Rerandomizable Garbling Beyond BHHO-Based KMHE and a Construction from LWE

Raphael Heitjohann, Jonas von der Heyden and Tibor Jager

University of Wuppertal, Wuppertal, Germany

Abstract. Rerandomizable garbling schemes (RGS) are a fundamental primitive enabling advanced cryptographic constructions such as multi-hop homomorphic encryption (Gentry, Halevi, Vaikuntanathan, CRYPTO 2010), iO-combiners (Ananth, Jain, Naor, Sahai, Yogev, CRYPTO 2016), and YOSO-style MPC (Acharya, Hazay, Kolesnikov, Prabhakaran, TCC 2022 and CRYPTO 2024). All known RGS instantiations rely on discrete-logarithm-type assumptions, concretely via the BHHO scheme (Boneh, Halevi, Hamburg, Ostrovsky, CRYPTO 2008) and its recent pairing-based variant (Heitjohann, von der Heyden, Jager, CRYPTO 2025). This reliance stems from the use of key-and-message homomorphic encryption (KMHE) that supports rerandomization and strong "key-privacy under leakage", which hides the applied homomorphic transformations and is currently only achievable using BHHO-style bitwise encryption in combination with permutation-based homomorphisms.

Our first main contribution is a new garbling technique that shifts the burden of hiding homomorphic transformations from the encryption scheme to the garbling itself. This allows us to construct RGS from KMHE schemes with substantially weaker properties, avoiding the need for bitwise encryption and permutation-based homomorphisms, and enabling the construction of RGS from KMHE with other homomorphisms, such as additively-homomorphic ones.

As a second contribution, we give a concrete lattice-based instantiation of the required KMHE primitive, based on LWE. This construction is a variant of the GSW FHE scheme (Gentry, Sahai, Waters, CRYPTO 2013), which uses new techniques for rerandomization and is tailored to the requirements of our new garbling scheme by supporting a "homomorphic"/"lossy" mode. This yields the first RGS construction that does not rely on discrete-logarithm assumptions and may provide post-quantum security.

Keywords: Garbled Circuits · Rerandomizable Garbling · Lattices · KMHE · SCALES · Post-quantum cryptography

1 Introduction

Rerandomizable garbled circuits and their applications. Garbled circuits [Yao86] are a fundamental tool in secure computation. They have evolved into powerful primitives that support advanced functionalities. One such primitive is a *rerandomizable* garbling scheme (RGS) [GHV10]. Intuitively, an RGS allows a garbled circuit to be rerandomized multiple times, possibly by different parties, such that the resulting garbling is indistinguishable from a fresh garbling of the same circuit. RGS have found many interesting applications. These include i-hop homomorphic encryption [GHV10], secure computation

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project numbers 458551681 and 503491151.

E-mail: heitjohann@uni-wuppertal.de (Raphael Heitjohann), jonasvdheyden@gmail.com (Jonas von der Heyden), jager@uni-wuppertal.de (Tibor Jager)

This work is licensed under a "CC BY 4.0" license.

Date of this document: 2026-10-28.

without simultaneous interaction [HLP11], indistinguishability obfuscation (iO) combiners [AJN+16], reusable garbled circuits [ZLXL23], and YOSO-style multi-party computation (MPC) protocols such as SCALES [AHKP22, AHKP24].

Reliance on DLOG-type assumptions through KMHE with strong properties. All known constructions of rerandomizable garbling schemes are based on discrete-logarithm-type assumptions. Concretely, existing instantiations rely on the BHHO encryption scheme [BHHO08] or its recent optimized pairing-based variant [HvdHJ25]. This is because all prior RGS constructions are built from *rerandomizable key-and-message homomorphic encryption* (KMHE), where the key space is a subset of the message space and the *same* homomorphic operation can be applied to the keys encrypting a ciphertext and an encrypted message, and rerandomized ciphertexts are indistinguishable from fresh ones. Additionally, and very importantly, strong homomorphic hiding properties known as *KMH-privacy* [AHKP22] or *key privacy under leakage* [HvdHJ25] are required, which we discuss in Section 1.1. Taken together, these requirements make the construction of suitable schemes extremely difficult, which may explain the lack of alternative schemes.

Our contributions. Our main observation is that the strong homomorphic hiding properties required by prior RGS constructions are not intrinsic to rerandomizable garbling, but rather an artifact of the *specific garbling technique*. By redesigning the garbling, we can substantially weaken the requirements imposed on the underlying KMHE. We make two main contributions, each of which might be of independent interest.

1. We introduce a new *double-column garbling* technique. In contrast to prior approaches, our construction shifts the task of hiding homomorphic transformations from the encryption scheme to the structure of the garbling itself. As a result, we eliminate the need for KMHE schemes that provide *KMH-privacy* or *key privacy under leakage*. Our construction only requires a rerandomizable KMHE scheme with *comparatively* weak and natural properties. For example, it can be instantiated with an additively-homomorphic KMHE that is enhanced with a "lossy" mode. Double-column garbling may find other applications, for example in the context of reusable garbling.

2. We provide a lattice-based instantiation of the required KMHE primitive. Our construction is based on Learning With Errors (LWE) [Reg05]. It supports addition-modulo-q as the homomorphism and a lossy encryption mode. While the scheme is *levelled*, which means the number of supported rerandomizations is limited due to noise growth, it suffices for known applications of rerandomizable garbling. We have implemented our construction in Rust as a proof of concept.[1] It uses deliberately insecure toy parameters to facilitate testing and demonstration.

Taken together, this yields the first construction of (levelled) rerandomizable garbling schemes that does not rely on discrete-logarithm-type assumptions, and the first one based on a plausibly post-quantum assumption. See Section 1.2 for how this relates to lattice-based reusable garbling and other nearby primitives. Beyond enabling post-quantum secure instantiations of existing protocols such as SCALES [AHKP22, AHKP24], we believe that our techniques clarify the conceptual role of KMHE in rerandomizable garbling, may facilitate the development of other KMHE-schemes that are suitable for rerandomizable garbling, and may be useful for the design of future garbling-based primitives.

[1] `https://github.com/jvdheyden/2lkmhe/`

1.1 Technical Overview

Garbled circuits in a nutshell. Recall that a Boolean circuit is a directed acyclic graph with vertices representing gates and edges representing wires connecting said gates. Each gate has two input wires, its output wire may be connected to any number of gates.

Circuit garbling. To garble a Boolean circuit, each wire is assigned one of two possible keys (henceforth called *labels*) representing the Boolean input value 0 or 1.

In Yao's seminal construction [Yao86], a *garbled gate* consists of four ciphertexts, each of which is double-encrypted, using one of the four possible label combinations from the two input wires as encryption keys. The encrypted messages are the labels of the output wire. Whether the 0- or the 1-label is encrypted is determined by the gate type (AND, OR, XOR, etc.) and by which input values the labels represent. See Figure 1 for an illustration.

A circuit consisting of multiple gates is garbled gate-by-gate. Importantly, this is done in a *consistent* way, i.e., if there is a wire from gate g to gate g_i, then the *messages* encrypted in the ciphertexts representing g are used as the *encryption keys* for the ciphertexts representing g_i. Finally, the ciphertexts within every gate are permuted.

Garbled circuit evaluation. The evaluator receives *one* label for every input wire, the one corresponding to the bit-value of the desired circuit input on that wire. These labels are called *active*. For a gate with two input wires, the evaluator is able to use these labels to decrypt exactly *one* out of the four ciphertexts representing this gate, namely the unique one that is encrypted with these two labels. The evaluator obtains the label corresponding to the output value of the gate. This way, the evaluator can proceed recursively through the circuit, decrypting exactly one ciphertext per gate until it has obtained the labels corresponding to the circuit output.



Figure 1: Garbled AND gate, using a two-key encryption scheme $\mathsf{Enc}(k_l, k_r, m)$ that uses two keys k_l and k_r to encrypt a message m. The ciphertexts are randomly permuted within each gate.

Rerandomizable garbling schemes (RGS). Intuitively, RGS are garbled circuits that can be *rerandomized*, given only the garbling but not the original circuit, such that they are computationally indistinguishable from a fresh garbled circuit. Note that this requires a *consistent* rerandomization of all wire labels, such that the rerandomized garbled circuit remains functionally identical, but with "fresh" wire labels.

Using rerandomizable KMHE to construct RGS. All known constructions of RGS for general Boolean circuits, including ours, are based on rerandomizable *key-and-message homomorphic encryption* (KMHE), where the *same* homomorphism can be applied to keys and messages. Previous constructions [GHV10, AHKP22, HvdHJ25] are based on Yao's original garbled circuits, but with a KMHE scheme to encrypt gates. The key-and-message homomorphism of the KMHE is used to consistently rerandomize the wire labels of a given garbled circuit without knowing the circuit. To this end, for every wire w of the circuit

a random homomorphism is chosen. Then, for every gate g with *output* wire w (which exists, unless w is an input wire), the homomorphism is applied to the encrypted *messages*. Subsequently for every gate g_i with *input* wire w (there exists at least one, unless w is an output wire), the same homomorphism is applied to the *encryption keys*. Using the same homomorphism on both "ends" of a wire ensures that the garbled circuit remains consistent, and requires an encryption scheme that permits the *same* homomorphism in the key- and the message-space.

To make the rerandomized garbled circuit indistinguishable from a fresh garbling, it is not sufficient to rerandomize only the wire labels. We also need to randomly permute the ciphertexts within every garbled gate and rerandomize all ciphertexts to make them look like fresh encryptions of the same message. This makes the ciphertexts unlinkable to the ciphertexts from the original circuit, and finally yields a rerandomized garbled circuit that is indistinguishable from a fresh garbling.

The need for *KMH-privacy* or *key privacy under leakage*. *KMH-privacy* [AHKP22] and *key privacy under leakage* [HvdHJ25] are two slightly different definitions that capture the same property of a KMHE scheme. This property is essential for all previous constructions of RGS. Let us explain this property.

The security definition of RGS essentially considers a setting where first a rerandomizable garbled circuit is semi-honestly generated by the adversary, such that the adversary knows all labels. Then the garbled circuit is honestly rerandomized. Finally, the adversary receives the rerandomized garbled circuit back for evaluation, along with *one* active label for every input wire. Essentially, we require that the adversary cannot distinguish a rerandomized garbled circuit from a fresh garbling of the same circuit, even when given active labels for the rerandomization and the entire randomness of the original garbling.

As described above, rerandomizable garbled circuits based on Yao's original garbling and KMHE [AHKP22, HvdHJ25] use only *one* homomorphism per wire, and the *same* homomorphism is applied to *both* labels of this wire. This is because the rerandomizer *does not know* which ciphertext encrypts the label corresponding to a 0- or 1-bit, or even only which ciphertexts encrypt the same label. Using the *same* homomorphism for both labels of a wire makes it possible to *consistently* rerandomize the garbled circuit, by applying the same homomorphism to all encrypted messages.

Recall that the adversary in the RGS security definition knows the labels of the original circuit, along with the active label for every input wire. If the active label would leak information about the homomorphism used for rerandomization, this would potentially reveal the rerandomized *inactive* label to the adversary, which would break garbling privacy. As a concrete example, consider a circuit consisting of a single gate g with input wires ℓ, r. The garbling consists of a (permuted) list of four ciphertexts

$$\mathsf{Enc}\left(\mathsf{k}_\ell^0, \mathsf{k}_r^0, \mathsf{k}^{g(0,0)}\right), \mathsf{Enc}\left(\mathsf{k}_\ell^0, \mathsf{k}_r^1, \mathsf{k}^{g(0,1)}\right), \mathsf{Enc}\left(\mathsf{k}_\ell^1, \mathsf{k}_r^0, \mathsf{k}^{g(1,0)}\right), \mathsf{Enc}\left(\mathsf{k}_\ell^1, \mathsf{k}_r^1, \mathsf{k}^{g(1,1)}\right).$$

This gate is semi-honestly created by the adversary, which thus knows all the labels $\mathsf{k}_\ell^0, \mathsf{k}_\ell^1, \mathsf{k}_r^0, \mathsf{k}_r^1, \mathsf{k}^0, \mathsf{k}^1$. This gate is rerandomized, using homomorphisms $\sigma_\ell, \sigma_r, \sigma_o$, which yields a (permuted) list of ciphertexts

$$\mathsf{Enc}\left(\sigma_\ell(\mathsf{k}_\ell^0), \sigma_r(\mathsf{k}_r^0), \sigma_o(\mathsf{k}^{g(0,0)})\right), \mathsf{Enc}\left(\sigma_\ell(\mathsf{k}_\ell^0), \sigma_r(\mathsf{k}_r^1), \sigma_o(\mathsf{k}^{g(0,1)})\right),$$
$$\mathsf{Enc}\left(\sigma_\ell(\mathsf{k}_\ell^1), \sigma_r(\mathsf{k}_r^0), \sigma_o(\mathsf{k}^{g(1,0)})\right), \mathsf{Enc}\left(\sigma_\ell(\mathsf{k}_\ell^1), \sigma_r(\mathsf{k}_r^1), \sigma_o(\mathsf{k}^{g(1,1)})\right).$$

The adversary receives active labels $\sigma_\ell(\mathsf{k}_\ell^{b_\ell})$ and $\sigma_r(\mathsf{k}_r^{b_r})$ for some circuit input $b_\ell, b_r \in \{0,1\}$. Let us consider the homomorphism σ_ℓ. We need that the values $(\mathsf{k}_\ell^0, \mathsf{k}_\ell^1, \sigma_\ell(\mathsf{k}_\ell^{b_\ell}))$, which are all known to the adversary, reveal only very little about σ_ℓ, as otherwise the adversary could obtain the *inactive* label $\sigma_\ell(\mathsf{k}_\ell^{1-b_\ell})$, which breaks garbling privacy.

Thus, due to the construction of the garbling and the fact that the *same* homomorphism is applied to both labels on a wire, we need this homomorphism σ_ℓ to remain hidden, even when given $(\mathsf{k}, \sigma_\ell(\mathsf{k}))$. This immediately rules out natural homomorphisms, such as additive ones of the form $\mathsf{k} \mapsto \mathsf{k} + \sigma_\ell$. Intuitively, a KMHE scheme where the homomorphism achieves this, and there is no additional leakage through the encryption scheme, has *KMH-privacy* or *key privacy under leakage*.

***KMH-privacy* or *key privacy under leakage* via permutations.** To achieve these properties, [GHV10, AHKP22, HvdHJ25] utilize homomorphisms that are *permutations*. Labels are random bitstrings of length ℓ with Hamming weight $\ell/2$. Then one can show that, given $k^0, k^1, \sigma(k^b)$ for $b \in \{0, 1\}$, the other label $\sigma(k^{1-b})$ still has very high entropy. In combination with a suitable encryption scheme [BHHO08, HvdHJ25] and the leftover hash lemma [HILL99], this yields a KMHE scheme with *KMH privacy* [AHKP22] or *key privacy under leakage* [HvdHJ25].

Unfortunately, it is not clear how to construct such KMHE *without* permutation homomorphisms. Clearly, simple(r) homomorphisms such as linear functions will not work. Therefore, schemes with permutation homomorphisms [GHV10, AHKP22] based on [BHHO08] and [HvdHJ25], all of which are based on discrete-log-type assumptions, are currently the *only* known KMHE constructions suitable for RGS. Constructing alternative schemes, in particular ones based on post-quantum assumptions, is an open problem in this area, which we solve.

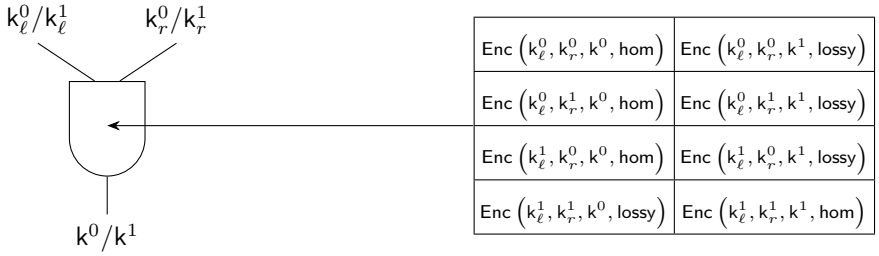


Figure 2: Double-column garbled AND gate.

A new approach to garbling. Instead of hiding information about one homomorphism per wire, we construct the garbling such that it is possible to apply two *different* homomorphisms to the two labels of the *same* wire, while at the same time enabling *consistent* application of homomorphisms across a garbled circuit. We propose a novel "double-column" garbling technique, where each garbled gate consists of four rows (as usual) and two columns, cf. Figure 2. Each column corresponds to one label of the output wire, so that one can apply a homomorphism per column. Importantly, we design the garbling so that it preserves the following invariant, which is also maintained during rerandomization:

Let g be a gate with input wires ℓ and r. Let g_ℓ, g_r be the gates with output wires ℓ, r, respectively. The left column of g_ℓ contains the labels that are used to encrypt the first and second row of g, and the right column contains the labels that encrypt the third and fourth row. Similarly, the left column of g_r contains the labels that encrypt the first and third row of g, and right column the labels that encrypt the second and fourth. Note that the indices here are grouped differently: the "left" input wire ℓ decides between the first/last two rows of g, while the "right" input wire r decides between an odd/even-indexed row of g. We now have a fixed, invariant, deterministic connection between the columns of one gate and the rows of subsequent gates.

This invariant enables consistent application of two different homomorphisms $\sigma^\ell_\ell, \sigma^r_\ell$ to the two labels of the same wire ℓ:

- Apply σ_ℓ^ℓ to the *messages* encrypted in the left column of g_ℓ, and to the *keys* encrypting the first and second row of g.
- Apply σ_ℓ^r to the *messages* encrypted in the right column of g_ℓ, and to the *keys* encrypting the third and fourth row of g.

Similarly, two different homomorphisms can be applied to the columns of g_r and the corresponding first/third and second/fourth rows of g.

It remains to explain how the garbled circuit is rerandomized. We use a KMHE scheme that permits two modes: a "homomorphic" mode, which is a correct KMHE scheme that permits application of homomorphisms, and a "lossy" mode where the ciphertexts reveal no information about the encrypted messages or the homomorphisms applied to them. Every row contains exactly one "homomorphic" ciphertext in one column, and a "lossy" one in the other. The function implemented by the gate determines how the homomorphic/lossy ciphertexts are distributed across the columns and rows, see Figure 2. This hides the homomorphism applied to inactive labels during evaluation: as in standard garbled circuits, the evaluator will only be able to decrypt one row, and this row carries only information in one column, because the other column is "lossy".

Again, it is not sufficient to only rerandomize the wire labels. Whether the left column of a garbled gate belongs to the 0-label and the right column to the 1-label, or vice versa, is chosen at random for every gate in the original garbling. This can be easily rerandomized, just by randomly swapping the columns of garbled gates. If the columns of a gate are swapped, then also the corresponding rows of subsequent gates must be swapped, so that the invariant is maintained. Finally, just like in previous constructions, the last step of circuit rerandomization is to rerandomize all ciphertexts, to make them look like fresh encryptions of the same message. This finally yields a rerandomized garbled circuit.

Ciphertext rerandomization is particularly difficult in the lattice-based setting and requires a careful design of the KMHE scheme. Below, we describe a *levelled* scheme that permits a (sufficiently large, but) bounded number of rerandomizations.

Constructing iDRE. A main application of RGS is constructing *Incremental Decomposable Randomized Encodings* (iDREs), introduced in [AHKP22]. Essentially, in an iDRE, a set of parties work together to prepare a *randomized encoding*, such as a garbling, of some function. This works *incrementally*, in the sense that the randomized encoding is built step by step. Each encoder "speaks once" and then disappears. Any single honest encoder is enough for privacy, and the final encoding reveals only the joint output.

The changes made by us to the definitions of RGS and our new garbling technique, including that we have a "levelled" notion, mean that we also have to adjust the construction of iDRE from [AHKP22]. Our construction uses essentially the same techniques as our RGS. See Section 6 for details.

A two-key KMHE based on LWE with levelled rerandomizability. Thanks to our new garbling technique, we can now use a simple additive offset for our homomorphism and do not have to rely on bit encryption anymore. We employ the LWE-based GSW scheme [GSW13] and the rerandomization techniques from [BdPMW16] to construct a rerandomizable two-key KMHE scheme with lossy mode. We construct a *two-key* variant, as this is what we need for garbled circuits. Let us now sketch a *one-key* variant of this construction; a detailed discussion and specification of our two-key construction can be found in Section 5. To encrypt a message $\mu \in \mathbb{Z}_q$ using a secret key $\mathbf{s} \in \mathbb{Z}_q^n$, GSW samples random $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$ and noise $\mathbf{e} \in \mathbb{Z}_q^m$ from an appropriate distribution, e.g. a discrete Gaussian, and outputs the ciphertext

$$\mathbf{C} \leftarrow \begin{pmatrix} \mathbf{A} \\ \mathbf{sA} + \mathbf{e} \end{pmatrix} + \mu\mathbf{G},$$

where $\mathbf{G} = \mathbf{I}_{n+1} \otimes \mathbf{g} \in \mathbb{Z}_q^{(n+1)\times(n+1)k}$, with $k = \lceil \log q \rceil$, denotes the gadget matrix [MP12] with gadget vector $\mathbf{g} = (2^0, 2^1, \ldots, 2^{k-1})$. Given $\mathbf{C}$ and $\mathbf{s}$, μ can then be recovered from $(-\mathbf{s}, 1) \cdot \mathbf{C}$ using the LWE gadget decoding algorithms from [MP12, GMP19].

To encrypt a secret key $\mathbf{s} \in \mathbb{Z}_q^n$, we can just encrypt each entry of $\mathbf{s}$ separately. The supported key and message homomorphism will be addition over $\mathbb{Z}_q^n$. To add an offset $\boldsymbol{\sigma} \in \mathbb{Z}_q^n$ to the key, we add $\boldsymbol{\sigma}\mathbf{A}$ to the last row of ciphertext $\mathbf{C}$. Note that this requires clear access to the matrix $\mathbf{A}$, which is currently obfuscated by $\mu\mathbf{G}$. Since we do not need the full gadget $\mathbf{G}$ for our scheme, we address this issue by using a reduced gadget

$$\widehat{\mathbf{G}} = \begin{pmatrix} \mathbf{0}^{n\times m} \\ \mathbf{0}^{m-k} \mid \mathbf{g} \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m},$$

where now $m \neq (n+1)k$ may hold. With this, it holds that

$$\mathbf{C} \leftarrow \begin{pmatrix} \mathbf{A} \\ \mathbf{s}\mathbf{A} + \mathbf{e} \end{pmatrix} + \mu\widehat{\mathbf{G}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}\mathbf{A} + \mathbf{e} + (\mathbf{0}^{m-k} \mid \mu\mathbf{g}) \end{pmatrix},$$

and we can still decrypt μ as before. For the message homomorphism we note that adding a $\tau \in \mathbb{Z}_q$ is easy by adding $\tau\mathbf{g}$ to the last row of $\mathbf{C}$. However, recall that we also need to support the lossy mode. If we just add τ to a ciphertext in lossy mode, then decryption would immediately leak τ since the message in lossy ciphertexts is always the same fixed lossyness indicator, for example $\mathbf{0}^k$.

To enable lossyness, we make use of the multiplication-by-constant message homomorphism support of GSW encryption. Specifically, [MP12] (see also Claim 3.1 in [AP14]) provide a function $\mathbf{G}_{\mathsf{det}}^{-1}$ which, given a matrix $\mathbf{V} \in \mathbb{Z}_q^{(n+1)\times m}$, outputs a matrix $\mathbf{X}$ with *small* entries such that $\mathbf{G}\mathbf{X} = \mathbf{V}$. It is easy to adapt this algorithm to the reduced gadget $\widehat{\mathbf{G}}$, resulting in $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ which, given a *vector* $\mathbf{v} \in \mathbb{Z}_q^k$, outputs a matrix $\mathbf{X}$ with also small entries such that

$$\widehat{\mathbf{G}}\mathbf{X} = \begin{pmatrix} \mathbf{0}^{n\times m} \\ \mathbf{0}^{m-k} \mid \mathbf{v} \end{pmatrix} \in \mathbb{Z}_q^{(n+1)\times m}.$$

To use this for the lossy homomorphism mode, we add a rerandomization helper ciphertext $\mathbf{D}$ to each ciphertext which encrypts $\zeta = 1$ if the encryption mode is hom and $\zeta = 0$ if the encryption mode is lossy:

$$\mathbf{D} \leftarrow \begin{pmatrix} \mathbf{A}' \\ \mathbf{s}\mathbf{A}' + \mathbf{e}' \end{pmatrix} + \zeta\widehat{\mathbf{G}}.$$

To add an offset $\tau \in \mathbb{Z}_q$ to the message in $\mathbf{C}$, we then compute

$$\mathbf{C}' \leftarrow \mathbf{C} + \mathbf{D} \cdot \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g}) = \begin{pmatrix} \mathbf{A} + \mathbf{A}' \cdot \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g}) \\ \mathbf{s}(\mathbf{A} + \mathbf{A}' \cdot \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g})) + \mathbf{e} + \mathbf{e}' \cdot \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g}) \end{pmatrix} + (\mu + \zeta\tau)\widehat{\mathbf{G}}.$$

Since the output of $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ is small, the noise remains small and decryption remains possible. If the ciphertext mode is hom, then $\zeta = 1$, and so the new message will be $\mu + \tau$. On the other hand, if the ciphertext mode is lossy, then $\zeta = 0$, and so the new message will remain μ, as desired.

It remains to show how the ciphertext $\mathbf{C}'$ can be rerandomized such that it fulfills (levelled) rerandomizability. To hide the noise $\mathbf{e} + \mathbf{e}' \cdot \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g})$, we follow the established practice of adding large smudging noise to drown out the existing noise [AJL+12]. This also serves to hide the $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g})$ value which may leak information about τ. The other randomness that needs to be refreshed is the matrices $\mathbf{A}$ and $\mathbf{A}'$. For this we rely on the approach from [AP14, BdPMW16]: By suitably randomizing the $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ algorithm, resulting in a randomized algorithm $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}$, and combined with the leftover hash lemma (LHL)

[HILL99], one can show that $\mathbf{A}' \cdot \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\tau\mathbf{g})$ looks uniform. Then overall, $\mathbf{A} + \mathbf{A}' \cdot \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\tau\mathbf{g})$ must look uniform as well. Rerandomization of $\mathbf{A}'$, as contained in $\mathbf{D}$, can be done even simpler via $\mathbf{A}' \cdot \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\mathbf{g})$. This does not affect the encrypted value ζ.

One major obstacle resulting from this approach is that, in the setting of RGS, the adversary may perform a number of *honest-but-curious* rerandomizations before the last honest rerandomization is done. We showcase this problem using the rerandomization of $\mathbf{D}$: The adversary will sample some $\mathbf{X}_1, \ldots, \mathbf{X}_\ell \leftarrow \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\mathbf{g})$ and compute $\mathbf{A}' \prod_{i=1}^{\ell} \mathbf{X}_i$ during rerandomization. Then the last honest rerandomization is done by again sampling a $\mathbf{X}_{\ell+1} \leftarrow \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\mathbf{g})$ and computing

$$\left(\mathbf{A}' \prod_{i=1}^{\ell} \mathbf{X}_i\right) \cdot \mathbf{X}_{\ell+1}.$$

We would like to apply the LHL now to show that this product looks uniform. The LHL states that, as long as the randomness extractor $\mathbf{A}'$ is uniformly sampled, and the secret $\mathbf{X}$ has sufficient min-entropy (satisfied by $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}$), their product $\mathbf{A}'\mathbf{X}$ is indistinguishable from uniform even when $\mathbf{A}'$ is known. However, the values $\mathbf{X}_1, \ldots, \mathbf{X}_\ell$ are known to the adversary. To address these known randomizers, we use a result from [GPV08] to show that sampling $\mathbf{X}_1, \ldots, \mathbf{X}_\ell$ from $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}$ and then computing $\mathbf{A}' \prod_{i=1}^{\ell} \mathbf{X}_i$ is statistically indistinguishable from first sampling $\mathbf{A}' \prod_{i=1}^{\ell} \mathbf{X}_i$ as a uniform matrix $\mathbf{Y}$ and then sampling $\mathbf{X}_1, \ldots, \mathbf{X}_\ell$ from a restricted distribution that ensures $\mathbf{A}' \prod_{i=1}^{\ell} \mathbf{X}_i = \mathbf{Y}$. Since $\mathbf{Y}$ is now uniformly sampled, we can apply the LHL to show that $\mathbf{Y}\mathbf{X}_{\ell+1}$ looks uniform.

1.2 Related Work

Rerandomizable garbling. The first RGS were devised in [GHV10] for secure function evaluation and i-hop homomorphic encryption, based on BHHO encryption [BHHO08]. In the context of cryptographic reverse firewalls, [MS15] constructed an RGS for circuits of bounded depth from ElGamal, with rerandomization that also applies to maliciously generated garbled circuits. Acharya et al. [AHKP22, AHKP24] later extended the techniques of [GHV10] to construct the SCALES protocol for constant-round outsourced MPC. [HvdHJ25] then significantly improved the time and space requirements of RGS with a refined, pairing-based variant of BHHO, by tailoring the underlying encryption scheme for use within RGS. Recently, [AHKP25] showed more efficient techniques for SCALES for permutation branching programs (PBPs).

Reusable garbling and reusable non-interactive MPC. There is a large body of work on lattice-based *reusable* garbled circuits [GKP+13, BGG+14, CC17, Agr17, HLL23]. Despite the similar terminology, reusability and rerandomizability are very different properties. A reusable garbled circuit can be evaluated on an unbounded number of encoded inputs, and its security is defined with respect to an evaluator who does not know the secrets of the garbler. An RGS, in contrast, has an additional algorithm that, given only a garbled circuit, outputs a garbling of the same circuit together with a transformation of the input encoding. Rerand privacy (Definition 11) requires the result to be indistinguishable from a fresh garbling even for an adversary who knows the randomness of the original garbling and of all rerandomizations except for the last. This is what allows SCALES and iDRE [AHKP22] to tolerate semi-honest corruption of all but one of the parties that jointly produce the garbling.

Similarly, multiparty reusable non-interactive secure computation (mrNISC) [BL20], which can also be based on LWE [BJKL21], lets parties publish reusable input encodings and later compute functions by sending a single message each. As observed in [AHKP22],

the computation of the input parties in such two-round protocols grows with the size of the computed circuit, whereas in SCALES this computation is delegated to a sequence of ephemeral servers that rerandomize the garbled circuit one after another, so that the computation of each input party is independent of the size of the circuit.

Thus, while these works provide lattice-based primitives that are close in terminology and motivation, to the best of our knowledge none of them yields an RGS with rerand privacy, or a SCALES protocol in the sense of [AHKP22]. We also note that for some applications of RGS, lattice-based solutions that do not use RGS are known. For example, multi-hop homomorphic encryption can be obtained from circuit-private FHE [GHV10, DS16], and the lattice-based iO combiner of [AJN+16] uses multi-key FHE instead of RGS. Our result is therefore mainly relevant for applications that use RGS directly, such as SCALES and iDRE [AHKP22].

Lattice-based key-and-message homomorphic encryption. Symmetric encryption with additive key and message homomorphisms based on (R)LWE has been used for secure aggregation [GIK+24, BCGL+25], where it is called key-additive homomorphic encryption (KAHE) in [BCGL+25]. These schemes are not designed to provide a lossy mode or rerandomization that hides the link between a ciphertext and its transformed version.

Relation to GSW-based circuit privacy. Our 2LKMHE scheme combines known lattice techniques: GSW encryption [GSW13], gadget inversion [MP12], randomized gadget inversion [AP14], which [BdPMW16] used to achieve circuit privacy, noise flooding [AJL+12], and a result of [GPV08], which we adapt to the setting of repeated rerandomization. The new aspects are (i) the use of a reduced gadget to keep $\mathbf{A}$ in the clear, so that keys can be shifted by adding $\boldsymbol{\sigma}\mathbf{A}$, (ii) a helper ciphertext that encrypts the mode and turns the multiplication-by-constant homomorphism of GSW into a lossy mode, and (iii) an analysis of repeated rerandomization where the adversary knows the randomness of all but the last rerandomization (Theorem 5). We instantiate these techniques in a two-key variant, as required for garbling. A similar reduced form of GSW, called Half-GSW, was used over module lattices in [SSE+24] to construct linear-only encryption. This setting involves neither key homomorphisms, nor a lossy mode, nor repeated rerandomization of the same ciphertext. Moreover, with the randomized evaluation of [BdPMW16], the noise of the output ciphertext still depends on the norm of the noise of the input ciphertexts. Since the RGS adversary knows the original randomness, and we cannot keep this norm constant after rerandomization (as the rerandomizer cannot predict the norm of the resulting noise), we additionally flood the noise. Multi-hop circuit privacy can also be achieved by sanitizing FHE ciphertexts via bootstrapping [DS16], but this does not provide the key homomorphism required for garbling.

2 Preliminaries

Notation. $\mathbb{N}$ denotes the set of natural numbers without 0. For a $t \in \mathbb{N}$ we write $[t]$ to indicate the set $\{1, \ldots, t\}$. For any finite set X, we write $x \leftarrow\!\!\$\ X$ to indicate that x is sampled uniformly at random from X.

We denote vectors using bold lower-case letters, for example $\mathbf{x}$. By default we assume vectors to be row vectors. To convert them to column notation we write $\mathbf{x}^\top$ to indicate the transpose of $\mathbf{x}$. The elements of a vector $\mathbf{x} \in \mathbb{R}^m$ are denoted by $\mathbf{x} = (x_1, \ldots, x_m)$. We are interested in two different norms of $\mathbf{x}$: The infinity norm $\|\mathbf{x}\|_\infty = \max_{i\in[m]} |x_i|$ and the Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i\in[m]} x_i^2}$. Matrices are written in bold capital-case letters, for example $\mathbf{X}$. We define the norm $\|\mathbf{X}\|$ of a matrix $\mathbf{X}$ to be the Euclidean norm of its longest column.

We use the usual notation $\lambda \in \mathbb{N}$ for the security parameter, and denote negligible functions in λ by $\mathsf{negl}(\lambda)$, and polynomials in λ by $\mathsf{poly}(\lambda)$. For D_1 and D_2 some distributions over the same infinite domain $\mathcal{W}$, and parameterized by λ, we say that D_1 and D_2 are statistically indistinguishable, or $D_1 \overset{\text{s}}{\approx} D_2$, if it holds that

$$\mathsf{SD}(D_1, D_2) \coloneqq \sup_{w \subseteq \mathcal{W}} |\Pr[D_1(w)] - \Pr[D_2(w)]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{SD}(D_1, D_2)$ denotes the statistical distance of D_1 and D_2. If D_1 and D_2 are computationally indistinguishable, then we write $D_1 \overset{\text{c}}{\approx} D_2$.

Leftover Hash Lemma. For $\mathcal{S}$ a distribution over some finite set X, we define its min-entropy to be

$$H_\infty(\mathcal{S}) = -\log\left(\max_{x \in X} \Pr[\mathcal{S} = x]\right).$$

The following lemma will be useful to rerandomize matrices.

Lemma 1 (Leftover Hash Lemma (LHL) [HILL99])**.** *Let q be prime, $\epsilon' \in (0,1)$, $\mathcal{S}$ be a distribution over $\mathbb{Z}_q^m$ such that $H_\infty(\mathcal{S}) \geq n \log q + 2\log(1/\epsilon')$. Then it holds that*

$$\mathsf{SD}\big(\{(\mathbf{A}, \mathbf{A}\mathbf{x}) \mid \mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}, \mathbf{x} \leftarrow \mathcal{S}\}, \{(\mathbf{A}, \mathbf{y}) \mid \mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}, \mathbf{y} \leftarrow\!\$\, \mathbb{Z}_q^n\}\big) \leq \epsilon'.$$

Lattices. A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. It can also be written via a basis $\mathbf{B} \in \mathbb{R}^{m \times \ell}$ as $\mathcal{L} = \mathbf{B} \cdot \mathbb{Z}^\ell$, where the columns of $\mathbf{B}$ are linearly independent. Then ℓ is the rank of the lattice, and we call $\mathcal{L}$ *full-rank* if $\ell = m$. By $\widetilde{\mathbf{B}}$ we denote the Gram-Schmidt orthogonalization of the basis $\mathbf{B}$. Of special interest is the q-ary parity-check lattice

$$\mathbf{\Lambda}_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{v} = \mathbf{0}^n \bmod q\},$$

where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. This lattice is always full-rank with rank m. It is called q-ary since $q\mathbb{Z}^m \subseteq \mathbf{\Lambda}_q^\perp(\mathbf{A})$ for any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

Discrete Gaussians over Lattices. For any $m \in \mathbb{N}$, positive real $\alpha > 0$, and $\mathbf{c} \in \mathbb{R}^m$, define Gaussian function $\rho_{\alpha,\mathbf{c}} : \mathbb{R}^m \to (0,1]$ as

$$\rho_{\alpha,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|_2^2 / \alpha^2).$$

We then call α the Gaussian parameter and $\mathbf{c}$ its center.

For any lattice $\mathcal{L} \subseteq \mathbb{R}^m$ we can then calculate its Gaussian measure with respect to parameter α and center $\mathbf{c} \in \mathbb{R}^m$ as $\rho_{\alpha,\mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_{\alpha,\mathbf{c}}(\mathbf{x})$. Based on this, we can define the discrete Gaussian distribution with parameter α over a lattice coset $\mathcal{L} + \mathbf{z}$, with $\mathbf{z} \in \mathbb{R}^m$, as

$$\mathcal{D}_{\mathcal{L}+\mathbf{z},\alpha,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\alpha,\mathbf{c}}(\mathbf{x})}{\rho_{\alpha,\mathbf{c}}(\mathcal{L} + \mathbf{z})},$$

where $\mathbf{x} \in \mathcal{L} + \mathbf{z}$. If we omit α and/or $\mathbf{c}$ in any of the previous definitions, then these are assumed to be 1 and $\mathbf{0}$, respectively.

For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{y} \in \mathbb{Z}_q^n$ by $\mathbf{x} \leftarrow \mathbf{A}^{-1}(\alpha, \mathbf{y})$ we mean that $\mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\alpha}$ conditioned on $\mathbf{A}\mathbf{x} = \mathbf{y} \bmod q$. We can also extend $\mathbf{A}^{-1}(\alpha, \cdot)$ to matrices by applying $\mathbf{A}^{-1}(\alpha, \cdot)$ to each column of the input matrix.

The following lemma will be useful for bounding the Gaussian measure of lattice cosets.

Lemma 2 (Lemma 2.9 in [MR07])**.** *For any lattice $\mathcal{L} \subseteq \mathbb{R}^m$, positive real α and center $\mathbf{c} \in \mathbb{R}^m$, we have $\rho_{\alpha,\mathbf{c}}(\mathcal{L}) \leq \rho_\alpha(\mathcal{L})$.*

The following Gaussian smudging lemma allows us to hide any vector $\mathbf{c}$ by adding a sufficiently wide Gaussian to it.

Lemma 3 (Gaussian Smudging Lemma, Lemma 2.9 in [LLW21])**.** *Let $m \in \mathbb{N}$. For any real $\alpha \geq \omega(\sqrt{\log m})$, and any $\mathbf{c} \in \mathbb{Z}^m$, it holds that $\mathsf{SD}(\mathcal{D}_{\mathbb{Z}^m,\alpha}, \mathcal{D}_{\mathbb{Z}^m,\alpha,\mathbf{c}}) \leq \|\mathbf{c}\|_2/\alpha$.*

Learning with Errors. Learning with Errors (LWE), first introduced by Regev in [Reg05], is an important problem in lattice cryptography, defined as follows.

Definition 1 (Learning with Errors (LWE))**.** Let ν be a positive real. We say that the (decisional) $\mathsf{LWE}_{q,n,m,\nu}$-assumption holds, if it holds that

$$(\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{e} \bmod q) \overset{\text{c}}{\approx} (\mathbf{A}, \mathbf{b}),$$

where $\mathbf{A} \leftarrow\!\$\ \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \leftarrow\!\$\ \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\nu}^m$, and $\mathbf{b} \leftarrow\!\$\ \mathbb{Z}_q^m$.

Hardness of the LWE problem is tied to the modulus-to-noise ratio q/ν, with a smaller ratio implying higher security. Due to the large errors involved in our construction, we will require the LWE assumption to hold even for super-polynomial modulus-to-noise ratios. It is believed that the LWE assumption holds as long as $\log(q/\nu) < n^{\epsilon''}$ for some $\epsilon'' \in (0, 1/2)$.

Smoothing Parameter. The smoothing parameter [MR07] bounds the Gaussian measure of the dual lattice.

Definition 2 (Smoothing Parameter)**.** For any m-dimensional lattice $\mathcal{L}$ and real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\mathcal{L})$ is the smallest $\alpha > 0$ such that $\rho_{1/\alpha}(\mathcal{L}^*) \leq 1 + \epsilon$.

The following lemma bounds the smoothing parameter based on the Gram-Schmidt orthogonalization of the lattice's basis.

Lemma 4 (Lemma 3.1 in [GPV08])**.** *For any m-dimensional lattice $\mathcal{L}$ with basis $\mathbf{B}$ and Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}}$, and real $\epsilon > 0$, we have*

$$\eta_\epsilon(\mathcal{L}) \leq \left\|\widetilde{\mathbf{B}}\right\| \cdot \sqrt{\ln(2m(1 + 1/\epsilon))/\pi}.$$

Using the smoothing parameter, a number of useful lemmas can be defined.

Lemma 5 (Implicit in proof of Lemma 4.4 in [MR07])**.** *For any m-dimensional lattice $\mathcal{L} \subset \mathbb{R}^m$, reals $\epsilon \in (0, 1)$, $\alpha \geq \eta_\epsilon(\mathcal{L})$, and center $\mathbf{c} \in \mathbb{R}^m$, we have*

$$\rho_{\alpha,\mathbf{c}}(\mathcal{L}) \geq (1 - \epsilon) \cdot \frac{\alpha^m}{\det \mathcal{L}}.$$

Lemma 6 (Implicit in proof of Lemma 4.4 in [MR07])**.** *For any m-dimensional lattice $\mathcal{L} \subset \mathbb{R}^m$, reals $\epsilon \in (0, 1)$, $\alpha \geq \eta_\epsilon(\mathcal{L})$, and center $\mathbf{c} \in \mathbb{R}^m$, we have*

$$\rho_\alpha(\mathcal{L}) \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot \rho_{\alpha,\mathbf{c}}(\mathcal{L}).$$

The following Gaussian tail bound shows that the probability of sampling vectors decreases exponentially with their distance from the center.

Lemma 7 (Lemma 2.6 in [MP12], also [Ban95, MR07])**.** *For any m-dimensional lattice $\mathcal{L}$, $\alpha > \eta_\epsilon(\mathcal{L})$ for some $\epsilon \in (0, 1)$, and $\mathbf{c} \in \mathrm{span}(\mathcal{L})$ it holds that*

$$\Pr[\|\mathcal{D}_{\mathcal{L}+\mathbf{c},\alpha}\|_2 \geq \alpha\sqrt{m}] \leq 2^{-m} \cdot \frac{1 + \epsilon}{1 - \epsilon}.$$

If $\mathbf{c} = \mathbf{0}$, then the bound holds for any $\alpha > 0$ with $\epsilon = 0$.

Lemma 8 (Combination of Lemmas 5.1, 5.3 in [GPV08])**.** *Let n and q be positive integers with q prime. If $m \geq 2n \log q$, then for at least a $1 - 2q^{-n}$ fraction of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ it holds that the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$ and $\eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A})) \leq \omega(\sqrt{\log m})$ for some negligible function $\epsilon(m)$.*

Lemma 9 (Lemma 5.2 in [GPV08], see also Lemma 3.13 in [WWW22])**.** *If the columns of $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ generate $\mathbb{Z}_q^n$, and $\alpha \geq \eta_\epsilon(\mathbf{\Lambda}_q^\perp(\mathbf{A}))$ for some $\epsilon \in (0, 1/2)$, then*

$$\mathsf{SD}\big(\{\mathbf{A}, \mathbf{x}, \mathbf{Ax} \mid \mathbf{x} \leftarrow \mathcal{D}_{\mathbb{Z}^m, \alpha}\}, \{\mathbf{A}, \mathbf{x}, \mathbf{y} \mid \mathbf{y} \leftarrow\!\$\, \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathbf{A}^{-1}(\alpha, \mathbf{y})\}\big) \leq 2\epsilon.$$

3 Two-Key Lossy KMHE

At the core of our new RGS construction is a new primitive we call *two-key lossy key-and-message-homomorphic encryption* (2LKMHE). 2LKMHE is a form of symmetric-key encryption where two keys are used for encryption and decryption. Furthermore, following previous works [AHKP22, HvdHJ25], key-and-message-homomorphic means that the scheme supports the *same* homomorphism on both the key and the message space. The "lossy" means that ciphertexts can be created in two modes: The *homomorphic* mode (hom) and the *lossy* mode (lossy). In hom mode, messages can be correctly decrypted and homomorphisms are correctly applied. In lossy mode, decryption always results in a special symbol indicating lossyness and the applied homomorphisms have no effect and are "hidden". This hiding is the main purpose of the lossy mode, which will be necessary to achieve security of the RGS construction. Our lattice-based construction will tolerate a (sufficiently large, but bounded) number of rerandomizations, due to growth of the LWE error. Such a scheme will be called *levelled*.

Definition 3 (Two-Key Lossy Key-And-Message-Homomorphic Encryption)**.** A levelled *two-key lossy key-and-message-homomorphic encryption scheme* (2LKMHE) is a tuple of PPT algorithms $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ with the following syntax.

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$**:** Given the security parameter λ and a maximum rerandomization level t_{max}, this algorithm returns public parameters pp. The public parameters define a key space $\mathcal{K}$, a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, a set of key transformations $\mathcal{F}_{\mathrm{key}} = \{\sigma \mid \mathcal{K} \to \mathcal{K}\}$, a set of message transformations $\mathcal{F}_{\mathrm{msg}} = \{\tau \mid \mathcal{M} \to \mathcal{M}\}$, and a lossyness indicator message $\perp \in \mathcal{M}$. $\mathcal{F}_{\mathrm{msg}}$ and $\mathcal{F}_{\mathrm{key}}$ must contain the identity function denoted by id.

$\mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{pp})$**:** Given pp, this algorithm returns a secret key $\mathsf{sk} \in \mathcal{K}$.

$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{sk}_1, \mu, \mathsf{mode})$**:** Given pp, two secret keys $\mathsf{sk}_0, \mathsf{sk}_1$, a message $\mu \in \mathcal{M}$, and a mode $\mathsf{mode} \in \{\mathsf{hom}, \mathsf{lossy}\}$, this algorithm returns a ciphertext $\mathsf{ct} \in \mathcal{C}$.

$\mathsf{ct}' \leftarrow \mathsf{Eval}(\mathsf{pp}, \sigma_0, \sigma_1, \tau, \mathsf{ct})$**:** Given pp, a ciphertext ct and three functions $\sigma_0, \sigma_1 \in \mathcal{F}_{\mathrm{key}}$, $\tau \in \mathcal{F}_{\mathrm{msg}}$, this algorithm returns a ciphertext ct'.

$\mu' \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{ct})$**:** Given pp, two secret keys sk_0 and sk_1, and a ciphertext ct, this algorithm returns a message μ'.

In the sequel, we denote with R_{Eval} the randomness space of Eval, and with R_{Enc} the randomness space of Enc. We may omit the public parameters pp in the sequel if they are implicitly clear from the context.

3.1 Correctness

In addition to standard correctness (i.e., decryption should yield the correct message if the same keys were used for encryption), we will require an additional property that ensures correct application of key and message transformations. Furthermore, correctness depends on the encryption mode (hom/lossy) used to create the ciphertext. That is, mode hom should ensure that Eval correctly applies the transformations to the keys *and* message, while for mode lossy we want the transformations to only be applied to the keys; the message should always be the lossyness indicator $\perp$.

Definition 4 (Statistical Correctness)**.** $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is statistically correct, if for all $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $t \in [t_{\mathsf{max}}]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$, $\mu \in \mathcal{M}$, $\{(\sigma_{0,i}, \sigma_{1,i})\}_{i\in[t-1]} \in \mathcal{F}_{\text{key}}^{2(t-1)}$, $\{\tau_i\}_{i\in[t-1]} \in \mathcal{F}_{\text{msg}}^{t-1}$, $\mathsf{mode} \in \{\mathsf{hom}, \mathsf{lossy}\}$, and

$$\mu' \leftarrow \begin{cases} \tau_{t-1}(\tau_{t-2}(\dots(\mu)\dots)) & \text{if } \mathsf{mode} = \mathsf{hom} \wedge t > 1 \\ \mu & \text{if } \mathsf{mode} = \mathsf{hom} \wedge t = 1 \\ \perp & \text{if } \mathsf{mode} = \mathsf{lossy} \end{cases},$$

there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\begin{array}{c} \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}'_0, \mathsf{sk}'_1, \mathsf{ct}') \\ \neq \mu' \end{array} \;\middle|\; \begin{array}{l} \mathsf{sk}_0,\mathsf{sk}_1 \leftarrow \mathsf{KGen}(\mathsf{pp}) \\ \mathsf{sk}'_\beta \leftarrow \begin{cases} \sigma_{\beta,t-1}(\sigma_{\beta,t-2}(\dots(\mathsf{sk}_\beta)\dots)) & \text{if } t > 1 \\ \mathsf{sk}_\beta & \text{if } t = 1 \end{cases} \\ \quad \forall \beta \in \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}_0, \mathsf{sk}_1, \mu, \mathsf{mode}) \\ \mathsf{ct}' \leftarrow \mathsf{Eval}(\sigma_{0,t-1}, \sigma_{1,t-1}, \tau_{t-1}, \\ \qquad\qquad \dots \mathsf{Eval}(\sigma_{0,1}, \sigma_{1,1}, \tau_1, \mathsf{ct})\dots) \quad \text{if } t > 1 \\ \mathsf{ct}' \leftarrow \mathsf{ct} \quad \text{if } t = 1 \end{array}\right] \leq \mathsf{negl}(\lambda).$$

On the Size of the Message Space. To ensure correctness of our RGS construction later, we will want to ensure that the probability of decrypting the lossyness indicator message $\perp$ is negligible, even after applying a random transformation $\tau \in \mathcal{F}_{\text{msg}}$ to some encrypted message which could map the message to $\perp$. This is similar to the "elusive range" requirement from [LP09]. In our construction in Section 5 this will be ensured via encoding $\perp$ as the all-zero message in a large message space of size $\Omega(2^\lambda)$.

3.2 Security Properties

Key privacy. Intuitively, *key privacy* ensures that a random homomorphic transformation properly rerandomizes keys, i.e., rerandomized keys "look" like fresh ones.

Definition 5 (Statistical Key Privacy)**.** $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ has *statistical key privacy*, if for all $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$ it holds that

$$\{\mathsf{sk}, \sigma(\mathsf{sk}) \mid \mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{pp}), \sigma \leftarrow\!\!\$\, \mathcal{F}_{\text{key}}\} \overset{\text{s}}{\approx} \left\{\mathsf{sk}, \mathsf{sk}' \;\middle|\; \mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{pp}), \mathsf{sk}' \leftarrow \mathsf{KGen}(\mathsf{pp})\right\}.$$

IND-CPA security. Next we define IND-CPA security, adapted to the two-key setting. Two-key IND-CPA security is a straightforward extension of the standard one-key IND-CPA security to the two-key setting. Intuitively, a ciphertext should hide the message, if at least one of the secret keys is unknown. Our definition largely follows the one from [LP09], although we extend it by requiring ciphertexts to hide the encryption mode in addition to the message.

Definition 6 (IND-CPA Security)**.** $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is *IND-CPA-secure*, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $t_{\mathsf{max}} \in \mathbb{N}$ it holds that

$$\Pr\left[\mathsf{2LKMH\text{-}IND\text{-}CPA}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{2LKMHE}}(1^\lambda) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the 2LKMH-IND-CPA experiment is defined in Figure 3.

$\mathsf{2LKMH\text{-}IND\text{-}CPA}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{2LKMHE}}(1^\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$

$(\mathsf{sk}_0, \mathsf{sk}_1, \{(\mu_{\beta,0}, \mathsf{mode}_{\beta,0}, \mu_{\beta,1}, \mathsf{mode}_{\beta,1}, \mu_{\beta,2}, \mathsf{mode}_{\beta,2})\}_{\beta\in\{0,1\}}) \leftarrow \mathcal{A}(1^\lambda, t_{\mathsf{max}}, \mathsf{pp})$

$\mathsf{sk}'_0, \mathsf{sk}'_1 \leftarrow \mathsf{KGen}(\mathsf{pp})$

$b \leftarrow\$ \{0,1\}$

$\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{sk}_0, \mathsf{sk}'_1, \mu_{b,0}, \mathsf{mode}_{b,0})$

$\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{sk}'_0, \mathsf{sk}_1, \mu_{b,1}, \mathsf{mode}_{b,1})$

$\mathsf{ct}_2 \leftarrow \mathsf{Enc}(\mathsf{sk}'_0, \mathsf{sk}'_1, \mu_{b,2}, \mathsf{mode}_{b,2})$

$b' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk}'_0,\cdot,\cdot,\cdot),\mathsf{Enc}(\mathsf{sk}'_1,\cdot,\cdot,\cdot),\mathsf{Enc}(\cdot,\mathsf{sk}'_1,\cdot,\cdot),\mathsf{Enc}(\cdot,\mathsf{sk}'_0,\cdot,\cdot),\mathsf{Enc}(\mathsf{sk}'_0,\mathsf{sk}'_1,\cdot,\cdot),\mathsf{Enc}(\mathsf{sk}'_1,\mathsf{sk}'_0,\cdot,\cdot)}(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2)$

return $(b' = b)$

Figure 3: Definition of Game 2LKMH-IND-CPA.

KMH Rerandomizability. The last security property we will need is *KMH rerandomizability*. Intuitively, this property will provide that the *last* rerandomization of an "adversarially-generated" ciphertext yields a ciphertext that is statistically close to a rerandomization of a fresh ciphertext. More precisely, we consider a setting where an adversary generates a ciphertext and possibly rerandomizes it (once or several times) by applying the Eval algorithm "adversarially", i.e., knowing the randomness, but in an *honest-but-curious* fashion. We require that the rerandomization of such a ciphertext, using Eval with fresh randomness unknown to the adversary, is statistically indistinguishable from a ciphertext that was generated and rerandomized with independent, fresh randomness for the same number of times.

Furthermore, KMH rerandomizability allows us to move any key or message transformations applied by Eval into the initial call to Enc, as long as the last call to Eval was done with fresh randomness. This means that instead of applying the key and message transformations via Eval, one can also directly input the transformed keys and messages into Enc.

Definition 7 (Statistical KMH Rerandomizability)**.** $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is *statistically KMH-rerandomizable* if for all adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $t_{\mathsf{max}} \in \mathbb{N}$ and $\mathsf{mode} \in \{\mathsf{hom}, \mathsf{lossy}\}$ it holds that

$$\Pr\left[\mathsf{2LKMH\text{-}RR}^{t_{\mathsf{max}},\mathsf{mode}}_{\mathcal{A},\mathsf{2LKMHE}}(1^\lambda) = 1\right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the 2LKMH-RR experiment is defined in Figure 4.

4 An RGS Construction From Rerandomizable 2KMHE

We now show how any rerandomizable 2LKMHE from Section 3 can be used to construct a rerandomizable garbling scheme (RGS). We start by defining rerandomizable garbling

$\mathsf{2LKMH\text{-}RR}^{t_{\mathsf{max}},\mathsf{mode}}_{\mathcal{A},\mathsf{2LKMHE}}(1^\lambda)$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$
$(t, \mathsf{sk}_0, \mathsf{sk}_1, (\sigma_{0,i}, \sigma_{1,i}, \tau_i)_{i\in[t-1]}, \mu) \leftarrow \mathcal{A}(1^\lambda, t_{\mathsf{max}}, \mathsf{mode}, \mathsf{pp})$
// Sample adversarial randomness
$r_1 \leftarrow\$ R_{\mathsf{Enc}}; r_2, \ldots, r_{t-1} \leftarrow\$ R_{\mathsf{Eval}}$
$b \leftarrow\$ \{0,1\}$
if $b = 0$ **then**
  // Last rerandomization randomness r_t is secret and not adversarial
  $r_t \leftarrow\$ R_{\mathsf{Eval}}$
  $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}_0, \mathsf{sk}_1, \mu, \mathsf{mode}; r_1)$
  $\mathsf{ct}' \leftarrow \mathsf{Eval}(\sigma_{0,t-1}, \sigma_{1,t-1}, \tau_{t-1}, \ldots \mathsf{Eval}(\sigma_{0,1}, \sigma_{1,1}, \tau_1, \mathsf{ct}; r_2)\ldots; r_t)$
else // $b = 1$
  // Homomorphisms should be applied correctly
  $\mathsf{sk}'_\beta \leftarrow \sigma_{\beta,t-1}(\ldots \sigma_{\beta,1}(\mathsf{sk}_\beta)\ldots) \quad \forall \beta \in \{0,1\}$
  $\mu' \leftarrow \tau_{t-1}(\ldots \tau_1(\mu)\ldots)$
  // Lossy mode means message is always fixed value $\perp$
  if $\mathsf{mode} = \mathsf{lossy}$ **then** $\mu' \leftarrow \perp$ **endif**
  // Create a ciphertext with independent randomness $r'_1, \ldots, r'_t$
  $r'_1 \leftarrow\$ R_{\mathsf{Enc}};\ r'_2, \ldots, r'_t \leftarrow\$ R_{\mathsf{Eval}}$
  $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}'_0, \mathsf{sk}'_1, \mu', \mathsf{mode}; r'_1)$
  $\mathsf{ct}' \leftarrow \mathsf{Eval}(\mathsf{id}, \mathsf{id}, \mathsf{id}, \ldots \mathsf{Eval}(\mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{ct}; r'_2)\ldots; r'_t)$
endif
$b' \leftarrow \mathcal{A}(\mathsf{ct}', r_1, \ldots, r_{t-1})$
return $(b' = b \wedge 1 < t \leq t_{\mathsf{max}})$

Figure 4: Definition of Game 2LKMH-RR.

schemes and their properties in Section 4.1. In Section 4.2 we give the construction of RGS and prove it to be correct and secure in Section 4.3.

4.1 Definitions

We mostly follow the definitions in [HvdHJ25], which are themselves based on [GHV10, AHKP22]. However, due to the restrictions imposed by our LWE-based instantiation, we will define a variant of RGS that only supports a maximum number of rerandomizations, which can be specified during setup. We call this a *levelled* RGS scheme. We have also included correctness of rerandomizations into the correctness definition, which previously was not explicitly defined.

Let $(\mathcal{F}^\lambda_{\mathsf{RGS}})_{\lambda\in\mathbb{N}}$ be a family of function sets, where $\mathcal{F}^\lambda_{\mathsf{RGS}} = \{f : \mathcal{X} \to \mathcal{Y}\}$ is a set of functions f. Additionally, we define leakage function family $(\phi_\lambda)_{\lambda\in\mathbb{N}}$ to be a family of functions $\phi_\lambda : \mathcal{F}^\lambda_{\mathsf{RGS}} \to \{0,1\}^*$ which map $f \in \mathcal{F}^\lambda_{\mathsf{RGS}}$ to the allowed leakage of the garbling scheme $\phi_\lambda(f)$.

Definition 8 (Levelled Rerandomizable Garbling Scheme)**.** A (levelled) *rerandomizable garbling scheme* (RGS) for function family $(\mathcal{F}^\lambda_{\mathsf{RGS}})_{\lambda\in\mathbb{N}}$ and leakage function family $(\phi_\lambda)_{\lambda\in\mathbb{N}}$ is a tuple of PPT algorithms $\mathsf{RGS} = (\mathsf{Setup}, \mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ with the following syntax.

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$**:** This algorithm takes as input the security parameter λ and a maximum level t_{max}, and returns public parameters pp. The public parameters define a

label space $\mathcal{L}$, function family $\mathcal{F}^{\lambda}_{\mathsf{RGS}}$ and leakage function ϕ_λ.

$(\mathcal{C}, L) \leftarrow \mathsf{Gb}(\mathsf{pp}, f)$**:** Given a function $f \in \mathcal{F}^{\lambda}_{\mathsf{RGS}}$, this algorithm returns a garbled circuit $\mathcal{C}$ and a set of input labels $L \in \mathcal{L}$.

$(\mathcal{C}', \pi_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C})$**:** Given a garbled circuit $\mathcal{C}$, this algorithm returns a garbled circuit $\mathcal{C}'$ and a label encoding function $\pi_{\mathsf{En}} : \mathcal{L} \to \mathcal{L}$.

$I \leftarrow \mathsf{En}(\mathsf{pp}, L, x)$**:** This algorithm takes a set of input labels L and an input $x \in \mathcal{X}$, and outputs an input encoding I.

$y \leftarrow \mathsf{Ev}(\mathsf{pp}, \mathcal{C}, I)$**:** Given a garbled circuit $\mathcal{C}$ and an input encoding I, this algorithm outputs a $y \in \mathcal{Y}$.

An RGS must fulfill correctness, garbling privacy, and rerand privacy as defined below.

Correctness. Correctness implies that evaluation of a garbled circuit $\mathcal{C}$ on an encoding of input x should result in the correct output $f(x)$. Our definition also integrates correctness of rerandomization, meaning that correctness holds even for rerandomized garblings.

Definition 9 (Statistical Correctness)**.** $\mathsf{RGS} = (\mathsf{Setup}, \mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ is statistically correct, if for all $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $t \in [t_{\mathsf{max}}]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$, $f \in \mathcal{F}^{\lambda}_{\mathsf{RGS}}$, and $x \in \mathcal{X}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\mathsf{Ev}(\mathsf{pp}, \mathcal{C}_t, I_t) = f(x) \;\middle|\; \begin{array}{l} (\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f) \\ (\mathcal{C}_i, \pi^i_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1}) \quad \forall i \in \{2, \ldots, t\}, \\ \qquad L_t \leftarrow \begin{cases} \pi^t_{\mathsf{En}}(\ldots \pi^2_{\mathsf{En}}(L_1)\ldots) & \text{if } t > 1 \\ L_1 & \text{if } t = 1, \end{cases} \\ \qquad I_t \leftarrow \mathsf{En}(\mathsf{pp}, L_t, x), \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Note that $t = 1$ corresponds to standard correctness without any rerandomizations.

Garbling Privacy. Garbling privacy ensures that an honestly created garbling does not leak the function f nor the input x, only the output $f(x)$. This should also hold for rerandomized garblings. This is formulated using a simulator Sim that can simulate the (rerandomized) garbling given only the output $f(x)$ and information about the topology of f as given by $\phi_\lambda(f)$. The (rerandomized) garbling may also leak the number of rerandomizations done as well as the maximum number of rerandomizations allowed, as indicated by the fact that Sim is given t_{max} and t.

Definition 10 (Computational Garbling Privacy)**.** $\mathsf{RGS} = (\mathsf{Setup}, \mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ has *computational garbling privacy*, if there exists a PPT simulator Sim such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $t_{\mathsf{max}} \in \mathbb{N}$ it holds that

$$\Pr[\mathsf{GARBLE\text{-}PRIV}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{Sim},\mathsf{RGS}}(1^\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the $\mathsf{GARBLE\text{-}PRIV}$ experiment is defined in Figure 5.

Rerand Privacy. Rerand privacy is similar to rerandomizability of 2LKMHE, albeit adjusted to the setting of RGS. Specifically, it guarantees that the last honest rerandomization of an "adversarially-generated" garbling yields a garbling that is statistically close to a rerandomization of a fresh garbling. By combining this with garbling privacy, it holds that a single honest rerandomization of an honest-but-curiously created garbling suffices to reestablish simulatability, such that giving out evaluation labels for some input x will neither leak the function f, nor input x.

$\mathsf{GARBLE\text{-}PRIV}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{Sim},\mathsf{RGS}}(1^\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$
$(f, x, t) \leftarrow \mathcal{A}(1^\lambda, t_{\mathsf{max}}, \mathsf{pp})$
$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 0$ **then**
  $(\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f)$
  $(\mathcal{C}_i, \pi^i_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1})$ for $i \in \{2, \ldots, t\}$
  $L_t \leftarrow \pi^t_{\mathsf{En}}(\cdots \pi^2_{\mathsf{En}}(L_1) \cdots)$
  $I_t \leftarrow \mathsf{En}(\mathsf{pp}, L_t, x)$
else // $b = 1$
  $(\mathcal{C}_t, I_t) \leftarrow \mathsf{Sim}(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f(x), \phi_\lambda(f))$
endif
$b' \leftarrow \mathcal{A}(\mathcal{C}_t, I_t)$
return $(b' = b \wedge t \leq t_{\mathsf{max}})$

Figure 5: Definition of Game GARBLE-PRIV.

Definition 11 (Statistical Rerand Privacy)**.** $\mathsf{RGS} = (\mathsf{Setup}, \mathsf{Gb}, \mathsf{Rerand}, \mathsf{En}, \mathsf{Ev})$ has *statistical rerand privacy*, if for all adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $t_{\mathsf{max}} \in \mathbb{N}$ it holds that

$$\Pr[\mathsf{RER\text{-}PRIV}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{RGS}}(1^\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the RER-PRIV experiment is defined in Figure 6.

4.2 RGS Construction

Building Blocks. Let $\mathsf{2LKMHE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a 2LKMHE scheme with associated key space $\mathcal{K}$ and message space $\mathcal{M}$ such that $\mathcal{K} \subseteq \mathcal{M}$. We also require $\mathcal{F}_{\mathrm{key}} = \mathcal{F}_{\mathrm{msg}}$ and refer to them simply as the transformation function set $\mathcal{F}$. Define

- the function family $\mathcal{F}^\lambda_{\mathsf{RGS}} = \{f : \{0,1\}^{\ell_{\mathsf{in}}(f)} \to \{0,1\}^{\ell_{\mathsf{out}}(f)}\}$ as the set of Boolean circuits of size polynomial in λ, and
- label space $\mathcal{L}_f = (\{0,1\} \times \mathcal{K} \times \{0,1\})^{2\ell_{\mathsf{in}}(f)}$. The first value (a bit) indicates the wire value the label corresponds to, the second value (a key) is the wire label, and the third value is a pointer bit necessary for evaluation of the garbled gate.

We generally consider Boolean circuits consisting of binary gates with two inputs and one output. The fan-out is arbitrary, meaning that each gate's output wire can be connected to arbitrarily many gates. For simplicity we also assume that circuit output wires are never inputs to other gates. Given a circuit $f \in \mathcal{F}^\lambda_{\mathsf{RGS}}$, we use $\mathcal{W}$ to denote the list of wires, $\mathcal{W}_\mathcal{O}$ the list of output wires, and $\mathcal{W}_I$ the list of input wires. M denotes the number of wires and N the number of gates. A gate g_i is described by a tuple $g_i = (w_\ell, w_r, w_o, \mathsf{op})$, where w_ℓ and w_r are the left and right input wires, respectively, w_o the output wire and op is the gate function, i.e. any function $\mathsf{op} : \{0,1\} \times \{0,1\} \to \{0,1\}$. The leakage function ϕ_λ, given a Boolean circuit, outputs its topology, meaning everything but the gate operation op of each gate.

We will also need a function k_{op} that, given input $(b_1, b_2, b_3) \in \{0,1\}^3$, outputs 0 exactly if $\mathsf{op}(b_1, b_2) = b_3 = 0$, and 1 exactly if $\mathsf{op}(b_1, b_2) = b_3 = 1$, and $\perp$ otherwise. Furthermore, an encryption mode indicator function m_{op}, that, given input $(b_1, b_2, b_3) \in \{0,1\}^3$, outputs hom exactly if $\mathsf{op}(b_1, b_2) = b_3$, and lossy otherwise.

$\mathsf{RER\text{-}PRIV}^{t_{\mathsf{max}}}_{\mathcal{A},\mathsf{RGS}}(1^\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$
$(f, x, t) \leftarrow \mathcal{A}(1^\lambda, t_{\mathsf{max}}, \mathsf{pp})$
// Sample adversarial randomness
$r_1 \leftarrow\!\!\$\ R_{\mathsf{Gb}};\ r_2, \ldots, r_{t-1} \leftarrow\!\!\$\ R_{\mathsf{Rerand}}$
$b \leftarrow\!\!\$\ \{0,1\}$
if $b = 0$ **then**
  // Last garbling randomness r_t is secret and not adversarial
  $r_t \leftarrow\!\!\$\ R_{\mathsf{Rerand}}$
  $(\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f; r_1)$
  $(\mathcal{C}_i, \pi^i_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1}; r_i)$ for $i \in \{2, \ldots, t\}$
else // $b = 1$
  // Create garbling with independent randomness $r'_1, \ldots, r'_t$
  $r'_1 \leftarrow\!\!\$\ R_{\mathsf{Gb}};\ r'_2, \ldots, r'_t \leftarrow\!\!\$\ R_{\mathsf{Rerand}}$
  $(\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f; r'_1)$
  $(\mathcal{C}_i, \pi^i_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1}; r'_i)$ for $i \in \{2, \ldots, t\}$
endif
$L_t \leftarrow \pi^t_{\mathsf{En}}(\cdots \pi^2_{\mathsf{En}}(L_1) \cdots);\ I_t \leftarrow \mathsf{En}(\mathsf{pp}, L_t, x)$
$b' \leftarrow \mathcal{A}(\mathcal{C}_t, I_t, r_1, \ldots, r_{t-1})$
return $(b' = b \wedge 1 < t \leq t_{\mathsf{max}})$

Figure 6: Definition of Game RER-PRIV.

Construction 1 (RGS from 2LKMHE)**.** *Our RGS construction consists of the following algorithms:*

$\mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$**:** *Generate* 2LKMHE *public parameters* $\mathsf{pp}_{\mathsf{L}} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$ *and sample output labels* $\mathsf{sk}_0, \mathsf{sk}_1 \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathsf{L}})$*. Output*

$$\mathsf{pp} \leftarrow (\mathsf{pp}_{\mathsf{L}}, \mathsf{sk}_0, \mathsf{sk}_1)\,.$$

$\mathsf{Gb}(\mathsf{pp}, f)$**:** *Parse* f *as a series of boolean gates* $g_1, \ldots, g_N$*, ordered topologically. For every wire* $w \in \mathcal{W} \setminus \mathcal{W}_{\mathcal{O}}$ *sample labels* $\mathsf{sk}^0_w, \mathsf{sk}^1_w \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathsf{L}})$*. Let* $\mathsf{sk}^\perp_w \leftarrow \perp$ *for all* $w \in \mathcal{W}$*. The output wire labels have already been fixed at setup, specifically* $\mathsf{sk}^0_w \leftarrow \mathsf{sk}_0$ *and* $\mathsf{sk}^1_w \leftarrow \mathsf{sk}_1$ *for all* $w \in \mathcal{W}_{\mathcal{O}}$*. Next, sample random permutation bits* $b_w \leftarrow\!\!\$\ \{0,1\}$ *for each wire* $w \in \mathcal{W}$*. Then, for every gate* $g_i = (w_\ell, w_r, w_o, \mathsf{op})$*, compute the garbling of* g_i *(for readability, we omit the input* pp_{L} *to* Enc *here and in the following) by first setting*

$$\begin{pmatrix} c_{0,0,0} & c_{0,0,1} \\ c_{0,1,0} & c_{0,1,1} \\ c_{1,0,0} & c_{1,0,1} \\ c_{1,1,0} & c_{1,1,1} \end{pmatrix} \leftarrow$$

$$\begin{pmatrix} \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(0,0,0)}_{w_o}, \mathsf{m}_{\mathsf{op}}(0,0,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(0,0,1)}_{w_o}, \mathsf{m}_{\mathsf{op}}(0,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(0,1,0)}_{w_o}, \mathsf{m}_{\mathsf{op}}(0,1,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(0,1,1)}_{w_o}, \mathsf{m}_{\mathsf{op}}(0,1,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(1,0,0)}_{w_o}, \mathsf{m}_{\mathsf{op}}(1,0,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(1,0,1)}_{w_o}, \mathsf{m}_{\mathsf{op}}(1,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(1,1,0)}_{w_o}, \mathsf{m}_{\mathsf{op}}(1,1,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(1,1,1)}_{w_o}, \mathsf{m}_{\mathsf{op}}(1,1,1)) \end{pmatrix},$$

and then permute this table by setting

$$G_i \leftarrow \begin{pmatrix} c_{b_{w_\ell},b_{w_r},b_{w_o}} & c_{b_{w_\ell},b_{w_r},1\oplus b_{w_o}} \\ c_{b_{w_\ell},1\oplus b_{w_r},b_{w_o}} & c_{b_{w_\ell},1\oplus b_{w_r},1\oplus b_{w_o}} \\ c_{1\oplus b_{w_\ell},b_{w_r},b_{w_o}} & c_{1\oplus b_{w_\ell},b_{w_r},1\oplus b_{w_o}} \\ c_{1\oplus b_{w_\ell},1\oplus b_{w_r},b_{w_o}} & c_{1\oplus b_{w_\ell},1\oplus b_{w_r},1\oplus b_{w_o}} \end{pmatrix},$$

Finally, output the garbling $\mathcal{C} = (\phi_\lambda(f), G_1, \ldots, G_N)$ *and the input labels* $L = \{(0, \mathsf{sk}^0_{w_i}, b_{w_i}), (1, \mathsf{sk}^1_{w_i}, 1 \oplus b_{w_i})\}_{w_i \in \mathcal{W}_I}$.

$\mathsf{Rerand}(\mathsf{pp}, \mathcal{C})$**:** *For every wire* $w \in \mathcal{W} \setminus \mathcal{W}_\mathcal{O}$*, sample key transformations* $\sigma^0_w, \sigma^1_w \leftarrow\!\!\$\ \mathcal{F}$*. For all output wires* $w \in \mathcal{W}_\mathcal{O}$*, let* $\sigma^0_w, \sigma^1_w \leftarrow \mathsf{id}$ *be the identity function. Next, sample random permutation bits* $b_w \leftarrow\!\!\$\ \{0,1\}$ *for each wire* $w \in \mathcal{W}$*, and parse* $\mathcal{C} = (\phi_\lambda(f), G_1, \ldots, G_N)$*.*

To rerandomize the garbling G_i *of* $g_i = (w_\ell, w_r, w_o, \cdot)$ *(use* $\phi_\lambda(f)$ *to determine wire connections) for all* $i \in [N]$*, first parse*

$$G_i = \begin{pmatrix} c_{0,0,0} & c_{0,0,1} \\ c_{0,1,0} & c_{0,1,1} \\ c_{1,0,0} & c_{1,0,1} \\ c_{1,1,0} & c_{1,1,1} \end{pmatrix},$$

and then rerandomize the garbled gate by computing

$$G'_i \leftarrow \begin{pmatrix} \mathsf{Eval}(\sigma^0_{w_\ell}, \sigma^0_{w_r}, \sigma^0_{w_o}, c_{b_{w_\ell},b_{w_r},b_{w_o}}) & \mathsf{Eval}(\sigma^0_{w_\ell}, \sigma^0_{w_r}, \sigma^1_{w_o}, c_{b_{w_\ell},b_{w_r},1\oplus b_{w_o}}) \\ \mathsf{Eval}(\sigma^0_{w_\ell}, \sigma^1_{w_r}, \sigma^0_{w_o}, c_{b_{w_\ell},1\oplus b_{w_r},b_{w_o}}) & \mathsf{Eval}(\sigma^0_{w_\ell}, \sigma^1_{w_r}, \sigma^1_{w_o}, c_{b_{w_\ell},1\oplus b_{w_r},1\oplus b_{w_o}}) \\ \mathsf{Eval}(\sigma^1_{w_\ell}, \sigma^0_{w_r}, \sigma^0_{w_o}, c_{1\oplus b_{w_\ell},b_{w_r},b_{w_o}}) & \mathsf{Eval}(\sigma^1_{w_\ell}, \sigma^0_{w_r}, \sigma^1_{w_o}, c_{1\oplus b_{w_\ell},b_{w_r},1\oplus b_{w_o}}) \\ \mathsf{Eval}(\sigma^1_{w_\ell}, \sigma^1_{w_r}, \sigma^0_{w_o}, c_{1\oplus b_{w_\ell},1\oplus b_{w_r},b_{w_o}}) & \mathsf{Eval}(\sigma^1_{w_\ell}, \sigma^1_{w_r}, \sigma^1_{w_o}, c_{1\oplus b_{w_\ell},1\oplus b_{w_r},1\oplus b_{w_o}}) \end{pmatrix}.$$

Output $\mathcal{C}' = (\phi_\lambda(f), G'_1, \ldots, G'_N)$ *and* π_{En}*, where* π_{En} *is defined by*

$$\begin{aligned} \pi_{\mathsf{En}} &: \mathcal{L}_f \to \mathcal{L}_f \\ (\beta, \mathsf{sk}_{w_i}, z) &\mapsto \left(\beta, \sigma_{w_i}^{z \oplus b_{w_i}}(\mathsf{sk}_{w_i}), z \oplus b_{w_i}\right) \quad \forall \beta \in \{0,1\}, w_i \in \mathcal{W}_I. \end{aligned}$$

$\mathsf{En}(\mathsf{pp}, L, x)$**:** *Given input labels* $L = \{(0, \mathsf{sk}^0_{w_i}, z^0_{w_i}), (1, \mathsf{sk}^1_{w_i}, z^1_{w_i})\}_{w_i \in \mathcal{W}_I}$ *and an input* $x \in \{0,1\}^{\ell_{\mathsf{in}}}$*, output input encoding* $I = \{(\mathsf{sk}^{x_i}_{w_i}, z^{x_i}_{w_i})\}_{w_i \in \mathcal{W}_I}$ *(assume that* w_i *is the* i*-th input wire).*

$\mathsf{Ev}(\mathsf{pp}, \mathcal{C}, I)$**:** *Parse* $I = \{(\mathsf{sk}_{w_i}, z_{w_i})\}_{w_i \in \mathcal{W}_I}$ *and* $\mathsf{pp} = (\mathsf{pp}_{\mathsf{2LKMHE}}, \mathsf{sk}_0, \mathsf{sk}_1)$*.*

The circuit is evaluated by decrypting gates using Dec *in topological order. Decryption of each gate then provides the evaluator with the output label for that gate, which corresponds to the input labels of subsequent gates, allowing them to proceed topologically. Eventually, labels* $\mathsf{sk}_{w_{o_i}}$ *for all the output wires* $\mathcal{W}_\mathcal{O} = (w_{o_1}, \ldots, w_{o_{\ell_{\mathsf{out}}}})$ *are obtained. The output can then be decoded by checking whether each output label is equal to* sk_0 *or* sk_1*, resulting in output* $y \in \{0,1\}^{\ell_{\mathsf{out}}}$ *with* $y_i = 0$ *if* $\mathsf{sk}_{w_{o_i}} = \mathsf{sk}_0$ *and* 1 *otherwise.*

Specifically, decryption of a garbling G_i *of gate* $g_i = (w_\ell, w_r, w_o, \cdot)$ *works as follows. Parse*

$$G_i = \begin{pmatrix} c_{0,0,0} & c_{0,0,1} \\ c_{0,1,0} & c_{0,1,1} \\ c_{1,0,0} & c_{1,0,1} \\ c_{1,1,0} & c_{1,1,1} \end{pmatrix},$$

and assume we have obtained label $(\mathsf{sk}_{w_\ell}, z_{w_\ell})$ *for the left input wire* w_ℓ *and a wire label* $(\mathsf{sk}_{w_r}, z_{w_r})$ *for the right input wire* w_r *from decrypting the (topologically) previous gates. Then compute*

$$\mu_0 \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{w_\ell}, \mathsf{sk}_{w_r}, c_{z_{w_\ell}, z_{w_r}, 0}) \quad \textit{and} \quad \mu_1 \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_{w_\ell}, \mathsf{sk}_{w_r}, c_{z_{w_\ell}, z_{w_r}, 1})$$

The output label is then $(\mu_0, 0)$ *if* $\mu_0 \neq \perp$ *and* $(\mu_1, 1)$ *if* $\mu_1 \neq \perp$*. Abort if neither case (or both) holds.*

Note that z_{w_ℓ} *and* z_{w_r} *act as "pointer bits", as in the point-and-permute technique by Beaver, Micali, and Rogaway [BMR90]. Thus, the evaluator needs to decrypt only one row of every garbled gate* G_i*.*

4.3 Correctness and Security

For the proofs of correctness and rerand-privacy, the following invariant about the structure of rerandomized garblings will be useful.

Lemma 10 (Rerandomization Invariant)**.** *For any* $\lambda, t_{\mathsf{max}} \in \mathbb{N}$*,* $t \in [t_{\mathsf{max}}]$*,* $f \in \mathcal{F}^{\lambda}_{\mathsf{RGS}}$*, and* $x \in \{0,1\}^{\ell_{\mathsf{in}}(f)}$*, let* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$*,* $(\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f; r_1)$*, and* $(\mathcal{C}_i, \pi^i_{\mathsf{En}}) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1}; r_i)$ *for* $i \in \{2, \ldots, t\}$*, where* $\mathsf{pp} = (\mathsf{pp}_{\mathsf{L}}, \mathsf{sk}_0, \mathsf{sk}_1)$*, the values*

$$r_1 = \left(\{\mathsf{sk}^0_w, \mathsf{sk}^1_w\}_{w \in \mathcal{W} \setminus \mathcal{W}_{\mathcal{O}}}, \{b^{(1)}_w\}_{w \in \mathcal{W}}\right)$$
$$r_i = \left(\{\sigma^0_{w,i-1}, \sigma^1_{w,i-1}\}_{w \in \mathcal{W} \setminus \mathcal{W}_{\mathcal{O}}}, \{b^{(i)}_w\}_{w \in \mathcal{W}}\right) \quad \forall i \in \{2, \ldots, t\}$$

represent the randomness used (aside from the randomness of the calls to Enc *and* Eval*, which we leave implicit), and*

$$\mathsf{sk}^0_w \leftarrow \mathsf{sk}_0,\ \mathsf{sk}^1_w \leftarrow \mathsf{sk}_1\ \forall w \in \mathcal{W}_{\mathcal{O}},\ \mathsf{sk}^{\perp}_w \leftarrow \perp\ \forall w \in \mathcal{W},\ \textit{and}$$
$$\sigma^0_{w,i-1}, \sigma^1_{w,i-1} \leftarrow \mathsf{id}\ \forall w \in \mathcal{W}_{\mathcal{O}}, i \in \{2, \ldots, t\}$$

denote the missing keys and transformation functions which are not chosen randomly. Also define

$$\bar{b}^{(i)}_w = b^{(1)}_w \oplus \cdots \oplus b^{(i)}_w.$$

Parse $\mathcal{C}_i = (\cdot, G^{(i)}_1, \ldots, G^{(i)}_N)$ *for all* $i \in [t]$*, and then write*

$$G^{(i)}_j = \begin{pmatrix} c^{(i,j)}_{0,0,0} & c^{(i,j)}_{0,0,1} \\ c^{(i,j)}_{0,1,0} & c^{(i,j)}_{0,1,1} \\ c^{(i,j)}_{1,0,0} & c^{(i,j)}_{1,0,1} \\ c^{(i,j)}_{1,1,0} & c^{(i,j)}_{1,1,1} \end{pmatrix} \quad \forall i \in [t], j \in [N].$$

Then, for each $j \in [N]$ *and* $(a_1, a_2, a_3) \in \{0,1\}^3$*, denoting* $g_j = (w_\ell, w_r, w_o, \mathsf{op})$*,* $c^{(t,j)}_{a_1,a_2,a_3}$ *is given by*

$$\mathsf{Eval}(\sigma^{a_1}_{w_\ell,t-1}, \sigma^{a_2}_{w_r,t-1}, \sigma^{a_3}_{w_o,t-1},$$
$$\mathsf{Eval}(\sigma^{a_1 \oplus \bar{b}^{(t)}_{w_\ell} \oplus \bar{b}^{(t-1)}_{w_\ell}}_{w_\ell,t-2}, \sigma^{a_2 \oplus \bar{b}^{(t)}_{w_r} \oplus \bar{b}^{(t-1)}_{w_r}}_{w_r,t-2}, \sigma^{a_3 \oplus \bar{b}^{(t)}_{w_o} \oplus \bar{b}^{(t-1)}_{w_o}}_{w_o,t-2}, \ldots$$
$$\mathsf{Eval}(\sigma^{a_1 \oplus \bar{b}^{(t)}_{w_\ell} \oplus \bar{b}^{(2)}_{w_\ell}}_{w_\ell,1}, \sigma^{a_2 \oplus \bar{b}^{(t)}_{w_r} \oplus \bar{b}^{(2)}_{w_r}}_{w_r,1}, \sigma^{a_3 \oplus \bar{b}^{(t)}_{w_o} \oplus \bar{b}^{(2)}_{w_o}}_{w_o,1},$$
$$\mathsf{Enc}(\mathsf{sk}^{a_1 \oplus \bar{b}^{(t)}_{w_\ell}}_{w_\ell}, \mathsf{sk}^{a_2 \oplus \bar{b}^{(t)}_{w_r}}_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(a_1 \oplus \bar{b}^{(t)}_{w_\ell}, a_2 \oplus \bar{b}^{(t)}_{w_r}, a_3 \oplus \bar{b}^{(t)}_{w_o})}_{w_o},$$
$$\mathsf{m}_{\mathsf{op}}(a_1 \oplus \bar{b}^{(t)}_{w_\ell}, a_2 \oplus \bar{b}^{(t)}_{w_r}, a_3 \oplus \bar{b}^{(t)}_{w_o}))) \ldots)),$$

where for $t = 1$ the expression consists only of the call to Enc*, and* $L_i = \pi^i_{\mathsf{En}}(\dots \pi^2_{\mathsf{En}}(L_1)\dots)$ *is given by*

$$L_i = \left\{ \left(0, \sigma^{\bar{b}^{(i)}_w}_{w,i-1}(\dots \sigma^{\bar{b}^{(2)}_w}_{w,1}(\mathsf{sk}^0_w)\dots), \bar{b}^{(i)}_w\right), \left(1, \sigma^{1\oplus\bar{b}^{(i)}_w}_{w,i-1}(\dots \sigma^{1\oplus\bar{b}^{(2)}_w}_{w,1}(\mathsf{sk}^1_w)\dots), 1\oplus\bar{b}^{(i)}_w\right)\right\}_{w\in\mathcal{W}_I}$$

which implies that $I_t \leftarrow \mathsf{En}(\mathsf{pp}, L_t, x)$ *is given by*

$$I_t = \left\{ \left(\sigma^{x_i\oplus\bar{b}^{(t)}_{w_i}}_{w_i,t-1}(\dots \sigma^{x_i\oplus\bar{b}^{(2)}_{w_i}}_{w_i,1}(\mathsf{sk}^{x_i}_{w_i})\dots), x_i\oplus\bar{b}^{(t)}_{w_i}\right)\right\}_{w_i\in\mathcal{W}_I}.$$

Proof. We start with the claimed structure of the garbled gates. Fix $j \in [N]$ with $g_j = (w_\ell, w_r, w_o, \mathsf{op})$ and a position $(a_1, a_2, a_3) \in \{0,1\}^3$, and let $\beta_1 \coloneqq a_1 \oplus \bar{b}^{(t)}_{w_\ell}$, $\beta_2 \coloneqq a_2 \oplus \bar{b}^{(t)}_{w_r}$, and $\beta_3 \coloneqq a_3 \oplus \bar{b}^{(t)}_{w_o}$. Since $a_1 \oplus \bar{b}^{(t)}_{w_\ell} \oplus \bar{b}^{(i)}_{w_\ell} = \beta_1 \oplus \bar{b}^{(i)}_{w_\ell}$ for all $i \in [t]$ (and similarly for w_r and w_o), the claimed expression for $c^{(t,j)}_{a_1,a_2,a_3}$ is ct_t, where we define

$$\begin{aligned}
\mathsf{ct}_1 &\coloneqq \mathsf{Enc}\Big(\mathsf{sk}^{\beta_1}_{w_\ell}, \mathsf{sk}^{\beta_2}_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3)}_{w_o}, \mathsf{m}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3)\Big),\\
\mathsf{ct}_i &\coloneqq \mathsf{Eval}\Big(\sigma^{\beta_1\oplus\bar{b}^{(i)}_{w_\ell}}_{w_\ell,i-1}, \sigma^{\beta_2\oplus\bar{b}^{(i)}_{w_r}}_{w_r,i-1}, \sigma^{\beta_3\oplus\bar{b}^{(i)}_{w_o}}_{w_o,i-1}, \mathsf{ct}_{i-1}\Big) \quad \forall i \in \{2,\dots,t\}.
\end{aligned}$$

Note that for $i = t$, the superscripts of the transformations are $\beta_1 \oplus \bar{b}^{(t)}_{w_\ell} = a_1$, $\beta_2 \oplus \bar{b}^{(t)}_{w_r} = a_2$, and $\beta_3 \oplus \bar{b}^{(t)}_{w_o} = a_3$, as desired. We show by induction over $i \in [t]$ that

$$c^{(i,j)}_{\beta_1\oplus\bar{b}^{(i)}_{w_\ell},\beta_2\oplus\bar{b}^{(i)}_{w_r},\beta_3\oplus\bar{b}^{(i)}_{w_o}} = \mathsf{ct}_i, \tag{1}$$

which for $i = t$ proves the claim, as the position is then $(\beta_1 \oplus \bar{b}^{(t)}_{w_\ell}, \beta_2 \oplus \bar{b}^{(t)}_{w_r}, \beta_3 \oplus \bar{b}^{(t)}_{w_o}) = (a_1, a_2, a_3)$.

For $i = 1$, by definition of Gb, any position $(a'_1, a'_2, a'_3) \in \{0,1\}^3$ of the unpermuted $G^{(1)}_j$ contains the entry

$$\mathsf{Enc}(\mathsf{sk}^{a'_1}_{w_\ell}, \mathsf{sk}^{a'_2}_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(a'_1,a'_2,a'_3)}_{w_o}, \mathsf{m}_{\mathsf{op}}(a'_1, a'_2, a'_3)),$$

such that, after permutation, the same position contains

$$\mathsf{Enc}(\mathsf{sk}^{a'_1\oplus b^{(1)}_{w_\ell}}_{w_\ell}, \mathsf{sk}^{a'_2\oplus b^{(1)}_{w_r}}_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(a'_1\oplus b^{(1)}_{w_\ell}, a'_2\oplus b^{(1)}_{w_r}, a'_3\oplus b^{(1)}_{w_o})}_{w_o}, \mathsf{m}_{\mathsf{op}}(a'_1 \oplus b^{(1)}_{w_\ell}, a'_2 \oplus b^{(1)}_{w_r}, a'_3 \oplus b^{(1)}_{w_o})).$$

Since $\bar{b}^{(1)}_w = b^{(1)}_w$ for all $w \in \mathcal{W}$, this shows Equation (1) for $i = 1$.

For $i \in \{2,\dots,t\}$, by definition of Rerand with randomness r_i, the ciphertext at position (a'_1, a'_2, a'_3) of $G^{(i)}_j$ is computed as

$$c^{(i,j)}_{a'_1,a'_2,a'_3} = \mathsf{Eval}(\sigma^{a'_1}_{w_\ell,i-1}, \sigma^{a'_2}_{w_r,i-1}, \sigma^{a'_3}_{w_o,i-1}, c^{(i-1,j)}_{a'_1\oplus b^{(i)}_{w_\ell}, a'_2\oplus b^{(i)}_{w_r}, a'_3\oplus b^{(i)}_{w_o}}).$$

For position $(a'_1, a'_2, a'_3) = (\beta_1\oplus\bar{b}^{(i)}_{w_\ell}, \beta_2\oplus\bar{b}^{(i)}_{w_r}, \beta_3\oplus\bar{b}^{(i)}_{w_o})$, we then have that $c^{(i,j)}_{\beta_1\oplus\bar{b}^{(i)}_{w_\ell},\beta_2\oplus\bar{b}^{(i)}_{w_r},\beta_3\oplus\bar{b}^{(i)}_{w_o}}$ is computed by applying Eval with transformations $\sigma^{\beta_1\oplus\bar{b}^{(i)}_{w_\ell}}_{w_\ell,i-1}$, $\sigma^{\beta_2\oplus\bar{b}^{(i)}_{w_r}}_{w_r,i-1}$, and $\sigma^{\beta_3\oplus\bar{b}^{(i)}_{w_o}}_{w_o,i-1}$ to the ciphertext of $G^{(i-1)}_j$ at position

$$\begin{aligned}
&\left(\beta_1 \oplus \bar{b}^{(i)}_{w_\ell} \oplus b^{(i)}_{w_\ell}, \beta_2 \oplus \bar{b}^{(i)}_{w_r} \oplus b^{(i)}_{w_r}, \beta_3 \oplus \bar{b}^{(i)}_{w_o} \oplus b^{(i)}_{w_o}\right)\\
&\quad = \left(\beta_1 \oplus \bar{b}^{(i-1)}_{w_\ell}, \beta_2 \oplus \bar{b}^{(i-1)}_{w_r}, \beta_3 \oplus \bar{b}^{(i-1)}_{w_o}\right).
\end{aligned}$$

By the induction hypothesis, the latter ciphertext is ct_{i-1}, so the former ciphertext is exactly ct_i.

Next we show that the labels have the claimed form, also by induction over $i \in [t]$. For $i = 1$, the claim is $L_1 = \{(0, \mathsf{sk}_w^0, b_w^{(1)}), (1, \mathsf{sk}_w^1, 1 \oplus b_w^{(1)})\}_{w \in \mathcal{W}_I}$, which are exactly the labels output by Gb. For $i \in \{2, \ldots, t\}$, fix $w \in \mathcal{W}_I$ and $\beta \in \{0, 1\}$. By the induction hypothesis, L_{i-1} contains the label $(\beta, \mathsf{sk}, \beta \oplus \bar{b}_w^{(i-1)})$ with $\mathsf{sk} = \sigma_{w,i-2}^{\beta \oplus \bar{b}_w^{(i-1)}}(\ldots \sigma_{w,1}^{\beta \oplus \bar{b}_w^{(2)}}(\mathsf{sk}_w^\beta)\ldots)$ (where $\mathsf{sk} = \mathsf{sk}_w^\beta$ if $i = 2$). By definition of π_{En}^i, which is output by Rerand with randomness r_i, this label is mapped to

$$\left(\beta, \sigma_{w,i-1}^{\beta \oplus \bar{b}_w^{(i-1)} \oplus b_w^{(i)}}(\mathsf{sk}), \beta \oplus \bar{b}_w^{(i-1)} \oplus b_w^{(i)}\right) = \left(\beta, \sigma_{w,i-1}^{\beta \oplus \bar{b}_w^{(i)}}(\mathsf{sk}), \beta \oplus \bar{b}_w^{(i)}\right),$$

which is the claimed label for wire value β in L_i. The claimed form of I_t then follows directly from the definition of En. □

Theorem 1. *Assuming* 2LKMHE *is statistically correct (Definition 4) and has message space size $O(1/\mathsf{negl}(\lambda))$, Construction 1 is statistically correct (Definition 9) as well.*

Proof. For every wire $w_i \in \mathcal{W}$, let $f(x)_i$ denote the value on the wire w_i when evaluating the circuit f on the input x. Then by definition of Ev and using Lemma 10, the evaluation of a garbling $G_j^{(t)}$ of gate $g_j = (w_{i_1}, w_{i_2}, w_{i_3}, \mathsf{op})$ with left and right input wire labels

$$\left(\sigma_{w_{i_1},t-1}^{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}}(\ldots \sigma_{w_{i_1},1}^{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(2)}}(\mathsf{sk}_{w_{i_1}}^{f(x)_{i_1}})\ldots), f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}\right), \text{ and}$$

$$\left(\sigma_{w_{i_2},t-1}^{f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}}(\ldots \sigma_{w_{i_2},1}^{f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(2)}}(\mathsf{sk}_{w_{i_2}}^{f(x)_{i_2}})\ldots), f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}\right)$$

implies that we use those labels to decrypt the ciphertexts $c^{(t,j)}_{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}, f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}, 0}$ and $c^{(t,j)}_{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}, f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}, 1}$. By Lemma 10, for $a_3 \in \{0, 1\}$, those ciphertexts are given by

$$\begin{aligned} c^{(t,j)}_{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}, f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}, a_3} = \mathsf{Enc}\Big(&\sigma_{w_{i_1},t-1}^{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}}(\ldots \sigma_{w_{i_1},1}^{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(2)}}(\mathsf{sk}_{w_{i_1}}^{f(x)_{i_1}})\ldots), \\ &\sigma_{w_{i_2},t-1}^{f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}}(\ldots \sigma_{w_{i_2},1}^{f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(2)}}(\mathsf{sk}_{w_{i_2}}^{f(x)_{i_2}})\ldots), \\ &\sigma_{w_{i_3},t-1}^{a_3}(\ldots \sigma_{w_{i_3},1}^{a_3 \oplus \bar{b}_{w_{i_3}}^{(t)} \oplus \bar{b}_{w_{i_3}}^{(2)}}(\mathsf{sk}_{w_{i_3}}^{\mathsf{k}_{\mathsf{op}}(f(x)_{i_1}, f(x)_{i_2}, a_3 \oplus \bar{b}_{w_{i_3}}^{(t)})})\ldots), \\ &\mathsf{m}_{\mathsf{op}}(f(x)_{i_1}, f(x)_{i_2}, a_3 \oplus \bar{b}_{w_{i_3}}^{(t)})\Big). \end{aligned}$$

Note that in the term above we collapse the Eval functions into the Enc term only to improve readability, the rest of the argument, i.e., application of correctness of 2LKMHE, also works if we keep the nested calls to Eval from Lemma 10. Now we need to argue that we always obtain the right output label belonging to the output wire value $f(x)_{i_3}$. First, note that if $a_3 \oplus \bar{b}_{w_{i_3}}^{(t)} \neq f(x)_{i_3} = \mathsf{op}(f(x)_{i_1}, f(x)_{i_2})$, then $\mathsf{m}_{\mathsf{op}}(f(x)_{i_1}, f(x)_{i_2}, a_3 \oplus \bar{b}_{w_{i_3}}^{(t)}) = \mathsf{lossy}$, which means that the label obtained from decrypting $c^{(t,j)}_{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}, f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}, a_3}$ is, by correctness of 2LKMHE (Definition 4), with overwhelming probability, $\mathsf{sk}_{w_{i_3}}^{\perp} = \perp$.

On the other hand, if $a_3 \oplus \bar{b}_{w_{i_3}}^{(t)} = f(x)_{i_3} = \mathsf{op}(f(x)_{i_1}, f(x)_{i_2})$, then $\mathsf{m}_{\mathsf{op}}(f(x)_{i_1}, f(x)_{i_2}, a_3 \oplus \bar{b}_{w_{i_3}}^{(t)}) = \mathsf{hom}$ and $\mathsf{k}_{\mathsf{op}}(f(x)_{i_1}, f(x)_{i_2}, a_3 \oplus \bar{b}_{w_{i_3}}^{(t)}) = a_3 \oplus \bar{b}_{w_{i_3}}^{(t)} = f(x)_{i_3}$, which means the label obtained from decrypting $c^{(t,j)}_{f(x)_{i_1} \oplus \bar{b}_{w_{i_1}}^{(t)}, f(x)_{i_2} \oplus \bar{b}_{w_{i_2}}^{(t)}, a_3}$ is exactly

$$\sigma_{w_{i_3},t-1}^{f(x)_{i_3} \oplus \bar{b}_{w_{i_3}}^{(t)}}(\ldots \sigma_{w_{i_3},1}^{f(x)_{i_3} \oplus \bar{b}_{w_{i_3}}^{(2)}}(\mathsf{sk}_{w_{i_3}}^{f(x)_{i_3}})\ldots),$$

which implies that the output label obtained (including pointer bit) is given by

$$\left(\sigma_{w_{i_3},t-1}^{f(x)_{i_3}\oplus\bar{b}_{w_{i_3}}^{(t)}}(\dots\sigma_{w_{i_3},1}^{f(x)_{i_3}\oplus\bar{b}_{w_{i_3}}^{(2)}}(\mathsf{sk}_{w_{i_3}}^{f(x)_{i_3}})\dots), a_3 = f(x)_{i_3}\oplus\bar{b}_{w_{i_3}}^{(t)}\right).$$

Note that the result of decryption will not be $\perp$ with overwhelming probability (in λ) as long as the message space of $\mathsf{2LKMHE}$ has size $O(1/\mathsf{negl}(\lambda))$, as discussed previously. Then, successively applying this to all evaluated garbled gates and using the fact that $\sigma_{w,i}^{b} = \mathsf{id}$ for all $w \in \mathcal{W}_{\mathcal{O}}$, $i \in [t-1]$, $b \in \{0,1\}$, and the size of f is polynomial in λ, we finally obtain the circuit output labels $\mathsf{sk}_{w_{i_3}}^{f(x)_{i_3}}$ for every circuit output gate $g_j = (\cdot,\cdot,w_{i_3},\cdot)$ with overwhelming probability, which implies that output label decoding results in Ev outputting $f(x)$, as desired. □

Theorem 2. *Assuming* $\mathsf{2LKMHE}$ *has IND-CPA security (Definition 6), Construction 1 has (computational) garbling privacy (Definition 10).*

Proof. We need to show that the simulator Sim can simulate the garbled circuit $\mathcal{C}$ for a function $f \in \mathcal{F}_{\mathsf{RGS}}^{\lambda}$ and input labels I for an input $x \in \mathcal{X}$, but without knowing either the function f or the input x. The issue with computing a real garbling without knowledge of f or x is that Sim cannot evaluate the functions k and m, and also is not able to compute the correct input labels for x. It does however know the topology of the circuit of f due to $\phi_\lambda(f)$. Concretely, $\mathsf{Sim}(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f(x), \phi_\lambda(f))$ proceeds as follows:

1. Sim runs $(\mathcal{C}_1, L_1) \leftarrow \mathsf{Gb}(\mathsf{pp}, f)$ as usual, but makes the following changes to ensure ϕ_λ suffices instead of using the unknown (to the simulator) function f and input x:
   - It programs the output labels such that obtaining sk_w^0 for $w \in \mathcal{W}_{\mathcal{O}}$ results in the output $f(x)$. Specifically, for the j-th output wire of f, it sets $\mathsf{sk}_{w_j}^0 \leftarrow \mathsf{sk}_{f(x)_j}$ and $\mathsf{sk}_{w_j}^1 \leftarrow \mathsf{sk}_{1\oplus f(x)_j}$, where $f(x)_j$ is the j-th bit of $f(x)$.
   - Next it ensures that evaluating the garbling on input $0^{\ell_{\mathsf{in}}}$ results in the evaluator obtaining labels sk_w^0 on all wires $w \in \mathcal{W}$. Concretely, when garbling any gate $g_i = (w_\ell, w_r, w_o, \mathsf{op})$, Sim replaces the unknown op function as used for m_{op} and k_{op} with the constant-zero function. Note that this also removes the need for knowing the (unknown) function f during garbling.
2. Lastly, Sim computes $(\mathcal{C}_i, \pi_{\mathsf{En}}^i) \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathcal{C}_{i-1})$ for $i \in \{2,\dots,t\}$, updates input labels $L_t \leftarrow \pi_{\mathsf{En}}^t(\cdots\pi_{\mathsf{En}}^2(L_1)\cdots)$, sets the input encoding to use the active input labels via $I_t \leftarrow \mathsf{En}(\mathsf{pp}, L_t, 0^{\ell_{\mathsf{in}}})$ (removes reliance on the unknown input x), and outputs $(\mathcal{C}_t, I_t)$. The only change in this step, compared to case $b = 0$ of the $\mathsf{GARBLE\text{-}PRIV}$ game, is that we encode the all-zero string $0^{\ell_{\mathsf{in}}}$ instead of the real x.

We now prove that the simulation is (computationally) indistinguishable from case $b = 0$ of the $\mathsf{GARBLE\text{-}PRIV}$ game (Figure 5), using a sequence of hybrids. The proof largely follows the strategy from Lindell and Pinkas [LP09] by replacing the simulated garbled gates by real ones, until the entire garbled circuit is as in a real garbling, i.e., case $b = 0$ of the $\mathsf{GARBLE\text{-}PRIV}$ game.

H_0**:** The output of the simulator Sim as described above.

$H_1,\dots,H_N$**:** Each $H_i \in \{H_1,\dots,H_N\}$ is equal to H_{i-1} but we change how the m and k functions used during garbling of gate g_{N-i+1} are defined. Concretely, we compute the values on the wires of f when evaluated on input x and denote these by γ_w for each $w \in \mathcal{W}$. Then compute the unpermuted garbling of $g_{N-i+1} = (w_\ell, w_r, w_o, \mathsf{op})$

as

$$\begin{pmatrix} c_{0,0,0} & c_{0,0,1} \\ c_{0,1,0} & c_{0,1,1} \\ c_{1,0,0} & c_{1,0,1} \\ c_{1,1,0} & c_{1,1,1} \end{pmatrix} \leftarrow$$

$$\begin{pmatrix} \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,0,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,0,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,0,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,1)) \end{pmatrix},$$

where $\hat{\mathsf{m}}_{\mathsf{op}}(b_1,b_2,b_3) \coloneqq \mathsf{m}_{\mathsf{op}}(b_1 \oplus \gamma_{w_\ell}, b_2 \oplus \gamma_{w_r}, b_3 \oplus \gamma_{w_o})$ and $\hat{\mathsf{k}}_{\mathsf{op}}(b_1,b_2,b_3) \coloneqq \mathsf{k}_{\mathsf{op}}(b_1 \oplus \gamma_{w_\ell}, b_2 \oplus \gamma_{w_r}, b_3 \oplus \gamma_{w_o}) \oplus \gamma_{w_o}$. Finally, permute as before.

$\hat{H}$: In this hybrid we perform a key re-labelling to make sure I_t is an input encoding of the real input x. Concretely, we replace all uses of some key sk^b_w, for some $b \in \{0,1\}, w \in \mathcal{W}$, with the key $\mathsf{sk}^{b \oplus \gamma_w}_w$.

H': This is case $b = 0$ of the GARBLE-PRIV game, i.e., a real garbling and rerandomization.

We proceed with showing computational indistinguishability of the sequence of hybrids.

Lemma 11. *For each $i \in \{1, 2, \ldots, N\}$, assuming* 2LKMHE *has IND-CPA security, hybrids H_{i-1} and H_i are computationally indistinguishable.*

Proof. The only difference between H_{i-1} and H_i is how the gate g_{N-i+1} is garbled. In H_{i-1} the unpermuted G_{N-i+1} is computed as

$$\begin{pmatrix} \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^0_{w_o}, \mathsf{hom}) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \bot, \mathsf{lossy}) \\ \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^0_{w_o}, \mathsf{hom}) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \bot, \mathsf{lossy}) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^0_{w_o}, \mathsf{hom}) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \bot, \mathsf{lossy}) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^0_{w_o}, \mathsf{hom}) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \bot, \mathsf{lossy}) \end{pmatrix}$$

while in H_i it is

$$\begin{pmatrix} \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,0,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,0,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,0,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,1)) \end{pmatrix} =$$

$$\begin{pmatrix} \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^0_{w_o}, \mathsf{hom}) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^0_{w_r}, \bot, \mathsf{lossy}) \\ \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,0)) & \mathsf{Enc}(\mathsf{sk}^0_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(0,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(0,1,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^0_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,0,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,0,1)) \\ \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,0)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,0)) & \mathsf{Enc}(\mathsf{sk}^1_{w_\ell}, \mathsf{sk}^1_{w_r}, \mathsf{sk}^{\hat{\mathsf{k}}_{\mathsf{op}}(1,1,1)}_{w_o}, \hat{\mathsf{m}}_{\mathsf{op}}(1,1,1)) \end{pmatrix},$$

due to the fact that $\hat{\mathsf{m}}_{\mathsf{op}}(0,0,0) = \mathsf{hom}$ (since $\mathsf{op}(\gamma_{w_\ell}, \gamma_{w_r}) = \gamma_{w_o}$ holds by definition of the γ-values) and $\hat{\mathsf{k}}_{\mathsf{op}}(0,0,0) = 0$ (for the same reason), and similarly $\hat{\mathsf{m}}_{\mathsf{op}}(0,0,1) = \mathsf{lossy}$ and $\hat{\mathsf{k}}_{\mathsf{op}}(0,0,1) = \bot$. Note that in both H_{i-1} and H_i the first (unpermuted) row, i.e., the *active* row, is the same, while the other rows may not be the same. Therefore, it suffices to show that the difference in the second to fourth rows is indistinguishable to a PPT adversary.

To show indistinguishability of H_{i-1} and H_i we introduce an intermediate hybrid, called $\bar{H}_i$. In $\bar{H}_i$ only the left column of the unpermuted G_{N-i+1} is computed as in H_i, the right column is as in H_{i-1}. We now employ two reductions to IND-CPA security of

2LKMHE, one for indistinguishability of H_{i-1} and $\bar{H}_i$ and one for $\bar{H}_i$ and H_i. Due to the topological ordering of the gates, in H_i, $\bar{H}_i$, and H_{i-1} all gates whose output wire is an input wire of g_{N-i+1} are still simulated as in H_0. This means that the keys $\mathsf{sk}^1_{w_\ell}$ and $\mathsf{sk}^1_{w_r}$ are not encrypted as messages in any garbled gate ciphertexts. Therefore, those keys only occur as encryption keys in gates that have w_ℓ and/or w_r as input. By treating them as the secret keys in the IND-CPA game, the reductions can then use their IND-CPA challengers to simulate the second to fourth rows of either H_{i-1} and $\bar{H}_i$, or $\bar{H}_i$ and H_i, respectively. All other gates that share at least one input wire w_ℓ and/or w_r with g_{N-i+1} can be simulated using the IND-CPA game's encryption oracles. □

Lemma 12. *Hybrids H_N and $\hat{H}$ are identically distributed.*

Proof. For every output wire $w_j \in \mathcal{W}_\mathcal{O}$, it holds that $\mathsf{sk}^b_{w_j} = \mathsf{sk}_{b\oplus f(x)_j} = \mathsf{sk}_{b\oplus\gamma_{w_j}}$ in H_N, so after the re-labelling, $\mathsf{sk}^0_{w_j} = \mathsf{sk}_0$ and $\mathsf{sk}^1_{w_j} = \mathsf{sk}_1$, as in a real garbling. Lastly, for every input wire $w_i \in \mathcal{W}_I$, I_t contains the rerandomized version of the key that was called $\mathsf{sk}^0_{w_i}$ in H_N, which is now called $\mathsf{sk}^{\gamma_{w_i}}_{w_i} = \mathsf{sk}^{x_i}_{w_i}$, so I_t is now an encoding of the real input x. Since only the names of the keys change, H_N and $\hat{H}$ are identically distributed. □

Lemma 13. *Hybrids $\hat{H}$ and H' are identically distributed.*

Proof. The only difference between $\hat{H}$ and H' lies in the permutation bits. After the re-labelling of the keys, it is not hard to see, using H_N and $\hat{H}$, that the ciphertext at position $(a_1, a_2, a_3) \in \{0,1\}^3$ of G_i in $\hat{H}$ is (combining encryption and permutation into a single step)

$$\mathsf{Enc}(\mathsf{sk}^{\beta_1}_{w_\ell}, \mathsf{sk}^{\beta_2}_{w_r}, \mathsf{sk}^{\mathsf{k}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3)}_{w_o}, \mathsf{m}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3)),$$

where $\beta_1 \coloneqq a_1 \oplus b_{w_\ell} \oplus \gamma_{w_\ell}$, $\beta_2 \coloneqq a_2 \oplus b_{w_r} \oplus \gamma_{w_r}$, and $\beta_3 \coloneqq a_3 \oplus b_{w_o} \oplus \gamma_{w_o}$. In H', the ciphertext at position (a_1, a_2, a_3) is computed in the same way, but with $\beta_1 \coloneqq a_1 \oplus b_{w_\ell}$, $\beta_2 \coloneqq a_2 \oplus b_{w_r}$, and $\beta_3 \coloneqq a_3 \oplus b_{w_o}$. Similarly, for every input wire $w \in \mathcal{W}_I$ and wire value $\beta \in \{0,1\}$, the pointer bit of the input label containing sk^β_w is $\beta \oplus \gamma_w \oplus b_w$ in $\hat{H}$ and $\beta \oplus b_w$ in H'. Hence, $\hat{H}$ is exactly H' with every permutation bit b_w replaced by $b_w \oplus \gamma_w$, consistently for all gates and input labels, while the subsequent rerandomizations and the input encoding are computed in the same way in both hybrids. Since the bits b_w are sampled uniformly and independently for all wires $w \in \mathcal{W}$, and the γ_w are fixed values that only depend on f and x, the bits $b_w \oplus \gamma_w$ are also uniformly and independently distributed. Therefore, $\hat{H}$ and H' are identically distributed. □

We have now shown the simulation Sim, i.e., case $b = 1$, and case $b = 0$ to be computationally indistinguishable, concluding the proof of Theorem 2. □

Theorem 3. *Assuming* 2LKMHE *has statistical key-privacy (Definition 5), and statistical KMH rerandomizability (Definition 7), Construction 1 has statistical rerand privacy (Definition 11).*

Proof. The main idea of the proof is to show that both cases of the RER-PRIV game are statistically close to a garbling that is computed from fresh wire labels and permutation bits that are independent of the randomness $r_1, \ldots, r_{t-1}$ given to $\mathcal{A}$. For this, we first use KMH rerandomizability to collapse the transformations applied to each ciphertext into its keys and message. Then, we use Lemma 10 to argue that each label transformation is applied to exactly one label, and use the secret randomness r_t of the last rerandomization, as well as key privacy, to argue that the resulting labels and permutation bits look fresh.

Before we introduce the proof hybrids, we define some helpful notation. We use the notation of Lemma 10 for the randomness $r_1, \ldots, r_t$ sampled in case $b = 0$. For every wire $w \in \mathcal{W}$, wire value $\beta \in \{0,1\}$, and $i \in [t]$, we denote by

$$\mathsf{sk}^{\beta,(i)}_w \coloneqq \sigma^{\beta\oplus\bar{b}^{(i)}_w}_{w,i-1}(\ldots\sigma^{\beta\oplus\bar{b}^{(2)}_w}_{w,1}(\mathsf{sk}^\beta_w)\ldots)$$

the *level-i label* of wire w for wire value β, where $\mathsf{sk}_w^{\beta,(1)} = \mathsf{sk}_w^{\beta}$. Note that $\mathsf{sk}_w^{\beta,(i)} = \sigma_{w,i-1}^{\beta\oplus\bar{b}_w^{(i)}}(\mathsf{sk}_w^{\beta,(i-1)})$ for $i \geq 2$, and $\mathsf{sk}_w^{\beta,(i)} = \mathsf{sk}_\beta$ for all output wires $w \in \mathcal{W}_\mathcal{O}$, since all transformations for output wires are the identity. By Lemma 10, the keys contained in L_i are exactly the level-i labels of the input wires.

Next, we define a new garbling algorithm which we will re-use in the hybrids. Let $K = \{\hat{\mathsf{sk}}_w^0, \hat{\mathsf{sk}}_w^1\}_{w\in\mathcal{W}}$ be a set of wire labels with $\hat{\mathsf{sk}}_w^\beta = \mathsf{sk}_\beta$ for all $w \in \mathcal{W}_\mathcal{O}$, $\beta \in \{0,1\}$, and let $B = \{B_w\}_{w\in\mathcal{W}}$ be a set of permutation bits. We denote by $\widehat{\mathsf{Gb}}_t(\mathsf{pp}, f, x; K, B)$ the following algorithm.

$\widehat{\mathsf{Gb}}_t(\mathsf{pp}, f, x; K, B)$**:** For every gate $g_j = (w_\ell, w_r, w_o, \mathsf{op})$ and every position $(a_1, a_2, a_3) \in \{0,1\}^3$, let $\beta_1 \coloneqq a_1 \oplus B_{w_\ell}$, $\beta_2 \coloneqq a_2 \oplus B_{w_r}$, and $\beta_3 \coloneqq a_3 \oplus B_{w_o}$, and compute

$$\hat{c}^{(j)}_{a_1,a_2,a_3} \leftarrow \mathsf{Eval}(\mathsf{id}, \mathsf{id}, \mathsf{id}, \ldots \\ \mathsf{Eval}(\mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{Enc}(\hat{\mathsf{sk}}_{w_\ell}^{\beta_1}, \hat{\mathsf{sk}}_{w_r}^{\beta_2}, \hat{\mathsf{sk}}_{w_o}^{\mathsf{k}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3)}, \mathsf{m}_{\mathsf{op}}(\beta_1,\beta_2,\beta_3))) \ldots),$$

where $\hat{\mathsf{sk}}_{w_o}^{\perp} \coloneqq \perp$, Eval is called $t-1$ times, and all calls to Enc and Eval use fresh randomness. Output the garbling $\hat{\mathcal{C}} \coloneqq (\phi_\lambda(f), \hat{G}_1, \ldots, \hat{G}_N)$, where $\hat{G}_j$ contains $\hat{c}^{(j)}_{a_1,a_2,a_3}$ at position (a_1, a_2, a_3), together with the input encoding $\hat{I} \coloneqq \{(\hat{\mathsf{sk}}_{w_i}^{x_i}, x_i \oplus B_{w_i})\}_{w_i\in\mathcal{W}_I}$.

Note that $\widehat{\mathsf{Gb}}_t$ garbles f exactly as Gb, but with wire labels K and permutation bits B, and additionally applies $t-1$ rerandomizations with identity transformations and without permutations to every ciphertext, before finally encoding input x.

We proceed with defining the proof hybrids.

H_0**:** Case $b = 0$ of the $\mathsf{RER\text{-}PRIV}$ game.

H_1**:** Sample $r_1, \ldots, r_t$ as in H_0 and compute from them the level-t labels and accumulated permutation bits

$$K^{(t)} \coloneqq \{\mathsf{sk}_w^{0,(t)}, \mathsf{sk}_w^{1,(t)}\}_{w\in\mathcal{W}} \quad \text{and} \quad B^{(t)} \coloneqq \{\bar{b}_w^{(t)}\}_{w\in\mathcal{W}}.$$

The view of $\mathcal{A}$ is $(\hat{\mathcal{C}}, \hat{I}, r_1, \ldots, r_{t-1})$, where $(\hat{\mathcal{C}}, \hat{I}) \leftarrow \widehat{\mathsf{Gb}}_t(\mathsf{pp}, f, x; K^{(t)}, B^{(t)})$. That is, all calls to Enc and Eval are computed with fresh randomness (sampled by $\widehat{\mathsf{Gb}}_t$) instead of the randomness contained in $r_1, \ldots, r_t$, the key and message transformations are applied directly to the keys and messages that are input to Enc, and ciphertexts in mode lossy encrypt $\perp$.

H_2**:** As H_1, but $(\hat{\mathcal{C}}, \hat{I}) \leftarrow \widehat{\mathsf{Gb}}_t(\mathsf{pp}, f, x; K^*, B^*)$, where $K^* = \{\hat{\mathsf{sk}}_w^0, \hat{\mathsf{sk}}_w^1\}_{w\in\mathcal{W}}$ consists of fresh keys $\hat{\mathsf{sk}}_w^\beta \leftarrow \mathsf{KGen}(\mathsf{pp}_\mathsf{L})$ for all $w \in \mathcal{W} \setminus \mathcal{W}_\mathcal{O}$ and $\hat{\mathsf{sk}}_w^\beta = \mathsf{sk}_\beta$ for all $w \in \mathcal{W}_\mathcal{O}$, and B^* consists of fresh uniformly random bits, both independent of $r_1, \ldots, r_t$.

H_3**:** Case $b = 1$ of the $\mathsf{RER\text{-}PRIV}$ game.

We proceed with showing statistical indistinguishability of all hybrids.

Lemma 14. *Assuming* $\mathsf{2LKMHE}$ *has statistical KMH-rerandomizability (Definition 7), hybrids H_0 and H_1 are statistically indistinguishable.*

Proof. In H_0, by Lemma 10, for each $j \in [N]$ and $(a_1, a_2, a_3) \in \{0,1\}^3$, denoting $g_j = (w_\ell, w_r, w_o, \mathsf{op})$, the ciphertext at position (a_1, a_2, a_3) of $G_j^{(t)}$, denoted by $c^{(t,j)}_{a_1,a_2,a_3}$, is given

by

$$\begin{gathered}
\mathsf{Eval}(\sigma^{a_1}_{w_\ell,t-1}, \sigma^{a_2}_{w_r,t-1}, \sigma^{a_3}_{w_o,t-1}, \\
\mathsf{Eval}(\sigma^{a_1\oplus\bar{b}^{(t)}_{w_\ell}\oplus\bar{b}^{(t-1)}_{w_\ell}}_{w_\ell,t-2}, \sigma^{a_2\oplus\bar{b}^{(t)}_{w_r}\oplus\bar{b}^{(t-1)}_{w_r}}_{w_r,t-2}, \sigma^{a_3\oplus\bar{b}^{(t)}_{w_o}\oplus\bar{b}^{(t-1)}_{w_o}}_{w_o,t-2}, \dots \\
\mathsf{Eval}(\sigma^{a_1\oplus\bar{b}^{(t)}_{w_\ell}\oplus\bar{b}^{(2)}_{w_\ell}}_{w_\ell,1}, \sigma^{a_2\oplus\bar{b}^{(t)}_{w_r}\oplus\bar{b}^{(2)}_{w_r}}_{w_r,1}, \sigma^{a_3\oplus\bar{b}^{(t)}_{w_o}\oplus\bar{b}^{(2)}_{w_o}}_{w_o,1}, \\
\mathsf{Enc}(\mathsf{sk}^{a_1\oplus\bar{b}^{(t)}_{w_\ell}}_{w_\ell}, \mathsf{sk}^{a_2\oplus\bar{b}^{(t)}_{w_r}}_{w_r}, \mathsf{sk}^{\mathsf{k_{op}}(a_1\oplus\bar{b}^{(t)}_{w_\ell}, a_2\oplus\bar{b}^{(t)}_{w_r}, a_3\oplus\bar{b}^{(t)}_{w_o})}_{w_o}, \\
\mathsf{m_{op}}(a_1\oplus\bar{b}^{(t)}_{w_\ell}, a_2\oplus\bar{b}^{(t)}_{w_r}, a_3\oplus\bar{b}^{(t)}_{w_o}))) \dots)).
\end{gathered}$$

Then, letting $\beta_1 \coloneqq a_1 \oplus \bar{b}^{(t)}_{w_\ell}$, $\beta_2 \coloneqq a_2 \oplus \bar{b}^{(t)}_{w_r}$, and $\beta_3 \coloneqq a_3 \oplus \bar{b}^{(t)}_{w_o}$, we can write the above as

$$\begin{gathered}
\mathsf{Eval}(\sigma^{\beta_1\oplus\bar{b}^{(t)}_{w_\ell}}_{w_\ell,t-1}, \sigma^{\beta_2\oplus\bar{b}^{(t)}_{w_r}}_{w_r,t-1}, \sigma^{\beta_3\oplus\bar{b}^{(t)}_{w_o}}_{w_o,t-1}, \\
\mathsf{Eval}(\sigma^{\beta_1\oplus\bar{b}^{(t-1)}_{w_\ell}}_{w_\ell,t-2}, \sigma^{\beta_2\oplus\bar{b}^{(t-1)}_{w_r}}_{w_r,t-2}, \sigma^{\beta_3\oplus\bar{b}^{(t-1)}_{w_o}}_{w_o,t-2}, \dots \\
\mathsf{Eval}(\sigma^{\beta_1\oplus\bar{b}^{(2)}_{w_\ell}}_{w_\ell,1}, \sigma^{\beta_2\oplus\bar{b}^{(2)}_{w_r}}_{w_r,1}, \sigma^{\beta_3\oplus\bar{b}^{(2)}_{w_o}}_{w_o,1}, \\
\mathsf{Enc}(\mathsf{sk}^{\beta_1}_{w_\ell}, \mathsf{sk}^{\beta_2}_{w_r}, \mathsf{sk}^{\mathsf{k_{op}}(\beta_1,\beta_2,\beta_3)}_{w_o}, \\
\mathsf{m_{op}}(\beta_1,\beta_2,\beta_3))) \dots)).
\end{gathered}$$

Then, application of KMH-rerandomizability implies statistical closeness to the claimed garbled gate $\hat{c}^{(j)}_{a_1,a_2,a_3}$ from H_1. Applying KMH-rerandomizability $8N$ times, once for each ciphertext, thus switches the garbled gates from H_0 to H_1. Additionally, by Lemma 10, the input encoding in H_0 is $I_t = \{(\mathsf{sk}^{x_i,(t)}_{w_i}, x_i \oplus \bar{b}^{(t)}_{w_i})\}_{w_i\in\mathcal{W}_I}$, which is exactly the input encoding $\hat{I}$ in H_1. □

Lemma 15. *Assuming* 2LKMHE *has statistical key privacy (Definition 5),* $H_1 \overset{s}{\approx} H_2$.

Proof. In both H_1 and H_2, the view of $\mathcal{A}$ is $(\widehat{\mathsf{Gb}}_t(\mathsf{pp}, f, x; K, B), V)$, where $V \coloneqq (r_1, \dots, r_{t-1})$, $\widehat{\mathsf{Gb}}_t$ uses fresh randomness, $(K, B) = (K^{(t)}, B^{(t)})$ in H_1, and $(K, B) = (K^*, B^*)$ in H_2. Since the K and B are the only differences between H_1 and H_2, and the view of $\mathcal{A}$ is computed from (V, K, B) in the same way in both hybrids (by running $\widehat{\mathsf{Gb}}_t$ with fresh randomness), it therefore suffices to show $(V, K^{(t)}, B^{(t)}) \overset{s}{\approx} (V, K^*, B^*)$. Concretely, V determines $\bar{b}^{(t-1)}_w$ and $\mathsf{sk}^{\beta,(t-1)}_w$ for all $w \in \mathcal{W}$ and $\beta \in \{0,1\}$, while the secret randomness r_t is independent of V.

For every $w \in \mathcal{W}$, it holds that $\bar{b}^{(t)}_w = \bar{b}^{(t-1)}_w \oplus b^{(t)}_w$, where $b^{(t)}_w$ is a uniformly random bit contained in r_t. Thus, $B^{(t)}$ is uniformly random and independent of V, i.e., it is identically distributed as the uniform B^*.

In regards to the transformation functions, the main task is to prove that each function is applied to exactly one key, as this is the condition for application of key privacy. For $w \in \mathcal{W} \setminus \mathcal{W}_\mathcal{O}$ and $\beta \in \{0,1\}$, let $\hat{\sigma}^\beta_w \coloneqq \sigma^{\beta\oplus\bar{b}^{(t)}_w}_{w,t-1}$, so that $\mathsf{sk}^{\beta,(t)}_w = \hat{\sigma}^\beta_w(\mathsf{sk}^{\beta,(t-1)}_w)$. The transformations $\sigma^0_{w,t-1}$ and $\sigma^1_{w,t-1}$ are independent and uniformly random elements of $\mathcal{F}$ contained in r_t. Since $(\hat{\sigma}^0_w, \hat{\sigma}^1_w)$ is either $(\sigma^0_{w,t-1}, \sigma^1_{w,t-1})$ or $(\sigma^1_{w,t-1}, \sigma^0_{w,t-1})$, depending on $\bar{b}^{(t)}_w$, also $\hat{\sigma}^0_w$ and $\hat{\sigma}^1_w$ are independent and uniformly random transformations. Each transformation $\hat{\sigma}^\beta_w$ is applied to the single key $\mathsf{sk}^{\beta,(t-1)}_w$ only, and no further information about $\hat{\sigma}^\beta_w$ is revealed (in particular, not via ciphertexts in mode lossy, which encrypt $\perp$ in H_1).

Therefore, we can apply key privacy of 2LKMHE to each of the $2|\mathcal{W} \setminus \mathcal{W}_\mathcal{O}|$ keys $\mathsf{sk}^{\beta,(t)}_w = \hat{\sigma}^\beta_w(\mathsf{sk}^{\beta,(t-1)}_w)$, one after the other, to replace it with a fresh key $\hat{\mathsf{sk}}^\beta_w \leftarrow \mathsf{KGen}(\mathsf{pp_L})$,

even given $\mathsf{sk}_w^{\beta,(t-1)}$. For the output wires $w \in \mathcal{W}_{\mathcal{O}}$, it holds that $\mathsf{sk}_w^{\beta,(t)} = \mathsf{sk}_\beta = \hat{\mathsf{sk}}_w^\beta$ in both H_1 and H_2. Overall, this shows $(V, K^{(t)}, B^{(t)}) \overset{s}{\approx} (V, K^*, B^*)$ and thus $H_1 \overset{s}{\approx} H_2$. □

From H_2 to H_3. The $\hat{\mathcal{C}}$ and $\hat{I}$ are now created using completely independent randomness from V. To go from H_2 to H_3, we can thus simply apply $H_0 \overset{s}{\approx} H_2$ to undo the changes we made in the hybrids, except that we use independent randomness $r'_1, \ldots, r'_{t-1}$ instead of the V known to $\mathcal{A}$. Clearly, giving the adversary $\mathcal{A}$ less information, i.e., the now unused V instead of the real $r'_1, \ldots, r'_{t-1}$, cannot increase its winning chances. Therefore, $H_2 \overset{s}{\approx} H_3$.

Overall, we have shown $H_0 \overset{s}{\approx} H_1 \overset{s}{\approx} H_2 \overset{s}{\approx} H_3$, i.e., cases $b = 0$ and $b = 1$ of the RER-PRIV game are statistically indistinguishable, which concludes the proof.

□

5 Constructing a 2LKMHE from LWE

In this section we show how to construct a 2LKMHE scheme from lattice-based hardness assumptions.

5.1 The Reduced Gadget

Our scheme is based on the GSW encryption scheme [GSW13]. GSW originally is designed to be a fully-homomorphic encryption scheme, which relies on the full gadget matrix from [MP12]. We instead make use of a reduced gadget, which improves parameters and is essential for both the key homomorphism and rerandomizability of our 2LKMHE construction.

For the following we assume that $q = q(\lambda), n = n(\lambda), m = m(\lambda)$ are positive integers with $q = q(\lambda)$ prime, and define $k = \lceil \log q \rceil$. The new, reduced gadget is defined as

$$\widehat{\mathbf{G}} = \begin{pmatrix} \mathbf{0}^{n \times m} \\ \mathbf{0}^{m-k} \mid \mathbf{g} \end{pmatrix} \in \mathbb{Z}_q^{(n+1) \times m},$$

where $\mathbf{g} = (2^0, 2, \ldots, 2^{k-1})$ is the usual gadget vector.

Of special interest will be the lattice $\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}})$. The following lemma states the relevant properties of this lattice.

Lemma 16. *Let q, n, m, k be positive integers with $k = \lceil \log q \rceil$, and $m \geq k$. Then the lattice $\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})$ has a basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ with determinant $\det \mathbf{S} = q$ and its Gram-Schmidt orthogonalization $\widetilde{\mathbf{S}}$ has norm $\left\|\widetilde{\mathbf{S}}\right\| \leq \sqrt{5}$.*

Proof. [MP12] specifies a basis $\mathbf{S}_k$ for the lattice $\mathbf{\Lambda}_q^\perp(\mathbf{g})$ with $\det(\mathbf{S}_k) = q$, which then also implies the basis

$$\mathbf{S} = \begin{pmatrix} \mathbf{I}_{m-k} & \\ & \mathbf{S}_k \end{pmatrix} \in \mathbb{Z}^{m \times m}$$

for the lattice $\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}})$, with the same determinant q. From Lemma 4.3 in [MP12] we know that the Gram-Schmidt orthogonalization $\widetilde{\mathbf{S}}_k$ of $\mathbf{S}_k$ has norm $\left\|\widetilde{\mathbf{S}}_k\right\| \leq \sqrt{5}$. □

5.2 Key-Homomorphic GSW Encryption with Reduced Gadget

GSW encryption using the reduced gadget works as follows: To encrypt a message $\mu \in \mathbb{Z}_q$ using a secret key $\mathbf{s} \in \mathbb{Z}_q^n$, we sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, and noise $\mathbf{e} \leftarrow \chi_{\mathsf{LWE}}$ from some suitable noise distribution χ_{LWE}, e.g. a discrete Gaussian, and set the ciphertext to

$$\mathbf{C} = \begin{pmatrix} \mathbf{A} \\ \mathbf{sA} + \mathbf{e} \end{pmatrix} + \mu \widehat{\mathbf{G}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{sA} + \mathbf{e} + (\mathbf{0}^{m-k} \mid \mu \mathbf{g}) \end{pmatrix}.$$

To decrypt with $\mathbf{s}$, we compute

$$(-\mathbf{s} \mid 1)\,\mathbf{C} = \mathbf{e} + \left(\mathbf{0}^{m-k} \mid \mu\mathbf{g}\right).$$

Decryption now reduces to the problem of recovering μ from the above noisy encoding. This is possible using an algorithm for the *LWE gadget decoding problem* [MP12, GMP19].

Lemma 17 (Theorem 5.1 in [GMP19])**.** *For any modulus q and gadget vector $\mathbf{g} = (1, 2, \ldots, 2^{k-1})$, there is a time and space $O(k)$ algorithm that recovers $\mu \in \mathbb{Z}_q$ and $\mathbf{e} \in \mathbb{Z}_q^k$ given $\mu\mathbf{g} + \mathbf{e}$, assuming that $\|\mathbf{e}\|_\infty \leq q/6$.*

Since the plaintext space is $\mathbb{Z}_q$, we can therefore encrypt secret keys in $\mathbb{Z}_q^n$ by encrypting each of the n elements separately. The key-homomorphism is also simple: To add an offset $\boldsymbol{\sigma} \in \mathbb{Z}_q^n$ to the key $\mathbf{s}$, simply use the given $\mathbf{A}$ to add $\boldsymbol{\sigma}\mathbf{A}$ to the last row of the ciphertext. Note that here the reduced gadget is helpful as with the regular gadget $\mathbf{G}$, the value $\mathbf{A}$ would not be accessible in plain.

5.3 Message-Homomorphism and Lossyness

The lossy mode of our 2LKMHE is enabled by the multiplication-by-a-constant message homomorphism support of GSW encryption, which we explain next.

If we take a scalar in $\mathbb{Z}_q$, and decompose it into bits, then computing the inner product of the bit vector with $\mathbf{g}$ will reconstruct the original scalar. Therefore multiplication by $\mathbf{g}$ can be seen as undoing the operation of bit decomposition. The inverse operation, i.e. bit decomposition itself, will be denoted by $\mathbf{g}^{-1} : \mathbb{Z}_q \to \{0,1\}^k$, which takes in a scalar represented as bits $a = \sum_{i=0}^{k-1} a_i 2^i \in \mathbb{Z}_q$ and maps it to the vector $(a_0, a_1, \ldots, a_{k-1})$, such that $\mathbf{g} \cdot (\mathbf{g}^{-1}(a))^\top = a$. We can extend this operation to our reduced gadget matrix $\widehat{\mathbf{G}}$ by defining $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1} : \mathbb{Z}_q^k \to \mathbb{Z}_q^{m \times m}$ that maps

$$(v_1, \ldots, v_k) \mapsto \begin{pmatrix} \mathbf{0}^{(m-k)\times m} \\ \mathbf{0}^{m-k} \mid \mathbf{g}^{-1}(v_1) \\ \vdots \\ \mathbf{0}^{m-k} \mid \mathbf{g}^{-1}(v_k) \end{pmatrix}^\top$$

for any $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{Z}_q^k$, which then satisfies

$$\widehat{\mathbf{G}}\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\mathbf{v}) = \begin{pmatrix} \mathbf{0}^{n\times m} \\ \mathbf{0}^{m-k} \mid \mathbf{v} \end{pmatrix}.$$

Given an encryption of some message $\mu \in \mathbb{Z}_q$, we can use the above to multiply the message by a scalar $\tau \in \mathbb{Z}_q$ by setting $\mathbf{X} \leftarrow \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g})$ and computing

$$\left(\begin{pmatrix} \mathbf{A} \\ \mathbf{sA} + \mathbf{e} \end{pmatrix} + \mu\widehat{\mathbf{G}}\right) \cdot \mathbf{X} = \begin{pmatrix} \mathbf{AX} \\ \mathbf{sAX} + \mathbf{eX} \end{pmatrix} + \mu\tau\widehat{\mathbf{G}}.$$

We can also use this to add lossyness to the scheme by adding an additional helper ciphertext. If the encryption mode is hom, then set $\zeta \leftarrow 1$, while if lossy, then set $\zeta \leftarrow 0$. Then encrypt ζ to obtain helper ciphertext

$$\mathbf{D} \leftarrow \begin{pmatrix} \mathbf{A}' \\ \mathbf{sA}' + \mathbf{e}' \end{pmatrix} + \zeta\widehat{\mathbf{G}}.$$

Then, to add $\tau \in \mathbb{Z}_q$ to the message $\mu \in \mathbb{Z}_q$, we set $\mathbf{X} \leftarrow \widehat{\mathbf{G}}_{\mathsf{det}}^{-1}(\tau\mathbf{g})$ and compute

$$\mathbf{C} + \mathbf{D} \cdot \mathbf{X} = \begin{pmatrix} \mathbf{A} + \mathbf{A}'\mathbf{X} \\ \mathbf{s}(\mathbf{A} + \mathbf{A}'\mathbf{X}) + \mathbf{e} + \mathbf{e}'\mathbf{X} \end{pmatrix} + (\mu + \zeta\tau)\widehat{\mathbf{G}}.$$

If the encryption mode is hom and thus $\zeta = 1$, then the new message will be $\mu + \tau$, while if the encryption mode is lossy and thus $\zeta = 0$, then the new message will still be μ, as desired.

5.4 Rerandomization

To rerandomize a ciphertext, we need to rerandomize the uniform matrix $\mathbf{A}$ as well as the noise $\mathbf{e}$.

- Rerandomization of $\mathbf{A}$ will be done by randomizing the $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ algorithm, following [AP14, BdPMW16]. Specifically, the $\mathbf{X}$ sampled for computation of $\mathbf{C} + \mathbf{D} \cdot \mathbf{X}$ will be randomized such that it contains sufficient min-entropy to apply the LHL to $\mathbf{A}'\mathbf{X}$. Intuitively, this yields that $\mathbf{A}'\mathbf{X}$, and thus also $\mathbf{A} + \mathbf{A}'\mathbf{X}$, look random.
- Rerandomization of the noise will be done by adding a smudging error. This will also serve to hide the leakage of the LHL secret $\mathbf{X}$, as contained in $\mathbf{e}'\mathbf{X}$.

We will now discuss how the randomization of $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ works, as described in [AP14, BdPMW16], but with respect to our reduced gadget $\widehat{\mathbf{G}}$. Consider its parity-check lattice

$$\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) = \left\{ \mathbf{z} \in \mathbb{Z}^m \middle| \widehat{\mathbf{G}}\mathbf{z}^{\top} = \mathbf{0}^{n+1} \bmod q \right\}.$$

Define randomized algorithm $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1} : \mathbb{Z}_q^k \to \mathbb{Z}^{m \times m}$ that takes in a vector $\mathbf{v} = (v_1, \ldots, v_k) \in \mathbb{Z}_q^k$, samples $\mathbf{c}_1, \ldots, \mathbf{c}_{m-k} \leftarrow \mathcal{D}_{\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}),\alpha}$ and $\mathbf{c}_i \leftarrow \mathcal{D}_{\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}})+(\mathbf{0}^{m-k}|\mathbf{g}^{-1}(v_{i-m+k})),\alpha}$ for $i \in \{m-k+1, \ldots, m\}$, and outputs

$$\begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_m \end{pmatrix}^{\top} \in \mathbb{Z}^{m \times m}.$$

Then if we sample $\mathbf{X} \leftarrow \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\mathbf{v})$ for some $\mathbf{v} \in \mathbb{Z}_q^k$ it holds that

$$\widehat{\mathbf{G}} \cdot \mathbf{X} = \begin{pmatrix} \mathbf{0}^{n \times m} \\ \mathbf{0}^{m-k} \mid \mathbf{v} \end{pmatrix},$$

therefore $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}$ preserves the functionality of $\widehat{\mathbf{G}}_{\mathsf{det}}^{-1}$ while adding min-entropy for use with the LHL. The following lemma quantifies said min-entropy.

Lemma 18 (Adapted from [BdPMW16], Lemma 3.8)**.** *For any $\epsilon \in (0,1)$, $\alpha \geq \eta_{\epsilon}(\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}))$, and $\mathbf{c} \in \mathbb{R}^m$, where $q \geq \alpha\sqrt{m}$, it holds that*

$$H_{\infty}\left(\mathcal{D}_{\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}})+\mathbf{c},\alpha} \bmod q\right) \geq -\log\left(\frac{q}{(1-\epsilon)\cdot\alpha^m} + 2^{-m} \cdot \frac{1+\epsilon}{1-\epsilon}\right).$$

Proof. For any $\mathbf{v} \in \mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) + \mathbf{c} \bmod q$, we have

$$\begin{aligned}
\left(\mathcal{D}_{\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}})+\mathbf{c},\alpha} \bmod q\right)(\mathbf{v}) &= \mathcal{D}_{\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}})+\mathbf{c},\alpha}(\mathbf{v} + q\mathbb{Z}^m) \\
&= \frac{\rho_{\alpha}(\mathbf{v} + q\mathbb{Z}^m)}{\rho_{\alpha}(\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) + \mathbf{c})} \\
&\leq \frac{\rho_{\alpha}(q\mathbb{Z}^m)}{\rho_{\alpha}(\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) + \mathbf{c})} = \frac{1}{\rho_{\alpha}(\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) + \mathbf{c})} + \frac{\rho_{\alpha}(q\mathbb{Z}^m \setminus \{\mathbf{0}\})}{\rho_{\alpha}(\mathbf{\Lambda}_q^{\perp}(\widehat{\mathbf{G}}) + \mathbf{c})},
\end{aligned}$$

where the inequality follows by Lemma 2. For first summand, Lemmas 5 and 16 gives

$$\rho_\alpha(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}) + \mathbf{c}) = \rho_{\alpha,-\mathbf{c}}(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})) \geq \frac{(1-\epsilon)\cdot\alpha^m}{\det(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}))} = \frac{(1-\epsilon)\cdot\alpha^m}{q}.$$

Since $q\mathbb{Z}^m \setminus \{\mathbf{0}\} \subseteq \mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})$ and every vector in $q\mathbb{Z}^m \setminus \{\mathbf{0}\}$ has Euclidean norm at least $q \geq \alpha\sqrt{m}$, application of Lemma 7 (with center $\mathbf{0}$ such that it holds for any $\alpha > 0$ and $\epsilon = 0$) results in

$$\begin{aligned}\rho_\alpha(q\mathbb{Z}^m \setminus \{\mathbf{0}\}) &\leq \rho_\alpha\left(\left\{\mathbf{x} \in \mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}) \;\middle|\; \|\mathbf{x}\|_2 \geq \alpha\sqrt{m}\right\}\right)\\ &= \rho_\alpha(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}))\cdot \Pr_{\mathbf{x}\leftarrow\mathcal{D}_{\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}),\alpha}}\left[\|\mathbf{x}\|_2 \geq \alpha\sqrt{m}\right] \leq 2^{-m}\cdot\rho_\alpha(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})).\end{aligned}$$

Applying Lemma 6 with center $-\mathbf{c}$, we thus obtain for the second summand that

$$\frac{\rho_\alpha(q\mathbb{Z}^m \setminus \{\mathbf{0}\})}{\rho_\alpha(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}) + \mathbf{c})} \leq 2^{-m}\cdot\frac{\rho_\alpha(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}))}{\rho_{\alpha,-\mathbf{c}}(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}}))} \leq 2^{-m}\cdot\frac{1+\epsilon}{1-\epsilon}.$$

Combining the bounds for both summands, every value of $\mathcal{D}_{\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})+\mathbf{c},\alpha} \bmod q$ has probability at most $\frac{q}{(1-\epsilon)\cdot\alpha^m} + 2^{-m}\cdot\frac{1+\epsilon}{1-\epsilon}$, which gives the claimed min-entropy bound. □

Lemma 18 is based on a similar lemma in [BdPMW16], where the authors calculated the min-entropy for the lattice $\mathbf{\Lambda}_q^\perp(\mathbf{G})$. Note that they use the full gadget $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}$ as opposed to our reduced gadget $\widehat{\mathbf{G}}$ since they need to support homomorphic multiplication for their application. This leads to worse min-entropy since $\mathbf{\Lambda}_q^\perp(\mathbf{G})$ has determinant q^n while we get $\det(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})) = q$. Unlike them, we also consider the min-entropy reduced modulo q, which seems to be a gap in [BdPMW16]. As long as q is large enough for the samples to rarely need to be reduced modulo q, this difference has almost no effect on the parameters, however.

The following lemma combines Lemma 18 and the LHL (Lemma 1) to show how to rerandomize a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.

Lemma 19 (Adapted from [BdPMW16], Lemma 3.1)**.** *Let $\epsilon, \epsilon' \in (0,1)$ with $\epsilon' = \mathsf{negl}(\lambda)$, and let $q, n, m, k \in \mathbb{N}$ with $m \geq n\log q + 1 + 2\log(1/\epsilon') + \log\left(\frac{1+\epsilon}{1-\epsilon}\right)$, $m = \mathsf{poly}(\lambda)$, and α be a positive real with*

$$\alpha \geq \max\left\{q^{1/m}2(1+\epsilon)^{-1/m}, \sqrt{5}\cdot\sqrt{\ln(2m(1+1/\epsilon))/\pi}\right\},$$

and $q \geq \alpha\sqrt{m}$. Then for any $\mathbf{v} \in \mathbb{Z}_q^k$ it holds that

$$\left\{(\mathbf{A}, \mathbf{AX}) \mid \mathbf{A} \hookleftarrow\$\, \mathbb{Z}_q^{n\times m}, \mathbf{X} \leftarrow \widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}(\mathbf{v})\right\} \overset{s}{\approx} \left\{(\mathbf{A}, \mathbf{Y}) \mid \mathbf{A} \hookleftarrow\$\, \mathbb{Z}_q^{n\times m}, \mathbf{Y} \hookleftarrow\$\, \mathbb{Z}_q^{n\times m}\right\}.$$

Proof. Combining Lemma 16 and Lemma 4 we have that

$$\alpha \geq \sqrt{5}\cdot\sqrt{\ln(2m(1+1/\epsilon))/\pi} \geq \left\|\widetilde{\mathbf{S}}\right\|\cdot\sqrt{\ln(2m(1+1/\epsilon))/\pi} \geq \eta_\epsilon(\mathbf{\Lambda}_q^\perp(\widehat{\mathbf{G}})).$$

Each of the m columns of $\mathbf{X}$ is sampled from $\mathcal{D}_{\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}})+\mathbf{c},\alpha}$ for some center $\mathbf{c} \in \mathbb{Z}^m$. We

can therefore apply the min-entropy bound (Lemma 18) to obtain

$$\begin{aligned} H_\infty(\mathcal{D}_{\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}})+\mathbf{c},\alpha} \bmod q) &\geq -\log\left(\frac{q}{(1-\epsilon)\cdot\alpha^m} + 2^{-m}\cdot\frac{1+\epsilon}{1-\epsilon}\right) \\ &\geq -\log\left(\frac{q(1+\epsilon)}{(1-\epsilon)q2^m} + 2^{-m}\cdot\frac{1+\epsilon}{1-\epsilon}\right) \\ &= -\log\left(\frac{2}{2^m}\cdot\frac{1+\epsilon}{1-\epsilon}\right) = m - 1 - \log\left(\frac{1+\epsilon}{1-\epsilon}\right) \\ &\geq n\log q + 2\log(1/\epsilon'), \end{aligned}$$

where we used that $\alpha \geq q^{1/m}2(1+\epsilon)^{-1/m}$. Finally we can apply the LHL (Lemma 1) m times (once for each column of $\mathbf{X}$) to obtain the desired result. □

Dealing with intermediate rerandomizations. Recall from the definition of rerandomizability of 2LKMHE (Definition 7) that rerandomizability should work even if there are honest-but-curious rerandomizations done *before* the last honest rerandomization. In the context of our scheme, this means that the adversary knows $\mathbf{A}$, and some matrices $\mathbf{X}_1, \ldots, \mathbf{X}_{t-2}$ which were sampled using $\widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}$. The last honest rerandomization is then done using another $\mathbf{X}_{t-1}$ sampled using $\widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}$, where the adversary does *not* obtain $\mathbf{X}_{t-1}$. Overall, $\mathbf{A}$ is then rerandomized $t-1$ times to obtain $\mathbf{A}\mathbf{X}_1 \ldots \mathbf{X}_{t-1}$. In this case, we cannot simply apply Lemma 19 to $\mathbf{A}\mathbf{X}_1$, since $\mathbf{X}_1$ is known to the adversary. Instead, in the sequel we prove that $\mathbf{A}\mathbf{X}_1 \ldots \mathbf{X}_{t-2}$ is statistically close to uniform, which, together with the secret $\mathbf{X}_{t-1}$, will enable us to recover applicability of Lemma 19. This is achieved by the following lemma.

Lemma 20. *Let q be prime, $n \in \mathbb{N}$, $k = \lceil \log q \rceil$, and $m \in \mathbb{N}$ such that $m - k \geq 2n\log q$ and $m = \mathsf{poly}(\lambda)$. Then for all $\mathbf{v} \in \mathbb{Z}_q^k$, and for any $\alpha \geq \omega(\sqrt{\log m})$, it holds that the distributions*

$$D_1 \coloneqq (\mathbf{A}, \mathbf{X}, \mathbf{A}\mathbf{X}), \text{ where } \mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n\times m}, \mathbf{X} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{v}), \text{ and}$$

$$D_2 \coloneqq (\mathbf{A}, \mathbf{X}, \mathbf{Y}), \text{ where } \mathbf{A}, \mathbf{Y} \leftarrow\!\$\, \mathbb{Z}_q^{n\times m} \text{ and } \mathbf{X} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{v}) \text{ conditioned on } \mathbf{A}\mathbf{X} = \mathbf{Y},$$

are statistically indistinguishable.

Proof. The proof largely follows the one of Corollary 5.4 in [GPV08], with some adaptations made to accommodate the fact that we sample the columns of $\mathbf{X}$ from a specific lattice and not the integer lattice $\mathbb{Z}^m$ as in [GPV08].

We first note that, given only $(\mathbf{A}, \mathbf{A}\mathbf{X})$, the distribution of $\mathbf{X}$ in D_1 is the same as in D_2. Therefore, by definition of conditional probability, it suffices to show that the tuples $(\mathbf{A}, \mathbf{A}\mathbf{X})$ (as in D_1) and $(\mathbf{A}, \mathbf{Y})$ (as in D_2) are statistically indistinguishable.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{Z}^m$ be the columns of $\mathbf{X}$, and $\mathbf{v} = (v_1, \ldots, v_k)$. By definition of $\widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}$, it holds that

$$\begin{aligned} &\mathbf{x}_1, \ldots, \mathbf{x}_{m-k} \leftarrow \mathcal{D}_{\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}}),\alpha} \\ &\mathbf{x}_i \leftarrow \mathcal{D}_{\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}})+(\mathbf{0}^{m-k}|\mathbf{g}^{-1}(v_{i-m+k})),\alpha} \quad \text{for } i \in \{m-k+1, \ldots, m\}. \end{aligned}$$

Since $\mathbf{\Lambda}^\perp(\widehat{\mathbf{G}}) = \mathbf{S}\mathbb{Z}^m = \mathbb{Z}^{m-k} \times \mathbf{S}_k\mathbb{Z}^k$, we can write each $\mathbf{x}_i$, for $i \in [m]$, as

$$\mathbf{x}_i = \begin{pmatrix} \overline{\mathbf{x}_i} \\ \underline{\mathbf{x}_i} \end{pmatrix},$$

where $\overline{\mathbf{x}_i} \leftarrow \mathcal{D}_{\mathbb{Z}^{m-k},\alpha}$ and $\underline{\mathbf{x}_i} \leftarrow \mathcal{D}_{\mathbf{S}_k\mathbb{Z}^k+\mathbf{c}_i,\alpha}$ for some $\mathbf{c}_i \in \{0,1\}^k$. Then, parsing $\mathbf{A} = (\mathbf{A}_\ell \mid \mathbf{A}_r)$ with $\mathbf{A}_\ell \in \mathbb{Z}_q^{n\times(m-k)}$ and $\mathbf{A}_r \in \mathbb{Z}_q^{n\times k}$, we can write

$$\mathbf{AX} = (\mathbf{A}_\ell \mid \mathbf{A}_r) \cdot \mathbf{X} = \left(\mathbf{A}_\ell\overline{\mathbf{x}_1} + \mathbf{A}_r\underline{\mathbf{x}_1} \mid \ldots \mid \mathbf{A}_\ell\overline{\mathbf{x}_m} + \mathbf{A}_r\underline{\mathbf{x}_m}\right).$$

Since $m - k \geq 2n\log q$, we can apply Lemma 8 to show that with probability at least $1 - 2q^{-n}$ it holds that the columns of $\mathbf{A}_\ell$ generate $\mathbb{Z}_q^n$ and $\alpha \geq \eta_\epsilon(\mathbf{\Lambda}^\perp(\mathbf{A}_\ell))$ for some negligible function $\epsilon = \epsilon(m) = \mathsf{negl}(\lambda)$. Then, application of Lemma 9 shows that each $\mathbf{A}_\ell\overline{\mathbf{x}_i}$, for $i \in [m]$, has statistical distance of 2ϵ from uniform over $\mathbb{Z}_q^n$, which, since the former is independent of the $\mathbf{A}_r\underline{\mathbf{x}_i}$ and m is polynomial in λ, implies that $\mathbf{AX}$ is statistically indistinguishable from uniform, as desired. □

Note that the use of the reduced gadget is essential here, as the regular gadget's $\mathbf{G}$ parity-check lattice $\mathbf{\Lambda}^\perp(\mathbf{G})$ is missing the needed sub-lattice $\mathbb{Z}^{m-k}$ (see [MP12, Lemma 4.3] for details).

The above lemma cannot replace the LHL on its own, as the distinguisher can still tell that $\mathbf{Y}$ satisfies $\mathbf{AX} = \mathbf{Y}$, which makes it easy to distinguish from a fresh uniform $\mathbf{Y}$. However, it is sufficient for use *with* Lemma 19 (as a randomness extractor), as there the side information given via $\mathbf{X}$ is not an issue.

5.5 Construction of 2LKMHE from LWE

In the following section we give a formal description of our 2LKMHE construction as well as proofs of security and correctness. We start by defining the parameters used.To achieve a computational security level of λ, a statistical security level of λ_2, and maximum rerandomization level $t_{\mathsf{max}} = t_{\mathsf{max}}(\lambda) \geq 2$ we set:

- $\epsilon, \epsilon' \in (0,1)$ with $\epsilon' = \mathsf{negl}(\lambda)$.
- Each application of the smudging lemma (Lemma 3) in the proof of rerandomizability contributes at most $2^{-\lambda_2}$ to the statistical distance. Asymptotically, one can simply set $\lambda_2 = \lambda$, for concrete parameters we use $\lambda_2 = 80$, see Section 5.10.
- $n = \Omega(\lambda)$, $\chi_{\mathsf{LWE}} = \mathcal{D}_{\mathbb{Z},\nu}$.
- q a prime with
$$q > 6\left(\sqrt{m} + 2^{-\lambda_2}\right)\gamma_{t_{\mathsf{max}}-1}, \tag{2}$$
where $\gamma_{t_{\mathsf{max}}-1}$ is the smudging parameter defined below, and $k = \lceil\log(q)\rceil$. This explicit bound ensures correctness (see Theorem 6) and implies $q \geq \alpha\sqrt{m}$. Asymptotically, it means $q = \Theta\left(2^{(t_{\mathsf{max}}-1)\lambda_2}\nu\alpha^{t_{\mathsf{max}}-1}\sqrt{m^{3(t_{\mathsf{max}}-1)+1}}\right)$ with leading constant 6. q, n, and ν must also be set such that LWE remains hard.
- $m \geq \max\left\{4n\log q + k, 2n\log q + 1 + 2\log(1/\epsilon') + \log\left(\frac{1+\epsilon}{1-\epsilon}\right)\right\}$ and for α as used by $\widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}$ we set
$$\alpha \geq \max\left\{q^{1/m}2(1+\epsilon)^{-1/m}, \sqrt{5}\cdot\sqrt{\ln(2m(1+1/\epsilon))/\pi}\right\}$$
with $\alpha \geq \omega(\sqrt{\log m})$ to enable rerandomization via application of Lemmas 19 and 20.

- The smudging noise is defined as $\chi_{\mathsf{smudge}}(i) = \mathcal{D}_{\mathbb{Z},\gamma_i}$ and $\chi_{\mathsf{smudge2}}(i) = \mathcal{D}_{\mathbb{Z},\gamma'_i}$ where

$$\begin{aligned}
\gamma'_1 &= 2^{\lambda_2}\nu\alpha m^{3/2},\\
\gamma'_i &= 2^{\lambda_2}\gamma'_{i-1}(2^{-\lambda_2}+\sqrt{m})\cdot\alpha m = 2^{i\lambda_2}\nu(2^{-\lambda_2}+\sqrt{m})^{i-1}\alpha^i m^{i+1/2}\\
&= \Theta\left(2^{i\lambda_2}\nu\alpha^i\sqrt{m^{3i}}\right) \quad \forall i\in\{2,\dots,t_{\mathsf{max}}-1\},\\
\gamma_1 &= 2^{\lambda_2}\nu(\sqrt{m}+\alpha m^{3/2}),\\
\gamma_i &= (1+2^{\lambda_2}\sqrt{m})\gamma_{i-1}+\gamma'_i = (1+2^{\lambda_2}\sqrt{m})^{i-1}\gamma_1+\sum_{j=0}^{i-2}(1+2^{\lambda_2}\sqrt{m})^j\gamma'_{i-j}\\
&= \Theta\left(2^{i\lambda_2}\nu\alpha^i\sqrt{m^{3i}}\right) \quad \forall i\in\{2,\dots,t_{\mathsf{max}}-1\},
\end{aligned}$$

  and $\gamma_i, \gamma'_i \geq \omega(\sqrt{\log m})$ for all $i \in [t_{\mathsf{max}}-1]$, to enable rerandomization via Lemma 3.

- Key space $\mathcal{K} \coloneqq \mathbb{Z}_q^n$, message space $\mathcal{M} \coloneqq \mathbb{Z}_q^n$, ciphertext space $\mathcal{C} \coloneqq \left(\mathbb{Z}_q^{(2n+1)\times m}\right)^{n+1} \times \mathbb{N}$, set of transformation functions $\mathcal{F} = \mathcal{F}_{\text{key}} = \mathcal{F}_{\text{msg}} \coloneqq \mathbb{Z}_q^n$ (representing the offset to add to the input), and lossyness indicator $\perp = \mathbf{0}^n$. The message space has size q^n, clearly large enough for garbling correctness to hold.

Note that due to the size of q required by the rerandomization, especially the smudging noise, the modulus to noise ratio of our LWE parameters is superpolynomial.

The parameters m and α depend on q and vice versa. Since m grows logarithmically in q, all of the above inequalities can be satisfied simultaneously. We give concrete parameters that satisfy all of them in Section 5.10.

We proceed by describing our construction of 2LKMHE.

Construction 2. *In the sequel, let* $\widehat{\mathbf{G}} \in \mathbb{Z}_q^{(2n+1)\times m}$ *be the reduced gadget matrix, as described in Section 5.1, but with dimensions adapted to a* two-key *scheme, i.e.,*

$$\widehat{\mathbf{G}} = \begin{pmatrix} \mathbf{0}^{2n\times m} \\ \mathbf{0}^{m-k} \mid \mathbf{g} \end{pmatrix} \in \mathbb{Z}_q^{(2n+1)\times m}.$$

Our 2LKMHE construction consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$: *Generate parameters as above and output*

$$\mathsf{pp} = (q, n, m, \chi_{\mathsf{LWE}}, \chi_{\mathsf{smudge}}, \chi_{\mathsf{smudge2}}, \alpha, t_{\mathsf{max}}).$$

$\mathsf{KGen}(\mathsf{pp})$: *Sample and output* $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n$.

$\mathsf{Enc}(\mathsf{pp}, \mathbf{s}_0, \mathbf{s}_1, \boldsymbol{\mu}, \mathsf{mode})$: *Sample* $\mathbf{A}_i, \mathbf{B}_i \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, $\mathbf{e}_i \leftarrow \chi_{\mathsf{LWE}}^m$ *for all* $i \in [n]$. *If* $\mathsf{mode} = \mathsf{lossy}$, *then overwrite the given message by additionally setting* $\boldsymbol{\mu} = \mathbf{0}^n$. *Parse* $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$. *Compute message ciphertexts*

$$\mathbf{C}_i \leftarrow \begin{pmatrix} \mathbf{A}_i \\ \mathbf{B}_i \\ \mathbf{s}_0\mathbf{A}_i + \mathbf{s}_1\mathbf{B}_i + \mathbf{e}_i \end{pmatrix} + \mu_i\widehat{\mathbf{G}} \in \mathbb{Z}_q^{(2n+1)\times m}.$$

If $\mathsf{mode} = \mathsf{lossy}$, *then set* $\zeta = 0$, *otherwise set* $\zeta = 1$. *Then compute homomorphism helper ciphertext by sampling* $\mathbf{A}, \mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, $\mathbf{e} \leftarrow \chi_{\mathsf{LWE}}^m$, *and setting*

$$\mathbf{D} \leftarrow \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{s}_0\mathbf{A} + \mathbf{s}_1\mathbf{B} + \mathbf{e} \end{pmatrix} + \zeta\widehat{\mathbf{G}} \in \mathbb{Z}_q^{(2n+1)\times m}.$$

Output ciphertext $\mathsf{ct} = (\mathbf{C}_1, \dots, \mathbf{C}_n, \mathbf{D}, 1)$.

$\mathsf{Eval}(\mathsf{pp}, \boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1, \boldsymbol{\tau}, \mathsf{ct})$**:** *Parse* $\mathsf{ct} = (\mathbf{C}_1, \ldots, \mathbf{C}_n, \mathbf{D}, t)$*. For each* $i \in [n]$*, we can apply the message transformation and rerandomize the ciphertext and* $\mathbf{D}$ *by computing*

$$\mathbf{C}'_i \leftarrow \mathbf{C}_i + \mathbf{D} \cdot \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\tau_i \mathbf{g}) \quad \forall i \in [n] \qquad \textit{and} \qquad \mathbf{D}' \leftarrow \mathbf{D} \cdot \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g})$$

We can then apply the key transformation and apply noise-flooding by first parsing

$$\mathbf{C}'_i = \begin{pmatrix} \mathbf{A}'_i \\ \mathbf{B}'_i \\ \mathbf{c}'_i \end{pmatrix} \quad \forall i \in [n] \qquad \textit{and} \qquad \mathbf{D}' = \begin{pmatrix} \mathbf{A}' \\ \mathbf{B}' \\ \mathbf{c}' \end{pmatrix},$$

and then sampling $\widehat{\mathbf{e}}_i \leftarrow \chi^m_{\mathsf{smudge}}(t) \forall i \in [n]$ *and* $\widehat{\mathbf{e}} \leftarrow \chi^m_{\mathsf{smudge2}}(t)$ *and setting*

$$\mathbf{C}''_i \leftarrow \mathbf{C}'_i + \begin{pmatrix} \mathbf{0}^{n \times m} \\ \mathbf{0}^{n \times m} \\ \boldsymbol{\sigma}_0 \mathbf{A}'_i + \boldsymbol{\sigma}_1 \mathbf{B}'_i + \widehat{\mathbf{e}}_i \end{pmatrix} \forall i \in [n] \quad \textit{and} \quad \mathbf{D}'' \leftarrow \mathbf{D}' + \begin{pmatrix} \mathbf{0}^{n \times m} \\ \mathbf{0}^{n \times m} \\ \sigma_0 \mathbf{A}' + \sigma_1 \mathbf{B}' + \widehat{\mathbf{e}} \end{pmatrix}.$$

Finally output the new ciphertext $\mathsf{ct}' = (\mathbf{C}''_1, \ldots, \mathbf{C}''_n, \mathbf{D}'', t+1)$*.*

$\mathsf{Dec}(\mathsf{pp}, \mathbf{s}_0, \mathbf{s}_1, \mathsf{ct})$: *Parse* $\mathsf{ct} = (\mathbf{C}_1, \ldots, \mathbf{C}_n, \cdot, \cdot)$*, and then for all* $i \in [n]$*, compute*

$$\mathbf{c}_i \leftarrow (-\mathbf{s}_0, -\mathbf{s}_1, 1) \cdot \mathbf{C}_i \in \mathbb{Z}_q^m,$$

and apply the LWE gadget decoding algorithm (with respect to $\mathbf{g}$*) to* $\mathbf{c}_i$ *to recover message entry* $\mu'_i \in \mathbb{Z}_q$*. Output message* $\mu' \leftarrow (\mu'_1, \ldots, \mu'_n) \in \mathbb{Z}_q^n$*.*

5.6 CPA Security

CPA security is a straightforward application of LWE to show that as long as one of the secret keys remains secret, the message is hidden.

Theorem 4. *Assuming that* $\mathsf{LWE}_{q,n,(2+p_0+p_1+p_2+p_3+p_4+p_5)(n+1)m,\nu}$ *(Definition 1) holds, Construction 2 is IND-CPA-secure (Definition 6) with respect to any PPT adversary* $\mathcal{A}$ *that makes* $p_0 = p_0(\lambda)$*,* $p_1 = p_1(\lambda)$*,* $p_2 = p_2(\lambda)$*,* $p_3 = p_3(\lambda)$*,* $p_4 = p_4(\lambda)$*, and* $p_5 = p_5(\lambda)$ *queries to its encryption oracles* $\mathsf{Enc}(\mathsf{sk}'_0, \cdot, \cdot, \cdot)$*,* $\mathsf{Enc}(\mathsf{sk}'_1, \cdot, \cdot, \cdot)$*,* $\mathsf{Enc}(\cdot, \mathsf{sk}'_1, \cdot, \cdot)$*,* $\mathsf{Enc}(\cdot, \mathsf{sk}'_0, \cdot, \cdot)$*,* $\mathsf{Enc}(\mathsf{sk}'_0, \mathsf{sk}'_1, \cdot, \cdot)$*, and* $\mathsf{Enc}(\mathsf{sk}'_1, \mathsf{sk}'_0, \cdot, \cdot)$*, respectively.*

Proof. Let $\mathcal{A}$ be an adversary against IND-CPA security of Construction 2. We show that as long as one of the keys $\mathbf{s}_0$ or $\mathbf{s}_1$ corresponds to the secret sk'_0 or sk'_1 not given to $\mathcal{A}$, the last row of each ciphertext part $\mathbf{C}_1, \ldots, \mathbf{C}_n$ and $\mathbf{D}$ is pseudorandom by LWE, which hides the message μ_i for all $i \in [n]$ as well as the encryption mode ζ. Consider the following sequence of hybrids.

H_0**:** Case $b = 0$ of the IND-CPA game.

H_1**:** Like H_0 but replace the $\mathbf{C}_1, \ldots, \mathbf{C}_n$ and $\mathbf{D}$ values in each ciphertext that is encrypted using sk'_0, i.e., $\mathsf{ct}_1, \mathsf{ct}_2$ and answers from oracles $\mathsf{Enc}(\mathsf{sk}'_0, \cdot, \cdot, \cdot)$, $\mathsf{Enc}(\cdot, \mathsf{sk}'_0, \cdot, \cdot)$, $\mathsf{Enc}(\mathsf{sk}'_0, \mathsf{sk}'_1, \cdot, \cdot)$, and $\mathsf{Enc}(\mathsf{sk}'_1, \mathsf{sk}'_0, \cdot, \cdot)$ with uniformly random ones. Note that this changes only the last rows of each matrix as the remaining elements are already uniformly random.

H_2**:** Like $H + 1$ but now replace the $\mathbf{C}_1, \ldots, \mathbf{C}_n$ and $\mathbf{D}$ values in each ciphertext that is encrypted using sk'_1 and not sk'_0, i.e., ct_0 and answers from oracle $\mathsf{Enc}(\cdot, \mathsf{sk}'_1, \cdot, \cdot)$ and $\mathsf{Enc}(\mathsf{sk}'_1, \cdot, \cdot, \cdot)$, with uniformly random ones.

Note that after H_2, the messages and encryption modes are perfectly hidden. Therefore we can transition to case $b = 1$ by injecting the messages and encryption modes for case $b = 1$ and then undoing the changes done by hybrids H_2 and H_1. Therefore, showing indistinguishability of the above three hybrids is sufficient.

$H_0 \overset{c}{\approx} H_1$**:** For any ciphertext $\mathsf{ct} = (\mathbf{C}_1, \ldots, \mathbf{C}_n, \mathbf{D}, t)$ obtained by $\mathcal{A}$, the (non-random) ciphertext parts are

$$\mathbf{s}_0\mathbf{A}_i + \mathbf{s}_1\mathbf{B}_i + \mathbf{e}_i \quad \forall i \in [n] \qquad \text{and} \qquad \mathbf{s}_0\mathbf{A} + \mathbf{s}_1\mathbf{B} + \mathbf{e}, \tag{3}$$

where $\mathbf{s}_0, \mathbf{s}_1 \leftarrow\!\!\$\ \mathbb{Z}_q^n$, $\mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{B}_1, \ldots, \mathbf{B}_n, \mathbf{A}, \mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, and $\mathbf{e}_1, \ldots, \mathbf{e}_n, \mathbf{e} \leftarrow \chi_{\mathsf{LWE}}^m = \mathcal{D}_{\mathbb{Z},\nu}^m$. Therefore, if $\mathbf{s}_0 = \mathsf{sk}_0'$ or $\mathbf{s}_1 = \mathsf{sk}_0'$, then application of LWE shows that the elements in Equation (3) are computationally indistinguishable from random. For all the relevant ciphertexts that reuse sk_0' we therefore need $(2 + p_0 + p_3 + p_4 + p_5)(n + 1)m$ LWE samples, so indistinguishability follows by application of $\mathsf{LWE}_{q,n,(2+p_0+p_3+p_4+p_5)(n+1)m,\nu}$.

$H_1 \overset{c}{\approx} H_2$**:** This uses the same argument as for $H_0 \overset{c}{\approx} H_1$ albeit using the secret $\mathbf{s}_1 = \mathsf{sk}_1'$. It thus follows by $\mathsf{LWE}_{q,n,(1+p_1+p_2)(n+1)m,\nu}$.

We now use $(2 + p_0 + p_1 + p_2 + p_3 + p_4 + p_5) \geq \max\{1 + p_1 + p_2, 2 + p_0 + p_3 + p_4 + p_5\}$ to obtain the desired result. □

5.7 Rerandomizability

Theorem 5. *Construction 2 is statistically KMH-rerandomizable (Definition 7).*

Proof. To help with notation, we start by giving identifiers to all the intermediate values. Specifically, define

$$\begin{aligned}
&\mathbf{s}'_{\beta,1} \leftarrow \mathbf{s}_\beta \quad \forall \beta \in \{0,1\},\\
&\mathbf{s}'_{\beta,i} \leftarrow \boldsymbol{\sigma}_{\beta,i-1} + \boldsymbol{\sigma}_{\beta,i-2} + \cdots + \boldsymbol{\sigma}_{\beta,1} + \mathbf{s}_\beta \quad \forall \beta \in \{0,1\},\\
&\boldsymbol{\mu}'_1 \leftarrow \boldsymbol{\mu},\\
&\boldsymbol{\mu}'_i \leftarrow \begin{cases} \boldsymbol{\tau}_{i-1} + \boldsymbol{\tau}_{i-2} + \cdots + \boldsymbol{\tau}_1 + \boldsymbol{\mu} & \text{if } \mathsf{mode} = \mathsf{hom} \\ \mathbf{0}^n & \text{if } \mathsf{mode} = \mathsf{lossy} \end{cases},\\
&\mathsf{ct}'_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathbf{s}_0, \mathbf{s}_1, \boldsymbol{\mu}, \mathsf{mode}; r_1),\\
&\mathsf{ct}'_i \leftarrow \mathsf{Eval}(\mathsf{pp}, \boldsymbol{\sigma}_{0,i-1}, \boldsymbol{\sigma}_{1,i-1}, \boldsymbol{\tau}_{i-1}, \ldots \mathsf{Eval}(\mathsf{pp}, \boldsymbol{\sigma}_{0,1}, \boldsymbol{\sigma}_{1,1}, \boldsymbol{\tau}_1, \mathsf{ct}'_1; r_2) \ldots; r_i),
\end{aligned}$$

for $i \in \{2, \ldots, t\}$. Note that ct'_t is exactly ct' as in case $b = 0$ of Definition 7. We now prove that case $b = 0$ is statistically indistinguishable from case $b = 1$. Specifically, we need to prove that the above computation of ct'_t is statistically indistinguishable from instead computing

$$\begin{aligned}
&\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathbf{s}'_{0,t}, \mathbf{s}'_{1,t}, \boldsymbol{\mu}'_t, \mathsf{mode}; r'_1)\\
&\mathsf{ct}_i \leftarrow \mathsf{Eval}(\mathsf{pp}, \mathbf{0}^n, \mathbf{0}^n, \mathbf{0}^n, \ldots \mathsf{Eval}(\mathsf{pp}, \mathbf{0}^n, \mathbf{0}^n, \mathsf{ct}_1; r'_2) \ldots; r'_i)
\end{aligned}$$

for $i \in \{2, \ldots, t\}$ and $r'_1 \leftarrow\!\!\$\ R_{\mathsf{Enc}}$, $r'_2, \ldots, r'_t \leftarrow\!\!\$\ R_{\mathsf{Eval}}$, and setting $\mathsf{ct}'_t \leftarrow \mathsf{ct}_t$. Here, the randomness $r'_1, \ldots, r'_t$ is unknown to the adversary.

Regarding the key homomorphisms note that Eval by definition simply adds the key offsets $\boldsymbol{\sigma}_{0,j}$ and $\boldsymbol{\sigma}_{1,j}$ to each key, so it does not make a difference whether the transformed key is directly given to Enc or the transformations are applied sequentially via Eval. For the message we have that by definition of Eval and $\widehat{\mathbf{G}}_{\mathsf{rand}}^{-1}$, the message in ct'_t is given by

$\zeta\boldsymbol{\mu} + \zeta\boldsymbol{\tau}_1 + \zeta\boldsymbol{\tau}_2 + \cdots + \zeta\boldsymbol{\tau}_{t-1}$. If $\mathsf{mode} = \mathsf{lossy}$, then $\zeta = 0$, so the message is $\mathbf{0}^n = \mu'_t$, while if $\mathsf{mode} = \mathsf{hom}$ (so $\zeta = 1$), then the message is $\boldsymbol{\mu} + \zeta\boldsymbol{\tau}_1 + \zeta\boldsymbol{\tau}_2 + \cdots + \zeta\boldsymbol{\tau}_{t-1} = \mu'_t$, as desired.

We now proceed with proving that the last rerandomization using honest r_t rerandomizes ct'_{t-1} sufficiently to allow us to replace $r_1, \ldots, r_{t-1}$ with fresh randomness. By definition of Eval, if we parse ciphertext ct'_{i-1}, for $i \in \{2, \ldots, t\}$, as

$$\mathsf{ct}'_{i-1} = \left(\mathbf{C}'_{i-1,1}, \ldots, \mathbf{C}'_{i-1,n}, \mathbf{D}'_{i-1}, i-1\right),$$

where

$$\mathbf{C}'_{i-1,j} = \begin{pmatrix} \mathbf{A}'_{i-1,j} \\ \mathbf{B}'_{i-1,j} \\ \mathbf{c}'_{i-1,j} \end{pmatrix} + \mu'_{i-1,j}\widehat{\mathbf{G}} \in \mathbb{Z}_q^{(2n+1)\times m} \quad \text{, and}$$
$$\mathbf{D}'_{i-1} = \begin{pmatrix} \mathbf{A}''_{i-1} \\ \mathbf{B}''_{i-1} \\ \mathbf{c}''_{i-1} \end{pmatrix} + \zeta\widehat{\mathbf{G}} \in \mathbb{Z}_q^{(2n+1)\times m},$$

with

$$\mathbf{c}'_{i-1,j} = \mathbf{s}'_{0,i-1}\mathbf{A}'_{i-1,j} + \mathbf{s}'_{1,i-1}\mathbf{B}'_{i-1,j} + \mathbf{e}'_{i-1,j},$$
$$\mathbf{c}''_{i-1} = \mathbf{s}'_{0,i-1}\mathbf{A}''_{i-1} + \mathbf{s}'_{1,i-1}\mathbf{B}''_{i-1} + \mathbf{e}''_{i-1},$$

for all $j \in [n]$, then by definition of Eval and $\widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}$, it holds that the next level of rerandomization, i.e. ct'_i, is given by

$$\mathbf{D}'_i = \begin{pmatrix} \mathbf{A}''_{i-1}\mathbf{X}''_{i-1} \\ \mathbf{B}''_{i-1}\mathbf{X}''_{i-1} \\ \mathbf{s}'_{0,i}\mathbf{A}''_{i-1}\mathbf{X}''_{i-1} + \mathbf{s}'_{1,i}\mathbf{B}''_{i-1}\mathbf{X}''_{i-1} + \mathbf{e}''_{i-1}\mathbf{X}''_{i-1} + \widehat{\mathbf{e}}''_{i-1} \end{pmatrix} + \zeta\widehat{\mathbf{G}},$$

where $\mathbf{X}''_{i-1} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g})$, $\widehat{\mathbf{e}}''_{i-1} \leftarrow \chi^m_{\mathsf{smudge2}}(i-1)$, and

$$\mathbf{C}'_{i,j} = \begin{pmatrix} \mathbf{A}'_{i,j} \\ \mathbf{B}'_{i,j} \\ \mathbf{s}'_{0,i}\mathbf{A}'_{i,j} + \mathbf{s}'_{1,i}\mathbf{B}'_{i,j} + \mathbf{e}'_{i-1,j} + \mathbf{e}''_{i-1}\mathbf{X}'_{i-1,j} + \widehat{\mathbf{e}}'_{i-1,j} \end{pmatrix} + (\mu'_{i-1,j} + \zeta\tau_{i-1,j})\widehat{\mathbf{G}},$$

where

$$\mathbf{X}'_{i-1,j} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\tau_{i-1,j}\mathbf{g}), \qquad \widehat{\mathbf{e}}'_{i-1,j} \leftarrow \chi^m_{\mathsf{smudge}}(i-1),$$
$$\mathbf{A}'_{i,j} = \mathbf{A}'_{i-1,j} + \mathbf{A}''_{i-1}\mathbf{X}'_{i-1,j}, \qquad \mathbf{B}'_{i,j} = \mathbf{B}'_{i-1,j} + \mathbf{B}''_{i-1}\mathbf{X}'_{i-1,j}.$$

To complete the proof, we now need to show that the last ciphertext ct_t looks like a ciphertext that has been rerandomized using fresh and secret randomness $r'_1, \ldots, r'_t$. Specifically, this consists of the following steps:

1. We prove that the noise terms $\mathbf{e}''_{t-1}\mathbf{X}''_{t-1} + \widehat{\mathbf{e}}''_{t-1}$ and $\mathbf{e}'_{t-1,j} + \mathbf{e}''_{i-1}\mathbf{X}'_{t-1,j} + \widehat{\mathbf{e}}'_{t-1,j}$ are statistically indistinguishable from fresh noise $\chi^m_{\mathsf{smudge2}}(t-1) = \mathcal{D}_{\mathbb{Z}^m,\gamma'_{t-1}}$ and $\chi^m_{\mathsf{smudge}}(t-1) = \mathcal{D}_{\mathbb{Z}^m,\gamma_{t-1}}$, respectively, using the smudging lemma (Lemma 3). Then we can insert the noise implied by $r'_1, \ldots, r'_t$ using another application of the smudging lemma.

2. Secondly we prove that each of the matrices $\mathbf{A}''_{t-1}\mathbf{X}''_{t-1}$, $\mathbf{B}''_{t-1}\mathbf{X}''_{t-1}$, $\mathbf{A}'_{t-1,j} + \mathbf{A}''_{t-1}\mathbf{X}'_{t-1,j}$, and $\mathbf{B}'_{t-1,j} + \mathbf{B}''_{t-1}\mathbf{X}'_{t-1,j}$ look uniform using Lemmas 19 and 20. We can then use Lemmas 19 and 20 again in the opposite direction to insert the rerandomization matrices implied by $r'_1, \ldots, r'_t$.

For the first property it suffices to show that

$$\left\|\mathbf{e}'_{t-1,j} + \mathbf{e}''_{t-1}\mathbf{X}'_{t-1,j}\right\|_2 \leq \gamma_{t-1} \cdot 2^{-\lambda_2}, \tag{4}$$

$$\left\|\mathbf{e}''_{t-1}\mathbf{X}''_{t-1}\right\|_2 \leq \gamma'_{t-1} \cdot 2^{-\lambda_2}, \tag{5}$$

such that we can apply the smudging lemma (Lemma 3) to show that

$$\mathsf{SD}\left(\mathcal{D}_{\mathbb{Z}^m,\gamma_{t-1}}, \mathcal{D}_{\mathbb{Z}^m,\gamma_{t-1},\mathbf{e}'_{t-1,j}+\mathbf{e}''_{t-1}\mathbf{X}'_{t-1,j}}\right) \leq \left\|\mathbf{e}'_{t-1,j} + \mathbf{e}''_{t-1}\mathbf{X}'_{t-1,j}\right\|_2/\gamma_{t-1} \leq 2^{-\lambda_2},$$

$$\mathsf{SD}\left(\mathcal{D}_{\mathbb{Z}^m,\gamma'_{t-1}}, \mathcal{D}_{\mathbb{Z}^m,\gamma'_{t-1},\mathbf{e}''_{t-1}\mathbf{X}''_{t-1}}\right) \leq \left\|\mathbf{e}''_{t-1}\mathbf{X}''_{t-1}\right\|_2/\gamma'_{t-1} \leq 2^{-\lambda_2}.$$

By definition of Enc and Eval for all $i \in \{2, \ldots, t\}$, $j \in [n]$ we have

$$\begin{aligned}\mathbf{e}'_{i,j} &= \mathbf{e}'_{i-1,j} + \mathbf{e}''_{i-1}\mathbf{X}'_{i-1,j} + \widehat{\mathbf{e}}'_{i-1,j},\\ \mathbf{e}''_{i} &= \mathbf{e}''_{i-1}\mathbf{X}''_{i-1} + \widehat{\mathbf{e}}''_{i-1},\end{aligned}$$

where $\mathbf{e}'_{1,j}, \mathbf{e}''_1 \leftarrow \chi^m_{\mathsf{LWE}}$, $\widehat{\mathbf{e}}'_{i-1,j} \leftarrow \chi^m_{\mathsf{smudge}}(i-1)$, and $\widehat{\mathbf{e}}''_{i-1} \leftarrow \chi^m_{\mathsf{smudge2}}(i-1)$.

For the next steps we will use the following lemma.

Lemma 21. *For any $\mathbf{v} \in \mathbb{Z}_q^k$ and any $\mathbf{e} \in \mathbb{Z}^m$, it holds that*

$$\|\mathbf{e}\mathbf{X}\|_2 \leq \|\mathbf{e}\|_2 \cdot \alpha m$$

with probability at least $1 - m \cdot 2^{-m} \cdot \frac{1+\epsilon}{1-\epsilon}$ over $\mathbf{X} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{v})$. In particular, this holds for any noise vector $\mathbf{e}$ that is independent of $\mathbf{X}$.

Proof. By $(\mathbf{x}_1, \ldots, \mathbf{x}_m) = \mathbf{X}$ denote the columns of $\mathbf{X}$. Definition of the $\|\cdot\|_2$-norm, and Cauchy-Schwarz, then ensure that

$$\begin{aligned}\|\mathbf{e}\mathbf{X}\|_2 &\leq \sqrt{m} \cdot \max_{i' \in [m]} |\langle \mathbf{e}, \mathbf{x}_{i'} \rangle|\\ &\leq \sqrt{m} \cdot \|\mathbf{e}\|_2 \cdot \max_{i' \in [m]} \|\mathbf{x}_{i'}\|_2.\end{aligned}$$

Each column of $\mathbf{X}$ is sampled from a zero-centered Gaussian with parameter α over some coset of the lattice $\boldsymbol{\Lambda}^{\perp}(\widehat{\mathbf{G}})$. We can therefore apply Lemma 7 to each column and a union bound over the m columns to obtain the desired

$$\Pr\left[\max_{i' \in [m]} \|\mathbf{x}_{i'}\|_2 > \alpha\sqrt{m}\right] \leq m \cdot 2^{-m} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

□

We will now prove Equation (5) by induction. For the base case $t = 2$ we have that by Lemma 21 and Lemma 7

$$\left\|\mathbf{e}''_1\mathbf{X}''_1\right\|_2 \leq \|\mathbf{e}''_1\|_2 \cdot \alpha m \leq \nu\alpha m^{3/2} \leq \gamma'_1 \cdot 2^{-\lambda_2},$$

where we used $\mathbf{e}''_1 \leftarrow \chi^m_{\mathsf{LWE}} = \mathcal{D}_{\mathbb{Z}^m,\nu}$. Now assume that Equation (5) holds for $t = i \geq 2$ and then show that it also holds for $t = i + 1$. Then with probability $1 - \mathsf{negl}(\lambda)$ we have

$$\begin{aligned}\left\|\mathbf{e}''_i\mathbf{X}''_i\right\|_2 &\leq \|\mathbf{e}''_i\|_2 \cdot \alpha m\\ &\leq \left(\left\|\mathbf{e}''_{i-1}\mathbf{X}''_{i-1}\right\|_2 + \left\|\widehat{\mathbf{e}}''_{i-1}\right\|_2\right) \cdot \alpha m\\ &\leq \left(\gamma'_{i-1} \cdot 2^{-\lambda_2} + \sqrt{m} \cdot \gamma'_{i-1}\right) \cdot \alpha m\\ &= \gamma'_{i-1}(2^{-\lambda_2} + \sqrt{m}) \cdot \alpha m\\ &= \gamma'_i \cdot 2^{-\lambda_2},\end{aligned}$$

where the first inequality follows as in the proof of Lemma 21, the second inequality follows by triangle-inequality and the third by the induction assumption and Lemma 7 applied to $\widehat{\mathbf{e}}''_{i-1} \leftarrow \mathcal{D}^m_{\mathbb{Z},\gamma'_{i-1}}$.

Showing that Equation (4) holds follows similarly. For the base case $t = 2$ we have

$$\begin{aligned}\left\|\mathbf{e}'_{1,j} + \mathbf{e}''_1\mathbf{X}'_{1,j}\right\|_2 &\leq \left\|\mathbf{e}'_{1,j}\right\|_2 + \left\|\mathbf{e}''_1\mathbf{X}'_{1,j}\right\|_2 \\ &\leq \nu\sqrt{m} + \nu\alpha m^{3/2} \leq \gamma_1 \cdot 2^{-\lambda_2},\end{aligned}$$

where we used the triangle inequality, Lemma 7 applied to $\mathbf{e}'_{1,j} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\nu}$, and Lemma 21. Again assume that Equation (4) holds for $t = i \geq 2$. Then

$$\begin{aligned}\left\|\mathbf{e}'_{i,j} + \mathbf{e}''_i\mathbf{X}'_{i,j}\right\|_2 &\leq \left\|\mathbf{e}'_{i,j}\right\|_2 + \left\|\mathbf{e}''_i\mathbf{X}'_{i,j}\right\|_2 \\ &\leq \left\|\mathbf{e}'_{i-1,j} + \mathbf{e}''_{i-1}\mathbf{X}'_{i-1,j}\right\|_2 + \left\|\widehat{\mathbf{e}}'_{i-1,j}\right\|_2 + \left\|\mathbf{e}''_i\mathbf{X}'_{i,j}\right\|_2 \\ &\leq \gamma_{i-1} \cdot 2^{-\lambda_2} + \sqrt{m} \cdot \gamma_{i-1} + \gamma'_i \cdot 2^{-\lambda_2} = \gamma_i \cdot 2^{-\lambda_2}.\end{aligned}$$

Now it just remains to show that each of the matrices $\mathbf{A}''_{t-1}\mathbf{X}''_{t-1}$, $\mathbf{B}''_{t-1}\mathbf{X}''_{t-1}$, $\mathbf{A}'_{t-1,j} + \mathbf{A}''_{t-1}\mathbf{X}'_{t-1,j}$, and $\mathbf{B}'_{t-1,j} + \mathbf{B}''_{t-1}\mathbf{X}'_{t-1,j}$ look uniform. By definition of Eval it holds that

$$\begin{pmatrix}\mathbf{A}''_{t-1}\mathbf{X}''_{t-1} \\ \mathbf{B}''_{t-1}\mathbf{X}''_{t-1}\end{pmatrix} = \begin{pmatrix}\mathbf{A}''_{t-1} \\ \mathbf{B}''_{t-1}\end{pmatrix}\mathbf{X}''_{t-1} = \begin{pmatrix}\mathbf{A}''_1 \\ \mathbf{B}''_1\end{pmatrix} \cdot \prod_{j=1}^{t-1}\mathbf{X}''_j,$$

where $\mathbf{A}''_1, \mathbf{B}''_1 \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$ and $\mathbf{X}''_i \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g})$ for all $i \in [t-1]$, and

$$\begin{aligned}\begin{pmatrix}\mathbf{A}'_{t-1,j} + \mathbf{A}''_{t-1}\mathbf{X}'_{t-1,j} \\ \mathbf{B}'_{t-1,j} + \mathbf{B}''_{i-1}\mathbf{X}'_{t-1,j}\end{pmatrix} &= \begin{pmatrix}\mathbf{A}'_{t-1,j} \\ \mathbf{B}'_{t-1,j}\end{pmatrix} + \begin{pmatrix}\mathbf{A}''_{t-1} \\ \mathbf{B}''_{t-1}\end{pmatrix} \cdot \mathbf{X}'_{t-1,j} \\ &= \begin{pmatrix}\mathbf{A}'_{t-2,j} \\ \mathbf{B}'_{t-2,j}\end{pmatrix} + \begin{pmatrix}\mathbf{A}''_{t-2} \\ \mathbf{B}''_{t-2}\end{pmatrix} \cdot \mathbf{X}'_{t-2,j} + \begin{pmatrix}\mathbf{A}''_{t-1} \\ \mathbf{B}''_{t-1}\end{pmatrix} \cdot \mathbf{X}'_{t-1,j} \\ &= \begin{pmatrix}\mathbf{A}'_{1,j} \\ \mathbf{B}'_{1,j}\end{pmatrix} + \sum_{\ell=1}^{t-1}\begin{pmatrix}\mathbf{A}''_{\ell} \\ \mathbf{B}''_{\ell}\end{pmatrix} \cdot \mathbf{X}'_{\ell,j} \\ &= \begin{pmatrix}\mathbf{A}'_{1,j} \\ \mathbf{B}'_{1,j}\end{pmatrix} + \sum_{\ell=1}^{t-1}\begin{pmatrix}\mathbf{A}''_{1} \\ \mathbf{B}''_{1}\end{pmatrix} \cdot \prod_{k=1}^{\ell-1}\mathbf{X}''_k \cdot \mathbf{X}'_{\ell,j},\end{aligned}$$

where $\mathbf{A}'_{1,j}, \mathbf{B}'_{1,j} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$ and $\mathbf{X}'_{\ell,j} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\tau_{\ell,j}\mathbf{g})$ for all $\ell \in [t-1], j \in [n]$. Note that the adversary is given randomness $r_1, \ldots, r_{t-1}$ in the rerandomizability game (Definition 7), therefore it knows all values except for the last $\mathbf{X}''_{t-1}$ and $\mathbf{X}'_{t-1,j}$ (which are computed from the secret r_t). Therefore, to complete the proof we have to show that even when given

$$\left(\mathbf{A}''_1, \mathbf{B}''_1, \left\{\mathbf{A}'_{1,j}, \mathbf{B}'_{1,j}, \mathbf{X}'_{\ell,j}, \mathbf{X}''_\ell\right\}_{\ell\in[t-2], j\in[n]}\right)$$

the terms

$$\begin{pmatrix}\mathbf{A}''_1 \\ \mathbf{B}''_1\end{pmatrix} \cdot \prod_{j=1}^{t-1}\mathbf{X}''_j \quad \text{and} \quad \left\{\begin{pmatrix}\mathbf{A}'_{1,j} \\ \mathbf{B}'_{1,j}\end{pmatrix} + \sum_{\ell=1}^{t-1}\begin{pmatrix}\mathbf{A}''_1 \\ \mathbf{B}''_1\end{pmatrix} \cdot \prod_{k=1}^{\ell-1}\mathbf{X}''_k \cdot \mathbf{X}'_{\ell,j}\right\}_{j\in[n]} \tag{6}$$

are indistinguishable from independently uniform. To set up use of the LHL, i.e., Lemma 19, with the secrets $\mathbf{X}''_{t-1}$ and $\{\mathbf{X}'_{t-1,j}\}_{j\in[n]}$, we first show that the matrix applied to the former as the LHL randomness extractor, i.e., the

$$\begin{pmatrix}\mathbf{A}''_1 \\ \mathbf{B}''_1\end{pmatrix} \cdot \prod_{j=1}^{t-2}\mathbf{X}''_j \tag{7}$$

looks uniform (albeit with some side information since the related $\mathbf{X}''_j$ are known to the adversary). To do this, we use a sequence of $t-2$ hybrids. In the first hybrid we apply Lemma 20, using that $m - k \geq 4n \log q$ and $\alpha \geq \omega(\sqrt{\log m})$, to obtain

$$\left\{ \left(\mathbf{A}''_1, \mathbf{B}''_1, \mathbf{X}''_1, \begin{pmatrix} \mathbf{A}''_1 \\ \mathbf{B}''_1 \end{pmatrix} \mathbf{X}''_1 \right) \;\middle|\; \mathbf{X}''_1 \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g}) \right\}$$
$$\overset{\mathrm{s}}{\approx} \left\{ (\mathbf{A}''_1, \mathbf{B}''_1, \mathbf{X}''_1, \mathbf{Y}''_1) \;\middle|\; \mathbf{Y}''_1 \leftarrow\!\$\, \mathbb{Z}_q^{2n \times m}, \mathbf{X}''_1 \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g}) \text{ s.t. } (\mathbf{A}''_1 \mid \mathbf{B}''_1)^\top \mathbf{X}''_1 = \mathbf{Y}''_1 \right\},$$

where $\mathbf{A}''_1, \mathbf{B}''_1 \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}$ on both sides. Hybrid i for $1 < i \leq t-2$ is then inductively defined by similarly replacing $\mathbf{Y}''_{i-1}\mathbf{X}''_i$ with uniform $\mathbf{Y}''_i$ such that indistinguishability of successive hybrids again follows by Lemma 20. Across all the hybrids, we thus show that

$$\left(\mathbf{A}''_1, \mathbf{B}''_1, \left\{ \mathbf{X}''_\ell, \begin{pmatrix} \mathbf{A}''_1 \\ \mathbf{B}''_1 \end{pmatrix} \cdot \prod_{j=1}^{\ell} \mathbf{X}''_j \right\}_{\ell \in [t-2]} \;\middle|\; \mathbf{X}''_1, \ldots, \mathbf{X}''_{t-2} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g}) \right) \overset{\mathrm{s}}{\approx}$$

$$\left(\mathbf{A}''_1, \mathbf{B}''_1, \left\{ \mathbf{X}''_\ell, \mathbf{Y}''_\ell \right\}_{\ell \in [t-2]} \;\middle|\; \begin{array}{c} \mathbf{Y}''_1, \ldots, \mathbf{Y}''_{t-2} \leftarrow\!\$\, \mathbb{Z}_q^{2n \times m}, \mathbf{X}''_1, \ldots, \mathbf{X}''_{t-2} \leftarrow \widehat{\mathbf{G}}^{-1}_{\mathsf{rand}}(\mathbf{g}) \\ \text{s. t. } \begin{pmatrix} \mathbf{A}''_1 \\ \mathbf{B}''_1 \end{pmatrix} \mathbf{X}''_1 = \mathbf{Y}''_1 \text{ and } \mathbf{Y}''_{\ell-1}\mathbf{X}''_\ell = \mathbf{Y}''_\ell \; \forall \ell \in \{2, \ldots, t-2\} \end{array} \right),$$

where the second line corresponds to hybrid $t-2$. Note that the term from Equation (7) has been replaced with a uniform $\mathbf{Y}''_{t-2} \leftarrow\!\$\, \mathbb{Z}_q^{2n \times m}$, which can now be used as a randomness extractor for Lemma 19. Equation (6) can then be written as

$$\mathbf{Y}''_{t-2}\mathbf{X}''_{t-1} \quad \text{and} \quad \left\{ \begin{pmatrix} \mathbf{A}'_{1,j} \\ \mathbf{B}'_{1,j} \end{pmatrix} + \sum_{\ell=1}^{t-1} \mathbf{Y}''_{\ell-1} \cdot \mathbf{X}'_{\ell,j} \right\}_{j \in [n]}, \tag{8}$$

where $\mathbf{Y}''_0 = \left(\begin{smallmatrix} \mathbf{A}''_1 \\ \mathbf{B}''_1 \end{smallmatrix}\right)$. Next, using the fact that the $\mathbf{X}''_{t-1}$ and $\{\mathbf{X}'_{t-1,j}\}_{j \in [n]}$ are secret, we can apply Lemma 19 to each of the $n+1$ terms

$$\mathbf{Y}''_{t-2}\mathbf{X}''_{t-1} \quad \text{and} \quad \left\{ \mathbf{Y}''_{t-2}\mathbf{X}'_{t-1,j} \right\}_{j \in [n]}$$

in Equation (8) to replace them with independently uniform $\mathbf{Y}''_{t-1}$ and $\left\{\mathbf{Y}'_{t-1,j}\right\}_{j \in [n]}$, respectively. Conditions for application of Lemma 19 hold due to the parameter choices

$$\begin{aligned} \epsilon' &= \mathsf{negl}(\lambda) \\ m &\geq 2n \log q + 1 + 2\log(1/\epsilon') + \log\left(\frac{1+\epsilon}{1-\epsilon}\right) \\ \alpha &\geq \max\left\{ q^{1/m} 2(1+\epsilon)^{-1/m}, \sqrt{5} \cdot \sqrt{\ln(2m(1+1/\epsilon))/\pi} \right\} \\ q &\geq \alpha\sqrt{m}. \end{aligned}$$

Now Equation (8) can be written as

$$\mathbf{Y}''_{t-1} \quad \text{and} \quad \left\{ \begin{pmatrix} \mathbf{A}'_{1,j} \\ \mathbf{B}'_{1,j} \end{pmatrix} + \sum_{\ell=1}^{t-2} \mathbf{Y}''_{\ell-1} \cdot \mathbf{X}'_{\ell,j} + \mathbf{Y}'_{t-1,j} \right\}_{j \in [n]}. \tag{9}$$

Lastly, we note that the right side of Equation (9) is indeed independently uniformly distributed since each $\mathbf{Y}'_{t-1,j}$ is itself uniform and the adversary obtains no other side information about each $\mathbf{Y}'_{t-1,j}$, i.e., it obtains the terms in Equation (9) but not $\mathbf{Y}'_{t-1,j}$ itself. Therefore, we can equivalently sample the right side of Equation (9) as an independently uniform value. This concludes the proof of Theorem 5. □

Remark 1 (Concrete statistical distance). Each application of the smudging lemma contributes at most $2^{-\lambda_2}$ to the statistical distance in the above proof. The proof applies it twice to each of the $n+1$ components of a ciphertext (once to remove the old noise and once to insert fresh noise), so that rerandomizing a single ciphertext contributes at most $2(n+1) \cdot 2^{-\lambda_2}$, plus the error terms of the LHL-based steps, which are of order ϵ' each. Since each garbled gate of our RGS contains eight ciphertexts, a garbled circuit with $|C|$ gates accumulates a statistical distance of roughly $16(n+1)|C| \cdot 2^{-\lambda_2}$.

5.8 Correctness

Theorem 6. *Construction 2 is statistically correct (Definition 4).*

Proof. Fix a level $t \in [t_{\mathsf{max}}]$ and $j \in [n]$ and use the notation from the proof of Theorem 5. Let $\mathbf{e} \in \mathbb{Z}^m$ be the noise that decryption recovers from the j-th message ciphertext, i.e., $(-\mathbf{s}'_{0,t}, -\mathbf{s}'_{1,t}, 1) \cdot \mathbf{C}'_{t,j} = \mathbf{e} + (\mathbf{0}^{m-k} \mid \mu'_{t,j}\mathbf{g})$. By Lemma 17, the gadget decoding algorithm from [GMP19] recovers $\mu'_{t,j}$ if $\|\mathbf{e}\|_\infty \leq q/6$, and we use $\|\mathbf{e}\|_\infty \leq \|\mathbf{e}\|_2$.

For $t = 1$, we have $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\nu}$, so $\|\mathbf{e}\|_2 \leq \nu\sqrt{m}$ by Lemma 7, which is well below $q/6$ by Equation (2).

For $t \geq 2$, we have

$$\mathbf{e} = \mathbf{e}'_{t,j} = \left(\mathbf{e}'_{t-1,j} + \mathbf{e}''_{t-1}\mathbf{X}'_{t-1,j}\right) + \widehat{\mathbf{e}}'_{t-1,j}.$$

The inductive argument in the proof of Theorem 5 shows that the term in parentheses has norm at most $\gamma_{t-1} \cdot 2^{-\lambda_2}$ (see Equation (4)), and the smudging noise $\widehat{\mathbf{e}}'_{t-1,j} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\gamma_{t-1}}$ has norm at most $\gamma_{t-1}\sqrt{m}$ by Lemma 7. Since the γ_i are increasing in i, we obtain

$$\|\mathbf{e}\|_\infty \leq \|\mathbf{e}\|_2 \leq \left(\sqrt{m} + 2^{-\lambda_2}\right)\gamma_{t-1} \leq \left(\sqrt{m} + 2^{-\lambda_2}\right)\gamma_{t_{\mathsf{max}}-1} < q/6,$$

where the last inequality is exactly our choice of q in Equation (2).

All of these bounds hold except with the failure probability of Lemmas 7 and 21. Each rerandomization uses one Gaussian tail bound and one application of Lemma 21 for each of the $n+1$ ciphertext components. By a union bound, decryption therefore fails with probability at most $t_{\mathsf{max}}(n+1)(m+1) \cdot 2^{-m} \cdot \frac{1+\epsilon}{1-\epsilon}$, which is negligible. □

5.9 Key Privacy

Theorem 7. *Construction 2 has key-privacy (Definition 5).*

Proof. Follows trivially from the fact that we rerandomize keys by sampling a random element $\boldsymbol{\sigma} \in \mathbb{Z}_q^n$ and then adding it to the key $\mathbf{s} \in \mathbb{Z}_q^n$. Since $\mathbb{Z}_q$ is a group (so addition is bijective), this process leads to a fresh random key $\boldsymbol{\sigma} + \mathbf{s} \in \mathbb{Z}_q^n$. □

5.10 Concrete parameters for 2LKMHE

Due to the complexity of the scheme and its parameter bounds, selection of concrete parameters is not trivial. We now explain how such parameters can be selected by the example of maximum rerandomization level $t_{\mathsf{max}} = 3$. We hope that this will also be useful for researchers looking to improve upon our results.

We distinguish two security parameters. The computational security parameter $\lambda = 128$ determines the LHL error $\epsilon' = 2^{-\lambda}$, and n and ν are chosen such that the resulting LWE instance is hard. We estimated its security with the lattice estimator [APS15] for an error with standard deviation $\nu/\sqrt{2\pi} \approx 51$. The best attacks are the primal attacks (uSVP and BDD) and the dual hybrid attack, all with BKZ block size $\beta \approx 470$. In the Core-SVP cost model [ADPS16], they cost about 2^{137} operations classically and 2^{124} quantumly. The statistical security parameter $\lambda_2 = 80$ bounds the statistical distance of each application

Table 1: Concrete parameters for 2LKMHE with $t_{\mathsf{max}} = 3$.

Parameter	Value	Definition
λ	128	Computational security parameter
λ_2	80	Statistical security parameter (smudging)
t_{max}	3	Maximum rerandomization level
ϵ	0.5	Smoothing slack parameter
ϵ'	$2^{-\lambda}$	LHL error parameter
ν	128	LWE error width
n	12 000	LWE dimension
m	12 336 257	Ciphertext width, $m = 4nk + k$
α	5.370 156 194 6	Rerandomization Gaussian parameter
$q_{\min}$	$\approx 2^{256.9}$	Correctness bound $6(\sqrt{m} + 2^{-\lambda_2})\gamma_2$
q	257-bit prime	Smallest prime larger than $q_{\min}$
k	257	Bit length of q

of the smudging lemma and thereby determines the smudging parameters γ_i and γ'_i. By Remark 1, rerandomizing a single ciphertext introduces a statistical distance of about $2(n+1) \cdot 2^{-\lambda_2} \approx 2^{-65}$ and a garbled circuit with 2^{20} gates accumulates about 2^{-42}. Each additional bit of λ_2 increases the bit length of q by about $t_{\mathsf{max}} - 1$ bits.

The modulus q is determined by the correctness condition Equation (2). It is therefore much larger than in typical LWE instantiations, which forces a large dimension n, since for fixed n and ν the hardness of LWE decreases as q/ν grows. Starting from a guess for k, we set $m = 4nk+k$, compute α, γ_i, and γ'_i, and then the lower bound $q_{\min} = 6(\sqrt{m}+2^{-\lambda_2})\gamma_{t_{\mathsf{max}}-1}$ on q, update $k = \lceil \log q_{\min} \rceil$, and repeat until k no longer changes. Finally, q is the smallest prime larger than $q_{\min}$.

6 Incremental Decomposable Randomized Encodings

Let $(\mathcal{F}^\lambda_{\mathsf{iDRE}})_{\lambda\in\mathbb{N}}$ be a family of function sets, where $\mathcal{F}^\lambda_{\mathsf{iDRE}} = \{f : \{0,1\}^{\ell_{\mathsf{in}}(f)} \to \{0,1\}^{\ell_{\mathsf{out}}(f)}\}$ is a set of functions. Let $\phi_\lambda(f)$ of a function f denote its allowed *garbling leakage*, i.e. the topology of the circuit representation of f. We proceed to recall the definition of iDRE from [AHKP22], but adapted to our "levelled" notion.

Definition 12 (Levelled incremental Decomposable Randomized Encoding (iDRE))**.** A (levelled) *incremental decomposable randomized encoding* (iDRE) for function family $(\mathcal{F}^\lambda_{\mathsf{iDRE}})_{\lambda\in\mathbb{N}}$ is a tuple of PPT algorithms $\mathsf{iDRE} = (\mathsf{Setup}, \mathsf{PreEn}, \mathsf{En}, \mathsf{En}^*, \mathsf{Combine}, \mathsf{Dec})$ with the following syntax.

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$**:** Given the security parameter λ and a maximum level t_{max}, this algorithm returns public parameters pp. The public parameters define a function family $\mathcal{F}^\lambda_{\mathsf{iDRE}}$.

$\{e^0_{ij}, e^1_{ij}\}_{i\in[\ell_{\mathsf{in}}(f)]} \leftarrow \mathsf{PreEn}(\mathsf{pp}, j)$**:** Given an integer j, this algorithm returns a set of m pairs $\{e^0_{ij}, e^1_{ij}\}_{i\in[\ell_{\mathsf{in}}(f)]}$.

$(Y, \{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]}) \leftarrow \mathsf{En}(\mathsf{pp}, f, r)$**:** Given a function $f \in \mathcal{F}^\lambda_{\mathsf{iDRE}}$ and randomness r, this algorithm outputs an encoding $(Y, \{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]})$ consisting of a function encoding Y and a set of labels $\{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]}$.

$(Y', \{s'_i\}_{i\in[\ell_{\mathsf{in}}(f)]}) \leftarrow \mathsf{En}^*(\mathsf{pp}, (Y, \{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]}), r)$**:** Given an encoding $(Y, \{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]})$ and randomness r, this algorithm outputs the next encoding $(Y', \{s'_i\}_{i\in[\ell_{\mathsf{in}}(f)]})$.

$Z_i \leftarrow \mathsf{Combine}(\mathsf{pp}, \{e_{ij}^{x_i}\}_j, s_i)$**:** Given a set of values $\{e_{ij}^{x_i}\}_j$ and a set of labels s_i for the i-th input, this algorithm outputs an input encoding Z_i for the i-th input.

$y \leftarrow \mathsf{Dec}(\mathsf{pp}, Y, \{Z_i\}_{i\in[\ell_{\mathsf{in}}(f)]})$**:** Given a function encoding Y and an input encoding $\{Z_i\}_{i\in[\ell_{\mathsf{in}}(f)]}$, this algorithm outputs a decoding $y \in \mathcal{Y}$.

We denote the randomness spaces of PreEn, En, and En^* by R_{PreEn}, R_{En}, and R_{En^*}, respectively.

We require an iDRE scheme to satisfy correctness and privacy. To aid with their definitions, we define a run of iDRE via the PPT algorithm $\Pi(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f, x)$, which, given $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $t \in [t_{\mathsf{max}}]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$, $f \in \mathcal{F}^\lambda_{\mathsf{iDRE}}$ and $x \in \{0,1\}^{\ell_{\mathsf{in}}(f)}$, computes

$$
\begin{aligned}
r_1, \ldots, r_t &\leftarrow\$ R_{\mathsf{PreEn}}, \hat{r}_1 \leftarrow\$ R_{\mathsf{En}}, \hat{r}_2, \ldots, \hat{r}_t \leftarrow\$ R_{\mathsf{En}^*} \\
\{e_{ij}^0, e_{ij}^1\}_{i\in[\ell_{\mathsf{in}}(f)]} &\leftarrow \mathsf{PreEn}(\mathsf{pp}, j; r_j) \quad \forall j \in [t], \\
B_j &\leftarrow \begin{cases} \mathsf{En}(\mathsf{pp}, f, r_1; \hat{r}_1) & \text{if } j = 1 \\ \mathsf{En}^*(\mathsf{pp}, B_{j-1}, r_j; \hat{r}_j) & \text{if } j \in \{2, \ldots, t\} \end{cases} \quad \forall j \in [t], \\
(Y, \{s_i\}_{i\in[\ell_{\mathsf{in}}(f)]}) &\leftarrow B_t, \\
Z_i &\leftarrow \mathsf{Combine}(\mathsf{pp}, \{e_{ij}^{x_i}\}_{j\in[t]}, s_i) \quad \forall i \in [\ell_{\mathsf{in}}(f)],
\end{aligned}
$$

and outputs $\{e_{ij}^b, B_j, Z_i, r_j, \hat{r}_j\}_{i\in[\ell_{\mathsf{in}}(f)], j\in[t], b\in\{0,1\}}$. Note that randomness r_j is shared by both PreEn and En / En^*. This is to ensure that the latter can reconstruct the values sampled by PreEn using said randomness.

Correctness states that decoding the encoding obtained after running iDRE should almost always result in the correct output.

Definition 13 (Statistical Correctness)**.** We say that $\mathsf{iDRE} = (\mathsf{Setup}, \mathsf{PreEn}, \mathsf{En}, \mathsf{En}^*, \mathsf{Combine}, \mathsf{Dec})$ has *statistical correctness* if for all $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $t \in [t_{\mathsf{max}}]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$, $f \in \mathcal{F}^\lambda_{\mathsf{iDRE}}$, and $x \in \{0,1\}^{\ell_{\mathsf{in}}(f)}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$
\Pr\left[\mathsf{Dec}(\mathsf{pp}, Y, \{Z_i\}_{i\in[\ell_{\mathsf{in}}(f)]}) \neq f(x) \;\middle|\; \begin{array}{c} \{\cdot, B_j, Z_i, \cdot, \cdot\}_{i,j} \leftarrow \Pi(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f, x) \\ (Y, \cdot) = B_t \end{array}\right] \leq \mathsf{negl}(\lambda).
$$

Privacy states that an honest-but-curious adversary that corrupts all encoders except for the j^*-th and can select some subset $C \subseteq [\ell_{\mathsf{in}}(f)]$ of the bits of input x should not learn anything about the remaining input bits $\{x_i\}_{i\notin C}$ except for what is leaked by function f and its output $f(x)$.

Definition 14 (Computational Privacy)**.** We say that $\mathsf{iDRE} = (\mathsf{Setup}, \mathsf{PreEn}, \mathsf{En}, \mathsf{En}^*, \mathsf{Combine}, \mathsf{Dec})$ has *computational privacy* if there exists a PPT simulator Sim such that for all $\lambda, t_{\mathsf{max}} \in \mathbb{N}$, $t \in [t_{\mathsf{max}}]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$, $f \in \mathcal{F}^\lambda_{\mathsf{iDRE}}$, $x \in \{0,1\}^{\ell_{\mathsf{in}}(f)}$, $j^* \in [t]$, and $C \subseteq [\ell_{\mathsf{in}}(f)]$ it holds that

$$
\begin{aligned}
&\left\{\mathsf{Sim}(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f, f(x), j^*, \{x_i\}_{i\in C})\right\} \\
&\overset{\mathrm{c}}{\approx} \left\{\{r_j, \hat{r}_j\}_{j\neq j^*}, B_{j^*}, \{e_{ij^*}^{x_i}\}_{i\in C}, \{Z_i\}_{i\notin C} \;\middle|\; \{e_{ij}^b, B_j, Z_i, r_j, \hat{r}_j\}_{i,j,b} \leftarrow \Pi(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f, x)\right\}
\end{aligned}
$$

6.1 Constructing iDRE from RGS and LKMHE

We now show how to construct an iDRE based on our RGS and an LKMHE scheme. An LKMHE scheme is essentially a 2LKMHE scheme but with only one key. One can easily adapt our construction of 2LKMHE from Section 5 to support only a single key, for example by adapting Enc to fix the second key $\mathbf{s}_1 \coloneqq \mathbf{0}^n$ and similarly setting the second key transformation in Eval to $\boldsymbol{\sigma}_1 \coloneqq \mathbf{0}^n$. This construction then also supports the same security notions, i.e. correctness, key privacy, IND-CPA security, and KMH rerandomizability, albeit simplified to the single-key setting. For the sake of brevity we will not redefine these single-key variants.

Construction of iDRE. Let RGS be the (levelled) RGS scheme from Section 4.2 for function family $\mathcal{F}^{\lambda}_{\mathsf{RGS}} = \{f : \{0,1\}^{\ell_{\mathsf{in}}(f)} \to \{0,1\}^{\ell_{\mathsf{out}}(f)}\}$ and with label space $\mathcal{L}_f = (\{0,1\} \times \mathcal{K}_{\mathsf{RGS}} \times \{0,1\})^{2\ell_{\mathsf{in}}(f)}$. Let LKMHE be a (levelled) single-key LKMHE scheme with message space $\mathcal{M}_{\mathsf{LKMHE}} = \mathcal{K}_{\mathsf{RGS}}$, and where the set of message transformations $\mathcal{F}_{\mathrm{msg}}$ is exactly the same as the set of key transformations of the 2LKMHE scheme used to construct RGS. Furthermore, we assume that the input wires $\mathcal{W}_I$ of the Boolean circuit representation of function f are numbered from 1 to $\ell_{\mathsf{in}}(f)$.

Our construction of iDRE for function family $\mathcal{F}^{\lambda}_{\mathsf{iDRE}} \coloneqq \mathcal{F}^{\lambda}_{\mathsf{RGS}}$ from RGS and LKMHE is given in Figure 7. We have marked the differences to the construction in [AHKP22] using red text.

The basic idea of the construction is that the first encoder will, when running En, garble the circuit f and then output the garbling. Then each subsequent encoder will run En^* to rerandomize the garbling via RGS.Rerand, and pass the rerandomized garbling to the next encoder. At the end, the final rerandomized garbling can be evaluated using Combine and Dec to obtain the desired output $f(x)$.

Note that evaluation requires the correct input labels for the garbled circuit to be available. Therefore, each encoder must also provide the (rerandomized) input labels along with the (rerandomized) garbling. To ensure that the labels are not leaked in clear to a corrupted encoder, they must also be encrypted. To keep the encrypted labels consistent with the labels inside the garbling, the used encryption scheme has to support a message homomorphism which allows us to apply the label rerandomization function π_{En} output by RGS.Rerand to those encrypted labels. The π_{En} of our RGS construction could be applied using a fully-homomorphic encryption scheme, but we created a way to use a single-key LKMHE scheme instead by relying on our double-column garbling technique. For further details we refer to the construction itself in Figure 7.

6.2 Correctness and Privacy

We proceed with proving that the iDRE in Figure 7 is correct and secure, i.e. it has privacy.

Theorem 8. *Assuming* RGS *and* LKMHE *are statistically correct, the iDRE in Figure 7 is statistically correct as well (Definition 13).*

Proof. Correctness of LKMHE ensures that LKMHE.Eval rerandomizes the encrypted input labels in c^b_{ij} consistently with the label encoding function π^j_{En}. Then, since the inactive label will with overwhelming probability decrypt to $\perp$, Combine will pick the correct Z^b_i with overwhelming probability as well. Lastly, correctness of RGS ensures that the garbled circuit is evaluated correctly, leading to output $f(x)$ with overwhelming probability. □

Theorem 9. *Assuming* RGS *has computational garbling and rerand privacy (Definitions 10 and 11), and* LKMHE *has computational key-privacy, IND-CPA security, and KMH rerandomizability, the iDRE in Figure 7 has computational privacy (Definition 14).*

Proof. The structure of the proof is highly similar to the proof of privacy of the iDRE construction in [AHKP22]. However, due to the changes made to their construction as well as the different RGS and LKMHE definitions we use, we provide a full proof. We will first define the simulator and then show that its output is indistinguishable from the real world using a sequence of hybrids.

The simulator $\mathsf{Sim}(\mathsf{pp}, 1^\lambda, t_{\mathsf{max}}, t, f, f(x), j^*, \{x_i\}_{i \in C})$ works as follows:

1. Sample random input bits $\{x'_i\}_{i \notin C}$ and define $x' = \{x'_i\}_{i \notin C} \cup \{x_i\}_{i \in C}$. Pick function $f' \in \mathcal{F}^{\lambda}_{\mathsf{iDRE}}$ such that $\phi_\lambda(f') = \phi_\lambda(f)$ and $f'(x') = f(x)$. Then compute $(\mathcal{C}_1, L_1) \leftarrow \mathsf{RGS.Gb}(f')$.

$\mathsf{Setup}(1^\lambda, t_{\mathsf{max}})$:

return $\mathsf{pp} = (\mathsf{pp}_{\mathsf{RGS}}, \mathsf{pp}_{\mathsf{LKMHE}}) \leftarrow (\mathsf{RGS.Setup}(1^\lambda, t_{\mathsf{max}}), \mathsf{LKMHE.Setup}(1^\lambda, t_{\mathsf{max}}))$

$\mathsf{PreEn}(\mathsf{pp}, j; r_j)$:

// Use randomness r_j to sample any values below

if $j = 1$ **then**

$e^b_{i1} = K^b_{i1} \leftarrow \mathsf{LKMHE.KGen}(\mathsf{pp}_{\mathsf{LKMHE}}) \quad \forall b \in \{0,1\}, \forall i \in [\ell_{\mathsf{in}}(f)]$

else

$e^b_{ij} = \rho^b_{ij} \leftarrow\!\!\$\ \mathcal{F}_{\mathrm{key}} \quad \forall b \in \{0,1\}, \forall i \in [\ell_{\mathsf{in}}(f)]$

return $\{e^0_{ij}, e^1_{ij}\}_{i \in [\ell_{\mathsf{in}}(f)]}$

$\mathsf{En}(\mathsf{pp}, f, \hat{r}_1)$:

$\{K^b_{i1}\}_{i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}} \leftarrow \mathsf{PreEn}(1, r_1)$

$(\mathcal{C}_1, L_1) \leftarrow \mathsf{RGS.Gb}(f)$

$\{(\mathsf{sk}^b_i, z^b_i)\}_{i \in [\ell_{\mathsf{in}}(f)]} \leftarrow \mathsf{RGS.En}(L_1, b^{\ell_{\mathsf{in}}(f)}) \quad \forall b \in \{0,1\}$

for $i = 1..\ell_{\mathsf{in}}(f), b = 0..1$ **do**

if $z^b_i = b$ **then** $k^{bb}_i \leftarrow \mathsf{sk}^b_i$; $\mathsf{mode}^{bb}_i \leftarrow \mathsf{hom}$; $k^{b1-b}_i \leftarrow \perp$; $\mathsf{mode}^{b1-b}_i \leftarrow \mathsf{lossy}$

else $k^{b1-b}_i \leftarrow \mathsf{sk}^b_i$; $\mathsf{mode}^{b1-b}_i \leftarrow \mathsf{hom}$; $k^{bb}_i \leftarrow \perp$; $\mathsf{mode}^{bb}_i \leftarrow \mathsf{lossy}$; **endif**

$c^b_{i1} \leftarrow \left(\mathsf{LKMHE.Enc}(K^b_{i1}, k^{b0}_i, \mathsf{mode}^{b0}_i), \mathsf{LKMHE.Enc}(K^b_{i1}, k^{b1}_i, \mathsf{mode}^{b1}_i)\right)$

return $B_1 \leftarrow (\mathcal{C}_1, \{c^b_{i1}\}_{i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}})$

$\mathsf{En}^*(\mathsf{pp}, B_{j-1}, \hat{r}_j)$:

Parse $B_{j-1} = (\mathcal{C}_{j-1}, \{c^b_{ij-1}\}_{i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}})$

$\{\rho^b_{ij}\}_{i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}} \leftarrow \mathsf{PreEn}(j, r_j)$

$(\mathcal{C}_j, \pi^j_{\mathsf{En}}) \leftarrow \mathsf{RGS.Rerand}(\mathcal{C}_{j-1})$

Parse π^j_{En} as function $(\beta, \mathsf{sk}_i, z) \mapsto (\beta, \sigma_i^{|z-b_i|}(\mathsf{sk}_i), |z - b_i|)$

for $i = 1..\ell_{\mathsf{in}}(f), b = 0..1$ **do**

Parse $(c^{b0}_{ij-1}, c^{b1}_{ij-1}) = c^b_{ij-1}$

// If $b_i = 1$, then we must swap c^{b0}_{ij-1} and c^{b1}_{ij-1} to match π^j_{En}

$c^b_{ij} \leftarrow \left(\mathsf{LKMHE.Eval}(\rho^b_{ij}, \sigma^0_i, c^{b|0-b_i|}_{ij-1}), \mathsf{LKMHE.Eval}(\rho^b_{ij}, \sigma^1_i, c^{b|1-b_i|}_{ij-1})\right)$

return $B_j \leftarrow (\mathcal{C}_j, \{c^b_{ij}\}_{i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}})$

$\mathsf{Combine}(\{e^{x_i}_{ij}\}_{j \in [t]}, s_i)$:

Parse $\{e^{x_i}_{ij}\}_{j \in [t]} = (K^{x_i}_{i1}, \rho^{x_i}_{i2}, \ldots, \rho^{x_i}_{it})$

Parse $s_i = (c^0_{it}, c^1_{it}) = ((c^{00}_{it}, c^{01}_{it}), (c^{10}_{it}, c^{11}_{it}))$

$K^{x_i}_{it} \leftarrow \rho^{x_i}_{it} \circ \cdots \circ \rho^{x_i}_{i2}(K^{x_i}_{i1})$

$Z^b_i \leftarrow \mathsf{LKMHE.Dec}(K^{x_i}_{it}, c^{x_i b}_{it}) \quad \forall b \in \{0,1\}$

if $Z^0_i \neq \perp$ **then return** $Z_i \leftarrow (Z^0_i, 0)$ **else return** $Z_i \leftarrow (Z^1_i, 1)$

$\mathsf{Dec}(\mathsf{pp}, Y, \{Z_i\}_{i \in [\ell_{\mathsf{in}}(f)]})$:

return $\mathsf{RGS.Ev}(Y, \{Z_i\}_{i \in [\ell_{\mathsf{in}}(f)]})$

Figure 7: Construction of iDRE from RGS and LKMHE. $\perp$ denotes the lossyness indicator message from LKMHE.

2. Sample $\{r_j, \hat{r}_j\}_{j\in[t]}$ uniformly as done by Π and compute $\{e^0_{ij}, e^1_{ij}\}_{i\in[\ell_{\mathsf{in}}(f)]} \leftarrow \mathsf{PreEn}(j, r_j)$ for all $j \in [t]$.

3. If $j^* = 1$, then compute active input labels $I \leftarrow \mathsf{RGS.En}(L_1, x')$ where we can parse $\{(\mathsf{sk}_i^{x'_i}, z_i^{x'_i})\}_{i\in[\ell_{\mathsf{in}}(f)]} = I$. Compute the ciphertexts $\{c^b_{ij^*}\}_{i\in[\ell_{\mathsf{in}}(f)],b\in\{0,1\}}$ as done in En but using $\mathsf{sk}_i^{\neg x'_i} \leftarrow \mathsf{sk}_i^{x'_i}$ and $z_i^{\neg x'_i} \leftarrow z_i^{x'_i}$. This has the effect that $c^0_{ij^*}$ and $c^1_{ij^*}$ both contain the same active label $\mathsf{sk}_i^{x'_i}$. Finally, set $B_{j^*} \leftarrow (\mathcal{C}_1, \{c^b_{ij^*}\}_{i\in[\ell_{\mathsf{in}}(f)],b\in\{0,1\}})$.

4. If $j^* \neq 1$, then additionally rerandomize by computing $(\mathcal{C}_j, \pi^j_{\mathsf{En}}) \leftarrow \mathsf{RGS.Rerand}(\mathcal{C}_{j-1})$ for all $j \in \{2, \dots, j^*\}$, compute active input labels $I \leftarrow \mathsf{RGS.En}(\pi^{j^*}_{\mathsf{En}}(\dots \pi^2_{\mathsf{En}}(L_1)\dots), x')$, and parse $\{(\widehat{\mathsf{sk}}_i^{x'_i}, \hat{z}_i^{x'_i})\}_{i\in[\ell_{\mathsf{in}}(f)]} = I$. Compute $\{c^b_{ij^*}\}_{i\in[\ell_{\mathsf{in}}(f)],b\in\{0,1\}}$ as follows:

   (a) First compute each c^b_{i1} as in $\mathsf{En}(f', r_1)$ but using the already transformed active labels $\widehat{\mathsf{sk}}_i^{x'_i}$ both as active label and inactive label, so overwriting $\mathsf{sk}^b_i \leftarrow \widehat{\mathsf{sk}}_i^{x'_i}$ and $z^b_i \leftarrow \hat{z}^b_i$ for all $b \in \{0,1\}$. Therefore the c^b_{i1} only encrypt the active labels of $\mathcal{C}_{j^*}$. Additionally, instead of using key K^b_{i1} to compute the ciphertexts inside c^b_{i1}, use the transformed key $\rho^b_{ij^*} \circ \dots \circ \rho^b_{i2}(K^b_{i1})$.

   (b) Then, continue similarly to En^*, by setting

$$c^b_{ij} \leftarrow \big(\mathsf{LKMHE.Eval}(\mathsf{id}, \mathsf{id}, c^{b0}_{ij-1}), \mathsf{LKMHE.Eval}(\mathsf{id}, \mathsf{id}, c^{b1}_{ij-1})\big)$$
$$\forall i \in [\ell_{\mathsf{in}}(f)], b \in \{0,1\}$$

   for all $j \in \{2, \dots, j^*\}$.

   Finally, set $B_{j^*} \leftarrow (\mathcal{C}_{j^*}, \{c^b_{ij^*}\}_{i\in[\ell_{\mathsf{in}}(f)],b\in\{0,1\}})$.

5. Compute $B_j \leftarrow \mathsf{En}^*(B_{j-1}, r_j)$ for all $j \in \{j^*+1, \dots, t\}$. Finally, run $\mathsf{Combine}(\{e^{x'_i}_{ij}\}, s_i)$ using the s_i contained in B_t to obtain $\{Z_i\}_{i\notin C}$, and output the simulation

$$\{r_j, \hat{r}_j\}_{j\neq j^*}, B_{j^*}, \{e^{x_i}_{ij^*}\}_{i\in C}, \{Z_i\}_{i\notin C}.$$

   Note that for the $i \notin C$, using $\{e^{x'_i}_{ij}\}$ for the call to $\mathsf{Combine}$ essentially corresponds to picking at random whether to decrypt c^0_{it} or c^1_{it}.

We now show that the output of the simulator Sim is indistinguishable from the real world, i.e. the right side of Definition 14, via a sequence of hybrids.

H_0**:** This is the ideal world, i.e. the output as computed by the simulator Sim above.

H_1**:** Instead of computing $\mathcal{C}_{j^*}$ and active labels I as in Sim, use the garbling privacy simulator $\mathsf{Sim}_{\mathsf{Gb}}$ to compute them via $(\mathcal{C}_{j^*}, I) \leftarrow \mathsf{Sim}_{\mathsf{Gb}}(1^\lambda, t_{\mathsf{max}}, j^*, \mathsf{pp}_{\mathsf{RGS}}, f'(x'), \phi_\lambda(f'))$. $H_0 \overset{\mathrm{c}}{\approx} H_1$ follows by garbling privacy of RGS.

H_2**:** We now change the inputs to $\mathsf{Sim}_{\mathsf{Gb}}$. Specifically, compute $(\mathcal{C}_{j^*}, I) \leftarrow \mathsf{Sim}_{\mathsf{Gb}}(1^\lambda, t_{\mathsf{max}}, j^*, \mathsf{pp}_{\mathsf{RGS}}, f(x), \phi_\lambda(f))$ using the real function f and input x. This step follows since $f'(x') = f(x)$ and $\phi_\lambda(f') = \phi_\lambda(f)$, so we did not actually change the inputs to $\mathsf{Sim}_{\mathsf{Gb}}$. Therefore H_1 and H_2 are equally distributed.

H_3**:** Switch back from using $\mathsf{Sim}_{\mathsf{Gb}}$ to computing $\mathcal{C}_{j^*}$ and I via $\mathsf{RGS.Gb}$, $\mathsf{RGS.Rerand}$, and $\mathsf{RGS.En}$. Note that now we have replaced usage of x' with x and f' with f, respectively. $H_2 \overset{\mathrm{c}}{\approx} H_3$ again follows by garbling privacy of RGS.

H_4: Replace the randomness used for $\mathsf{RGS.Gb}(f)$ and $\mathsf{RGS.Rerand}(\mathcal{C}_{j-1})$, for $j \in \{2, \ldots, j^* - 1\}$ to use real-world randomness $\{\hat{r}_j\}_{j\in\{1,\ldots,j^*-1\}}$. $H_3 \overset{c}{\approx} H_4$ follows by rerand privacy of RGS.

H_5: Replace each transformed key $\rho_{ij^*}^{\neg x_i} \circ \cdots \circ \rho_{i2}^{\neg x_i}(K_{i1}^{\neg x_i})$ used in step 4a) with a fresh and independently sampled key. Since each $\rho_{ij^*}^{\neg x_i}$ is uniform and secret, $H_4 \overset{c}{\approx} H_5$ follows by key privacy of LKMHE. Note that if $j^* = 1$, then $K_{i1}^{\neg x_i}$ is fresh and independently sampled already, so this hybrid is unnecessary.

H_6: Change the $\{c_{ij^*}^b\}_{i\in[\ell_{\mathsf{in}}(f)],b\in\{0,1\}}$ to encrypt both active and inactive labels. Specifically, change step 3 of Sim to compute $\{(\mathsf{sk}_i^b, z_i^b)\}_{i\in[\ell_{\mathsf{in}}(f)]} \leftarrow \mathsf{RGS.En}(L_1, b^{\ell_{\mathsf{in}}(f)})$, for $b \in \{0,1\}$, and then compute the $c_{ij^*}^b$ based on the corresponding label (sk_i^b, z_i^b) as in the real world. Step 4 of Sim is changed similarly by using $\mathsf{RGS.En}$ to obtain both active and inactive input labels and then encrypting said labels in the c_{i1}^b. Since we are only changing the messages and encryption modes of ciphertexts $c_{i1}^{\neg x_i}$ whose keys were replaced by fresh ones in H_5, $H_5 \overset{c}{\approx} H_6$ follows by CPA security of LKMHE.

H_7: Undo the changes made in H_5. $H_6 \overset{c}{\approx} H_7$ follows again by key privacy of LKMHE.

H_8: We change step 4 of Sim again: Instead of immediately encrypting the rerandomized labels output by $\mathsf{RGS.En}$ inside the c_{i1}^b, we first encrypt the initial labels in L_1 to obtain the c_{i1}^b. Then we create the c_{ij}^b, for $j \in \{2, \ldots, j^*\}$, via

$$c_{ij}^b \leftarrow \left(\mathsf{LKMHE.Eval}(\rho_{ij}^b, \sigma_i^0, c_{ij-1}^{b|0-b_i|}), \mathsf{LKMHE.Eval}(\rho_{ij}^b, \sigma_i^1, c_{ij-1}^{b|1-b_i|})\right)$$

using the values b_i, σ_i^0, and σ_i^1 obtained from π_{En}^j. Since the randomness $\hat{r}_{j^*}$ is secret and uniform, $H_7 \overset{c}{\approx} H_8$ follows by KMH rerandomizability of LKMHE.

H_8 now corresponds exactly to the real world, where $\left\{\{r_j, \hat{r}_j\}_{j\neq j^*}, B_{j^*}, \{e_{ij^*}^{x_i}\}_{i\in C}, \{Z_i\}_{i\notin C}\right\}$ is computed via $\{e_{ij}^b, B_j, Z_i, r_j, \hat{r}_j\}_{i,j,b} \leftarrow \Pi(1^\lambda, t_{\mathsf{max}}, t, \mathsf{pp}, f, x)$. Therefore, since each hybrid is computationally indistinguishable, the real world and simulation are also computationally indistinguishable, completing the proof. □

7 Conclusion

We presented a new approach to rerandomizable garbling based on double-column garbling. The main conceptual point is that the strong homomorphism hiding requirements appearing in prior constructions are not inherent to rerandomizable garbling itself, but arise from the traditional single-column paradigm. By moving part of the hiding burden from the underlying homomorphic encryption scheme into the structure of the garbling, the double-column approach makes it possible to work with simpler forms of homomorphism, in particular additive ones. This yields a new perspective on the design of rerandomizable garbling and leads to an LWE-based instantiation. To the best of our knowledge, this gives the first route to RGS from lattice assumptions and shows that the primitive is not limited to the discrete-log-based setting.

At the same time, our constructions should mainly be viewed as conceptual. Like many other constructions in this area, they are not yet efficient enough for practical deployment. This is similar to the early development of FHE, where the first schemes were primarily of theoretical interest and only later became substantially more practical. The main value of our work is to identify the new technique of double-column garbling and to show that it admits an LWE-based realization. An important direction for future work is to obtain more practical schemes, either through improved lattice-based 2LKMHE constructions, more efficient garbling methods, or further refinements of the double-column paradigm.

References

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 327–343. USENIX Association, August 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

[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. doi:10.1007/978-3-319-63688-7_1.

[AHKP22] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. SCALES - MPC with small clients and larger ephemeral servers. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 502–531. Springer, Cham, November 2022. doi:10.1007/978-3-031-22365-5_18.

[AHKP24] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. Malicious security for SCALES - outsourced computation with ephemeral servers. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part IX*, volume 14928 of *LNCS*, pages 3–38. Springer, Cham, August 2024. doi:10.1007/978-3-031-68400-5_1.

[AHKP25] Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. Towards building efficient SCALES protocols. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 35–67. Springer, Singapore, December 2025. doi:10.1007/978-981-95-5116-3_2.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_29.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 491–520. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_17.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 297–314. Springer, Berlin, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_17.

[APS15] Martin Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9, 10 2015. doi:10.1515/jmc-2015-0016.

[Ban95] W Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in rn. *Discrete & Computational Geometry*, 13(2):217–231, March 1995.

[BCGL+25] James Bell-Clark, Adrià Gascón, Baiyu Li, Mariana Raykova, and Phillipp Schoppmann. Willow: Secure aggregation with one-shot clients. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VIII*, volume 16007 of *LNCS*, pages 285–318. Springer, Cham, August 2025. doi:10.1007/978-3-032-01913-4_9.

[BdPMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 62–89. Springer, Berlin, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_3.

[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. doi:10.1007/978-3-642-55220-5_30.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, Berlin, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_7.

[BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 724–753. Springer, Cham, October 2021. doi:10.1007/978-3-030-77886-6_25.

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Cham, November 2020. doi:10.1007/978-3-030-64378-2_13.

[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. doi:10.1145/100216.100287.

[CC17] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 446–476. Springer, Cham, April / May 2017. doi:10.1007/978-3-319-56620-7_16.

[DS16] Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 294–310. Springer, Berlin, Heidelberg, May 2016. doi:10.1007/978-3-662-49890-3_12.

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, Berlin, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_9.

[GIK+24] Adrià Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. Computationally secure aggregation and private information retrieval in the shuffle model. In Bo Luo, Xiaojing Liao, Jun Xu, Engin

Kirda, and David Lie, editors, *ACM CCS 2024*, pages 4122–4136. ACM Press, October 2024. doi:10.1145/3658644.3670391.

[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. doi:10.1145/2488608.2488678.

[GMP19] Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 655–684. Springer, Cham, May 2019. doi:10.1007/978-3-030-17656-3_23.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. doi:10.1145/1374376.1374407.

[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. doi:10.1007/978-3-642-40041-4_5.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023. doi:10.1109/FOCS57990.2023.00031.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 132–150. Springer, Berlin, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_8.

[HvdHJ25] Raphael Heitjohann, Jonas von der Heyden, and Tibor Jager. Rerandomizable garbling, revisited. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part IV*, volume 16003 of *LNCS*, pages 619–651. Springer, Cham, August 2025. doi:10.1007/978-3-032-01884-7_20.

[LLW21] Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 498–527. Springer, Cham, October 2021. doi:10.1007/978-3-030-77870-5_18.

[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. doi:10.1007/s00145-008-9036-8.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_41.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007. arXiv:https://doi.org/10.1137/S0097539705447360, doi:10.1137/S0097539705447360.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Berlin, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_22.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. doi:10.1145/1060590.1060603.

[SSE+24] Ron Steinfeld, Amin Sakzad, Muhammed F. Esgin, Veronika Kuchta, Mert Yassi, and Raymond K. Zhao. LUNA: Quasi-optimally succinct designated-verifier zero-knowledge arguments from lattices. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3167–3181. ACM Press, October 2024. doi:10.1145/3658644.3670345.

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Cham, November 2022. doi:10.1007/978-3-031-22318-1_23.

[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. doi:10.1109/SFCS.1986.25.

[ZLXL23] Qingsong Zhao, Ximeng Liu, Huanliang Xu, and Yanbin Li. Practical reusable garbled circuits with parallel updates. *Comput. Stand. Interfaces*, 86:103721, 2023. URL: https://doi.org/10.1016/j.csi.2023.103721, doi:10.1016/J.CSI.2023.103721.