# Efficient Laconic Cryptography from Learning With Errors

(Full Version)

Nico Döttling[1], Dimitris Kolonelos[2,3], Russell W. F. Lai[4], Chuanwei Lin[1], Giulio Malavolta[5], and Ahmadreza Rahimi[5]

[1] CISPA Helmholtz Center for Information Security
nico.doettling@gmail.com, chuanwei.lin@cispa.de
[2] IMDEA Software Institute
dimitris.kolonelos@imdea.org
[3] Universidad Politecnica de Madrid
[4] Aalto University
russell.lai@aalto.fi
[5] Max Planck Institute for Security and Privacy
giulio.malavolta@hotmail.it, ahmadreza.rahimi@mpi-sp.org

**Abstract.** Laconic cryptography is an emerging paradigm that enables cryptographic primitives with sublinear communication complexity in just two messages. In particular, a two-message protocol between Alice and Bob is called *laconic* if its communication and computation complexity are essentially independent of the size of Alice's input. This can be thought of as a dual notion of fully-homomorphic encryption, as it enables "Bob-optimized" protocols. This paradigm has led to tremendous progress in recent years. However, all existing constructions of laconic primitives are considered only of *theoretical interest*: They all rely on non-black-box cryptographic techniques, which are highly impractical.

This work shows that non-black-box techniques are not necessary for basic laconic cryptography primitives. We propose a *completely algebraic* construction of laconic encryption, a notion that we introduce in this work, which serves as the cornerstone of our framework. We prove that the scheme is secure under the standard Learning With Errors assumption (with polynomial modulus-to-noise ratio). We provide proof-of-concept implementations for the first time for laconic primitives, demonstrating the construction is indeed practical: For a database size of $2^{50}$, encryption and decryption are in the order of single digit *milliseconds*.

Laconic encryption can be used as a black box to construct other laconic primitives. Specifically, we show how to construct:

- Laconic oblivious transfer
- Registration-based encryption scheme
- Laconic private-set intersection protocol

All of the above have essentially optimal parameters and similar practical efficiency. Furthermore, our laconic encryption can be preprocessed such that the online encryption step is entirely combinatorial and therefore much more efficient. Using similar techniques, we also obtain identity-based encryption with an unbounded identity space and tight security proof (in the standard model).

# Table of Contents

## 1 Introduction

Laconic cryptography [CDG+17,QWW18,DGI+19,DGGM19] is an emerging paradigm to securely compute on large amounts of data in just two messages, while incurring very small communication. Specifically, in the laconic setting the receiver Alice has an input of very large size, whereas we typically think of the sender Bob's input as smaller in size. In the first message, Alice publishes a succinct hash $h$ of her input $D$, which may be thought of as a large database $D \in \{0,1\}^n$. Such a compressing hash function cannot be unkeyed, therefore laconic protocols also rely on public parameters, which are typically also required to be succinct[6]. Given the hash $h$, Bob can encrypt his input $x$ with respect to $h$, obtaining a succinct ciphertext $\mathsf{ctxt}$. Importantly, the workload of Bob should also be independent of $n$. Such a ciphertext $\mathsf{ctxt}$ enables Alice to compute a joint function of her input $D$ and Bob's input $x$, while Bob has the guarantee that Alice learns nothing but the legitimate function output. The specific choice of the function $f$ computed by such a protocol leads to different laconic primitives:

- In laconic OT [CDG+17], Bob's input consists of an index $i$ and two messages $m_0$ and $m_1$. The function $f$ is given by $f(D,(i,m_0,m_1)) = (i, m_{D[i]})$, i.e. Alice learns the index $i$, and if the $i$-th bit of the database $D$ is 0 she learns $m_0$, otherwise $m_1$. The setting of laconic OT typically imposes an additional efficiency requirement concerning Alice. Concretely, we require Alice's second phase to have a runtime essentially independent of $n$.
- In laconic function evaluation (LFE) [QWW18], Alice's input $D$ is a (large) boolean circuit $C$, and the function computed by an LFE protocol is $f(C,x) = C(x)$. The construction provided in [QWW18] satisfies a somewhat relaxed succinctness guarantee: While the size of the communication does not scale with the size of the circuit $C$, it scales polynomially with the depth of $C$. Furthermore, the runtime of the second phase of Alice scales linearly with the size of $C$.

**Implications.** The notion of laconic OT in particular has had broader implications: The core-ideas underlying laconic OT led to a series of constructions of identity-based encryption (IBE) from weaker assumptions [DG17b,DG17a,DGHM18,BLSV18] and gave rise to the notion of registration-based encryption (RBE) [GHMR18,GHM+19,GV20]. These constructions make essential use of the above-mentioned more stringent efficiency-property of the laconic OT constructions they are based on. Consequently, these primitives are not known to be generically constructible from LFE.

Furthermore, the techniques developed in the context of laconic cryptography were key to making progress on a broad range of problems: trapdoor functions from the computational Diffie-Hellman assumption [GH18], private-information retrieval (PIR) from the decisional Diffie-Hellman assumption [DGI+19], two-round multi-party computation protocols from minimal assumptions [GS17,GS18b,BL18], adaptively secure garbled circuits [GS18a], laconic conditional disclosure of secrets [DGGM19], and laconic private set intersection [ABD+21,ALOS22].

**Reverse Delegation.** Laconic cryptography can be seen as enabling *reverse delegation* without requiring additional rounds of communication. In a standard delegation scheme, a user outsources its computation to an untrusted server with the goal of learning the output while keeping its input private. The canonical cryptographic tool that enables delegation is fully-homomorphic encryption (FHE) [Gen09], since it allows the server to perform the computation without knowing the user's input. Reverse delegation allows a user (Bob, in our previous example) to delegate the computation completely to the server (Alice) while also letting her learn the output of the computation and nothing beyond that. For instance, [CDG+17] provided a protocol to let Bob reverse-delegate RAM computations to Alice, such that Bob's overhead and the size of the communication scales only with runtime of the RAM program, but not with the size of Alice's (large) input. Likewise, the laconic function evaluation scheme of [QWW18] allows to reverse-delegate circuit computations to Alice, while incurring a communication overhead that only scales with the depth of the circuit.

[6] That is, independent or at least sublinear in $n$.

**A Non-Blackbox "Barrier" for Practicality.** So far, the aforementioned progress in designing new cryptographic primitives has been almost exclusively of *theoretical interest*. In essence, the lack of practicality of these new solutions can be explained by their *non-blackbox* use of underlying cryptographic building blocks. For example, essentially all known constructions of laconic OT involve a re-encryption step, also called *deferred encryption* [BLSV18], which gives the receiver Alice the ability to produce ciphertexts under keys that were not known to the sender Bob at the time of encryption. In the above-mentioned constructions, this re-encryption step is implemented using garbled circuits [Yao86] for circuits which perform public-key cryptographic operations. The non-black box use of cryptographic primitives is such a grave source of inefficiency that, to the best of our knowledge, not even the basic laconic OT has ever been implemented as a proof of concept. On a slightly different note, we remark that while the LFE scheme of [QWW18] does not make use of garbled circuits, it relies on a different non-blackbox mechanism based on FHE to bootstrap a weaker notion called attribute-based LFE into fully-fledged LFE.

In summary, the present state of affairs sees laconic cryptography as a powerful theoretical tool for enabling new cryptographic primitives and realizing powerful notions from weaker assumptions. However, the resulting schemes are practically inefficient, thus calling into question the relevance of this framework beyond theoretical feasibility results. Motivated by this gap, we ask:

*Can we realize truly efficient laconic cryptography?*

Towards a positive resolution to this question, it seems insufficient to optimize existing techniques. Instead, a conceptual reworking of basic laconic primitives will be required.

## 1.1 Our Results

This work shows that garbled circuits (and other non-black box cryptographic techniques) are not needed to construct laconic cryptography. We establish a new paradigm for constructing *concretely efficient* laconic cryptographic schemes based on the hardness of the standard learning with errors (LWE) problem with a polynomial modulus-to-noise ratio. In contrast to prior works, we show that our schemes are practical with a proof of concept implementation. In the following, we discuss our contributions in more detail.

**Laconic Encryption.** We propose the notion of *laconic encryption* as the central abstraction of our framework. Laconic encryption allows Alice to construct a binary tree whose leaves are public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ and sends the root of the tree to Bob. Given only the root of the tree and an index $\mathsf{ind}$, Bob can then encrypt a message with respect to $\mathsf{pk}_{\mathsf{ind}}$, which can only be decrypted with the corresponding secret key $\mathsf{sk}_{\mathsf{ind}}$. Such a scheme is called laconic since Alice's message is independent of $n$, as she only sends the root of the tree.

We then show how to construct laconic encryption *efficiently* and with *(asymptotically) optimal parameters* without relying on garbled circuits or other non-black box cryptographic techniques. At a technical level, our construction relies on the algebraic properties of the SIS-based hash tree. It exploits the gadget matrix to efficiently re-encrypt the message layer-by-layer. In order to demonstrate the security of the scheme, we introduce a new variant of the (ring/module) LWE problem, in which the adversary is also given a leakage on the error. Then we prove that this problem is as hard as the standard (ring/module) LWE problem, with an essentially tight reduction. Our proof relies on spectral analysis of positive definite matrices, a subject of independent interest.

**Applications.** We show how laconic encryption enables a wide range of laconic cryptographic primitives with minimal overhead. The following constructions use laconic encryption in a black-box sense, and the additional methods required are combinatorial. That is, all of the resulting schemes are *concretely efficient* and have near-optimal parameters. Specifically, we show how to construct:

– **Laconic OT**: As an immediate application of laconic encryption, we construct a laconic OT protocol with essentially optimal parameters.

– **Registration-Based Encryption**: Registration-based encryption (RBE) is a notion recently introduced in [GHMR18] to solve the key-escrow problem for identity-based encryption (IBE) while preserving the "encrypt with respect to identity" functionality. Laconic encryption enables the first concretely efficient RBE construction that the size of the public parameters scales *logarithmically* with the number of users.
– **Laconic Private-Set Intersection**: Private-set intersection (PSI) allows Alice and Bob to check whether they have a common item in their database without revealing anything about other items. Laconic encryption allows us to construct an efficient *laconic* PSI protocol where the communication complexity is independent of the size of Alice's database.

**Optimizations and Extensions.** We explore a number of optimizations and extensions for our laconic encryption construction. First, we show that the encryption algorithm can be *pre-processed*: In an input-independent offline phase, the encryptor can prepare auxiliary information at essentially the same cost as the encryption algorithm. In an online phase, where the message $\mathsf{msg}$ and the index $\mathsf{ind}$ are known, the encryptor can use the auxiliary information prepared earlier to produce a correctly-formed ciphertext. Importantly, the online phase is entirely combinatorial, and all public-key operations happen in the offline phase.

Second, we explore the possibility of plugging-in different encryption schemes in our construction. Natively, our laconic encryption supports only dual-Regev ciphertexts [GPV08], whereas for some applications it may be desirable to use support other encryption schemes. We show how our scheme can be adapted to support a large class of algorithms, which includes LPN-based encryption [Ale03] and recently NIST-standardized lattice-based schemes [BDK$^+$17]. To solve this challenge, we develop a new special-purpose randomized encoding scheme, which may be of independent interest.

Finally, we show that our construction of laconic encryption can be turned into that of identity-based encryption (IBE) [BF01] with similar efficiency properties. Our IBE is the first scheme that simultaneously achieves: (i) Constant-size public parameters, (ii) an *unbounded* identity space, (iii) a tight proof of (adaptive) security against a standard assumption (specifically, LWE).

**Implementation and Benchmark.** To demonstrate the practicality of our laconic encryption scheme, we implemented a proof of concept in Go (see Section 12). We ran the benchmarks for the scheme with a database size/index space of $2^{50}$ and achieved encryption and decryption times below 10 milliseconds on a personal computer. We believe these times can be improved using further optimizations, which are beyond the scope of this work.

### 1.2 Related Work

We mention prior works that study practical variants of laconic cryptographic primitives. In [ALOS22] the authors show a variant of laconic private-set intersection that is practically efficient and leads to substantial improvements in real-world protocols. However, the variant that is implemented has a long common reference string, linear in the size of Alice's database $D$; thus it is not fully laconic.

In [HLWW22] the authors propose the notion of registered attribute-based encryption, as an extension of the notion of RBE, and they show a constructions based on bilinear pairings. Compared to our work, their scheme has a long common reference string (in fact, quadratic in $n$), the runtime of the key generation and registration algorithms is linear in $n$, and they have an a-priori bound on the number of users. On the flip-side, they achieve the attribute-based functionality, that we do not consider in this work.

Another recent work [GKMR22] proposes the first practically efficient registration-based encryption scheme, and shows the first proof of concept implementation. Contrary to this work, their scheme is asymptotically only sublinear in the size of $D$ (specifically, $\sqrt{n}$ as opposed to $\mathsf{polylog}(n)$), and requires an a-priori bound on $n$. Furthermore, they rely on the hardness of problems over bilinear pairings and thus their scheme is immediately insecure in the quantum settings.

## 2 Technical Overview

We give a brief overview of the new ideas and the technical innovations introduced in this work. We start by describing the new notion of laconic encryption, how to construct it efficiently from the standard LWE assumptions, and the challenges that arise during the security proofs. Then we outline the new cryptographic schemes that are enabled by this new notion and possible optimizations and extensions. In favor of a more intuitive description, the following outline considers the special case of $\mathbb{Z}$-lattices; however, in the technical sections, we prove all of our statements for the more general $\mathcal{R}$-module settings.

## 2.1 Laconic Encryption

Before delving into the description of our scheme, we introduce the syntax of laconic encryption, and we recall how prior work (implicitly) addresses the challenges needed to build this notion.

**Syntax and Properties.** A laconic encryption scheme allows Alice to (iteratively) construct a digest (e.g., via a Merkle hash tree) of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ where $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and $\mathsf{pp}$ can be thought of as a uniformly random string, which is common to all participants. We denote by $\mathsf{st}$ the message that Alice sends to Bob, which consists of the digest (e.g., the root of the Merkle tree). Importantly, the size of $\mathsf{pp}$ and $\mathsf{st}$ is only polynomial in the security parameter, and in particular, it does not depend on $n$. On input a message $\mathsf{msg}$ and an index $\mathsf{ind} \in [n]$, Bob can then compute a ciphertext $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg})$. Correctness requires that anyone possessing the corresponding secret key can decrypt $\mathsf{ctxt}$, more specifically:

$$\mathsf{msg} = \mathsf{Dec}(\mathsf{sk}_{\mathsf{ind}}, \mathsf{wit}_{\mathsf{ind}}, \mathsf{ctxt})$$

where $\mathsf{wit}_{\mathsf{ind}}$ is some (public) auxiliary information, whose size is logarithmic in $n$. The reader can think of this information as being the Merkle tree opening, i.e., the root-to-leaf path, of the key $\mathsf{pk}_{\mathsf{ind}}$. For security, we require that if the adversary does not know the secret key associated with index $\mathsf{ind}$ (or if no key is added to the tree at that particular index), then:

$$\mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg}_0) \approx \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg}_1).$$

In fact, we will require (and prove) a slight strengthening of this property, i.e., that ciphertexts should look pseudorandom to anyone who cannot decrypt them.

**Prior Works.** To gain some intuition on why constructing laconic encryption is a challenging problem, it is useful to recall how prior works [CDG+17] (implicitly) build this cryptographic primitive. Loosely speaking, their main leverage is a construction of a structured two-to-one hash function $\mathsf{Hash}$ (which can be constructed from a variety of computational assumptions) that supports an *encryption* functionality. More specifically, given a digest $d \leftarrow \mathsf{Hash}(D)$, Bob can compute a ciphertext $\mathsf{ctxt} \leftarrow \mathsf{Enc}(d, \mathsf{ind}, (\mathsf{msg}_0, \mathsf{msg}_1))$ that allows Alice (who knows the database $D$) to recover $\mathsf{msg}_{D_{\mathsf{ind}}}$, whereas the message $\mathsf{msg}_{D_{1-\mathsf{ind}}}$ remains computationally hidden. While this looks like a promising start, it should be noted that the hash function is only two-to-one, and therefore the size of the digest $d$ is only half that of the original database $D$. If one were to naively recurse this scheme, the encryption algorithm would quickly start running in exponential time.

To circumvent the runtime issue, the strategy of [CDG+17] is to rely on garbled circuits [Yao86]. More specifically, to boost the compression of the hash function, they define a binary tree of hash values and use garbled circuits to (asymptotically efficiently) implement a re-encryption gadget from one layer to another. Given a digest $d_i \leftarrow \mathsf{Hash}(D_{i+1})$, where $D_{i+1} = (d_{i+1,0}, d_{i+1,1})$ are the digests at a lower layer, the encryption algorithm uses the above procedure to encrypt the *labels* of a garbled circuit, that internally runs the encryption $\mathsf{Enc}$ for the layer below. Crucially, the size of the labels is independent of the size of the garbled circuit (except for its input) and therefore this encryption strategy can be recursed without incurring an exponential blow-up. Although this framework achieves asymptotically optimal parameters, it is prohibitively expensive to use garbled circuits for public-key operations. In contrast, our strategy (described below in detail) will bypass this barrier by leveraging the algebraic properties of a particular hash function.

**Our Approach.** As hinted above, a strategy of constructing laconic encryption is to design a mechanism allowing to "encrypt with respect to a Merkle tree opening", and successfully executing this strategy requires an "encryption-friendly" hash function. Our starting point is the following variant of Ajtai's [Ajt96,GGH96] collision-resistant hash function based on the short integer solution (SIS) assumption:

$$f(\mathbf{x}_0, \mathbf{x}_1) \coloneqq \mathbf{A}_0(-\mathbf{G}^{-1}(\mathbf{x}_0)) + \mathbf{A}_1(-\mathbf{G}^{-1}(\mathbf{x}_1)) \bmod q$$

where $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ are uniformly random matrices with $m \approx n \log q$, $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{Z}_q^n$ are vectors, and $\mathbf{G}^{-1}$ denote the binary-decomposition operator (so that for any $\mathbf{x} \in \mathbb{Z}_q^n$ we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$). A very similar hash function was used in [LLNW16] to build lattice-based Merkle-tree accumulators, ring signatures, and group signatures. At first glance, it may seem that the hash function $f$ is not encryption-friendly since the binary-decomposition operation $\mathbf{G}^{-1}$ is highly non-linear. What enables us to encrypt with respect to a Merkle tree opening is the crucial observation that a hash chain formed by $f$ *induces* a linear relation.

More concretely, consider the Merkle tree built using the hash function $f$ where the node indexed by $\mathsf{str} \in \{0,1\}^*$ is labeled by $\mathbf{y}_{\mathsf{str}}$. Suppose that $\mathbf{u}_{\mathsf{str}} = -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{str}})$ for each $\mathsf{str} \in \{0,1\}^*$. Closing into the top of the tree, we observe that $(\mathbf{u}_0, \mathbf{u}_1)$ is a short (in fact binary) vector satisfying the linear relation:

$$\begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{G} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{y}_\epsilon \\ -\mathbf{y}_0 \end{pmatrix} \bmod q.$$

Where $\mathbf{y}_\epsilon$ is the node denoting the root of the tree. In other words, the vector $(\mathbf{u}_0, \mathbf{u}_1)$ is a valid solution to the (inhomogeneous) SIS instance

$$\left( \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{G} & \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{y}_\epsilon \\ -\mathbf{y}_0 \end{pmatrix} \right).$$

Likewise, $(\mathbf{u}_0, \mathbf{u}_1)$ is also a valid solution to the (inhomogeneous) SIS instance

$$\left( \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{0} & \mathbf{G} \end{pmatrix}, \begin{pmatrix} \mathbf{y}_\epsilon \\ -\mathbf{y}_1 \end{pmatrix} \right).$$

**Dual-Regev Encryption.** It turns out that this structure synergizes remarkably well with the dual-Regev encryption scheme [GPV08]. Recall that in the dual-Regev encryption scheme [GPV08], whose security is based on the standard LWE assumption, a public key is a SIS instance and the corresponding secret key is the SIS solution. Specifically, in the following assume that the matrix $\mathbf{A} = (\mathbf{A}_0\ \mathbf{A}_1)$ is part of the public parameters. Further assume that $\mathbf{y}_0$ and $\mathbf{y}_1$ are dual-Regev public keys with respect to $\mathbf{A}$. That is, for $b \in \{0,1\}$ we generate $\mathbf{y}_b$ by choosing a uniformly random $\mathbf{w}_b \in \{0,1\}^{2m}$ and set $\mathbf{y}_b = \mathbf{A} \cdot \mathbf{w}_b \bmod q$. Here, $\mathbf{w}_b$ is the secret key corresponding to $\mathbf{y}_b$. By the leftover-hash-lemma [HILL99,Reg05], the $\mathbf{y}_b$ are statistically close to uniform. To encrypt a message $\mathsf{msg}$ under $\mathbf{y}_b$, we choose an LWE secret $\mathbf{r}_1$ and compute a ciphertext $(\mathbf{c}_1, d_1)$ via

$$\begin{aligned} \mathbf{c}_1 &\approx \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{A} \bmod q, \\ d_1 &\approx \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{y}_b + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

Here, we use the "$\approx$" notation to omit the LWE error. The function $\mathsf{Encode}(\cdot)$ protects the message $\mathsf{msg}$ against small errors, a popular choice is to encode a message bit $\mathsf{msg}$ in the most-significant bit, i.e. $\mathsf{Encode}(\cdot) = \frac{q}{2} \cdot \mathsf{msg}$. To decrypt a ciphertext $(\mathbf{c}_1, d_1)$ using a secret key $\mathbf{w}_b$ we compute

$$d_1 - \mathbf{c}_1^{\mathsf{T}} \cdot \mathbf{w}_b \approx \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{y}_b + \mathsf{Encode}(\mathsf{msg}) - \mathbf{r}_1^{\mathsf{T}} \cdot \underbrace{\mathbf{A} \cdot \mathbf{w}_b}_{=\mathbf{y}_b} = \mathsf{Encode}(\mathsf{msg}) \bmod q,$$

from which the message $\mathsf{msg}$ can be efficiently recovered.

**Encrypting to Hash Values.** Now assume that we are not given $\mathbf{y}_0$ and $\mathbf{y}_1$, but only their hash value

$$\mathbf{y} = \mathbf{A} \cdot \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{y}_0) \\ -\mathbf{G}^{-1}(\mathbf{y}_1) \end{pmatrix} \bmod q.$$

Our goal is to produce a ciphertext "for the key $\mathbf{y}_b$" given only the hash value $\mathbf{y}$. Towards this goal, let us examine what happens when we generate a dual-Regev encryption scheme with respect to the "public key"

$$\mathsf{pk} := \left( \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{G} & \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} \right),$$

Choosing LWE secrets $\mathbf{r}_0$ and $\mathbf{r}_1$ we compute a ciphertext $\mathsf{ctxt} = (\mathbf{c}, d)$ by

$$\begin{aligned} \mathbf{c}^\mathsf{T} &\approx (\mathbf{r}_0^\mathsf{T}, \mathbf{r}_1^\mathsf{T}) \cdot \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{G} & \mathbf{0} \end{pmatrix} = \mathbf{r}_0^\mathsf{T} \cdot \mathbf{A} + \mathbf{r}_1^\mathsf{T} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \\ d &\approx (\mathbf{r}_0^\mathsf{T}, \mathbf{r}_1^\mathsf{T}) \cdot \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} + \mathsf{Encode}(\mathsf{msg}) = \mathbf{r}_0^\mathsf{T} \cdot \mathbf{y} + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

If we "decrypt" the ciphertext using $(\mathbf{u}_0 = -\mathbf{G}^{-1}(\mathbf{y}_0), \mathbf{u}_1 = -\mathbf{G}^{-1}(\mathbf{y}_1))$ as the secret key, we obtain

$$\begin{aligned} d - \mathbf{c}^\mathsf{T} \cdot \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix} &\approx \mathbf{r}_0^\mathsf{T} \cdot \mathbf{y} + \mathsf{Encode}(\mathsf{msg}) - \mathbf{r}_0^\mathsf{T} \cdot \underbrace{(\mathbf{A}_0\ \mathbf{A}_1) \cdot \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix}}_{=\mathbf{y}} - \mathbf{r}_1^\mathsf{T} \cdot \underbrace{(\mathbf{G}\ \mathbf{0}) \cdot \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix}}_{=-\mathbf{y}_0} \\ &= \mathbf{r}_1^\mathsf{T} \cdot \mathbf{y}_0 + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

Consequently, this "decryption operation" has produced (part of) a ciphertext encrypted under the public key $\mathbf{y}_0$! Analogously, if we use the public key

$$\mathsf{pk} := \left( \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \mathbf{0} & \mathbf{G} \end{pmatrix}, \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} \right),$$

the above decryption operation would result in a ciphertext component $\mathbf{r}_1^\mathsf{T} \cdot \mathbf{y}_1 + \mathsf{Encode}(\mathsf{msg}) \bmod q$. Thus, decryption of such ciphertext with $(\mathbf{u}_0 = -\mathbf{G}^{-1}(\mathbf{y}_0), \mathbf{u}_1 = -\mathbf{G}^{-1}(\mathbf{y}_1))$ is effectively a re-encryption to either public key $\mathbf{y}_0$ or $\mathbf{y}_1$.

To make such a ciphertext decryptable under one of the corresponding secret keys, we add an additional ciphertext component $\mathbf{c}_1^\mathsf{T} = \mathbf{r}_1^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q$ to $\mathsf{ctxt}$. Then, a ciphertext $\mathsf{ctxt}$ for $\mathbf{y}_b$ comprises of

$$\begin{aligned} \mathbf{c}^\mathsf{T} &\approx \mathbf{r}_0^\mathsf{T} \cdot \mathbf{A} + \mathbf{r}_1^\mathsf{T} \cdot ((1-b) \cdot \mathbf{G}\ b \cdot \mathbf{G}) \bmod q \\ \mathbf{c}_1^\mathsf{T} &\approx \mathbf{r}_1^\mathsf{T} \cdot \mathbf{A} \bmod q \\ d &\approx \mathbf{r}_0^\mathsf{T} \cdot \mathbf{y} + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

Finally, observe that it doesn't matter if the $\mathbf{y}_b$ are actually dual-Regev public keys or itself a hash value, the ciphertext structures are identical! Hence, for a larger tree we can apply this mechanism recursively, which results in one additional ciphertext component $\mathbf{c}_i^\mathsf{T} \approx \mathbf{r}_i^\mathsf{T} \cdot \mathbf{A} + \mathbf{r}_1^\mathsf{T} \cdot ((1-b_i) \cdot \mathbf{G}\ b_i \cdot \mathbf{G}) \bmod q$ *per level of the tree*, where the $b_i$ define the path through the tree.

**Security of the Construction.** We will now focus on establishing the security of this construction with the goal of basing security on the LWE assumption. For this purpose, we need to consider the error terms in our construction explicitly. Let $\mathbf{A} = (\mathbf{A}_0\ \mathbf{A}_1)$. A ciphertext $\mathsf{ctxt} = (\mathbf{c}, \mathbf{c}_1, d)$ for $b = 0$ is computed by

$$\begin{aligned} \mathbf{c}^\mathsf{T} &= \mathbf{r}_0^\mathsf{T} \cdot \mathbf{A} + \mathbf{r}_1^\mathsf{T} \cdot (\mathbf{G}\ \mathbf{0}) + \mathbf{e} \bmod q \\ \mathbf{c}_1^\mathsf{T} &= \mathbf{r}_1^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q \end{aligned}$$

$$d = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q,$$

where $\mathbf{e}$, $\mathbf{e}_1$ and $e^*$ are short error vectors.

On the face of it, this looks almost like a classical LWE encryption. Hence, one might try to reduce security directly to the LWE problem. That is, given LWE samples $(\mathbf{A}, \mathbf{v}^{\mathsf{T}} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q)$ and $(\mathbf{y}, v = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + e^* \bmod q)$ we can simulate a ciphertext by computing

$$\begin{aligned} \mathbf{c}^{\mathsf{T}} &= \mathbf{v}^{\mathsf{T}} + \mathbf{r}_1^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \\ \mathbf{c}_1^{\mathsf{T}} &= \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q \\ d &= v + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

By replacing $\mathbf{v}^{\mathsf{T}}$ and $v$ by uniformly random values, as per the LWE assumption, the term $d$ now hides $\mathsf{msg}$ and security follows.

However, upon closer inspection there is a problem with this approach: The matrix $\mathbf{A}$ and the vector $\mathbf{y}^{\mathsf{T}}$ are *not* independent from the view of an adversary. Specifically, the adversary knows an explicit relation between $\mathbf{A}$ and $\mathbf{y}^{\mathsf{T}}$, namely

$$\mathbf{y}^{\mathsf{T}} = \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{y}_0)) + \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{y}_1)) =: \mathbf{A} \cdot \mathbf{z} \bmod q,$$

as $\mathbf{y}_0$ and $\mathbf{y}_1$ are known to the adversary. Here $\mathbf{z} := \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{y}_0) \\ -\mathbf{G}^{-1}(\mathbf{y}_1) \end{pmatrix}$ is a binary (and thus short) vector (denoted $(\mathbf{u}_0, \mathbf{u}_1)$ above). For this reason, $\mathbf{v}^{\mathsf{T}} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q$ and $v = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + e^* \bmod q$ are easily distinguishable from uniformly random values: It holds that $v - \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} = e^* - \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} \bmod q$ is short, whereas for uniformly random $\mathbf{v}^{\mathsf{T}}$ and $v$ this expression is, with high probability, not short.

**Drowning Out Correlations.** However, there is a fairly routine solution to this issue using a technique called *drowning*. The idea is, given LWE samples $(\mathbf{A}, \mathbf{v}^{\mathsf{T}} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e} \bmod q)$, to *simulate* $v$ from $\mathbf{v}$ and $\mathbf{z}$ by computing it via

$$v \approx \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} = (\mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}}) \cdot \mathbf{z} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} \cdot \mathbf{z} + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} \bmod q.$$

Yet, now the error terms in $\mathbf{v}^{\mathsf{T}}$ and $v$ are obliviously correlated. To get rid of this correlation, we can opt to *drown* it out: If $e^*$ is chosen from a suitable short distribution which produces super-polynomially larger values than $\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}$, then it holds that $\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^* \approx_s e^*$, i.e. $\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^*$ and $e^*$ are statistically close. Hence, we can simulate $v$ by computing $v = \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + e^* \bmod q$.

Hence, our security proof now proceeds as follows. Given LWE samples $(\mathbf{A}, \mathbf{v}^{\mathsf{T}} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q)$ we can simulate a ciphertext $\mathsf{ctxt} = (\mathbf{c}, \mathbf{c}_1, d)$ by sampling $e^*$ and setting

$$\begin{aligned} \mathbf{c}^{\mathsf{T}} &= \mathbf{v}^{\mathsf{T}} + \mathbf{r}_1^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \\ \mathbf{c}_1^{\mathsf{T}} &= \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q \\ d &= \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q. \end{aligned}$$

If $(\mathbf{A}, \mathbf{v})$ are well-formed LWE samples, then by the above discussion,

$$\begin{aligned} d &= \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \\ &= \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \\ &\approx_s \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y} + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q, \end{aligned}$$

i.e. such a $\mathsf{ctxt} = (\mathbf{c}, \mathbf{c}_1, d)$ is statistically close to a real ciphertext. Under the LWE assumption, we can now replace $\mathbf{v}$ with a uniformly random $\mathbf{v}'$ and get

$$\mathbf{c}^{\mathsf{T}} = \mathbf{v'}^{\mathsf{T}} + \mathbf{r}_1^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q$$

$$d = \mathbf{v}'^{\mathsf{T}} \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q.$$

Now, since $\mathbf{v}'$ is uniformly random, we can equivalently choose it by computing $\mathbf{v}'^{\mathsf{T}} = \mathbf{v}''^{\mathsf{T}} - \mathbf{r}_1^{\mathsf{T}}(\mathbf{G}\ \mathbf{0})$, where $\mathbf{v}''$ is also chosen uniformly random. That is, we compute $\mathsf{ctxt} = (\mathbf{c}, \mathbf{c}_1, d)$ by

$$\begin{aligned}
\mathbf{c}^{\mathsf{T}} &= \mathbf{v}''^{\mathsf{T}} \\
\mathbf{c}_1^{\mathsf{T}} &= \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q \\
d &= (\mathbf{v}''^{\mathsf{T}} - \mathbf{r}_1^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0})) \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \\
&= \mathbf{v}''^{\mathsf{T}} \cdot \mathbf{z} - \mathbf{r}_1^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \cdot \mathbf{z} + e^* + \mathsf{Encode}(\mathsf{msg}) \\
&= \mathbf{v}''^{\mathsf{T}} \cdot \mathbf{z} + \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{y}_0 + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q,
\end{aligned}$$

as $(\mathbf{G}\ \mathbf{0}) \cdot \mathbf{z} = -\mathbf{y}_0 \bmod q$. Going a step further, we can compute $d$ by $d = \mathbf{v}''^{\mathsf{T}} \cdot \mathbf{z} + d_1 \bmod q$, where $d_1 = \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{y}_0 + e^* + \mathsf{Encode}(\mathsf{msg}) \bmod q$ is the payload part of an encryption of $\mathsf{msg}$ under the public key $\mathbf{y}_0$. In other words, we are now in a situation where we can simulate a ciphertext $\mathsf{ctxt} = (\mathbf{c}, \mathbf{c}_1, d)$ given and encryption $(\mathbf{c}_1, d_1)$ of $\mathsf{msg}$ under the public key $\mathbf{y}_0$! Hence, we can now immediately appeal to the fact that, from the view of the adversary, $\mathbf{y}_0$ looks indeed uniformly random to argue security: Via the LWE assumption, $(\mathbf{A}, \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_1 \bmod q)$ and $(\mathbf{y}_0, \mathbf{r}_1^{\mathsf{T}} \cdot \mathbf{y}_0 + e_1^* \bmod q)$ are indistinguishable from uniform. Thus, from the adversary's view $d_1$ looks uniformly random, and therefore $d = \mathbf{v}''^{\mathsf{T}} \cdot \mathbf{z} + d_1 \bmod q$ also looks uniformly random. In fact, from the adversary's view all ciphertext components look uniformly random and independent.

**LWE with Error-Leakage.** Drowning is, however, a rather heavy-handed approach that, for all intents and purposes, ruins the LWE parameters. Specifically, to use this approach we need to assume the security of LWE with *superpolynomial modulus-to-noise ratio*. This means, in turn, that the underlying worst-to-average case reduction of LWE [Reg05] reduces LWE to worst-case lattice problems with super-polynomial approximation factors. Moreover, it forces us to use a superpolynomially large modulus $q$.

We will now look a bit closer at the above drowning step. Specifically, given $\mathbf{z}$ and $\mathbf{v}^{\mathsf{T}} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e} \bmod q$ we computed

$$v = \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + e^* = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{A} \cdot \mathbf{z} + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^* = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_0 + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^* \bmod q.$$

Our main observation is the following: If we were somehow given an *advice* $\mathbf{l} = -\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^*$ about $\mathbf{e}$ and $e^*$, we could use $\mathbf{l}$ to *switch* the correlated error term $\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}$ in $\mathbf{v}^{\mathsf{T}} \cdot \mathbf{z}$ to a fresh and uncorrelated $e^*$. Namely by computing $v = \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + \mathbf{l} \bmod q$. Then it holds that

$$v = \mathbf{v}^{\mathsf{T}} \cdot \mathbf{z} + \mathbf{l} = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_0 + \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} - \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^* = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_0 + e^* \bmod q.$$

Thus, such an advice $\mathbf{l}$ is sufficient to make the security argument in the last paragraph work. Our hope now is that the advice $\mathbf{l} = -\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^*$ *does not fully reveal* $\mathbf{e}$ and $e^*$, i.e. that $\mathbf{e}$ and $e^*$ mutually conceal one another, even if the parameters of these error terms are way below the drowning regime.

This motivates the definition of *Learning with Errors with Error-Leakage*, elLWE for short. As the name suggests, in this variant of the LWE problem the adversary gets a *leak* or advice about the LWE error term. To make this definition useful for our purposes, we will allow the leak to depend on the LWE matrix $\mathbf{A}$. Consequently, we will define elLWE similarly to the regular LWE assumption, but via an *interactive experiment*. The security experiment of elLWE is given as follows, where we assume that a modulus $q$, dimensions $n, m$ and error distributions $\chi, \chi^*$ are parametrized by the security parameter.

The elLWE Security Experiment:

- In the first step, the experiment chooses a uniformly random matrix $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}$ and provides $\mathbf{A}$ to the adversary.
- Given the matrix $\mathbf{A}$, the adversary now chooses a *short* vector $\mathbf{z} \in \mathbb{Z}^m$ and provides $\mathbf{z}$ to the experiment.
- The experiment samples $\mathbf{e} \leftarrow\!\!\$\ \chi_1$ and $e^* \leftarrow\!\!\$\ \chi^*$ and sets $\mathbf{l} = \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z} + e^*$.

– Now the experiment flips a random bit $b \leftarrow\!\!\$ \{0,1\}$. If $b = 0$ it chooses a uniformly random $\mathbf{r} \leftarrow\!\!\$ \mathbb{Z}_q^n$ and sets $\mathbf{v}^\mathsf{T} = \mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}^\mathsf{T} \bmod q$. If $b = 1$ it chooses $\mathbf{v} \leftarrow\!\!\$ \mathbb{Z}_q^m$ uniformly at random.
– The experiment now provides $(\mathbf{A}, \mathbf{v}, \mathbf{l})$ to the adversary. The adversary then produces a guess $b' \in \{0,1\}$ for the bit $b$
– If $b' = b$ the adversary wins, and loses otherwise.

As usual, we say that elLWE is secure if no PPT adversary has non-negligible advantage in this experiment. Now, via the above discussion we can routinely reduce the security of our construction to elLWE.

We remark that the elLWE problem generalizes the *extended LWE problem* [OPW11,AP12]. Specifically, in the extended LWE problem the vector $\mathbf{z}$ is chosen at random from a Gaussian distribution instead of adversarially (as in the case of the elLWE problem). For the precise definitions concerning elLWE refer to Section 6.

**From LWE to elLWE.** As an additional technical contribution of this work, we provide a hardness result for elLWE. Specifically, we show that the security of elLWE can be based on *standard LWE with polynomial modulus-to-noise ratio.* In this paragraph, we will sketch the main ideas underlying this result. In a nutshell, the main idea of our approach is to choose the leakage term $\mathbf{l}$ independent of the LWE error, and then *adjust* the LWE error in such a way that it *conforms* with the leakage. More precisely in the case of Gaussian $\mathbf{e}$ and $e^*$, we will show the following. There is a (sufficiently wide) Gaussian distribution $\hat{\mathbf{e}}$, such that for every (short) vector $\mathbf{z}$ there is an *efficiently sampleable pair of correlated random variables* $(\mathbf{f}_\mathbf{z}, f_\mathbf{z})$ (independent of $\hat{\mathbf{e}}$), such that

$$(\mathbf{e}^\mathsf{T}, \mathbf{e}^\mathsf{T} \cdot \mathbf{z} + e^*) \approx_s (\hat{\mathbf{e}}^\mathsf{T} + \mathbf{f}_\mathbf{z}^\mathsf{T}, f_\mathbf{z}).$$

In other words, $f_\mathbf{z}$ simulates the leakage $\mathbf{e}^\mathsf{T} \cdot \mathbf{z} + e^*$, whereas $\mathbf{f}_\mathbf{z}$ can be used to *additively* adjust an independent Gaussian $\hat{\mathbf{e}}$ to have the same distribution as $\mathbf{e}$ given the leakage $\mathbf{e}^\mathsf{T} \cdot \mathbf{z} + e^*$. Equipped with such an efficiently sampleable pair $(\mathbf{f}_\mathbf{z}, f_\mathbf{z})$, reducing elLWE to LWE is almost straightforward: Given an LWE instance $(\mathbf{A}, \mathbf{v}^\mathsf{T})$ we run the elLWE adversary on $\mathbf{A}$, who returns $\mathbf{z}$. The reduction now samples $(\mathbf{f}_\mathbf{z}, f_\mathbf{z})$, provides $(\mathbf{A}, \mathbf{v}^\mathsf{T} + \mathbf{f}_\mathbf{z}^\mathsf{T}, f_\mathbf{z})$ to the adversary, and outputs whatever the adversary outputs.

On one side, if $\mathbf{v}^\mathsf{T}$ is an LWE sample, i.e. $\mathbf{v}^\mathsf{T} = \mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \hat{\mathbf{e}} \bmod q$, then

$$\begin{aligned}(\mathbf{A}, \mathbf{v}^\mathsf{T} + \mathbf{f}_\mathbf{z}^\mathsf{T}, f_\mathbf{z}) &= (\mathbf{A}, \mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \hat{\mathbf{e}} + \mathbf{f}_\mathbf{z}^\mathsf{T} \bmod q, f_\mathbf{z}) \\ &\approx_s (\mathbf{A}, \mathbf{r}^\mathsf{T} \cdot \mathbf{A} + \mathbf{e}^\mathsf{T} \bmod q, \mathbf{e}^\mathsf{T} \cdot \mathbf{z} + e^*),\end{aligned}$$

is statistically close to a correctly formed elLWE sample for $b = 0$.

On the other hand, if $\mathbf{v}$ is chosen uniformly random, then $\mathbf{v}' := \mathbf{v} + \mathbf{f}_\mathbf{z} \bmod q$ is also uniformly random. Consequently $(\mathbf{A}, \mathbf{v} + \mathbf{f}_\mathbf{z} \bmod q, f_\mathbf{z}) \approx_s (\mathbf{A}, \mathbf{v}', \mathbf{e}^\mathsf{T}\mathbf{z} + e^*)$, i.e. it is statistically close to an elLWE sample for $b = 1$. The claim follows. Notice that this reduction is *tight*, i.e. it does not (substantially) degrade the adversary's runtime or advantage. Further notice that this reduction is agnostic of the structure of the matrix $\mathbf{A}$ and the secret $\mathbf{r}$. Consequently, it is applicable to any *structured* LWE variant [BD20].

**Constructing the Leakage Simulator.** We will now briefly discuss how such a pair $(\mathbf{f}_\mathbf{z}, f_\mathbf{z})$ can be constructed. For simplicity, assume that $\mathbf{e}$ and $\mathbf{z}$ are scalars, i.e. $\mathbf{e} = e$ and $\mathbf{z} = z$. To further simplify matters, assume first that $e$ and $e^*$ are continuous Gaussians instead of *discrete* Gaussians. In this perspective, $(e, ez + e^*)$ is a pair of *correlated Gaussians*, i.e. a 2-dimensional Gaussian with (possibly) non-diagonal covariance matrix. If $e \sim D_\sigma$ and $e^* \sim D_{\sigma^*}$[7], then a routine calculation shows that the covariance matrix $\mathbf{C}$ of $(e, ez + e^*)$ is

$$\mathbf{C} = \begin{pmatrix} \sigma^2 & \sigma^2 z \\ \sigma^2 z & \sigma^2 z^2 + {\sigma^*}^2 \end{pmatrix}.$$

Our idea now is, basically speaking, to find an alternative way to represent this distribution. Specifically, we want to alternatively compute $(e, ez + e^*)$ via $(\hat{e} + we^\dagger, e^\dagger)$, where $\hat{e} \sim D_{\hat{\sigma}}$ and $e^\dagger \sim D_{\sigma^\dagger}$ are independent

[7]We denote the continuous Gaussian distribution with *parameter* $\sigma$ by $D_\sigma$, i.e. the probability density function of $D_\sigma$ is proportional to $e^{-\pi \frac{x^2}{\sigma^2}}$

Gaussians and $w$ is fixed (depending on $\sigma, \sigma^*$ and $z$). Again, a routine calculation finds that the covariance matrix $\mathbf{C}'$ of $(\hat{e} + we^\dagger, e^\dagger)$ is

$$\mathbf{C}' = \begin{pmatrix} \hat{\sigma}^2 + {\sigma^\dagger}^2 w^2 & {\sigma^\dagger}^2 w \\ {\sigma^\dagger}^2 w & {\sigma^\dagger}^2 \end{pmatrix}.$$

Now, two centered multivariate Gaussians are identically distributed, if and only if they have the same covariance matrix. Consequently, setting $\mathbf{C} = \mathbf{C}'$ and solving for $\hat{\sigma}^2, {\sigma^\dagger}^2$ and $w$ yields

$$\begin{aligned} {\sigma^\dagger}^2 &= \sigma^2 z^2 + {\sigma^*}^2, \\ w &= \frac{\sigma^2 z}{{\sigma^\dagger}^2} = \frac{\sigma^2 z}{\sigma^2 z^2 + {\sigma^*}^2}, \\ \hat{\sigma}^2 &= \sigma^2 - {\sigma^\dagger}^2 w^2 = \sigma^2 - \frac{\sigma^4 z^2}{\sigma^2 z^2 + {\sigma^*}^2} = \left(1 - \frac{1}{1 + \frac{{\sigma^*}^2}{\sigma^2 z^2}}\right)\sigma^2. \end{aligned} \tag{1}$$

That is, for these parameters of $\hat{\sigma}, \sigma^\dagger$ and $w$ it holds that $(e, ez + e^*) \equiv (\hat{e} + we^\dagger, e^\dagger)$, i.e. the two pairs are identically distributed. Thus, we can define $(\mathbf{f}_z, f_z)$ by $\mathbf{f}_z = we^\dagger$ and $f_z = e^\dagger$.

Now, recall that in our reduction $\hat{e}$ corresponds to the error-term in the underlying LWE-instance. Thus, we should choose $\sigma^*$ so as to ensure that $\hat{e} \sim D_{\hat{\sigma}}$ is a sufficiently wide Gaussian, while $\sigma^*$ should not be too large. A reasonable choice for $\sigma^*$ (which simplifies calculations) is to choose it such that $\sigma^* \geq \sigma \cdot \beta$, where $\beta$ is an upper bound for $|z|$ (recall that $z$ is adversarially chosen but short). For this choice of $\sigma^*$, it holds by (1) that $\hat{\sigma} \geq \sigma/\sqrt{2}$. In other words, for this parameter choice $\sigma^*$ is only a factor $\beta$ bigger than $\sigma$, whereas $\hat{\sigma}$ is only a factor $1/\sqrt{2}$ smaller than $\sigma$. In essence, this means that the reduction roughly preserves the LWE parameters, up to small factors.

The final piece of our reduction is to make this leakage simulator work for discrete Gaussians instead of continuous Gaussians. For this, we will make use of Peikert's randomized rounding approach [Pei10]. That is, a discrete Gaussian can be computed as the randomized rounding of a continuous Gaussian. This, together with Regev's discrete-to-continuous Gaussian smoothing lemma [Reg05], allows us to adapt the simulator for continous Gaussians to discrete Gaussians. While the simplified analysis above only uses simple arithmetic, the actual analysis in Section 5, while similar in spirit, relies on more involved concepts from singular value analysis to deal with high-dimensional multivariate Gaussians.

## 2.2 Applications

**Laconic OT.** As a warm-up application, it is easy to see that laconic encryption immediately implies laconic OT. Alice can construct a binary tree of keys with the following procedure: For each index pair $(2\mathsf{ind}, 2\mathsf{ind}-1)$, Alice inserts in the tree a uniformly sampled public key either in the even position if $D_{\mathsf{ind}} = 0$, or in the odd position if $D_{\mathsf{ind}} = 1$. Bob can then simply encrypt $\mathsf{msg}_0$ with respect to the index $2\mathsf{ind}$ and $\mathsf{msg}_1$ with respect to index $2\mathsf{ind} - 1$. Since Alice is semi-honest, the security of laconic encryption immediately carries over.

**Registration-Based Encryption.** Laconic Encryption almost implies RBE: Each user $\mathsf{ind}$ generates a key-pair and sends her $\mathsf{pk}_{\mathsf{ind}}$ for registration to an (untrusted) Key Curator, which is added to the database $D \leftarrow D \cup \{\mathsf{pk}_i\}$. Then the digest $d$ (the root of the tree) and the witnesses $\mathsf{wit}_j$ of all users are updated accordingly. Encryption and decryption with respect to $\mathsf{ind}$ work exactly as in laconic encryption. The crucial caveat is that in RBE, being highly dynamic, it's unrealistic to consider that the users are receiving an updated $\mathsf{wit}$ each time a new user registers. Therefore, there is an additional strict efficiency requirement: No user's witness should change more than $\log N$ times throughout the lifetime of the system ($N$ being the total number of users). This requirement minimizes the interaction between a user and the key curator.

Garg et.al [GHMR18] achieve this requirement by providing a direct construction based on Merkle trees. In a nutshell, to accumulate the public keys, there are multiple Merkle trees with an increasing number of leaves. A new public key enters a (degenerate) tree that consists of a single leaf. Then, as soon as the number of its leaves is the same with the next tree, the two trees are merged. This means that a tree (and therefore its corresponding paths-witnesses) is changing only when its leaves are doubled. Overall, this translates to $\log N$ number of trees and thus at most $\log N$ number of updates per user's witness. We generalize this idea and show a generic transformation from any laconic encryption scheme to a registration-based encryption scheme. A more detailed overview and a formal description can be found in Section 8.

**Laconic PSI.** We present a semi-honestly secure laconic PSI from laconic encryption. Here the receiver who owns a large database chooses a message for the sender to encrypt, and then it checks whether the ciphertext can be decrypted correctly with respect to the indices registered on the receiver's side.

We first need to have a hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^\ell$ to map elements into the universe of indices. For simplicity, we assume the sender's set is a singleton set $S_\mathsf{S} = \{y\}$. Besides sampling a hash function $\mathsf{H}$, the setup phase is the same as the laconic encryption. Then the receiver constructs a binary tree with the freshly generated public keys with respect to the indices where the elements in $S_\mathsf{R}$ are mapped. In the meantime, the receiver generates the witnesses. Then the receiver sends the updated $\mathsf{st}$ and a random message $\mathsf{msg}$. Next, the sender encrypts $\mathsf{msg}$ with $\mathsf{st}$ with respect to the index $\mathsf{H}(y)$, and sends the ciphertext $\mathsf{ctxt}$ to the receiver. Finally, upon receiving the ciphertext, the receiver will check for all $x_k \in S_\mathsf{R}$, whether it holds that $\mathsf{Dec}(\mathsf{sk}_k, \mathsf{wit}_k, \mathsf{ctxt}) = \mathsf{msg}$. If it finds such a $k$, $x_k$ will be output as the intersection of $S_\mathsf{S}$ and $S_\mathsf{R}$. The actual protocol will be obtained by running the above for every element in the senders set. Correctness and security of this protocol follows from the guarantees of the laconic encryption scheme. For more details, we refer the reader to Section 9.

**Identity-Based Encryption.** We also show that our laconic encryption scheme can be modified to construct an IBE. The basic idea is simple: Instead of constructing a tree of public keys iteratively, the key authority *implicitly* defines an exponentially large tree by sampling the root of the tree at random. The key difference is that now the authority must choose the matrices in the public parameters with a trapdoor. This way, when the user $\mathsf{ind}$ wants to register to the system, the authority can provide it with the appropriate root-to-leaf path (which will function as the secret key) by sampling pre-images, starting from the root and all the way down to the corresponding leaf.

Compared with other LWE-based constructions [CHKP10,ABB10,BL16,DGHM18,DG17a], our IBE supports an *unbounded* identity space, and has a tight security reduction of *full (adaptive)* security in the standard model. This is achieved with a new simulation strategy that relies on two alternating pairs of matrices $(\mathbf{B}_{0,\mathsf{even}}, \mathbf{B}_{1,\mathsf{even}})$ and $(\mathbf{B}_{0,\mathsf{odd}}, \mathbf{B}_{1,\mathsf{odd}})$, for left and right children and for even and odd layers, respectively. In the security proof, the simulator can "forget" the trapdoor of any one of the four matrices, and it can still issue decryption keys using the remaining trapdoors. This way, one can substitute ciphertext components one-by-one with uniformly sampled vectors. Proceeding until the last layer completes the security proof. A more detailed overview can be found in Section 11.

**Pre-Processing and Other Extensions.** To increase the efficiency of our laconic encryption even further, we also construct a pre-processing variant of our scheme. Informally, the encryption algorithm $\mathsf{Enc}$ is split into an offline part ($\mathsf{OfflineEnc}$), which is input-independent, and an online part ($\mathsf{OnlineEnc}$). Crucially, the online algorithm is much more efficient and does not perform any public-key operation. The main observation is that each element of the ciphertext $\mathbf{c}_i$ depends only on *a single bit* of the corresponding index/identity. Thus, we can let the $\mathsf{OfflineEnc}$ algorithm computing both possible ciphertexts for each bit of the index (making sure to use the randomness consistently), and output two commitments. The $\mathsf{OnlineEnc}$ algorithm is on the other hand given the index $\mathsf{ind}$, so it can complete the encryption by simply revealing the openings of the commitments corresponding to $(\mathsf{ind}_1, \ldots, \mathsf{ind}_\ell)$. As for the message, the $\mathsf{OfflineEnc}$ algorithm can simply encrypt a random bit $r$, and when the message $\mathsf{msg}$ is given to the $\mathsf{OnlineEnc}$ algorithm, it can simply output $\mathsf{msg} \oplus r$. This way, the $\mathsf{OnlineEnc}$ is entirely combinatorial, and all the public-key operation happen in an offline and input-independent phase. A formal description of this procedure can be found in Section 10.1.

We also explore a number of other extensions of laconic encryption: In Section 10.2 we describe how we can make the encryption algorithm compatible with other encryption schemes (possibly not even lattice-based), and in Appendix B we present an alternative laconic encryption construction that offers different efficiency trade-offs.

## 3 Preliminaries

Let $(n,p,q) = (n,p,q)(\lambda)$ with $p < q$. Let $m := n \cdot \lceil \log_p q \rceil$. Define the $(p,q)$-ary gadget matrix

$$\mathbf{G} := \mathbf{I}_n \otimes \left(1 \; p \; \ldots \; p^{\lfloor \log_p q \rfloor}\right)$$

and denote the (balanced) $p$-ary decomposition by $\mathbf{G}^{-1}(\cdot)$. For a bit $b \in \{0,1\}$, denote $\bar{b} := 1 - b$.

### 3.1 Lattices

Let $\mathcal{K} = \mathbb{Q}(\zeta)$ be a cyclotomic field and $\mathcal{R} = \mathbb{Z}[\zeta]$ its ring of integers, where $\zeta \in \mathbb{C}$ is a root of unity. Write $d_{\mathcal{R}}$ for the degree of (the cyclotomic polynomial defining $\mathcal{K}$ and) $\mathcal{R}$. The (infinity) norm $\|\cdot\|$ of an element $a = \sum_{i=0}^{d_{\mathcal{R}}-1} a_i \zeta^i \in \mathcal{R}$ is defined as the norm of its coefficient vector $(a_0, \ldots, a_{d_{\mathcal{R}}-1}) \in \mathbb{Z}^{d_{\mathcal{R}}}$, i.e. $\|a\| = \max_{i=0}^{d_{\mathcal{R}}-1} |a_i|$. For a vector $\mathbf{x} = (x_0, \ldots, x_{m-1}) \in \mathcal{R}^m$, its norm is defined as $\|\mathbf{x}\| \coloneqq \max_{i=0}^{m-1} \|x_i\|$. For $q \in \mathbb{N}$, write $\mathcal{R}_q \coloneqq \mathcal{R}/q\mathcal{R}$. Let $\chi$ be a distribution over $\mathcal{R}$.

**Definition 1** **($\mathsf{LWE}_{\mathcal{R},n,q,\chi}$ Assumption).** *Let $\mathcal{R}, n, m, q, \chi$ be parametrised by $\lambda$. The (decision) $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi}$ assumption states that for any PPT adversary $\mathcal{A}$*

$$\left| \Pr\left[ \mathcal{A}(\mathbf{A},\mathbf{b}) = 1 \;\middle|\; \begin{array}{l} \mathbf{A} \leftarrow\$\, \mathcal{R}_q^{n \times m} \\ \mathbf{s} \leftarrow\$\, \mathcal{R}_q^{n} \\ \mathbf{e} \leftarrow\$\, \chi^m \\ \mathbf{b}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q \end{array} \right] - \Pr\left[ \mathcal{A}(\mathbf{A},\mathbf{b}) = 1 \;\middle|\; \begin{array}{l} \mathbf{A} \leftarrow\$\, \mathcal{R}_q^{n \times m} \\ \mathbf{b} \leftarrow\$\, \mathcal{R}_q^{m} \end{array} \right] \right|$$
$$\leq \mathsf{negl}(\lambda).$$

*The $\mathsf{LWE}_{\mathcal{R},n,q,\chi}$ assumption is said to hold if the $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi}$ assumption holds for all $m = \mathsf{poly}(\lambda)$.*

**Definition 2** **(Discrete Gaussian Distributions).** *Let $m \in \mathbb{N}$ and $s > 0$. The discrete Gaussian function over $\mathbb{R}$ with parameter $s$ is defined as $\rho_s(x) \coloneqq \exp\left(-\pi \frac{|x|^2}{s^2}\right)$ with support $\mathbb{R}$. The discrete Gaussian distribution over $\mathbb{Z}$ with parameter $s$ is defined as $\mathcal{D}_{\mathbb{Z},s}(x) \coloneqq \frac{\rho_s(x)}{\sum_{x' \in \mathbb{Z}} \rho_s(x')}$ with support $\mathbb{Z}$. The discrete Gaussian distribution over $\mathcal{R}$ with parameter $s$, denoted by $\mathcal{D}_{\mathcal{R},s}$ is induced by sampling $d_{\mathcal{R}}$ independent samples $x_i \leftarrow\$\, \mathcal{D}_{\mathbb{Z},s}$ and outputing $x = \sum_{i=0}^{d_{\mathcal{R}}-1} x_i \cdot \zeta^i$.*

We recall a version of the leftover hash lemma over cyclotomic rings.

**Lemma 1** **(Adapted from [BJRW20, Lemma 7]).** *Let $n = \mathsf{poly}(\lambda)$, $p, q \in \mathbb{N}$, and $m \geq n \cdot \log_p q + \omega(\log \lambda)$. The following distributions are statistically close in $\lambda$:*

$$\left\{ (\mathbf{B},\mathbf{y}) : \begin{array}{l} \mathbf{B} \leftarrow\$\, \mathcal{R}_q^{n \times m} \\ \mathbf{x} \leftarrow\$\, \mathcal{R}_p^{m} \\ \mathbf{y} \coloneqq \mathbf{B} \cdot \mathbf{x} \bmod q \end{array} \right\} \qquad \text{and} \qquad \left\{ (\mathbf{B},\mathbf{y}) : \begin{array}{l} \mathbf{B} \leftarrow\$\, \mathcal{R}_q^{n \times m} \\ \mathbf{y} \leftarrow\$\, \mathcal{R}_p^{n} \end{array} \right\}.$$

**Lemma 2** **(Derived from [MP12, Section 2.4]).** *For any $k > 0$,*

$$\Pr[\|u\| > k \cdot s \mid u \leftarrow\$\, \mathcal{D}_{\mathcal{R},s}] < 2 \cdot d_{\mathcal{R}} \cdot \exp(-\pi \cdot k^2).$$

**Definition 3** **(Ring Expansion Factor).** *The expansion factor of $\mathcal{R}$, denoted by $\gamma_{\mathcal{R}}$, is $\gamma_{\mathcal{R}} \coloneqq \max_{a,b \in \mathcal{R} \setminus \{0\}} \frac{\|a \cdot b\|}{\|a\| \cdot \|b\|}$.*

**Proposition 1** **([AL21]).** *If $\mathcal{R}$ is a prime-power cyclotomic ring, then $\gamma_{\mathcal{R}} \leq 2 \deg_{\mathcal{R}}$. If $\mathcal{R}$ is a power-of-2 cyclotomic ring, then $\gamma_{\mathcal{R}} \leq \deg_{\mathcal{R}}$.*

## 4 Preliminaries on Spectral Analysis

**Linear Algebra** We say a symmetric matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is *positive definite*, if $\mathbf{x}^{\mathsf{T}}\mathbf{S}\mathbf{x} > 0$ for all non-zero $\mathbf{x} \in \mathbb{R}^n$. Analogously, we say that $\mathbf{\Sigma}$ is positive semi-definite, if $\mathbf{x}^{\mathsf{T}}\mathbf{S}\mathbf{x} \geq 0$ for all non-zero $\mathbf{x} \in \mathbb{R}^n$. We can use positive semi-definiteness to define a partial ordering on symmetric matrices: We define $\mathbf{\Sigma}_1 \geq \mathbf{\Sigma}_2$ iff $\mathbf{\Sigma}_1 - \mathbf{\Sigma}_2$ is positive semi-definite, i.e. if for all $\mathbf{x} \in \mathbb{R}^n$ we have that $\mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}_1\mathbf{x} \geq \mathbf{x}^{\mathsf{T}}\mathbf{\Sigma}_2\mathbf{x}$.

For every matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ we can find a *singular value decomposition* $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$, where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, i.e. it holds that $\mathbf{U}\mathbf{U}^{\mathsf{T}} = \mathbf{I}$ and $\mathbf{V}\mathbf{V}^{\mathsf{T}} = \mathbf{I}$, and $\mathbf{D} \in \mathbb{R}^{m \times n}$ is an upper diagonal matrix. The entries on the diagonal of $\mathbf{D}$ are called the singular values of $\mathbf{M}$. We denote by $\sigma_{\mathsf{max}}(\mathbf{M})$ the largest singular value of $\mathbf{M}$, and by $\sigma_{\mathsf{min}}(\mathbf{M})$ the smallest singular value of $\mathbf{M}$.

Every positive semi-definite matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ has a singular-value decomposition of the $\mathbf{\Sigma} = \mathbf{U}\mathbf{D}^2\mathbf{U}^{\mathsf{T}}$, where $\mathbf{U} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix. Via the singular value decomposition we can compute a square root $\sqrt{\mathbf{\Sigma}}$ of a positive semi-definite matrix $\mathbf{\Sigma}$ by $\sqrt{\mathbf{\Sigma}} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathsf{T}}$.

It holds for every positive semi-definite definite matrix $\mathbf{\Sigma}$ that $\sigma_{\mathsf{min}}(\mathbf{\Sigma}) \cdot \mathbf{I} \leq \mathbf{\Sigma} \leq \sigma_{\mathsf{max}}(\mathbf{\Sigma}) \cdot \mathbf{I}$. Furthermore, $\sigma_{\mathsf{min}}(\mathbf{\Sigma})$ is the largest value and $\sigma_{\mathsf{max}}(\mathbf{\Sigma})$ the smallest scalar value for which this holds.

**Lattices and Gaussians** We recall the standard facts about lattices. A lattice $\mathbf{\Lambda} \subseteq \mathbb{R}^m$ is the set of all integer-linear combinations of a set of linearly independent basis-vectors, i.e. for every lattice $\mathbf{\Lambda}$ there exists a full-rank matrix $\mathbf{B} \in \mathbb{R}^{k \times m}$ such that $\mathbf{\Lambda} = \mathbf{\Lambda}(\mathbf{B}) = \{\mathbf{z} \cdot \mathbf{B} \mid \mathbf{z} \in \mathbb{Z}^k\}$. We call $k$ the rank of $\mathbf{\Lambda}$ and $\mathbf{B}$ a basis of $\mathbf{\Lambda}$, and we say that $\mathbf{\Lambda}$ is full-rank if $k = m$. The Gaussian function $\rho_\sigma : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\rho_\sigma(\mathbf{x}) = e^{-\pi \cdot \frac{\|\mathbf{x}\|^2}{\sigma^2}}.$$

For a a non-singular matrix $\mathbf{B}$ we define $\rho_{\mathbf{B}}(\mathbf{x}) = \rho(\mathbf{x}\mathbf{B}^{-1})$.

The continuous gaussian distribution $D_{\mathbf{B}}$ on $\mathbb{R}^n$ has the probability density function $\rho_{\mathbf{B}}(\mathbf{x})/\rho_{\mathbf{B}}(\mathbb{R}^n)$. We call $\mathbf{\Sigma} = \mathbf{B}^{\mathsf{T}}\mathbf{B}$ the covariance matrix of the gaussian $D_{\mathbf{B}}$. For a lattice $\mathbf{\Lambda}$, the discrete gaussian distribution $D_{\mathbf{\Lambda},\mathbf{B}}$ supported on $\mathbf{\Lambda}$ has the probability mass function $\rho_{\mathbf{B}}(\mathbf{x})/\rho_{\mathbf{B}}(\mathbf{\Lambda})$.

For a lattice $\mathbf{\Lambda}$ and a positive real $\epsilon > 0$, the *smoothing parameter* $\eta_\epsilon(\mathbf{\Lambda})$ is defined to be the smallest real number $s$ for which $\rho_{1/s}(\mathbf{\Lambda}^* \backslash \{0\}) \leq \epsilon$. For a matrix $\mathbf{B}$ we write $\mathbf{B} \geq \eta_\epsilon(\mathbf{\Lambda})$ if $\eta_\epsilon(\mathbf{\Lambda}\mathbf{B}^{-1}) \leq 1$.

We will make use of the following properties of gaussian distributions with respect to lattices.

**Lemma 3 ([Reg05, Claim 3.9], simplified).** *Let $\mathbf{\Lambda} \subseteq \mathbb{R}^n$ be a lattice and let $\sigma_1, \sigma_2 > 0$ be such that*

$$\frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} \geq \eta_\epsilon(\mathbf{\Lambda}).$$

*Let $\mathbf{e} \sim D_{\mathbf{\Lambda},\sigma_1}$ be a discrete gaussian and $\mathbf{e}' \sim D_{\sigma_2 \cdot \mathbf{I}}$ be a continuous gaussian. Then $\mathbf{e} + \mathbf{e}'$ is $4\epsilon$ close to the continuous gaussian $D_{\sqrt{\sigma_1^2 + \sigma_2^2} \cdot \mathbf{I}}$.*

We will use the following simple corollary of Lemma 3.

**Lemma 4.** *Let $\mathbf{\Lambda} \subseteq \mathbb{R}^n$ be a lattice, let $\epsilon > 0$ and let $\mathbf{\Sigma}_0 \geq \eta_\epsilon(\mathbf{\Lambda})$. Let $\mathbf{\Sigma}_1 = \sigma_1^2\mathbf{\Sigma}_0$ and let $\mathbf{\Sigma}_2$ be such that $\mathbf{\Sigma}_2 \geq \sigma_2^2\mathbf{\Sigma}_0$. Assume that $\frac{s}{\sigma^2} + \frac{1}{\sigma_2^2} \leq 1$. Let $\mathbf{e} \sim D_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}_1}}$, $\tilde{\mathbf{e}} \sim D_{\sqrt{\mathbf{\Sigma}_2}}$ and $\mathbf{e}^* \sim D_{\sqrt{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}}$. Then it holds that*

$$\mathbf{e} + \tilde{\mathbf{e}} \approx_{4\epsilon} \mathbf{e}^*.$$

**Theorem 1 ([Pei10, Thm 3.1], second part).** *Let $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2 > 0$ be two positive definite matrices such that $\mathbf{\Sigma} = \mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 > \mathbf{0}$ and $\mathbf{\Sigma}_3^{-1} = \mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1} > \mathbf{0}$. Let $\mathbf{\Lambda}$ be a lattice with $\sqrt{\mathbf{\Sigma}_1} \geq \eta_\epsilon(\mathbf{\Lambda})$ for some $\epsilon > 0$. Let $\mathbf{c} \in \mathbb{R}^n$ be arbitrary. Consider the following sampling procedure for $\mathbf{x} \in \mathbf{\Lambda}$:*

- *Choose $\mathbf{x}' \leftarrow\!\$\; D_{\sqrt{\mathbf{\Sigma}_2}}$.*
- *Choose $\mathbf{x} \leftarrow\!\$\; \mathbf{x}_1 + D_{\mathbf{\Lambda}+\mathbf{c}-\mathbf{x}',\sqrt{\mathbf{\Sigma}_1}}$.*

*Then it holds that the marginal distribution of* $\mathbf{x}$ *is within statistical distance* $8\epsilon$ *to* $D_{\mathbf{\Lambda}+\mathbf{c},\sqrt{\mathbf{\Sigma}}}$. *Furthermore, the conditional distribution of* $\mathbf{x}'$ *given* $\mathbf{x} = \mathbf{z}$ *is within statistical distance* $2\epsilon$ *of* $\mathbf{\Sigma}_3\mathbf{\Sigma}_1^{-1}\mathbf{z} + D_{\sqrt{\mathbf{\Sigma}_3}}$.

Theorem 1 motivates the definition of a convenient *gaussian rounding function.*

**Definition 4.** *Let* $\mathbf{\Lambda}$ *be a lattice (or more generally a coset of a lattice) and* $\mathbf{\Sigma}$ *be a positive definite matrix. Then* $\mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}}}(\mathbf{x})$ *is a random variable following the distribution* $\mathbf{x} + D_{\mathbf{\Lambda}-\mathbf{x},\sqrt{\mathbf{\Sigma}}}$.

Note that it follows instantly from the definition of $\mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}}}$ that it holds for any $\mathbf{x} \in \mathbb{R}^n$ and any $\mathbf{z} \in \mathbf{\Lambda}$ that $\mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}}}(\mathbf{x}+\mathbf{z}) \equiv \mathbf{z} + \mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}}}(\mathbf{x})$, as $\Lambda - (\mathbf{x}+\mathbf{z}) = \Lambda - \mathbf{x}$.

Using this notion, we will make use of Theorem 1 in the form of the following corollary.

**Lemma 5.** *Let* $\mathbf{\Sigma}_0$ *be a positive definite matrix and let* $s > 0$. *Let* $\mathbf{\Lambda} \subseteq \mathbb{R}^n$ *be a lattice with* $\sqrt{\mathbf{\Sigma}_0} \geq \eta_\epsilon(\mathbf{\Lambda})$ *for some* $\epsilon > 0$. *Let* $\mathbf{e}' \sim D_{s\cdot\sqrt{\mathbf{\Sigma}_0}}$, $\mathbf{e} \sim D_{\mathbf{\Lambda},\sqrt{(1+s^2)\mathbf{\Sigma}_0}}$ *and* $\tilde{\mathbf{e}} \sim D_{\sqrt{(s^2+1)/s^2\cdot\mathbf{\Sigma}_0}}$. *Let* $\sigma^* = \frac{s^2}{s^2+1}$. *Then it holds that*

$$\mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}') \approx_{8\epsilon} \mathbf{e}$$

*and furthermore*

$$(\mathbf{e}', \mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}')) \approx_{10\epsilon} (\sigma^*(\mathbf{e}+\tilde{\mathbf{e}}), \mathbf{e}).$$

*Proof.* The first item follows from Theorem 1 by setting $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_2 = s^2 \cdot \mathbf{\Sigma}_0$. For this parameter choice it holds that $\mathbf{\Sigma}_3 = \frac{s^2}{s^2+1}\mathbf{\Sigma}_0$ and hence $\mathbf{\Sigma}_3\mathbf{\Sigma}_1^{-1}\mathbf{z} = \sigma^*\mathbf{z}$ for all $\mathbf{z} \in \mathbb{R}^n$. We establish the second item as follows. Letting $\mathbf{r}_{\mathbf{e}'} = \mathsf{round}_{\mathbf{\Lambda},\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}')$, it holds by the triangle inequality that

$$\begin{aligned}
\Delta((\mathbf{e}',\mathbf{r}_{\mathbf{e}'}),(\sigma^*(\mathbf{e}+\tilde{\mathbf{e}}),\mathbf{e})) &\leq \Delta((\mathbf{e}',\mathbf{r}_{\mathbf{e}'}),(\sigma^*\mathbf{r}_{\mathbf{e}'} + \sigma^*\tilde{\mathbf{e}},\mathbf{r}_{\mathbf{e}'}))\\
&\quad + \Delta((\sigma^*\mathbf{r}_{\mathbf{e}'} + \sigma^*\tilde{\mathbf{e}},\mathbf{r}_{\mathbf{e}'}),(\sigma^*\mathbf{e} + \sigma^*\tilde{\mathbf{e}},\mathbf{e}))\\
&= \sum_{\mathbf{z}\in\mathbf{\Lambda}} \Pr[\mathbf{r}_{\mathbf{e}} = \mathbf{z}]\,\underbrace{\Delta(\mathbf{e}|\mathbf{r}_{\mathbf{e}} = \mathbf{z}, \sigma^*\mathbf{z} + \sigma^*\tilde{\mathbf{e}})}_{\leq 2\epsilon \text{ by Theorem 1}} + \Delta(\mathbf{r}_{\mathbf{e}},\mathbf{d})\\
&\leq \underbrace{2\epsilon\,(\sum_{\mathbf{z}\in\mathbf{\Lambda}} \Pr[\mathbf{r}_{\mathbf{e}} = \mathbf{z}])}_{=1} + 8\epsilon\\
&= 10\epsilon.
\end{aligned}$$

□

## 4.1 Spectral Bounds

We will briefly establish a few convenient tools to bound the spectra of matrices.

**Lemma 6.** *Let* $\mathbf{M}, \mathbf{S}, \mathbf{\Sigma}$ *be positive definite matrices in* $\mathbb{R}^{n\times n}$. *Then it holds that*

$$\frac{1}{\sigma_{\mathsf{max}}(\mathbf{M})}\mathbf{I} \leq \mathbf{M}^{-1} \leq \frac{1}{\sigma_{\mathsf{min}}(\mathbf{M})}\mathbf{I}$$

*and*

$$(\mathbf{S}+\mathbf{\Sigma})^{-1} \leq \frac{1}{1+\frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma})}{\sigma_{\mathsf{max}}(\mathbf{S})}}\mathbf{S}^{-1}$$

*Proof.* Let $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{U}^\mathsf{T}$ be the singular value decomposition of $\mathbf{M}$. Then $\mathbf{M}^{-1} = \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^\mathsf{T}$. Note that $\mathbf{D}^{-1} \leq \frac{1}{\sigma_{\mathsf{min}}(\mathbf{M})}\mathbf{I}$. Hence it holds for all $\mathbf{x} \in \mathbb{R}^n$ that

$$\mathbf{x}^\mathsf{T}\mathbf{M}^{-1}\mathbf{x} = (\mathbf{U}^\mathsf{T}\mathbf{x})^\mathsf{T}\mathbf{D}^{-1}(\mathbf{U}^\mathsf{T}\mathbf{x}) \leq \frac{1}{\sigma_{\mathsf{min}}(\mathbf{M})}(\mathbf{U}^\mathsf{T}\mathbf{x})^\mathsf{T}(\mathbf{U}^\mathsf{T}\mathbf{x}) = \mathbf{x}^\mathsf{T}\left(\frac{1}{\sigma_{\mathsf{min}}(\mathbf{M})}\mathbf{I}\right)\mathbf{x},$$

from which the first item follows. To establish the second item, let $\mathbf{Y} = \sqrt{\mathbf{S}}$. Note that $\mathbf{Y}$ is also positive definite. It holds for all $\mathbf{x} \in \mathbb{R}^n$ that

$$\begin{aligned}\mathbf{x}^{\mathsf{T}}(\mathbf{S}+\mathbf{\Sigma})^{-1}\mathbf{x} &= \mathbf{x}^{\mathsf{T}}\mathbf{Y}^{-1}(\mathbf{Y}^{-1}(\mathbf{S}+\mathbf{\Sigma})\mathbf{Y}^{-1})^{-1}\mathbf{Y}^{-1}\mathbf{x} \\ &= (\mathbf{Y}^{-1}\mathbf{x})^{\mathsf{T}}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})(\mathbf{Y}\mathbf{x}) \\ &\le \frac{1}{\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})}(\mathbf{Y}^{-1}\mathbf{x})^{\mathsf{T}}(\mathbf{Y}^{-1}\mathbf{x}) \\ &= \frac{1}{\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})}\mathbf{x}^{\mathsf{T}}\mathbf{Y}^{-2}\mathbf{x} \\ &= \mathbf{x}^{\mathsf{T}}\left(\frac{1}{\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})}\mathbf{S}^{-1}\right)\mathbf{x},\end{aligned}$$

i.e. $(\mathbf{S}+\mathbf{\Sigma})^{-1} \le \frac{1}{\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})}\mathbf{S}^{-1}$. It remains to lower-bound $\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1})$. It holds for all $\mathbf{x} \in \mathbb{R}^n$ that

$$\begin{aligned}\mathbf{x}^{\mathsf{T}}\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1}\mathbf{x} &= (\mathbf{Y}^{-1}\mathbf{x})^{\mathsf{T}}\mathbf{\Sigma}(\mathbf{Y}^{-1}\mathbf{x}) \\ &\ge \sigma_{\min}(\mathbf{\Sigma})(\mathbf{Y}^{-1}\mathbf{x})^{\mathsf{T}}(\mathbf{Y}^{-1}\mathbf{x}) \\ &= \sigma_{\min}(\mathbf{\Sigma})\mathbf{x}^{\mathsf{T}}\mathbf{Y}^{-2}\mathbf{x} \\ &= \sigma_{\min}(\mathbf{\Sigma})\mathbf{x}^{\mathsf{T}}\mathbf{S}^{-1}\mathbf{x} \\ &\ge \frac{\sigma_{\min}(\mathbf{\Sigma})}{\sigma_{\max}(\mathbf{S})}\mathbf{x}^{\mathsf{T}}\mathbf{x},\end{aligned}$$

i.e. $\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1} \ge \frac{\sigma_{\min}(\mathbf{\Sigma})}{\sigma_{\max}(\mathbf{S})}\mathbf{I}$ and hence

$$\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1} \ge \left(1+\frac{\sigma_{\min}(\mathbf{\Sigma})}{\sigma_{\max}(\mathbf{S})}\right)\mathbf{I}.$$

This implies $\sigma_{\min}(\mathbf{I}+\mathbf{Y}^{-1}\mathbf{\Sigma}\mathbf{Y}^{-1}) \ge 1+\frac{\sigma_{\min}(\mathbf{\Sigma})}{\sigma_{\max}(\mathbf{S})}$. We can conclude that

$$(\mathbf{S}+\mathbf{\Sigma})^{-1} \le \frac{1}{1+\frac{\sigma_{\min}(\mathbf{\Sigma})}{\sigma_{\max}(\mathbf{S})}}\mathbf{S}^{-1}.$$

□

**Lemma 7.** *Let $m \ge n$ be integers and let $\mathbf{M} \in \mathbb{R}^{m\times n}$ be a full rank matrix. Let $\mathbf{T} = \mathbf{M}^{\mathsf{T}}(\mathbf{M}\mathbf{M}^{\mathsf{T}})^{-1}\mathbf{M}$. Then all non-zero singular values of $\mathbf{T}$ are 1.*

*Proof.* Let $\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathsf{T}}$ be the singular value decomposition of $\mathbf{M}$, where $\mathbf{S} = \begin{pmatrix}\mathbf{D} & \mathbf{0}\end{pmatrix}$. Then it holds that

$$\begin{aligned}\mathbf{T} &= \mathbf{V}\mathbf{S}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}}(\mathbf{U}\mathbf{S}\mathbf{V}^{\mathsf{T}}\mathbf{V}\mathbf{S}^{\mathsf{T}}\mathbf{U}^{\mathsf{T}})^{-1}\mathbf{U}\mathbf{S}\mathbf{V}^{\mathsf{T}} \\ &= \mathbf{V}\mathbf{S}^{\mathsf{T}}\mathbf{D}^{-2}\mathbf{S}\mathbf{V}^{\mathsf{T}},\end{aligned}$$

as $\mathbf{S}\mathbf{S}^{\mathsf{T}} = \mathbf{D}^2$. Finally, note that

$$\begin{aligned}\mathbf{S}^{\mathsf{T}}\mathbf{D}^{-2}\mathbf{S} &= \begin{pmatrix}\mathbf{D} \\ \mathbf{0}\end{pmatrix}\mathbf{D}^{-2}\begin{pmatrix}\mathbf{D} & \mathbf{0}\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{D}\mathbf{D}^{-2}\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{pmatrix} \\ &= \begin{pmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{pmatrix},\end{aligned}$$

i.e.

$$\mathbf{T} = \mathbf{V}\begin{pmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{pmatrix}\mathbf{V}^{\mathsf{T}},$$

and the claim follows. □

## 5 Simulating Gaussian Leakage

### 5.1 Continuous Gaussian Leakage

We will need the following Lemma about continuous gaussians conditionals.

**Lemma 8 (Gaussian conditionals).** *Let $n, k$ be integers. Let $\mathbf{\Sigma}_1 \in \mathbb{R}^{n \times n}$ and $\mathbf{\Sigma}_2 \in \mathbb{R}^{k \times k}$ be positive definite matrices. Fix a matrix $\mathbf{Z} \in \mathbb{R}^{k \times n}$ and let $\mathbf{\Sigma}_2' = \mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2$, $\mathbf{W} = \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}{\mathbf{\Sigma}_2'}^{-1}$ and $\mathbf{\Sigma}_1' = \mathbf{\Sigma}_1 - \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}(\mathbf{\Sigma}_2')^{-1}\mathbf{Z}\mathbf{\Sigma}_1$. Then the the following holds: If $\mathbf{e}_1 \sim D_{\sqrt{\mathbf{\Sigma}_1}}$, $\mathbf{e}_2 \sim D_{\sqrt{\mathbf{\Sigma}_2}}$, $\mathbf{e}_1' \sim D_{\sqrt{\mathbf{\Sigma}_1'}}$, $\mathbf{e}_2' \sim D_{\sqrt{\mathbf{\Sigma}_2'}}$, then*

$$(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2) \equiv (\mathbf{e}_1' + \mathbf{W}\mathbf{e}_2', \mathbf{e}_2'),$$

*i.e. $(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2)$ and $(\mathbf{e}_1' + \mathbf{W}\mathbf{e}_2', \mathbf{e}_2')$ are identically distributed. Moreover, it holds that*

$$\begin{aligned} \left(1 - \frac{1}{1 + \frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)}{\sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2}}\right)\mathbf{\Sigma}_1 \leq& \mathbf{\Sigma}_1' \leq \mathbf{\Sigma}_1 \\ \mathbf{\Sigma}_2 \leq& \mathbf{\Sigma}_2' \leq \mathbf{\Sigma}_2 + \sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2\mathbf{I} \end{aligned}$$

Specifically, Lemma 8 provides us with an alternative way to compute the pair $(\mathbf{e}_1, \mathbf{e}_2' = \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2)$: Instead of first sampling $\mathbf{e}_1$ and $\mathbf{e}_2$ and then computing $\mathbf{e}_2' = \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2$, we can first sample $\mathbf{e}_2'$ from a suitable gaussian and then sample a matching $\mathbf{e}_1 = \mathbf{e}_1' + \mathbf{W}\mathbf{e}_2'$ depending on $\mathbf{e}_2'$. An alternative interpretation of the lemma is that given $\mathbf{e}_2' = \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2$, the conditional distribution of $\mathbf{e}_1$ given $\mathbf{e}_2'$ follows $\mathbf{e}_1' + \mathbf{W}\mathbf{e}_2'$.

*Proof.* In the following, we will assume that the matrix $\mathbf{Z}$ has full rank. This will only affect the lower bound on $\mathbf{\Sigma}_1'$. If $\mathbf{Z}$ is rank-deficient, the singular value decomposition of $\mathbf{Z}$ provides us with $\mathbf{Z} = \mathbf{U}\begin{pmatrix}\mathbf{Z}' \\ \mathbf{0}\end{pmatrix}$ for an orthogonal matrix $\mathbf{U}$ and a matrix $\mathbf{Z}' \in \mathbb{R}^{r \times n}$ (where $r$ is the rank of $\mathbf{Z}$). Since rotation with $\mathbf{U}$ does not change the spectrum of $\mathbf{\Sigma}_2$ (i.e. the shape of the gaussian $D_{\sqrt{\mathbf{\Sigma}_2}}$), we may assume for simplicity that $\mathbf{Z} = \begin{pmatrix}\mathbf{Z}' \\ \mathbf{0}\end{pmatrix}$. We can decompose $\mathbf{\Sigma}_2$ as $\mathbf{\Sigma}_2 = \bar{\mathbf{\Sigma}}_2 + (\mathbf{\Sigma}_2 - \bar{\mathbf{\Sigma}}_2)$, where $\bar{\mathbf{\Sigma}}_2 = \sigma_{\mathsf{min}}(\mathbf{\Sigma}_2) \cdot \mathbf{I}$. Note that both $\bar{\mathbf{\Sigma}}_2$ and $\mathbf{\Sigma}_2 - \bar{\mathbf{\Sigma}}_2$ are positive semi-definite and that $\sigma_{\mathsf{min}}(\bar{\mathbf{\Sigma}}_2) = \sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)$. Hence we can decompose $\mathbf{e}_2$ as $\mathbf{e}_2 = \mathbf{e}_2^{(1)} + \mathbf{e}_2^{(2)}$, where $\mathbf{e}_2^{(1)} \sim D_{\sqrt{\bar{\mathbf{\Sigma}}_2}}$ and $\mathbf{e}_2^{(2)} \sim D_{\sqrt{\mathbf{\Sigma}_2 - \bar{\mathbf{\Sigma}}_2}}$. That is $\mathbf{e}_2^{(1)}$ is now a spherical gaussian, for which all components are independent. We can then prove the statement of the Lemma for the full rank matrix $\mathbf{Z}'$ instead of $\mathbf{Z}$ and $\mathbf{e}_2^{(1)}$ instead of $\mathbf{e}_2$. This concludes the detour about rank-deficient $\mathbf{Z}$.

We will now prove that for

$$\begin{aligned} \mathbf{W} &= \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}(\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2)^{-1} \\ \mathbf{\Sigma}_1' &= \mathbf{\Sigma}_1 - \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}(\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2)^{-1}\mathbf{Z}\mathbf{\Sigma}_1 \\ \mathbf{\Sigma}_2' &= \mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2 \end{aligned}$$

the claim of the lemma holds. Assume for now that $\mathbf{\Sigma}_1'$ and $\mathbf{\Sigma}_2'$ are positive definite. We will show this claim later when bounding the spectra of $\mathbf{\Sigma}_1'$ and $\mathbf{\Sigma}_2'$. It will be convenient to express $\mathbf{W}$ in terms of $\mathbf{\Sigma}_2'$ and $\mathbf{\Sigma}_1'$ in terms of $\mathbf{W}$. By inspection, it holds that

$$\mathbf{W} = \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}{\mathbf{\Sigma}_2'}^{-1} \tag{2}$$

$$\mathbf{\Sigma}_1' = \mathbf{\Sigma}_1 - \mathbf{W}\mathbf{\Sigma}_2'\mathbf{W}^{\mathsf{T}}. \tag{3}$$

Now let

$$\mathbf{e} = \begin{pmatrix}\mathbf{e}_1 \\ \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2\end{pmatrix} \text{ and } \mathbf{e}' = \begin{pmatrix}\mathbf{e}_1' + \mathbf{W}\mathbf{e}_2' \\ \mathbf{e}_2'\end{pmatrix}$$

Both $\mathbf{e}$ and $\mathbf{e}'$ are multivariate gaussians and it follows routinely that both $\mathbf{e}$ and $\mathbf{e}'$ have expectation 0. Thus it suffices to show that $\mathbf{e}$ and $\mathbf{e}'$ have the same covariance matrix in order to show that they follow the same distribution.

It holds that

$$\begin{aligned}
\mathbf{\Sigma} &= \mathsf{E}[\mathbf{e}\mathbf{e}^{\mathsf{T}}] \\
&= \mathsf{E}\begin{bmatrix} \mathbf{e}_1\mathbf{e}_1^{\mathsf{T}} & \mathbf{e}_1(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)^{\mathsf{T}} \\ (\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)\mathbf{e}_1^{\mathsf{T}} & (\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)^{\mathsf{T}} \end{bmatrix} \\
&= \begin{pmatrix} \mathsf{E}[\mathbf{e}_1\mathbf{e}_1^{\mathsf{T}}] & \mathsf{E}[\mathbf{e}_1(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)^{\mathsf{T}}] \\ \mathsf{E}[(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)\mathbf{e}_1^{\mathsf{T}}] & \mathsf{E}[(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)(\mathbf{Z}\mathbf{e}_1+\mathbf{e}_2)^{\mathsf{T}}] \end{pmatrix} \\
&= \begin{pmatrix} \mathsf{E}[\mathbf{e}_1\mathbf{e}_1^{\mathsf{T}}] & \mathsf{E}[\mathbf{e}_1\mathbf{e}_1^{\mathsf{T}}]\mathbf{Z}^{\mathsf{T}} + \mathsf{E}[\mathbf{e}_1\mathbf{e}_2^{\mathsf{T}}] \\ \mathbf{Z}\mathsf{E}[(\mathbf{e}_1\mathbf{e}_1^{\mathsf{T}}] + \mathsf{E}[\mathbf{e}_2\mathbf{e}_1^{\mathsf{T}}] & \mathbf{Z}\mathsf{E}[\mathbf{e}_1\mathbf{e}_1^{\mathsf{T}}]\mathbf{Z}^{\mathsf{T}} + \mathbf{Z}\mathsf{E}[\mathbf{e}_1\mathbf{e}_2^{\mathsf{T}}] + \mathsf{E}[\mathbf{e}_2\mathbf{e}_1^{\mathsf{T}}]\mathbf{Z}^{\mathsf{T}} + \mathsf{E}[\mathbf{e}_2\mathbf{e}_2^{\mathsf{T}}] \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{\Sigma}_1 & \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} \\ \mathbf{Z}\mathbf{\Sigma}_1 & \mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2 \end{pmatrix}.
\end{aligned}$$

Likewise, it holds that

$$\begin{aligned}
\mathbf{\Sigma}' &= \mathsf{E}[\mathbf{e}'\mathbf{e}'^{\mathsf{T}}] \\
&= \begin{pmatrix} \mathbf{W}\mathsf{E}[\mathbf{e}_2'\mathbf{e}_2'^{\mathsf{T}}]\mathbf{W}^{\mathsf{T}} + \mathbf{W}\mathsf{E}[\mathbf{e}_2'\mathbf{e}_1'^{\mathsf{T}}] + \mathsf{E}[\mathbf{e}_1'\mathbf{e}_2'^{\mathsf{T}}]\mathbf{W}^{\mathsf{T}} + \mathsf{E}[\mathbf{e}_1\mathbf{e}_1'^{\mathsf{T}}] & \mathbf{W}\mathsf{E}[\mathbf{e}_2'\mathbf{e}_2'^{\mathsf{T}}] + \mathsf{E}[\mathbf{e}_1'\mathbf{e}_2'^{\mathsf{T}}] \\ \mathsf{E}[\mathbf{e}_2'\mathbf{e}_2'^{\mathsf{T}}]\mathbf{W}^{\mathsf{T}} + \mathsf{E}[\mathbf{e}_2'\mathbf{e}_1'^{\mathsf{T}}] & \mathsf{E}[\mathbf{e}_2'\mathbf{e}_2'^{\mathsf{T}}] \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{W}\mathbf{\Sigma}_2'\mathbf{W}^{\mathsf{T}} + \mathbf{\Sigma}_1' & \mathbf{W}\mathbf{\Sigma}_2' \\ \mathbf{\Sigma}_2'\mathbf{W}^{\mathsf{T}} & \mathbf{\Sigma}_2' \end{pmatrix}.
\end{aligned}$$

Plugging in (2) and (3) we obtain that

$$\mathbf{\Sigma}' = \begin{pmatrix} \mathbf{\Sigma}_1 & \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} \\ \mathbf{Z}\mathbf{\Sigma}_1 & \mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2 \end{pmatrix} = \mathbf{\Sigma},$$

i.e. $\mathbf{\Sigma}$ and $\mathbf{\Sigma}'$ are indeed identical and therefore $\mathbf{e}$ and $\mathbf{e}'$ follow the same distribution.

We will now compute spectral bounds for $\mathbf{\Sigma}_1'$ and $\mathbf{\Sigma}_2'$. Concerning $\mathbf{\Sigma}_2'$, we can bound the largest singular value of $\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}$ as follows. It holds for all $\mathbf{x} \in \mathbb{R}^n$ that

$$\begin{aligned}
\mathbf{x}^{\mathsf{T}}\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}\mathbf{x} &= (\mathbf{Z}^{\mathsf{T}}\mathbf{x})^{\mathsf{T}}\mathbf{\Sigma}_1(\mathbf{Z}^{\mathsf{T}}\mathbf{x}) \\
&\leq \sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\mathbf{x}^{\mathsf{T}}\mathbf{Z}\mathbf{Z}^{\mathsf{T}}\mathbf{x} \\
&\leq \sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2\mathbf{x}^{\mathsf{T}}\mathbf{x} \\
&= \mathbf{x}^{\mathsf{T}}(\sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2\mathbf{I})\mathbf{x},
\end{aligned}$$

i.e. it holds that

$$\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} \leq \sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2\mathbf{I}. \tag{4}$$

It follows that

$$\mathbf{\Sigma}_2 \leq \mathbf{\Sigma}_2' \leq \mathbf{\Sigma}_2 + \sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2\mathbf{I}.$$

We will finally consider $\mathbf{\Sigma}_1'$. For this purpose we define $\mathbf{S} = \mathbf{Z}\sqrt{\mathbf{\Sigma}_2}$ and note that $\mathbf{Z}\mathbf{\Sigma}_2\mathbf{Z}^{\mathsf{T}} = \mathbf{S}\mathbf{S}^{\mathsf{T}}$ and $\mathbf{Z}\mathbf{\Sigma}_1 = \mathbf{S}\sqrt{\mathbf{\Sigma}_1}$. Now it holds that

$$\begin{aligned}
\mathbf{\Sigma}_1' &= \mathbf{\Sigma}_1 - \mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}}(\mathbf{Z}\mathbf{\Sigma}_1\mathbf{Z}^{\mathsf{T}} + \mathbf{\Sigma}_2)^{-1}\mathbf{Z}\mathbf{\Sigma}_1 \\
&= \mathbf{\Sigma}_1 - \sqrt{\mathbf{\Sigma}_1}\mathbf{S}^{\mathsf{T}}(\mathbf{S}\mathbf{S}^{\mathsf{T}} + \mathbf{\Sigma}_2)^{-1}\mathbf{S}\sqrt{\mathbf{\Sigma}_1} \qquad (5) \\
&\geq \mathbf{\Sigma}_1 - \frac{1}{1 + \frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)}{\sigma_{\mathsf{max}}(\mathbf{S}\mathbf{S}^{\mathsf{T}})}}\sqrt{\mathbf{\Sigma}_1}\mathbf{S}^{\mathsf{T}}(\mathbf{S}\mathbf{S}^{\mathsf{T}})^{-1}\mathbf{S}\sqrt{\mathbf{\Sigma}_1} \qquad (6)
\end{aligned}$$

$$\geq \mathbf{\Sigma}_1 - \frac{1}{1+\frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)}{\sigma_{\mathsf{max}}(\mathbf{S}\mathbf{S}^{\mathsf{T}})}}\mathbf{\Sigma}_1 \tag{7}$$

$$= \left(1 - \frac{1}{1+\frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)}{\sigma_{\mathsf{max}}(\mathbf{S}\mathbf{S}^{\mathsf{T}})}}\right)\mathbf{\Sigma}_1$$

$$\geq \left(1 - \frac{1}{1+\frac{\sigma_{\mathsf{min}}(\mathbf{\Sigma}_2)}{\sigma_{\mathsf{max}}(\mathbf{\Sigma}_1)\sigma_{\mathsf{max}}(\mathbf{Z})^2}}\right)\mathbf{\Sigma}_1, \tag{8}$$

where (5) follows from the definition of $\mathbf{S}$, (6) follows by Lemma 6, (7) follows by Lemma 7 and (8) follows by (4) as $\mathbf{S}\mathbf{S}^{\mathsf{T}} = \mathbf{Z}\mathbf{\Sigma}_2\mathbf{Z}^{\mathsf{T}}$. □

## 5.2 Discrete Gaussian Leakage

The following Theorem gives us a way to *obliviously* simulate an error term $\mathbf{e}_1$ together with leakage $\mathbf{l} = \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2$. Given a gaussian $\mathbf{e}$ we can choose the leakage term independently of $\mathbf{e}$, and then adjusting $\mathbf{e}$ *additively* depending on the leakage term.

**Theorem 2.** *Let $\epsilon, \beta > 0$ and let $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2$ be lattices with (possibly) different dimension. Let $\mathbf{Z}$ be such that for all $\mathbf{x} \in \mathbf{\Lambda}_1$ it holds that $\mathbf{Z}\mathbf{x} \in \mathbf{\Lambda}_2$. Furthermore let $\sigma_{\mathsf{max}}(\mathbf{Z}) \leq \beta$. Let $\mathbf{\Sigma}_0$ be a positive definite matrix with same dimensions as $\mathbf{\Lambda}_1$ and $\sqrt{\mathbf{\Sigma}_0} \geq \eta_\epsilon(\mathbf{\Lambda}_1)$ and let $\mathbf{T}_0$ be a positive definite matrix with same dimensions as $\mathbf{\Lambda}_2$ and $\sqrt{\mathbf{T}_0} \geq \eta_\epsilon(\mathbf{\Lambda}_2)$. Let $s, t \geq 2\sqrt{2}$ and assume that*

$$t^2\sigma_{\mathsf{min}}(\mathbf{T}_0) \geq \frac{(s^2+1)(s^2+2)}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\cdot\beta^2.$$

*Further let*

$$\begin{aligned}
\mathbf{\Sigma} &= (s^2+1)\mathbf{\Sigma}_0\\
\mathbf{T} &= (t^2+1)\mathbf{T}_0\\
\mathbf{T}' &= t^2\mathbf{T}_0 + (s^2+1)\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}}\\
\tilde{\mathbf{\Sigma}} &= \frac{3}{4}s^2\mathbf{\Sigma}_0 - (s^2+1)^2\mathbf{\Sigma}_0\mathbf{Z}\mathbf{T}'^{-1}\mathbf{Z}\mathbf{\Sigma}_0\\
\bar{\mathbf{\Sigma}} &= \frac{s^2}{4}\cdot\mathbf{\Sigma}_0\\
\mathbf{W} &= (s^2+1)\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}}\mathbf{T}'^{-1}.
\end{aligned}$$

*Now let $\mathbf{e}_1 \sim D_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}}}$ and $\mathbf{e}_2 \sim D_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T}}}$, $\mathbf{e} \sim D_{\mathbf{\Lambda}_1,\sqrt{\bar{\mathbf{\Sigma}}}}$, $\tilde{\mathbf{e}} \sim D_{\sqrt{\tilde{\mathbf{\Sigma}}}}$ and $\mathbf{d}^* \sim D_{\sqrt{\mathbf{T}'}}$. Then it holds that*

$$(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2) \approx_{22\epsilon} (\mathbf{e} + \mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\tilde{\mathbf{e}} + \mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T}_0}}(\mathbf{d}^*)).$$

*Proof.* Let $\mathbf{e}_1 \sim D_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}}}$ and $\mathbf{e}_2 \sim D_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T}}}$, where $\mathbf{\Sigma} = (s^2+1)\mathbf{\Sigma}_0$ and $\mathbf{T} = (t^2+1)\mathbf{T}_0$. Now let $\mathbf{e}_2' \sim D_{t\sqrt{\mathbf{T}_0}}$. By Lemma 5 (first item) it holds that

$$(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathbf{e}_2) \approx_{8\epsilon} (\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T}_0}}(\mathbf{e}_2')).$$

We will now further decompose $\mathbf{e}_2'$. Let $\mathbf{T}^* = t^2\mathbf{T}_0 - \frac{s^2+1}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}}$. Note that

$$\sigma_{\mathsf{min}}(\mathbf{T}^*) \geq t^2\sigma_{\mathsf{min}}(\mathbf{T}_0) - \frac{s^2+1}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}\mathbf{Z}^2 > 0, \tag{9}$$

hence $\mathbf{T}^*$ is positive definite. Now let $\mathbf{e}_2^* \sim D_{\sqrt{\mathbf{T}^*}}$ and $\tilde{\mathbf{e}}_1 \sim D_{\sqrt{(s^2+1)/s^2 \cdot \mathbf{\Sigma}_0}}$. Hence $\mathbf{e}_2^* + \mathbf{Z}\tilde{\mathbf{e}}_1$ is a continuous gaussian with covariance matrix

$$\mathbf{T}^* + \mathbf{Z}((s^2+1)/s^2 \cdot \mathbf{\Sigma}_0)\mathbf{Z}^{\mathsf{T}} = t^2\mathbf{T}_0 - \frac{s^2+1}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}} + \frac{s^2+1}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}} = t^2\mathbf{T}_0,$$

thus $\mathbf{e}_2^* + \mathbf{Z}\tilde{\mathbf{e}}_1$ and $\mathbf{e}_2'$ have the same covariance matrix and are thus identically distributed, i.e. it holds that $\mathbf{e}_2' \equiv \mathbf{e}_2^* + \mathbf{Z}\tilde{\mathbf{e}}_1$. We thus have

$$\begin{aligned}(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2')) &\equiv (\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1 + \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}\tilde{\mathbf{e}}_1)) \\ &\equiv (\mathbf{e}_1, \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}\tilde{\mathbf{e}}_1 + \mathbf{Z}\mathbf{e}_1)) \\ &\equiv (\mathbf{e}_1, \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}(\tilde{\mathbf{e}}_1 + \mathbf{e}_1))).\end{aligned}$$

Now let $\mathbf{e}_1^* \sim D_{s\sqrt{\mathbf{\Sigma}_0}}$. Since $\mathbf{e}_1 \sim D_{\mathbf{\Lambda}_1,\sqrt{(s^1+1)\mathbf{\Sigma}_0}}$ and $\tilde{\mathbf{e}}_1 \sim D_{\sqrt{\mathbf{\Sigma}_0}}$ it holds by Lemma 5 (second item) that $(\mathbf{e}_1^*, \mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}_1^*)) \approx_{10\epsilon} ((s^2/(s^2+1) \cdot (\mathbf{e}_1 + \tilde{\mathbf{e}}_1), \mathbf{e}_1)$. Letting $\mathbf{Z}^* = (s^2+1)/s^2 \cdot \mathbf{Z}$ it holds that

$$\begin{aligned}(\mathbf{e}_1, \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}(\tilde{\mathbf{e}}_1 + \mathbf{e}_1))) &\equiv (\mathbf{e}_1, \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}^*(s^2/(s^2+1) \cdot (\tilde{\mathbf{e}}_1 + \mathbf{e}_1)))) \\ &\approx_{10\epsilon} (\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}_1^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}^*\mathbf{e}_1^*)).\end{aligned}$$

We are now ready to invoke Lemma 8. Let

$$\begin{aligned}\mathbf{T}' &= \mathbf{T}^* + \mathbf{Z}^*(s^2\mathbf{\Sigma}_0)(\mathbf{Z}^*)^{\mathsf{T}} \\ &= t^2\mathbf{T}_0 - \frac{s^2+1}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}} + \mathbf{Z}^*(s^2\mathbf{\Sigma}_0)(\mathbf{Z}^*)^{\mathsf{T}} \\ &= t^2\mathbf{T}_0 - \frac{s^2+1}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}} + \frac{(s^2+1)^2}{s^2}\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}} \\ &= t^2\mathbf{T}_0 + (s^2+1)\mathbf{Z}\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}},\end{aligned}$$

further

$$\begin{aligned}\mathbf{W} &= s^2\mathbf{\Sigma}_0(\mathbf{Z}^*)^{\mathsf{T}}\mathbf{T}'^{-1} \\ &= (s^2+1)\mathbf{\Sigma}_0\mathbf{Z}^{\mathsf{T}}\mathbf{T}'^{-1}\end{aligned}$$

and

$$\begin{aligned}\mathbf{\Sigma}' &= s^2\mathbf{\Sigma}_0 - (s^2\mathbf{\Sigma}_0)\mathbf{Z}^{*\mathsf{T}}\mathbf{T}'^{-1}\mathbf{Z}^*(s^2\mathbf{\Sigma}_0) \\ &= s^2\mathbf{\Sigma}_0 - (s^2+1)^2\mathbf{\Sigma}_0\mathbf{Z}\mathbf{T}'^{-1}\mathbf{Z}\mathbf{\Sigma}_0.\end{aligned}$$

Now let $\mathbf{e}^* \sim D_{\sqrt{\mathbf{\Sigma}'}}$ and $\mathbf{d}^* \sim D_{\sqrt{\mathbf{T}'}}$. Recalling that $\mathbf{e}_1^* \sim D_{s^2\sqrt{\mathbf{\Sigma}_0}}$ and $\mathbf{e}_2^* \sim D_{\sqrt{\mathbf{T}^*}}$, it now holds by Lemma 8 that $(\mathbf{e}_1^*, \mathbf{Z}^*\mathbf{e}_1^* + \mathbf{e}_2^*) \equiv (\mathbf{e}^* + \mathbf{W}\mathbf{d}^*, \mathbf{d}^*)$. Hence it holds that

$$(\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}_1^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{e}_2^* + \mathbf{Z}^*\mathbf{e}_1^*)) \equiv (\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}^* + \mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{d}^*)).$$

Recall that by equation (9) it holds that

$$\sigma_{\mathsf{min}}(\mathbf{T}^*) \geq t^2\sigma_{\mathsf{min}}(\mathbf{T}_0) - \frac{s^2+1}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}\mathbf{Z}^2.$$

Further recalling that by assumption we have that

$$t^2\sigma_{\mathsf{min}}(\mathbf{T}_0) \geq \frac{(s^2+1)(s^2+2)}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}\mathbf{Z}^2,$$

we have that

$$\begin{aligned}\sigma_{\mathsf{min}}(\mathbf{T}^*) &\geq \frac{(s^2+1)(s^2+2)}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}\mathbf{Z}^2 - \frac{s^2+1}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}\mathbf{Z}^2\\ &= \frac{(s^2+1)^2}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\sigma_{\mathsf{max}}(\mathbf{Z})^2\\ &= \sigma_{\mathsf{max}}(s^2\mathbf{\Sigma}_0)\cdot\sigma_{\mathsf{max}}(\mathbf{Z}^*)^2.\end{aligned}$$

Thus, together with the bound on $\mathbf{\Sigma}'$ in Lemma 8 it holds that

$$\mathbf{\Sigma}' \geq \left(1 - \frac{1}{1+\frac{\sigma_{\mathsf{min}}(\mathbf{T}^*)}{\sigma_{\mathsf{max}}(s^2\mathbf{\Sigma}_0)\cdot\sigma_{\mathsf{max}}(\mathbf{Z}^*)^2}}\right)s^2\mathbf{\Sigma}_0 \geq \frac{s^2}{2}\mathbf{\Sigma}_0.$$

In the final step, we will decompose $\mathbf{e}^* \sim D_{\sqrt{\mathbf{\Sigma}'}}$ as the sum of a discrete and a continuous gaussian. Let $\mathbf{e} \sim D_{\mathbf{\Lambda}_1, s/2\cdot\sqrt{\mathbf{\Sigma}_0}}$ and $\tilde{\mathbf{e}} \sim D_{\sqrt{\mathbf{\Sigma}'-s^2/4\mathbf{\Sigma}_0}}$. First note that $\Sigma' - s^2/4\mathbf{\Sigma}_0$ is positive definite, as it holds that

$$\mathbf{\Sigma}' - s^2/4\cdot\mathbf{\Sigma}_0 \geq s^2/2\cdot\mathbf{\Sigma}_0 - s^2/4\cdot\mathbf{\Sigma}_0 = s^2/4\cdot\mathbf{\Sigma}_0 > 0.$$

As by $s \geq 2\sqrt{2}$ it holds that

$$\frac{1}{s^2/4} + \frac{1}{s^2/4} = \frac{8}{s^2} \leq 1,$$

it holds by Lemma 4 that $\mathbf{e} + \tilde{\mathbf{e}} \approx_{4\epsilon} \mathbf{e}^*$. From this we can conclude that

$$\begin{aligned}(\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}^*+\mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{d}^*)) &\approx_{4\epsilon} (\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{e}+\tilde{\mathbf{e}}+\mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{d}^*))\\ &\equiv (\mathbf{e}+\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\tilde{\mathbf{e}}+\mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{d}^*)).\end{aligned}$$

Putting all together, it holds that

$$(\mathbf{e}_1, \mathbf{Z}\mathbf{e}_1+\mathbf{e}_2) \approx_{22\epsilon} (\mathbf{e}+\mathsf{round}_{\mathbf{\Lambda}_1,\sqrt{\mathbf{\Sigma}_0}}(\tilde{\mathbf{e}}+\mathbf{W}\mathbf{d}^*), \mathsf{round}_{\mathbf{\Lambda}_2,\sqrt{\mathbf{T_0}}}(\mathbf{d}^*)),$$

which concludes the proof. □

## 6 LWE with Error-Leakage

In this Section we will introduce a hardness assumption we call *LWE with Error-Leakage* and show that the problem is as hard as standard LWE for a certain parameter regime.

**Definition 5 ($\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\mathcal{L}}$ Assumption).** *Let $\mathcal{R}, n, m, k, q, \chi, \bar{\chi}, \mathcal{L}$ be parametrised by $\lambda$, where $\mathcal{L} \subset \mathcal{R}^{m\times k}$ is an efficiently decidable set. The (decisional) $\mathsf{LWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\mathcal{L}}$ problem is defined via the following experiment, where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a two-stage PPT adversary.*

- *The challenger chooses $\mathbf{A} \leftarrow\$ \mathcal{R}_q^{n\times m}$ uniformly at random and provides $\mathbf{A}$ to $\mathcal{A}_1$.*
- *$\mathcal{A}_1$ outputs a matrix $\mathbf{Z} \in \mathcal{L}$ to the challenger (If $\mathbf{Z} \notin \mathcal{L}$ the challenger aborts and outputs a random bit).*
- *The challenger chooses a uniformly random bit $b \leftarrow\$ \{0,1\}$.*
- *If $b = 0$, he samples $\mathbf{s} \leftarrow\$ \mathcal{R}_q^n$, $\mathbf{e} \leftarrow\$ \chi^m$, $\bar{\mathbf{e}} \leftarrow\$ \bar{\chi}^k$ and sets $\mathbf{y}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}\cdot\mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q$ and $\mathbf{l} = \mathbf{e}^{\mathsf{T}}\cdot\mathbf{Z} + \bar{\mathbf{e}}^{\mathsf{T}}$.*
- *If $b = 1$, he samples $\mathbf{y} \leftarrow\$ \mathcal{R}_q^m$ uniformly at random, $\mathbf{e} \leftarrow\$ \chi^m$, $\bar{\mathbf{e}} \leftarrow\$ \bar{\chi}^k$ and computes $\mathbf{l} = \mathbf{e}^{\mathsf{T}}\cdot\mathbf{Z} + \bar{\mathbf{e}}^{\mathsf{T}}$.*
- *The challenger now runs $\mathcal{A}_2$ on input $(\mathbf{A}, \mathbf{y}, \mathbf{l})$, upon which $\mathcal{A}_2$ outputs a bit $b'$.*
- *If $b' = b$, the challenger outputs 1, otherwise 0.*

*We say that $\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\mathcal{L}}$ holds if every PPT adversary $\mathcal{A}$ has at most negligible advantage in the above experiment.*

As the set $\mathcal{L}$ that specifies the admissible leakage functions we will use the $L_\infty$-ball in $\mathcal{R}^m$

$$\mathcal{L}_\infty(\beta) = \{\mathbf{z}^\mathrm{T} \in \mathcal{R}^m \mid \|\mathbf{z}\|_\infty \leq \beta\}.$$

The following Lemma bounds the maximal singular value of elements of $\mathcal{L}_\infty(\beta)$, interpreted as matrices in $\mathbb{R}^{d\times dm}$ via the coefficient embedding.

**Lemma 9.** *Let $\mathcal{R}$ be a ring of degree $d$ with an embedding $\Phi : \mathcal{R} \to \mathbb{R}^d$. We can describe $\Phi$ via a transformation matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$, i.e. we have a generating set $\mathbf{r}_1, \dots, \mathbf{r}_d \in \mathcal{R}$ for which it hold that $\Phi(\sum_{i=1}^d \alpha_i \mathbf{r}_i) = \mathbf{M}\mathbf{a}$, where $\mathbf{a} = (\alpha_1, \dots, \alpha_d)$.*

*Now assume that it holds that $\|\Phi(\mathbf{a}\cdot\mathbf{b})\|_\infty \leq \gamma \cdot \|\Phi(\mathbf{a})\|_\infty \cdot \|\Phi(\mathbf{b})\|_\infty$. Fix an $\mathbf{a} \in \mathcal{R}$ and define the linear transformation $\Theta : \mathbb{R}^n \to \mathbb{R}^n$ via*

$$\Theta(\mathbf{x}) = \Phi(\mathbf{a}\cdot\Phi^{-1}(\mathbf{x})) = \mathbf{A}\cdot\mathbf{x}.$$

*Then $\mathbf{A}$ has maximal singular value $\gamma\beta\sqrt{d}$.*

*Proof.* It holds that

$$\|\mathbf{A}\mathbf{x}\|_2 \leq \sqrt{d}\|\mathbf{A}\mathbf{x}\|_\infty = \sqrt{d}\|\Phi(\mathbf{a}\cdot\Phi^{-1}(\mathbf{x}))\|_\infty \leq \gamma\sqrt{d}\|\Phi(\mathbf{a})\|_\infty \cdot \|\Phi(\Phi^{-1}(\mathbf{x}))\|_\infty = \gamma\beta\sqrt{d}\|\mathbf{x}\|_\infty \leq \gamma\beta\sqrt{d}\|\mathbf{x}\|_2$$

We will now show that for suitable discrete gaussian error distributions, *LWE* implies $\mathsf{elLWE}$ tightly with only a small loss in parameters.

**Theorem 3.** *Let $\beta > 0$ be a parameter. Let $\epsilon > 0$ be negligible. Let $\mathcal{R}$ be a suitable ring with an embedding as a lattice in $\mathbb{R}^n$, and let $\mathbf{\Sigma}_0$ be a covariance matrix with $\sqrt{\mathbf{\Sigma}_0} \geq \eta_\epsilon(\mathcal{R})$. Let $s, t \geq 2\sqrt{2}$ be such that*

$$t^2\sigma_{\mathsf{min}}(\mathbf{\Sigma}_0) \geq \frac{(s^2+1)(s^2+2)}{s^2}\sigma_{\mathsf{max}}(\mathbf{\Sigma}_0)\cdot\beta^2.$$

*Now let $\chi = D_{\mathcal{R},\sqrt{(s^2+1)\mathbf{\Sigma}_0}}$, $\bar{\chi} = D_{\mathcal{R},\sqrt{(t^2+1)\mathbf{\Sigma}_0}}$ and $\chi^* = D_{\mathcal{R},s/2\cdot\sqrt{\mathbf{\Sigma}_0}}$. Then, assuming that $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi^*}$ is hard, $\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\beta}$ is also hard. More precisely, assume the exists a distinguisher $\mathcal{D}$ with advantage $\delta$ against $\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\beta}$. Then there exists a distinguisher $\mathcal{D}'$ with advantage $\delta - 44\epsilon$ against $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi^*}$.*

*Proof.* Let $\mathcal{D}$ be a PPT distinguisher against $\mathsf{elLWE}$ with advantage $\delta$. We will construct the distinguisher $\mathcal{D}'$ as follows.

- Given $(\mathbf{A}, \mathbf{y})$, provide $\mathbf{A}$ to $\mathcal{D}_1$, which outputs a matrix $\mathbf{Z}$.
- From $\mathbf{Z}$, compute matrices $\mathbf{T}', \tilde{\mathbf{\Sigma}}$ and $\mathbf{W}$ as in the statement of Theorem 2 (setting $\mathbf{T}_0 = \mathbf{\Sigma}_0$).
- Choose $\mathbf{d}^* \sim D^k_{\sqrt{\mathbf{T}'}}$ and $\tilde{\mathbf{e}} \sim D^m_{\sqrt{\tilde{\mathbf{\Sigma}}}}$.
- Compute $\mathbf{y}' = \mathbf{y} + \mathsf{round}_{\mathcal{R}^m,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{W}\mathbf{d}^* + \tilde{\mathbf{e}})$ and $\mathbf{l} = \mathsf{round}_{\mathcal{R}^k,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{d}^*)$.
- Run $\mathcal{D}_2$ on input $(\mathbf{A}, \mathbf{y}', \mathbf{l})$ and output whatever $\mathcal{D}_2$ outputs.

If $(\mathbf{A}, \mathbf{y})$ is a well-formed $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi^*}$ sample it holds that $\mathbf{y} = \mathbf{s}\mathbf{A} + \mathbf{e}$, where $\mathbf{e} \sim D_{\mathcal{R}^m,s/2\cdot\sqrt{\mathbf{\Sigma}_1}}$. Consequently, by Theorem 2 it holds that

$$\begin{aligned}(\mathbf{A}, \mathbf{y}', \mathbf{l}) &\equiv (\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{e} + \mathsf{round}_{\mathcal{R}^m,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{W}\mathbf{d}^* + \tilde{\mathbf{e}}), \mathsf{round}_{\mathcal{R}^k,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{d}^*))\\ &\approx_{22\epsilon} (\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{e}_1, \mathbf{e}_1\mathbf{Z} + \mathbf{e}_2)\end{aligned}$$

where $\mathbf{e}_1 \sim \chi^m$ and $\mathbf{e}_2 \sim \bar{\chi}^k$. That is, in this case the sample computed by $\mathcal{D}'$ is statistically close to a sample of $\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\beta}$ for $b = 0$.

On the other hands, if $\mathbf{y}$ is distributed uniformly random, we can write $\mathbf{y}$ as $\mathbf{y} = \mathbf{u} + \mathbf{e}$ for a uniformly random $\mathbf{u}$ and $\mathbf{e} \sim \chi^m$. Consequently, in this case it holds also by Theorem 2 that

$$\begin{aligned}(\mathbf{A}, \mathbf{y}', \mathbf{l}) &\equiv (\mathbf{A}, \mathbf{u} + \mathbf{e} + \mathsf{round}_{\mathcal{R}^m,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{W}\mathbf{d}^* + \tilde{\mathbf{e}}), \mathsf{round}_{\mathcal{R}^k,\sqrt{\mathbf{\Sigma}_0}}(\mathbf{d}^*))\\ &\approx_{22\epsilon} (\mathbf{A}, \mathbf{u} + \mathbf{e}_1, \mathbf{e} - 1\mathbf{Z} + \mathbf{e}_2)\\ &\equiv (\mathbf{A}, \mathbf{u}', \mathbf{e}_1\mathbf{Z} + \mathbf{e}_2)\end{aligned}$$

That is, in this case the sample computed by $\mathcal{D}'$ is statistically close to a sample of $\mathsf{elLWE}_{\mathcal{R},n,m,k,q,\chi,\bar{\chi},\beta}$ for $b = 1$. Putting these two facts together it follows that $\mathcal{D}'$ has advantage $\delta - 44\epsilon$. □

## 7 Laconic Encryption

We define a new notion of encryption called laconic encryption and give an efficient construction based on the LWE assumption. Laconic encryption is essentially registration-based encryption with less stringent efficiency requirements.

### 7.1 Definition

**Definition 6 (Laconic Encryption).** *A laconic encryption scheme for message space $\mathcal{M}$ consists of a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{WGen}, \mathsf{Dec})$ *with the following syntax:*

- $(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$*: The setup algorithm is a randomized algorithm which takes as input the security parameter* $1^\lambda$ *and a length parameter* $1^\ell$*. It generates the public parameters* $\mathsf{pp}$*, a state* $\mathsf{st}$*, and some auxiliary information* $\mathsf{aux}$*.*
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: The key generation algorithm takes as input the public parameters* $\mathsf{pp}$ *and outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{st}' \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$*: The membership update algorithm, with (read-and-write-)random access to the auxiliary information* $\mathsf{aux}$*, takes as input the public parameters* $\mathsf{pp}$*, the state* $\mathsf{st}$*, an index* $\mathsf{ind} \in \{0,1\}^\ell$*, and a public key* $\mathsf{pk}$ *(or $\perp$). It outputs updated state* $\mathsf{st}'$*.*
- $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg})$*: The encryption algorithm is a randomized algorithm which takes as input the public parameters* $\mathsf{pp}$*, the state* $\mathsf{st}$*, an index* $\mathsf{ind} \in \{0,1\}^\ell$*, and a message* $\mathsf{msg} \in \mathcal{M}$*. It outputs a ciphertext* $\mathsf{ctxt}$*.*
- $\mathsf{wit} \leftarrow \mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$*: The witness generation algorithm, with (read-)random access to the auxiliary information* $\mathsf{aux}$*, takes as input the public parameters* $\mathsf{pp}$*, the state* $\mathsf{st}$*, an index* $\mathsf{ind} \in \{0,1\}^\ell$*, and a public key* $\mathsf{pk}$*. It outputs a (non)-membership witness* $\mathsf{wit}$*.*
- $\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt})$*: The decryption algorithm takes as input a secret key* $\mathsf{sk}$*, a membership witness* $\mathsf{wit}$*, and a ciphertext* $\mathsf{ctxt}$*. It outputs a message* $\mathsf{msg}$*.*

*Furthermore, there exists $t \in \mathsf{poly}(\lambda, \ell)$ such that all above algorithms run in time at most $t(\lambda, \ell)$.*

An instance of laconic encryption is initialized by a server running the $\mathsf{Setup}$ algorithm and publishing the public parameters $\mathsf{pp}$ and the state $\mathsf{st}$. The auxiliary information $\mathsf{aux}$ is kept (not necessarily secret) by the server for generating membership witnesses later. The state $\mathsf{st}$ and the auxiliary information $\mathsf{aux}$ can be thought of as a short digest and a long encoding, respectively, of a set of registered tuples $(\mathsf{ind}, \mathsf{pk})$ of index and public keys, which is initially empty.

Given the public parameters $\mathsf{pp}$, anyone can generate a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The membership update algorithm allows registering a new key $\mathsf{pk} \neq \perp$ under an index $\mathsf{ind}$, possibly replacing a previously registered key or removing the key previously registered under index $\mathsf{ind}$ (if any) if $\mathsf{pk} = \perp$. Correspondingly, the witness generation algorithm generates a (non-)membership witness $\mathsf{wit}$ of $(\mathsf{ind}, \mathsf{pk})$ (not) being registered in the state $\mathsf{st}$. The encryption algorithm allows one to encrypt a message with respect to an index $\mathsf{ind}$, such that the ciphertext can be decrypted given a membership witness of $(\mathsf{ind}, \mathsf{pk})$ and the secret key $\mathsf{sk}$ corresponding to $\mathsf{pk}$. Crucially, the encryption algorithm does not require the knowledge of the public key registered under index $\mathsf{ind}$.

Note that the size of the auxiliary information $\mathsf{aux}$ could be linear in the number of registered keys (exponential in $\ell$). No algorithm requires $\mathsf{aux}$ as input, $\mathsf{WGen}$ and $\mathsf{Upd}$ only require random access to it. Therefore, by the definition of PPT, all algorithms are required to run in time $\mathsf{poly}(\lambda, \ell)$.

With the above-intended use of laconic encryption in mind, we define correctness and security. Our correctness definition considers a scenario where the public parameters have underdone an arbitrary sequence of updates such that in the latest version a tuple $(\mathsf{ind}, \mathsf{pk})$ is registered. In this case, if a message is encrypted with respect to $(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$, then decrypting the ciphertext with the secret key $\mathsf{sk}$ corresponding to $\mathsf{pk}$ recovers the message with overwhelming probability.

```
$\mathsf{Correctness}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$
$\mathsf{Honest} :=$ Empty dictionary
$(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$(\mathsf{ind}^*, \mathsf{msg}^*) \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}, \mathsf{Upd}\mathcal{O}}(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$
$\textbf{if } \mathsf{ind}^* \notin \mathsf{Honest} \textbf{ then return } 1$
$\mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}^*, \mathsf{msg}^*)$
$(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Honest}[\mathsf{ind}^*]$
$\mathsf{wit}^* \leftarrow \mathsf{WGen}(\mathsf{pp}, \mathsf{st}, \mathsf{aux}, \mathsf{ind}^*, \mathsf{pk}^*)$
$\mathsf{msg} = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{wit}^*, \mathsf{ctxt}^*)$
$\textbf{return } (\mathsf{msg} = \mathsf{msg}^*)$
```

```
$\mathsf{KGen}\mathcal{O}(\mathsf{ind})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathsf{st} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
$\mathsf{Honest}[\mathsf{ind}] := (\mathsf{pk}, \mathsf{sk})$
$\textbf{return } (\mathsf{st}, \mathsf{aux}, \mathsf{pk})$
```

```
$\mathsf{Upd}\mathcal{O}(\mathsf{ind}, \mathsf{pk})$
$\mathsf{pp} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
$\textbf{if } \mathsf{pk} = \perp \textbf{ then } \mathsf{Honest}[\mathsf{ind}] := \perp$
$\textbf{else } \mathsf{Malicious} := \mathsf{Malicious} \cup \{\mathsf{ind}\}$
$\textbf{return } (\mathsf{pp}, \mathsf{aux})$
```

```
$\mathsf{Security}^b_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$
$\mathsf{Malicious} := \emptyset$
$(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$((\mathsf{ind}_0, \mathsf{msg}_0), (\mathsf{ind}_1, \mathsf{msg}_1)) \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}, \mathsf{Upd}\mathcal{O}}(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$
$\textbf{if } \{\mathsf{ind}_0, \mathsf{ind}_1\} \cap \mathsf{Malicious} \neq \emptyset \textbf{ then return } 0$
$\mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}_b, \mathsf{msg}_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}, \mathsf{Upd}\mathcal{O}}(\mathsf{ctxt}^*)$
$\textbf{return } b'$
```

```
$\mathsf{Pseudorandomness}^b_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$
$\mathsf{Malicious} := \emptyset$
$(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$(\mathsf{ind}^*, \mathsf{msg}^*) \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}, \mathsf{Upd}\mathcal{O}}(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$
$\textbf{if } \mathsf{ind}^* \in \mathsf{Malicious} \textbf{ then return } 0$
$\textbf{if } b = 0 \textbf{ then } \mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}^*, \mathsf{msg}^*)$
$\textbf{else } \mathsf{ctxt}^* \leftarrow\!\!\$\ \mathcal{C}$
$b' \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}, \mathsf{Upd}\mathcal{O}}(\mathsf{ctxt}^*)$
$\textbf{return } b'$
```

**Fig. 1.** Correctness, security, pseudorandomness and update privacy experiments for laconic encryption.

**Definition 7 (Correctness).** *A laconic encryption scheme $\Pi$ is said to be statistically correct if for any (unbounded) algorithm $\mathcal{A}$, any $\ell = \mathsf{poly}(\lambda)$, it holds that*

$$\Pr\left[\mathsf{Correctness}_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

*where the experiment* $\mathsf{Correctness}_{\Pi,\mathcal{A}}$ *is defined in Fig. 1.*

Our security definition combines both index and message-hiding. It requires that if each of two adversarially chosen indices $\mathsf{ind}_0, \mathsf{ind}_1$ is either registered by an honest party (so that the secret key is unknown to the adversary) or not registered, then for any adversarially chosen messages $\mathsf{msg}_0, \mathsf{msg}_1$ the adversary should not be able to distinguish a ciphertext encrypting $\mathsf{msg}_0$ with respect to $\mathsf{ind}_0$ from that encrypting $\mathsf{msg}_1$ with respect to $\mathsf{ind}_1$.

**Definition 8 (Security).** *A laconic encryption scheme $\Pi$ is said to be secure if for any PPT (stateful) adversary $\mathcal{A}$, any $\ell = \mathsf{poly}(\lambda)$, it holds that*

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

*where the experiment* $\mathsf{Security}^b_{\Pi,\mathcal{A}}$ *is defined in Fig. 1.*

We will further define a slightly stronger security notion called *pseudorandom ciphertexts*. In essence, this property guarantees that if an index $\mathsf{ind}^*$ has not been registered, then a ciphertext with respect to $\mathsf{ind}^*$ looks pseudorandom.

**Definition 9 (Pseudorandom Ciphertexts).** *A laconic encryption scheme $\Pi$ with ciphertext space $\mathcal{C}$ is said to be have* pseudorandom ciphertexts *if for any PPT (stateful) adversary $\mathcal{A}$, any $\ell = \mathsf{poly}(\lambda)$, it holds that*

$$\begin{aligned}&\left|\Pr\left[\mathsf{Pseudorandomness}^0_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right] - \Pr\left[\mathsf{Pseudorandomness}^1_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right]\right| \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

*where the experiment $\mathsf{Pseudorandomness}^b_{\Pi,\mathcal{A}}$ is defined in Fig. 1.*

We also define a security notion called *update privacy.* It requires that the state $\mathsf{st}$ hides the indices where the public keys have been registered.

**Definition 10 (Update Privacy).** *A laconic encryption scheme $\Pi$ is said to be updated private if the distribution of the state $\mathsf{st}$ is (statistically close to) independent of the update operations $\mathsf{Upd}^{\mathsf{aux}}$.*

## 7.2 Our Construction

We construct a laconic encryption scheme for the message space $\mathcal{M} = \mathcal{R}_2$ in Fig. 2.

**Overview.** Our construction is parametrised by $n, m, p, q \in \mathbb{N}$ and distributions $\chi, \bar{\chi}$ over $\mathcal{R}$. The public parameters consists of random matrices $\mathbf{A}_0, \mathbf{A}_1 \leftarrow\$ \mathcal{R}_q^{n \times m}$ and $\mathbf{B} \leftarrow\$ \mathcal{R}_q^{n \times m}$, and initially two copies of a random vector $\mathbf{y}_\epsilon = \mathbf{y}^* \leftarrow\$ \mathcal{R}_q^n$. The vector $\mathbf{y}_\epsilon$ serves as the label of the root node $\epsilon$, while the vector $\mathbf{y}^*$ is a special "terminator" label which is assigned to the top-most node of any uninitialized path. In addition to the public parameters, the setup also generates some auxiliary information $\mathsf{aux}$, which consists of the (already assigned) labels of all nodes in the Merkle tree. At setup, since no public key is registered yet, no path in the tree is assigned, and therefore the root node is assigned the terminator label, i.e. $\mathbf{y}_\epsilon = \mathbf{y}^*$, and $\mathsf{aux}$ contains only $\mathbf{y}_\epsilon$. Throughout the lifetime of the system, $\mathbf{y}_\epsilon$ and $\mathsf{aux}$ will be updated, while the rest of the public parameters will be fixed once and for all.

Each pair of secret and public keys takes the form of $(\mathbf{x}, \mathbf{y}) \in \mathcal{R}_p^m \times \mathcal{R}_q^n$ satisfying $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$. To insert a public key $\mathbf{y}_{\mathsf{ind}}$ or to replace an old key with $\mathbf{y}_{\mathsf{ind}}$ at index $\mathsf{ind} \in \{0,1\}^\ell$, the public parameters and auxiliary information are updated as follows. First, the key $\mathbf{y}_{\mathsf{ind}}$ is assigned (or reassigned) as the label of the leaf node $\mathsf{ind}$. Then, following the path from the leaf node $\mathsf{ind}$ (exclusive) towards the root, the label of each node $\mathsf{str}$ (and possibly their children) in the path is computed recursively as follows:

(i) If the left child $(\mathsf{str}\|0)$ is not assigned a label yet, assign it the terminator label $\mathbf{y}^*$.
(ii) Similarly, if the right child $(\mathsf{str}\|1)$ is not assigned a label yet, assign it the terminator label $\mathbf{y}^*$.
(iii) Let $\mathbf{y}_{\mathsf{str}\|0}$ and $\mathbf{y}_{\mathsf{str}\|1}$ be the labels of the children of $\mathsf{str}$.
(iv) Compute $\mathbf{u}_{\mathsf{str}\|0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{str}\|0})$ and $\mathbf{u}_{\mathsf{str}\|1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{str}\|1})$.
(v) Compute $\mathbf{y}_{\mathsf{str}} := \mathbf{A}_0 \cdot \mathbf{u}_{\mathsf{str}\|0} + \mathbf{A}_1 \cdot \mathbf{u}_{\mathsf{str}\|1} \bmod q$.
(vi) Assign (or reassign) $\mathbf{y}_{\mathsf{str}}$ as the label of node $\mathsf{str}$.

To encrypt a message $\mathsf{msg} \in \mathcal{R}_2$ with respect to index $\mathsf{ind}$ against the current root label $\mathbf{y}_\epsilon$, the encryptor essentially performs dual-Regev encryption using $(\mathbf{A}_{\mathsf{ind}}, \mathbf{v})$ as the public key, where

$$\mathbf{A}_{\mathsf{ind}} := \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 & & & & & & \\ \bar{\mathsf{ind}}_1 \cdot \mathbf{G} & \mathsf{ind}_1 \cdot \mathbf{G} & \mathbf{A}_0 & \mathbf{A}_1 & & & & \\ & & \bar{\mathsf{ind}}_2 \cdot \mathbf{G} & \mathsf{ind}_2 \cdot \mathbf{G} & & & & \\ & & & & \ddots\ \ddots & & & \\ & & & & & \mathbf{A}_0 & \mathbf{A}_1 & \\ & & & & & \bar{\mathsf{ind}}_\ell \cdot \mathbf{G} & \mathsf{ind}_\ell \cdot \mathbf{G} & \\ & & & & & & & \mathbf{B} \end{pmatrix} \quad \text{and} \quad \mathbf{v} := \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

In other words, the message $\mathsf{msg} \in \mathcal{M}$ is encrypted as $(\mathbf{c}, d) \in \mathcal{R}_q^{(2\ell+1)m} \times \mathcal{R}_q$ where

$$\mathbf{c}^{\mathrm{T}} = \mathbf{r}^{\mathrm{T}} \cdot \mathbf{A}_{\mathsf{ind}} + \mathbf{e}^{\mathrm{T}} \bmod q, \text{ and}$$
$$d = \mathbf{r}^{\mathrm{T}} \cdot \mathbf{v} + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$$

for some random LWE secret $\mathbf{r}$ and small noises $e$ and $\mathbf{e}$. In Fig. 2, we write the expressions of the ciphertext components explicitly without referring to $\mathbf{A}_{\mathsf{ind}}$ and $\mathbf{v}$.

To decrypt a ciphertext $(\mathbf{c}, d)$ encrypted with respect to index $\mathsf{ind}$ with the key $\mathbf{x}_{\mathsf{ind}}$, the decryptor first fetches its membership witness, i.e. a Merkle tree opening proof, $\mathsf{wit} = \mathbf{u}_{[\mathsf{ind}]}$ where

$$\mathbf{u}_{[\mathsf{ind}]}^{\mathrm{T}} := (\mathbf{u}_0^{\mathrm{T}}, \mathbf{u}_1^{\mathrm{T}}, \mathbf{u}_{\mathsf{ind}_1\|0}^{\mathrm{T}}, \mathbf{u}_{\mathsf{ind}_1\|1}^{\mathrm{T}}, \ldots, \mathbf{u}_{\mathsf{ind}_{1:\ell-1}\|0}^{\mathrm{T}}, \mathbf{u}_{\mathsf{ind}_{1:\ell-1}\|1}^{\mathrm{T}})$$

from the auxiliary information $\mathsf{aux}$. Together with the secret key $\mathsf{sk} = \mathbf{x}_{\mathsf{ind}}$, the relations

$$\mathbf{A}_{\mathsf{ind}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} = \mathbf{v} \bmod q \qquad \text{and} \qquad \left\| \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right\| \leq p/2$$

are satisfied. The user therefore computes $d - \mathbf{c}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \bmod q$, and round the result to the nearest multiple of $\frac{q}{2}$. The decryption is correct whenever $\left| e - \mathbf{e}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right| < (q-1)/4$.

**Analysis.** We analyze the correctness and the security of the construction in the following.

**Theorem 4.** *Let $\mathcal{R}, \ell, m, p, q, s, t$ be such that $s < t$, $\chi = \mathcal{D}_{\mathcal{R},s}$, $\bar{\chi} = \mathcal{D}_{\mathcal{R},t}$, and $q > ((2\ell+1) \cdot m \cdot \gamma_{\mathcal{R}} \cdot p + 4) \cdot \sqrt{\lambda} \cdot t + 1$. The construction in Fig. 2 is correct with overwhelming probability in $\lambda$.*

*Proof.* Observe that decryption is correct whenever $\left| e - \mathbf{e}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right| < (q-1)/4$. By Lemma 2, with overwhelming probability in $\lambda$, we have $\|e\| \leq \frac{\sqrt{\lambda}}{2} \cdot t$ and $\|\mathbf{e}\| \leq \frac{\sqrt{\lambda}}{2} \cdot s < \frac{\sqrt{\lambda}}{2} \cdot t$. Since $\begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \in \mathcal{R}_p^{(2\ell+1)m}$, we have $\left\| \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right\| \leq p/2$. Combining these facts yields

$$\left\| e - \mathbf{e}^{\mathrm{T}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right\| \leq (2\ell+1) \cdot m \cdot \gamma_{\mathcal{R}} \cdot \frac{\sqrt{\lambda}}{2} \cdot t \cdot \frac{p}{2} + \sqrt{\lambda} \cdot t < (q-1)/4$$

with overwhelming probability in $\lambda$. □

**Theorem 5.** *If $d_{\mathcal{R}} \geq \lambda$, $m \geq n \cdot \log_p q + \omega(\log \lambda)$, and the $\mathsf{LWE}_{\mathcal{R},n,q,\chi}$ assumption holds, the laconic encryption in Fig. 2 is secure. More specifically, for every PPT adversary $\mathcal{A}$ against the pseudorandom ciphertext security of the construction in Fig. 2, there exist PPT adversaries $\mathcal{A}_1$ against $\mathsf{elLWE}_{\mathcal{R},n,m,1,q,\chi,\bar{\chi},p/2}$, $\mathcal{A}_2$ against $\mathsf{LWE}_{\mathcal{R},n,2m,q,\chi}$ and $\mathcal{A}_3$ against $\mathsf{LWE}_{\mathcal{R},n,m+1,q,\chi}$ such that*

$$\mathsf{adv}(\mathcal{A}) \geq \ell \cdot \mathsf{adv}(\mathcal{A}_1) + \mathsf{adv}(\mathcal{A}_2) + \ell \cdot \mathsf{adv}(\mathcal{A}_3) + \mathsf{lhl}(\lambda)$$

*where $\mathsf{lhl}$ is the statistical distance defined by Lemma 1.*

*Proof.* Denote the construction by $\Pi$ and write $\mathcal{C} := \mathcal{R}_q^{(2\ell+1)m+1}$ for the ciphertext space. Before we discuss the hybrids, we will briefly analyze the structure of the challenge ciphertext. In the following, let $\mathsf{ind}^* = (\mathsf{ind}_1^*, \ldots, \mathsf{ind}_\ell^*)$, and let $k$ be such that $\mathsf{ind}_{1:k}^* = (\mathsf{ind}_1^*, \ldots, \mathsf{ind}_k^*)$ is a *leaf node* in the tree for which the adversary *does not* have a corresponding preimage. Furthermore, we denote $\mathbf{y}_i^* = \mathbf{y}_{\mathsf{ind}_{1:i}^*}$ at the nodes $\mathsf{ind}_1^*, \ldots, \mathsf{ind}_{1:k}^*$. In the following let $\mathsf{ctxt}^* = (\mathbf{c}_0, \ldots, \mathbf{c}_\ell, d)$ be the challenge ciphertext. Consider the following hybrids.

```
$\mathsf{Setup}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1, \mathbf{B} \leftarrow\!\!\$\, \mathcal{R}_q^{n \times m}$
$\mathbf{y}_\epsilon := \mathbf{y}^* \leftarrow\!\!\$\, \mathcal{R}_q^n$
$\mathcal{T} := \{ \epsilon \}$
$\mathsf{pp} := (\mathbf{A}_0, \mathbf{A}_1, \mathbf{B}, \mathbf{y}^*)$
$\mathsf{st} := \mathbf{y}_\epsilon$
$\mathsf{aux} := (\mathcal{T}, \{ \mathbf{y}_v \}_{v \in \mathcal{T}})$
return $(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$
```

```
$\mathsf{KGen}(\mathsf{pp})$
$\mathbf{x} \leftarrow\!\!\$\, \mathcal{R}_p^m$
$\mathbf{y} := \mathbf{B} \cdot \mathbf{x} \bmod q$
return $(\mathsf{pk}, \mathsf{sk}) := (\mathbf{y}, \mathbf{x})$
```

```
$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
if $\mathsf{pk} = \perp$ then $\mathcal{T} := \mathcal{T} \setminus \{ \mathsf{ind} \}$
else
  $\mathcal{T} := \mathcal{T} \cup \{ \mathsf{ind} \}$
  $\mathbf{y}_{\mathsf{ind}} := \mathsf{pk}$
$\mathsf{st}' \leftarrow \mathsf{TreeUpdate}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
return $\mathsf{st}'$
```

```
$\mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg})$
$\mathbf{r}_j \leftarrow\!\!\$\, \mathcal{R}_q^n, \ \forall j \in \{ 0, \ldots, \ell \}$
for $j = 0, \ldots, \ell - 1$ do
  $\mathbf{e}_j \leftarrow\!\!\$\, \chi^{2m}$
  $\mathbf{B}_j := \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ \bar{\mathsf{ind}}_{j+1} \cdot \mathbf{G} & \mathsf{ind}_{j+1} \cdot \mathbf{G} \end{pmatrix}$
  $\mathbf{c}_j^{\mathsf{T}} := (\mathbf{r}_j^{\mathsf{T}}, \mathbf{r}_{j+1}^{\mathsf{T}}) \cdot \mathbf{B}_j + \mathbf{e}_j^{\mathsf{T}} \bmod q$
$\mathbf{e}_\ell \leftarrow\!\!\$\, \chi^m, \ e \leftarrow\!\!\$\, \bar{\chi}$
$\mathbf{c}_\ell^{\mathsf{T}} := \mathbf{r}_\ell^{\mathsf{T}} \cdot \mathbf{B} + \mathbf{e}_\ell^{\mathsf{T}} \bmod q$
$d := \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_\epsilon + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$
return $\mathsf{ctxt} := (\mathbf{c}_0, \ldots, \mathbf{c}_\ell, d)$
```

```
$\mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
for $j = \ell - 1, \ldots, 0$ do
  $\mathbf{u}_{\mathsf{ind}_{1:j} \| 0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j} \| 0})$
  $\mathbf{u}_{\mathsf{ind}_{1:j} \| 1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j} \| 1})$
return $\mathsf{wit} := (\mathbf{u}_{\mathsf{ind}_{1:j} \| 0}, \mathbf{u}_{\mathsf{ind}_{1:j} \| 1})_{j=0}^{\ell-1}$
```

```
$\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt})$
parse sk as $\mathbf{x}_{\mathsf{ind}}$
$\bar{\mu} := d - \sum_{j=0}^{\ell-1} \mathbf{c}_j^{\mathsf{T}} \cdot \begin{pmatrix} \mathbf{u}_{\mathsf{ind}_{1:j} \| 0} \\ \mathbf{u}_{\mathsf{ind}_{1:j} \| 1} \end{pmatrix} - \mathbf{c}_\ell^{\mathsf{T}} \cdot \mathbf{x}_{\mathsf{ind}} \bmod q$
if $|\bar{\mu}| < q/4$ then return $0$
else return $1$
```

```
$\mathsf{TreeUpdate}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
for $j = \ell - 1, \ldots, 0$ do
  if $(\mathsf{ind}_{1:j} \| 0) \notin \mathcal{T} \wedge (\mathsf{ind}_{1:j} \| 1) \notin \mathcal{T}$ then    // Both children of $\mathsf{ind}_{1:j}$ are unassigned.
    $\mathcal{T} := \mathcal{T} \setminus \{ \mathsf{ind}_{1:j} \}$
  else
    if $(\mathsf{ind}_{1:j} \| \bar{\mathsf{ind}}_{j+1}) \notin \mathcal{T}$ then    // $\mathsf{ind}_{1:j+1}$ is assigned but its sibling not.
      $\mathbf{y}_{\mathsf{ind}_{1:j} \| \bar{\mathsf{ind}}_{j+1}} := \mathbf{y}^*$
    $\mathbf{u}_{\mathsf{ind}_{1:j} \| 0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j} \| 0}), \ \mathbf{u}_{\mathsf{ind}_{1:j} \| 1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j} \| 1})$
    $\mathcal{T} := \mathcal{T} \cup \{ \mathsf{ind}_{1:j} \}$    // Assign $\mathsf{ind}_{1:j}$ if any of its children is assigned.
    $\mathbf{y}_{\mathsf{ind}_{1:j}} := \mathbf{A}_0 \cdot \mathbf{u}_{\mathsf{ind}_{1:j} \| 0} + \mathbf{A}_1 \cdot \mathbf{u}_{\mathsf{ind}_{1:j} \| 1} \bmod q$
return st
```

**Fig. 2.** Construction of laconic encryption.

– $\mathcal{H}_0$: Identical to $\mathsf{Pseudorandomness}^0_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$. Note that in this hybrid

$$\mathbf{c}_j^{\mathsf{T}} = \mathbf{r}_j^{\mathsf{T}} \cdot (\mathbf{A}_0 \ \mathbf{A}_1) + \mathbf{r}_{i+1}^{\mathsf{T}} \cdot (\bar{\mathsf{ind}}_{i+1} \cdot \mathbf{G} \ \mathsf{ind}_{i+1} \cdot \mathbf{G}) + \mathbf{e}_j^{\mathsf{T}} \bmod q \ \forall j \in \{ 0, \ldots, \ell - 1 \},$$
$$\mathbf{c}_\ell^{\mathsf{T}} = \mathbf{r}_\ell^{\mathsf{T}} \cdot \mathbf{B} + \mathbf{e}_\ell^{\mathsf{T}} \bmod q, \text{ and}$$
$$d = \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_0^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

– $\mathcal{H}_1$: Compute $\mathsf{ctxt}^*$ as follows. Choose $\mathbf{c}_0, \ldots, \mathbf{c}_{k-1} \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ uniformly at random, choose $\mathbf{e}_0, \ldots, \mathbf{e}_{k-1} \leftarrow\!\!\$\ \chi^{2m}$, and set

$$d = \sum_{j=0}^{k-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \mathbf{r}_k^{\mathsf{T}} \cdot \mathbf{y}_k^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q,$$

where $\mathbf{z}_j = \begin{pmatrix} \mathbf{u}_{\mathsf{ind}^*_{1:j}\|0} \\ \mathbf{u}_{\mathsf{ind}^*_{1:j}\|1} \end{pmatrix} \in \mathcal{R}_p^{2m}$. Furthermore, compute $\mathbf{c}_k, \ldots, \mathbf{c}_\ell$ as in $\mathcal{H}_0$.

– $\mathcal{H}_2$: In this hybrid we choose $\mathbf{c}_k \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ and $d \leftarrow\!\!\$\ \mathcal{R}_q$ uniformly at random.

– $\mathcal{H}_3$: In this hybrid we choose $\mathbf{c}_i \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ uniformly at random for $i = k+1, \ldots, \ell-1$ and $\mathbf{c}_\ell \leftarrow\!\!\$\ \mathcal{R}_q^m$ uniformly at random.

Note that in $\mathcal{H}_3$ all ciphertext components are chosen uniformly random. Hence the claim of the theorem follows. We will establish the indistinguishability of successive hybrids via a sequence of lemmata. □

**Lemma 10.** *For any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_1$ against* $\mathsf{eILWE}_{\mathcal{R},n,m,1,q,\chi,\bar{\chi},p/2}$ *such that*

$$|\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]| \le \ell \cdot \mathsf{adv}(\mathcal{A}_1).$$

*Proof.* To show that $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally indistinguishable, we define the following sub-hybrids $\mathcal{H}'_0, \ldots, \mathcal{H}'_\ell$ and $\mathcal{H}''_0, \ldots, \mathcal{H}''_\ell$.

– $\mathcal{H}'_i$ (for $i = 0, \ldots, \ell$): $\mathcal{H}'_0$ is identical to $\mathcal{H}_0$ and hybrids $\mathcal{H}'_{>k}$ are identical to $\mathcal{H}_1$. For the middle cases, i.e. $1 \le i \le k$, we define hybrid $\mathcal{H}'_i$ so that $\mathbf{c}_0, \ldots, \mathbf{c}_{i-1}$ and $d$ are computed as in $\mathcal{H}_1$, and $\mathbf{c}_i, \ldots, \mathbf{c}_\ell$ are computed as in $\mathcal{H}_0$. Specifically, different from $\mathcal{H}_0$, we choose $\mathbf{c}_0, \ldots, \mathbf{c}_{i-1} \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ uniformly at random, choose $\mathbf{e}_0, \ldots, \mathbf{e}_{i-1} \leftarrow\!\!\$\ \chi^{2m}$ and set

$$d = \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{y}_i^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

– $\mathcal{H}''_i$ (for $i = 1, \ldots, \ell$): If $i > k$, then this hybrid is identical to $\mathcal{H}'_i$. Else ($1 \le i \le k$), $\mathbf{c}_j$ for all $j \in [0 : \ell] \setminus \{ i-1 \}$ are computed as in $\mathcal{H}'_i$, and $\mathbf{c}_{i-1}$ is computed as follows. Choose $\hat{\mathbf{c}}_{i-1}$ uniformly at random and set

$$\mathbf{c}_{i-1}^{\mathsf{T}} = \hat{\mathbf{c}}_{i-1}^{\mathsf{T}} + \mathbf{r}_i^{\mathsf{T}} \cdot (\overline{\mathsf{ind}}_i \cdot \mathbf{G}\ \mathsf{ind}_i \cdot \mathbf{G}) \bmod q.$$

Furthermore, we set

$$d = \sum_{j=0}^{i-2} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + (\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{z}_{i-1} + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

First, observe that $\mathcal{H}'_i$ and $\mathcal{H}''_i$ are in fact identically distributed: In $\mathcal{H}''_i$, since $\hat{\mathbf{c}}_{i-1}$ is uniformly and independently distributed, we can equivalently compute it as

$$\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} = \bar{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{r}_i^{\mathsf{T}} \cdot (\overline{\mathsf{ind}}_i \cdot \mathbf{G}\ \mathsf{ind}_i \cdot \mathbf{G})$$

for a uniformly random and independent $\bar{\mathbf{c}}_{i-1}$. This makes $\mathbf{c}_{i-1} = \bar{\mathbf{c}}_{i-1}$ uniformly random, as in $\mathcal{H}'_i$. Substituting the new $\hat{\mathbf{c}}_i$ to the expression of $d$ in $\mathcal{H}''_i$, we have

$$\begin{aligned} d &= \sum_{j=0}^{i-2} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + (\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{z}_{i-1} + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=0}^{i-2} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \mathbf{z}_j + (\mathbf{c}_{i-1}^{\mathsf{T}} - \mathbf{r}_i^{\mathsf{T}} (\overline{\mathsf{ind}}_i \mathbf{G}\ \mathsf{ind}_i \mathbf{G}) - \mathbf{e}_{i-1}^{\mathsf{T}}) \mathbf{z}_{i-1} + e + \left\lfloor \frac{q}{2} \right\rfloor \mathsf{msg} \bmod q \end{aligned}$$

$$= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j - \mathbf{r}_i^{\mathsf{T}} \cdot (\overline{\mathsf{ind}}_i \cdot \mathbf{G}\ \mathsf{ind}_i \cdot \mathbf{G}) \cdot \mathbf{z}_{i-1} + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$$
$$= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{y}_i^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q,$$

as in $\mathcal{H}_i'$, where the last equality was due to $(\overline{\mathsf{ind}}_i \cdot \mathbf{G}\ \mathsf{ind}_i \cdot \mathbf{G}) \cdot \mathbf{z}_{i-1} = -\mathbf{y}_i^*$.

The main technical part of this proof lies in establishing indistinguishability between hybrids $\mathcal{H}_i'$ and $\mathcal{H}_{i+1}''$ for $i \in \{0, \ldots, \ell - 1\}$. Note that the case $k < i \leq \ell - 1$ is trivial since $\mathcal{H}_i'' = \mathcal{H}_i' = \mathcal{H}_1$ for $i > k$. In the following, we focus on the remaining case $0 \leq i \leq k$. We will show that these two hybrids are indistinguishable under $\mathsf{elLWE}$. Assume towards contradiction that

$$\Pr\left[\mathcal{H}_i'(\mathcal{A}) = 1\right] - \Pr\left[\mathcal{H}_{i+1}''(\mathcal{A}) = 1\right] \geq \epsilon.$$

We will show that this implies a PPT adversary $\mathcal{A}_1'$ against $\mathsf{elLWE}$ with advantage $\epsilon$.

The adversary $\mathcal{A}_1'$ is specified as follows. As input it receives a matrix $\mathbf{A} \in \mathcal{R}_q^{n \times 2m}$, and it parses $\mathbf{A}$ as $\mathbf{A} = (\mathbf{A}_0\ \mathbf{A}_1)$ where $\mathbf{A}_0, \mathbf{A}_1 \in \mathcal{R}_q^{n \times m}$. Now $\mathcal{A}_1'$ simulates $\mathcal{H}_i'(\mathcal{A})$ with the matrices $\mathbf{A}_0, \mathbf{A}_1$ thus obtained, until the adversary $\mathcal{A}$ queries the challenge ciphertext. Now it chooses $\mathbf{z}^* = -\mathbf{z}_i$ and sends $\mathbf{z}^*$ to its challenger. Note that $\mathbf{z}^*$ is a legit query as $\|\mathbf{z}^*\| \leq p/2$. Now $\mathcal{A}_1'$ obtaining a leak $l$ and $\mathbf{y}$. Next, it computes the challenge ciphertext as in $\mathcal{H}_i'(\mathcal{A})$, except that it sets

$$\mathbf{c}_i = \mathbf{y} + \mathbf{r}_{i+1} \cdot (\overline{\mathsf{ind}}_{i+1} \cdot \mathbf{G}\ \mathsf{ind}_{i+1} \cdot \mathbf{G}) \bmod q$$

and

$$d = \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \mathbf{z}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

Note that the remaining ciphertext components are the same as in $\mathcal{H}_i'(\mathcal{A})$ and $\mathcal{H}_{i+1}''(\mathcal{A})$. From there on, $\mathcal{A}_1'$ continues the simulation of $\mathcal{H}_i'(\mathcal{A})$ and outputs whatever $\mathcal{H}_i'(\mathcal{A})$ outputs.

Now let $b \in \{0, 1\}$ be the challenge bit of the $\mathsf{elLWE}$ experiment. We claim that if $b = 0$, then $\mathcal{A}_1'$ faithfully simulates $\mathcal{H}_i'(\mathcal{A})$. On the other hand, we claim that for $b = 1$ the $\mathcal{A}_1'$ faithfully simulates $\mathcal{H}_{i+1}''(\mathcal{A})$. From these two claims it follows that $\mathcal{A}_1'$ has advantage $\epsilon$.

– For $b = 0$, it holds that $\mathbf{y}^{\mathsf{T}} = \mathbf{r}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} = \mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A}_0\ \mathbf{A}_1) + \mathbf{e}^{\mathsf{T}} \bmod q$ and $l = \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}^* + e = -\mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}_i + e \bmod q$. Renaming $\mathbf{r}$ to $\mathbf{r}_i$ and $\mathbf{e}$ to $\mathbf{e}_i$, it holds that

$$\begin{aligned} \mathbf{c}_i^{\mathsf{T}} &= \mathbf{y}^{\mathsf{T}} + \mathbf{r}_{i+1}^{\mathsf{T}} \cdot (\overline{\mathsf{ind}}_{i+1} \cdot \mathbf{G}\ \mathsf{ind}_{i+1} \cdot \mathbf{G}) \bmod q \\ &= \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{A}_0\ \mathbf{A}_1) + \mathbf{r}_{i+1}^{\mathsf{T}} \cdot (\overline{\mathsf{ind}}_{i+1} \cdot \mathbf{G}\ \mathsf{ind}_{i+1} \cdot \mathbf{G}) + \mathbf{e}_i^{\mathsf{T}} \bmod q \end{aligned}$$

and

$$d = \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \tag{10}$$
$$= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + (\mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}_i^{\mathsf{T}}) \cdot \mathbf{z}_i - \mathbf{e}_i^{\mathsf{T}} \mathbf{z}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \tag{11}$$
$$= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{y}_i^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q, \tag{12}$$

where the last equality holds as $\mathbf{y}_i^* = \mathbf{A}\mathbf{z}_i$. We can conclude that in this case the simulation of $\mathcal{A}_1'$ and $\mathcal{H}_i'(\mathcal{A})$ are identically distributed.

– For $b = 1$, it holds that $\mathbf{y} = \hat{\mathbf{c}}_i$ for a uniformly random $\hat{\mathbf{c}}_i \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ and $l = \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}^* + e = -\mathbf{e}^{\mathsf{T}}\mathbf{z}_i + e$. It therefore holds that

$$\begin{aligned}\mathbf{c}_i^{\mathsf{T}} &= \mathbf{y}^{\mathsf{T}} + \mathbf{r}_{i+1}^{\mathsf{T}} \cdot (\bar{\mathsf{ind}}_{i+1} \cdot \mathbf{G}\ \mathsf{ind}_{i+1} \cdot \mathbf{G}) \bmod q \\ &= \hat{\mathbf{c}}_i^{\mathsf{T}} + \mathbf{r}_{i+1}^{\mathsf{T}} \cdot (\bar{\mathsf{ind}}_{i+1} \cdot \mathbf{G}\ \mathsf{ind}_{i+1} \cdot \mathbf{G}) \bmod q\end{aligned}$$

and

$$\begin{aligned}d &= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \hat{\mathbf{c}}_i^{\mathsf{T}} \cdot \mathbf{z}_i - \mathbf{e}_i^{\mathsf{T}}\mathbf{z}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=0}^{i-1} (\mathbf{c}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + (\hat{\mathbf{c}}_i^{\mathsf{T}} - \mathbf{e}_i^{\mathsf{T}}) \cdot \mathbf{z}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.\end{aligned}$$

I.e. it holds that in this case the simulation of $\mathcal{A}'_1$ and $\mathcal{H}''_{i+1}(\mathcal{A})$ are identically distributed. □

**Lemma 11.** *For any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_2$ against $\mathsf{LWE}_{\mathcal{R},n,2m,q,\chi}$ such that*

$$|\Pr[\mathcal{H}_1(\mathcal{A}) = 1] - \Pr[\mathcal{H}_2(\mathcal{A}) = 1]| \le \mathsf{adv}(\mathcal{A}_2) + \mathsf{lhl}(\lambda).$$

*Proof.* In the following we describe the adversary $\mathcal{A}_2$ for the case where $k \neq \ell$, i.e., the challenge identity is not registered. For the case $k = \ell$, the argument is the same, except that we first invoke Lemma 1 to switch the matrix $\mathbf{B}$ to uniformly sampled, which introduces an additive (statistical) term $\mathsf{lhl}(\lambda)$ in the distance between the two hybrids.

$\mathcal{A}_2$ first queries $2m$ LWE samples from its oracle and arranges them in matrix form as $(\mathbf{A}, \mathbf{v})$, this $\mathbf{A}$ is then parsed as $\mathbf{A} = (\mathbf{A}_0\ \mathbf{A}_1)$ and uses $\mathbf{A}_0$ and $\mathbf{A}_1$ in $\mathsf{pp}$, whereas the vector $\mathbf{v}$ is stored. Next, $\mathcal{A}_2$ queries $m$ LWE samples from its oracle and arranges them in matrix form as $(\mathbf{B}, \mathbf{v}')$, this $\mathbf{B}$ is then used as part of $\mathsf{pp}$. Now $\mathcal{A}_2$ simulates $\mathcal{H}_1$, but whenever a new honest key $\mathsf{pk}_i^*$ is generated, $\mathcal{A}_2$ queries its LWE oracle and obtains $(\hat{\mathbf{y}}_i, \hat{v}_i)$, sets $\mathsf{pk}_i = \hat{\mathbf{y}}_i$ and stores $\hat{v}_i$. The challenge ciphertext is generated as follows: Assume the challenge identity $\mathsf{ind}^*$ terminates in a public key $\mathsf{pk}_{i^*}$. The challenge ciphertext is computed as in $\mathcal{H}_2(\mathcal{A})$, except that we set

$$d = \sum_{j=0}^{k-1} (\mathbf{u}_j - \mathbf{e}_j)^{\mathsf{T}} \cdot \mathbf{z}_j + \hat{v}_{i^*} + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$$

and $\mathbf{c}_k^{\mathsf{T}} = \mathbf{v}$ if $k < \ell$ and $\mathbf{c}_k^{\mathsf{T}} = \mathbf{v}'$ if $k = \ell$. $\mathcal{A}_2$ then continues simulation of $\mathcal{H}_2(\mathcal{A})$ and outputs whatever $\mathcal{H}_2(\mathcal{A})$ outputs.

First observe that if $(\mathbf{A}, \mathbf{v})$, $(\mathbf{B}, \mathbf{v}')$ and $\{(\hat{\mathbf{y}}_{i^*}, \hat{v}_{i^*})\}$ are LWE samples, i.e. $\mathbf{v} = \mathbf{s}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q$, $(\mathbf{v}')^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}} \cdot \mathbf{B} + \hat{\mathbf{e}}^{\mathsf{T}} \bmod q$ and $\hat{v}_{i^*} = \mathbf{s}^{\mathsf{T}} \cdot \hat{\mathbf{y}}_{i^*} + \hat{e}_{i^*} \bmod q$, then the simulation of $\mathcal{A}_2$ is identically distributed to $\mathcal{H}_2(\mathcal{A})$. On the other hand, if $\mathbf{c}$, $\mathbf{v}'$ and the $\hat{v}_{i^*}$ are uniformly random and independent, then the simulation of $\mathcal{A}_2$ is identically distributed to $\mathcal{H}_3(\mathcal{A})$. The claim of the lemma follows. □

**Lemma 12.** *For any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_3$ against $\mathsf{LWE}_{\mathcal{R},n,m+1,q,\chi}$ such that*

$$|\Pr[\mathcal{H}_2(\mathcal{A}) = 1] - \Pr[\mathcal{H}_3(\mathcal{A}) = 1]| \le \ell \cdot \mathsf{adv}(\mathcal{A}_3).$$

*Proof.* Consider the following hybrids $\mathcal{H}'''_0, \dots, \mathcal{H}'''_\ell$, where $\mathcal{H}'''_0$ is identically distributed to $\mathcal{H}_2$, and $\mathcal{H}'''_\ell$ is identically distributed to $\mathcal{H}_3$.

– $\mathcal{H}'''_{i+1}$ (For $i = 0, \dots, \ell - 1$): Identically distributed to $\mathcal{H}_i$, except that, for $i > k$, $\mathbf{c}_i$ is chosen uniformly at random.

Assume towards contradiction that

$$\mathsf{Pr}\left[\mathcal{H}'''_{i+1}(\mathcal{A}) = 1\right] - \mathsf{Pr}\left[\mathcal{H}'''_{i}(\mathcal{A}) = 1\right] \geq \epsilon$$

for some $i \in \{0, \ldots, \ell\}$. We will show that this implies a PPT adversary $\mathcal{A}_3$ against $\mathsf{LWE}_{\mathcal{R},n,m,q,\chi}$ with advantage $\epsilon$. The adversary $\mathcal{A}_3$ receives an input $(\mathbf{A}, \mathbf{v})$ and proceeds as follows. $\mathcal{A}_3$ simulates $\mathcal{H}'''_i(\mathcal{A})$, except for the following modifications. First, it uses the matrix $\mathbf{A} = (\mathbf{A}_0\ \mathbf{A}_1)$ in the public parameters. Next, when the challenge ciphertext is generated, if $i > k$ it sets $\mathbf{c}_i = \mathbf{v}$. $\mathcal{A}_3$ then continues the simulation and outputs whatever its simulation of $\mathcal{H}'''_{i-1}(\mathcal{A})$ outputs.

Now, it follows routinely that if $(\mathbf{A}, \mathbf{v})$ is an LWE sample, i.e. $\mathbf{v}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}} \cdot \mathbf{A} + \mathbf{e}^{\mathsf{T}} \bmod q$, then the simulation of $\mathcal{A}_3$ is distributed identically to $\mathcal{H}'''_i(\mathcal{A})$. On the other hand, if $\mathbf{v}$ is uniformly random, then the simulation of $\mathcal{A}_3$ is distributed identically to $\mathcal{H}'''_{i+1}(\mathcal{A})$. We can conclude that $\mathcal{A}_3$ has advantage $\epsilon$. □

**Update Privacy.** There is a simple modification to construction of laconic encryption in Fig. 2 which yields update-privacy. The idea is to make the hash-function $f_{\mathbf{A}_0,\mathbf{A}_1}(\mathbf{y}_0, \mathbf{y}_1) = \mathbf{A}_0 \cdot (-\mathbf{G}^{-1}(\mathbf{y}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}^{-1}(\mathbf{y}_1))$ randomized such that $f_{\mathbf{A}_0,\mathbf{A}_1}(\mathbf{y}_0, \mathbf{y}_1)$ *statistically hides* $\mathbf{y}_0$ and $\mathbf{y}_1$. This can be achieved by slightly modifying the gadget matrix $\mathbf{G}$ into $\mathbf{G}'$ and making $\mathbf{G}'^{-1}$ randomized [8], and replacing the parameter $m$ with a slightly larger $m' = m + n\log(q)$. Specifically, we set $\mathbf{G}' = (\mathbf{G}\ \mathbf{0}) \in \mathbb{Z}^{n \times m'}$, i.e. we obtain $\mathbf{G}'$ by appending $n\log(q)$ all-zero columns to $\mathbf{G}$. Furthermore, we define $\mathbf{G}'^{-1}(\mathbf{x}) = \begin{pmatrix} \mathbf{G}^{-1}(\mathbf{x}) \\ \mathbf{r} \end{pmatrix}$, where $\mathbf{r} \leftarrow\!\!\$\ \mathcal{R}_2^{2n\log(q)}$ is chosen uniformly at random. Note that it still holds that $\mathbf{G}'\mathbf{G}'^{-1}(\mathbf{x}) = \mathbf{x}$ for all $\mathbf{x} \in \mathcal{R}_q^n$. The modified hash function is now

$$f'(\mathbf{y}_0, \mathbf{y}_1) = \mathbf{A}_0 \cdot (-\mathbf{G}'^{-1}(\mathbf{y}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}'^{-1}(\mathbf{y}_1)).$$

Now, decomposing $\mathbf{A}_0 = (\mathbf{A}_{0,1}\ \mathbf{A}_{0,2})$ and $\mathbf{A}_1 = (\mathbf{A}_{1,1}\ \mathbf{A}_{1,2})$, where $\mathbf{A}_{0,1}, \mathbf{A}_{1,1} \in \mathcal{R}_q^{n \times m}$ and $\mathbf{A}_{0,2}, \mathbf{A}_{1,2} \in \mathcal{R}_q^{n \times n\log(q)}$, it holds that

$$\begin{aligned} f'(\mathbf{y}_0, \mathbf{y}_1) &= \mathbf{A}_0 \cdot (-\mathbf{G}'^{-1}(\mathbf{y}_0)) + \mathbf{A}_1 \cdot (-\mathbf{G}'^{-1}(\mathbf{y}_1)) \\ &= (\mathbf{A}_{0,1}\ \mathbf{A}_{0,2}) \cdot (-\begin{pmatrix} \mathbf{G}^{-1}(\mathbf{y_0}) \\ \mathbf{r}_0 \end{pmatrix}) + (\mathbf{A}_{1,1}\ \mathbf{A}_{1,2}) \cdot (-\begin{pmatrix} \mathbf{G}^{-1}(\mathbf{y}_1) \\ \mathbf{r}_1 \end{pmatrix}) \\ &= (\mathbf{A}_{0,1}\ \mathbf{A}_{1,1}) \begin{pmatrix} -\mathbf{G}^{-1}(\mathbf{y_0}) \\ -\mathbf{G}^{-1}(\mathbf{y_1}) \end{pmatrix} - \underbrace{(\mathbf{A}_{0,2}\ \mathbf{A}_{1,2}) \begin{pmatrix} \mathbf{r}_0 \\ \mathbf{r}_1 \end{pmatrix}}_{=:\mathbf{v}}. \end{aligned}$$

Since the matrix $(\mathbf{A}_{0,2}\ \mathbf{A}_{1,2}) \in \mathcal{R}_q^{n \times 2n\log(q)}$ is chosen uniformly random and $(\mathbf{r}_0, \mathbf{r}_1)$ is uniformly random in $\mathcal{R}_2^{2n\log(q)}$, it holds by the leftover hash lemma (Lemma 1) that $\mathbf{v}$ is $2^{-n}$-close to uniform. Consequently, the hash-value $f'(\mathbf{y}_0, \mathbf{y}_1)$ is statistically close to uniform. Hence, update privacy of the modified construction follows. We can conclude the following lemma.

**Lemma 13.** *The modified construction of Fig. 2 using* $\mathbf{G}'$ *and the randomized* $\mathbf{G}'^{-1}$ *is update private.*

**Properties and Efficiency.** We remark about some properties and efficiency of our construction. The public parameters (initially) consists of three uniformly random matrices $\mathbf{A}_0, \mathbf{A}_1, \mathbf{B} \in \mathcal{R}_q^{n \times m}$ which can be sampled with public coin. Subsequent updates to the public parameters, more specifically to $\mathbf{y}_\epsilon \in \mathcal{R}_q^n$, are deterministic. Suppose we pick $q$ to be linear in $\ell$, then the $\mathsf{Setup}$ and $\mathsf{KGen}$ algorithms run in time logarithmic in $\ell$, while the $\mathsf{Upd}$, $\mathsf{Enc}$, $\mathsf{WGen}$, and $\mathsf{Dec}$ algorithms run in time quasi-linear in $\ell$.

[8][BdMW16] defined a similar notion of *randomized* $\mathbf{G}^{-1}$*, which however samples a discrete gaussian preimage*

## 8 Registration-Based Encryption

In Registration-Based Encryption (RBE) [GHMR18] users, characterized by an identity $\mathsf{ind}$, generate a pair of public-secret keys $(\mathsf{pk}, \mathsf{sk})$ and register $\mathsf{pk}$, upon entering the system, to an (untrusted) Key Curator (KC). KC is then generating a short public information $\mathsf{st}$ and a (public) witness for the user $\mathsf{wit}$. Given the identity of the user $\mathsf{ind}$, the public state $\mathsf{st}$ (and possibly some universal public parameters $\mathsf{pp}$), but without the corresponding public key, anyone can encrypt a meesage for the user. The user can then decrypt using her secret key $\mathsf{sk}$ and witness $\mathsf{wit}$. This witness is updated during the lifetime of the system, upon new user registrations.

As noted in Section 7, RBE is essentially equivalent to (static) Laconic Encryption (LE) with an additional efficiency requirement: the total number of witness updates any user should ask is at most poly-logarithmic in the number of users. Although the two notions are conceptually close, this distinguishing feature turns out to be cumbersome enough to differentiate the two primitives.

In more detail, RBE is (1) static, in the sense that an identity-user can register her public key once but cannot replace or delete it. Therefore, if $N$ number of users participate in the RBE, exactly $N$ public parameter updates happen. Additionally, the crucial efficiency property of RBE is that throughout these $N$ registrations, each user's witness $\mathsf{wit}$ is updated at most $\mathsf{poly}(\lambda, \log N)$ times. This means that any user $\mathsf{ind}$ needs to call $\mathsf{WGen}^{\mathsf{aux}}(\cdot, \mathsf{ind}, \cdot)$ at most $\mathsf{poly}(\lambda, \log N)$ times. In recent work, [MQR22] showed that if we keep the public parameters of length $\mathsf{poly}(\log N)$ in any RBE construction, $\Omega(\log N/\log\log N)$ calls to $\mathsf{WGen}^{\mathsf{aux}}$ are necessary so as to satisfy the RBE properties. This essentially minimizes the communication between a user with the Key Curator.

In a Laconic Encryption scheme, $N$ public parameter updates (i.e. $\mathsf{Upd}^{\mathsf{aux}}$ calls) can result to $N$ witness updates. As a matter of fact, our construction of Section 7.2 requires $N$ witness updates: at each new registration all witnesses should be updated.

Below we formally define RBE assuming LE. The definition is equivalent to the one originally introduced by Garg et. al. [GHMR18], restated through the lens of laconic encryption.

**Definition 11 (Registration-based Encryption).** *A registration-based scheme is a laconic encryption scheme, consisting of the PPT algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{WGen}, \mathsf{Dec})$*, with the following additional property:*

- ***Efficiency of the number of updates**. For any identity* $\mathsf{ind} \in \{0,1\}^{\ell}$*, the total number of* $\mathsf{WGen}^{\mathsf{aux}}(\cdot, \mathsf{ind}, \cdot)$ *invocations is upper bounded by* $\mathsf{poly}(\lambda, \ell)$*.*

Furthermore, we recall the definition of weakly-efficient RBE [GHMR18], which relaxes the efficiency requirements of RBE. Essentially, it allows the Key Curator to run in time linear in the number of registred users, instead of poly-logarithmic.

**Definition 12 (Weakly-Efficient Registration-based Encryption).** *A weakly-efficient registration-based scheme is a registration-based encryption scheme where the running time of* $\mathsf{Upd}$ *is bound by* $\mathsf{poly}(\lambda, N)$*, for* $N$ *the total number of users registered.*

### 8.1 Our Construction

Here we present our (weakly-efficient) Registration-Based Encryption scheme. In fact, we give a generic construction of weakly-efficient RBE from any Laconic Encryption scheme.

Assume any Laconic Encryption scheme $(\mathsf{LE.Setup}, \mathsf{LE.KGen}, \mathsf{LE.Upd}, \mathsf{LE.Enc}, \mathsf{LE.WGen}, \mathsf{LE.Dec})$. Our end-goal is to prevent the number of updates from being linear in the number of registration. In other words, not every $\mathsf{wit}$ should become invalid (and thus need an update) after new registration. In LE the short state $\mathsf{st}$ "accumulates" all the $N$ current set of keys. In our transformation instead, we generate $\ell$ different states $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_\ell)$ that 'accumulate' $N_1 > N_2 > \ldots > N_\ell$ keys resp., where $N_i$ can be either a power-of-2 or 0.

A new $\mathsf{pk}$ is at first put in an empty state with $N_k = 0$, then as long as $N_i = N_{i-1}$ the two states are merged: all the public keys of $N_i$ are removed from $\mathsf{st}_i$ and moved to $\mathsf{st}_{i-1}$, which gives as $N'_{i-1} \leftarrow 2N_{i-1}$ and $N'_i \leftarrow 0$. At this point all the witnesses of the corresponding identities in $\mathsf{st}_{i-1}$ should be updated. However this means that the only way a witness is updated is when it moves to the next power-of-2 state, i.e. with double the number of identities. This can happen at most $\log N$ number of times. On the other hand, merge may, in the worst case, require $O(N)$ time, which gives us a weakly-efficient RBE.

The above transformation generalizes the underlying data structure of the Garg et. al. [GHMR18] construction.

On another note, in LE we (implicitly) assume that after each witness update the owner is receiving the new witness. In RBE, since updates sparsely happen, we desire the user to identify when the update happened so that it communicates with KC to receive the new witness. For this each ciphertext should include some information from which the identity can reason whether the current witness is valid or it has to be updated. For this, in addition to the encryption of the message $\mathsf{msg}$, we plug into the ciphertext a commitment to $\mathsf{msg}$ with randomness $\mathsf{r}$, $c = \mathsf{com}(\mathsf{msg}; \mathsf{r})$, using any commitment scheme $\Gamma$, and an encryption of $\mathsf{r}$, $\mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{r})$.[9] If $\mathsf{msg}; \mathsf{r}$ is an opening of $c$ then the decryption is correct and the witness is valid, otherwise the decryption gives different $\mathsf{msg}', \mathsf{r}'$ that do not open $c$ and the witness should be updated.

In the following construction we assume that the auxiliary information ($\mathsf{aux}$) in the Laconic Encryption Scheme at the very least contains:

(i) The set of identities registered, denoted $\mathcal{I}$.
(ii) The corresponding public keys of the identities registered, denoted $\mathcal{PK}$.
(iii) The number of identities currently registered, denoted $N$.[10]

We assume RAM access to the above information of $\mathsf{aux}$ and that they are updated accordingly with every $\mathsf{LE.Upd}^{\mathsf{aux}}$ call. Additionally, we remark that depending on the LE construction, $\mathsf{aux}$ may contain further information.

Our weakly-efficient RBE generic construction is in Figure 3.

*Remark 1 (Generic $O(\sqrt{N})$-efficient RBE transformation).* An alternative generic transformation is to consider $O(\sqrt{N})$ states $\mathsf{st} = (\mathsf{st}_1, \ldots, \mathsf{st}_{\sqrt{N}})$ instead of $\log N$, where each state 'accumulates' up to $\sqrt{N}$ identities (and no merging ever occurs). This gives an RBE where the size of the public information $\|\mathsf{st}\|$ and the maximum number of updates are $O(\sqrt{N})$ instead of $\log N$, but the running time of $\mathsf{Upd}$ is $O(\sqrt{N})$ instead of $O(N)$. This transformation and the transformation of Figure 3 provide different tradeoffs for the user's and KC's overhead.

**Correctness and Security.** Correctness follows from correctness of the laconic encryption scheme and the commitment scheme.

Security of the scheme comes from the security of the laconic encryption and the hiding property of the commitment scheme. As discussed above, the definition of security for RBE is the same as that of LE. Therefore we can argue indistinguishability for each component $\mathsf{ctxt}_i^{(1)}, \mathsf{ctxt}_i^{(2)}$ of the RBE ciphertext using the security of $\mathsf{LE}$. Then for each $c_i$ component we use the hiding property of $\Gamma$. This is formally described below with a sequence of hybrid games.

**Theorem 6.** *If $\mathsf{LE}$ is a secure laconic encryption scheme and $\Gamma$ a hiding commitment scheme, the registration-based encryption in Fig. 3 is secure.*

[9]For convinience we assume that the randomness space of the commitment scheme and the message of the LE scheme coincide. In the opposite case one can always encode $\mathsf{r}$ in the message space of LE with possibly many messages (e.g. bit decomposition) and encrypt them message-by-message $(\mathsf{Enc}(\mathsf{Encode}_1(\mathsf{r})), \ldots, \mathsf{Enc}(\mathsf{Encode}_k(\mathsf{r})))$.

[10]The number of identities $N$ can be infered from the set of identities $\mathcal{I}$. For ease of presentation we consider $N$ explicitly stored.

```
$\mathsf{Setup}(1^\lambda, 1^\ell)$
$(\mathsf{pp}, \mathsf{st}_0, \mathsf{aux}_0) \leftarrow \mathsf{LE.Setup}(1^\lambda, 1^\ell)$
$\mathsf{st} := (\mathsf{st}_0, \dots, \mathsf{st}_0)$   // $\ell$-sized vector.
$\mathsf{aux} := (\mathsf{aux}_0, \dots, \mathsf{aux}_0)$   // $\ell$-sized vector.
return $(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$

$\mathsf{KGen}(\mathsf{pp})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LE.KGen}(\mathsf{pp})$
return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
find the smallest $i \in [\ell] : N_i = 0$   // Using $\mathsf{aux}$.
parse $\mathsf{st} := (\mathsf{st}_1, \dots, \mathsf{st}_\ell)$
$\mathsf{st}'_i \leftarrow \mathsf{LE.Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \mathsf{pk})$
$(\mathsf{st}'_1, \dots, \mathsf{st}'_{i-1}) := (\mathsf{st}_1, \dots, \mathsf{st}_{i-1})$   // Initilization.
while $N_i = N_{i-1} \land i > 1$ do
    $(\mathsf{st}'_i, \mathsf{st}'_{i-1}) \leftarrow \mathsf{Merge}^{\mathsf{aux}}(\mathsf{st}'_i, \mathsf{st}'_{i-1})$
    $i := i - 1$
return $\mathsf{st}' := (\mathsf{st}'_1, \dots, \mathsf{st}'_i, \mathsf{st}_{i+1}, \dots, \mathsf{st}_\ell)$

$\mathsf{Merge}^{\mathsf{aux}}(\mathsf{st}_i, \mathsf{st}_{i-1})$
foreach $\mathsf{ind} \in \mathcal{I}_i$ do
    $\mathsf{st}'_{i-1} \leftarrow \mathsf{LE.Upd}^{\mathsf{aux}_{i-1}}(\mathsf{pp}, \mathsf{st}_{i-1}, \mathsf{ind}, \mathsf{pk}_{\mathsf{ind}})$
    $\mathsf{st}'_i \leftarrow \mathsf{LE.Upd}^{\mathsf{aux}_i}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \bot)$
return $(\mathsf{st}'_i, \mathsf{st}'_{i-1})$

$\mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg})$
parse $\mathsf{st} := (\mathsf{st}_1, \dots, \mathsf{st}_\ell)$
for $j = 1, \dots, \ell$ do
    $\mathsf{r}_i \leftarrow\$ \mathcal{R}_\Gamma$
    $\mathsf{ctxt}_i^{(1)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \mathsf{msg})$
    $\mathsf{ctxt}_i^{(2)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \mathsf{r}_i)$
    $c_i \leftarrow \mathsf{com}(\mathsf{msg}; \mathsf{r}_i)$
    $\mathsf{ctxt}_i \leftarrow (\mathsf{ctxt}_i^{(1)}, \mathsf{ctxt}_i^{(2)}, c_i)$
return $\mathsf{ctxt} := (\mathsf{ctxt}_1, \dots, \mathsf{ctxt}_\ell)$

$\mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
find $i \in [\ell] : \mathsf{ind} \in \mathcal{I}_i$   // Using $\mathsf{aux}$.
parse $\mathsf{st} := (\mathsf{st}_1, \dots, \mathsf{st}_\ell)$
$\mathsf{wit}_i \leftarrow \mathsf{LE.WGen}^{\mathsf{aux}_i}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \mathsf{pk})$
return $\mathsf{wit} := (\mathsf{wit}_i, i)$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt})$
parse $\mathsf{st} := (\mathsf{st}_1, \dots, \mathsf{st}_\ell)$
parse $\mathsf{wit} := (\mathsf{wit}_i, i)$
parse $\mathsf{ctxt} := (\mathsf{ctxt}_1, \dots, \mathsf{ctxt}_\ell)$
parse $\mathsf{ctxt}_i := (\mathsf{ctxt}_i^{(1)}, \mathsf{ctxt}_i^{(2)}, c)$
$\mathsf{msg} \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_i, \mathsf{wit}_i, \mathsf{ctxt}_i^{(1)})$
$\mathsf{r} \leftarrow \mathsf{LE.Dec}(\mathsf{sk}_i, \mathsf{wit}_i, \mathsf{ctxt}_i^{(2)})$
if $c \neq \Gamma.\mathsf{com}(\mathsf{msg}; \mathsf{r})$ then return $\mathsf{GetUpd}$
else return $\mathsf{msg}$
```

**Fig. 3.** Our generic construction of registration-based encryption from any laconic encryption scheme.

*Proof.* Let the security experiment $\mathsf{Security}^b_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$ of Figure 1 for a uniformly random $b \leftarrow\$ \{0,1\}$. The ciphertext has the form:

$$\mathsf{ctxt} = \Big((\mathsf{ctxt}_1^{(1)}, \mathsf{ctxt}_1^{(2)}, c_1), \ldots, (\mathsf{ctxt}_\ell^{(1)}, \mathsf{ctxt}_\ell^{(2)}, c_\ell)\Big)$$

where $\mathsf{ctxt}_i^{(1)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}_b, \mathsf{msg}_b)$, $\mathsf{ctxt}_i^{(2)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}_b, \mathsf{r}_i)$, $c_i \leftarrow \Gamma.\mathsf{com}(\mathsf{msg}_b; \mathsf{r}_i)$.

We define the following sequence of hybrid games:

- Hybrid 0: is identical to the orginal security game.
- Hybrid 1: is the same as Hybrid 0, except $c_1$ is computed as $\Gamma.\mathsf{com}(\tilde{\mathsf{msg}}; \tilde{\mathsf{r}})$ for an arbitrary message: $\tilde{\mathsf{msg}} \in \mathcal{M}_\Gamma$ and $\tilde{\mathsf{r}} \leftarrow\$ \mathcal{R}_\Gamma$. Hybrid 0 and Hybrid 1 are indistinguishable according to the hiding property of the commitments scheme $\Gamma$.
- . . .
- Hybrid $\ell$: similarly it is the same as Hybrid $\ell - 1$ except for $c_\ell$ which is computed as a commitment to an arbitrary message from the message space of $\Gamma$, $\Gamma.\mathsf{com}(\tilde{\mathsf{msg}}; \tilde{\mathsf{r}})$. From the hiding property of $\Gamma$ Hybrid $\ell$ is indistinguishable from Hybrid $\ell - 1$.
- Hybrid $\ell + 1$: it is the same as Hybrid $\ell$ except $\mathsf{ctxt}_1^{(1)}$ is replaced with an encryption of an arbitrary element $\hat{\mathsf{msg}}$ and identity $\hat{\mathsf{ind}}_1$, $\mathsf{ctxt}_1^{(1)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{msg}})$. Hybrid $\ell + 1$ is indistinguishable from Hybrid $\ell$ from the security of $\mathsf{LE}$: an adversary that is able to distinguish between the two games can be used to break the security of the laconic encryption scheme (for $\mathsf{st}_1$ and $\mathsf{ctxt}_1^{(1)}$).
- . . .
- Hybrid $2\ell$: is the same as the Hybrid $2\ell - 1$ except $\mathsf{ctxt}_\ell^{(1)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{msg}})$. The two Hybrids are indistinguishable from $\mathsf{LE}$ security.
- Hybrid $2\ell + 1$: is the same as the Hybrid $2\ell$ except $\mathsf{ctxt}_1^{(2)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{r}})$. The two Hybrids are indistinguishable from $\mathsf{LE}$ security.
- . . .
- Hybrid $3\ell$: is the same as the Hybrid $3\ell - 1$ except $\mathsf{ctxt}_\ell^{(2)} \leftarrow \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{r}})$. The two Hybrids are indistinguishable from $\mathsf{LE}$ security.

  The final ciphertext has the form:

$$\mathsf{ctxt}^* = \Big((\mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{msg}}), \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_1, \hat{\mathsf{ind}}, \hat{\mathsf{r}}), \Gamma.\mathsf{com}(\tilde{\mathsf{msg}}; \tilde{\mathsf{r}})), \ldots,$$
$$(\mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_\ell, \hat{\mathsf{ind}}, \hat{\mathsf{msg}}), \mathsf{LE.Enc}(\mathsf{pp}, \mathsf{st}_\ell, \hat{\mathsf{ind}}, \hat{\mathsf{r}}), \Gamma.\mathsf{com}(\tilde{\mathsf{msg}}; \tilde{\mathsf{r}}))\Big)$$

  for arbitrary $\hat{\mathsf{ind}}, \hat{\mathsf{msg}}, \tilde{\mathsf{msg}}.\tilde{\mathsf{r}}$ (independent of the choice of $\mathcal{A}$'s $\mathsf{ind}_0, \mathsf{msg}_0, \mathsf{ind}_1, \mathsf{msg}_1$) so the adversary can successfully guess $b$ with probability 1/2, $\Pr[\text{Hybrid } 3\ell = 0] = 1/2$.

  Therefore:

$$\Pr[\text{Hybrid } 0 = 0] = \frac{1}{2} + \ell \cdot \mathsf{adv}_{\mathsf{hid}}(\mathcal{A}) + 2\ell \cdot \mathsf{adv}_{\mathsf{LEsec}}(\mathcal{A}) = \frac{1}{2} + \mathsf{negl}(\lambda)$$

□

**Instantiation and Efficiency.** The above RBE construction can be instantiated with our laconic encryption scheme of section 7.2 and any (bit-)commitment scheme, the simpllest being the one based on SIS [Ajt96].[11] Here, we discuss the (asymptotic) efficiency of this concrete instantiation. We note that in all cryptographic operations and element sizes below there is an inherent $O(\lambda)$ factor omitted.

The running time of the $\mathsf{Setup}$ algorithm is $O(\ell)$ (it generates one $\mathsf{LE}$ key in $O(1)$ time and outputs two $O(\ell)$-vectors). $\mathsf{KGen}$ runs in $O(1)$ time. For the registration, the Key Curator should run $\mathsf{Upd}$ which takes, in the worst case, $O(N)$ steps, where $N$ is the number of curently registered identities. For this our RBE scheme is weakly efficient. Encrypting and decrypting a ciphertext takes $O(\ell^2)$ time, each of the $\ell$ components of

[11] Alternative (lattice-based) commitment schemes can be considered, with the goal of improving concrete efficiency, without though changing the asymptotic behaviour of the RBE scheme.

the ciphertext takes $O(\ell)$ time to compute according to $\mathsf{LE.Enc}$ and $\mathsf{LE.Dec}$ respectively. Finally, witness generation takes $O(\ell)$ time.

For the communication complexity, $\|\mathsf{pp}\| = O(1)$, $\|\mathsf{st}\| = O(\ell)$, $\|\mathsf{sk}\|, \|\mathsf{pk}\| = O(1)$. The size of of the auxiliary information is $\|\mathsf{aux}\| = O(N)$. Then the ciphertext has size $\|\mathsf{ctxt}\| = O(\ell^2)$ and the witness $\|\mathsf{wit}\| = O(\ell)$.

Finally, as discussed above the maximum number of update queries a user should make to the Key Curator is exactly $\log N$.

We stress that the asymptotic behaviour of our RBE scheme matches exactly the one by Garg et. al. [GHMR18]. However, our construction does not make use of expensive non-black-box cryptographic primitives such as garbled circuits or indistinguishable obfuscation. All the operations are purely algebraic. This greatly improves the concrete efficiency of our scheme.

### 8.2 Efficient Verifiable Registration-Based Encryption

Here we discuss how to achieve verifiability in our (concrete) RBE scheme. Verifiable RBE was introduced by Goyal et. al. [GV20] and states that the behaviour of the Key Curator shall be verifiable. That is, anyone should be able to ask the Key Curator if an identity $\mathsf{ind}$ is registered or not and the KC should be able to answer with a short (non-)membership proof for $\mathsf{ind}$.

Our concrete RBE scheme (instantiated with our laconic encryption of 7.2 and the SIS-based commitment scheme [Ajt96]) is verifiable in a straightforward way. The data structure accumulating users' public keys are $\ell$ Merkle-trees, where the $i$-th position (the $i$-th leave) is reserved for the public key of the identity $\mathsf{ind} = i$. If the user $i$ is registered then exactly one of the $\ell$ Merkle trees should contain $\mathsf{pk}_i$ and the rest should be unassigned, in position $i$. If the user is not registered then all $\ell$ Merkle trees should be null in position $i$.

More specifically, an RBE membership proof for $(\mathsf{ind}, \mathsf{pk})$ should be $\ell - 1$ non-membership proofs and exactly 1 membership proof for position $\mathsf{ind}$. Similarly, an RBE non-membership proof for $\mathsf{ind}$ should be exactly $\ell$ non-membership proofs for the position $\mathsf{ind}$. For our construction, a Merkle-tree membership proof is simply the path (and the siblings) from leaf $\mathsf{ind}$ up to the root. For the Merkle-tree non-membership proof we exploit the fact that if the leaf $\mathsf{ind}$ is unassigned there should be (exactly) one terminator value $\mathbf{y}^*$ in the (hypothetical) path from leaf $\mathsf{ind}$ up to the root. That is one of $\mathbf{u}_{\mathsf{ind}_{1:1}}, \mathbf{u}_{\mathsf{ind}_{1:2}}, \ldots, \mathbf{u}_{\mathsf{ind}_{1:\ell}}$ should be $\mathbf{y}^*$. Therefore the non-membership proof consists of the path (and its siblings) from the terminator node in the path of $\mathsf{ind}$ up to the root.

We formally describe $(\mathsf{MProve}, \mathsf{MVerify}, \mathsf{NMProve}, \mathsf{NMVerify})$, the membership and non-membership algorithms for verifiability, in Figure 4.

## 9 Laconic Private-Set Intersection

In a private set intersection (PSI) protocol [FNP04], two parties holding private sets $X$ and $Y$ can jointly compute the intersection $X \cap Y$ without revealing any other information about their sets to each other. There are several laconic PSI protocols constructed from accumulators [ABD+21,ALOS22]. We now present a laconic PSI protocol from our laconic encryption scheme $\Pi$.

– $\mathsf{PSI.Setup}$: On input the security parameter $\lambda$, the setup algorithm runs the setup of laconic encryption $(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda)$ and generates a collision-resistant hash function $\mathsf{H} : \{0,1\}^* \mapsto \{0,1\}^\ell$.
– $\mathsf{PSI.R}_1$: On input a non-empty set $S_\mathsf{R} = \{x_i\}_{i\in[m]}$, the receiver algorithm proceeds with the following subroutine. For $i \in [m]$:
  - Sample a pair of keys $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KGen}(\mathsf{pp})$.
  - Insert $\mathsf{pk}_i$ at position $\mathsf{H}(x_i)$ computing $\mathsf{st} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{H}(x_i), \mathsf{pk}_i)$ and generate the corresponding witness $\mathsf{wit}_i \leftarrow \mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{H}(x_i), \mathsf{pk}_i)$.

  It samples a message $\mathsf{msg} \leftarrow \mathcal{R}_2$. Finally, it sends the updated $\mathsf{st}$ and $\mathsf{msg}$ to the sender.
– $\mathsf{PSI.S}$: On input $\mathsf{st}$, $\mathsf{msg}$ and a set $S_\mathsf{S} = \{y_j\}_{j\in[n]}$, the sender samples an one-way permutation $\pi$ on $[n]$ and encrypts $\mathsf{msg}$ with the updated $\mathsf{st}$: $\mathsf{ctxt}_j \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{H}(y_{\pi(j)}), \mathsf{msg})$ and $j \in [n]$, and sends the set of ciphertexts $\{\mathsf{ctxt}_j\}_{j\in[n]}$ to the receiver.

```
$\mathsf{MProve}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
$(\mathsf{wit}_i, i) \leftarrow \mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
for $j = 1, \ldots, \ell, j \neq i$ do
    $\tilde{\mathsf{wit}}_j \leftarrow \tilde{\mathsf{W}}\mathsf{Gen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
return
$\pi := (\tilde{\mathsf{wit}}_1, \ldots, \tilde{\mathsf{wit}}_{i-1}, \mathsf{wit}_i, \tilde{\mathsf{wit}}_{i+1}, \ldots, \tilde{\mathsf{wit}}_\ell, i)$

$\mathsf{MVerify}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk}, \pi)$
if $\mathsf{WVer}(\mathsf{pp}, \mathsf{st}_i, \mathsf{ind}, \mathsf{wit}_i, \mathsf{pk}) \neq 1$ then
    return $0$
for $j = 1, \ldots, \ell, j \neq i$ do
    if $\mathsf{WVer}(\mathsf{pp}, \mathsf{st}_j, \mathsf{ind}, \tilde{\mathsf{wit}}_j, \mathbf{y}^*) \neq 1$ then
        return $0$
return $1$

$\mathsf{NMProve}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
for $j = 1, \ldots, \ell$ do
    $\tilde{\mathsf{wit}}_j \leftarrow \tilde{\mathsf{W}}\mathsf{Gen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
return $\pi := (\tilde{\mathsf{wit}}_1, \ldots, \tilde{\mathsf{wit}}_\ell)$

$\mathsf{NMVerify}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \pi)$
for $j = 1, \ldots, \ell$ do
    if $\mathsf{WVer}(\mathsf{pp}, \mathsf{st}_j, \mathsf{ind}, \tilde{\mathsf{wit}}_j, \mathbf{y}^*) \neq 1$ then
        return $0$
return $1$

$\tilde{\mathsf{W}}\mathsf{Gen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
find the smallest $\ell^* \in [\ell] : \mathsf{ind}_{1:\ell^*} \notin \mathcal{T}$
for $j = \ell^* - 1, \ldots, 0$ do
    $\mathbf{u}_{\mathsf{ind}_{1:j}\|0} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j}\|0})$
    $\mathbf{u}_{\mathsf{ind}_{1:j}\|1} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j}\|1})$
return $\tilde{\mathsf{wit}} := (\mathbf{u}_{\mathsf{ind}_{1:j}\|0}, \mathbf{u}_{\mathsf{ind}_{1:j}\|1})_{j=0}^{\ell^*-1}$

$\mathsf{WVer}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{wit}, \mathbf{y})$
parse $\mathsf{wit} := (\mathbf{u}_{\mathsf{ind}_{1:j}\|0}, \mathbf{u}_{\mathsf{ind}_{1:j}\|1})_{j=0}^{\ell^*-1}$
if $\mathbf{u}_{\mathsf{ind}_{\ell^*}} \neq -\mathbf{G}^{-1}(\mathbf{y})$ then
    return $0$
for $j = \ell^* - 1, \ldots, 0$ do
    $\mathbf{y}_{\mathsf{ind}_{1:j}} := \mathbf{A}_0 \cdot \mathbf{u}_{\mathsf{ind}_{1:j}\|0} + \mathbf{A}_1 \cdot \mathbf{u}_{\mathsf{ind}_{1:j}\|1}$
    $\mathbf{u}_{\mathsf{ind}_{1:j}} \leftarrow -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j}})$
if $\mathbf{y}_{\mathsf{ind}_{1:0}} \neq \mathsf{st}$ then
    return $0$
return $1$
```

**Fig. 4.** Membership and non-membership algorithms for our RBE scheme.

- $\mathsf{PSI.R}_2$: On input a set of ciphertexts $\{\mathsf{ctxt}_j\}_{j\in[n]}$, the receiver initializes a set $Z=\emptyset$. For all $j\in[n]$ and $k\in[m]$, the receiver does the following:
  - Compute $\mathsf{msg}_k=\mathsf{Dec}(\mathsf{pp},\mathsf{sk}_k,\mathsf{wit}_k,\mathsf{ctxt}_j)$.
  - If $\mathsf{msg}_k=\mathsf{msg}$ , add $x_k$ to the output $Z$.

  The receiver outputs $Z$.

**Theorem 7.** *If $\Pi$ is a secure laconic encryption scheme and* $\mathsf{H}$ *is collision-resistant, the laconic PSI protocol is correct and secure in the semi-honest model.*

*Proof.* Intuitively, the correctness follows that of laconic encryption scheme. The security against corrupted sender is from update privacy which means $\mathsf{st}$ hides the indices where the public keys have been registered. The security against corrupted receiver is from the pseudorandomness property of the encryption scheme such that a ciphertext with respect to an index not registered looks pseudorandom.

We now show the correctness of this protocol and that it is possible to simulate the transcripts for both parties which are computationally indistinguishable from the view in a real world.

**Correctness.** For simplicity, we first assume the sender holds an element $y$ and $y=x_k$. The receiver will output $x_k$ by checking

$$\mathsf{Dec}\left(\mathsf{sk}_k,\mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp},\mathsf{H}(x_k),\mathsf{pk}_k),\mathsf{ctxt}\right)=\mathsf{msg}.$$

Since $\mathsf{H}$ is collision-resistant, it happens with negligible probability that $y\neq x_k$ but they are mapped to the same index.

Then, we assume that the sender holds an element $y^*$ which does not get any match with the receiver's element. Since $\mathsf{msg}\in\mathcal{R}_2$ and the degree of $\mathcal{R}_2$ is $d_{\mathcal{R}}$, it holds that for all $k\in[m]$,

$$\Pr[\mathsf{Dec}\left(\mathsf{sk}_k,\mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp},\mathsf{H}(x_k),\mathsf{pk}_k),\mathsf{ctxt}^*\right)=\mathsf{msg}]\leq 1/2^{d_{\mathcal{R}}}$$

which is negligible under an appropriate choice of $d_{\mathcal{R}}$.

To achieve the correctness of a full protocol, we repeat the checking for each element in the sender's set.

**Security - Corrupted sender.** On input the set $S_{\mathsf{S}}$ and nothing else, we simulate the view of the corrupted sender by running the setup algorithm of laconic encryption scheme except that $\mathbf{y}_\epsilon,\mathbf{y}^*\leftarrow\$\ \mathcal{R}_q^n$ are distinct, and setting $\mathsf{st}:=\mathbf{y}_\epsilon$.

Due to update privacy of the laconic encryption scheme, $\mathsf{st}$ in the real world hides the indices perfectly. Therefore, the simulated $\mathsf{st}$ is perfectly indistinguishable from that in the real world. This concludes the security argument in the case of a corrupted sender.

**Security - Corrupted receiver.** On input the set $S_{\mathsf{R}}$ and the intersection $Z$ of a corrupted receiver, we simulate the view in the following way:

(i) Simulate the setup phase as an honest party, i.e., $(\mathsf{pp},\mathsf{st},\mathsf{aux})\leftarrow\mathsf{Setup}(1^\lambda)$ and samples an $\mathsf{H}$.
(ii) Update $\mathsf{st}$ by iterating the computation for all elements in $S_{\mathsf{R}}$ and sample a message $\mathsf{msg}\leftarrow\{0,1\}^\lambda$.
(iii) Let $Z=\{z_1,\ldots,z_\zeta\}$.
(iv) Pick a random subset $\Gamma\subset[n]$ with $|\Gamma|=|Z|=\zeta$. Let $\Gamma=\{\gamma_1,\ldots,\gamma_\zeta\}$. For all $j\in[\zeta]$, encrypt the message as in the protocol, i.e., $\mathsf{ctxt}_{\gamma_j}\leftarrow\mathsf{Enc}\left(\mathsf{pp},\mathsf{st},\mathsf{H}\left(z_j\right),\mathsf{msg}\right)$. This simulates the positions where the receiver gets a match.
(v) For the complimentary set $\Delta=Z\setminus\Gamma$, let $\Delta=\{\delta_1,\ldots,\delta_\eta\}$ where $\eta=n-\zeta$. For all $j\in[\eta]$, sample a binary string $r_j$ and compute $\mathsf{ctxt}_{\eta_j}\leftarrow\mathsf{Enc}\left(\mathsf{pp},\mathsf{st},\mathsf{H}\left(r_j\right),\mathsf{msg}\right)$. This simulates the positions where the receiver does not get a match.
(vi) Output $\{\mathsf{ctxt}_j\}_{j\in[n]}$as the simulated view.

We now show that the simulated view is indistinguishable from the view in the real execution using the following sequence of hybrids.

- Hybrid 0: Identical to the view of $\mathsf{R}$ in the real protocol.
- Hybrid $(1, 0)$: Instead of sampling the permutation $\pi$, we pick sets $(\Gamma, \Delta)$ where $\Gamma = \{\gamma_1, \ldots, \gamma_\zeta\}$ and $\Delta$ is the complimentary set, like in the simulation. For each $j$ we find the index $\sigma(j)$ such that $z_j = y_{\sigma(j)}$ and choose a random permutation $\pi$ such that $\pi(\gamma_j) = \sigma(j)$ and the remaining positions are filled at random.

  The indistinguishability between Hybrid 0 to $(1, 0)$ is immediate since in both cases $\pi$ is an uniform permutation in $[n]$.
- Hybrid $(1, j)$ for $j \in [n]$: In each hybrid, we change the distribution of a single $\mathsf{ctxt}_j$ such that in Hybrid $(1, j-1)$ it is generated as in the real protocol while in $(1, j)$ it is generated as in the simulation.

  For $j \in \Gamma$, $\mathsf{ctxt}_j$ is identically distributed in the real protocol and the simulation, and thus Hybrid $(1, j-1)$ and $(1, j)$ for $j \in \Gamma$ is perfectly indistinguishable.

  For $j \in \Delta$, under the pseudorandomness of the laconic encryption scheme, it is with negligible probability that an adversary can distinguish a ciphertext $\mathsf{ctxt}$ with respect to ind from that with respect to $\mathsf{ind}^*$ where none has been registered.

  Therefore, Hybrid $(1, j-1)$ and $(1, j)$ for $j \in \Delta$ is computationally indistinguishable.
- Hybrid 2: Identical to the view of $\mathsf{R}$ in the simulation.

  In Hybrid $(1, n)$, each $\mathsf{ctxt}_j$ from the real protocol has been replaced with that from the simulation. Therefore, Hybrid $(1, n)$ and 2 are indistinguishable, and this concludes the proof.

□

## 10 Optimizations and Extensions

In the following we describe a number of extensions and improvements for our laconic encryption scheme.

### 10.1 Pre-Processing

We now discuss our variant of laconic encryption with pre-processing. Loosely speaking, the $\mathsf{Enc}$ algorithm is split into an offline part, which is input-independent, and an online part. Importantly, the online algorithm is much more efficient and does not perform any "public-key operation". We introduce the updated syntax next, only for those algorithms that differ from the standard laconic encryption definition

**Definition 13 (Offline/Online Encryption).** *A laconic encryption scheme for message space $\mathcal{M}$ with offline/online encryption has the following syntax:*

- $(\mathsf{ctxt}, \mathsf{rand}) \leftarrow \mathsf{OfflineEnc}(\mathsf{pp}, \mathsf{st})$*: The offline encryption algorithm takes as input the public parameters* $\mathsf{pp}$ *and the state* $\mathsf{st}$*, and outputs a ciphertext* $\mathsf{ctxt}$ *and some internal randomness* $\mathsf{rand}$*.*
- $\mathsf{online} \leftarrow \mathsf{OnlineEnc}(\mathsf{pp}, \mathsf{st}, \mathsf{rand}, \mathsf{ind}, \mathsf{msg})$*: The online encryption algorithm takes as input the public parameters* $\mathsf{pp}$*, the state* $\mathsf{st}$*, the randomness* $\mathsf{rand}$*, an index* $\mathsf{ind} \in \{0,1\}^\ell$*, and a message* $\mathsf{msg} \in \mathcal{M}$*. It outputs an online information* $\mathsf{online}$*.*
- $\mathsf{msg} \leftarrow \mathsf{OnlineDec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt}, \mathsf{online})$*: The online decryption algorithm takes as input the public parameters* $\mathsf{pp}$*, a secret key* $\mathsf{sk}$*, a membership witness* $\mathsf{wit}$*, a ciphertext* $\mathsf{ctxt}$*, and the online information* $\mathsf{online}$*. It outputs a message* $\mathsf{msg}$*.*

Correctness and security are defined identically as in Section 7, except for the syntactical modifications required to split the encryption algorithm in two subroutines $\mathsf{Enc} := (\mathsf{OfflineEnc}, \mathsf{OnlineEnc})$.

Next we describe how to equip our laconic encryption scheme in Section 7 with pre-processing. We will make use of the one-time pad for which, for convenience, we define the syntax in the following:

$$(\mathsf{ciph}, \mathsf{key}) \leftarrow \mathsf{OTPEnc}(\mathsf{msg}) \text{ and } \mathsf{msg} \leftarrow \mathsf{OTPDec}(\mathsf{ciph}, \mathsf{key}).$$

We present the online/offline algorithms in Fig. 5. Informally, our scheme pre-computes all possible branches of the laconic encryption and commits to the corresponding ciphertext elements one by one. In the online phase, the encryption algorithm can simply open the subset indexed by $\mathsf{ind}$, which then constitutes a valid

```
$\mathsf{OfflineEnc}(\mathsf{pp}, \mathsf{st})$
$r \leftarrow\!\!\$\ \mathcal{R}_2$
$\mathbf{r}_j \leftarrow\!\!\$\ \mathcal{R}_q^n,\ \forall j \in \{0, \dots, \ell\}$
for $j = 0, \dots, \ell - 1$ do
  $\mathbf{e}_j \leftarrow\!\!\$\ \chi^{2m}$
  for $b = 0, 1$ do
    $\mathbf{B}_{j,b} := \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 \\ (b \oplus 1) \cdot \mathbf{G} & b \cdot \mathbf{G} \end{pmatrix}$
    $\mathbf{c}_{j,b}^{\intercal} := (\mathbf{r}_j^{\intercal}, \mathbf{r}_{j+1}^{\intercal}) \cdot \mathbf{B}_{j,b} + \mathbf{e}_j^{\intercal} \bmod q$
    $(\mathsf{ciph}_{j,b}, \mathsf{key}_{j,b}) \leftarrow \mathsf{OTPEnc}(\mathbf{c}_{j,b})$
$\mathbf{e}_\ell \leftarrow\!\!\$\ \chi^m,\ e \leftarrow\!\!\$\ \chi$
$\mathbf{c}_\ell^{\intercal} := \mathbf{r}_\ell^{\intercal} \cdot \mathbf{B} + \mathbf{e}_\ell^{\intercal} \bmod q$
$d := \mathbf{r}_0^{\intercal} \cdot \mathbf{y}_\epsilon + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot r \bmod q$
$\mathsf{ctxt} := (\mathsf{ciph}_{0,0}, \dots, \mathsf{ciph}_{\ell-1,1}, \mathbf{c}_\ell, d)$
$\mathsf{rand} := (\mathsf{key}_{0,0}, \dots, \mathsf{key}_{\ell-1,1}, r)$
return $(\mathsf{ctxt}, \mathsf{rand})$
```

```
$\mathsf{OnlineEnc}(\mathsf{pp}, \mathsf{st}, \mathsf{rand}, \mathsf{ind}, \mathsf{msg})$
parse rand as $(\mathsf{key}_{0,0}, \dots, \mathsf{key}_{\ell-1,1}, r)$
return online $:= (\mathsf{key}_{0,\mathsf{ind}_1}, \dots, \mathsf{key}_{\ell-1,\mathsf{ind}_\ell}, r \oplus \mathsf{msg})$
  // We abuse notation and treat $r \in \mathcal{R}_2$ as a bit string.

$\mathsf{OnlineDec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt}, \mathsf{online})$
parse ctxt as $(\mathsf{ciph}_{0,0}, \dots, \mathsf{ciph}_{\ell-1,1}, \mathbf{c}_\ell, d)$
parse online as $(\mathsf{key}_{0,\mathsf{ind}_1}, \dots, \mathsf{key}_{\ell-1,\mathsf{ind}_\ell}, s)$
for $j = 0, \dots, \ell - 1$ do
  $\mathbf{c}_j \leftarrow \mathsf{OTPDec}(\mathsf{ciph}_{j,\mathsf{ind}_{j+1}}, \mathsf{key}_{j,\mathsf{ind}_{j+1}})$
$\mathsf{ctxt}^* := (\mathbf{c}_0, \dots, \mathbf{c}_\ell, d)$
return $\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt}^*) \oplus s$
```

**Fig. 5.** Laconic encryption with pre-processing.

laconic encryption ciphertext. Crucially, the OnlineEnc algorithm is entirely combinatorial, and it does not perform any expensive public-key operation. Correctness is immediate, and below we state the security of our scheme.

**Theorem 8.** *If $\Pi$ is a secure laconic encryption scheme, the offline/online variant in Fig. 5 is also secure.*

*Proof.* The proof follows immediately by observing that OTPEnc hides the unopened messages unconditionally, and thus the ciphertext $\mathsf{ctxt}^*$ is identically distributed to a standard ciphertext of $\Pi$ for the index ind. Thus, security follows by an invocation of the security of $\Pi$. □

### 10.2 Compatibility with Other Public-Keys

Our construction of laconic encryption is tied to a particular encryption algorithm which, at least superficially, resembles the dual-Regev encryption algorithm [GPV08]. A natural question is whether this forces one to use this particular encryption scheme for the *leaf nodes* or whether our laconic encryption paradigm can support any encryption scheme of our choice. An obvious advantage of the latter is to be able to use *already existing* keys for other schemes. In the following we outline how one can modify the laconic encryption algorithm to achieve this goal.

**A Strawman Solution.** To gain an intuition on how we can achieve this generically, it is convenient to observe that laconic encryption implies laconic OT in a black-box sense (this intuition is made formal in Appendix A). Henceforth, we can assume that we have a laconic OT scheme with essentially the same complexity and computational overhead of the laconic encryption scheme described in Section 7. Then, a naive solution to make the scheme compatible with any encryption algorithm is to parse the database as a collection of public keys of the desired scheme (in their binary encoding), that is, the database is defined as $D = \mathsf{pk}_1 \| \dots \| \mathsf{pk}_n$. The hash of the receiver is defined as the output of OTReceive on input $D$.

To encrypt a message msg with respect of a public-key $\mathsf{pk}_i$, the encryption algorithm computes a randomized encoding [Yao86] for the function

$$f(x) = \mathsf{Enc}^*(x, \mathsf{msg})$$

where $\mathsf{Enc}^*$ is an arbitrary encryption algorithm. Let $\{\mathsf{lab}_{j,0}, \mathsf{lab}_{j,1}\}_{j\in[|\mathsf{pk}|]}$ be the corresponding labels, the encryption algorithm sends the randomized encoding along with

$$\{\mathsf{OTSend}((\mathsf{lab}_{j,0}, \mathsf{lab}_{j,1}), (i-1)\cdot|\mathsf{pk}| + j)\}_{j\in[|\mathsf{pk}|]}\,.$$

By executing the decoding algorithm, the receiver can then recover $\{\mathsf{lab}_{j,\mathsf{pk}_j}\}_{j\in[|\mathsf{pk}|]}$, which in turn allows it to run the decoding algorithm for the randomized encoding, which yields a ciphertext $c = \mathsf{Enc}^*(\mathsf{pk}_i, \mathsf{msg})$. Security follows by a standard argument invoking the security of the laconic OT and the simulatability of the randomized encoding algorithm.

**A Black-Box Solution.** While the solution sketched above does indeed work for any encryption scheme, it is not satisfactory because it makes non black-box use of the $\mathsf{Enc}^*$ algorithm, which is a well-known source of inefficiency. Here we show how to get around this problem when $\mathsf{Enc}^*$ is a linear algorithm (over any modulus $q$). Specifically, we assume that there exists a linear map $L_{\mathbf{r},\mathsf{msg}}$ such that

$$L_{\mathbf{r},\mathsf{msg}}(\mathsf{pk}) = \mathsf{Enc}^*(\mathsf{pk}, \mathsf{msg}; \mathbf{r}).$$

Note that this property is satisfied by a large class of encryption algorithms, such as code-based encryption [Ale03], primal Regev encryption [Reg10] and derivatives, including NIST-standardized algorithms [BDK+17].

First, we observe that $L_{\mathbf{r},\mathsf{msg}}(\mathsf{pk})$ can be still be interpreted as a linear map, even if the public-key is decomposed in its binary representation. This is because we can rewrite $L_{\mathbf{r},\mathsf{msg}}(\mathsf{pk}) : \mathbb{Z}_q^{\eta\times n} \to \mathbb{Z}_q^{\eta}$ as

$$\begin{aligned} L_{\mathbf{r},\mathsf{msg}}(\mathsf{pk}) &= \left(\mathsf{pk}_1\cdot\mathbf{c}_1, \ldots, \mathsf{pk}_\eta\cdot\mathbf{c}_\eta\right)\\ &= \left(\sum_{j\in[n]} \mathsf{pk}_{1,j}\cdot\mathbf{c}_{1,j}, \ldots, \sum_{j\in[n]} \mathsf{pk}_{\eta,j}\cdot\mathbf{c}_{\eta,j}\right), \end{aligned}$$

let then $\mathbf{G}^{-1}(\mathsf{pk}_{i,j}) \in \{0,1\}^{\log(q)}$ be the binary decomposition of the corresponding element of $\mathsf{pk}$, we can expand the above expression as

$$= \left(\sum_{j\in[n]}\sum_{\kappa\in[\log(q)]} \mathbf{G}^{-1}(\mathsf{pk}_{1,j})_\kappa\cdot 2^{\kappa-1}\mathbf{c}_{1,j}, \ldots, \sum_{j\in[n]}\sum_{\kappa\in[\log(q)]} \mathbf{G}^{-1}(\mathsf{pk}_{\eta,j})_\kappa\cdot 2^{\kappa-1}\mathbf{c}_{\eta,j}\right).$$

We are now going to show a *special-purpose* randomized encoding for recovering the $i$-th element (for $i\in[\eta]$) of the above expression. The pairs labels of the randomized encoding consist of

$$\left\{\mathsf{lab}_{0,i,j,\kappa} = \mathsf{r}_{i,j,\kappa}, \mathsf{lab}_{1,i,j,\kappa} = \mathsf{r}_{i,j,\kappa} + 2^{\kappa-1}\mathbf{c}_{i,j}\right\}_{j\in[n],\kappa\in[\log(q)]}$$

where $\mathsf{r}_{i,j,\kappa}$ are sampled uniformly from $\mathbb{Z}_q$ conditioned on $\sum_{j\in[n]}\sum_{\kappa\in[\log(q)]}\mathsf{r}_{i,j,\kappa} = 0$. There is nothing else needed from the encoding algorithm. The decoding algorithm, given $\{\mathsf{lab}_{\mathbf{G}^{-1}(\mathsf{pk}_{i,j})_k,i,j,\kappa}\}_{j\in[n],\kappa\in[\log(q)]}$, can recover the $i$-th element of the above sum by simply computing

$$\begin{aligned} \sum_{j\in[n]}\sum_{\kappa\in[\log(q)]} \mathsf{lab}_{\mathbf{G}^{-1}(\mathsf{pk}_{i,j})_k,i,j,\kappa} &= \sum_{j\in[n]}\sum_{\kappa\in[\log(q)]} \mathsf{r}_{i,j,\kappa} + \mathbf{G}^{-1}(\mathsf{pk}_{i,j})_\kappa\cdot 2^{\kappa-1}\mathbf{c}_{i,j}\\ &= \sum_{j\in[n]}\sum_{\kappa\in[\log(q)]} \mathbf{G}^{-1}(\mathsf{pk}_{i,j})_\kappa\cdot 2^{\kappa-1}\mathbf{c}_{i,j}. \end{aligned}$$

Note that the randomized encoding consists merely of a collection of labels and the decoding algorithm is very efficient as it only consists of the sum of a subset of such labels. Furthermore, the labels are uniform, conditioned on them summing up to the desired coefficient, so the scheme is trivially simulatable.

Plugging this scheme into the compiler above, we observe that the encryption algorithm now consists exclusively of a collection of OT sender messages and the decoding algorithm only sums the resulting messages to obtain a ciphertext under the desired key.

## 11 IBE Construction

**Definition.** We recall the syntax and the security definition of identity-based encryption (IBE) as follows.

**Definition 14 (Identity-based Encryption).** *An identity-based encryption (IBE) scheme for message space $\mathcal{M}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$*: The setup algorithm takes as input the security parameter* $1^\lambda$ *and a length parameter* $1^\ell$*. It generates a master public key* $\mathsf{mpk}$ *and a master secret key* $\mathsf{msk}$*.*
- $\mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{msk}, \mathsf{id})$*: The key generation algorithm inputs the master secret key* $\mathsf{msk}$ *and an identity* $\mathsf{id} \in \{0,1\}^\ell$ *and outputs an identity secret key* $\mathsf{sk}$*.*
- $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg})$*: The encryption algorithm inputs the master public key* $\mathsf{mpk}$*, an identity* $\mathsf{id} \in \{0,1\}^\ell$*, and a message* $\mathsf{msg} \in \mathcal{M}$*. It outputs a ciphertext* $\mathsf{ctxt}$*.*
- $\mathsf{msg} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt})$*: The decryption algorithm inputs an identity secret key* $\mathsf{sk}$ *and a ciphertext* $\mathsf{ctxt}$*. It outputs a message* $\mathsf{msg}$*.*

**Definition 15 (Correctness).** *An identity-based encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be statistically correct if for any* $\ell = \mathsf{poly}(\lambda)$*, any identity* $\mathsf{id} \in \{0,1\}^\ell$*, and any message* $\mathsf{msg} \in \mathcal{M}$ *it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) = \mathsf{msg} \;\middle|\; \begin{array}{l} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ \mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{msk}, \mathsf{id}) \\ \mathsf{ctxt} \in \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

**Definition 16 (Pseudorandom Ciphertexts).** *An identity-based encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with ciphertext space* $\mathcal{C}$ *is said to be have* pseudorandom ciphertexts *if for any PPT (stateful) adversary* $\mathcal{A}$*, any* $\ell = \mathsf{poly}(\lambda)$*, it holds that*

$$\left|\Pr\left[\mathsf{Pseudorandomness}^0_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right] - \Pr\left[\mathsf{Pseudorandomness}^1_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell) = 1\right]\right|$$

$\leq \mathsf{negl}(\lambda)$ *where the experiment* $\mathsf{Pseudorandomness}^b_{\Pi,\mathcal{A}}$ *is defined in Fig. 6.*

```
$\mathsf{Pseudorandomness}^b_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$
    $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
    $(\mathsf{id}^*, \mathsf{msg}^*) \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}}(\mathsf{mpk})$
    if $b = 0$ then $\mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{msg}^*)$
    else $\mathsf{ctxt}^* \leftarrow\$ \mathcal{C}$
    $b' \leftarrow \mathcal{A}^{\mathsf{KGen}\mathcal{O}}(\mathsf{ctxt}^*)$
    if $\mathsf{id}^* \in \mathcal{Q}$ then return $0$
    return $b'$

$\mathsf{KGen}\mathcal{O}(\mathsf{id})$
    $\mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{msk}, \mathsf{id})$
    $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{id}\}$
    return $\mathsf{sk}$
```

**Fig. 6.** The security-experiment for the pseudorandom ciphertexts property for IBE

**Trapdoor Sampling.** We require the trapdoor generation and sampling algorithms with the following syntax:

$$(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(\mathcal{R}, n, 2m, q, \beta) \text{ and } \mathbf{u} \leftarrow \mathsf{SampPre}(\mathsf{td}, \mathbf{y}, \mathbf{x}).$$

We require that the following distributions are statistically close:

$$\left\{ (\mathbf{A},\mathbf{x},\mathbf{y},\mathbf{u}) : \begin{array}{l} \mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{n\times 2m} \\ \mathbf{x} \leftarrow\!\!\$\ \mathcal{R}_p^m \\ \mathbf{u} \leftarrow\!\!\$\ \chi^m \\ \mathbf{y} \coloneqq \mathbf{A}\cdot(\mathbf{x},\mathbf{u}) \bmod q \end{array} \right\} \quad \text{and} \quad \left\{ (\mathbf{A},\mathbf{x},\mathbf{y},\mathbf{u}) : \begin{array}{l} (\mathbf{A},\mathsf{td}) \leftarrow \mathsf{TrapGen}(\mathcal{R},n,2m,q,\beta) \\ \mathbf{x} \leftarrow\!\!\$\ \mathcal{R}_p^m \\ \mathbf{y} \leftarrow\!\!\$\ \mathcal{R}_p^m \\ \mathbf{u} \leftarrow \mathsf{SampPre}(\mathsf{td},\mathbf{y},\mathbf{x}) \end{array} \right\},$$

and that the vectors $\mathbf{u}$ from both distributions have $\|\mathbf{u}\| \leq \beta$ with overwhelming probability. Note that this can be achieved as a standard modification of the [MP12] by sampling a trapdoor matrix $(\mathbf{A},\mathsf{td}) \leftarrow \mathsf{TrapGen}(\mathcal{R},n,m,q,\beta)$ along with a uniform $\tilde{\mathbf{A}} \leftarrow \mathcal{R}_q^{n\times m}$ and defining $\mathbf{A}^*$ to be the row concatenation of $\tilde{\mathbf{A}}$ and $\mathbf{A}$. Next, note that sampling $\mathbf{u}$ such that $\mathbf{A}^*\cdot(\mathbf{x},\mathbf{u}) = \mathbf{y}$ reduces to the constraint $\mathbf{A}\mathbf{u} = \mathbf{y} + \tilde{\mathbf{A}}\mathbf{x}$ and therefore $\mathbf{u}$ can be sampled using the standard $\mathsf{SampPre}$ algorithm together with the trapdoor $\mathsf{td}$.

**Construction Overview.** We construct an identity-based encryption scheme for the message space $\mathcal{M} = \mathcal{R}_2$ in Fig. 7. Our IBE construction is conceptually similar to the laconic encryption scheme constructed in Section 7 but with a crucial difference: Instead of encrypting with respect to a Merkle tree opening, we encrypt with respect to a path in a "trapdoored GGM tree". Different from a standard GGM tree [GGM84] which can be publicly computed from the root, a trapdoor is required to traverse down a trapdoored GGM tree. We will elaborate on this below.

Our construction is parametrised by $n,m,p,q \in \mathbb{N}$ and distributions $\chi$ and $\bar{\chi}$ over $\mathcal{R}$. The master public key consists of random matrice $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{n\times m}$ and $\mathbf{B} \leftarrow\!\!\$\ \mathcal{R}_q^{n\times m}$, four trapdoored matrices $\mathbf{B}_{b,b'}$ for $(b,b') \in \{0,1\}^2$, where $(\mathbf{B}_{b,b'},\mathsf{td}_{b,b'}) \leftarrow \mathsf{TrapGen}(\mathcal{R},1^n,1^m,q,p/2)$, and a random vector $\mathbf{y}_\epsilon \leftarrow\!\!\$\ \mathcal{R}_q^n$ which serves as the root label of the GGM tree. Correspondingly, the master secret key consists of the trapdoors $\mathsf{td}_{b,b'}$ and a key $k \leftarrow\!\!\$\ \{0,1\}^\lambda$ for a pseudorandom function $\mathsf{PRF}$. Looking ahead, the GGM tree will be partitioned into odd and even layers, i.e. with parity 1 and 0, and the trapdoor $\mathsf{td}_{b,b'}$ will be used to traverse to $b$-children at layers with parity $b'$.

The identity secret key generation algorithm takes as input an identity $\mathsf{id} = (\mathsf{id}_1,\ldots,\mathsf{id}_\ell) \in \{0,1\}^\ell$ and traverses down the GGM tree from the root to the leaf indexed by $\mathsf{id}$. For $i \in [\ell]$, write $\oplus i$ for the parity of $i$. At each node $\mathsf{id}_{1:i}$, it samples $\mathbf{x}_{\mathsf{id}_{1:i}} \leftarrow\!\!\$\ \mathcal{R}_p^m$ using $\mathsf{PRF}(k,0\|\mathsf{id}_{1:i-1})$ as randomness, then computes[12]

$$\begin{aligned} \mathbf{y}_{\mathsf{id}_{1:i}} &\coloneqq \mathbf{B}\cdot\mathbf{x}_{\mathsf{id}_{1:i}} \bmod q, \\ \mathbf{u}_{\mathsf{id}_{1:i}} &\coloneqq -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:i}}), \text{ and} \\ \mathbf{v}_{\mathsf{id}_{1:i}} &\leftarrow \mathsf{SampPre}(\mathsf{td}_{\mathsf{id}_i,\oplus i}, \mathbf{y}_{\mathsf{id}_{1:i-1}} - \mathbf{A}\cdot\mathbf{u}_{1:i} \bmod q; \mathsf{PRF}(k,1\|\mathsf{id}_{1:i-1})). \end{aligned}$$

Note that $(\mathbf{u}_{\mathsf{id}_{1:i}},\mathbf{v}_{\mathsf{id}_{1:i}})$ and $\mathbf{u}_{\mathsf{id}_{1:i-1}}$ satisfy

$$\underbrace{\mathbf{G}\cdot\mathbf{u}_{\mathsf{id}_{1:i-1}}}_{-\mathbf{y}_{\mathsf{id}_{1:i-1}}} + \mathbf{A}\cdot\mathbf{u}_{\mathsf{id}_{1:i}} + \mathbf{B}_{\mathsf{id}_i,\oplus i}\cdot\mathbf{v}_{\mathsf{id}_{1:i}} = \mathbf{0} \bmod q$$

and $(\mathbf{u}_{\mathsf{id}},\mathbf{x}_{\mathsf{id}})$ satisfy

$$\mathbf{G}\cdot\mathbf{u}_{\mathsf{id}} + \mathbf{B}\cdot\mathbf{x}_{\mathsf{id}} = \mathbf{0} \bmod q.$$

The key generation algorithm finally outputs $\mathsf{sk}_{\mathsf{id}} \coloneqq (\mathbf{u}_{[\mathsf{id}]},\mathbf{x}_{\mathsf{id}})$ where

$$\mathbf{u}_{[\mathsf{id}]}^{\mathsf{T}} \coloneqq (\mathbf{u}_{\mathsf{id}_1}^{\mathsf{T}},\mathbf{v}_{\mathsf{id}_1}^{\mathsf{T}},\mathbf{u}_{\mathsf{id}_{1:2}}^{\mathsf{T}},\mathbf{v}_{\mathsf{id}_{1:2}}^{\mathsf{T}},\ldots,\mathbf{u}_{\mathsf{id}_{1:\ell}}^{\mathsf{T}},\mathbf{v}_{\mathsf{id}_{1:\ell}}^{\mathsf{T}}).$$

[12]We remark that only $\mathbf{x}_{\mathsf{id}} = \mathbf{x}_{1:\ell}$ is included in $\mathsf{sk}_{\mathsf{id}}$ while the only purpose of $\mathbf{x}_{1:i}$, for $i \in [\ell-1]$, is to derive $\mathbf{y}_{\mathsf{id}_{1:i}}$. Therefore we could simplify the key generation algorithm by sampling $\mathbf{y}_{\mathsf{id}_{1:i}}$ directly using $\mathsf{PRF}(k,0\|\mathsf{id}_{1:i-1})$ as randomness. We decide to present the construction in its current form to highlight the structure.

To encrypt a message $\mathsf{msg} \in \mathcal{M}$ to an identity $\mathsf{id}$, the encryptor performs dual-Regev encryption using $(\mathbf{A}_{\mathsf{id}}, \mathbf{v})$ as the public key, where

$$\mathbf{A}_{\mathsf{id}} \coloneqq \begin{pmatrix} \mathbf{A}\ \mathbf{B}_{\mathsf{id}_1,1} & & & & \\ \mathbf{G} & \mathbf{A}\ \mathbf{B}_{\mathsf{id}_2,0} & & & \\ & \mathbf{G} & & & \\ & & \ddots\ \ddots & & \\ & & & \mathbf{A}\ \mathbf{B}_{\mathsf{id}_{\ell-1},\oplus(\ell-1)} & \\ & & & \mathbf{G} & \mathbf{A}\ \mathbf{B}_{\mathsf{id}_\ell,\oplus\ell} \\ & & & & \mathbf{G} \qquad \mathbf{B} \end{pmatrix} \qquad \text{and} \qquad \mathbf{v} \coloneqq \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

In other words, the message $\mathsf{msg} \in \mathcal{M}$ is encrypted as $(\mathbf{c}, d) \in \mathcal{R}_q^{(2\ell+1)m} \times \mathcal{R}_q$ where

$$\mathbf{c}^{\mathtt{T}} = \mathbf{r}^{\mathtt{T}} \cdot \mathbf{A}_{\mathsf{id}} + \mathbf{e}^{\mathtt{T}} \bmod q$$
$$d = \mathbf{r}^{\mathtt{T}} \cdot \mathbf{v} + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$$

for some random LWE secret $\mathbf{r}$ and small noises $e$ and $\mathbf{e}$. In Fig. 7, we write the expressions of the ciphertext components explicitly without referring to $\mathbf{A}_{\mathsf{id}}$ and $\mathbf{v}$.

Note the the secret key $\mathsf{sk}_{\mathsf{id}}$ satisfies the relation

$$\mathbf{A}_{\mathsf{id}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{id}]} \\ \mathbf{x}_{\mathsf{id}} \end{pmatrix} = \mathbf{v} \bmod q \qquad \text{and} \qquad \left\| \mathbf{u}_{[\mathsf{id}]} \right\| \leq p/2.$$

Therefore, to decrypt a ciphertext $(\mathbf{c}, d)$, the user with identity secret key $\mathsf{sk}_{\mathsf{id}}$ computes $d - \mathbf{c}^{\mathtt{T}} \cdot \mathbf{u}_{[\mathsf{id}]} \bmod q$ and round the result to the nearest multiple of $\frac{q}{2}$. The decryption is correct whenever $\left| e - \mathbf{e}^{\mathtt{T}} \cdot \mathbf{u}_{[\mathsf{id}]} \right| < (q-1)/4$.

**Analysis.** We analyze the correctness and security of the IBE construction.

**Theorem 9.** *Let $\mathcal{R}, \ell, m, p, q, s, t$ be such that $s < t$, $\chi = \mathcal{D}_{\mathcal{R},s}$, $\bar{\chi} = \mathcal{D}_{\mathcal{R},t}$, and $q > ((2\ell+1) \cdot m \cdot \gamma_{\mathcal{R}} \cdot p + 4) \cdot \sqrt{\lambda} \cdot s + 1$. The construction in Fig. 7 is correct with overwhelming probability in $\lambda$.*

*Proof.* Write
$$\mathbf{e}^{\mathtt{T}} \coloneqq \left( \mathbf{e}_0^{\mathtt{T}} \ldots \mathbf{e}_{\ell-1}^{\mathtt{T}}\ \mathbf{e}_\ell^{\mathtt{T}} \right).$$

Observe that decryption is correct whenever $|e - \mathbf{e}^{\mathtt{T}} \cdot \mathsf{sk}_{\mathsf{id}}| < (q-1)/4$. By Lemma 2, with overwhelming probability in $\lambda$, we have $\|e\| \leq \frac{\sqrt{\lambda}}{2} \cdot t$ and $\|\mathbf{e}\| \leq \frac{\sqrt{\lambda}}{2} \cdot s < \frac{\sqrt{\lambda}}{2} \cdot t$. Since $\mathsf{sk}_{\mathsf{id}} \in \mathcal{R}_p^{(2\ell+1)m}$, we have $\|\mathsf{sk}_{\mathsf{id}}\| \leq p/2$. Combining these facts yields

$$\left\| e - \mathbf{e}^{\mathtt{T}} \cdot \begin{pmatrix} \mathbf{u}_{[\mathsf{ind}]} \\ \mathbf{x}_{\mathsf{ind}} \end{pmatrix} \right\| \leq (2\ell+1) \cdot m \cdot \gamma_{\mathcal{R}} \cdot \frac{\sqrt{\lambda}}{2} \cdot s \cdot \frac{p}{2} + \sqrt{\lambda} \cdot s < (q-1)/4$$

with overwhelming probability in $\lambda$. □

We next argue about security. Before stating our main theorem, for each $(b, b') \in \{0,1\}^2$, we describe a useful stateful oracle $\mathsf{KGen}\mathcal{O}^*_{b,b'}$ that generates keys like the standard $\mathsf{KGen}\mathcal{O}$ oracle, except that it only uses three out of four trapdoors, i.e. excluding $\mathsf{td}_{b,b'}$. The oracle $\mathsf{KGen}\mathcal{O}^*_{b,b'}$ maintains a binary tree of depth at most $\ell$ where a node (indexed by) $\mathsf{id}_{1:i} = (\mathsf{id}_1, \ldots, \mathsf{id}_i) \in \{0,1\}^i$ is labelled by $(\mathbf{u}_{id_{1:i}}, \mathbf{v}_{id_{1:i}})$. A node $\mathsf{id}_{1:i}$ is said to be a challenge node if $\mathsf{id}_i = b$ and $\oplus i = b'$. Between each invocation of $\mathsf{KGen}\mathcal{O}^*_{b,b'}$, the following invariant is maintained for each node $\mathsf{id}_{1:i}$ of the tree:

**Rule 1.** If $\mathsf{id}_{1:i}$ is labelled, its parent $\mathsf{id}_{1:i-1}$ is also labelled.
**Rule 2.** If $\mathsf{id}_{1:i}$ is a challenge node and its parent $\mathsf{id}_{1:i-1}$ is labelled, then $\mathsf{id}_{1:i}$ is also labelled.

```
$\mathsf{Setup}(1^\lambda)$
$\mathbf{A}, \mathbf{B} \leftarrow\!\!\$\, \mathcal{R}_q^{n \times m},\ \mathbf{y}_\epsilon \leftarrow\!\!\$\, \mathcal{R}_q^n$
$(\mathbf{B}_{b,b'}, \mathsf{td}_{b,b'}) \leftarrow \mathsf{TrapGen}(\mathcal{R}, 1^n, 1^m, q, p/2),\ \forall (b, b') \in \{0,1\}^2$
$k \leftarrow\!\!\$\, \{0,1\}^\lambda$
$\mathsf{mpk} := \big(\mathbf{A}, \mathbf{B}, (\mathbf{B}_{b,b'})_{(b,b') \in \{0,1\}^2}, \mathbf{y}_\epsilon\big)$
$\mathsf{msk} := \big((\mathsf{td}_{b,b'})_{(b,b') \in \{0,1\}^2}, k\big)$
$\mathbf{return}\ (\mathsf{mpk}, \mathsf{msk})$

$\mathsf{KGen}(\mathsf{msk}, \mathsf{id} \in \{0,1\}^\ell)$
$\mathbf{for}\ i \in [\ell]\ \mathbf{do}$
    $\mathbf{x}_{\mathsf{id}_{1:i}} \overset{\mathsf{PRF}(k, 0 \| \mathsf{id}_{1:i-1})}{\leftarrow\!\!\$} \mathcal{R}_p^m$
    $\mathbf{y}_{\mathsf{id}_{1:i}} := \mathbf{B} \cdot \mathbf{x}_{\mathsf{id}_{1:i}} \bmod q$
    $\mathbf{u}_{\mathsf{id}_{1:i}} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:i}})$
    $\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow \mathsf{SampPre}(\mathsf{td}_{\mathsf{id}_i, \oplus i}, \mathbf{y}_{1:i-1} - \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} \bmod q; \mathsf{PRF}(k, 1 \| \mathsf{id}_{1:i-1}))$
$\mathbf{u}_{[\mathsf{id}]}^\mathsf{T} := (\mathbf{u}_{\mathsf{id}_1}^\mathsf{T}, \mathbf{v}_{\mathsf{id}_1}^\mathsf{T}, \mathbf{u}_{\mathsf{id}_{1:2}}^\mathsf{T}, \mathbf{v}_{\mathsf{id}_{1:2}}^\mathsf{T}, \ldots, \mathbf{u}_{\mathsf{id}_{1:\ell}}^\mathsf{T}, \mathbf{v}_{\mathsf{id}_{1:\ell}}^\mathsf{T})$
$\mathbf{return}\ \mathsf{sk}_\mathsf{id} := (\mathbf{u}_{[\mathsf{id}]}, \mathbf{x}_\mathsf{id})$

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{msg})$
$\mathbf{r}_j \leftarrow\!\!\$\, \mathcal{R}_q^n,\ \forall j \in \{0, \ldots, \ell\}$
$\mathbf{for}\ j \in [\ell]\ \mathbf{do}$
    $\mathbf{e}_{j-1} \leftarrow\!\!\$\, \chi^{2m}$
    $\mathbf{c}_{j-1}^\mathsf{T} := (\mathbf{r}_{j-1}^\mathsf{T}, \mathbf{r}_j^\mathsf{T}) \cdot \begin{pmatrix} \mathbf{A} & \mathbf{B}_{\mathsf{id}_j, \oplus j} \\ \mathbf{G} & \end{pmatrix} + \mathbf{e}_{j-1}^\mathsf{T} \bmod q,\ \forall j \in [\ell]$
$\mathbf{e}_\ell \leftarrow\!\!\$\, \chi^m,\ e \leftarrow\!\!\$\, \bar{\chi}$
$\mathbf{c}_\ell^\mathsf{T} := \mathbf{r}_\ell^\mathsf{T} \cdot \mathbf{B} + \mathbf{e}_\ell^\mathsf{T} \bmod q$
$d := \mathbf{r}_0^\mathsf{T} \cdot \mathbf{y}_\epsilon + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$
$\mathbf{return}\ \mathsf{ctxt} := (\mathbf{c}_0, \ldots, \mathbf{c}_\ell, d)$

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{id}, \mathsf{sk}_\mathsf{id}, \mathsf{ctxt})$
$\bar{\mu} := d - \sum_{j=1}^{\ell} \mathbf{c}_{j-1}^\mathsf{T} \cdot \begin{pmatrix} \mathbf{u}_{\mathsf{id}_{1:j}} \\ \mathbf{v}_{\mathsf{id}_{1:j}} \end{pmatrix} - \mathbf{c}_\ell^\mathsf{T} \cdot \mathbf{x}_\mathsf{id} \bmod q$
$\mathbf{if}\ |\bar{\mu}| < q/4\ \mathbf{then\ return}\ 0$
$\mathbf{else\ return}\ 1$
```

**Fig. 7.** Construction of identity-based encryption.

On input a new identity $\mathsf{id}$, the oracle $\mathsf{KGen}\mathcal{O}^*_{b,b'}$ samples $\mathbf{u}_{\mathsf{id}_{1:\ell}} \leftarrow\!\!\$\ \mathcal{R}_p^m$, then proceeds to label the nodes on the path $\mathsf{id}$ from leaf to root as follows. For each $i$ from $\ell$ to 1:

– If $\mathbf{u}_{\mathsf{id}_{1:i-1}}$ is not set, meaning that $\mathbf{v}_{\mathsf{id}_{1:i}}$ is also not set (Rule 1), then they will be set as follows.
  - If the sibling of $\mathsf{id}_{1:i}$ is a challenge node, meaning that it must not have been labelled since the parent is not labelled (Rule 2), sample $(\mathbf{u}_{\mathsf{sibling}(\mathsf{id}_{1:i})}, \mathbf{v}_{\mathsf{sibling}(\mathsf{id}_{1:i})}) \leftarrow\!\!\$\ (\mathcal{R}_p^m)^2$ and compute

$$\mathbf{u}_{\mathsf{id}_{1:i-1}} := -\mathbf{G}^{-1}\left(\mathbf{A} \cdot \mathbf{u}_{\mathsf{sibling}(\mathsf{id}_{1:i})} + \mathbf{B}_{1-\mathsf{id}_i, \oplus i} \cdot \mathbf{v}_{\mathsf{sibling}(\mathsf{id}_{1:i})} \bmod q\right)$$

  and

$$\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow \mathsf{SampPre}\left(\mathsf{td}_{\mathsf{id}_i, \oplus i}, -\left(\mathbf{G} \cdot \mathbf{u}_{\mathsf{id}_{1:i-1}} + \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} \bmod q\right)\right).$$

  Note that since the sibling of $\mathsf{id}_{1:i}$ is a challenge node, $\mathsf{id}_{1:i}$ cannot be a challenge node, i.e. $(b, b') \neq (\mathsf{id}_i, \oplus i)$.
  - Else, sample $\mathbf{v}_{1:i} \leftarrow\!\!\$\ \mathcal{R}_p^m$ and compute

$$\mathbf{u}_{\mathsf{id}_{1:i-1}} := -\mathbf{G}^{-1}\left(\mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} + \mathbf{B}_{\mathsf{id}_i, \oplus i} \cdot \mathbf{v}_{\mathsf{id}_{1:i}} \bmod q\right).$$

– Else (i.e. $\mathbf{u}_{\mathsf{id}_{1:i-1}}$ of the parent is set), if $\mathbf{v}_{\mathsf{id}_{1:i}}$ is not set, compute

$$\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow \mathsf{SampPre}\left(\mathsf{td}_{\mathsf{id}_i, \oplus i}, -\left(\mathbf{G} \cdot \mathbf{u}_{\mathsf{id}_{1:i-1}} + \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} \bmod q\right)\right).$$

  Note that $\mathsf{id}_{1:i}$ is not a challenge node, i.e. $(b, b') \neq (\mathsf{id}_i, \oplus i)$, since the parent node is labelled (Rule 2).

A more precise description of $\mathsf{KGen}\mathcal{O}^*_{b,b'}$ is given below. For notational convenience, we write the algorithm as sampling uniform terms upon every invocation, whereas the actual oracle will keep track of the randomness used in previous iterations to keep things consistent. Note that this makes the algorithm stateful, but it does not affect the scheme since it is only used in the security analysis.

```
$\mathsf{KGen}\mathcal{O}^*_{b,b'}(\mathsf{id})$
if $\mathbf{x}_{\mathsf{id}} = \bot$ then
  $\mathbf{x}_{\mathsf{id}} \leftarrow\!\!\$\ \mathcal{R}_p^m,\ \mathbf{y}_{\mathsf{id}} := \mathbf{B} \cdot \mathbf{x}_{\mathsf{id}} \bmod q,\ \mathbf{u}_{\mathsf{id}} := \mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}})$
for $i = \ell, \dots, 1$ do
  if $\mathbf{u}_{\mathsf{id}_{1:i-1}} = \bot$ then
    if $1 - \mathsf{id}_i = b \wedge \oplus i = b'$ then
      $s := \mathsf{sibling}(\mathsf{id}_{1:i})$
      $\mathbf{x}_s \leftarrow\!\!\$\ \mathcal{R}_p^m,\ \mathbf{y}_s := \mathbf{B} \cdot \mathbf{x}_s \bmod q,\ \mathbf{u}_s := -\mathbf{G}^{-1}(\mathbf{y}_s)$
      $\mathbf{v}_s \leftarrow\!\!\$\ \mathcal{R}_p^m$
      $\mathbf{y}_{\mathsf{id}_{1:i-1}} := \mathbf{A} \cdot \mathbf{u}_s + \mathbf{B}_{1-\mathsf{id}_i, \oplus i} \cdot \mathbf{v}_s \bmod q$
      $\mathbf{u}_{\mathsf{id}_{1:i-1}} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:i-1}})$
      $\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow \mathsf{SampPre}\left(\mathsf{td}_{\mathsf{id}_i, \oplus i}, \mathbf{y}_{\mathsf{id}_{1:i-1}} - \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} \bmod q\right)$
    else
      $\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow\!\!\$\ \mathcal{R}_p^m$
      $\mathbf{y}_{\mathsf{id}_{1:i-1}} := \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} + \mathbf{B}_{\mathsf{id}_i, \oplus i} \cdot \mathbf{v}_{\mathsf{id}_{1:i}} \bmod q$
      $\mathbf{u}_{\mathsf{id}_{1:i-1}} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{id}_{1:i-1}})$
  elseif $\mathbf{v}_{\mathsf{id}_{1:i}} = \bot$ then
    $\mathbf{v}_{\mathsf{id}_{1:i}} \leftarrow \mathsf{SampPre}\left(\mathsf{td}_{\mathsf{id}_i, \oplus i}, \mathbf{y}_{\mathsf{id}_{1:i-1}} - \mathbf{A} \cdot \mathbf{u}_{\mathsf{id}_{1:i}} \bmod q\right)$
$\mathbf{u}_{[\mathsf{id}]}^{\mathsf{T}} := (\mathbf{u}_{\mathsf{id}_1}^{\mathsf{T}}, \mathbf{v}_{\mathsf{id}_1}^{\mathsf{T}}, \mathbf{u}_{\mathsf{id}_{1:2}}^{\mathsf{T}}, \mathbf{v}_{\mathsf{id}_{1:2}}^{\mathsf{T}}, \dots, \mathbf{u}_{\mathsf{id}_{1:\ell}}^{\mathsf{T}}, \mathbf{v}_{\mathsf{id}_{1:\ell}}^{\mathsf{T}})$
return $\mathsf{sk}_{\mathsf{id}} := (\mathbf{u}_{[\mathsf{id}]}, \mathbf{x}_{\mathsf{id}})$
```

To conclude the analysis of this algorithm, the following Lemma shows that the distribution induced by invocations of $\mathsf{KGen}\mathcal{O}^*_{b,b'}$ is computationally close to that of the standard key generation oracle $\mathsf{KGen}\mathcal{O}$.

**Lemma 14.** *For all polynomials $q$, all $(b,b') \in \{0,1\}^2$, and all (unbounded) adversaries $\mathcal{A}$ making at most $q$ queries to the oracle, it holds that the following distributions are statistically close:*

$$\left\{\mathcal{A}^{\mathsf{KGen}\mathcal{O}}(\mathsf{mpk}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)\right\} \text{ and } \left\{\mathcal{A}^{\mathsf{KGen}\mathcal{O}^*_{b,b'}}(\mathsf{mpk}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)\right\}.$$

*Proof.* The statistical distance between each answer to the query is bounded by a negligible function $\mathsf{trap}(\lambda)$, by the definition of $\mathsf{TrapGen}$ and $\mathsf{SampPre}$. Thus, by a union bound, the statistical distance between the two experiment is bounded by $q \cdot \mathsf{trap}(\lambda)$, which is negligible. □

The following identities will prove to be useful when simulating ciphertexts. For all $j \in [\ell]$, it holds that

$$\begin{aligned}
\begin{pmatrix}\mathbf{A}\ \mathbf{B}_{\mathsf{id}_j,\oplus j}\\ \mathbf{G}\end{pmatrix} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} &= \begin{pmatrix}\mathbf{y}_{\mathsf{id}_{1:j-1}}\\ -\mathbf{y}_{\mathsf{id}_{1:j}}\end{pmatrix} \bmod q\\
\left(\mathbf{r}_{j-1}^{\mathsf{T}}\ \mathbf{r}_j^{\mathsf{T}}\right) \cdot \begin{pmatrix}\mathbf{A}\ \mathbf{B}_{\mathsf{id}_j,\oplus j}\\ \mathbf{G}\end{pmatrix} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} &= \left(\mathbf{r}_{j-1}^{\mathsf{T}}\ \mathbf{r}_j^{\mathsf{T}}\right) \cdot \begin{pmatrix}\mathbf{y}_{\mathsf{id}_{1:j-1}}\\ -\mathbf{y}_{\mathsf{id}_{1:j}}\end{pmatrix} \bmod q\\
\mathbf{c}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} &= \left(\mathbf{r}_{j-1}^{\mathsf{T}}\ \mathbf{r}_j^{\mathsf{T}}\right) \cdot \begin{pmatrix}\mathbf{y}_{\mathsf{id}_{1:j-1}}\\ -\mathbf{y}_{\mathsf{id}_{1:j}}\end{pmatrix} + \mathbf{e}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} \bmod q.
\end{aligned}$$

For any $k \in [\ell]$, summing the above for $j \in [k]$ yields the following:

$$\begin{aligned}
\sum_{j=1}^{k} \mathbf{c}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} &= \sum_{j=1}^{k} \left(\mathbf{r}_{j-1}^{\mathsf{T}}\ \mathbf{r}_j^{\mathsf{T}}\right) \cdot \begin{pmatrix}\mathbf{y}_{\mathsf{id}_{1:j-1}}\\ -\mathbf{y}_{\mathsf{id}_{1:j}}\end{pmatrix} + \sum_{j=1}^{k} \mathbf{e}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} \bmod q\\
&= \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_\epsilon - \mathbf{r}_k^{\mathsf{T}} \cdot \mathbf{y}_{\mathsf{id}_{1:k}} + \sum_{j=1}^{k} \mathbf{e}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} \bmod q\\
&= d - \mathbf{r}_k^{\mathsf{T}} \cdot \mathbf{y}_{\mathsf{id}_{1:k}} + \sum_{j=1}^{k} \mathbf{e}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} - e - \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q\\
d \approx \sum_{j=1}^{k} \mathbf{c}_{j-1}^{\mathsf{T}} \cdot \begin{pmatrix}\mathbf{u}_{\mathsf{id}_{1:j}}\\ \mathbf{v}_{\mathsf{id}_{1:j}}\end{pmatrix} &+ \mathbf{r}_k^{\mathsf{T}} \cdot \mathbf{y}_{\mathsf{id}_{1:k}} + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.
\end{aligned}$$

**Theorem 10.** *If $d_{\mathcal{R}} \geq \lambda$, $m \geq n \cdot \log_p q + \omega(\log \lambda)$, the $\mathsf{LWE}_{\mathcal{R},n,q,\chi}$ assumption holds, and $\mathsf{PRF}$ is a secure pseudorandom function, the IBE scheme in Fig. 7 is secure. More specifically, for every PPT adversary $\mathcal{A}$ against the security of the construction in Fig. 7, there exist PPT adversaries $\mathcal{A}_1$ against $\mathsf{elLWE}_{\mathcal{R},n,m,1,q,\chi,\bar{\chi},p/2}$, $\mathcal{A}_{\mathsf{LWE}}$ against $\mathsf{LWE}_{\mathcal{R},n,m+1,q,\chi}$ and $\mathcal{A}_{\mathsf{PRF}}$ agains the $\mathsf{PRF}$ such that*

$$\mathsf{adv}(\mathcal{A}) \geq \ell \cdot \mathsf{adv}(\mathcal{A}_1) + \mathsf{adv}(\mathcal{A}_{\mathsf{LWE}}) + \mathsf{adv}(\mathcal{A}_{\mathsf{PRF}}) + q \cdot \mathsf{trap}(\lambda)$$

*where $\mathsf{trap}$ is the statistical distance between $\mathsf{TrapGen}$ and $\mathsf{SampPre}$, and $q$ is the number of queries issued by the adversary.*

*Proof.* In the following let $\mathbf{y}_i^* = \mathbf{y}_{\mathsf{id}^*_{1:i}}$ for $0 \leq i \leq \ell$ and let $\mathsf{ctxt}^* = (\mathbf{c}_0, \ldots, \mathbf{c}_\ell, d)$ be the challenge ciphertext. Consider the following hybrids.

– $\mathcal{H}_0$: Identical to $\mathsf{Pseudorandomness}^0_{\Pi,\mathcal{A}}(1^\lambda, 1^\ell)$. Note that in this hybrid

$$\begin{aligned}
\mathbf{c}_{j-1}^{\mathsf{T}} &= \mathbf{r}_{j-1}^{\mathsf{T}} \cdot (\mathbf{A}\ \mathbf{B}_{\mathsf{id}^*_j,\oplus j}) + \mathbf{r}_j^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) + \mathbf{e}_{j-1}^{\mathsf{T}} \bmod q\ \forall j \in [\ell],\\
\mathbf{c}_\ell^{\mathsf{T}} &= \mathbf{r}_\ell^{\mathsf{T}} \cdot \mathbf{B} + \mathbf{e}_\ell^{\mathsf{T}} \bmod q, \text{ and}\\
d &= \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_0^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.
\end{aligned}$$

– $\mathcal{H}_1$: We modify the $\mathsf{KGen}$ oracle to use truly random coins for its subroutines but keep a state of previous invocations to keep things consistent.
– $\mathcal{H}_2$: Compute $\mathsf{ctxt}^*$ as follows. Choose $\mathbf{c}_0, \ldots, \mathbf{c}_{\ell-1} \leftarrow\$ \mathcal{R}_q^{2m}$ uniformly at random, choose $\mathbf{e}_0, \ldots, \mathbf{e}_{\ell-1} \leftarrow\$ \chi^{2m}$, and set

$$d = \sum_{j=1}^{\ell} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + \mathbf{r}_\ell^{\mathsf{T}} \cdot \mathbf{y}_\ell^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q,$$

where $\mathbf{w}_j = \begin{pmatrix} \mathbf{u}_{\mathsf{id}^*_{1:j}} \\ \mathbf{v}_{\mathsf{id}^*_{1:j}} \end{pmatrix} \in \mathcal{R}_p^{2m}$. The remaining term $\mathbf{c}_\ell$ is computed as in $\mathcal{H}_0$ and $\mathcal{H}_1$.
– $\mathcal{H}_3$: In this hybrid, we choose $\mathbf{c}_\ell \leftarrow\$ \mathcal{R}_q^m$ and $d \leftarrow\$ \mathcal{R}_q$ uniformly at random.

Note that in $\mathcal{H}_3$ all ciphertext components are chosen uniformly at random. Hence the claim of the theorem follows.

It is clear that for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_{\mathsf{PRF}}$ against the security of $\mathsf{PRF}$ and a PPT adversary $\mathcal{A}_{\mathsf{LWE}}$ against $\mathsf{LWE}_{\mathcal{R},n,m+1,q,\chi}$ such that

$$|\Pr[\mathcal{H}_0(\mathcal{A}) = 1] - \Pr[\mathcal{H}_1(\mathcal{A}) = 1]| \leq \mathsf{adv}(\mathcal{A}_{\mathsf{PRF}})$$

and

$$|\Pr[\mathcal{H}_2(\mathcal{A}) = 1] - \Pr[\mathcal{H}_3(\mathcal{A}) = 1]| \leq \mathsf{adv}(\mathcal{A}_{\mathsf{LWE}}).$$

It therefore remains to show that $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable, which we will do in Lemma 15. □

**Lemma 15.** *There exists a PPT adversary $\mathcal{A}_1$ against $\mathsf{elLWE}_{\mathcal{R},n,m,1,q,\chi,\bar{\chi},p/2}$ such that*

$$|\Pr[\mathcal{H}_1(\mathcal{A}) = 1] - \Pr[\mathcal{H}_2(\mathcal{A}) = 1]| \leq \ell \cdot \mathsf{adv}(\mathcal{A}_1) + q \cdot \mathsf{trap}(\lambda).$$

*where $q$ is the number of queries issued by the adversary and $\mathsf{trap}$ is a negligible function.*

*Proof.* To show that $\mathcal{H}_1$ and $\mathcal{H}_2$ are computationally indistinguishable, we define the following sub-hybrids $\mathcal{H}'_0, \ldots, \mathcal{H}'_\ell$ and $\mathcal{H}''_0, \ldots, \mathcal{H}''_\ell$.

– $\mathcal{H}'_i$ (for $i = 0, \ldots, \ell$): $\mathcal{H}'_0$ is identical to $\mathcal{H}_1$ and hybrid $\mathcal{H}'_\ell$ is identical to $\mathcal{H}_2$. For the middle cases, i.e. $1 \leq i < \ell$, we define hybrid $\mathcal{H}'_i$ so that $\mathbf{c}_0, \ldots, \mathbf{c}_{i-1}$ and $d$ are computed as in $\mathcal{H}_1$, and $\mathbf{c}_i, \ldots, \mathbf{c}_\ell$ are computed as in $\mathcal{H}_0$. Specifically, different from $\mathcal{H}_1$, we choose $\mathbf{c}_0, \ldots, \mathbf{c}_{i-1} \leftarrow\$ \mathcal{R}_q^{2m}$ uniformly at random, choose $\mathbf{e}_0, \ldots, \mathbf{e}_{i-1} \leftarrow\$ \chi^{2m}$ and set

$$d = \sum_{j=1}^{i} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + \mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{y}_i^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

– $\mathcal{H}''_i$ (for $i = 1, \ldots, \ell$): In this hybrid, we compute $\mathbf{c}_j$ for all $j \in [0 : \ell] \setminus \{ i-1 \}]$ as in $\mathcal{H}'_i$, and $\mathbf{c}_{i-1}$ is computed as follows. Choose $\hat{\mathbf{c}}_{i-1}$ uniformly at random and set

$$\mathbf{c}_{i-1}^{\mathsf{T}} = \hat{\mathbf{c}}_{i-1}^{\mathsf{T}} + \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q.$$

Furthermore, we set

$$d = \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + (\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

– $\mathcal{H}'''_i$ (for $i = 1, \ldots, \ell$): This hybrid is almost identical to $\mathcal{H}''_i$ except that the $\mathsf{KGen}\mathcal{O}$ oracle is replaced with the $\mathsf{KGen}\mathcal{O}^*_{\mathsf{id}^*_i,\oplus i}$ oracle.

First, observe that $\mathcal{H}'_i$ and $\mathcal{H}''_i$ are in fact identically distributed: In $\mathcal{H}''_i$, since $\hat{\mathbf{c}}_{i-1}$ is uniformly and independently distributed, we can equivalently compute it as

$$\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} = \bar{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0})$$

for a uniformly random and independent $\bar{\mathbf{c}}_{i-1}$. This makes $\mathbf{c}_{i-1} = \bar{\mathbf{c}}_{i-1}$ uniformly random, as in $\mathcal{H}'_i$. Substituting the new $\hat{\mathbf{c}}_i$ to the expression of $d$ in $\mathcal{H}''_i$, we have

$$\begin{aligned} d &= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + (\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + (\mathbf{c}_{i-1}^{\mathsf{T}} - \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=1}^{i} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j - \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \cdot \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=1}^{i} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + \mathbf{r}_i^{\mathsf{T}} \cdot \mathbf{y}_i^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q, \end{aligned}$$

as in $\mathcal{H}'_i$, where the last equality was due to $(\mathbf{G}\ \mathbf{0}) \cdot \mathbf{w}_i = \mathbf{G} \cdot \mathbf{u}_{\mathsf{id}^*_{1:i}} = -\mathbf{y}_i^*$.

Then, by Lemma 14 the hybrids $\mathcal{H}''_i$ and $\mathcal{H}'''_i$ are statistically close for all $i \in [\ell]$. Note that in $\mathcal{H}'''_i$ the matrix $\mathbf{B}_{\mathsf{id}^*_i,\oplus i}$ is uniformly random and the computation throughout $\mathcal{H}'''_i$ does not depend on the trapdoor $\mathsf{td}_{\mathsf{id}^*_i,\oplus i}$.

The main technical part of this proof lies in establishing indistinguishability between hybrids $\mathcal{H}'_{i-1}$ and $\mathcal{H}'''_i$ for $i \in [\ell]$. We will show that these two hybrids are indistinguishable under $\mathsf{elLWE}$. Assume towards contradiction that

$$\Pr\left[\mathcal{H}'_{i-1}(\mathcal{A}) = 1\right] - \Pr\left[\mathcal{H}'''_i(\mathcal{A}) = 1\right] \geq \epsilon.$$

We will show that this implies a PPT adversary $\mathcal{A}'_1$ against $\mathsf{elLWE}$ with advantage $\epsilon$.

The adversary $\mathcal{A}'_1$ is specified as follows. As input it receives a matrix $\mathbf{A}' \in \mathcal{R}_q^{n \times 2m}$, and it parses $\mathbf{A}'$ as $\mathbf{A}' = (\mathbf{A}\ \mathbf{B}_{\mathsf{id}^*_i,\oplus i})$ where $\mathbf{A}, \mathbf{B}_{\mathsf{id}^*_i,\oplus i} \in \mathcal{R}_q^{n \times m}$. Now $\mathcal{A}'_1$ simulates $\mathcal{H}'_i(\mathcal{A})$ with the matrices $\mathbf{A}, \mathbf{B}_{\mathsf{id}^*_i,\oplus i}$ thus obtained, until the adversary $\mathcal{A}$ queries the challenge ciphertext. Now it chooses $\mathbf{z}^* = -\mathbf{w}_i$ and sends $\mathbf{z}^*$ to its challenger. Note that $\mathbf{z}^*$ is a legit query as $\|\mathbf{w}_i\| \leq p/2$. Now $\mathcal{A}'_1$ obtaining a leak $l$ and $\mathbf{y}$. Next, it computes the challenge ciphertext as in $\mathcal{H}'_{i-1}(\mathcal{A})$, except that it sets

$$\mathbf{c}_{i-1} = \mathbf{y} + \mathbf{r}_i \cdot (\mathbf{G}\ \mathbf{0}) \bmod q$$

and

$$d = \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \mathbf{w}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q.$$

Note that the remaining ciphertext components are the same as in $\mathcal{H}'_{i-1}(\mathcal{A})$ and $\mathcal{H}'''_i(\mathcal{A})$. From there on, $\mathcal{A}'_1$ continues the simulation of $\mathcal{H}'_{i-1}(\mathcal{A})$ and outputs whatever $\mathcal{H}'_{i-1}(\mathcal{A})$ outputs.

Now let $b \in \{0,1\}$ be the challenge bit of the $\mathsf{elLWE}$ experiment. We claim that if $b = 0$, then $\mathcal{A}'_1$ faithfully simulates $\mathcal{H}'_{i-1}(\mathcal{A})$. On the other hand, we claim that for $b = 1$ the $\mathcal{A}'_1$ faithfully simulates $\mathcal{H}'''_i(\mathcal{A})$. From these two claims it follows that $\mathcal{A}'_1$ has advantage $\epsilon$.

– For $b = 0$, it holds that $\mathbf{y}^{\mathsf{T}} = \mathbf{r}^{\mathsf{T}} \cdot \mathbf{A}' + \mathbf{e}^{\mathsf{T}} = \mathbf{r}^{\mathsf{T}} \cdot (\mathbf{A}\ \mathbf{B}_{\mathsf{id}^*_i,\oplus i}) + \mathbf{e}^{\mathsf{T}} \bmod q$ and $l = \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}^* + e = -\mathbf{e}^{\mathsf{T}} \cdot \mathbf{w}_i + e \bmod q$. Renaming $\mathbf{r}$ to $\mathbf{r}_{i-1}$ and $\mathbf{e}$ to $\mathbf{e}_{i-1}$, it holds that

$$\begin{aligned} \mathbf{c}_{i-1}^{\mathsf{T}} &= \mathbf{y}^{\mathsf{T}} + \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \\ &= \mathbf{r}_{i-1}^{\mathsf{T}} \cdot (\mathbf{A}\ \mathbf{B}_{\mathsf{id}^*_i,\oplus i}) + \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) + \mathbf{e}_{i-1}^{\mathsf{T}} \bmod q \end{aligned}$$

and

$$d = \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \mathbf{w}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \tag{13}$$

$$= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + (\mathbf{r}_{i-1}^{\mathsf{T}} \cdot (\mathbf{A}\ \mathbf{B}_{\mathsf{id}_i^*, \oplus i}) + \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{w}_i - \mathbf{e}_{i-1}^{\mathsf{T}} \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \tag{14}$$

$$= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + \mathbf{r}_{i-1}^{\mathsf{T}} \cdot \mathbf{y}_{i-1}^* + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q, \tag{15}$$

where the last equality holds as $\mathbf{y}_{i-1}^* = (\mathbf{A}\ \mathbf{B}_{\mathsf{id}_i^*, \oplus i}) \cdot \mathbf{w}_i$. We can conclude that in this case the simulation of $\mathcal{A}_1'$ and $\mathcal{H}_{i-1}'(\mathcal{A})$ are identically distributed.

– For $b = 1$, it holds that $\mathbf{y} = \hat{\mathbf{c}}_{i-1}$ for a uniformly random $\hat{\mathbf{c}}_{i-1} \leftarrow\!\!\$\ \mathcal{R}_q^{2m}$ and $l = \mathbf{e}^{\mathsf{T}} \cdot \mathbf{z}^* + e = -\mathbf{e}^{\mathsf{T}} \mathbf{w}_i + e$. It therefore holds that

$$\begin{aligned} \mathbf{c}_{i-1}^{\mathsf{T}} &= \mathbf{y}^{\mathsf{T}} + \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \\ &= \hat{\mathbf{c}}_{i-1}^{\mathsf{T}} + \mathbf{r}_i^{\mathsf{T}} \cdot (\mathbf{G}\ \mathbf{0}) \bmod q \end{aligned}$$

and

$$\begin{aligned} d &= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j - \mathbf{y}^{\mathsf{T}} \cdot \mathbf{z}^* + l + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + \hat{\mathbf{c}}_{i-1}^{\mathsf{T}} \cdot \mathbf{w}_i - \mathbf{e}_{i-1}^{\mathsf{T}} \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q \\ &= \sum_{j=1}^{i-1} (\mathbf{c}_{j-1} - \mathbf{e}_{j-1})^{\mathsf{T}} \cdot \mathbf{w}_j + (\hat{\mathbf{c}}_{i-1}^{\mathsf{T}} - \mathbf{e}_{i-1}^{\mathsf{T}}) \cdot \mathbf{w}_i + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q. \end{aligned}$$

I.e. it holds that in this case the simulation of $\mathcal{A}_1'$ and $\mathcal{H}_i'''(\mathcal{A})$ are identically distributed. □

**Hybrid IBE/RBE.** We also mention that one can envision a hybrid IBE/RBE construction by combining the above scheme with the laconic encryption presented earlier in this paper. In this hybrid scheme, the key curator would insert master public keys in the tree, instead of regular keys. This would enable RBE where each user can delegate decryption of some predicates (specifically identity-based predicates) to other users. We leave the exact definition and construction of this notion as ground for future work.

## 12 Implementation

We implemented a prototype of the laconic encryption scheme from Section 7 in Go. We used the Lattigo [lat22] library for fast ring operations such as the number theoretic transform (NTT) for polynomial multiplications. Our code can be found in the following repository:

https://github.com/ahmadrezarahimi/laconic-encryption

The benchmarks have been conducted on a personal computer with an Intel Core i7-10700 3.8GHz CPU and 128GB of RAM running Ubuntu 22.04.1 LTS with kernel 5.15.0-48-generic.

|  | Time (milliseconds) |  |  |  |  | Size (MegaBytes) |  |  |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{R}_q$ | Setup | Upd | Enc | WGen | Dec | pp | ctxt | aux |
| $\mathbb{Z}_q[X]/(X^{256}+1)$ | 47 | 1200 | 2.85 | 0.048 | 6.00 | 8 | 49 | 367 |
| $\mathbb{Z}_q[X]/(X^{512}+1)$ | 85 | 1389 | 8.28 | 0.051 | 10.86 | 16 | 97 | 699 |
| $\mathbb{Z}_q[X]/(X^{1024}+1)$ | 167 | 1813 | 13.60 | 0.055 | 21.49 | 32 | 194 | 1370 |

**Table 1.** Benchmarks for our Laconic Encryption scheme. aux size depends on how many public keys are registered on the tree, while the rest depends only on the scheme specifications, such as ring type and $\ell$.

**Parameters.** We use the following parameters for our LWE assumption in the implementation: For the choice of the ring modulo, we set $q = 5 \cdot 2^{55} + 1$, which is a 58 bit NTT-friendly prime number, and we set $p = 2$. We ran the benchmarks for degrees $d = 256, 512, 1024$ (using the same notations as Section 7) with parameters:

$$\mathcal{R}_q \coloneqq \mathbb{Z}_q[X]/(X^d + 1) \qquad \mathcal{R}_p \coloneqq \mathbb{Z}_2[X]/(X^d + 1)$$
$$\ell = 50 \qquad n = 4 \qquad m = 232 \qquad \bar{m} = 512.$$

We sampled the errors from polynomials ring of degree $d \in \{256, 512, 1024\}$ where coefficients come from a truncated discrete Gaussian modulo $q$ with standard deviation $2^{30}$ and truncation factor of $2^{32}$.

**Benchmarks.** The benchmarks were run with three different rings, with $d \in \{256, 512, 1024\}$. Each time we ran the Setup and created an implicit binary Merkle Tree of depth $\ell$ and stored it in a SQLite database (we refer to the database as aux). We also did not include creating an empty database in our Setup time, as the choice of database might differ based on the application.

Then for the update algorithm, we uniformly sampled 1000 public-key and secret-key pairs $\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i\in[1000]}$ from KGen algorithm, and uniformly sampled 1000 indices $\{\mathsf{ind}_1, \cdots, \mathsf{ind}_{1000}\}$ where $\mathsf{ind}^i \in [0, 2^{50})$ for $i \in [1000]$ (Here by $\mathsf{ind}_i$ we refer to $i$-th index, not the $i$-th bit of ind). We ran the update algorithm on all $\{(\mathsf{pk}_i, i_i)\}_{i\in[1000]}$ and calculated the running time to create a tree with depth $\ell$ and updating 1000 leaves and their corresponding paths to the root. Finally, we set pp as the tree's root.

Using the public parameter, for each $\mathsf{id}_i$ we sampled a random message $\mathsf{msg} \leftarrow\!\$\ \mathcal{R}_p$ and encrypted msg toward $\mathsf{ind}_i$ with respect to the pp. As part of our implementation, we also provide the randomness and Gaussian noise as inputs to the encryption algorithm, and we do not include the generation time of the randomness and Gaussian noise in the time taken to perform the encryption. As soon as each encryption is completed, we immediately attempt to decrypt the encrypted message by calling the WGen algorithm on the corresponding ind then we run decryption on with corresponding sk and witnesses. The average run time of each algorithm (along with the sizes of the public parameters, ciphertexts, and auxiliary information) are displayed in Table 1.

We emphasize that the run time of Upd, Enc, Dec and WGen only depends on $\ell$, which is the depth of the tree, and not on the number of registered public-keys on the trees. Furthermore, these algorithms can be parallelized, as the computation for each layer of the tree does not depend on other layers. Hence, we used this feature and implemented a parallel version of each algorithm using the Goroutines, a feature of Go language. As a result, our scheme's run times are significantly improved, and its scalability is demonstrated for larger identity spaces.

## Acknowledgments

Nico Döttling and Chuanwei Lin: Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Giulio Malavolta and Ahmadreza Rahimi: This work was partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA   390781972. Dimitris Kolonelos: Received funding from projects from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), from the Spanish Government under project PRODIGY (TED2021-132464B-I00), and from the Madrid Regional Government under project BLOQUES (S2018/TCS-4339). The last two projects are co-funded by European Union EIE, and NextGenerationEU/PRTR funds.

## References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Heidelberg, August 2010.

ABD$^+$21. Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 94–125. Springer, Heidelberg, November 2021.

ACL$^+$22. Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 102–132. Springer, Heidelberg, August 2022.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996.

AL21. Martin R. Albrecht and Russell W. F. Lai. Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 519–548, Virtual Event, August 2021. Springer, Heidelberg.

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

ALOS22. Diego Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. Laconic private set-intersection from pairings. Cryptology ePrint Archive, Report 2022/529, 2022. https://eprint.iacr.org/2022/529.

AP12. Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 334–352. Springer, Heidelberg, May 2012.

BD20. Zvika Brakerski and Nico Döttling. Lossiness and entropic hardness for ring-LWE. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 1–27. Springer, Heidelberg, November 2020.

BDK$^+$17. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals – kyber: a cca-secure module-lattice-based kem. Cryptology ePrint Archive, Paper 2017/634, 2017. https://eprint.iacr.org/2017/634.

BdMW16. 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, Heidelberg, August 2016.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BJRW20. Katharina Boudgoust, Corentin Jeudy, Adeline Roux-Langlois, and Weiqiang Wen. Towards classical hardness of module-LWE: The linear rank case. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 289–317. Springer, Heidelberg, December 2020.

BL16. Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 404–434. Springer, Heidelberg, December 2016.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018.

BLSV18. Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018.

CDG$^+$17. Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017.

CF13. Dario Catalano and Dario Fiore. Vector commitments and their applications. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, Heidelberg, February / March 2013.

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May / June 2010.

DG17a. Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Heidelberg, November 2017.

DG17b. Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

DGGM19. Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In David Zuckerman, editor, *60th FOCS*, pages 661–685. IEEE Computer Society Press, November 2019.

DGHM18. Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 3–31. Springer, Heidelberg, March 2018.

DGI$^+$19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2019.

FNP04. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

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

GGH96. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. Cryptology ePrint Archive, Report 1996/009, 1996. https://eprint.iacr.org/1996/009.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 276–288. Springer, Heidelberg, August 1984.

GH18. Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 362–391. Springer, Heidelberg, August 2018.

GHM$^+$19. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 63–93. Springer, Heidelberg, April 2019.

GHMR18. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 689–718. Springer, Heidelberg, November 2018.

GKMR22. Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. Cryptology ePrint Archive, Paper 2022/1505, 2022. https://eprint.iacr.org/2022/1505.

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.

GS17. Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th FOCS*, pages 588–599. IEEE Computer Society Press, October 2017.

GS18a. Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 535–565. Springer, Heidelberg, April / May 2018.

GS18b. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018.

GV20. Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 621–651. Springer, Heidelberg, August 2020.

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

HLWW22. Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. Cryptology ePrint Archive, Paper 2022/1500, 2022. https://eprint.iacr.org/2022/1500.

lat22. Lattigo v4. Online: https://github.com/tuneinsight/lattigo, August 2022. EPFL-LDS, Tune Insight SA.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2016.

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, Heidelberg, April 2012.

MQR22. Mohammad Mahmoody, Wei Qi, and Ahmadreza Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. Cryptology ePrint Archive, Paper 2022/1285, 2022. https://eprint.iacr.org/2022/1285.

OPW11. Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 525–542. Springer, Heidelberg, August 2011.

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 80–97. Springer, Heidelberg, August 2010.

PPS21. Chris Peikert, Zachary Pepin, and Chad Sharp. Vector and functional commitments from lattices. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 480–511. Springer, Heidelberg, November 2021.

QWW18. Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

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.

Reg10. Oded Regev. The learning with errors problem (invited survey). In *2010 IEEE 25th Annual Conference on Computational Complexity*, pages 191–204, 2010.

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.

## A Laconic Oblivious Transfer

For completeness, we discuss how to construct laconic OT from a laconic encryption scheme $\Pi$ at essentially no cost. The idea is very simple: We sample one public key per database position $i$ and we insert it in position $2 \cdot \mathsf{ind}$ if the corresponding bit is $D_{\mathsf{ind}} = 0$ and in position $2 \cdot \mathsf{ind} - 1$ if the corresponding bit is $D_{\mathsf{ind}} = 1$. We sketch the algorithms in the following and for formal definitions, we refer the reader to [CDG+17].

– $\mathsf{OTSetup}$: The setup algorithm simply runs the setup $(\mathsf{pp}, \mathsf{st}, \mathsf{aux}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ of the laconic encryption.
– $\mathsf{OTReceive}$: On input a database $D \in \{0,1\}^d$, the receiver algorithm proceeds with the following subroutine. For $\mathsf{ind} = 1, \ldots, d$:
  • Sample $(\mathsf{pk}_{\mathsf{ind}}, \mathsf{sk}_{\mathsf{ind}}) \leftarrow \mathsf{KGen}(\mathsf{pp})$.
  • Insert the public key at position $2 \cdot \mathsf{ind} - D_{\mathsf{ind}}$ computing $\mathsf{pp} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, 2 \cdot \mathsf{ind} - D_{\mathsf{ind}}, \mathsf{pk}_{\mathsf{ind}})$ and generate the corresponding witness $\mathsf{wit}_{\mathsf{ind}} \leftarrow \mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, 2 \cdot \mathsf{ind} - D_{\mathsf{ind}}, \mathsf{pk}_{\mathsf{ind}})$.

  Finally, it sends the updated $\mathsf{pp}$.

- $\mathsf{OTSend}$: On input two messages $(\mathsf{msg}_0, \mathsf{msg}_1)$ and an index $\mathsf{ind}$, the sender algorithm computes $\mathsf{ctxt}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, 2 \cdot \mathsf{ind}, \mathsf{msg}_0)$ and $\mathsf{ctxt}_1 \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{st}, 2 \cdot \mathsf{ind} - 1, \mathsf{msg}_1)$ and sends both $(\mathsf{ctxt}_0, \mathsf{ctxt}_1)$ to the receiver.
- $\mathsf{OTDecode}$: The receiver selects $\mathsf{ctxt}_{D_\mathsf{ind}}$ and computes $\mathsf{msg}_{D_\mathsf{ind}} \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{sk}_\mathsf{ind}, \mathsf{wit}_\mathsf{ind}, \mathsf{ctxt}_{D_\mathsf{ind}})$.

It is not hard to see that the scheme satisfies semi-honest sender privacy, by the security of the laconic encryption scheme. Also note that the size of the receiver message $\mathsf{pp}$ is only bounded by a polynomial in $\lambda$ and in particular it is independent of the size of the database $D$. Furthermore, the size of the ciphertexts and the runtime of the decoding algorithm is also independent of the size of the database. Thus, we achieve the same asymptotics of the scheme in [CDG+17].

## A.1 Laconic Program Evaluation

The notion of laconic OT can be generalized to functions $f$ beyond projections, where the receiver hashes a database $D$, the sender encrypts a message $\mathsf{msg}$, and at the end of the interaction the receiver learns $\mathsf{msg}$ if and only if $f(D, x) = 1$, where $x$ is chosen by the sender. This is similar to the notion of laconic function evaluation (LFE) studied in [QWW18], except for an important difference: in LFE, the runtime of the decoding algorithm is proportional to the size of $D$, even if the function $f$ only depends on a few bits of $D$. In other words, LFE does not preserve *locality* in the access of the database, whereas laconic OT does. Here we sketch how one can obtain the best of both worlds, in a notion that we call *laconic program evaluation* (LPE).

**Decomposable LFE.** To describe our construction, it is necessary to recall a property of the LFE scheme from [QWW18], which we refer to as decomposability. Loosely speaking, this property says that the encryption algorithm can be split into subroutines, each depending on, at most, a single bit of $\mathsf{msg}$. Specifically, we require that the LFE ciphertext is computed as by running the following algorithm.

$$\mathsf{LFEEnc}(\mathsf{pp}, x, \mathsf{msg}) : \mathsf{LFEEnc}_1(\mathsf{pp}, x_1; \mathbf{r}), \dots, \mathsf{LFEEnc}_n(\mathsf{pp}, x_n; \mathbf{r}), \mathsf{LFEEnc}_{n+1}(\mathsf{pp}, \mathsf{msg}; \mathbf{r})$$

for some $\mathbf{r} \leftarrow\$ \{0,1\}^*$. It can be verified that the scheme presented in [QWW18] does indeed satisfy this syntactical requirement.[13]

**Construction of LPE.** We are now ready to sketch our construction of LPE, as a generic composition of a laconic OT and a decomposable LFE schemes. We present the description of the algorithms below.

- $\mathsf{LPESetup}$: The setup algorithm runs the setup of the laconic OT $(\mathsf{pp}, \mathsf{aux}) \leftarrow \mathsf{OTSetup}(1^\lambda)$ and of the LFE $\mathsf{pp}_f \leftarrow \mathsf{LFESetup}(1^\lambda, f)$.
- $\mathsf{LPEReceive}$: On input a database $D \in \{0,1\}^d$, the receiver algorithm updates the public parameters of the laconic OT $\mathsf{pp} \leftarrow \mathsf{OTReceive}(\mathsf{pp}, D)$.
- $\mathsf{LPESend}$: On input a message $\mathsf{msg}$ and an attribute $x \in \{0,1\}^\chi$, the algorithm samples some randomness $\mathbf{r} \leftarrow \{0,1\}^*$ and proceeds as follows. Let $\Delta$ be the set of bits of $D$ that are taken as input by $f(\cdot, x)$, and let $\delta = |\Delta|$. For all $i = 1, \dots, \delta$:
  - Compute $c_{i,0} \leftarrow \mathsf{LFEEnc}_i(\mathsf{pp}_f, 0; \mathbf{r})$ and $c_{i,0} \leftarrow \mathsf{LFEEnc}_i(\mathsf{pp}_f, 1; \mathbf{r})$.
  - Compute $e_i \leftarrow \mathsf{OTSend}(\mathsf{pp}, \Delta_i, c_{i,0}, c_{i,1})$.

  Then for all $i = 1, \dots, \chi$:
  - Compute $c_i \leftarrow \mathsf{LFEEnc}_{\delta+i}(\mathsf{pp}_f, x_i; \mathbf{r})$.

  The algorithms also computes $d \leftarrow \mathsf{LFEEnc}_{\delta+\chi+1}(\mathsf{pp}_f, \mathsf{msg}; \mathbf{r})$ and sends $(e_{1,0}, e_1, \dots, e_\delta, c_1, \dots, c_\chi, d)$ as the ciphertext.
- $\mathsf{LPEDecode}$: The receiver algorithm proceeds as follows. For all $i = 1, \dots, \delta$:
  - Compute $c'_i \leftarrow \mathsf{OTDecode}(\mathsf{pp}, e_i)$.

  Then it defines $\mathbf{c} := (c'_1, \dots, c'_\delta, c_1, \dots, c_\chi, d)$ and decrypts it using the $\mathsf{LFEDec}$ algorithm.

[13]The scheme is referred to as *attribute-based LFE* in [QWW18], but here we simply call it LFE to avoid an overload of notation.

It is not hard to verify that the construction satisfies the efficiency constraints that we discussed above. Furthermore, by the security of the laconic OT, the ciphertexts $c_{i,D_{\Delta_i}\oplus 1}$ are computationally hidden and therefore the view of the adversary consists only of $\mathbf{c}$, which is a well-formed LFE ciphertext. Thus, the adversary learns $\mathsf{msg}$ if and only if $f(D,x)=1$.

## B Alternative Laconic Encryption Constructions

Here we discuss alternative, to the one in Figure 2, constructions of laconic encryption (LE), which admit a better asymptotic behaviour by involving a trusted setup.

### B.1 Construction from Peikert et. al. Vector Commitment

The construction of Section 7 makes use of a binary Merkle tree with a lattice-based hash function in order to 'accumulate' the public keys. In this section we construct an LE scheme using a $d$-ary tree instead, where in place of the hash function we use a vector commitment [CF13]. This specialized tree was first described by Peikert et. al. [PPS21].[14] The use of vector commitments at each level of the tree allows for the witness to contain a single element per level, instead of the $d-1$ siblings a naive $d$-ary Merkle-tree would. Therefore, a witness consists of $\ell = \log_d(|\mathcal{IND}|) = \frac{|\mathcal{IND}|}{\log_2(d)}$ elements, where $d$ can be any polynomial in the security parameter and $\mathcal{IND}$ is the indices space. This saves a $\log(\lambda)$ factor in the ciphertexts' size.

On the downside, the specialized tree requires a private-coin trusted setup: matrices need to be generated with respect to SIS-trapdoors [GPV08,MP12]. Furthermore, the public parameters $\mathsf{pp}$ need to contain $O(d)$ matrices and the witness generation algorithm requires addional $O(d^2)$ matrices.

**Notation.** We introduce some additional notation for $d$-ary trees. Let $(\mathsf{ind}_1,\ldots,\mathsf{ind}_\ell)$ be the $d$-ary representation of $\mathsf{ind}$, i.e. $\mathsf{ind} = \sum_{i=1}^{\ell} \mathsf{ind}_i d^{i-1}$. We denote $\mathsf{ind}_j$ the $j$-th $d$-digit of $\mathsf{ind}$ and more generally $\mathsf{ind}_{1:j}$ the first $j$ $d$-digits of $\mathsf{ind}$, where trivially $\mathsf{ind}_{1:1} := \mathsf{ind}_1$ and $\mathsf{ind}_{1:\ell} := \mathsf{ind}$. For instance, for $\mathsf{ind} = (342)_5$ (the number 273 in digital) we have $\mathsf{ind}_2 = 4, \mathsf{ind}_{1:1} = 3, \mathsf{ind}_{1:2} = 34, \mathsf{ind}_{1:3} = \mathsf{ind} = 342$.

***d*-ary specialized trees.** To ease the presentation of our construction we briefly recall here the [PPS21]-based specialized tree.

At the core of the construction is an SIS-based vector commitment [PPS21] that works as follows. At the setup phase, $d$ random matrices with respective SIS trapdoors [GPV08,MP12] $\{\mathbf{T}_i \in \mathcal{R}_q^{n\times k}\}_{i\in\mathbb{Z}_d}$ and $d$ uniformly matrices $\{\mathbf{A}_i \in \mathcal{R}_q^{n\times m}\}_{i\in\mathbb{Z}_d}$ are generated. Also, using the SIS trapdoors, $d^2-d$ matrices $\{\mathbf{R}_{i,j} \in \mathcal{R}_p^{k\times m}\}_{i\in\mathbb{Z}_d, j\in\mathbb{Z}_d, i\neq j}$ are computed such that $\mathbf{T}_i\mathbf{R}_{i,j} = \mathbf{A}_j$. The hash of $d$ elements $\mathbf{u}_i \in \mathcal{R}_p^m$ is computed as:

$$\boldsymbol{\gamma} \leftarrow \sum_{i=0}^{d-1} \mathbf{A}_i\mathbf{u}_i$$

Then in order to verify that an element $\mathbf{u}_i$ was correctly hashed in $\boldsymbol{\gamma}$ one needs a single value $\mathbf{p}_i \in \mathcal{R}_q^k$ (instead of the $d-1$ siblings $\{\mathbf{u}_j\}_{j\in\mathbb{Z}_d, j\neq i}$). This is done as follows:

$$\boldsymbol{\gamma} \overset{?}{=} \mathbf{T}_i\mathbf{p}_i + \mathbf{A}_i\mathbf{u}_i, \quad \text{where } \mathbf{p}_i \leftarrow \sum_{j=0, j\neq i}^{d-1} \mathbf{R}_{i,j}\mathbf{u}_j$$

The $d$-ary tree construction recursively computes the hash of $d^\ell$ leaves by using the above vector commitment as a hash at each level. Since $\boldsymbol{\gamma} \in \mathcal{R}_q^n$ does not match the input space of the vector commitment,

[14]Peikert et. al. [PPS21] construct a slightly more involved tree construction, towards achieving an additional property, stateless updatability. We do not need the aforementioned property, therefore we use a simpler tree construction.

$\mathcal{R}_p^m$, it is mapped to it by using the gadget matrix, $\mathbf{u} \leftarrow -\mathbf{G}^{-1}(\boldsymbol{\gamma})$. Overall a witness for the leaf $\mathsf{ind}$ is the path to the root, together with the corresponding $\mathbf{p}$ values:

$$\mathsf{wit} := \left(\mathbf{u}_{\mathsf{ind}_{1:1}}, \mathbf{p}_{\mathsf{ind}_{1:1}}, \mathbf{u}_{\mathsf{ind}_{1:2}}, \mathbf{p}_{\mathsf{ind}_{1:2}}, \ldots, \mathbf{u}_{\mathsf{ind}_{1:\ell}}, \mathbf{p}_{\mathsf{ind}_{1:\ell}}\right)$$

and the verification that $\mathbf{y}_{\mathsf{ind}}$) is the $\mathsf{ind}$'th leaf:

$$\begin{aligned}
\mathbf{T}_{\mathsf{ind}_1} \cdot \mathbf{p}_{\mathsf{ind}_{1:1}} + \mathbf{A}_1 \cdot \mathbf{u}_{\mathsf{ind}_{1:1}} &= \mathbf{y}_\epsilon \\
-\mathbf{G} \cdot \mathbf{u}_{\mathsf{ind}_{1:1}} &= \mathbf{y}_{\mathsf{ind}_{1:1}} \\
\mathbf{T}_{\mathsf{ind}_2} \cdot \mathbf{p}_{\mathsf{ind}_{1:2}} + \mathbf{A}_2 \cdot \mathbf{u}_{\mathsf{ind}_{1:2}} &= \mathbf{y}_{\mathsf{ind}_{1:1}} \\
-\mathbf{G} \cdot \mathbf{u}_{\mathsf{ind}_{1:2}} &= \mathbf{y}_{\mathsf{ind}_{1:2}} \\
&\vdots \\
\mathbf{T}_{\mathsf{ind}_\ell} \cdot \mathbf{p}_{\mathsf{ind}_{1:\ell}} + \mathbf{A}_\ell \cdot \mathbf{u}_{\mathsf{ind}_{1:\ell}} &= \mathbf{y}_{\mathsf{ind}_{1:\ell-1}} \\
-\mathbf{G} \cdot \mathbf{u}_{\mathsf{ind}_{1:\ell}} &= \mathbf{y}_{\mathsf{ind}}
\end{aligned}$$

**Our construction.** Similarly to our construction of Section 7, we have the lattice parameters $n, m, p, q \in \mathbb{N}$ and a distribution $\chi$ over $\mathcal{R}$. Each public and secret key-pair is $(\mathbf{x}, \mathbf{y}) \in \mathcal{R}_p^m \times \mathcal{R}_q^n$ such that $\mathbf{B} \cdot \mathbf{x} = \mathbf{y} \bmod q$, where $\mathbf{B} \in \mathcal{R}_q^m$ is a public uniformly sampled matrix. To encrypt a mesasge $\mathsf{msg}$ for the index $\mathsf{ind}$ we apply dual-Regev to $(\mathbf{A}_{\mathsf{ind}}, \mathbf{v})$, where:

$$\mathbf{A}_{\mathsf{ind}} := \begin{pmatrix}
\mathbf{T}_{\mathsf{ind}_1} & \mathbf{A}_{\mathsf{ind}_1} & & & & & & \\
\mathbf{0} & \mathbf{G} & \mathbf{T}_{\mathsf{ind}_2} & \mathbf{A}_{\mathsf{ind}_2} & & & & \\
& & \mathbf{0} & \mathbf{G} & & & & \\
& & & & \ddots & \ddots & & \\
& & & & & \mathbf{T}_{\mathsf{ind}_\ell} & \mathbf{A}_{\mathsf{ind}_\ell} & \\
& & & & & \mathbf{0} & \mathbf{G} & \mathbf{B}
\end{pmatrix} \quad \text{and} \quad \mathbf{v} := \begin{pmatrix} \mathbf{y}_\epsilon \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}$$

Finally for this construction it suffices that there is at least one terminator value $\mathbf{u}^* = -\mathbf{G}^{-1}\mathbf{y}^*$ at each hash computed. For this, we append an artificial input to the hash, using a vector commitment of $d+1$ inputs where the position $d$ is always reserved for the terminator value.

Our alternative laconic encryption scheme is formally described in Figure 8.

### B.2 Construction from Albrecht et. al. Vector Commitment

Albrecht et. al. [ACL$^+$22] recently introduced a vector commitment based on a new lattice assumption that can virtually achieve optimal public-parameters overhead. That is, the public parameters generated in the setup phase are $O(d)$ (in contrast to $O(d^2)$ of [PPS21]).

Therefore, we can construct a $d$-ary Merkle tree in the exact spirit to the above: using the lattice-based vector commitment of Albrecht et. al. as a hash, at each level. This gives us a laconic encryption scheme with $\frac{|\mathcal{IND}|}{\log_2(d)}$-sized ciphertext and an optimal, $O(d)$, storage overhead for the parameters of the Key Curator.

```
$\mathsf{Setup}(1^\lambda)$
$\mathbf{T}_0, \ldots, \mathbf{T}_d \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times k}$   // Generate with trapdoors
$\mathbf{A}_0, \ldots, \mathbf{A}_d, \mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}$
$\mathbf{R}_{i,j} \in \mathbb{Z}^{k \times m} : \mathbf{T}_i \mathbf{R}_{i,j} = \mathbf{A}_j$   // Using the trapdoors
$\mathbf{y}_\epsilon := \mathbf{y}^* \leftarrow\!\!\$\ \mathbb{Z}_q^n$
$\mathcal{T} := \{\epsilon\}$
$\mathsf{pp} := \big(\{\mathbf{A}_i\}_{i \in \mathbb{Z}_{d+1}}, \{\mathbf{T}_i\}_{i \in \mathbb{Z}_{d+1}}, \{\mathbf{R}_{i,j}\}_{i,j \in \mathbb{Z}_{d+1}, i \neq j}, \mathbf{B}, \mathbf{y}^*\big)$
$\mathsf{st} := \mathbf{y}_\epsilon$
$\mathsf{aux} := (\mathcal{T}, \{\mathbf{y}_v\}_{v \in \mathcal{T}})$
return $(\mathsf{pp}, \mathsf{st}, \mathsf{aux})$

$\mathsf{KGen}(\mathsf{pp})$
$\mathbf{x} \leftarrow\!\!\$\ \mathbb{Z}_p^m$
$\mathbf{y} := \mathbf{B} \cdot \mathbf{x} \bmod q$
return $(\mathsf{pk}, \mathsf{sk}) := (\mathbf{y}, \mathbf{x})$

$\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
if $\mathsf{pk} = \bot$ then $\mathcal{T} := \mathcal{T} \setminus \{\mathsf{ind}\}$
else
  $\mathcal{T} := \mathcal{T} \cup \{\mathsf{ind}\}$
  $\mathbf{y}_{\mathsf{ind}} := \mathsf{pk}$
$\mathsf{st}' \leftarrow \mathsf{TreeUpdate}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
return $\mathsf{st}'$

$\mathsf{TreeUpdate}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind})$
for $j = \ell, \ldots, 1$ do
  if $(\mathsf{ind}_{1:j} \| i) \notin \mathcal{T},\ \forall i \in \mathbb{Z}_d$ then
    $\mathcal{T} := \mathcal{T} \setminus \{\mathsf{ind}_{1:j}\}$
  else
    if $(\mathsf{ind}_{1:j} \| d) \notin \mathcal{T}$ then
      $\mathbf{y}_{\mathsf{ind}_{1:j} \| d} := \mathbf{y}^*$
    $\mathbf{u}_{\mathsf{ind}_{1:j} \| i} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{ind}_{1:j} \| i}),\ \forall i \in \mathbb{Z}_{d+1}$
    $\mathcal{T} := \mathcal{T} \cup \{\mathsf{ind}_{1:j}\}$
    $\mathbf{y}_{\mathsf{ind}_{1:j}} := \sum_{i=0}^{d} \mathbf{A}_i \mathbf{u}_{\mathsf{ind}_{1:j} \| i} \bmod q$
return $\mathsf{st}$
```

```
$\mathsf{Enc}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{msg})$
$\mathbf{r}_j \leftarrow\!\!\$\ \mathbb{Z}_q^n,\ \forall j \in \{1, \ldots, \ell+1\}$
for $j = 1, \ldots, \ell$ do
  $\mathbf{e}_j \leftarrow\!\!\$\ \chi^{2m}$
  $\mathbf{B}_j := \begin{pmatrix} \mathbf{T}_{\mathsf{ind}_j} & \mathbf{A}_{\mathsf{ind}_j} \\ 0 & \mathbf{G} \end{pmatrix}$
  $\mathbf{c}_j^{\mathsf{T}} := (\mathbf{r}_{j-1}^{\mathsf{T}}, \mathbf{r}_j^{\mathsf{T}}) \cdot \mathbf{B}_j + \mathbf{e}_j^{\mathsf{T}} \bmod q$
$\mathbf{e}_\ell \leftarrow\!\!\$\ \chi^m,\ e \leftarrow\!\!\$\ \chi$
$\mathbf{c}_{\ell+1}^{\mathsf{T}} := \mathbf{r}_\ell^{\mathsf{T}} \cdot \mathbf{B} + \mathbf{e}_{\ell+1}^{\mathsf{T}} \bmod q$
$c_\epsilon := \mathbf{r}_0^{\mathsf{T}} \cdot \mathbf{y}_\epsilon + e + \left\lfloor \frac{q}{2} \right\rfloor \cdot \mathsf{msg} \bmod q$
return $\mathsf{ctxt} := (\mathbf{c}_1, \ldots, \mathbf{c}_{\ell+1}, c_\epsilon)$

$\mathsf{WGen}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{st}, \mathsf{ind}, \mathsf{pk})$
for $j = \ell, \ldots, 1$ do
  $\mathsf{prefix} := \mathsf{ind}_{1:j-1}$
  $\mathsf{MSB} := \mathsf{ind}_j$
  $\mathbf{u}_{\mathsf{prefix} \| i} := -\mathbf{G}^{-1}(\mathbf{y}_{\mathsf{prefix} \| i}),\ \forall i \in \mathbb{Z}_d$
  $\mathbf{p}_{\mathsf{ind}_{1:j}} := \sum_{i=0, i \neq \mathsf{MSB}}^{d} \mathbf{R}_{\mathsf{MSB}, i} \mathbf{u}_{\mathsf{prefix} \| i}$
return $\mathsf{wit} := \left(\mathbf{u}_{\mathsf{ind}_{1:j}}, \mathbf{p}_{\mathsf{ind}_{1:j}}\right)_{j=1}^{\ell}$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{wit}, \mathsf{ctxt})$
parse $\mathsf{sk}$ as $\mathbf{x}_{\mathsf{ind}}$
parse $\mathsf{wit} := (\mathbf{u}_{\mathsf{ind}_{1:1}}, \mathbf{p}_{\mathsf{ind}_{1:1}}, \ldots, \mathbf{u}_{\mathsf{ind}_{1:\ell}}, \mathbf{p}_{\mathsf{ind}_{1:\ell}})$
$\bar{\mu} := c_\epsilon - \sum_{j=1}^{\ell} \mathbf{c}_j^{\mathsf{T}} \cdot \begin{pmatrix} \mathbf{p}_{\mathsf{ind}_{1:j}} \\ \mathbf{u}_{\mathsf{ind}_{1:j}} \end{pmatrix} - \mathbf{c}_{\ell+1}^{\mathsf{T}} \cdot \mathbf{x}_{\mathsf{ind}} \bmod q$
if $|\bar{\mu}| < q/4$ then return $0$
else return $1$
```

**Fig. 8.** Alternative construction of laconic encryption.