security of our garbling scheme in the simulation-based model. Our SWHE scheme requires key-dependent message (KDM) security to achieve the special distributed decryption properties. We present two efficient instantiations under either the ring learning with errors (RLWE) assumption [LPR10] (building upon prior work [GSW13]) or the NTRU assumption [HPS98, PS21] (building upon prior works [LTV12, BLLN13, Klu22, BIP+22]).

Our scheme is highly efficient in communication. In particular, the GC size is as small as $|C| + \widetilde{O}(\lambda^2)$ bits (more specifically, $|C|$ bits plus λ GSW ciphertexts, each of size $O(\lambda \cdot \mathsf{poly}\log\lambda)$ bits), significantly less than the state-of-the-art classical garbling scheme that needs $1.5\lambda + 5$ bits per AND gate [RR21]. Using a low-complexity PRG (e.g., [App12, AIK04, AIK08, AK19, CM01]), the computational cost is dominated by only $O(1)$ homomorphic addition/multiplication operations per gate, without any FHE bootstrapping.

1.2 Comparison with Other Solutions

Succinct/reusable garbling and randomized encoding. Succinct garbling/randomized encoding schemes [LP14, CHJV15, BGL+15, KLW15, AL18] and reusable garbling schemes [GKP+13, GGH+13b, BGG+14, Agr17] rely on computation-heavy cryptographic primitives and strong assumptions, such as indistinguishability obfuscation ($i\mathcal{O}$) for $\mathsf{P/poly}$, subexponential LWE, or assumptions related to multilinear maps. Our scheme has higher communication than these schemes but more conservative assumptions and better concrete efficiency.

Fully homomorphic encryption. Fully homomorphic encryption (FHE) [Gen09] is an alternative approach to design constant-round secure two-party computation (2PC). The communication is linear in the input size and independent of the circuit size by encrypting the input bits. However, the computation cost of FHE is high, primarily due to the expensive bootstrapping process required. In contrast, our scheme does not need bootstrapping and involves only $O(1)$ homomorphic addition/multiplication operations. In addition, there exists the efficient black-box approach

(e.g., the "cut-and-choose" technique [LP15]) to transform a semi-honest 2PC protocol based on our garbling scheme into a maliciously secure protocol. However, using a FHE scheme to design a maliciously secure 2PC protocol requires the use of homomorphic message authentication codes [GW13, CF13, FGP14] or zero-knowledge proofs [FNP20, BCFK21, GNS23], which brings significantly more computational overhead.

2 Technical Overview

In this section, we provide a high-level explanation of our garbling scheme. As a warm-up, we present an intermediate garbling scheme in the random oracle (RO) model. Next, we explain how to adapt the same idea but replace RO with a special homomorphic encryption (HE) scheme, reducing communication size to 1 bit per gate. Finally, we show how to instantiate the HE scheme under the RLWE assumption. Below, we use $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$ to denote a polynomial ring with integer coefficients modulo a polynomial $X^n + 1$, and define $\mathcal{R}_p = \mathcal{R}/p\mathcal{R} = \mathbb{Z}_p[X]/(X^n + 1)$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$.

2.1 Constructing Garbled Circuits from Random Oracle

We present a garbling scheme in the RO model for Boolean circuits, but the labels are ring elements over $\mathcal{R}_p$, with an even p. Given a ring element A, we define $\mathsf{LSB}(A) = A[1] \mod 2$, where $A[1]$ is the first coefficient of A. Since p is even, for any $X, Y \in \mathcal{R}_p$, we have $\mathsf{LSB}(X + Y) = \mathsf{LSB}(X) \oplus \mathsf{LSB}(Y)$ and $\mathsf{LSB}(-X) = \mathsf{LSB}(X)$. Similar to classic garbling schemes, we have a global offset $\Delta \in \mathcal{R}$ such that $\mathsf{LSB}(\Delta) = 1$. We also assign each wire a pair of elements over $\mathcal{R}_p$, namely A^0 and A^1, as the 0-label and 1-label of this wire. For each wire, the 0-label A^0 is uniform while the 1-label is defined to be $A^1 = A^0 + (-1)^a \cdot \Delta$, where the wire mask $a = \mathsf{LSB}(A^0)$. Given these definitions, we can obtain the following quick facts. For any $v \in \{0, 1\}$,

I $A^v = A^0 + (-1)^a \cdot v \cdot \Delta$.

We prove this in two cases: for $v = 0$, it simplifies to $A^0 = A^0$, which is obviously true. For $v = 1$, it simplifies to $A^1 = A^0 + (-1)^a \cdot \Delta$, which is true based on the definition of A^1.

II $\mathsf{LSB}(A^v) = a \oplus v$.

Applying $\mathsf{LSB}()$ to I, we get $\mathsf{LSB}(A^v) = \mathsf{LSB}(A^0 + (-1)^a \cdot v \cdot \Delta)$. Since p is even, it equals to $\mathsf{LSB}(A^0) \oplus \mathsf{LSB}((-1)^a \cdot v \cdot \Delta) = a \oplus ((-1)^a \cdot v \mod 2) = a \oplus v$.

III $A^v = A^a + \mathsf{LSB}(A^v) \cdot \Delta$.

From I, we have $A^v - A^a = (-1)^a \cdot (v - a) \cdot \Delta$. It can be verified case by case that $(-1)^a \cdot (v - a)$ always equals $v \oplus a$.

For a gate that computes a function $g : \{0, 1\}^2 \to \{0, 1\}$, there are two pairs of garbled labels, (A^0, A^1) and (B^0, B^1), one for each input wire, and a pair of garbled labels, (C^0, C^1), for the output wire. Correspondingly, the three wire masks are $a = \mathsf{LSB}(A^0)$, $b = \mathsf{LSB}(B^0)$, and $c = \mathsf{LSB}(C^0)$. We maintain the invariant that if the underlying wire value is v for a wire with labels (A^0, A^1), then the evaluator should obtain A^v.

For any pair of masked bits $i, j \in \{0,1\}$, the underlying real input values are $a \oplus i$ and $b \oplus j$, the real output bit $z_{i,j} = g(a \oplus i, b \oplus j) \in \{0,1\}$ and the garbled output label would be $C^{z_{i,j}} = C^0 + (-1)^c \cdot z_{i,j} \cdot \Delta$ (due to I). Let $H : \mathcal{R}_p^2 \to \mathcal{R}_p$ be a hash function modeled as a random oracle. The garbled table of this gate is defined below, where the gate index is omitted for simplicity.

Masked bits	Input labels	Output labels	Garbled table
$(0,0)$	(A^a, B^b)	$C^{g(a,b)}$	$\tau_0 = H(A^a, B^b) - C^{g(a,b)}$
$(1,0)$	$(A^{a\oplus1}, B^b)$	$C^{g(a\oplus1,b)}$	$\tau_1 = H(A^{a\oplus1}, B^b) - C^{g(a\oplus1,b)}$
$(0,1)$	$(A^a, B^{b\oplus1})$	$C^{g(a,b\oplus1)}$	$\tau_2 = H(A^a, B^{b\oplus1}) - C^{g(a,b\oplus1)}$
$(1,1)$	$(A^{a\oplus1}, B^{b\oplus1})$	$C^{g(a\oplus1,b\oplus1)}$	$\tau_3 = H(A^{a\oplus1}, B^{b\oplus1}) - C^{g(a\oplus1,b\oplus1)}$

The evaluator, given input labels $A \in \{A^0, A^1\}$ and $B \in \{B^0, B^1\}$, computes the masked values $\alpha = \mathsf{LSB}(A)$ and $\beta = \mathsf{LSB}(B)$, and then calculates the output label $C = H(A, B) - \tau_{\alpha+2\beta}$. The invariant II implies that $A = A^{a\oplus\alpha}$ and $B = B^{b\oplus\beta}$, so $C = H(A^{a\oplus\alpha}, B^{b\oplus\beta}) - \tau_{\alpha+2\beta} = C^{g(a\oplus\alpha,b\oplus\beta)} = C^{z_{\alpha,\beta}}$.

Note that we use ring elements to represent the labels instead of bit strings for later lattice-based optimization. Previous works that used ring elements for garbled labels focused on arithmetic garbling, whereas we still concentrate on Boolean circuits, which leads to more complex label definitions.

Optimization via garbled row reduction. We can optimize the scheme above by setting $\tau_0 = 0$ and deriving the output labels accordingly, following the garbled row reduction (GRR) technique [NPS99]. Specifically, we first get $C^{z_{0,0}} = H(A^a, B^b)$ where $z_{0,0} = g(a,b)$. Taking $\mathsf{LSB}()$ on both sides, we get $c = \mathsf{LSB}(H(A^a, B^b)) \oplus z_{0,0}$ using II. Then we can obtain $C^0 = C^{z_{0,0}} - (-1)^c \cdot z_{0,0} \cdot \Delta$ (I). Now we can compute, for any $i, j \in \{0,1\}$, $C^{z_{i,j}} = C^0 + (-1)^c \cdot z_{i,j} \cdot \Delta = H(A^a, B^b) + (-1)^c \cdot (z_{i,j} - z_{0,0}) \cdot \Delta$ and the garbled table is shown as below. The gate-evaluation process is the same as that of the previous scheme.

Input labels	Truth table	Garbled table
(A^a, B^b)	$z_{0,0} = g(a,b)$	$\tau_0 = 0$
$(A^{a\oplus1}, B^b)$	$z_{1,0} = g(a\oplus1, b)$	$\tau_1 = H(A^{a\oplus1}, B^b) - H(A^a, B^b) - (-1)^c \cdot (z_{1,0} - z_{0,0}) \cdot \Delta$
$(A^a, B^{b\oplus1})$	$z_{0,1} = g(a, b\oplus1)$	$\tau_2 = H(A^a, B^{b\oplus1}) - H(A^a, B^b) - (-1)^c \cdot (z_{0,1} - z_{0,0}) \cdot \Delta$
$(A^{a\oplus1}, B^{b\oplus1})$	$z_{1,1} = g(a\oplus1, b\oplus1)$	$\tau_3 = H(A^{a\oplus1}, B^{b\oplus1}) - H(A^a, B^b) - (-1)^c \cdot (z_{1,1} - z_{0,0}) \cdot \Delta$

Abstracting our optimizations via homomorphic RO. To build towards our final scheme, we introduce another garbling scheme that uses a random oracle H with a special homomorphic property. Note that while our final scheme leverages properties similar to this homomorphic RO, we do not directly instantiate this RO. More specifically, given a global key $\Delta \in \mathcal{R}$ with $\mathsf{LSB}(\Delta) = 1$, the special homomorphism states that for any $X, Y \in \mathcal{R}_p$ with $\mathsf{LSB}(X) = \mathsf{LSB}(Y) = 0$ and any $i, j \in \{0,1\}$, we have

$$H(X + i \cdot \Delta, Y + j \cdot \Delta) = H(X, Y) + i \cdot H(\Delta, 0) + j \cdot H(0, \Delta) + i \cdot j \cdot H(\Delta, \Delta) \,.$$

We introduce this homomorphism to make the garbled table independent of input labels but still dependent on wire masks. Thus, this makes it possible to connect independent garbled tables without sending correction information linear to the size of garbled labels eventually.

We now apply this homomorphic RO to the above GRR scheme. We have already known from II and III that $A^{a\oplus i} = A^a + \mathsf{LSB}(A^{a\oplus i})\cdot\Delta = A^a + (a\oplus(a\oplus i))\cdot\Delta = A^a + i\cdot\Delta$ and that $B^{b\oplus j} = B^b + j\cdot\Delta$. Now, due to the homomorphic property of H and that $\mathsf{LSB}(A^a) = \mathsf{LSB}(B^b) = 0$ (II), we have that for any $i, j \in \{0,1\}$,

$$H(A^{a\oplus i}, B^{b\oplus j}) = H(A^a, B^b) + i\cdot H(\Delta, 0) + j\cdot H(0,\Delta) + i\cdot j\cdot H(\Delta,\Delta).$$

Plugging it back to the GRR garbled table above, we can simplify the three rows as follows.

$$\begin{aligned}
\tau_1 &= H(A^{a\oplus 1}, B^b) - H(A^a, B^b) - (-1)^c\cdot(z_{1,0} - z_{0,0})\cdot\Delta \\
&= H(\Delta, 0) - (-1)^c\cdot(z_{1,0} - z_{0,0})\cdot\Delta\ , \\
\tau_2 &= H(A^a, B^{b\oplus 1}) - H(A^a, B^b) - (-1)^c\cdot(z_{0,1} - z_{0,0})\cdot\Delta \\
&= H(0, \Delta) - (-1)^c\cdot(z_{0,1} - z_{0,0})\cdot\Delta\ , \\
\tau_3' &= H(A^{a\oplus 1}, B^{b\oplus 1}) - H(A^a, B^b) - (-1)^c\cdot(z_{1,1} - z_{0,0})\cdot\Delta \\
&= H(\Delta, 0) + H(0,\Delta) + H(\Delta,\Delta) - (-1)^c\cdot(z_{1,1} - z_{0,0})\cdot\Delta\ ,
\end{aligned}$$

where τ_1 and τ_2 are in the same format but not τ_3'. To align all of them, we instead define τ_3 slightly differently:

$$\tau_3 = \tau_3' - \tau_1 - \tau_2 = H(\Delta,\Delta) - (-1)^c\cdot(z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0})\cdot\Delta.$$

Now with τ_1, τ_2, and τ_3, both parties can reconstruct the garbled table equivalent to the GRR garbled table above by computing $\tau_3' = \tau_1 + \tau_2 + \tau_3$.

Input labels	Truth table	Garbled table
(A^a, B^b)	$z_{0,0} = g(a,b)$	$\tau_0 = 0$
$(A^{a\oplus 1}, B^b)$	$z_{1,0} = g(a\oplus 1, b)$	$\tau_1 = H(\Delta,0) - (-1)^c\cdot(z_{1,0} - z_{0,0})\cdot\Delta$
$(A^a, B^{b\oplus 1})$	$z_{0,1} = g(a, b\oplus 1)$	$\tau_2 = H(0,\Delta) - (-1)^c\cdot(z_{0,1} - z_{0,0})\cdot\Delta$
$(A^{a\oplus 1}, B^{b\oplus 1})$	$z_{1,1} = g(a\oplus 1, b\oplus 1)$	$\tau_3 = H(\Delta,\Delta) - (-1)^c\cdot(z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0})\cdot\Delta$

The gate-evaluation process works as follows: given input labels A and B, the evaluator first computes the masked bits $\alpha = \mathsf{LSB}(A)$ and $\beta = \mathsf{LSB}(B)$, and then computes the output label $C = H(A,B) - (\alpha\cdot\tau_1 + \beta\cdot\tau_2 + \alpha\beta\cdot\tau_3)$. To verify correctness, we use the fact that $A = A^{a\oplus\alpha}$ and $B = B^{b\oplus\beta}$ (II), which leads to $C = C^{z_{\alpha,\beta}}$, since

$$\begin{aligned}
& C - H(A^a, B^b) \\
=& H(A^{a\oplus\alpha}, B^{b\oplus\beta}) - H(A^a, B^b) - (\alpha\cdot\tau_1 + \beta\cdot\tau_2 + \alpha\beta\cdot\tau_3) \\
=& H(A^a + \alpha\Delta, B^b + \beta\Delta) - H(A^a, B^b) - (\alpha\cdot\tau_1 + \beta\cdot\tau_2 + \alpha\beta\cdot\tau_3) \\
=& \alpha\cdot H(\Delta,0) + \beta\cdot H(0,\Delta) + \alpha\beta\cdot H(\Delta,\Delta) - (\alpha\cdot\tau_1 + \beta\cdot\tau_2 + \alpha\beta\cdot\tau_3) \\
=& (-1)^c\cdot\Delta\cdot\big(\alpha\cdot(z_{1,0} - z_{0,0}) + \beta\cdot(z_{0,1} - z_{0,0}) + \alpha\beta\cdot(z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0})\big) \\
=& (-1)^c\cdot(z_{\alpha,\beta} - z_{0,0})\cdot\Delta\ .
\end{aligned}$$

Note that all three garbled rows can now be viewed as "encryptions" of bits. For example, τ_1 can be viewed as the encryption of the bit $(-1)^c\cdot(z_{1,0} - z_{0,0})$ using the key $(\Delta, 0)$. Homomorphic RO enables the evaluator to assemble a valid GC using these encrypted bits and evaluate it easily. Once we reduce the GC to a set of encrypted bits, intuitively, we can send many encrypted random bits and correct them with very small communication.

2.2 Imitating a Homomorphic RO

Our final garbling scheme is conceptually similar to the one based on a homomorphic RO; however, we use a specialized homomorphic encryption (HE) scheme instead. Below, we first introduce the properties of this specialized HE scheme

Defining SWHE with distributed decryption. We define a private-key somewhat homomorphic encryption (SWHE) scheme with distributed decryption, consisting of three polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Here, the key-generation algorithm Gen and the encryption algorithm Enc follow the standard definition. However, in contrast to the standard decryption, given a secret key sk and a ciphertext $\tau = \mathsf{Enc}(sk, m)$, the decryption algorithm $\mathsf{Dec}(sk, \tau)$ outputs $m \cdot sk$ instead of m, following the GSW scheme [GSW13].

We assume that a secret key $sk \in \mathcal{R}$ with $\mathsf{LSB}(sk) = 1$ is used, where sk plays the role of Δ in the application of the above garbling scheme. This scheme must satisfy the following properties of message homomorphism and distributed decryption, which are informally defined below.

1. **Message homomorphism.** Given the ciphertexts $\tau_1, \ldots, \tau_\ell$ on messages $m_1, \ldots, m_\ell$ and a low-depth circuit f, the following property holds:

$$\mathsf{Dec}\left(sk, \widetilde{f}(\tau_1, \ldots, \tau_\ell)\right) = f(m_1, \ldots, m_\ell) \cdot sk \ ,$$

   where $\widetilde{f}$ represents the homomorphic computation of f over the ciphertexts and can be performed in polynomial time.

2. **Distributed decryption.** For a ciphertext τ on a message m, the following properties hold:

   - **Linear distributed decryption.** Let $sk_0 \in \mathcal{R}_p$ be a uniform element such that $\mathsf{LSB}(sk_0) = 0$, and set $sk_1 = sk_0 + sk \mod p$, where $\mathsf{LSB}(sk_1) = 1$ for an even p. Then, with overwhelming probability, we have

$$\begin{aligned} \mathsf{Dec}(sk_i, \tau) &= -\mathsf{Dec}(sk_i, -\tau) \text{ for } i \in \{0, 1\}, \\ \mathsf{Dec}(sk_1, \tau) &= \mathsf{Dec}(sk_0, \tau) + \mathsf{Dec}(sk, \tau) \ . \end{aligned}$$

     The above two equations also imply $\mathsf{Dec}(sk, \tau) = -\mathsf{Dec}(sk, -\tau)$.

   - **Correlated-key distributed decryption.** Let $sk_0, sk_0' \in \mathcal{R}_p$ be two uniform elements such that $\mathsf{LSB}(sk_0) = \mathsf{LSB}(sk_0') = 0$. Let $sk_1 = sk_0 + sk \mod p$ and $sk_1' = sk_0' + sk \mod p$, where $\mathsf{LSB}(sk_1) = \mathsf{LSB}(sk_1') = 1$ given that p is even. There exists a polynomial-time algorithm $\widehat{\mathsf{Dec}}$ such that with overwhelming probability,

$$\begin{aligned} \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) &= -\widehat{\mathsf{Dec}}(sk_0, sk_0', -\tau) \\ \widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) - \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) &= i \cdot j \cdot \mathsf{Dec}(sk, \tau) \text{ , for any } i, j \in \{0, 1\} \ . \end{aligned}$$

We present a high-level idea to instantiate the SWHE scheme with the above properties in Section 2.5.

Using SWHE with distributed decryption for garbled circuits. Given any SWHE (with distributed decryption) ciphertexts τ_1, τ_2 and τ_3 encrypted with secret key sk, we introduce a function

Eval to achieve the functionality of the homomorphic random oracle H. This function Eval is specified as follows:

$$\mathsf{Eval}(X, Y) \stackrel{\text{def}}{=} \mathsf{Dec}(X, \tau_1) + \mathsf{Dec}(Y, \tau_2) + \widehat{\mathsf{Dec}}(X, Y, \tau_3) \ ,$$

where $X, Y \in \mathcal{R}_p$. Here, τ_1, τ_2 and τ_3 are inputs to Eval, omitted for simplicity. Based on distributed decryption properties, we show that for each $i, j \in \{0, 1\}$, for two uniform elements X, Y with $\mathsf{LSB}(X) = \mathsf{LSB}(Y) = 0$, with overwhelming probability, we have

$$\begin{aligned}
&\mathsf{Eval}(X + i \cdot sk, Y + j \cdot sk) - \mathsf{Eval}(X, Y) \\
&= \mathsf{Dec}(X + i \cdot sk, \tau_1) + \mathsf{Dec}(Y + j \cdot sk, \tau_2) + \widehat{\mathsf{Dec}}(X + i \cdot sk, Y + j \cdot sk, \tau_3) \\
&\quad - \Big(\mathsf{Dec}(X, \tau_1) + \mathsf{Dec}(Y, \tau_2) + \widehat{\mathsf{Dec}}(X, Y, \tau_3)\Big) \\
&= i \cdot \mathsf{Dec}(sk, \tau_1) + j \cdot \mathsf{Dec}(sk, \tau_2) + i \cdot j \cdot \mathsf{Dec}(sk, \tau_3) \ .
\end{aligned} \tag{1}$$

The SWHE with distributed decryption and the homomorphic RO H have slightly different properties but serve the same purpose. The evaluator uses the homomorphic RO H to ensure that for any $i, j \in \{0, 1\}$:

$$H(X + i \cdot \Delta, Y + j \cdot \Delta) = H(X, Y) + i \cdot H(\Delta, 0) + j \cdot H(0, \Delta) + i \cdot j \cdot H(\Delta, \Delta) \ ,$$

where Δ plays the same role as sk. This property is similar to the one that SWHE with distributed decryption satisfies, as seen in Equation (1); the only caveat is that we use $\mathsf{Dec}(sk, \tau_1)$, $\mathsf{Dec}(sk, \tau_2)$, and $\mathsf{Dec}(sk, \tau_3)$ to achieve the same goal of $H(\Delta, 0)$, $H(0, \Delta)$, and $H(\Delta, \Delta)$, respectively.

2.3 Constructing Garbled Circuits from HE Ciphertexts

Now we demonstrate how to use a SWHE scheme with distributed decryption to achieve the same functionality as the homomorphic random oracle H in the previous scheme, where communication requires three ciphertexts per gate. The garbled table of a gate g consists of

$$\begin{aligned}
\tau_1 &= \mathsf{Enc}(\Delta, (-1)^c \cdot (z_{1,0} - z_{0,0})), \\
\tau_2 &= \mathsf{Enc}(\Delta, (-1)^c \cdot (z_{0,1} - z_{0,0})), \\
\tau_3 &= \mathsf{Enc}(\Delta, (-1)^c \cdot (z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0})),
\end{aligned} \tag{2}$$

where Δ is the secret key of the SWHE scheme, $z_{i,j} = g(a \oplus i, b \oplus j)$ for each $i, j \in \{0, 1\}$, and $a = \mathsf{LSB}(A^0)$, $b = \mathsf{LSB}(B^0)$ and $c = \mathsf{LSB}(C^0)$ represent the three wire masks. We define the gate-evaluation function Eval as

$$\mathsf{Eval}(A, B, \tau_1, \tau_2, \tau_3) \stackrel{\text{def}}{=} \mathsf{Dec}(A, \tau_1) + \mathsf{Dec}(B, \tau_2) + \widehat{\mathsf{Dec}}(A, B, \tau_3) \ ,$$

omitting τ_1, τ_2, τ_3 when clear from context.

Garbling. Since A^0 and B^0 are known before garbling a gate, the garbler can compute $z_{i,j} = g(\mathsf{LSB}(A^0) \oplus i, \mathsf{LSB}(B^0) \oplus j)$ for each $i, j \in \{0, 1\}$. However, computing the wire mask c is more challenging. Recall that c is determined based on $c \oplus z_{0,0} = \mathsf{LSB}(C^{z_{0,0}})$ from (II), and we obtain the label based on GRR as $C^{z_{0,0}} = \mathsf{Eval}(A^a, B^b, \tau_1, \tau_2, \tau_3)$. However, since the garbled rows $\{\tau_i\}$

cannot be computed without knowledge of c, the garbler cannot yet perform the Eval operation. This leads to a deadlock: to obtain τ_i, we need to know c first; however, to determine c, we need to know τ_i. In classical GRR, this issue does not arise, as $C^{z_{0,0}} = H(A^a, B^b)$ can be computed without needing τ_1, τ_2, τ_3.

Fortunately, a method exists to compute c without the final garbled rows. Specifically, the garbler first computes

$$\begin{aligned}\widetilde{\tau}_1 &= \mathsf{Enc}(\Delta, (z_{1,0} - z_{0,0}))\ ,\\ \widetilde{\tau}_2 &= \mathsf{Enc}(\Delta, (z_{0,1} - z_{0,0}))\ ,\\ \widetilde{\tau}_3 &= \mathsf{Enc}(\Delta, (z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0}))\ ,\end{aligned}$$

then determines $c = \mathsf{LSB}\left(\mathsf{Eval}(A^a, B^b, \widetilde{\tau}_1, \widetilde{\tau}_2, \widetilde{\tau}_3)\right) \oplus z_{0,0}$ and sets $\tau_i = (-1)^c \cdot \widetilde{\tau}_i$. This approach works because

$$\begin{aligned}&\mathsf{LSB}(\mathsf{Eval}(A^a, B^b, \widetilde{\tau}_1, \widetilde{\tau}_2, \widetilde{\tau}_3))\\ &= \mathsf{LSB}(\mathsf{Dec}(A^a, \widetilde{\tau}_1) + \mathsf{Dec}(B^b, \widetilde{\tau}_2) + \widehat{\mathsf{Dec}}(A^a, B^b, \widetilde{\tau}_3))\\ &= \mathsf{LSB}\left((-1)^c \cdot \left(\mathsf{Dec}(A^a, \tau_1) + \mathsf{Dec}(B^b, \tau_2) + \widehat{\mathsf{Dec}}(A^a, B^b, \tau_3)\right)\right)\\ &= \mathsf{LSB}((-1)^c \cdot \mathsf{Eval}(A^a, B^b, \tau_1, \tau_2, \tau_3))\\ &= \mathsf{LSB}(\mathsf{Eval}(A^a, B^b, \tau_1, \tau_2, \tau_3)) = \mathsf{LSB}(C^{z_{0,0}})\ ,\end{aligned}$$

using the fact that for any $x \in \mathbb{Z}_p$, $-x = x \mod 2$ when p is even. This means that the garbler can compute c by evaluating on $\widetilde{\tau}_i$'s, which are computable with just A^a and B^b regardless of the actual value of c. It is important to emphasize that the garbler cannot directly adopt Equation (2) with c to compute τ_i via fresh encryption, since c may not align with the freshly encrypted τ_i. Finally, the garbler computes $C^{z_{0,0}} = \mathsf{Eval}(A^a, B^b, \tau_1, \tau_2, \tau_3)$ and obtains C^0 and C^c as follows: $C^0 = C^{z_{0,0}} - (-1)^c \cdot z_{0,0} \cdot \Delta$ (I) and $C^c = C^0 - \mathsf{LSB}(C^0) \cdot \Delta = C^0 - c \cdot \Delta$ (III). The pair of labels (C^0, C^1) may be used as the input labels for subsequent gates. Therefore, we must ensure that C^0 is uniform over $\mathcal{R}_p$ in order that the distributed decryption property still holds for subsequent gates. To do this, we use a random element R over $\mathcal{R}_p$ to randomize the pair of labels (C^0, C^1) as $(C^0 + R, C^1 + R)$. This approach works because Equation (1) still holds after the randomization (see Appendix A.1 for the detailed correctness analysis). Following prior work [BKS19], we employ a pseudo-random function (PRF) to generate the random element R for each gate, and thus the communication, to transmit the random elements for all gates from the garbler to the evaluator, can be compressed into only λ bits. For the sake of simplicity, we omit the PRF and randomization in the following, and refer the reader to Section 4.2 for details.

Gate-Evaluation. For each gate g, given the HE ciphertexts τ_1, τ_2, τ_3, as well as the input labels (A, B) with the masked bits $\alpha = \mathsf{LSB}(A)$ and $\beta = \mathsf{LSB}(B)$, the evaluator can compute the output label as $C = \mathsf{Eval}(A, B)$. The correctness follows from the analysis below. According to Equation (1), we have

$$\begin{aligned}&\mathsf{Eval}(A, B) = \mathsf{Eval}(A^a + \alpha \cdot \Delta, B^b + \beta \cdot \Delta)\\ &= \mathsf{Eval}(A^a, B^b) + \alpha \cdot \mathsf{Dec}(\Delta, \tau_1) + \beta \cdot \mathsf{Dec}(\Delta, \tau_2) + \alpha\beta \cdot \mathsf{Dec}(\Delta, \tau_3)\\ &= \mathsf{Eval}(A^a, B^b) + (-1)^c \cdot (z_{\alpha,\beta} - z_{0,0}) \cdot \Delta = C^0 + (-1)^c \cdot z_{\alpha,\beta} \cdot \Delta = C^{z_{\alpha,\beta}}\ ,\end{aligned}$$

where $C^{z_{0,0}} = \mathsf{Eval}(A^a, B^b)$ and 0-label $C^0 = C^{z_{0,0}} - (-1)^c \cdot z_{0,0} \cdot \Delta$ (I). We use the fact that $A = A^a + \mathsf{LSB}(A) \cdot \Delta = A^a + \alpha \cdot \Delta$ and $B = B^b + \beta \cdot \Delta$, which follow from II and III.

2.4 Reducing Communication of Generating HE Ciphertexts

So far, we have converted the garbled table into bit ciphertexts, where these bits are correlated. Now, we show that, for garbling a circuit of size N, the garbled circuits can be compressed to λ ciphertexts plus N bits. This is achieved by breaking down all ciphertexts into N ciphertexts, each encrypting an independent random bit, plus N bits of "stitching information". The former is then compressed using the homomorphic evaluation of a PRG. Specifically, the garbler sends the evaluator HE ciphertexts $\sigma_i \leftarrow \mathsf{Enc}(\Delta, s_i)$ for $i \in [\lambda]$, where $(s_1, \ldots, s_\lambda)$ constitutes a uniform PRG seed. Both parties then homomorphically compute the PRG over the ciphertexts $\sigma_1, \ldots, \sigma_\lambda$ to generate $\mathsf{Enc}(\Delta, r_w)$ for each wire w. Next, the garbler integrates these ciphertexts $\mathsf{Enc}(\Delta, r_w)$ for each wire w into the garbled circuits by sending one extra bit per wire. This process occurs in three steps for each gate, while maintaining the invariant that both parties hold the ciphertext of the wire mask for each input wire.

1. Given the invariant, the garbler has input labels A^0 and B^0 for each gate, and both parties have $\mathsf{Enc}(\Delta, a)$ and $\mathsf{Enc}(\Delta, b)$, where $a = \mathsf{LSB}(A^0)$ and $b = \mathsf{LSB}(B^0)$. Both parties perform homomorphic evaluations on ciphertexts $\mathsf{Enc}(\Delta, a)$, $\mathsf{Enc}(\Delta, b)$ and $\mathsf{Enc}(\Delta, r_c)$ to obtain:

$$\begin{aligned}
\widetilde{\tau}_1 &= \mathsf{Enc}(\Delta, (-1)^{r_c} \cdot (z_{1,0} - z_{0,0})) \,, \\
\widetilde{\tau}_2 &= \mathsf{Enc}(\Delta, (-1)^{r_c} \cdot (z_{0,1} - z_{0,0})) \,, \\
\widetilde{\tau}_3 &= \mathsf{Enc}(\Delta, (-1)^{r_c} \cdot (z_{0,0} + z_{1,1} - z_{0,1} - z_{1,0})) \,.
\end{aligned}$$

   Here, $z_{i,j} = g(a \oplus i, b \oplus j)$ for each $i, j \in \{0, 1\}$, r_c represents the pseudo-random bit associated with the output wire and $\mathsf{Enc}(\Delta, r_c)$ is its encryption. This process requires both parties to homomorphically compute a circuit with a multiplication depth of 2, where $(-1)^{r_c}$ is expressed as the linear function $1 - 2r_c$. Then, the garbler obtains $c = \mathsf{Eval}(A^a, B^b, \widetilde{\tau}_1, \widetilde{\tau}_2, \widetilde{\tau}_3) \oplus z_{0,0}$ similar to the scheme in the previous section.

2. To maintain the invariant and construct the garbled circuit, the garbler sends a single bit $v_c = r_c \oplus c$. Both parties can now obtain $\mathsf{Enc}(\Delta, c)$ using $\mathsf{Enc}(\Delta, r_c)$ and v_c.

3. Both parties can reconstruct the final garbled table as follows:

$$\begin{aligned}
\tau_1 &= (-1)^{v_c} \cdot \widetilde{\tau}_1 = \mathsf{Enc}(\Delta, (-1)^{v_c \oplus r_c} \cdot (z_{1,0} - z_{0,0})) \\
&= \mathsf{Enc}(\Delta, (-1)^{c} \cdot (z_{1,0} - z_{0,0})) \,,
\end{aligned}$$

   with similar computations for τ_2, τ_3.

In total, $d + 2$ levels of homomorphic multiplications are required, where d is the multiplication depth of the PRG. The noise does not accumulate between gates, as each gate uses fresh ciphertexts generated from the homomorphic evaluation of the PRG.

2.5 SWHE with Distributed Decryption from RLWE

Our final task is to construct an SWHE scheme that supports distributed decryption properties. Here, we provide an intuitive overview of our RLWE-based instantiation, inspired by the GSW HE scheme [GSW13], which is detailed in Section 5. Additionally, we demonstrate in Appendix C how the same properties can be achieved based on the NTRU assumption.

Overview of RLWE-based GSW Scheme. The ring learning with errors (RLWE) assumption states that it is hard to distinguish uniform samples in $\mathcal{R}_q^2$ from LWE samples of the form $(a_i, b_i = a_i \cdot sk + e_i)$, where $a_i \in \mathcal{R}_q$ is sampled uniformly, and $sk, e_i \in \mathcal{R}$ are drawn from an error distribution (particularly, a discrete Gaussian distribution), with $\|sk\|_\infty$ and $\|e_i\|_\infty$ being small with overwhelming probability. In the context of our garbling application, we set the first coefficient of the secret key sk as $sk[1] = 1$, with $\mathsf{LSB}(sk) = 1$ [1]. Additionally, similar to the GSW scheme [GSW13], we set the second coefficient of the secret key sk as $sk[2] = q/p$ to simplify decryption in the garbling scheme [2].

A GSW ciphertext $\tau \in \mathcal{R}_q^{2\times 2}$ for a message $m \in \mathbb{Z}_p$ is given by:

$$\tau = \begin{bmatrix} a_1 & b_1 = a_1 \cdot sk + e_1 - m \cdot sk \\ a_2 & b_2 = a_2 \cdot sk + e_2 + m \end{bmatrix},$$

where $a_1, a_2 \in \mathcal{R}_q$ are uniformly sampled and $e_1, e_2 \in \mathcal{R}$ are sampled from an error distribution. We observe that $\tau \cdot \boldsymbol{v} - \boldsymbol{e} = m \cdot \boldsymbol{v}$, where we define $\boldsymbol{v} = [-sk\ 1]^\mathsf{T}$ and $\boldsymbol{e} = [e_1\ e_2]^\mathsf{T}$. To decrypt, we compute

$$x = \left\lfloor \frac{b_1 - a_1 \cdot sk}{q/p} \right\rceil \approx \left\lfloor \frac{-m \cdot sk}{q/p} \right\rceil,$$

and output the second coefficient of the polynomial $-x$ as the message m, where $\lfloor u \rceil$ denotes the rounding function that maps each real-number coefficient of a polynomial-ring element u to the closest integer. Additionally, we can partially decrypt τ using only an inner product operation, ignoring the rounding operation. Specifically, we compute $\langle [a_1\ b_1], \boldsymbol{v} \rangle = (b_1 - a_1 \cdot sk)$ to directly obtain approximations of $(q/p) \cdot m$.

The GSW scheme allows homomorphic additions and multiplications because $\tau \cdot \boldsymbol{v} \approx m \cdot \boldsymbol{v}$. Suppose we have two ciphertexts, τ_1 and τ_2, encrypting messages m_1 and m_2, respectively. Then, $\tau_1 + \tau_2$ is a valid ciphertext for $m_1 + m_2$, because $(\tau_1 + \tau_2) \cdot \boldsymbol{v} \approx (m_1 + m_2) \cdot \boldsymbol{v}$. Similarly, $\tau_1 \cdot \tau_2$ is a valid ciphertext for $m_1 \cdot m_2$, if $\|\tau_1\|_\infty$ is sufficiently small to control noise growth. In this case, $\tau_1 \cdot \tau_2 \cdot \boldsymbol{v} \approx \tau_1 \cdot m_2 \cdot \boldsymbol{v} \approx m_1 \cdot m_2 \cdot \boldsymbol{v}$. To ensure $\|\tau_1\|_\infty$ remains small, the GSW scheme employs the bit decomposition technique, as detailed in Section 5.1. Here, we omit the bit decomposition technique for clarity.

We adopt two lemmas (i.e., the lifting and rounding lemmas) from [BKS19] to support distributed decryption. Informally, the lifting lemma shows that, if z_0 is uniform in $\mathcal{R}_p$ and $z_1 = z_0 + m \mod p$, then $z_1 = z_0 + m$ (without modulo p) holds with overwhelming probability, provided that $\|m\|_\infty / p$ is sufficiently small. The rounding lemma states that, if t_0 is uniform in $\mathcal{R}_q$ and $t_1 = t_0 + (q/p) \cdot m + e \in \mathcal{R}_q$, the equation $\lfloor (p/q) \cdot t_1 \rceil = \lfloor (p/q) \cdot t_0 \rceil + m$ holds with overwhelming probability, provided that $p \cdot \|e\|_\infty / q$ is sufficiently small.

The GSW ciphertexts can directly support linear distributed decryption when both $\|m\|_\infty / p$ and $p \cdot \|e\|_\infty / q$ are relatively small. Let $sk_0 \in \mathcal{R}_p$ be sampled uniformly such that $\mathsf{LSB}(sk_0) = 0$ and $sk_1 = sk_0 + sk \mod p$ with $\mathsf{LSB}(sk_1) = 1$. According to the lifting lemma, $sk_1 = sk_0 + sk$ (i.e., the operation of "mod p" can be removed) with overwhelming probability. Thus, we have:

$$b_1 - a_1 \cdot sk_1 = b_1 - a_1 \cdot (sk_0 + sk) = -a_1 \cdot sk_0 + (b_1 - a_1 \cdot sk)\ .$$

[1] Our NTRU-based scheme can relax the restriction on $sk[1] = 1$ to $sk[1] \mod 2 = 1$.

[2] For simplicity, we w.l.o.g. assume $p \mid q$, as in prior work [BKS19]. When the bit decomposition technique is applied, the restriction on $sk[2]$ can be eliminated [GSW13].

Then, based on the rounding lemma, we have, with overwhelming probability:

$$\underbrace{\left\lfloor \frac{b_1 - a_1 \cdot sk_1}{q/p} \right\rceil}_{\mathsf{Dec}(sk_1,\tau)} = \underbrace{\left\lfloor \frac{-a_1 \cdot sk_0}{q/p} \right\rceil}_{\mathsf{Dec}(sk_0,\tau)} + \underbrace{\left\lfloor \frac{b_1 - a_1 \cdot sk}{q/p} \right\rceil}_{\mathsf{Dec}(sk,\tau)} .$$

Extended GSW. Before discussing how to support correlated-key distributed decryption, we introduce an extended GSW (eGSW) scheme, which incorporates an additional secret key $\widehat{sk} \in \mathcal{R}$ and a ciphertext $\widehat{\tau} \in \mathcal{R}_q^4$. The eGSW encryption algorithm, denoted as $\widehat{\mathsf{Enc}}(sk, \widehat{sk}, m)$, uses both sk and $\widehat{sk}$ to encrypt a message $m \in \mathbb{Z}_p$, where $sk, \widehat{sk}$ follow an error distribution. This process generates an eGSW ciphertext:

$$\widehat{\tau} = \left(a,\ b^1 = a \cdot sk + e^1,\ b^2 = a \cdot \widehat{sk} + e^2,\ b^3 = b^1 \cdot \widehat{sk} + e^3 + (q/p) \cdot m \cdot \widehat{sk}\right) ,$$

where $a \in \mathcal{R}_q$ is sampled uniformly and e^1, e^2, e^3 follow an error distribution. It can be verified that eGSW ciphertexts are additively homomorphic (see Section 5.2 for details). Below, we introduce two additional properties of the eGSW scheme.

- It is cheap to convert a GSW ciphertext to an eGSW ciphertext. This is achieved by homomorphically evaluating the GSW partial decryption (i.e., the inner product only) on the ciphertext

  $$\left(\boldsymbol{a},\ \boldsymbol{b}^1 = \boldsymbol{a} \cdot sk + \boldsymbol{e}^1,\ \boldsymbol{b}^2 = \boldsymbol{a} \cdot \widehat{sk} + \boldsymbol{e}^2,\ \boldsymbol{b}^3 = \boldsymbol{b}^1 \cdot \widehat{sk} + \boldsymbol{e}^3 + \boldsymbol{v} \cdot \widehat{sk} \right) ,$$

  which encrypts the GSW key-related vector $\boldsymbol{v} = [-sk\ 1]^\mathsf{T}$. Here, $\boldsymbol{a}$ is uniform in $\mathcal{R}_q^2$ and $\boldsymbol{e}^1, \boldsymbol{e}^2, \boldsymbol{e}^3 \in \mathcal{R}^2$ are sampled from an error distribution. This ciphertext is very close to the eGSW ciphertext $\widehat{\mathsf{Enc}}(sk, \widehat{sk}, sk)$, except that $\boldsymbol{b}^3$ lacks the factor of (q/p). We denote the above process as $[\tau]_{\mathsf{gsw}\to\mathsf{egsw}}$ for a GSW ciphertext $\tau \in \mathcal{R}_q^{2\times 2}$. It first finds $\boldsymbol{s} \in \mathcal{R}_q^2$ such that $\boldsymbol{s}^\mathsf{T} \cdot \tau \cdot \boldsymbol{v} \approx (q/p) \cdot m$, and then outputs an eGSW ciphertext shown as follows:

  $$\left(a = \langle \boldsymbol{u}, \boldsymbol{a} \rangle ,\ b^1 = \langle \boldsymbol{u}, \boldsymbol{b}^1 \rangle ,\ b^2 = \langle \boldsymbol{u}, \boldsymbol{b}^2 \rangle ,\ b^3 = \langle \boldsymbol{u}, \boldsymbol{b}^3 \rangle \approx b^1 \cdot \widehat{sk} + (q/p) \cdot m \cdot \widehat{sk}\right) .$$

  where $\boldsymbol{u} = \boldsymbol{s}^\mathsf{T} \cdot \tau$. It is easy to observe $-[\tau]_{\mathsf{gsw}\to\mathsf{egsw}} = [-\tau]_{\mathsf{gsw}\to\mathsf{egsw}}$.

- It is also easy to non-interactively convert from an additive secret sharing of sk to that of $\widehat{sk}$, or vice versa. We use $[\cdot]_{sk\to\widehat{sk}}$ to denote the conversion from the sharing of sk into the sharing of $\widehat{sk}$, and $[\cdot]_{\widehat{sk}\to sk}$ to denote the opposite conversion. Specifically, given two additive secret sharings $(sk_0, sk_1), (\widehat{sk}_0, \widehat{sk}_1) \in \mathcal{R}_p^2$ such that $sk_0, \widehat{sk}_0 \in \mathcal{R}_p$ be two uniform elements with $\mathsf{LSB}(sk_0) = \mathsf{LSB}(\widehat{sk}_0) = 1$, $sk_1 = sk_0 + sk$ and $\widehat{sk}_1 = \widehat{sk}_0 + \widehat{sk}$ over $\mathcal{R}_p$, these conversions ensure

  $$\left[sk_1\right]_{sk\to\widehat{sk}} = \left[sk_0\right]_{sk\to\widehat{sk}} + \widehat{sk} . \tag{3}$$

  $$\left[\widehat{sk}_1\right]_{\widehat{sk}\to sk} = \left[\widehat{sk}_0\right]_{\widehat{sk}\to sk} + sk . \tag{4}$$

  Both conversions can be realized following a similar idea used in linear distributed decryption. For example, given a ciphertext $(a, b = a \cdot sk + e + (q/p) \cdot \widehat{sk})$, according to the lifting lemma, we

have $sk_1 = sk_0 + sk$ and $sk_1[1] = sk_0[1] + 1$ (without modulo p) with overwhelming probability. This results in

$$\begin{aligned} sk_1[1] \cdot b - a \cdot sk_1 &= (sk_0[1] + 1) \cdot b - a \cdot (sk_0 + sk) \\ &= (sk_0[1] \cdot b - a \cdot sk_0) + (b - a \cdot sk) \ . \end{aligned}$$

From the rounding lemma, with overwhelming probability, this conversion associated with equation (3) can be done using the following observation:

$$\underbrace{\left\lfloor \frac{sk_1[1] \cdot b - a \cdot sk_1}{q/p} \right\rceil}_{[sk_1]_{sk \to \widehat{sk}}} = \underbrace{\left\lfloor \frac{sk_0[1] \cdot b - a \cdot sk_0}{q/p} \right\rceil}_{[sk_0]_{sk \to \widehat{sk}}} + \underbrace{\left\lfloor \frac{b - a \cdot sk}{q/p} \right\rceil}_{\widehat{sk}} \ .$$

The conversion related to the equation (4) can be performed similarly. Due to $\left\lfloor \frac{-x}{q/p} \right\rceil = - \left\lfloor \frac{x}{q/p} \right\rceil$, it is easy to see that for any $y \in \mathcal{R}_p$, $-[y]_{sk \to \widehat{sk}} = [-y]_{sk \to \widehat{sk}}$.

Supporting correlated-key distributed decryption. We now present the construction of the $\widehat{\mathsf{Dec}}$ algorithm. Let $sk_0, sk_0' \in \mathcal{R}_p$ be sampled uniformly such that $\mathsf{LSB}(sk_0) = \mathsf{LSB}(sk_0') = 0$, $sk_1 = sk_0 + sk \mod p$ and $sk_1' = sk_0' + sk \mod p$ with $\mathsf{LSB}(sk_1) = \mathsf{LSB}(sk_1') = 1$. Given a ciphertext $\tau \leftarrow \mathsf{Enc}(sk, m)$, $\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau)$ is defined as follows:

1. Switch the second additive secret sharing, by computing $\widehat{sk}_j = \left[sk_j'\right]_{sk \to \widehat{sk}}$.
2. Convert τ to an eGSW ciphertext by computing $(a, b^1, b^2, b^3) = [\tau]_{\mathsf{gsw} \to \mathsf{egsw}}$.
3. Set $i = \mathsf{LSB}(sk_i)$ and $j = \mathsf{LSB}(sk_j')$, and compute

$$x = sk_i \cdot \widehat{sk}_j \cdot a - i \cdot \widehat{sk}_j \cdot b^1 - j \cdot sk_i \cdot b^2 + i \cdot j \cdot b^3 \mod q.$$

4. Set $y = \lfloor (p/q) \cdot x \rceil$ and switch it back to an additive secret sharing of sk by computing $[y]_{\widehat{sk} \to sk}$.

It is easy to observe that $\widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) = -\widehat{\mathsf{Dec}}(sk_0, sk_0', -\tau)$ holds with overwhelming probability, since (1) for any GSW ciphertext τ, $-[\tau]_{\mathsf{gsw} \to \mathsf{egsw}} = [-\tau]_{\mathsf{gsw} \to \mathsf{egsw}}$; (2) for any $x \in \mathcal{R}_q$ and $y \in \mathcal{R}_p$, $\lfloor (p/q) \cdot (-x) \rceil = - \lfloor (p/q) \cdot x \rceil$ and $-[y]_{\widehat{sk} \to sk} = [-y]_{\widehat{sk} \to sk}$. Then, with overwhelming probability, we have that $\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) - \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) = i \cdot j \cdot \mathsf{Dec}(sk, \tau)$ for any $i, j \in \{0, 1\}$, whose analysis is postponed to Theorem 2 of Section 5.2.

3 Preliminaries

In this section, we present the notation and definitions used in our garbling scheme and instantiations of SWHE with distributed decryption.

Notation. We will use λ to denote the computational security parameter. Given $a, b \in \mathbb{N}$ with $a \leq b$, we write $[a, b] = \{a, \ldots, b\}$ and $[n] = \{1, \ldots, n\}$. We use $x \xleftarrow{\$} \mathcal{S}$ (resp., $x \leftarrow \mathcal{D}$) to denote sampling x from a set $\mathcal{S}$ uniformly at random (resp., according to a distribution $\mathcal{D}$). For two distributions X and Y, we denote by $X \overset{c}{\approx} Y$ that X is computationally indistinguishable

from Y. We use bold lower-case letters like $\boldsymbol{a}$ to denote column vectors. For a vector $\boldsymbol{a}$, we use $\boldsymbol{a}[i]$ to denote the i-th component of $\boldsymbol{a}$, where $\boldsymbol{a}[1]$ is the first component. For a vector $\boldsymbol{v}$ of length n, the Euclidean norm is $\|\boldsymbol{v}\|_2 \stackrel{\text{def}}{=} \sqrt{\sum_{i\in[n]}(\boldsymbol{v}[i])^2}$ and the infinity norm is denoted by $\|\boldsymbol{v}\|_\infty \stackrel{\text{def}}{=} \max_{i\in[n]} |\boldsymbol{v}[i]|$. For two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, we use $\boldsymbol{x} \approx \boldsymbol{y}$ to denote that $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty$ is relatively small. Let $\mathcal{R} \stackrel{\text{def}}{=} \mathbb{Z}[X]/(X^n + 1)$ be a polynomial ring with integer coefficients modulo a polynomial $X^n + 1$. We define $\mathcal{R}_q \stackrel{\text{def}}{=} \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(X^n + 1)$ for some integer q. Viewing a polynomial in $\mathcal{R}$ as a vector in $\mathbb{Z}_q^n$ is natural, and vice versa. Thus, we use $a[i]$ to denote the i-th coefficient of a polynomial $a \in \mathcal{R}$. For a polynomial $x \in \mathcal{R}$, we define $\mathsf{LSB}(x) = x[1] \mod 2$. We use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function such that $\mathsf{negl}(\lambda) = o(\lambda^{-c})$ for every constant c, and $\mathsf{poly}(\cdot)$ to denote a polynomial function with $\mathsf{poly}(\lambda) = O(\lambda^c)$ for some constant c. We denote the rounding function by $\lfloor\cdot\rceil : \mathbb{R} \to \mathbb{Z}$, which maps $x \in \mathbb{R}$ to the closest integer $y \in \mathbb{Z}$.

Boolean circuits. Following the notation in [BHR12], we define a Boolean circuit that is a 5-tuple $f = (n, m, |f|, \mathsf{Gateinputs}, G)$, where n is the number of inputs, m is the number of outputs, and $|f|$ is the number of gates. We use $N = n + |f|$ to denote the number of wires. We write $\mathsf{Inputs}(f) = \{1, \ldots, n\}$, $\mathsf{GateIndex}(f) = \{n + 1, \ldots, N\}$, $\mathsf{WireIndex}(f) = \{1, \ldots, N\}$ and $\mathsf{Ouputs}(f) = \{N - m + 1, \ldots, N\}$. For a given circuit f and a gate index i, the function $\{\alpha, \beta\} \leftarrow \mathsf{Gateinputs}(f, i)$ outputs the indices of the two incoming wires for the i-th gate within the circuit f. The function $G : \mathsf{GateIndex}(f) \times \{0, 1\}^2 \to \{0, 1\}$ determines the functionality of each gate by mapping a gate index and two input bits to an output bit. We employ v_i to represent the real bit associated with the i-th wire.

3.1 Low-Depth Pseudo-Random Generator

A pseudo-random generator (PRG) expands a short seed into longer randomness. This PRG should be implemented with a low-depth circuit to support efficient homomorphic evaluation over SWHE ciphertexts. Specifically, $\mathsf{PRG} : \{0, 1\}^\lambda \to \{0, 1\}^m$ uses a λ-bit secret seed to produce $m > \lambda$ bits, matching our circuit size. We require that PRG satisfies two conditions: (1) the standard pseudo-randomness, i.e., the PRG output should be computationally indistinguishable from a random string; (2) the PRG needs to be implemented in a circuit with a depth of $O(\log(\lambda))$. We also consider weak pseudo-random functions (weak PRFs), since for any weak PRF, $\mathsf{wPRF}_x(\cdot)$, combined with a random oracle, $H(\cdot)$, allows $\mathsf{PRG}(x) = (\mathsf{wPRF}_x(H(1)), \cdots, \mathsf{wPRF}_x(H(m)))$ is a PRG.

- **Low-depth PRG using AES.** AES is widely recognized as a fixed-depth PRG. That is, for a secret seed $s \in \{0, 1\}^\lambda$, $\mathsf{PRG}(s)$ can be computed as $(\mathsf{AES}_s(1), \cdots, \mathsf{AES}_s(m))$, using s as the AES secret key. AES can be represented as a Boolean circuit with 2-fan-in gates and a fixed depth, such as 40, for a security parameter of $\lambda = 128$ [BLO16]. Note that the multiplication depth of 40 can be easily supported by SWHE schemes, e.g, our instantiations (shown in Section 5 and Appendix C).

- **PRG in NC^1 from LWE.** Banerjee, Peikert, and Rosen [BPR12] introduced the Learning with Rounding (LWR) problem and proposed a reduction from LWR to LWE. The LWR problem facilitates the construction of a simple and practical PRG, representable by circuits in NC^1 [BPR12].

- **PRG in NC^0.** Constructing PRG in NC^0 has been studied over two decades [CM01, AIK04, IKOS08, AIK08, ABW10, Gol11, App12, AK19]. For example, Applebaum [App12] proposed a PRG in NC^0 from a variant of Goldreich's one-way function [Gol11]. Furthermore, Applebaum, Ishai, and Kushilevitz [AIK08] constructed a PRG in NC^0 from the hardness of decoding "sparsely generated" linear code [Ale03].

- **Constant-depth weak PRF from LPN.** The recent works [BIP+18, DGH+21, APRR24] propose weak PRF constructions based on a variant of the Learning Parity with Noise (LPN) problem and show that these constructions can be implemented using constant-depth circuits with unbounded fan-in gates.

3.2 Definition of Garbling Schemes

We recall the definition of a garbling scheme from [RR21], which modifies the original definition from [BHR12] with two changes. First, the correctness is adjusted to allow a negligible probability of failure. Second, the authenticity property is enhanced by allowing the adversary to obtain an extra decoding information d. We defer the definition of authenticity to Appendix B.1. Following the prior work [BHR12], we define a side-information function $\Phi(\cdot)$ that deterministically maps a circuit f into a string $\Phi(f)$ indicating which side information of f is revealed. In this paper, we focus on the classical setting, i.e., $\Phi_{\mathsf{circ}}(f) = f$, which reveals the entire circuit f. The garbler executes the Garble and Encode algorithms, and the evaluator executes the Eval and Decode algorithms. These algorithms are defined as below.

Definition 1 (Syntax)**.** *A garbling scheme consists of the following polynomial-time algorithms:*

- $(F, e, d) \leftarrow \mathsf{Garble}(1^\lambda, f)$*: This probabilistic algorithm takes as input a security parameter* λ *and a circuit* f*, and outputs a garbled circuit* F*, an encoding information* e *and a decoding information* d*.*

- $X \leftarrow \mathsf{Encode}(e, x)$*: This deterministic algorithm takes as input the encoding information* e *and an input* x*, and outputs a garbled input* X*.*

- $Y \leftarrow \mathsf{Eval}(F, X)$*: This deterministic algorithm takes as input* F *and* X*, and outputs a garbled output* Y*.*

- $y \leftarrow \mathsf{Decode}(d, Y)$*: This deterministic algorithm takes as input* d *and* Y*, and outputs a circuit output* y*.*

Correctness: For any circuit f and input x, for $(F, e, d) \leftarrow \mathsf{Garble}(1^\lambda, f)$, the following equality holds except with probability $\mathsf{negl}(\lambda)$,

$$\mathsf{Decode}\Big(d, \mathsf{Eval}\big(F, \mathsf{Encode}(e, x)\big)\Big) = f(x)\,.$$

Privacy. The privacy property of garbling schemes is to guarantee that the input is private against the evaluator, even if it obtains the transcript (F, X, d). We have the following definition for privacy.

Real World $\mathsf{Exp}^{\mathsf{prv},\Phi}_{real,\mathcal{A}}(1^\lambda, f, x)$

1. $(F, e, d) \leftarrow \mathsf{Garble}(1^\lambda, f)$;
2. $X \leftarrow \mathsf{Encode}(e, x)$;
3. $b \leftarrow \mathcal{A}(F, X, d)$.

Ideal World $\mathsf{Exp}^{\mathsf{prv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}(1^\lambda, f, x)$

1. $(F, X, d) \leftarrow \mathcal{S}(1^\lambda, \Phi(f), f(x))$
2. $b \leftarrow \mathcal{A}(F, X, d)$.

Figure 1: Experiments for privacy with respect to a side-information function Φ.

Real World $\mathsf{Exp}^{\mathsf{obv},\Phi}_{real,\mathcal{A}}(1^\lambda, f, x)$

1. $(F, e, d) \leftarrow \mathsf{Garble}(1^\lambda, f)$;
2. $X \leftarrow \mathsf{Encode}(e, x)$;
3. $b \leftarrow \mathcal{A}(F, X)$.

Ideal World $\mathsf{Exp}^{\mathsf{obv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}(1^\lambda, f, x)$

1. $(F, X) \leftarrow \mathcal{S}(1^\lambda, \Phi(f))$
2. $b \leftarrow \mathcal{A}(F, X)$.

Figure 2: Experiments for obliviousness w.r.t. a side-information function Φ.

Definition 2 (Privacy)**.** *There exists a probabilistic polynomial time (PPT) simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$, for any function f and input x, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{prv},\Phi}_{real,\mathcal{A}}(1^\lambda, f, x) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{prv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}(1^\lambda, f, x) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{prv},\Phi}_{real,\mathcal{A}}$ *and* $\mathsf{Exp}^{\mathsf{prv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}$ *are defined in Figure 1.*

Obliviousness. The obliviousness also guarantees the privacy of inputs against the evaluator, even if it obtains the information (F, X). Different from privacy, obliviousness does not let the adversary obtain the decoding information d, and thus it cannot learn the output $f(x)$. Obliviousness is useful when the garbling scheme is used as a part of a large system where the evaluator should not obtain the output.

Definition 3 (Obliviousness)**.** *There exists a PPT simulator $\mathcal{S}$ such that, for any PPT adversary $\mathcal{A}$, for any function f and input x, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{obv},\Phi}_{real,\mathcal{A}}(1^\lambda, f, x) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{obv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}(1^\lambda, f, x) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{obv},\Phi}_{real,\mathcal{A}}$ *and* $\mathsf{Exp}^{\mathsf{obv},\Phi}_{ideal,\mathcal{A},\mathcal{S}}$ *are defined in Figure 2.*

3.3 Basic Definitions and Lemmas for Lattice-based Cryptography

Below, we recall the basic definitions and lemmas used in our SWHE instantiations with distributed decryption.

Definition 4 (Discrete Gaussian Distribution)**.** *Let D_σ^m be a discrete Gaussian distribution over $\mathbb{Z}^m$ centered at $\mathbf{0}$ with standard deviation σ, i.e., $\forall \boldsymbol{x} \in \mathbb{Z}^m$, $D_\sigma^m(\boldsymbol{x}) = \rho_\sigma^m(\boldsymbol{x})/\rho_\sigma^m(\mathbb{Z}^m)$, where $\rho_\sigma^m(\boldsymbol{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m e^{-\frac{\sum_{i\in[m]}(\boldsymbol{x}[i])^2}{2\sigma^2}}$ is the corresponding continuous normal distribution over $\mathbb{R}^m$ and $\rho_\sigma^m(\mathbb{Z}^m) = \sum_{\boldsymbol{z}\in\mathbb{Z}^m} \rho_\sigma^m(\boldsymbol{z})$.*

Modular rounding. We utilize the notation $\lfloor\cdot\rceil$ to denote rounding real numbers to the nearest integers. For integers p and q where $2 \le p \le q$, we define the "modular rounding" function as follows:

$$\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p \text{ that maps } x \to \lfloor (p/q)\cdot x \rceil \ .$$

This function can be extended element-wise to vectors and matrices over $\mathbb{Z}_q$. A probability distribution $\mathcal{S}$ over $\mathcal{R}$ is said to be B-bounded if it satisfies that $\Pr_{\boldsymbol{x}\leftarrow\mathcal{S}}[\|\boldsymbol{x}\|_\infty > \mathsf{B}]$ is negligible in λ. We denote by $\mathcal{D}_\mathsf{B}$ a distribution such that if $\boldsymbol{e} \leftarrow \mathcal{D}$, then each coefficient of $\boldsymbol{e} \in \mathcal{R}$ is independently bounded by B. In our SWHE instantiations, we use a discrete Gaussian distribution to define $\mathcal{D}_\mathsf{B}$, while other error distributions may also be applicable.

Lemma 1 (Lifting Lemma [BKS19])**.** *Let $p \in \mathbb{N}$ be a modulus with $p \ge n^{\omega(1)}$and let $z_0 \in \mathcal{R}$ be a uniformly random ring element such that all coefficients are in $\mathbb{Z}_p$. For any $m \in \mathcal{R}$, let $z_1 = z_0 + m \mod p$, and we have*

$$\Pr\left[z_1 = z_0 + m\right] \ge 1 - n\cdot\left(\|m\|_\infty + 1\right)/p \ .$$

Lemma 2 (Rounding Lemma [BKS19])**.** *Let $p, q \in \mathbb{N}$ be modulus values with $q/p \ge n^{\omega(1)}$ and $p \mid q$. Let $t_0 \in \mathcal{R}_q$ be a uniformly random ring element, and $t_1 = t_0 + (q/p)\cdot m + e$ over $\mathcal{R}_q$ for some $m \in \mathcal{R}_p$ and $e \in \mathcal{R}$. Then, we have*

$$\Pr\left[\lfloor t_1\rceil_p = \lfloor t_0\rceil_p + m\right] \ge 1 - n\cdot\left(\|e\|_\infty + 1\right)\cdot p/q \ .$$

RLWE. The ring learning with errors (RLWE) problem was introduced by Lyubaskevsky, Peikert, and Regev [LPR10], and is defined as below. To assess its hardness, the Blockwise Korkine-Zolotarev (BKZ) algorithm [SE94] is always used. Specifically, we use the state-of-the-art BKZ algorithm [MV10] to estimate the asymptotic hardness of the RLWE problem as stated in Claim 1.

Definition 1 (Decisional RLWE [LPR10])**.** *Let $\mathcal{R}_q = \mathbb{Z}_q[X]/\left(X^n + 1\right)$. For dimension $n \in \mathbb{N}$, number of samples $m \in \mathbb{N}$ and $q \in \mathbb{N}$, the $\mathsf{RLWE}(n, m, \mathsf{B}, q)$ problem is to distinguish between the following two distributions:*

$$\left\{\left(a_i \xleftarrow{\$} \mathcal{R}_q, b_i \overset{def}{=} a_i\cdot s + e_i\right)\right\}_{i\in[m]} \text{ and } \left\{\left(a_i \xleftarrow{\$} \mathcal{R}_q, u_i \xleftarrow{\$} \mathcal{R}_q\right)\right\}_{i\in[m]} ,$$

where $s, e_i \leftarrow \mathcal{D}_\mathsf{B}$.

Claim 1 ([HKM18, MV10])**.** *The BKZ algorithm solves the RLWE problem with a modulus $q = n^Q$ and an error norm $\|\boldsymbol{e}_i\|_2 = n^\mathcal{S}$ in time $2^{O(\beta)}$, where the block size $\beta = O\left(\frac{Q}{(Q-\mathcal{S})^2}\cdot n\right)$ and n is the dimension.*

4 Garbling Scheme with Communication of 1-Bit per Gate

Firstly, we present the formal definition of somewhat homomorphic encryption (SWHE) schemes with distributed decryption. Then, we give the details of our garbling scheme that communicates only 1 bit per gate, using SWHE with distributed decryption as a building block. Finally, we demonstrate that our garbling scheme ensures both privacy and obliviousness. The extension of this scheme to support authenticity is detailed in Appendix B.

4.1 Somewhat Homomorphic Encryption with Distributed Decryption

Inspired by the notion of encryption schemes with nearly linear decryption [BKS19], we propose a private-key somewhat homomorphic encryption scheme with distributed decryption. Without loss of generality, we adopt two polynomial rings, $\mathcal{R}$ and $\mathcal{R}_p$ (with even p), in the following definition. Note that this definition can be easily extended to other finite fields and rings.

Definition 5 (SWHE with Distributed Decryption)**.** *A somewhat homomorphic encryption scheme with distributed decryption in the private-key setting, denoted by* $\mathsf{SWHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, consists of three polynomial-time algorithms and satisfies the message homomorphism and distributed decryption properties. Let* $\mathcal{K}$ *be the space of secret keys generated by* Gen*,* $\mathcal{R}_p$ *be the space of key shares,* $\mathcal{M} \subseteq \mathcal{R}_p$ *be the message space, and* $\mathcal{C}$ *be the ciphertext space.*

- ***Key Generation:*** $(\mathsf{params}, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$*. The key generation algorithm takes as input a security parameter* λ *and a maximum multiplication depth* L*. It outputs a set of parameters* params*, along with a secret key* $sk \in \mathcal{K}$ *such that* $\mathsf{LSB}(sk) = 1$*. Here,* params *is an implicit input to the following algorithms, omitted for simplicity.*

- ***Encryption:*** $\tau \leftarrow \mathsf{Enc}(sk, m)$*. The encryption algorithm takes as input a secret key* $sk \in \mathcal{K}$ *and a message* $m \in \mathcal{M}$*, and outputs a ciphertext* $\tau \in \mathcal{C}$*.*

- ***Decryption:*** $m \cdot sk \leftarrow \mathsf{Dec}(sk, \tau)$*. The decryption algorithm takes as input a secret key* sk *and a ciphertext* τ*, and outputs* $m \cdot sk$ *over* $\mathcal{R}_p$*. It is w.l.o.g. assumed that* m *can be recovered from* $m \cdot sk$ *over* $\mathcal{R}_p$ *with secret key* sk*.*

The SWHE *scheme needs to satisfy the following properties.*

1. ***Message homomorphism.*** *For any* $(\mathsf{params}, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$*, any message* $m_i \in \mathcal{M}$*, and its corresponding ciphertext* $\tau_i = \mathsf{Enc}(sk, m_i)$*, for each* $i \in [\ell]$*, where integer* $\ell \geq 1$*, the following holds with probability* $1 - \mathsf{negl}(\lambda)$*, for any polynomial-sized circuit* f *with a low depth* L *(e.g.,* f *is in* NC^1*) :*

$$\mathsf{Dec}\big(sk, \widetilde{f}(\tau_1, \ldots, \tau_\ell)\big) = f(m_1, \ldots, m_\ell) \cdot sk \ ,$$

   where $\widetilde{f}$ *denotes the homomorphic evaluation of circuit* f *on the ciphertexts, which can be performed in time* $\mathsf{poly}(\lambda)$*.*

2. ***Distributed decryption.*** *For each* $(\mathsf{params}, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ *and ciphertext* $\tau = \mathsf{Enc}(sk, m)$ *on any message* $m \in \mathcal{M}$*, the following properties hold:*

   - ***Linear distributed decryption.*** *Let* sk_0 *be an element sampled uniformly from* $\mathcal{R}_p$ *such that* $\mathsf{LSB}(sk_0) = 0$*. Let* $sk_1 = sk_0 + sk$ *over* $\mathcal{R}_p$*. Then, the following equations hold with*

probability $1 - \mathsf{negl}(\lambda)$,

$$\mathsf{Dec}(sk_i, \tau) = -\mathsf{Dec}(sk_i, -\tau) \textit{ for } i \in \{0, 1\}$$
$$\mathsf{Dec}(sk_1, \tau) = \mathsf{Dec}(sk_0, \tau) + \mathsf{Dec}(sk, \tau) \ .$$

The first equation is not equivalent to $\mathsf{Dec}(sk, \tau) = -\mathsf{Dec}(sk, -\tau)$*, since neither* sk_0 *nor* sk_1 *is a secret key. However, the two equations described as above imply* $\mathsf{Dec}(sk, \tau) = -\mathsf{Dec}(sk, -\tau)$.

- ***Correlated-key distributed decryption.*** *Let* sk_0, sk_0' *be two elements sampled uniformly from* $\mathcal{R}_p$ *such that* $\mathsf{LSB}(sk_0) = 0$ *and* $\mathsf{LSB}(sk_0') = 0$. *Let* $sk_1 = sk_0 + sk$ *over* $\mathcal{R}_p$ *and* $sk_1' = sk_0' + sk$ *over* $\mathcal{R}_p$. *There exists a polynomial-time algorithm* $\widehat{\mathsf{Dec}}$ *such that the following equations hold with probability* $1 - \mathsf{negl}(\lambda)$,

$$\widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) = -\widehat{\mathsf{Dec}}(sk_0, sk_0', -\tau)$$
$$\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) - \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) = i \cdot j \cdot \mathsf{Dec}(sk, \tau) \textit{ for each } i, j \in \{0, 1\} \ .$$

The message homomorphism and distributed decryption properties, as described above, have implied the correctness of the decryption algorithm. Below, we focus on defining the security.

CPA security for SWHE with distributed decryption. We require that the SWHE scheme satisfies the indistinguishability under chosen-plaintext attacks (i.e., CPA security in short). Specifically, a CPA experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa}}(1^\lambda, b)$ interacts with a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ as follows:

1. Run $(\mathsf{params}, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ and send params to $\mathcal{A}$.
2. $\mathcal{A}$ has access to an encryption oracle $\mathsf{Enc}(sk, \cdot)$ and outputs two messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$.
3. Run $\tau^* \leftarrow \mathsf{Enc}(sk, m_b)$, and send it to $\mathcal{A}$.
4. $\mathcal{A}$ continues to query $\mathsf{Enc}(sk, \cdot)$, and then outputs a bit b'.

We say that an SWHE scheme is CPA secure, if

$$\left|\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda) \, .$$

In Section 5 and Appendix C, we present two efficient instantiations of the SWHE scheme with distributed decryption from the RLWE and NTRU assumptions, respectively, both with KDM security. The KDM security is crucial for instantiating the $\widehat{\mathsf{Dec}}$ algorithm with key-dependent ciphertexts.

4.2 Our Garbling Scheme from SWHE with Distributed Decryption

Based on a somewhat homomorphic encryption (SWHE) scheme with distributed decryption, we construct a communication-efficient garbling scheme, as shown in Figure 3. We use $\Delta \in \mathcal{K}$, where $\mathsf{LSB}(\Delta) = 1$, as the secret key for this SWHE scheme. Both parties homomorphically compute a low-depth PRG, as described in Section 3.1, on the ciphertexts of a random seed to generate N

$\mathsf{Garble}(1^\lambda, f)$: The garbler executes the garble algorithm as follows:

1. Run $(\mathsf{params}, \Delta) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ with $\mathsf{LSB}(\Delta) = 1$, where Δ is the secret key. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ as a PRF key.
2. For each $i \in [\lambda]$, sample $s_i \xleftarrow{\$} \{0,1\}$, run $\sigma_i \leftarrow \mathsf{Enc}(\Delta, s_i)$, and then set $s = (s_1, \cdots, s_\lambda)$ and $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$.
3. Compute $(\widehat{r}_1, \cdots, \widehat{r}_N) \leftarrow \mathsf{PRG}(s)$ with $\widehat{r}_i \in \{0,1\}$, and homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N)$ with $\widehat{\tau}_i = \mathsf{Enc}(\Delta, \widehat{r}_i)$ for $i \in [N]$, where $N = |\mathsf{WireIndex}(f)|$.
4. For each $i \in \mathsf{Inputs}(f)$, sample a 0-label $W_i^0 \xleftarrow{\$} \mathcal{R}_p$, set $\pi_i := \mathsf{LSB}(W_i^0)$, and compute a 1-label $W_i^1 := W_i^0 + (-1)^{\pi_i} \cdot \Delta \in \mathcal{R}_p$. Then, for each $i \in \mathsf{Inputs}(f)$, set $v_i := \widehat{r}_i \oplus \pi_i$ and homomorphically compute $v_i \oplus \widehat{\tau}_i$ to obtain $\widetilde{\tau}_i := \mathsf{Enc}(\Delta, \pi_i)$.
5. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, $W_\alpha^{\pi_\alpha} = W_\alpha^0 - \pi_\alpha \cdot \Delta \in \mathcal{R}_p$ and $W_\beta^{\pi_\beta} = W_\beta^0 - \pi_\beta \cdot \Delta \in \mathcal{R}_p$, then do the following:
   (a) For each $i, j \in \{0,1\}$, compute $z_{g,i,j} = G(g, \pi_\alpha \oplus i, \pi_\beta \oplus j)$ and homomorphically compute $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$, resulting in the ciphertext $\widetilde{\tau}_{g,i,j}$ that encrypts $(-1)^{\widehat{\tau}_g} \cdot z_{g,i,j}$.
   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.
   (c) Run $\widetilde{\mathcal{L}}_{g,0,0} \leftarrow \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,3}) + \mathsf{PRF}(K, g)$. Then, compute $\pi_g := \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) \oplus z_{g,0,0}$ and $v_g := \widehat{r}_g \oplus \pi_g$.
   (d) Compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$ and $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ and $\mathcal{L}_{g,0,0} \leftarrow \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + \mathsf{PRF}(K, g)$. Then, homomorphically compute $v_g \oplus \widehat{\tau}_g$ to obtain a ciphertext $\widetilde{\tau}_g := \mathsf{Enc}(\Delta, \pi_g)$.
   (e) Compute the 0-label on the output wire $W_g^0 := \mathcal{L}_{g,0,0} - (-1)^{\pi_g} \cdot z_{g,0,0} \cdot \Delta$.
6. Output a garbled circuit $F = \big(\mathsf{params}, K, \sigma, \{v_i\}_{i \in \mathsf{WireIndex}(f)}\big)$, an encoding information $e = \big(\{W_i^0\}_{i \in \mathsf{Inputs}(f)}, \Delta\big)$ and a decoding information $d = \{\pi_i\}_{i \in \mathsf{Ouputs}(f)}$.

$\mathsf{Encode}(e, x)$: The garbler computes $\pi_i := \mathsf{LSB}(W_i^0)$ and $X_i := W_i^0 + (-1)^{\pi_i} \cdot x_i \cdot \Delta$ for each $i \in \mathsf{Inputs}(f)$, and outputs $X = \{X_i\}_{i \in \mathsf{Inputs}(f)}$.

$\mathsf{Eval}(F, X)$: The evaluator performs the following steps:

1. For each $i \in \mathsf{Inputs}(f)$, set $W_i := X_i$.
2. Homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N)$, where each $\widehat{\tau}_i = \mathsf{Enc}(\Delta, \widehat{r}_i)$ and $N = |\mathsf{WireIndex}(f)|$.
3. For each $i \in \mathsf{WireIndex}(f)$, homomorphically compute $v_i \oplus \widehat{\tau}_i$ to obtain an HE ciphertext $\widetilde{\tau}_i := \mathsf{Enc}(\Delta, \pi_i)$.
4. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, and then do the following:
   (a) For each $i, j \in \{0,1\}$, homomorphically compute an HE ciphertext $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$.
   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.
   (c) Compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$ and $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$. Run $W_g \leftarrow \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + \mathsf{PRF}(K, g)$.
5. Output $Y = \{W_i\}_{i \in \mathsf{Ouputs}(f)}$.

$\mathsf{Decode}(d, Y)$: The evaluator computes $y_i := \mathsf{LSB}(Y_i) \oplus \pi_i$ for each $i \in \mathsf{Ouputs}(f)$, and outputs y.

Figure 3: The Garble, Encode, Eval, Decode algorithms in our garbling scheme.

ciphertexts, each encrypting a pseudo-random bit, where N denotes the number of all wires. If the PRG has a multiplication depth of a small integer d, the SWHE scheme needs to support $d+2$ levels of homomorphic multiplications. We use a pseudo-random function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathcal{R}_p$ to randomize the output labels on each gate, where the key K is involved in the garbled circuit F following the previous approach [BKS19]. Note that PRF is not used to protect the privacy, and instead guarantees the correctness by making the distributed decryption property hold for each gate. Based on the technical overview in Section 2, it is easy to analyze the correctness of our garbling scheme by going through each algorithm, and for completeness, we provide the correctness analysis of our scheme in Appendix A.1.

The garbler can send some public parameters in params to the evaluator in advance. These parameters, which do not rely on the secret key, can be reused for generating multiple garbled circuits. Our garbling scheme (see Figure 3) can be further optimized as follows: for each $i \in \mathsf{Inputs}(f)$, the garbler directly sets $\pi_i = \widehat{r}_i$ (i.e., setting $v_i = 0$), and then samples a 0-label $W_i^0 \xleftarrow{\$} \mathcal{R}_p$ such that $\mathsf{LSB}(W_i^0) = \pi_i$. This optimization can reduce the communication of the garbled table by $|\mathsf{Inputs}(f)|$ bits, requiring only $\lambda \cdot |ct| + |C|$ bits, where $|ct| = O(\lambda \cdot \mathrm{polylog}\lambda)$ represents the size of a single HE ciphertext and $|C|$ is the circuit size.

Complexity analysis. To analyze the communication cost, we focus on the size of the garbled circuit F, as it dominates the communication. The size of F is given by $\lambda \cdot |ct| + |C| + \lambda$ bits. Thus, our scheme essentially achieves a total communication cost of just 1 bit per gate.

For the computational cost, we focus on the number of homomorphic operations, which dominate the computation. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{|C|}$ be a PRG requiring n additions, m multiplications. Consequently, the ciphertexts $\widehat{\tau}_1, \cdots, \widehat{\tau}_{|C|}$ are generated using n homomorphic additions and m homomorphic multiplications. Thus, when selecting a low-complexity PRG (e.g., [App12, AIK04, AIK08, AK19, CM01]), we have that $n + m = O(|C|)$, leading to an amortized computation cost of $O(1)$ homomorphic addition/multiplication operations per gate. In addition to homomorphic operations associated with the PRG, for each gate, both parties perform $O(1)$ homomorphic addition/multiplication operations to produce a constant number of HE ciphertexts, and execute $O(1)$ instances of the algorithms Dec and $\widehat{\mathsf{Dec}}$ to generate $(\widetilde{\mathcal{L}}_{g,0,0}, \mathcal{L}_{g,0,0})$ or W_g. As detailed in Section 5 and Appendix C, for our RLWE-based and NTRU-based instantiations, both Dec and $\widehat{\mathsf{Dec}}$ require only $O(1)$ normal addition/multiplication operations over a polynomial ring, which is a small part of the total computation. In conclusion, the amortized computation cost per gate is primarily determined by $O(1)$ homomorphic addition/multiplication operations.

Proof of security. We show that our garbling scheme satisfies both privacy and obliviousness, as defined in Section 3.2. Similar to the proof approach of the classical garbling scheme half-gates [ZRE15], we replace all garbled tables with uniform elements in one hybrid. In particular, our garbled tables consist of HE ciphertexts on uniform bits, as well as the XOR of uniform bits and wire masks. Note that for an even p, $\mathsf{LSB}(W)$ is a uniform bit, if $W \in \mathcal{R}_p$ is a random element.

Theorem 1. *Our garbling scheme (Figure 3) satisfies both privacy (Definition 2) and obliviousness (Definition 3), provided that the SWHE scheme with distributed decryption is CPA secure,* PRG *is a pseudorandom generator, and* PRF *is a pseudorandom function.*

The proof of Theorem 1 is postponed to Appendix A.2.

5 SWHE with Distributed Decryption from RLWE

In this section, we present a somewhat homomorphic encryption (SWHE) scheme with distributed decryption under the RLWE assumption, building on the GSW scheme [GSW13]. We first give an overview of the original GSW-SWHE scheme, and then show how to achieve the distributed decryption property.

5.1 GSW-SWHE from RLWE

Firstly, we recall some useful functions used in the GSW scheme [GSW13], including $\mathsf{BitDecomp}$, $\mathsf{BitDecomp}^{-1}$, $\mathsf{Flatten}$ and $\mathsf{Powersof2}$. For $a \in \mathcal{R}_q$, $\mathsf{BitDecomp}(a)$ outputs an ℓ-dimension row vector $(a_0, a_1, \ldots, a_{\ell-1})$, representing the bit decomposition of a, where $\ell = \lfloor \log(q) \rfloor + 1$, and $a_i \in \mathcal{R}_2$ is a ring element such that each coefficient is the i-bit component of the binary representation of the a's corresponding coefficient. Here we write $\mathcal{R}_2 = \mathbb{Z}_2[X]/(X^n + 1)$. For an ℓ-dimension row vector $\boldsymbol{a} = (a_0, a_1, \ldots, a_{\ell-1})$, the inverse of $\mathsf{BitDecomp}(\boldsymbol{a})$, denoted by $\mathsf{BitDecomp}^{-1}(\boldsymbol{a})$, is defined by $\sum_{i=0}^{\ell-1}(2^i \cdot a_i)$ over $\mathcal{R}_q$. For an ℓ-dimension row vector $\boldsymbol{a}$, $\mathsf{Flatten}(\boldsymbol{a})$ is defined as $\mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\boldsymbol{a}))$ and outputs an ℓ-dimension row vector, where all coefficients of each component are bits. For $b \in \mathcal{R}_q$, $\mathsf{Powersof2}(b)$ generates an ℓ-dimension row vector $\left(2^0 \cdot b, 2^1 \cdot b, \ldots, 2^{\ell-1} \cdot b\right)$. For any $\boldsymbol{a} \in \mathcal{R}_q^\ell$ and $b \in \mathcal{R}_q$, we have

$$\langle\, \boldsymbol{a}\,,\, \mathsf{Powersof2}(b)\,\rangle = \mathsf{BitDecomp}^{-1}(\boldsymbol{a}) \cdot b = \langle\, \mathsf{Flatten}(\boldsymbol{a})\,,\, \mathsf{Powersof2}(b)\,\rangle\ ,$$

where for any two vectors $\boldsymbol{x}, \boldsymbol{y}$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes their inner product. All the above operations can be applied to matrices as well. For example, for a matrix $\mathbf{A} \in \mathcal{R}_q^{m \times n}$, where $\mathbf{A}_{i,j}$ represents the entry in the i-th row and j-th column of $\mathbf{A}$, $\mathsf{BitDecomp}(\mathbf{A})$ is defined as:

$$\mathsf{BitDecomp}(\mathbf{A}) \stackrel{\text{def}}{=} \begin{bmatrix} \mathsf{BitDecomp}(\mathbf{A}_{1,1}) & \cdots & \mathsf{BitDecomp}(\mathbf{A}_{1,n}) \\ \vdots & \vdots & \vdots \\ \mathsf{BitDecomp}(\mathbf{A}_{m,1}) & \cdots & \mathsf{BitDecomp}(\mathbf{A}_{m,n}) \end{bmatrix} \in \mathcal{R}_2^{m \times (n\ell)}\ .$$

The GSW-SWHE scheme in the private-key setting consists of the following three algorithms:

- **Key generation.** $\mathsf{Gen}(1^\lambda, 1^L)$ samples a secret key $sk \leftarrow \mathcal{D}_{\mathsf{B}}$, and outputs sk along with a set of parameters $\mathsf{params} = (n, p, q, \mathsf{B}, \ell, N, s)$, where $\ell = \lfloor \log(q) \rfloor + 1$, $N = 2\ell$, $q > 4\mathsf{B} \cdot p \cdot (N+1)^L \cdot n^L$ and $s = \lfloor \log(q/p) \rfloor$.

- **Encryption.** On input a secret key sk and a message $m \in \mathbb{Z}_p$, $\mathsf{Enc}(sk, m)$ samples $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^N$ and $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathsf{B}}^N$, computes $\boldsymbol{b} := \boldsymbol{a} \cdot sk + \boldsymbol{e}$, and then outputs

$$\tau \leftarrow \mathsf{Flatten}\left(m \cdot \mathbf{I}_N + \mathsf{BitDecomp}\left(\begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \end{bmatrix}\right)\right)\ ,$$

  where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

- **Decryption.** On input a secret key sk and a ciphertext $\tau \in \mathcal{R}_2^{N \times N}$, $\mathsf{Dec}(sk, \tau)$ sets a vector $\boldsymbol{v}^\mathsf{T} := \mathsf{Powersof2}\left(\begin{bmatrix} -sk & 1 \end{bmatrix}\right)$, and then computes

$$\tau \cdot \boldsymbol{v} := \left(m \cdot \mathbf{I}_N + \mathsf{BitDecomp}\left(\begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \end{bmatrix}\right)\right) \cdot \boldsymbol{v} = m \cdot \boldsymbol{v} + \boldsymbol{b} - \boldsymbol{a} \cdot sk \approx m \cdot \boldsymbol{v}\,. \tag{5}$$

  Observe that the last ℓ components of $\boldsymbol{v}^\mathsf{T}$ are $\mathsf{Powersof2}(1) = \begin{bmatrix} 1 & 2 & \cdots & 2^{\ell-1} \end{bmatrix}$. Then, it extracts the $(\ell+s+1)$-th component of $\tau \cdot \boldsymbol{v}$, denoted as $(\tau \cdot \boldsymbol{v})\,[\ell+s+1]$, and outputs $m := \left\lfloor \frac{(\tau \cdot \boldsymbol{v})[\ell+s+1]}{2^s} \right\rceil$.

Following the analysis in [GSW13], the error of the final ciphertext in GSW-SWHE is bounded by $(N+1)^L \cdot n^L \cdot \mathsf{B}$, where L represents the multiplication depth of a circuit. For any $sk \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$, two messages m_1 and m_2, and their ciphertexts $\tau_1 = \mathsf{Enc}(sk, m_1)$ and $\tau_2 = \mathsf{Enc}(sk, m_2)$, the homomorphic addition and multiplication operations Add and Mult are respectively defined as follows:

- $\mathsf{Add}(\tau_1, \tau_2)$: To add ciphertexts $\tau_1, \tau_2 \in \mathcal{R}_2^{N \times N}$, output $\mathsf{Flatten}(\tau_1 + \tau_2)$. The correctness of this operation is obvious.
- $\mathsf{Mult}(\tau_1, \tau_2)$: To multiply ciphertexts $\tau_1, \tau_2 \in \mathcal{R}_2^{N \times N}$, output $\mathsf{Flatten}(\tau_1 \cdot \tau_2)$. To verify the correctness, we decrypt $\mathsf{Mult}(\tau_1, \tau_2)$ as $\mathsf{Mult}(\tau_1, \tau_2) \cdot \boldsymbol{v}$:

$$\mathsf{Mult}(\tau_1, \tau_2) \cdot \boldsymbol{v} = \tau_1 \cdot \tau_2 \cdot \boldsymbol{v} \approx \tau_1 \cdot m_2 \cdot \boldsymbol{v} = m_2 \cdot \tau_1 \cdot \boldsymbol{v} \approx m_1 \cdot m_2 \cdot \boldsymbol{v} \,,$$

  where the approximate equality "$\approx$" is due to Equation (5).

5.2 Our SWHE Scheme with Distributed Decryption from RLWE

Our SWHE scheme extends the GSW scheme to support distributed decryption. Inspired by prior work [BKS19], the linear distributed decryption can be directly applied to GSW ciphertexts. However, supporting the correlated-key distributed decryption property is more challenging. To address this, we introduce an extended GSW (eGSW) scheme based on the KDM security.

Compared to the GSW scheme, the eGSW scheme additionally introduces a secret key $\widehat{sk}$ and an extended encryption algorithm $\widehat{\tau} \leftarrow \widehat{\mathsf{Enc}}(sk, \widehat{sk}, m)$. We are able to convert a GSW ciphertext $\mathsf{Enc}(sk, m)$ to an eGSW ciphertext $\widehat{\mathsf{Enc}}(sk, \widehat{sk}, m)$, and can also non-interactively convert between an additive secret sharing of sk and that of $\widehat{sk}$. Both conversions have been explained in Section 2.5, and their details are shown as below. In addition, eGSW samples $sk, \widehat{sk} \leftarrow \mathcal{D}_{\mathsf{B}}$ such that $sk[1] = \widehat{sk}[1] = 1$. Furthermore, eGSW modifies the decryption algorithm Dec to realize linear distributed decryption as follows:

1. Dec now outputs $m \cdot sk$ rather than m.
2. Dec computes an inner product to approximately yield $(q/p) \cdot m \cdot sk$, and then performs the rounding.

The encryption algorithm Enc, homomorphic addition Add and homomorphic multiplication Mult of the eGSW scheme is the same as the GSW scheme, and thus they are omitted. The other algorithms of our eGSW scheme are as follows:

- **Key generation.** $\mathsf{Gen}(1^\lambda, 1^L)$ samples two secret keys $sk, \widehat{sk} \leftarrow \mathcal{D}_{\mathsf{B}}$ such that $sk[1] = \widehat{sk}[1] = 1$, and then generates a set of public parameters $\mathsf{params} = \{n, p, q, \mathsf{B}, \ell, N, \tau_{sk \to \widehat{sk}}, \tau_{\widehat{sk} \to sk}, \tau_{\mathsf{gsw} \to \mathsf{egsw}}\}$ as below:
  - $\mathsf{B} = \mathsf{poly}(\lambda)$, $p \mid q$, $p = \lambda^{\omega(1)}$ is an even, $q/p^2 = \lambda^{\omega(1)} \cdot (N+1)^L \cdot n^L$, $\ell = \lfloor \log(q) \rfloor + 1$ and $N = 2\ell$.
  - Sample $a, \widehat{a} \xleftarrow{\$} \mathcal{R}_q$ and $e, \widehat{e} \leftarrow \mathcal{D}_{\mathsf{B}}$, and then compute two key-switching ciphertexts $\tau_{sk \to \widehat{sk}}$ and $\tau_{\widehat{sk} \to sk}$ as follows:

$$\tau_{sk \to \widehat{sk}} = (\, a \,,\, b = a \cdot sk + e + (q/p) \cdot \widehat{sk} \,) \,,$$
$$\tau_{\widehat{sk} \to sk} = (\, \widehat{a} \,,\, \widehat{b} = \widehat{a} \cdot \widehat{sk} + \widehat{e} + (q/p) \cdot sk \,) \,.$$

– Sample $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^N$ and $\boldsymbol{e}^1, \boldsymbol{e}^2, \boldsymbol{e}^3 \leftarrow \mathcal{D}_{\mathsf{B}}^N$, and then compute a GSW-to-eGSW ciphertext $\tau_{\mathsf{gsw}\to\mathsf{egsw}}$ as follows:

$$\tau_{\mathsf{gsw}\to\mathsf{egsw}} = \left(\boldsymbol{a}, \boldsymbol{b}^1 = \boldsymbol{a}\cdot sk + \boldsymbol{e}^1, \boldsymbol{b}^2 = \boldsymbol{a}\cdot \widehat{sk} + \boldsymbol{e}^2, \boldsymbol{b}^3 = \boldsymbol{b}^1\cdot \widehat{sk} + \boldsymbol{e}^3 + \boldsymbol{v}\cdot \widehat{sk}\right),$$

where $\boldsymbol{v} \stackrel{\text{def}}{=} \mathsf{Powersof2}\left(\begin{bmatrix}-sk & 1\end{bmatrix}\right)^{\mathsf{T}}$.

- **Decryption.** On input a key $x \in \mathcal{R}$ (it is either the secret key sk or an additive share of sk for our instantiation) and a ciphertext $\tau \in \mathcal{R}_2^{N\times N}$, $\mathsf{Dec}(x,\tau)$ computes over $\mathcal{R}_q$

$$(\alpha,\beta) := \mathsf{BitDecomp}^{-1}\left(-\mathsf{BitDecomp}\left(\begin{bmatrix}(q/p) & 0\end{bmatrix}\right)\cdot\tau\right),$$

and then output $\left\lfloor \mathsf{LSB}(x)\cdot\beta - \alpha\cdot x\right\rceil_p$. Note that if $x = sk$ with $\mathsf{LSB}(sk) = 1$, then we have $\beta - \alpha\cdot sk \approx (q/p)\cdot m\cdot sk$ and thus $\left\lfloor \beta - \alpha\cdot sk\right\rceil_p = m\cdot sk$.

- **Extended encryption.** On input two secret keys $sk, \widehat{sk}$ as well as a message $m \in \mathbb{Z}_p$, $\widehat{\mathsf{Enc}}(sk, \widehat{sk}, m)$ samples $a \xleftarrow{\$} \mathcal{R}_q$ and $e^1, e^2, e^3 \leftarrow \mathcal{D}_{\mathsf{B}}$, and then outputs an eGSW ciphertext

$$\widehat{\tau} = \left(\, a\,,\, b^1 = a\cdot sk + e^1\,,\, b^2 = a\cdot\widehat{sk} + e^2\,,\, b^3 = b^1\cdot\widehat{sk} + e^3 + (q/p)\cdot m\cdot\widehat{sk}\,\right).$$

- **GSW-to-eGSW conversion $[\cdot]_{\mathsf{gsw}\to\mathsf{egsw}}$.** Given a GSW ciphertext $\tau = \mathsf{Enc}(sk, m)$ for a message $m \in \mathbb{Z}_p$ and the GSW-to-eGSW ciphertext $\tau_{\mathsf{gsw}\to\mathsf{egsw}} = (\boldsymbol{a}, \boldsymbol{b}^1, \boldsymbol{b}^2, \boldsymbol{b}^3)$, the GSW-to-eGSW conversion function $\left[\tau\right]_{\mathsf{gsw}\to\mathsf{egsw}}$ generates an eGSW ciphertext on the message m as

$$\left(a = \langle \boldsymbol{u}, \boldsymbol{a}\rangle\,,\, b^1 = \langle \boldsymbol{u}, \boldsymbol{b}^1\rangle\,,\, b^2 = \langle \boldsymbol{u}, \boldsymbol{b}^2\rangle\,,\, b^3 = \langle \boldsymbol{u}, \boldsymbol{b}^3\rangle \approx b^1\cdot\widehat{sk} + (q/p)\cdot m\cdot\widehat{sk}\right),$$

where $\boldsymbol{u} = \mathsf{BitDecomp}\left(\begin{bmatrix}0 & (q/p)\end{bmatrix}\right)\cdot\tau$.

- **Key-switching.** Given the key-switching ciphertexts $\tau_{sk\to\widehat{sk}}$ and $\tau_{\widehat{sk}\to sk}$, we define two key-switching functions $[\cdot]_{sk\to\widehat{sk}}$ and $[\cdot]_{\widehat{sk}\to sk}$. In particular, for any $y \in \mathcal{R}_p$, we have

$$[y]_{sk\to\widehat{sk}} \text{ outputs } \left\lfloor y[1]\cdot b - a\cdot y\right\rceil_p, \text{ and } [y]_{\widehat{sk}\to sk} \text{ outputs } \left\lfloor y[1]\cdot \widehat{b} - \widehat{a}\cdot y\right\rceil_p.$$

Let $sk_0, \widehat{sk_0} \in \mathcal{R}_p$ be two uniform elements such that $\mathsf{LSB}(sk_0) = 0$ and $\mathsf{LSB}(\widehat{sk_0}) = 0$. Let $sk_1 = sk_0 + sk$ and $\widehat{sk_1} = \widehat{sk_0} + \widehat{sk}$ over $\mathcal{R}_p$. For correctness, we have $\left[sk_1\right]_{sk\to\widehat{sk}} = \left[sk_0\right]_{sk\to\widehat{sk}} + \widehat{sk}$ and $\left[\widehat{sk_1}\right]_{\widehat{sk}\to sk} = \left[\widehat{sk_0}\right]_{\widehat{sk}\to sk} + sk$ over $\mathcal{R}_p$. See Section 2.5 and the following Theorem 2 for the correctness analysis.

- **Algorithm $\widehat{\mathsf{Dec}}$ for correlated-key distributed decryption.** Given a ciphertext $\tau = \mathsf{Enc}(sk, m)$ as well as two additive secret sharings (sk_0, sk_1) and (sk_0', sk_1') such that $\mathsf{LSB}(sk_0) = \mathsf{LSB}(sk_0') = 0$, $sk_1 = sk_0 + sk$ and $sk_1' = sk_0' + sk$ over $\mathcal{R}_p$, for any $i, j \in \{0,1\}$, $\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau)$ performs the following steps:

1. Perform a key-switching operation $\widehat{sk_j} := \left[sk_j'\right]_{sk\to\widehat{sk}}$.
2. Convert τ into an eGSW ciphertext by computing $(a, b^1, b^2, b^3) := \left[\tau\right]_{\mathsf{gsw}\to\mathsf{egsw}}$.

3. Compute $x := sk_i \cdot \widehat{sk}_j \cdot a - i \cdot \widehat{sk}_j \cdot b^1 - j \cdot sk_i \cdot b^2 + i \cdot j \cdot b^3$ over $\mathcal{R}_q$, where $i = \mathsf{LSB}(sk_i)$ and $j = \mathsf{LSB}(sk'_j)$.
4. Set $y := \lfloor x \rceil_p$ and perform another key-switching operation $z := [y]_{\widehat{sk} \to sk}$. Output $z \in \mathcal{R}_p$.

Theorem 2. *Our eGSW scheme is a somewhat homomorphic encryption scheme with distributed decryption under the RLWE assumption and the assumption that the eGSW scheme is key-dependent message (KDM) secure.*

Proof. We first analyze the correctness of the decryption algorithm when the input is secret key sk. In particular, we have

$$\begin{aligned}
\beta - \alpha \cdot sk &= \begin{bmatrix} \alpha & \beta \end{bmatrix} \cdot \begin{bmatrix} -sk \\ 1 \end{bmatrix} = \mathsf{BitDecomp}\left(\begin{bmatrix} \alpha & \beta \end{bmatrix}\right) \cdot \boldsymbol{v} \\
&= -\mathsf{BitDecomp}\left(\begin{bmatrix} (q/p) & 0 \end{bmatrix}\right) \cdot \tau \cdot \boldsymbol{v} \\
&\approx -\mathsf{BitDecomp}\left(\begin{bmatrix} (q/p) & 0 \end{bmatrix}\right) \cdot m \cdot \boldsymbol{v} \\
&= (q/p) \cdot m \cdot sk \,,
\end{aligned}$$

where $\boldsymbol{v} \stackrel{\text{def}}{=} \mathsf{Powersof2}\left(\begin{bmatrix} -sk & 1 \end{bmatrix}\right)^{\mathsf{T}}$ and $\tau \cdot \boldsymbol{v} \approx m \cdot \boldsymbol{v}$ based on Equation (5). Then, taking the rounding $\lfloor \cdot \rceil_p$ on both sides of the above equation, we get $\left\lfloor \beta - \alpha \cdot sk \right\rceil_p = m \cdot sk$.

Following the GSW scheme [GSW13], GSW ciphertexts support homomorphic addition and multiplication operations for a maximum multiplication depth L. It is straightforward to see that eGSW ciphertexts support homomorphic addition operations. That is, given two messages m_1, m_2 along with their eGSW ciphertexts $\widehat{\tau}_1 = \widehat{\mathsf{Enc}}(sk, \widehat{sk}, m_1)$ and $\widehat{\tau}_2 = \widehat{\mathsf{Enc}}(sk, \widehat{sk}, m_2)$, we have $\widehat{\tau}_1 + \widehat{\tau}_2 = \widehat{\mathsf{Enc}}(sk, \widehat{sk}, m_1 + m_2)$. Therefore, given a GSW ciphertext $\tau = \mathsf{Enc}(sk, m)$, the GSW-to-eGSW conversion $[\tau]_{\mathsf{gsw} \to \mathsf{egsw}}$ outputs an eGSW ciphertext on the message m. Additionally, we can easily see that $-[\tau]_{\mathsf{gsw} \to \mathsf{egsw}} = [-\tau]_{\mathsf{gsw} \to \mathsf{egsw}}$.

CPA security. We use a sequence of hybrids to prove that $\mathsf{Enc}(sk, m_0)$ is computationally indistinguishable from $\mathsf{Enc}(sk, m_1)$ for two messages $m_0, m_1 \in \mathbb{Z}_p$ chosen by a PPT adversary $\mathcal{A}$.

Hybrid 0. This is the real game and denoted by $\mathcal{G}_0$. This hybrid sends params and $\mathsf{Enc}(sk, m_c)$ to $\mathcal{A}$ for a random bit c.

Hybrid 1. This hybrid, denoted by $\mathcal{G}_1$, is the same as $\mathcal{G}_0$, except that replacing three ciphertexts in params with the ciphertexts on the message 0.

The eGSW scheme as described above employs three encryption algorithms: $\mathsf{Enc}(sk, \cdot)$, $\mathsf{Enc}(\widehat{sk}, \cdot)$, and $\widehat{\mathsf{Enc}}(sk, \widehat{sk}, \cdot)$. It includes three ciphertexts on key-dependent messages: two key-switching ciphertexts $\tau_{sk \to \widehat{sk}}$ and $\tau_{\widehat{sk} \to sk}$ as well as one GSW-to-eGSW ciphertext $\tau_{\mathsf{gsw} \to \mathsf{egsw}}$. Based on the assumption that the eGSW scheme is KDM secure, the adversary cannot distinguish these ciphertexts on secret keys $sk, \widehat{sk}$ from three ciphertexts on zero. Therefore, $\mathcal{G}_1$ is computationally indistinguishable from $\mathcal{G}_0$.

Hybrid 2. This hybrid, denoted by $\mathcal{G}_2$, is the same as $\mathcal{G}_1$, except for replacing the ciphertexts on zero in params with random vectors or matrices over $\mathcal{R}_q$.

In $\mathcal{G}_1$, the key-switching ciphertexts $\tau_{sk \to \widehat{sk}} = (a, b = a \cdot sk + e)$ and $\tau_{\widehat{sk} \to sk} = (\widehat{a}, \widehat{b} = \widehat{a} \cdot \widehat{sk} + \widehat{e})$. Under the RLWE assumption, they are computationally indistinguishable from random vectors in $\mathcal{R}_q^2$.

The GSW-to-eGSW ciphertext $\tau_{\mathsf{gsw}\to\mathsf{egsw}} = \left(\boldsymbol{a}, \boldsymbol{b}^1 = \boldsymbol{a}\cdot sk + \boldsymbol{e}^1, \boldsymbol{b}^2 = \boldsymbol{a}\cdot\widehat{sk} + \boldsymbol{e}^2, \boldsymbol{b}^3 = \boldsymbol{b}^1\cdot\widehat{sk} + \boldsymbol{e}^3\right)$ in $\mathcal{G}_1$. Under the RLWE assumption, we can first replace $\boldsymbol{b}^1$ with a uniform vector $\boldsymbol{u}^1 \xleftarrow{\$} \mathcal{R}_q^N$. Then, we can replace $(\boldsymbol{b}^2, \boldsymbol{b}^3)$ with two uniform vectors $\boldsymbol{u}^2, \boldsymbol{u}^3 \xleftarrow{\$} \mathcal{R}_q^N$ under the RLWE assumption. Therefore, $\mathcal{G}_2$ is computationally indistinguishable from $\mathcal{G}_1$.

Hybrid 3. This hybrid, denoted by $\mathcal{G}_3$, is the same as $\mathcal{G}_2$, except that replacing $\boldsymbol{a}, \boldsymbol{b}$ used to generate $\mathsf{Enc}(sk, m_c)$ with random vectors in $\mathcal{R}_q^N$.

It is easy to see that $\mathcal{G}_3$ is computationally indistinguishable from $\mathcal{G}_2$ under the RLWE assumption. In hybrid $\mathcal{G}_3$, the challenge ciphertext is produced with random vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{R}_q^N$, which makes the ciphertext be independent of bit c. Therefore, the advantage of $\mathcal{A}$ to guess the bit c in $\mathcal{G}_3$ is 0. Due to that $\mathcal{G}_0$ is computationally indistinguishable from $\mathcal{G}_3$ via the sequence of hybrids, we obtain that the advantage of $\mathcal{A}$ (guessing c) in $\mathcal{G}_0$ is negligible in λ.

Linear distributed decryption. Let $sk_0 \in \mathcal{R}_p$ be a uniform element with $\mathsf{LSB}(sk_0) = 0$. Let $sk_1 = sk_0 + sk$ over $\mathcal{R}_p$. Based on Lemma 1, we have

$$\Pr\left[\, sk_1 = sk_0 + sk \text{ (without modulo } p)\, \right] \geq 1 - n\cdot(\|sk\|_\infty + 1)/p = 1 - \mathsf{negl}(\lambda)\ .$$

Given a GSW ciphertext $\tau = \mathsf{Enc}(sk, m)$, the Dec algorithm first computes

$$(\alpha, \beta) = \mathsf{BitDecomp}^{-1}\left(-\mathsf{BitDecomp}\left(\begin{bmatrix}(q/p) & 0\end{bmatrix}\right)\cdot\tau\right)\ .$$

Then, we have $\beta - \alpha\cdot sk_1 = -\alpha\cdot sk_0 + \beta - \alpha\cdot sk$. Based on Lemma 2, with probability at least $1 - n\cdot(\|e\|_\infty + 1)\cdot p/q = 1 - \mathsf{negl}(\lambda)$, we have

$$\mathsf{Dec}(sk_1, \tau) = \mathsf{Dec}(sk_0, \tau) + \mathsf{Dec}(sk, \tau)\,,$$

where $\mathsf{Dec}(x, \tau) = \lfloor \mathsf{LSB}(x)\cdot\beta - \alpha\cdot x\rceil_p$ for $x \in \{sk_0, sk_1, sk\}$, and that $\mathsf{LSB}(sk_0) = 0$ and $\mathsf{LSB}(sk) = 1$ imply $\mathsf{LSB}(sk_1) = 1$ for an even p. For $i \in \{0, 1\}$, we observe

$$\mathsf{Dec}(sk_i, \tau) = -\mathsf{Dec}(sk_i, -\tau)\ ,$$

because, for any $x \in \mathcal{R}_q$ and $\boldsymbol{y} \in \mathcal{R}_q^N$, $\lfloor -x\rceil_p = -\lfloor x\rceil_p$ and $-\mathsf{BitDecomp}^{-1}(\boldsymbol{y}) = \mathsf{BitDecomp}^{-1}(-\boldsymbol{y})$. The correctness of key-switching functions can be analyzed similarly (see also Section 2.5 for the correctness analysis).

Correlated-key distributed decryption. Let $sk_0, sk_0' \in \mathcal{R}_p$ be two uniform elements such that $\mathsf{LSB}(sk_0) = 0$ and $\mathsf{LSB}(sk_0') = 0$. Let $sk_1 = sk_0 + sk$ and $sk_1' = sk_0' + sk$ over $\mathcal{R}_p$. Given any GSW ciphertext τ as well as any $x \in \mathcal{R}_q$ and $y \in \mathcal{R}_p$, we have that $-[\tau]_{\mathsf{gsw}\to\mathsf{egsw}} = [-\tau]_{\mathsf{gsw}\to\mathsf{egsw}}$, $\lfloor -x\rceil_p = -\lfloor x\rceil_p$ and $-[y]_{\widehat{sk}\to sk} = [-y]_{\widehat{sk}\to sk}$, except with probability $\mathsf{negl}(\lambda)$. Therefore, except with probability $\mathsf{negl}(\lambda)$, for any $i, j \in \{0, 1\}$, we have

$$\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) = -\widehat{\mathsf{Dec}}(sk_i, sk_j', -\tau)\ .$$

Below, we show that for any $i, j \in \{0, 1\}$, with probability $1 - \mathsf{negl}(\lambda)$,

$$\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) = \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) + i\cdot j\cdot\mathsf{Dec}(sk, \tau)\ .$$

Based on Lemma 1, we obtain $sk_1 = sk_0 + sk$ and $sk_1' = sk_0' + sk$ (without modulo p), except with probability $\mathsf{negl}(\lambda)$. The eGSW ciphertext $(a, b^1, b^2, b^3) = [\tau]_{\mathsf{gsw}\to\mathsf{egsw}}$ satisfies the following relation:

$$\left(\, a\,,\ b^1 \approx a\cdot sk\,,\ b^2 \approx a\cdot\widehat{sk}\,,\ b^3 \approx a\cdot sk\cdot\widehat{sk} + (q/p)\cdot m\cdot\widehat{sk}\,\right)\,,$$

where the errors are omitted. For each $i, j \in \{0, 1\}$, we have the following:

- If $(i,j) = (0,0)$, this is a trivial case. Let $x_{00} = sk_0 \cdot \widehat{sk_0} \cdot a$.
- If $(i,j) = (1,0)$, from $sk_1 = sk_0 + sk$, we have

$$x_{10} = sk_1 \cdot \widehat{sk_0} \cdot a - \widehat{sk_0} \cdot b^1 = \widehat{sk_0} \cdot (a \cdot sk_0 + a \cdot sk - b^1) \approx sk_0 \cdot \widehat{sk_0} \cdot a = x_{00} \,.$$

  Based on Lemma 2, we have $\lfloor x_{10} \rceil_p = \lfloor x_{00} \rceil_p$ except with probability $\mathsf{negl}(\lambda)$, which implies $\widehat{\mathsf{Dec}}(sk_1, sk'_0, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau)$.
- If $(i,j) = (0,1)$, from $\widehat{sk_1} = \widehat{sk_0} + \widehat{sk}$, we have

$$x_{01} = sk_0 \cdot \widehat{sk_1} \cdot a - sk_0 \cdot b^2 = sk_0 \cdot (a \cdot \widehat{sk_0} + a \cdot \widehat{sk} - b^2) \approx sk_0 \cdot \widehat{sk_0} \cdot a = x_{00} \,.$$

  Based on Lemma 2, we obtain $\lfloor x_{01} \rceil_p = \lfloor x_{00} \rceil_p$ except with probability $\mathsf{negl}(\lambda)$, meaning that $\widehat{\mathsf{Dec}}(sk_0, sk'_1, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau)$.
- If $(i,j) = (1,1)$, from $sk_1 = sk_0 + sk$ and $\widehat{sk_1} = \widehat{sk_0} + \widehat{sk}$, we obtain

$$sk_1 \cdot \widehat{sk_1} = sk_0 \cdot \widehat{sk_0} + sk \cdot \widehat{sk_0} + sk_0 \cdot \widehat{sk} + sk \cdot \widehat{sk} = sk_0 \cdot \widehat{sk_0} + sk \cdot \widehat{sk_1} + sk_1 \cdot \widehat{sk} - sk \cdot \widehat{sk} \,.$$

  Therefore, we have

$$\begin{aligned} x_{11} &= sk_1 \cdot \widehat{sk_1} \cdot a - \widehat{sk_1} \cdot b^1 - sk_1 \cdot b^2 + b^3 \\ &= (sk_0 \cdot \widehat{sk_0} + sk \cdot \widehat{sk_1} + sk_1 \cdot \widehat{sk} - sk \cdot \widehat{sk}) \cdot a - \widehat{sk_1} \cdot b^1 - sk_1 \cdot b^2 + b^3 \\ &= sk_0 \cdot \widehat{sk_0} \cdot a + \widehat{sk_1} \cdot (a \cdot sk - b^1) + sk_1 \cdot (a \cdot \widehat{sk} - b^2) + (b^3 - a \cdot sk \cdot \widehat{sk}) \\ &\approx sk_0 \cdot \widehat{sk_0} \cdot a + (q/p) \cdot m \cdot \widehat{sk} = x_{00} + (q/p) \cdot m \cdot \widehat{sk} \,. \end{aligned}$$

  Let $y_{11} = \lfloor x_{11} \rceil_p$ and $y_{00} = \lfloor x_{00} \rceil_p$. From Lemma 2, we get $y_{11} = y_{00} + m \cdot \widehat{sk}$ over $\mathcal{R}_p$, except with probability $\mathsf{negl}(\lambda)$. Then, based on the key-switching property, we have

$$\left[y_{11}\right]_{\widehat{sk} \to sk} = \left[y_{00}\right]_{\widehat{sk} \to sk} + m \cdot sk \,,$$

  which implies $\widehat{\mathsf{Dec}}(sk_1, sk'_1, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau) + \mathsf{Dec}(sk, \tau)$.

We complete the proof by combining all the above analyses. □

References

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Berlin, Heidelberg, August 2016.

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 171–180. ACM Press, June 2010.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 3–35. Springer, Cham, August 2017.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc^0. *Comput. Complex.*, 17(1):38–69, 2008.

[AK19] Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In David Zuckerman, editor, *60th FOCS*, pages 171–179. IEEE Computer Society Press, November 2019.

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 455–472. Springer, Cham, November 2018.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003.

[App12] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 805–816. ACM Press, May 2012.

[APRR24] Navid Alamati, Guru-Vamsi Policharla, Srinivasan Raghuraman, and Peter Rindal. Improved alternating-moduli PRFs and post-quantum signatures. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VIII*, volume 14927 of *LNCS*, pages 274–308. Springer, Cham, August 2024.

[BCFK21] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Cham, May 2021.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Cham, November 2018.

[BIP+22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 188–215. Springer, Cham, December 2022.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 3–33. Springer, Cham, May 2019.

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, December 2013.

[BLO16] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 578–590. ACM Press, October 2016.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012.

[CF13] Dario Catalano and Dario Fiore. Practical homomorphic MACs for arithmetic circuits. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 336–352. Springer, Berlin, Heidelberg, May 2013.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015.

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in nc^0. In Jirí Sgall, Ales Pultr, and Petr Kolman, editors, *MFCS 2001*, volume 2136 of *LNCS*, pages 272–284. Springer, 2001.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Cham, August 2017.

[DGH+21] Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 517–547, Virtual Event, August 2021. Springer, Cham.

[Dv21] Léo Ducas and Wessel P. J. van Woerden. NTRU fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 3–32. Springer, Cham, December 2021.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.

[FNO15] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 191–219. Springer, Berlin, Heidelberg, April 2015.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Cham, May 2020.

[FP83] Ulrich Fincke and Michael Pohst. A procedure for determining algebraic integers of given norm. In *European Computer Algebra Conference 1983*, pages 194–202. Springer, 1983.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Berlin, Heidelberg, May 2013.

[GGH+13b] Craig Gentry, Sergey Gorbunov, Shai Halevi, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. How to compress (reusable) garbled circuits. Cryptology ePrint Archive, Report 2013/687, 2013.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Berlin, Heidelberg, August 2010.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GLNP15] Shay Gueron, Yehuda Lindell, Ariel Nof, and Benny Pinkas. Fast garbling of circuits under standard assumptions. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 567–578. ACM Press, October 2015.

[GNS23] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.

[Gol11] Oded Goldreich. Candidate one-way functions based on expander graphs. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650 of *LNCS*, pages 76–87. Springer, 2011.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Berlin, Heidelberg, August 2013.

[GW13] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 301–320. Springer, Berlin, Heidelberg, December 2013.

[HK20] David Heath and Vladimir Kolesnikov. Stacked garbling for disjunctive zero-knowledge proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 569–598. Springer, Cham, May 2020.

[HKE12] Yan Huang, Jonathan Katz, and David Evans. Quid-Pro-Quo-tocols: Strengthening semi-honest protocols with dual execution. In *2012 IEEE Symposium on Security and Privacy*, pages 272–284. IEEE Computer Society Press, May 2012.

[HKM18] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving LWE. *Designs, Codes, and Cryptography*, 86(1):55–83, 2018.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International algorithmic number theory symposium*, pages 267–288. Springer, 1998.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Berlin, Heidelberg, August 2011.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, May 2008.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, November 2013.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206. ACM Press, April 1983.

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Cham, April / May 2017.

[Klu22] Kamil Kluczniak. NTRU-v-um: Secure fully homomorphic encryption from NTRU with small modulus. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1783–1797. ACM Press, November 2022.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015.

[KMR14] Vladimir Kolesnikov, Payman Mohassel, and Mike Rosulek. FleXOR: Flexible garbling for XOR gates that beats free-XOR. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 440–457. Springer, Berlin, Heidelberg, August 2014.

[KP17] Yashvanth Kondi and Arpita Patra. Privacy-free garbled circuits for formulas: Size zero and information-theoretic. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 188–222. Springer, Cham, August 2017.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Berlin, Heidelberg, July 2008.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014.

[LP15] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *Journal of Cryptology*, 28(2):312–350, April 2015.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Berlin, Heidelberg, May / June 2010.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 351–358. ACM Press, June 2010.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the First ACM Conference on Electronic Commerce (EC-99)*, pages 129–139. ACM, 1999.

[Pei16] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, 2016.

[PS21] Alice Pellet-Mary and Damien Stehlé. On the hardness of the NTRU problem. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 3–35. Springer, Cham, December 2021.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 250–267. Springer, Berlin, Heidelberg, December 2009.

[RR21] Mike Rosulek and Lawrence Roy. Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 94–124, Virtual Event, August 2021. Springer, Cham.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Berlin, Heidelberg, December 2009.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Berlin, Heidelberg, April 2015.

A Correctness Analysis and Security Proof of Our Garbling Scheme

A.1 Correctness Analysis

In this section, we prove correctness of our garbling scheme shown in Figure 3 via a hybrid argument. Let ϵ_0 denote the probability that the correctness holds. First, for each gate $g \in \mathsf{GateIndex}(f)$, we replace $\mathsf{PRF}(K, g)$ with a uniform element in $\mathcal{R}_p$. After the replacement, we use ϵ_1 to denote the probability that the correctness holds, and will prove $\epsilon_1 = 1 - \mathsf{negl}(\lambda)$. Following prior work [BKS19], if the probabilities ϵ_0 and ϵ_1 differ noticeably, then we can construct an adversary $\mathcal{A}$, who breaks the pseudorandomness of PRF outputs with noticeable probability. Specifically, $\mathcal{A}$ executes the Garble algorithm, except that instead of evaluating $\mathsf{PRF}(K, g)$, it queries its PRF oracle. If the correctness does not hold, $\mathcal{A}$ outputs *real*. Otherwise, it outputs *random*. Therefore, the difference $|\epsilon_0 - \epsilon_1|$ is bounded by the successful probability attacking the pseudorandomness of PRF outputs. From $\epsilon_1 = 1 - \mathsf{negl}(\lambda)$, we obtain $\epsilon_0 = 1 - \mathsf{negl}(\lambda)$. Below, we replace $\mathsf{PRF}(K, g)$ with a uniform element $R_g \in \mathcal{R}_p$ for each gate $g \in \mathsf{GateIndex}(f)$, and prove $\epsilon_1 = 1 - \mathsf{negl}(\lambda)$. In this case, we have that $W_g^{\pi_g}$ is uniform in $\mathcal{R}_p$ for each gate $g \in \mathsf{GateIndex}(f)$.

Second, we show that the Garble algorithm correctly computes each wire mask, i.e., $\mathsf{LSB}(W_g^0) = \pi_g$ for each $g \in \mathsf{GateIndex}(f)$, where W_g^0 is the 0-label. By definition, for each input wire $i \in \mathsf{Inputs}(f)$, it trivially holds that $\mathsf{LSB}(W_i^0) = \pi_i$. For each gate $g \in \mathsf{GateIndex}(f)$, according to the definitions of $\widetilde{\mathcal{L}}_{g,0,0}$ and $\mathcal{L}_{g,0,0}$, we have

$$\widetilde{\mathcal{L}}_{g,0,0} = \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,3}) + R_g\,,$$
$$\mathcal{L}_{g,0,0} = \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + R_g\,,$$

where $\tau_{g,1} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$ and $\tau_{g,3} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$. The ciphertext triples $(\tau_{g,1}, \tau_{g,2}, \tau_{g,3})$ and $(\widetilde{\tau}_{g,1}, \widetilde{\tau}_{g,2}, \widetilde{\tau}_{g,3})$ exhibit two possible relationships: for $j \in [3]$, either $\tau_{g,j} = \widetilde{\tau}_{g,j}$ or $\tau_{g,j} = -\widetilde{\tau}_{g,j}$, depending on the value of v_g. When $v_g = 0$, we have $\tau_{g,j} = \widetilde{\tau}_{g,j}$ and thus $\mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) = \mathsf{LSB}(\mathcal{L}_{g,0,0})$. When $v_g = 1$, we have $\tau_{g,j} = -\widetilde{\tau}_{g,j}$, and in the following, we prove that $\mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) = \mathsf{LSB}(\mathcal{L}_{g,0,0})$ holds except with probability $\mathsf{negl}(\lambda)$. Based on the distributed decryption property, we have

$$\begin{aligned}&\mathsf{LSB}\left(\mathsf{Dec}(W_\alpha^{\pi_\alpha}, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,3}) + R_g\right)\\ =&\mathsf{LSB}\left(-\mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) - \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) - \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + R_g\right)\,.\end{aligned}$$

Using the fact that for any $X, Y \in \mathcal{R}_p$, $\mathsf{LSB}(X + Y) = \mathsf{LSB}(X) \oplus \mathsf{LSB}(Y)$ and $\mathsf{LSB}(-X) = \mathsf{LSB}(X)$ when p is even, we have $\mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) = \mathsf{LSB}(\mathcal{L}_{g,0,0})$. Therefore, $\pi_g = \mathsf{LSB}(\mathcal{L}_{g,0,0}) \oplus z_{g,0,0} = \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) \oplus z_{g,0,0}$. Furthermore, since the garbler computes W_g^0 as $\mathcal{L}_{g,0,0} - (-1)^{\pi_g} \cdot z_{g,0,0} \cdot \Delta$, we obtain $\mathsf{LSB}(W_g^0) = (\pi_g \oplus z_{g,0,0}) - (-1)^{\pi_g} \cdot z_{g,0,0}$. If $z_{g,0,0} = 0$, then it is obvious that $\mathsf{LSB}(W_g^0) = \pi_g$. If $z_{g,0,0} = 1$, then $\mathsf{LSB}(W_g^0) = (\pi_g \oplus 1) - (-1)^{\pi_g} = 1 - \pi_g - (-1)^{\pi_g} = \pi_g$.

Finally, we show that the Eval algorithm produces the label $W_g = W_g^0 + (-1)^{\pi_g} \cdot x_g \cdot \Delta$ for each gate $g \in \mathsf{GateIndex}(f)$ where $x_g \in \{0, 1\}$ is the real value on the output wire of the gate g. We prove it using an induction, and it is obvious for input wires, i.e., $W_i = W_i^0 + (-1)^{\pi_i} \cdot x_i \cdot \Delta$

for each $i \in \mathsf{Inputs}(f)$. For each $g \in \mathsf{GateIndex}(f)$ in topological order, we give the following analysis. Based on the message homomorphism property, for each $g \in \mathsf{GateIndex}(f)$, we have $\widehat{\tau}_g = \mathsf{Enc}(\Delta, \widehat{r}_g)$ and $\widetilde{\tau}_g = \mathsf{Enc}(\Delta, \pi_g)$ and thus for each $(i,j) \in \{0,1\}^2$, the ciphertext $\widetilde{\tau}_{g,i,j}$ satisfies

$$\begin{aligned}\mathsf{Dec}(\Delta, \widetilde{\tau}_{g,i,j}) &= \mathsf{Dec}\left(\Delta, (1-2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)\right) \\ &= (1-2\widehat{r}_g) \cdot G(g, \pi_\alpha \oplus i, \pi_\beta \oplus j) \cdot \Delta = (-1)^{\widehat{r}_g} \cdot z_{g,i,j} \cdot \Delta\,,\end{aligned}$$

where $z_{g,i,j} = G(g, \pi_\alpha \oplus i, \pi_\beta \oplus j)$ for $i,j \in \{0,1\}$ represent the truth table for the gate g. Then, the ciphertexts $\widetilde{\tau}_{g,1} = \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} = \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} = \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$ are computed such that

$$\begin{aligned}\mathsf{Dec}(\Delta, \widetilde{\tau}_{g,1}) &= (-1)^{\widehat{r}_g} \cdot (z_{g,1,0} - z_{g,0,0}) \cdot \Delta \\ \mathsf{Dec}(\Delta, \widetilde{\tau}_{g,2}) &= (-1)^{\widehat{r}_g} \cdot (z_{g,0,1} - z_{g,0,0}) \cdot \Delta \\ \mathsf{Dec}(\Delta, \widetilde{\tau}_{g,3}) &= (-1)^{\widehat{r}_g} \cdot (z_{g,0,0} + z_{g,1,1} - z_{g,0,1} - z_{g,1,0}) \cdot \Delta\,.\end{aligned}$$

For each gate $g \in \mathsf{GateIndex}(f)$, according to $v_g = \widehat{r}_g \oplus \pi_g$ along with $\tau_{g,1} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$ and $\tau_{g,3} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$, we obtain

$$\begin{aligned}\mathsf{Dec}(\Delta, \tau_{g,1}) &= (-1)^{\pi_g} \cdot (z_{g,1,0} - z_{g,0,0}) \cdot \Delta \\ \mathsf{Dec}(\Delta, \tau_{g,2}) &= (-1)^{\pi_g} \cdot (z_{g,0,1} - z_{g,0,0}) \cdot \Delta \\ \mathsf{Dec}(\Delta, \tau_{g,3}) &= (-1)^{\pi_g} \cdot (z_{g,0,0} + z_{g,1,1} - z_{g,0,1} - z_{g,1,0}) \cdot \Delta\,.\end{aligned}$$

For each gate $g \in \mathsf{GateIndex}(f)$, let $(s_\alpha, s_\beta) = (\mathsf{LSB}(W_\alpha), \mathsf{LSB}(W_\beta))$ for a pair of input wires $(\alpha, \beta) \leftarrow \mathsf{GateInputs}(f, g)$, where (W_α, W_β) is a pair of input labels. According to the induction, we have $W_\alpha = W_\alpha^0 + (-1)^{\pi_\alpha} \cdot x_\alpha \cdot \Delta$ and $W_\beta = W_\beta^0 + (-1)^{\pi_\beta} \cdot x_\beta \cdot \Delta$. Furthermore, we know $s_\alpha = x_\alpha \oplus \pi_\alpha$ and $s_\beta = x_\beta \oplus \pi_\beta$, and also rewrite $W_\alpha = W_\alpha^{\pi_\alpha} + s_\alpha \cdot \Delta$ and $W_\beta = W_\beta^{\pi_\beta} + s_\beta \cdot \Delta$. We show that there exists a 0-label W_g^0, such that $W_g = W_g^0 + (-1)^{\pi_g} \cdot x_g \cdot \Delta$. Based on the distributed decryption property, with probability $1 - \mathsf{negl}(\lambda)$, we have

$$\begin{aligned}W_g &= \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + R_g \\ &= \mathsf{Dec}(W_\alpha^{\pi_\alpha} + s_\alpha \cdot \Delta, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta} + s_\beta \cdot \Delta, \tau_{g,2}) \\ &\quad + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha} + s_\alpha \cdot \Delta, W_\beta^{\pi_\beta} + s_\beta \cdot \Delta, \tau_{g,3}) + R_g \\ &= \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + R_g \\ &\quad + s_\alpha \cdot \mathsf{Dec}(\Delta, \tau_{g,1}) + s_\beta \cdot \mathsf{Dec}(\Delta, \tau_{g,2}) + s_\alpha s_\beta \cdot \mathsf{Dec}(\Delta, \tau_{g,3}) \\ &= \mathcal{L}_{g,0,0} + (-1)^{\pi_g} \cdot (x_g - z_{g,0,0}) \cdot \Delta\,.\end{aligned}$$

Computing $W_g^0 = \mathcal{L}_{g,0,0} - (-1)^{\pi_g} \cdot z_{g,0,0} \cdot \Delta$, we obtain $W_g = W_g^0 + (-1)^{\pi_g} \cdot x_g \cdot \Delta$ for a real value x_g on the output wire of the gate g.

From the above analysis, for each $i \in \mathsf{Ouputs}(f)$, we have $Y_i = W_i^0 + (-1)^{\pi_i} \cdot y_i \cdot \Delta$ evaluated by $\mathsf{Eval}(F, X)$ and the decoding information $d_i = \pi_i = \mathsf{LSB}(W_i^0)$, where $y = f(x)$. Therefore, $\mathsf{Decode}(d, Y)$ outputs $\mathsf{LSB}(Y_i) \oplus \pi_i = (\pi_i \oplus y_i) \oplus \pi_i = y_i$ for each $i \in \mathsf{Ouputs}(f)$, which completes the correctness proof.

$\mathcal{S}(1^\lambda, f, f(x))$: Simulate the Garble and Encode algorithms as follows:

1. Run $(\mathsf{params}, \Delta) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ with $\mathsf{LSB}(\Delta) = 1$, where Δ is the secret key. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ as a PRF key.
2. For each $i \in [\lambda]$, run $\sigma_i \leftarrow \mathsf{Enc}(\Delta, 0)$, and then set $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$.
3. Homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N)$, where $N = |\mathsf{WireIndex}(f)|$.
4. For each $i \in \mathsf{Inputs}(f)$, sample a garbled label $W_i \xleftarrow{\$} \mathcal{R}_p$ and a random bit $v_i \xleftarrow{\$} \{0,1\}$, and homomorphically compute an HE ciphertext $\widetilde{\tau}_i := v_i \oplus \widehat{\tau}_i$.
5. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, and then do the following:
   (a) For each $i, j \in \{0,1\}$, homomorphically compute an HE ciphertext $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$.
   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.
   (c) Sample $v_g \xleftarrow{\$} \{0,1\}$, and compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$, $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ and $W_g \leftarrow \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + \mathsf{PRF}(K, g)$. Then, homomorphically compute an HE ciphertext $\widetilde{\tau}_g := v_g \oplus \widehat{\tau}_g$.
6. For each $i \in \mathsf{Inputs}(f)$, set $X_i := W_i$. For each $i \in \mathsf{Ouputs}(f)$, compute $d_i := \mathsf{LSB}(W_i) \oplus y_i$ where $(y_i)_{i \in \mathsf{Ouputs}(f)}$ constitute the circuit output $f(x)$.
7. Output a garbled circuit $F = \big(\mathsf{params}, K, \{v_i\}_{i \in \mathsf{WireIndex}(f)}, \sigma\big)$, a set of input labels $X = \{X_i\}_{i \in \mathsf{Inputs}(f)}$ and a decoding information $d = \{d_i\}_{i \in \mathsf{Ouputs}(f)}$.

Figure 4: The construction of simulator $\mathcal{S}$ for privacy.

A.2 Security Proof

Theorem 3 (Theorem 1, restated)**.** *Our garbling scheme (Figure 3) satisfies both privacy (Definition 2) and obliviousness (Definition 3), provided that the SWHE scheme with distributed decryption is CPA secure,* PRG *is a pseudo-random generator, and* PRF *is a pseudo-random function.*

Proof. The proof for obliviousness is identical to that for privacy, with the exception that the simulator does not receive the circuit output $y = f(x)$ and does not need to compute the decoding information d. Therefore, we focus on the proof for privacy, and construct a probabilistic polynomial time (PPT) simulator shown in Figure 4. We use a sequence of hybrids to establish the computational indistinguishability between the real world and the ideal world.

Hybrid 0. This is the real world and denoted by $\mathcal{G}_0$. This hybrid generates the transcript (F, X, d) as shown in Figure 3.

Hybrid 1. This hybrid, denoted by $\mathcal{G}_1$, is the same as $\mathcal{G}_0$, except that using the Garble and Encode algorithms described in Figure 5 to generate a transcript (F, X, d). In Figure 5, the steps 4, 5 and 6

$\mathcal{G}_1(1^\lambda, f, x)$: Simulate the Garble and Encode algorithms as follows:

1. Run $(\mathsf{params}, \Delta) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ with $\mathsf{LSB}(\Delta) = 1$, where Δ is the secret key. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ as a PRF key.

2. For each $i \in [\lambda]$, sample $s_i \xleftarrow{\$} \{0,1\}$, run $\sigma_i \leftarrow \mathsf{Enc}(\Delta, s_i)$, and then set $s = (s_1, \cdots, s_\lambda)$ and $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$.

3. Compute $(\widehat{r}_1, \cdots, \widehat{r}_N) \leftarrow \mathsf{PRG}(s)$ with $\widehat{r}_i \in \{0,1\}$ and homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N) = (\mathsf{Enc}(\Delta, \widehat{r}_1), \cdots, \mathsf{Enc}(\Delta, \widehat{r}_N))$, where $N = |\mathsf{WireIndex}(f)|$.

4. For each $i \in \mathsf{Inputs}(f)$, sample a garbled label $W_i \xleftarrow{\$} \mathcal{R}_p$ and set a wire mask $\pi_i := \mathsf{LSB}(W_i) \oplus x_i$. Then, for each $i \in \mathsf{Inputs}(f)$, set $v_i := \widehat{r}_i \oplus \pi_i$ and homomorphically compute $\widetilde{\tau}_i := v_i \oplus \widehat{\tau}_i$.

5. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, compute the real value $x_g \in \{0,1\}$ with (f, x), and then do the following:

   (a) For each $i, j \in \{0,1\}$, homomorphically compute an HE ciphertext $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$.

   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.

   (c) Compute $\widetilde{W}_g \leftarrow \mathsf{Dec}(W_\alpha, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta, \widetilde{\tau}_{g,2}) + \widetilde{\mathsf{Dec}}(W_\alpha, W_\beta, \widetilde{\tau}_{g,3}) + \mathsf{PRF}(K, g)$. Then, compute a wire mask $\pi_g := \mathsf{LSB}(\widetilde{W}_g) \oplus x_g$, and set $v_g := \widehat{r}_g \oplus \pi_g$.

   (d) Compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$, $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ and compute $W_g \leftarrow \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widetilde{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + \mathsf{PRF}(K, g)$. Then, homomorphically compute $v_g \oplus \widehat{\tau}_g$ to obtain an HE ciphertext $\widetilde{\tau}_g$.

6. For each $i \in \mathsf{Inputs}(f)$, set $X_i := W_i$. For each $i \in \mathsf{Ouputs}(f)$, compute $y = f(x)$ and $d_i := \mathsf{LSB}(W_i) \oplus y_i$. .

7. Output a garbled circuit $F = \big(\mathsf{params}, K, \{v_i\}_{i \in \mathsf{WireIndex}(f)}, \sigma\big)$, a set of input labels $X = \{X_i\}_{i \in \mathsf{Inputs}(f)}$ and a decoding information $d = \{d_i\}_{i \in \mathsf{Ouputs}(f)}$.

Figure 5: The hybrid $\mathcal{G}_1$ used in the proof, where the differences are marked in blue compared to the real world $\mathcal{G}_0$.

change the computation of all labels, all wire masks and the decoding information to eliminate the dependence of secret key Δ. This hybrid still knows the input x.

The differences between $\mathcal{G}_0$ and $\mathcal{G}_1$ are listed as follows:

1. For each $i \in \mathsf{Inputs}(f)$, X_i is sampled uniformly from $\mathcal{R}_p$ in $\mathcal{G}_1$, while $X_i = W_i^0 + (-1)^{\pi_i} \cdot x_i \cdot \Delta$ for a uniform element $W_i^0 \in \mathcal{R}_p$ in $\mathcal{G}_0$.

2. For each $i \in \mathsf{Inputs}(f)$, the wire mask $\pi_i = \mathsf{LSB}(W_i) \oplus x_i$ in $\mathcal{G}_1$, while $\pi_i = \mathsf{LSB}(W_i^0)$ in $\mathcal{G}_0$. Here, both W_i and W_i^0 are uniform in $\mathcal{R}_p$.

$\mathcal{G}_2(1^\lambda, f, x)$ // $\boxed{\mathcal{G}_3(1^\lambda, f, x)}$: Simulate the Garble and Encode algorithms as follows:

1. Run $(\mathsf{params}, \Delta) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ with $\mathsf{LSB}(\Delta) = 1$, where Δ is the secret key. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ as a PRF key.

2. For each $i \in [\lambda]$, sample $s_i \xleftarrow{\$} \{0,1\}$ and run $\sigma_i \leftarrow \mathsf{Enc}(\Delta, 0)$, and then set $s = (s_1, \cdots, s_\lambda)$ and $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$.

3. Compute $(\widehat{r}_1, \cdots, \widehat{r}_N) \leftarrow \mathsf{PRG}(s)$. // $\boxed{\text{Sample } \widehat{r}_i \xleftarrow{\$} \{0,1\} \text{ for each } i \in [N].}$
Homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N)$, where $N = |\mathsf{WireIndex}(f)|$.

4. For each $i \in \mathsf{Inputs}(f)$, sample a garbled label $W_i \xleftarrow{\$} \mathcal{R}_p$ and set a wire mask $\pi_i := \mathsf{LSB}(W_i) \oplus x_i$. Then, for each $i \in \mathsf{Inputs}(f)$, set $v_i := \widehat{r}_i \oplus \pi_i$ and homomorphically compute $\widetilde{\tau}_i := v_i \oplus \widehat{\tau}_i$.

5. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, compute the real value $x_g \in \{0,1\}$ with (f, x), and then do the following:
   (a) For each $i, j \in \{0,1\}$, homomorphically compute an HE ciphertext $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$.
   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.
   (c) Compute $\widetilde{W}_g \leftarrow \mathsf{Dec}(W_\alpha, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta, \widetilde{\tau}_{g,2}) + \widetilde{\mathsf{Dec}}(W_\alpha, W_\beta, \widetilde{\tau}_{g,3}) + \mathsf{PRF}(K, g)$. Then, compute a wire mask $\pi_g := \mathsf{LSB}(\widetilde{W}_g) \oplus x_g$, and set $v_g := \widehat{r}_g \oplus \pi_g$.
   (d) Compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$, $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ and $W_g \leftarrow \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + \mathsf{PRF}(K, g)$. Then, homomorphically compute $v_g \oplus \widehat{\tau}_g$ to obtain an HE ciphertext $\widetilde{\tau}_g$.

6. For each $i \in \mathsf{Inputs}(f)$, set $X_i := W_i$. For each $i \in \mathsf{Ouputs}(f)$, compute $y = f(x)$ and $d_i := \mathsf{LSB}(W_i) \oplus y_i$.

7. Output a garbled circuit $F = \big(\mathsf{params}, K, \{v_i\}_{i \in \mathsf{WireIndex}(f)}, \sigma\big)$, a set of input labels $X = \{X_i\}_{i \in \mathsf{Inputs}(f)}$ and a decoding information $d = \{d_i\}_{i \in \mathsf{Ouputs}(f)}$.

Figure 6: The hybrids $\mathcal{G}_2$ and $\mathcal{G}_3$ used in the proof, where the differences are marked in blue compared to $\mathcal{G}_1$. The difference between $\mathcal{G}_2$ and $\mathcal{G}_3$ is marked by a rectangular box.

3. For each $g \in \mathsf{GateIndex}(f)$, the wire mask $\pi_g = \mathsf{LSB}(\widetilde{W}_g) \oplus x_g$ for a real value x_g in $\mathcal{G}_1$, while $\pi_g = \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) \oplus z_{g,0,0}$ with $z_{g,0,0} = G(g, \pi_\alpha, \pi_\beta)$ in $\mathcal{G}_0$.

4. For each $g \in \mathsf{GateIndex}(f)$, the label W_g, associated with real value x_g, is computed in $\mathcal{G}_1$, while the 0-label W_g^0 is computed in $\mathcal{G}_0$.

5. For each $i \in \mathsf{Ouputs}(f)$, the decoding information $d_i = \mathsf{LSB}(W_i) \oplus y_i$ in $\mathcal{G}_1$, while d_i is directly set as π_i in $\mathcal{G}_0$.

First of all, we analyze the first two differences. In $\mathcal{G}_0$, for each $i \in \mathsf{Inputs}(f)$, $X_i = W_i^0 + (-1)^{\pi_i} \cdot x_i \cdot \Delta$ is uniform in $\mathcal{R}_p$, since W_i^0 is sampled uniformly from $\mathcal{R}_p$. In $\mathcal{G}_1$, for each $i \in \mathsf{Inputs}(f)$, $X_i = W_i \in \mathcal{R}_p$ is directly sampled at random. Thus, for each $i \in \mathsf{Inputs}(f)$, X_i in $\mathcal{G}_0$ and $\mathcal{G}_1$ has the identical distribution. For each $i \in \mathsf{Inputs}(f)$, due to the uniformity of W_i (in $\mathcal{G}_1$) and W_i^0 (in $\mathcal{G}_0$), the wire mask π_i has the identical distribution in two hybrids. Furthermore, W_i in $\mathcal{G}_1$ plays the role of the label on the real value x_i, and has the identical distribution as X_i output by Encode in $\mathcal{G}_0$. In other words, for hybrid $\mathcal{G}_1$, there exists a uniformly random 0-label $W_i^0 \in \mathcal{R}_p$ such that $W_i = W_i^0 + (-1)^{\pi_i} \cdot x_i \cdot \Delta$ and $\mathsf{LSB}(W_i^0) = \pi_i$.

In the following, we analyze the third and fourth differences by an induction. From the above analysis, we obtain that two hybrids have no difference for each input wire $i \in \mathsf{Inputs}(f)$. For each $g \in \mathsf{GateIndex}(f)$, according to the induction, we obtain that both (W_α, π_α) and (W_β, π_β) have the identical distribution in two hybrids for a pair of gate-input wires $(\alpha, \beta) = \mathsf{GateInputs}(f, g)$. In addition, there exists two 0-labels W_α^0 and W_β^0 such that $W_\alpha = W_\alpha^0 + (-1)^{\pi_\alpha} \cdot x_\alpha \cdot \Delta$, $W_\beta = W_\beta^0 + (-1)^{\pi_\beta} \cdot x_\beta \cdot \Delta$, $\mathsf{LSB}(W_\alpha^0) = \pi_\alpha$ and $\mathsf{LSB}(W_\beta^0) = \pi_\beta$. Note that the identical distribution of wire masks π_α and π_β implies that the ciphertexts $\widetilde{\tau}_{g,1}, \widetilde{\tau}_{g,2}, \widetilde{\tau}_{g,3}$ has the same distribution in two hybrids. We rewrite $W_\alpha = W_\alpha^{\pi_\alpha} + s_\alpha \cdot \Delta$ and $W_\beta = W_\beta^{\pi_\beta} + s_\beta \cdot \Delta$, where $s_\alpha = \pi_\alpha \oplus x_\alpha$ and $s_\beta = \pi_\beta \oplus x_\beta$. Similar to the correctness analysis shown in Appendix A.1 (requiring the pseudorandomness of PRF outputs), we have $\widetilde{W}_g = \widetilde{\mathcal{L}}_{g,0,0} + (x_g - z_{g,0,0}) \cdot \Delta$, except with probability $\mathsf{negl}(\lambda)$, where $\widetilde{\mathcal{L}}_{g,0,0} = \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,3}) + \mathsf{PRF}(K, g)$.

From $\pi_g = \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) \oplus z_{g,0,0}$, we have $\mathsf{LSB}(\widetilde{W}_g) = x_g \oplus \pi_g$, and vice versa. Therefore, except with probability $\mathsf{negl}(\lambda)$, the wire mask π_g in $\mathcal{G}_1$ has the same distribution as that in $\mathcal{G}_0$, and so is $v_g = \widehat{r}_g \oplus \pi_g$. Since $\tau_{g,1} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$, $\tau_{g,3} = (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ have the same distribution in two hybrids, we have that $W_g = \mathsf{Dec}(W_\alpha, \tau_{g,1}) + \mathsf{Dec}(W_\beta, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha, W_\beta, \tau_{g,3}) + \mathsf{PRF}(K, g)$ in $\mathcal{G}_1$ has the identical distribution as that in $\mathcal{G}_0$. According to the correctness analysis shown in Appendix A.1, with probability $1 - \mathsf{negl}(\lambda)$, we have $W_g = \mathcal{L}_{g,0,0} + (-1)^{\pi_g} \cdot (x_g - z_{g,0,0}) \cdot \Delta$, where $\mathcal{L}_{g,0,0} = \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + \mathsf{PRF}(K, g)$. By defining $W_g^0 = \mathcal{L}_{g,0,0} - (-1)^{\pi_g} \cdot z_{g,0,0} \cdot \Delta$ in $\mathcal{G}_1$, we obtain $W_g = W_g^0 + (-1)^{\pi_g} \cdot x_g \cdot \Delta$, which has the same distribution as in $\mathcal{G}_0$. Based on the correctness analysis, we know $\mathsf{LSB}(\mathcal{L}_{g,0,0}) = \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0})$. From $\mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) = \pi_g \oplus z_{g,0,0}$ and $\mathsf{LSB}(W_g^0) = \mathsf{LSB}(\mathcal{L}_{g,0,0}) - (-1)^{\pi_g} \cdot z_{g,0,0} \mod 2$, we have $\mathsf{LSB}(W_g^0) = \pi_g$.

Finally, we analyze the fifth difference. For each $i \in \mathsf{Ouputs}(f)$, except with probability $\mathsf{negl}(\lambda)$, $\mathsf{LSB}(W_i)$ is the XOR of real value y_i and π_i, and thus $d_i = \mathsf{LSB}(W_i) \oplus y_i$ in $\mathcal{G}_1$ has the same distribution as π_i in $\mathcal{G}_0$, as the wire mask π_i has the identical distribution in two hybrids. In conclusion, $\mathcal{G}_1$ is indistinguishable from $\mathcal{G}_0$.

Hybrid 2. This hybrid, denoted by $\mathcal{G}_2$, is the same as $\mathcal{G}_1$, except that each ciphertext σ_i is replaced with a fresh ciphertext $\mathsf{Enc}(\Delta, 0)$, as shown in Figure 6. Even if all ciphertexts $\sigma_1, \ldots, \sigma_\lambda$ encrypt zero, the tuple (F, X, d) is still computed as a circuit output $f(x)$.

The only difference between two hybrids $\mathcal{G}_1$ and $\mathcal{G}_2$ is the generation of ciphertexts $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$. In $\mathcal{G}_1$, each ciphertext σ_i is generated using the secret key Δ to encrypt a random bit s_i. In $\mathcal{G}_2$, each σ_i encrypts a fixed bit 0. We can bound the difference between $\mathcal{G}_1$ and $\mathcal{G}_2$ by a reduction to the CPA security of the SWHE scheme with distributed decryption. It is well-known that a single challenge ciphertext is polynomially equivalent to multiple challenge ciphertexts for CPA security. For the sake of simplicity, we use the multi-challenge version of CPA security to

$\underline{\mathsf{Exp}^{\mathsf{auth}}_{\mathcal{A}}(1^\lambda, f, x)}$

1. $(F, e, d) \leftarrow \mathsf{Garble}(1^\lambda, f)$; $X \leftarrow \mathsf{Encode}(e, x)$; $\widehat{Y} \leftarrow \mathcal{A}(F, X, d)$;
2. If $\mathsf{Decode}(d, \widehat{Y}) \neq \perp$ and $\widehat{Y} \neq \mathsf{Eval}(F, X)$, then output 1, else output 0.

Figure 7: Experiment for authenticity.

construct the reduction.

Specifically, let $\mathcal{A}$ be a PPT adversary who attempts to distinguish (F, X, d) in $\mathcal{G}_2$ from that in $\mathcal{G}_1$. We construct a PPT algorithm $\mathcal{B}$, who is given params^*, to attack the CPA security. Firstly, $\mathcal{B}$ samples $s_i \xleftarrow{\$} \{0,1\}$ for $i \in [\lambda]$, sends $(s_i, 0)$ for each $i \in [\lambda]$ to its CPA experiment, then obtains λ challenge ciphertexts $\sigma_1^*, \cdots, \sigma_\lambda^*$. Then, $\mathcal{B}$ generates (F^*, X^*, d^*) just as in $\mathcal{G}_1$, except that using $\sigma^* = (\sigma_1^*, \cdots, \sigma_\lambda^*)$ as the ciphertexts on random bits $s_1, \ldots, s_\lambda$ and params^* as the set of public parameters. Then, $\mathcal{B}$ sends (F^*, X^*, d^*) to $\mathcal{A}$, and outputs a bit according to the output of $\mathcal{A}$. If $\sigma_i^* = \mathsf{Enc}(\Delta, s_i)$ for each $i \in [\lambda]$, (F^*, X^*, d^*) has the same distribution as that in $\mathcal{G}_1$; otherwise (i.e., $\sigma_i^* = \mathsf{Enc}(\Delta, 0)$ for each $i \in [\lambda]$), (F^*, X^*, d^*) has the identical distribution as that in $\mathcal{G}_2$. Therefore, if $\mathcal{A}$ distinguishes $\mathcal{G}_1$ from $\mathcal{G}_2$ with probability ϵ, then $\mathcal{B}$ breaks the CPA security of the SWHE scheme with the same probability.

Hybrid 3. This hybrid, denoted by $\mathcal{G}_3$, is the same as $\mathcal{G}_2$, except that each bit $\widehat{r}_i$ with $i \in [N]$ is replaced with a random bit, as shown in Figure 6.

The only difference between two hybrids $\mathcal{G}_2$ and $\mathcal{G}_3$ is: $\mathcal{G}_2$ generates the bits $\widehat{r}_1, \ldots, \widehat{r}_N$ by $\mathsf{PRG}(s)$, while $\mathcal{G}_3$ samples them uniformly at random. Note that the seed $s = (s_1 \cdots, s_\lambda)$ is now uniform and independent of ciphertexts $\sigma_1, \cdots, \sigma_\lambda$. Thus, it is straightforward to bound the difference between $\mathcal{G}_2$ and $\mathcal{G}_3$ by the pseudorandomness of the output of PRG.

Hybrid 4. This hybrid, denoted by $\mathcal{G}_4$, is the same as $\mathcal{G}_3$, except that for each $g \in \mathsf{WireIndex}(f)$, replacing v_g with a random bit and removing the computation of π_g and $\widetilde{W}_g$. Specifically, the hybrid $\mathcal{G}_4$ generates the transcript (F, X, d), as shown in Figure 4.

When $v_g \in \{0,1\}$ is sampled at random, the computation of π_g and $\widetilde{W}_g$ is redundant and can be removed. Thus, the difference between $\mathcal{G}_3$ and $\mathcal{G}_4$ lies in the generation of $\{v_g\}$. In $\mathcal{G}_3$, each $v_g \in \{0,1\}$ is computed as $v_g = \pi_g \oplus \widehat{r}_g$ for a uniform bit $\widehat{r}_g \in \{0,1\}$. In $\mathcal{G}_4$, each $v_g \in \{0,1\}$ is sampled uniformly. Therefore, $\mathcal{G}_3$ has the same distribution as $\mathcal{G}_4$.

Hybrid $\mathcal{G}_4$ behaves exactly as in the ideal world, which completes the proof. □

B Our Garbling Scheme for Authenticity

In this section, we show how to modify our garbling scheme from privacy and obliviousness to authenticity, following the standard approach [HKE12, ZRE15]. We first recall the security definition of garbling schemes for authenticity. Then, we describe the detailed modifications of our garbling scheme to realize authenticity.

B.1 Definition for Authenticity

Following prior works [GGP10, BHR12, RR21], we give the definition of the authenticity property. The authenticity allows the adversary to learn the circuit f and input x, i.e., no privacy is guaranteed. However, the authenticity guarantees that the adversary cannot forge a garbled output $\widehat{Y}$ such that $\widehat{Y}$ is successful in being decoded but $\widehat{Y} \neq \mathsf{Eval}(F, X)$, even if it obtains the information (F, X, d). The authenticity property is useful for designing constant-round zero-knowledge proofs, e.g., [JKO13, FNO15, KP17, HK20].

Definition 6 (Authenticity)**.** *For any PPT adversary $\mathcal{A}$, for any function f and input x, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr\left[\mathsf{Exp}^{\mathsf{auth}}_{\mathcal{A}}(1^\lambda, f, x) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathsf{Exp}^{\mathsf{auth}}_{\mathcal{A}}(1^\lambda, f, x)$ is defined in Figure 7.

B.2 Concrete Construction of Garbling for Authenticity

We modify the garbling scheme shown in Figure 3 to realize authenticity. In Figure 8, we describe the modifications of the Garble and Decode algorithms (marked in blue) using a cryptographic hash function $H : \mathcal{R}_q \to \{0,1\}^\lambda$ modeled as a non-programmable random oracle.

Theorem 4. *Our modified scheme (Figure 8) satisfies the authenticity property (Definition 6), provided that the SWHE scheme with distributed decryption is CPA secure,* PRG *is a pseudorandom generator,* PRF *is a pseudorandom function, and H is a non-programmable random oracle.*

Sketch. For the interaction with a PPT adversary $\mathcal{A}$, we run the simulator $\mathcal{S}$ (shown in Figure 4) to generate (F, X, d) and send it to $\mathcal{A}$, where $\mathcal{S}$ changes the computation of the decoding information d, i.e., for each $i \in \mathsf{Ouputs}(f)$, $\mathcal{S}$ computes d_i as follows:

- If $y_i = 0$, then $\mathcal{S}$ sets
$$d_i = \left(H\left(Y_i\right), V_i \xleftarrow{\$} \{0,1\}^\lambda\right)\,.$$
- If $y_i = 1$, then $\mathcal{S}$ computes
$$d_i = \left(V_i \xleftarrow{\$} \{0,1\}^\lambda, H\left(Y_i\right)\right)\,.$$

In the above computation, Y_i is the label on the output wire i computed by $\mathcal{S}$, and $y = f(x)$ known by $\mathcal{S}$. In the non-programmable random oracle model, the probability, that $\mathcal{A}$ finds a ring element $U_i \in \mathcal{R}_p$ such that $H(U_i) = V_i$ for some $i \in \mathsf{Ouputs}(f)$, is negligible in λ. Therefore, the probability, that $\mathcal{A}$ succeeds in violating authenticity, is bounded by $\mathsf{negl}(\lambda)$.

Below, we use a sequence of hybrids to prove that the adversary's view constructed by the above $\mathcal{S}$ is computationally indistinguishable from the real view. Specifically, for each $i \in \mathsf{Ouputs}(f)$, we first replace a random element V_i with $H(Z_i)$ in the first hybrid, where $Z_i = Y^i - (-1)^{y_i} \cdot \Delta$ is another label on the output wire i. According to the construction of $\mathcal{S}$, the adversary cannot learn secret key Δ from the transcript (F, X, d), under the assumption that the SWHE scheme is CPA secure. That is, for each $i \in \mathsf{Ouputs}(f)$, Z_i is kept secret against adversary $\mathcal{A}$. Therefore, for each $i \in \mathsf{Ouputs}(f)$, a uniform element V_i is computationally indistinguishable from $H(Z_i)$ in

$\mathsf{Garble}(1^\lambda, f)$: The garbler executes the garble algorithm as follows:

1. Run $(\mathsf{params}, \Delta) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ with $\mathsf{LSB}(\Delta) = 1$, where Δ is the secret key. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$ as a PRF key.

2. For each $i \in [\lambda]$, sample $s_i \xleftarrow{\$} \{0,1\}$, run $\sigma_i \leftarrow \mathsf{Enc}(\Delta, s_i)$, and then set $s = (s_1, \cdots, s_\lambda)$ and $\sigma = (\sigma_1, \cdots, \sigma_\lambda)$.

3. Compute $(\widehat{r}_1, \cdots, \widehat{r}_N) \leftarrow \mathsf{PRG}(s)$ with $\widehat{r}_i \in \{0,1\}$, and homomorphically compute $\widetilde{\mathsf{PRG}}(\sigma)$ to generate $(\widehat{\tau}_1, \cdots, \widehat{\tau}_N)$ with $\widehat{\tau}_i = \mathsf{Enc}(\Delta, \widehat{r}_i)$ for $i \in [N]$, where $N = |\mathsf{WireIndex}(f)|$.

4. For each $i \in \mathsf{Inputs}(f)$, sample a 0-label $W_i^0 \xleftarrow{\$} \mathcal{R}_p$, set $\pi_i := \mathsf{LSB}(W_i^0)$, and compute a 1-label $W_i^1 := W_i^0 + (-1)^{\pi_i} \cdot \Delta \in \mathcal{R}_p$. Then, for each $i \in \mathsf{Inputs}(f)$, set $v_i := \widehat{r}_i \oplus \pi_i$ and homomorphically compute $v_i \oplus \widehat{\tau}_i$ to obtain $\widetilde{\tau}_i := \mathsf{Enc}(\Delta, \pi_i)$.

5. For each $g \in \mathsf{GateIndex}(f)$ in topological order, set $(\alpha, \beta) \leftarrow \mathsf{Gateinputs}(f, g)$, $W_\alpha^{\pi_\alpha} = W_\alpha^0 - \pi_\alpha \cdot \Delta \in \mathcal{R}_p$ and $W_\beta^{\pi_\beta} = W_\beta^0 - \pi_\beta \cdot \Delta \in \mathcal{R}_p$, then do the following:
   (a) For each $i, j \in \{0,1\}$, compute $z_{g,i,j} = G(g, \pi_\alpha \oplus i, \pi_\beta \oplus j)$ and homomorphically compute $\widetilde{\tau}_{g,i,j} \leftarrow (1 - 2\widehat{\tau}_g) \cdot G(g, \widetilde{\tau}_\alpha \oplus i, \widetilde{\tau}_\beta \oplus j)$, resulting in the ciphertext $\widetilde{\tau}_{g,i,j}$ that encrypts $(-1)^{\widehat{r}_g} \cdot z_{g,i,j}$.
   (b) Homomorphically compute HE ciphertexts $\widetilde{\tau}_{g,1} := \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,0}$, $\widetilde{\tau}_{g,2} := \widetilde{\tau}_{g,0,1} - \widetilde{\tau}_{g,0,0}$ and $\widetilde{\tau}_{g,3} := \widetilde{\tau}_{g,0,0} + \widetilde{\tau}_{g,1,1} - \widetilde{\tau}_{g,1,0} - \widetilde{\tau}_{g,0,1}$.
   (c) Run $\widetilde{\mathcal{L}}_{g,0,0} \leftarrow \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \widetilde{\tau}_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,2}) + \widetilde{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \widetilde{\tau}_{g,3}) + \mathsf{PRF}(K, g)$. Then, compute $\pi_g := \mathsf{LSB}(\widetilde{\mathcal{L}}_{g,0,0}) \oplus z_{g,0,0}$ and $v_g := \widehat{r}_g \oplus \pi_g$.
   (d) Compute $\tau_{g,1} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,1}$, $\tau_{g,2} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,2}$ and $\tau_{g,3} := (-1)^{v_g} \cdot \widetilde{\tau}_{g,3}$ and $\mathcal{L}_{g,0,0} \leftarrow \mathsf{Dec}(W_\alpha^{\pi_\alpha}, \tau_{g,1}) + \mathsf{Dec}(W_\beta^{\pi_\beta}, \tau_{g,2}) + \widehat{\mathsf{Dec}}(W_\alpha^{\pi_\alpha}, W_\beta^{\pi_\beta}, \tau_{g,3}) + \mathsf{PRF}(K, g)$. Then, homomorphically compute $v_g \oplus \widehat{\tau}_g$ to obtain a ciphertext $\widetilde{\tau}_g := \mathsf{Enc}(\Delta, \pi_g)$.
   (e) Compute the 0-label on the output wire $W_g^0 := \mathcal{L}_{g,0,0} - (-1)^{\pi_g} \cdot z_{g,0,0} \cdot \Delta$.

6. For each $i \in \mathsf{Ouputs}(f)$, compute $d_i = (H(W_i^0), H(W_i^1))$.

7. Output a garbled circuit $F = \big(\mathsf{params}, K, \sigma, \{v_i\}_{i \in \mathsf{WireIndex}(f)}\big)$, an encoding information $e = \big(\{W_i^0\}_{i \in \mathsf{Inputs}(f)}, \Delta\big)$ and a decoding information $d = \{d_i\}_{i \in \mathsf{Ouputs}(f)}$.

$\mathsf{Decode}(d, Y)$: The evaluator performs the following steps:

1. For each $i \in \mathsf{Ouputs}(f)$, parse d_i as $(V_{i,0}, V_{i,1})$. If $H(Y_i) = V_{i,0}$, set $y_i := 0$; if $H(Y_i) = V_{i,1}$, set $y_i := 1$; otherwise, abort and output $\perp$.

2. Set $y = \big(y_i\big)_{i \in \mathsf{Ouputs}(f)}$. Then, output y.

Figure 8: Our garbling scheme with authenticity, where the differences are marked in blue compared to the garbling scheme with privacy and obliviousness shown in Figure 3.

the non-programmable random oracle model. The rest of the proof follows the same sequence of hybrids as the proof of Theorem 1. Eventually, we reach the real game. We have already proved that the subsequent hybrids are computationally indistinguishable in Theorem 1. Overall, the adversary's view output by $\mathcal{S}$ is computationally indistinguishable from the real view, which completes the proof. □

C SWHE with Distributed Decryption from NTRU

C.1 NTRU Problem and Lattice Basis Reduction

The NTRU cryptosystem, pioneered by Hoffstein, Pipher, and Silverman [HPS98], has been applied in more advanced cryptographic constructions [LTV12, GGH13a]. The NTRU problem is defined as follows.

Definition 2 (Decisional NTRU [PS21])**.** *Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n+1)$. For dimension $n \in \mathbb{N}$, number of samples $m \in \mathbb{N}$, and modulus $q \in \mathbb{N}$, the $\mathsf{NTRU}(n, m, \mathsf{B}, q)$ problem is to distinguish between the following two distributions:*

$$\left\{h_i \stackrel{def}{=} f_i/g\right\}_{i\in[m]} \text{ and } \left\{u_i \xleftarrow{\$} \mathcal{R}_q\right\}_{i\in[m]},$$

where $f_1, \ldots, f_m, g \leftarrow \mathcal{D}_{\mathsf{B}}$, with that g is invertible in $\mathcal{R}_q$.

Hardness of NTRU problem. A reduction from the decisional NTRU problem to the computational RLWE problem was presented in [SSTX09, LPR10] and detailed in [Pei16, Section 4.4.4]. Pellet-Mary and Stehlé [PS21] have introduced a reduction from the worst-case approximate shortest vector problem over ideal lattices to an average-case version of the computational NTRU problem. Additionally, they gave a reduction from another average-case variant of the computational NTRU problem to the decisional NTRU problem.

Lattice basis reduction and the overstretched regime. Lattice basis reduction algorithms aim to minimize the length and orthogonality of an input basis. In practical applications, the Blockwise Korkine-Zolotare (BKZ) algorithm [SE94] is highly relevant, which is a block-wise generalization of the LLL algorithm [LLL82]. To express the complexity of the BKZ algorithm, we follow the approach outlined in [HKM18] and relate its runtime to the block size β. Within BKZ, an SVP solver is called for a sub-lattice of dimension β. Guillaume Hanrot, Pujol, and Stehlé [HPS11] have demonstrated that by invoking a polynomial number of SVP solver, the BKZ algorithm yields a basis in which the first (i.e., shortest) vector becomes progressively shorter as the β increases. Consequently, the running time of BKZ can be denoted as $T_{\mathsf{BKZ}} = \mathsf{poly}(n) \cdot T_{\mathsf{SVP}}(\beta)$, where $T_{\mathsf{SVP}}(\beta)$ signifies the running time of an SVP-solver in dimension β. Current algorithms exhibit exponential runtime in β, ranging from the earlier $2^{O(\beta^2)}$ [FP83], to $2^{O(\beta \log \beta)}$ [Kan83], and recently to $2^{O(\beta)}$ [MV10] (but the memory complexity of $2^{O(\beta)}$).

More recently, as shown in prior works such as [ABD16, CJL16, KF17, Dv21], lattice reduction attacks on NTRU lattices, particularly for a large modulus q, exhibit a significant deviation from their behavior on (ring-)LWE lattices with equivalent parameters. These works leverage the specific algebraic structure of the NTRU lattice to enhance the effectiveness of lattice reduction attacks. Notably, they have conducted their analyses with larger modulus q and smaller parameter

B, revealing that the NTRU problem is considerably more tractable than previously believed. Especially, their findings indicate that the NTRU problem can be classified as an easy problem when $\log^2(q) \geq \widetilde{\Omega}(n)$, which deviates from the previously established condition of $\log(q) \geq \widetilde{\Omega}(n)$, as detailed in Claim 2. To resist lattice reduction attacks, our NTRU-based SWHE scheme slightly increases the modulus q, keeping it a small superpolynomial. As stated in Claim 2, the NTRU assumption holds for such parameter choice.

Claim 2 ([Dv21]). *The BKZ algorithm solves the* NTRU *problem with a modulus* $q = n^Q$ *and an norm* $\|f_i\|_2 = n^S$ *in time* $2^{O(\beta)}$*, where the block size* $\beta = O\left(\frac{8S}{Q^2+1} \cdot n\right)$ *and* n *is the dimension.*

C.2 GSW-SWHE from NTRU

In the following, we recall the GSW-SWHE scheme [GSW13] under the NTRU assumption. We utilize the operations defined in Section 5, including BitDecomp, $\mathsf{BitDecomp}^{-1}$, Flatten and Powersof2. The NTRU-based GSW-SWHE scheme in the private-key setting consists of the following three algorithms:

- **Key generation.** $\mathsf{Gen}(1^\lambda, 1^L)$ samples a secret key $sk \leftarrow \mathcal{D}_\mathsf{B}$ such that it is invertible in $\mathcal{R}_q$, and outputs sk along with a set of parameters $\mathsf{params} = (n, p, q, \mathsf{B}, N, s)$, where $N = \lfloor \log(q) \rfloor + 1$ and $s = \lfloor \log(q/p) \rfloor$.

- **Encryption.** On input a secret key sk and a message $m \in \mathbb{Z}_p$, $\mathsf{Enc}(sk, m)$ samples $\boldsymbol{e} \leftarrow \mathcal{D}_\mathsf{B}^N$, and outputs a ciphertext

$$\tau := \mathsf{Flatten}\left(m \cdot \mathbf{I}_N + \mathsf{BitDecomp}\left(\left[\boldsymbol{e}/sk\right]\right)\right) ,$$

  where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

- **Decryption.** On input a secret key sk and a ciphertext $\tau \in \mathcal{R}_2^{N \times N}$, $\mathsf{Dec}(sk, \tau)$ sets a vector $\boldsymbol{v}^\mathsf{T} = \mathsf{Powersof2}(sk)$, and then computes

$$\tau \cdot \boldsymbol{v} = \left(m \cdot \mathbf{I}_N + \mathsf{BitDecomp}\left(\left[\boldsymbol{e}/sk\right]\right)\right) \cdot \boldsymbol{v} = m \cdot \boldsymbol{v} + \boldsymbol{e} \approx m \cdot \boldsymbol{v} \,. \tag{6}$$

  Then, it extracts the $(s+1)$-th element of $\tau \cdot \boldsymbol{v}$, denoted as $(\tau \cdot \boldsymbol{v})[s+1]$ and outputs $m = \left\lfloor \frac{(\tau \cdot \boldsymbol{v})[s+1]}{2^s} \right\rceil / sk$.

The NTRU-based GSW scheme supports homomorphic addition and multiplication operations, and the error of the final ciphertext is bounded by $(N+1)^L \cdot n^L \cdot \mathsf{B}$, where L represents the multiplication depth of a circuit. For any $(\mathsf{params}, sk) \leftarrow \mathsf{Gen}(1^\lambda, 1^L)$ along with two messages m_1 and m_2 as well as their ciphertexts $\tau_1 = \mathsf{Enc}(sk, m_1)$ and $\tau_2 = \mathsf{Enc}(sk, m_2)$, the homomorphic addition operation Add and homomorphic multiplication operation Mult are defined as follows:

- $\mathsf{Add}(\tau_1, \tau_2)$: To add ciphertexts $\tau_1, \tau_2 \in \mathcal{R}_2^{N \times N}$, output $\mathsf{Flatten}(\tau_1 + \tau_2)$. The correctness of this operation is obvious.

- $\mathsf{Mult}(\tau_1, \tau_2)$. To multiply ciphertexts $\tau_1, \tau_2 \in \mathcal{R}_2^{N \times N}$, output $\mathsf{Flatten}(\tau_1 \cdot \tau_2)$. To verify the correctness, we decrypt $\mathsf{Mult}(\tau_1, \tau_2)$ as $\mathsf{Mult}(\tau_1, \tau_2) \cdot \boldsymbol{v}$:

$$\mathsf{Mult}(\tau_1, \tau_2) \cdot \boldsymbol{v} = \tau_1 \cdot \tau_2 \cdot \boldsymbol{v} \approx \tau_1 \cdot m_2 \cdot \boldsymbol{v} = m_2 \cdot \tau_1 \cdot \boldsymbol{v} \approx m_1 \cdot m_2 \cdot \boldsymbol{v} \,,$$

  where the approximate equality "$\approx$" is due to Equation (6).

C.3 Our SWHE Scheme with Distributed Decryption from NTRU

We propose a NTRU-based SWHE scheme with distributed decryption via extending the above GSW scheme, and denote the extended scheme as eGSW. The encryption algorithm Enc, homomorphic addition Add and homomorphic multiplication Mult of the eGSW scheme is identical to the NTRU-based GSW scheme, and thus they are omitted. The other algorithms of our NTRU-based eGSW scheme are described as follows:

- **Key generation.** $\mathsf{Gen}(1^\lambda, 1^L)$ samples a secret keys $sk \leftarrow \mathcal{D}_{\mathsf{B}}$ such that $\mathsf{LSB}(sk) = 1$ and sk is invertible in $\mathcal{R}_q$, and then generates a set of public parameters $\mathsf{params} = (n, p, q, \mathsf{B}, N, mk)$ as below:
  - $p \mid q$, $p = \lambda^{\omega(1)}$ is an even, $q/p^2 = \lambda^{\omega(1)} \cdot (N+1)^L \cdot n^L$ and $N = \lfloor \log(q) \rfloor + 1$;
  - Sample $e \leftarrow \mathcal{D}_{\mathsf{B}}$ and compute $mk := (e + q/p)/sk$ over $\mathcal{R}_q$.

  It outputs (params, sk).

- **Decryption.** On input a key $x \in \mathcal{R}$ (it is either the secret key sk or an additive share of sk for our instantiation) and a ciphertext $\tau \in \mathcal{R}_2^{N \times N}$, $\mathsf{Dec}(x, \tau)$ computes over $\mathcal{R}_q$

$$t = \mathsf{BitDecomp}^{-1}\left(\mathsf{BitDecomp}(q/p) \cdot \tau\right) ,$$

  then outputs $\lfloor t \cdot x \rceil_p$. If we use sk to decrypt τ, we have $t \cdot sk \approx (q/p) \cdot m \cdot sk$ and $\lfloor t \cdot sk \rceil_p = m \cdot sk$.

- **Algorithm $\widehat{\mathsf{Dec}}$ for correlated-key distributed decryption.** Given a ciphertext $\tau = \mathsf{Enc}(sk, m)$ as well as two additive secret sharings (sk_0, sk_1) and (sk'_0, sk'_1) such that $\mathsf{LSB}(sk_0) = \mathsf{LSB}(sk'_0) = 0$, $sk_1 = sk_0 + sk$ and $sk'_1 = sk'_0 + sk$ over $\mathcal{R}_p$, for any $i, j \in \{0, 1\}$, $\widehat{\mathsf{Dec}}(sk_i, sk'_j, \tau)$ performs the following steps:
  1. Compute $x := sk_i \cdot sk'_j \cdot mk$ over $\mathcal{R}_q$, where $i = \mathsf{LSB}(sk_i)$ and $j = \mathsf{LSB}(sk'_j)$.
  2. Compute $y := \lfloor x \rceil_p - j \cdot sk_i - i \cdot sk'_j$ over $\mathcal{R}_p$.
  3. Run $z \leftarrow \mathsf{Dec}(y, -\tau)$ and output $z \in \mathcal{R}_p$.

In Theorem 5, we show that $\widehat{\mathsf{Dec}}(sk_i, sk'_j, \tau)$ satisfies the correlated-key distributed decryption property.

Theorem 5. *Our eGSW scheme is a somewhat homomorphic encryption scheme with distributed decryption under the NTRU assumption and the assumption that the eGSW scheme is the key-dependent message (KDM) secure.*

Proof. We begin by analyzing the correctness of $\mathsf{Dec}(sk, \tau)$ for the secret key sk and a ciphertext $\tau = \mathsf{Enc}(sk, m)$. Specifically,

$$\begin{aligned} t \cdot sk &= \mathsf{BitDecomp}^{-1}\left(\mathsf{BitDecomp}(q/p) \cdot \tau\right) \cdot sk \\ &= \mathsf{BitDecomp}(q/p) \cdot \tau \cdot \boldsymbol{v} \\ &\approx \mathsf{BitDecomp}\left(q/p\right) \cdot m \cdot \boldsymbol{v} \\ &= (q/p) \cdot m \cdot sk \ , \end{aligned}$$

where $\boldsymbol{v}^{\mathsf{T}} \stackrel{\text{def}}{=} \mathsf{Powersof2}(sk)$ and $\tau \cdot \boldsymbol{v} \approx m \cdot \boldsymbol{v}$ based on Equation (6). Applying the rounding $\lfloor \cdot \rceil_p$ to both sides of the above equation, we obtain $\lfloor t \cdot sk \rceil_p = m \cdot sk$, meaning that $\mathsf{Dec}(sk, \tau) = m \cdot sk$. Following the GSW scheme [GSW13], the NTRU-based GSW ciphertexts support homomorphic addition and multiplication operations for a maximum multiplication depth L.

CPA security. We use a sequence of hybrids to prove that $\mathsf{Enc}(sk, m_0)$ is computationally indistinguishable from $\mathsf{Enc}(sk, m_1)$ for two messages $m_0, m_1 \in \mathbb{Z}_p$ chosen by a PPT adversary $\mathcal{A}$.

Hybrid 0. This is the real game and denoted by $\mathcal{G}_0$. This hybrid sends params and $\mathsf{Enc}(sk, m_b)$ to $\mathcal{A}$ for a random bit b.

Hybrid 1. This hybrid, denoted by $\mathcal{G}_1$, is the same as $\mathcal{G}_0$, except that replacing mk in params with a random element $r \in \mathcal{R}_q$.

We rewrite $mk = e/sk + (q/p)/sk$. Based on the assumption that the eGSW scheme is KDM secure, it is infeasible to distinguish mk from $mk' = e/sk$. Under the NTRU assumption, mk' is computationally indistinguishable from a random element $r \in \mathcal{R}_q$. Therefore, $\mathcal{G}_1$ is computationally indistinguishable from $\mathcal{G}_0$.

Hybrid 2. This hybrid, denoted by $\mathcal{G}_2$, is the same as $\mathcal{G}_1$, except that replacing $\boldsymbol{e}/sk$ used in the computation of $\mathsf{Enc}(sk, m_b)$ with a random vector in $\mathcal{R}_q^N$.

It is straighforward to observe that $\mathcal{G}_2$ is computationally indistinguishable from $\mathcal{G}_1$ under the NTRU assumption. Following a similar analysis in the proof of Theorem 2, we have that the advantage of $\mathcal{A}$ (guessing the bit b) in $\mathcal{G}_0$ is negligible in λ.

Linear distributed decryption. Let $sk_0 \in \mathcal{R}_p$ be a uniform element with $\mathsf{LSB}(sk_0) = 0$. Let $sk_1 = sk_0 + sk$ over $\mathcal{R}_p$. Based on Lemma 1, we obtain

$$\Pr[\ sk_1 = sk_0 + sk \text{ (without modulo } p)\ ] \geq 1 - n \cdot (\|sk\|_\infty + 1)/p \geq 1 - \mathsf{negl}(\lambda)\ .$$

For a ciphertext $\tau \leftarrow \mathsf{Enc}(sk, m)$, we define $t := \mathsf{BitDecomp}^{-1}(\mathsf{BitDecomp}(q/p) \cdot \tau)$. Thus, we have $sk_1 \cdot t = (sk_0 + sk) \cdot t = sk_0 \cdot t + sk \cdot t$. Based on Lemma 2, we have the following, except with probability $\mathsf{negl}(\lambda)$,

$$\mathsf{Dec}(sk_1, \tau) = \mathsf{Dec}(sk_0, \tau) + \mathsf{Dec}(sk, \tau)\ .$$

One can observe that for any $i \in \{0, 1\}$,

$$\mathsf{Dec}(sk_i, \tau) = -\mathsf{Dec}(sk_i, -\tau)\ \ ,$$

since, for any $x \in \mathcal{R}_q$ and $\boldsymbol{y} \in \mathcal{R}_q^N$, $\lfloor -x \rceil_p = -\lfloor x \rceil_p$ and $-\mathsf{BitDecomp}^{-1}(\boldsymbol{y}) = \mathsf{BitDecomp}^{-1}(-\boldsymbol{y})$.

Correlated-key distributed decryption. Let $sk_0, sk_0' \in \mathcal{R}_p$ be two uniform elements such that $\mathsf{LSB}(sk_0) = 0$ and $\mathsf{LSB}(sk_0') = 0$. Let $sk_1 = sk_0 + sk$ and $sk_1' = sk_0' + sk$ over $\mathcal{R}_p$.

Given a ciphertext $\tau \leftarrow \mathsf{Enc}(sk, m)$, we have, with probability $1 - \mathsf{negl}(\lambda)$,

$$\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) = -\widehat{\mathsf{Dec}}(sk_i, sk_j', -\tau)\ \ ,$$

because, for any $x \in \mathcal{R}_q$ and $\boldsymbol{y} \in \mathcal{R}_q^N$, $\lfloor -x \rceil_p = -\lfloor x \rceil_p$ and $-\mathsf{BitDecomp}^{-1}(\boldsymbol{y}) = \mathsf{BitDecomp}^{-1}(-\boldsymbol{y})$. In the following, we show that for any $i, j \in \{0, 1\}$, with probability $1 - \mathsf{negl}(\lambda)$,

$$\widehat{\mathsf{Dec}}(sk_i, sk_j', \tau) = \widehat{\mathsf{Dec}}(sk_0, sk_0', \tau) + i \cdot j \cdot \mathsf{Dec}(sk, \tau)\ .$$

Based on Lemma 1, except with probability $\mathsf{negl}(\lambda)$, we have

$$sk_1 = sk_0 + sk \text{ (without modulo } p) \quad \text{and} \quad sk'_1 = sk'_0 + sk \text{ (without modulo } p) \,.$$

For each $i, j \in \{0, 1\}$, we have the following:

- If $(i, j) = (0, 0)$, this is a trivial case. Let $x_{00} = sk_0 \cdot sk'_0 \cdot mk$ and $y_{00} = \lfloor x_{00} \rceil_p$.
- If $(i, j) = (1, 0)$, from $sk_1 = sk_0 + sk$, we have

$$\begin{aligned} x_{10} &= sk_1 \cdot sk'_0 \cdot mk = sk_0 \cdot sk'_0 \cdot mk + sk \cdot mk \cdot sk'_0 \\ &\approx sk_0 \cdot sk'_0 \cdot mk + (q/p) \cdot sk'_0 = x_{00} + (q/p) \cdot sk'_0 \,. \end{aligned}$$

  Based on Lemma 2, except with probability $\mathsf{negl}(\lambda)$, we have

$$y_{10} = \lfloor x_{10} \rceil_p - sk'_0 = \lfloor x_{00} \rceil_p + sk'_0 - sk'_0 = y_{00} \,,$$

  which implies $\widehat{\mathsf{Dec}}(sk_1, sk'_0, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau)$.

- If $(i, j) = (0, 1)$, from $sk'_1 = sk'_0 + sk$, we have

$$\begin{aligned} x_{01} &= sk_0 \cdot sk'_1 \cdot mk = sk_0 \cdot sk'_0 \cdot mk + sk \cdot mk \cdot sk_0 \\ &\approx sk_0 \cdot sk'_0 \cdot mk + (q/p) \cdot sk_0 = x_{00} + (q/p) \cdot sk_0 \,. \end{aligned}$$

  Based on Lemma 2, except with probability $\mathsf{negl}(\lambda)$, we obtain

$$y_{01} = \lfloor x_{01} \rceil_p - sk_0 = \lfloor x_{00} \rceil_p + sk_0 - sk_0 = y_{00} \,,$$

  which implies $\widehat{\mathsf{Dec}}(sk_0, sk'_1, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau)$.

- If $(i, j) = (1, 1)$, from $sk_1 = sk_0 + sk$ and $sk'_1 = sk'_0 + sk$, we have

$$\begin{aligned} x_{11} &= sk_1 \cdot sk'_1 \cdot mk = (sk_0 \cdot sk'_0 + sk \cdot sk'_0 + sk_0 \cdot sk + sk \cdot sk) \cdot mk \\ &= (sk_0 \cdot sk'_0 + sk \cdot sk'_1 + sk_1 \cdot sk - sk \cdot sk) \cdot mk \\ &= sk_0 \cdot sk'_0 \cdot mk + sk \cdot mk \cdot (sk_1 + sk'_1 - sk) \\ &\approx x_{00} + (q/p) \cdot (sk_1 + sk'_1 - sk) \,. \end{aligned}$$

  From Lemma 2, except with probability $\mathsf{negl}(\lambda)$, we have

$$\begin{aligned} y_{11} &= \lfloor x_{11} \rceil_p - sk_1 - sk'_1 = \lfloor x_{00} + (q/p) \cdot (sk_1 + sk'_1 - sk) \rceil_p - sk_1 - sk'_1 \\ &= \lfloor x_{00} \rceil_p - sk = y_{00} - sk \,. \end{aligned}$$

  Therefore, based on Lemma 2, except with probability $\mathsf{negl}(\lambda)$, we obtain

$$\begin{aligned} \widehat{\mathsf{Dec}}(sk_1, sk'_1, \tau) &= \mathsf{Dec}(y_{11}, -\tau) = \lfloor -y_{00} \cdot t + sk \cdot t \rceil_p \\ &= \lfloor -y_{00} \cdot t \rceil_p + m \cdot sk \\ &= \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau) + \mathsf{Dec}(sk, \tau) \,, \end{aligned}$$

  where $t = \mathsf{BitDecomp}^{-1}(\mathsf{BitDecomp}(q/p) \cdot \tau)$ and $sk \cdot t \approx (q/p) \cdot m \cdot sk$.

In conclusion, for any $i, j \in \{0, 1\}$, with probability $1 - \mathsf{negl}(\lambda)$, we have

$$\widehat{\mathsf{Dec}}(sk_i, sk'_j, \tau) = \widehat{\mathsf{Dec}}(sk_0, sk'_0, \tau) + i \cdot j \cdot \mathsf{Dec}(sk, \tau) \,,$$

which completes the proof. □