Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller

Daniele Micciancio* Chris Peikert†

September 14, 2011

Abstract

We give new methods for generating and using "strong trapdoors" in cryptographic lattices, which are simultaneously simple, efficient, easy to implement (even in parallel), and asymptotically optimal with very small hidden constants. Our methods involve a new kind of trapdoor, and include specialized algorithms for inverting LWE, randomly sampling SIS preimages, and securely delegating trapdoors. These tasks were previously the main bottleneck for a wide range of cryptographic schemes, and our techniques substantially improve upon the prior ones, both in terms of practical performance and quality of the produced outputs. Moreover, the simple structure of the new trapdoor and associated algorithms can be exposed in applications, leading to further simplifications and efficiency improvements. We exemplify the applicability of our methods with new digital signature schemes and CCA-secure encryption schemes, which have better efficiency and security than the previously known lattice-based constructions.

1 Introduction

Cryptography based on lattices has several attractive and distinguishing features:

- On the *security* front, the best attacks on the underlying problems require exponential $2^{\Omega(n)}$ time in the main security parameter n, even for quantum adversaries. By constrast, for example, mainstream factoring-based cryptography can be broken in subexponential $2^{\tilde{O}(n^{1/3})}$ time classically, and even in polynomial $n^{O(1)}$ time using quantum algorithms. Moreover, lattice cryptography is supported by strong worst-case/average-case security reductions, which provide solid theoretical evidence that the random instances used in cryptography are indeed asymptotically hard, and do not suffer from any unforeseen "structural" weaknesses.

*University of California, San Diego. Email: textttdaniele@cs.ucsd.edu. This material is based on research sponsored by DARPA under agreement number FA8750-11-C-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

†School of Computer Science, College of Computing, Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under Grant CNS-0716786 and CAREER Award CCF-1054495, by the Alfred P. Sloan Foundation, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under Contract No. FA8750-11-C-0098. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the National Science Foundation, the Sloan Foundation, DARPA or the U.S. Government.

- On the *efficiency* and *implementation* fronts, lattice cryptography operations can be extremely simple, fast and parallelizable. Typical operations are the selection of uniformly random integer matrices $\mathbf{A}$ modulo some small $q = \mathrm{poly}(n)$, and the evaluation of simple linear functions like

$$f_{\mathbf{A}}(\mathbf{x}) := \mathbf{A}\mathbf{x} \bmod q \quad \text{and} \quad g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$$

  on short integer vectors $\mathbf{x}$, $\mathbf{e}$.[1] (For commonly used parameters, $f_{\mathbf{A}}$ is surjective while $g_{\mathbf{A}}$ is injective.) Often, the modulus q is small enough that all the basic operations can be directly implemented using machine-level arithmetic. By contrast, the analogous operations in number-theoretic cryptography (e.g., generating huge random primes, and exponentiating modulo such primes) are much more complex, admit only limited parallelism in practice, and require the use of "big number" arithmetic libraries.

In recent years lattice-based cryptography has also been shown to be extremely versatile, leading to a large number of theoretical applications ranging from (hierarchical) identity-based encryption [GPV08, CHKP10, ABB10a, ABB10b], to fully homomorphic encryption schemes [Gen09b, Gen09a, vGHV10, BV11b, BV11a, GH11, BGV11], and much more (e.g., [LM08, PW08, Lyu08, PV08, PVW08, Pei09b, ACPS09, Rüc10, Boy10, GHV10, GKV10]).

Not all lattice cryptography is as simple as selecting random matrices $\mathbf{A}$ and evaluating linear functions like $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x} \bmod q$, however. In fact, such operations yield only collision-resistant hash functions, public-key encryption schemes that are secure under passive attacks, and little else. Richer and more advanced lattice-based cryptographic schemes, including chosen ciphertext-secure encryption, "hash-and-sign" digital signatures, and identity-based encryption also require generating a matrix $\mathbf{A}$ together with some *"strong" trapdoor*, typically in the form of a nonsingular square matrix (a basis) $\mathbf{S}$ of short integer vectors such that $\mathbf{A}\mathbf{S} = \mathbf{0} \bmod q$. (The matrix $\mathbf{S}$ is usually interpreted as a basis of a lattice defined by using $\mathbf{A}$ as a "parity check" matrix.) Applications of such strong trapdoors also require certain efficient inversion algorithms for the functions $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$, using $\mathbf{S}$. Appropriately inverting $f_{\mathbf{A}}$ can be particularly complex, as it typically requires sampling *random preimages* of $f_{\mathbf{A}}(\mathbf{x})$ according to a Gaussian-like probability distribution (see [GPV08]).

Theoretical solutions for all the above tasks (generating $\mathbf{A}$ with strong trapdoor $\mathbf{S}$ [Ajt99, AP09], trapdoor inversion of $g_{\mathbf{A}}$ and preimage sampling for $f_{\mathbf{A}}$ [GPV08]) are known, but they are rather complex and not very suitable for practice, in either runtime or the "quality" of their outputs. (The quality of a trapdoor $\mathbf{S}$ roughly corresponds to the Euclidean lengths of its vectors — shorter is better.) The current best method for trapdoor generation [AP09] is conceptually and algorithmically complex, and involves costly computations of Hermite normal forms and matrix inverses. And while the dimensions and quality of its output are *asymptotically* optimal (or nearly so, depending on the precise notion of quality), the hidden constant factors are rather large. Similarly, the standard methods for inverting $g_{\mathbf{A}}$ and sampling preimages of $f_{\mathbf{A}}$ [Bab85, Kle00, GPV08] are inherently sequential and time-consuming, as they are based on an orthogonalization process that uses high-precision real numbers. A more efficient and parallelizable method for preimage sampling (which uses only small-integer arithmetic) has recently been discovered [Pei10], but it is still more complex than is desirable for practice, and the quality of its output can be slightly worse than that of the sequential algorithm when using the same trapdoor $\mathbf{S}$.

More compact and efficient trapdoors appear necessary for bringing advanced lattice-based schemes to practice, not only because of the current unsatisfactory runtimes, but also because the concrete security of lattice cryptography can be quite sensitive to even small changes in the main parameters. As already

[1] Inverting these functions corresponds to solving the "short integer solution" (SIS) problem [Ajt96] for $f_{\mathbf{A}}$, and the "learning with errors" (LWE) problem [Reg05] for $g_{\mathbf{A}}$, both of which are widely used in lattice cryptography and enjoy provable worst-case hardness.

mentioned, two central objects are a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that serves as a public key, and an associated secret matrix $\mathbf{S} \in \mathbb{Z}^{m \times m}$ consisting of short integer vectors having "quality" s, where smaller is better. Here n is the main security parameter governing the hardness of breaking the functions, and m is the dimension of a lattice associated with $\mathbf{A}$, which is generated by the vectors in $\mathbf{S}$. Note that the security parameter n and lattice dimension m need not be the same; indeed, typically we have $m = \Theta(n \lg q)$, which for many applications is optimal up to constant factors. (For simplicity, throughout this introduction we use the base-2 logarithm; other choices are possible and yield tradeoffs among the parameters.) For the trapdoor quality, achieving $s = O(\sqrt{m})$ is asymptotically optimal, and random preimages of $f_{\mathbf{A}}$ generated using $\mathbf{S}$ have Euclidean length $\beta \approx s\sqrt{m}$. For security, it must be hard (*without* knowing the trapdoor) to find any preimage having length bounded by β. Interestingly, the computational resources needed to do so can increase dramatically with only a moderate decrease in the bound β (see, e.g., [GN08, MR09]). Therefore, improving the parameters m and s by even small constant factors can have a significant impact on concrete security. Moreover, this can lead to a "virtuous cycle" in which the increased security allows for the use of a smaller security parameter n, which leads to even smaller values of m, s, and β, etc. Note also that the schemes' key sizes and concrete runtimes are reduced as well, so improving the parameters yields a "win-win-win" scenario of simultaneously smaller keys, increased concrete security, and faster operations. (This phenomenon is borne out concretely; see Figure 2.)

1.1 Contributions

The first main contribution of this paper is a new method of trapdoor generation for cryptographic lattices, which is simultaneously simple, efficient, easy to implement (even in parallel), and asymptotically optimal with small hidden constants. The new trapdoor generator strictly subsumes the prior ones of [Ajt99, AP09], in that it proves the main theorems from those works, but with improved concrete bounds for all the relevant quantities (simultaneously), and via a conceptually simpler and more efficient algorithm. To accompany our trapdoor generator, we also give specialized algorithms for trapdoor inversion (for $g_{\mathbf{A}}$) and preimage sampling (for $f_{\mathbf{A}}$), which are simpler and more efficient in our setting than the prior general solutions [Bab85, Kle00, GPV08, Pei10].

Our methods yield large constant-factor improvements, and in some cases even small asymptotic improvements, in the lattice dimension m, trapdoor quality s, and storage size of the trapdoor. Because trapdoor generation and inversion algorithms are the main operations in many lattice cryptography schemes, our algorithms can be plugged in as 'black boxes' to deliver significant concrete improvements in all such applications. Moreover, it is often possible to expose the special (and very simple) structure of our trapdoor directly in cryptographic schemes, yielding additional improvements and potentially new applications. (Below we summarize a few improvements to existing applications, with full details in Section 6.)

We now give a detailed comparison of our results with the most relevant prior works [Ajt99, AP09, GPV08, Pei10]. The quantitative improvements are summarized in Figure 1.

Simpler, faster trapdoor generation and inversion algorithms. Our trapdoor generator is exceedingly simple, especially as compared with the prior constructions [Ajt99, AP09]. It essentially amounts to just one multiplication of two random matrices, whose entries are chosen independently from appropriate probability distributions. Surprisingly, this method is nearly identical to Ajtai's original method [Ajt96] of generating a random lattice together with a "weak" trapdoor of one or more short vectors (but *not* a full basis), with one added twist. And while there are no detailed runtime analyses or public implementations of [Ajt99, AP09], it is clear from inspection that our new method is significantly more efficient, since it does not involve any expensive Hermite normal form or matrix inversion computations.

Our specialized, parallel inversion algorithms for $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ are also simpler and more practically efficient than the general solutions of [Bab85, Kle00, GPV08, Pei10] (though we note that our trapdoor generator is entirely compatible with those general algorithms as well). In particular, we give the first *parallel* algorithm for inverting $g_{\mathbf{A}}$ under asymptotically optimal error rates (previously, handling such large errors required the sequential "nearest-plane" algorithm of [Bab85]), and our preimage sampling algorithm for $f_{\mathbf{A}}$ works with smaller integers and requires much less offline storage than the one from [Pei10].

Tighter parameters. To generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that is within negligible statistical distance of uniform, our new trapdoor construction improves the lattice dimension from $m > 5n \lg q$ [AP09] down to $m \approx 2n \lg q$. (In both cases, the base of the logarithm is a tunable parameter that appears as a multiplicative factor in the quality of the trapdoor; here we fix upon base 2 for concreteness.) In addition, we give the first known *computationally pseudorandom* construction (under the LWE assumption), where the dimension can be as small as $m = n(1 + \lg q)$, although at the cost of an $\Omega(\sqrt{n})$ factor worse quality s.

Our construction also greatly improves the quality s of the trapdoor. The best prior construction [AP09] produces a basis whose Gram-Schmidt quality (i.e., the maximum length of its Gram-Schmidt orthogonalized vectors) was loosely bounded by $20\sqrt{n \lg q}$. However, the Gram-Schmidt notion of quality is useful only for less efficient, sequential inversion algorithms [Bab85, GPV08] that use high-precision real arithmetic. For the more efficient, parallel preimage sampling algorithm of [Pei10] that uses small-integer arithmetic, the parameters guaranteed by [AP09] are asymptotically worse, at $m > n \lg^2 q$ and $s \geq 16\sqrt{n \lg^2 q}$. By contrast, our (statistically secure) trapdoor construction achieves the "best of both worlds:" asymptotically optimal dimension $m \approx 2n \lg q$ and quality $s \approx 1.6\sqrt{n \lg q}$ or better, with a parallel preimage sampling algorithm that is slightly more efficient than the one of [Pei10].

Altogether, for any n and typical values of $q \geq 2^{16}$, we conservatively estimate that the new trapdoor generator and inversion algorithms collectively provide at least a $7 \lg q \geq 112$-*fold improvement* in the length bound $\beta \approx s\sqrt{m}$ for $f_{\mathbf{A}}$ preimages (generated using an efficient algorithm). We also obtain similar improvements in the size of the error terms that can be handled when efficiently inverting $g_{\mathbf{A}}$.

New, smaller trapdoors. As an additional benefit, our construction actually produces a *new kind of trapdoor* — not a basis — that is at least 4 times smaller in storage than a basis of corresponding quality, and is at least as powerful, i.e., a good basis can be efficiently derived from the new trapdoor. We stress that our specialized inversion algorithms using the new trapdoor provide almost exactly the same quality as the inefficient, sequential algorithms using a derived basis, so there is no trade-off between efficiency and quality. (This is in contrast with [Pei10] when using a basis generated according to [AP09].) Moreover, the storage size of the new trapdoor grows only linearly in the lattice dimension m, rather than quadratically as a basis does. This is most significant for applications like hierarchical ID-based encryption [CHKP10, ABB10a] that *delegate* trapdoors for increasing values of m. The new trapdoor also admits a very simple and efficient delegation mechanism, which unlike the prior method [CHKP10] does not require any costly operations like linear independence tests, or conversions from a full-rank set of lattice vectors into a basis. In summary, the new type of trapdoor and its associated algorithms are *strictly preferable* to a short basis in terms of algorithmic efficiency, output quality, and storage size (simultaneously).

Ring-based constructions. Finally, and most importantly for practice, all of the above-described constructions and algorithms extend immediately to the *ring* setting, where functions analogous to $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ require only quasi-linear $\tilde{O}(n)$ space and time to specify and evaluate (respectively), which is a factor of $\tilde{\Omega}(n)$ improvement over the matrix-based functions defined above. See the representative works [Mic02, PR06, LM06, LMPR08, LPR10] for more details on these functions and their security foundations.

	[Ajt99, AP09] constructions	This work (fast $f_{\mathbf{A}}^{-1}$)	Factor Improvement
Dimension m	slow $f_{\mathbf{A}}^{-1}$ [Kle00, GPV08]: $> 5n \lg q$ fast $f_{\mathbf{A}}^{-1}$ [Pei10]: $> n \lg^2 q$	$2n \lg q$ $(\overset{s}{\approx})$ $n(1 + \lg q)$ $(\overset{c}{\approx})$	$2.5 - \lg q$
Quality s	slow $f_{\mathbf{A}}^{-1}$: $\approx 20\sqrt{n \lg q}$ fast $f_{\mathbf{A}}^{-1}$: $\approx 16\sqrt{n \lg^2 q}$	$\approx 1.6\sqrt{n \lg q}$ $(\overset{s}{\approx})$	$12.5 - 10\sqrt{\lg q}$
Length $\beta \approx s\sqrt{m}$	slow $f_{\mathbf{A}}^{-1}$: $> 45n \lg q$ fast $f_{\mathbf{A}}^{-1}$: $> 16n \lg^2 q$	$\approx 2.3\, n \lg q$ $(\overset{s}{\approx})$	$19 - 7 \lg q$

Figure 1: Summary of parameters for our constructions and algorithms versus prior ones. In the column labelled "this work," $\overset{s}{\approx}$ and $\overset{c}{\approx}$ denote constructions producing public keys $\mathbf{A}$ that are statistically close to uniform, and computationally pseudorandom, respectively. (All quality terms s and length bounds β omit the same statistical "smoothing" factor for $\mathbb{Z}$, which is about 4–5 in practice.)

To illustrate the kinds of concrete improvements that our methods provide, in Figure 2 we give representative parameters for the canonical application of GPV sigantures [GPV08], comparing the old and new trapdoor constructions for nearly equal levels of concrete security. We stress that these parameters are not highly optimized, and making adjustments to some of the tunable parameters in our constructions may provide better combinations of efficiency and concrete security. We leave this effort for future work.

1.2 Techniques

The main idea behind our new method of trapdoor generation is as follows. Instead of building a random matrix $\mathbf{A}$ through some specialized and complex process, we start from a carefully crafted *public* matrix $\mathbf{G}$ (and its associated lattice), for which the associated functions $f_{\mathbf{G}}$ and $g_{\mathbf{G}}$ admit very efficient (in both sequential and parallel complexity) and high-quality inversion algorithms. In particular, preimage sampling for $f_{\mathbf{G}}$ and inversion for $g_{\mathbf{G}}$ can be performed in essentially $O(n \log n)$ sequential time, and can even be performed by n parallel $O(\log n)$-time operations or table lookups. (This should be compared with the general algorithms for these tasks, which require at least quadratic $\Omega(n^2 \log^2 n)$ time, and are not always parallelizable for optimal noise parameters.) We emphasize that $\mathbf{G}$ is *not* a cryptographic key, but rather a fixed and public matrix that may be used by all parties, so the implementation of all its associated operations can be highly optimized, in both software and hardware. We also mention that the simplest and most practically efficient choices of $\mathbf{G}$ work for a modulus q that is a power of a small prime, such as $q = 2^k$, but a crucial search/decision reduction for LWE was not previously known for such q, despite its obvious practical utility. In Section 3 we provide a very general reduction that covers this case and others, and subsumes all of the known (and incomparable) search/decision reductions for LWE [BFKL93, Reg05, Pei09b, ACPS09].

To generate a *random* matrix $\mathbf{A}$ with a trapdoor, we take two additional steps: first, we extend $\mathbf{G}$ into a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{G}]$, for uniform $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ and sufficiently large $\bar{m}$. (As shown in [CHKP10], inversion of $g_{\mathbf{A}'}$ and preimage sampling for $f_{\mathbf{A}'}$ reduce very efficiently to the corresponding tasks for $g_{\mathbf{G}}$ and $f_{\mathbf{G}}$.) Finally, we simply apply to $\mathbf{A}'$ a certain random unimodular transformation defined by the matrix $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$, for a random "short" secret matrix $\mathbf{R}$ that will serve as the trapdoor, to obtain

$$\mathbf{A} = \mathbf{A}' \cdot \mathbf{T} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

	[AP09] with fast $f_{\mathbf{A}}^{-1}$	This work	Factor Improvement
Sec param n	436	284	1.5
Modulus q	2^{32}	2^{24}	256
Dimension m	446,644	13,812	32.3
Quality s	10.7×10^3	418	25.6
Length β	12.9×10^6	91.6×10^3	141
Key size (bits)	6.22×10^9	92.2×10^6	$\mathbf{67.5}$
Key size (ring-based)	$\approx 16 \times 10^6$	$\approx 361 \times 10^3$	$\approx \mathbf{44.3}$

Figure 2: Representative parameters for GPV signatures (using fast inversion algorithms) for the old and new trapdoor generation methods. Using the methodology from [MR09], both sets of parameters have security level corresponding to a parameter δ of at most 1.007, which is estimated to require about 2^{46} core-years on a 64-bit 1.86GHz Xeon using the state-of-the-art in lattice basis reduction [GN08, CN11]. We use a smoothing parameter of $r = 4.5$ for $\mathbb{Z}$, which corresponds to statistical error of less than 2^{-90} for each randomized-rounding operation during signing. Key sizes are calculated using the Hermite normal form optimization. Key sizes for *ring-based* GPV signatures are approximated to be smaller by a factor of about $0.9n$.

The transformation given by $\mathbf{T}$ has the following properties:

- It is very easy to compute and invert, requiring essentially just one multiplication by $\mathbf{R}$ in both cases. (Note that $\mathbf{T}^{-1} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$.)
- It results in a matrix $\mathbf{A}$ that is distributed essentially uniformly at random, as required by the security reductions (and worst-case hardness proofs) for lattice-based cryptographic schemes.
- For the resulting functions $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$, preimage sampling and inversion very simply and efficiently reduce to the corresponding tasks for $f_{\mathbf{G}}$, $g_{\mathbf{G}}$. The overhead of the reduction is essentially just a single matrix-vector product with the secret matrix $\mathbf{R}$ (which, when inverting $f_{\mathbf{A}}$, can largely be precomputed even before the target value is known).

As a result, the cost of the inversion operations ends up being very close to that of computing $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$ in the forward direction. Moreover, the fact that the running time is dominated by matrix-vector multiplications with the *fixed* trapdoor matrix $\mathbf{R}$ yields theoretical (but asymptotically significant) improvements in the context of batch execution of several operations relative to the same secret key $\mathbf{R}$: instead of evaluating several products $\mathbf{Rz}_1, \mathbf{Rz}_2, \ldots, \mathbf{Rz}_n$ individually at a total cost of $\Omega(n^3)$, one can employ fast matrix multiplication techniques to evaluate $\mathbf{R}[\mathbf{z}_1, \ldots, \mathbf{z}_n]$ as a whole is subcubic time. Batch operations can be exploited in applications like the multi-bit IBE of [GPV08] and its extensions to HIBE [CHKP10, ABB10a, ABB10b].

Related techniques. At the surface, our trapdoor generator appears similar to the original "GGH" approach of [GGH97] for generating a lattice together with a short basis. That technique works by choosing some random short vectors as the secret "good basis" of a lattice, and then transforms them into a public "bad basis" for the *same* lattice, via a unimodular matrix having large entries. (Note, though, that this does not produce a lattice from Ajtai's worst-case-hard family.) A closer look reveals, however, that (worst-case hardness aside) our method is actually *not* an instance of the GGH paradigm: here the initial short basis of the lattice

defined by $\mathbf{G}$ (or the semi-random matrix $[\bar{\mathbf{A}}|\mathbf{G}]$) is *fixed* and *public*, while the random unimodular matrix $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$ actually *produces a new lattice* by applying a (reversible) linear transformation to the original lattice. In other words, in contrast with GGH we multiply a (short) unimodular matrix on the "other side" of the original short basis, thus changing the lattice it generates.

A more appropriate comparison is to Ajtai's original method [Ajt96] for generating a random $\mathbf{A}$ together with a "weak" trapdoor of one or more short lattice vectors (but *not* a full basis). There, one simply chooses a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{0}]$ and outputs $\mathbf{A} = \mathbf{A}' \cdot \mathbf{T} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$, with short vectors $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$. Perhaps surprisingly, our strong trapdoor generator is just a simple twist on Ajtai's original weak generator, replacing $\mathbf{0}$ with the gadget $\mathbf{G}$.

Our constructions and inversion algorithms also draw upon several other techniques from throughout the literature. The trapdoor basis generator of [AP09] and the LWE-based "lossy" injective trapdoor function of [PW08] both use a fixed "gadget" matrix analogous to $\mathbf{G}$, whose entries grow geometrically in a structured way. In both cases, the gadget is concealed (either statistically or computationally) in the public key by a small combination of uniformly random vectors. Our method for adding tags to the trapdoor is very similar to a technique for doing the same with the lossy TDF of [PW08], and is identical to the method used in [ABB10a] for constructing compact (H)IBE. Finally, in our preimage sampling algorithm for $f_{\mathbf{A}}$, we use the "convolution" technique from [Pei10] to correct for some statistical skew that arises when converting preimages for $f_{\mathbf{G}}$ to preimages for $f_{\mathbf{A}}$, which would otherwise leak information about the trapdoor $\mathbf{R}$.

1.3 Applications

Our improved trapdoor generator and inversion algorithms can be plugged into any scheme that uses such tools as a "black box," and the resulting scheme will inherit all the efficiency improvements. (Every application we know of admits such a black-box replacement.) Moreover, the special properties of our methods allow for further improvements to the design, efficiency, and security reductions of existing schemes. Here we summarize some representative improvements that are possible to obtain; see Section 6 for complete details.

Hash-and-sign digital signatures. Our construction and supporting algorithms plug directly into the "full domain hash" signature scheme of [GPV08], which is strongly unforgeable in the random oracle model, with a tight security reduction. One can even use our computationally secure trapdoor generator to obtain a smaller public verification key, though at the cost of a hardness-of-LWE assumption, and a somewhat stronger SIS assumption (which affects concrete security). Determining the right balance between key size and security is left for later work.

In the standard model, there are two closely related types of hash-and-sign signature schemes:

- The one of [CHKP10], which has signatures of bit length $\tilde{O}(n^2)$, and is existentially unforgeable (later improved to be strongly unforgeable [Rüc10]) assuming the hardness of inverting $f_{\mathbf{A}}$ with solution length bounded by $\beta = \tilde{O}(n^{1.5})$.[2]
- The scheme of [Boy10], a lattice analogue of the pairing-based signature of [Wat05], which has signatures of bit length $\tilde{O}(n)$ and is existentially unforgeable assuming the hardness of inverting $f_{\mathbf{A}}$ with solution length bounded by $\beta = \tilde{O}(n^{3.5})$.

We improve the latter scheme in several ways, by: *(i)* improving the length bound to $\beta = \tilde{O}(n^{2.5})$; *(ii)* reducing the online runtime of the signing algorithm from $\tilde{O}(n^3)$ to $\tilde{O}(n^2)$ via chameleon hashing [KR00]; *(iii)* making the scheme strongly unforgeable *a la* [GPV08, Rüc10]; *(iv)* giving a tighter and simpler security reduction

[2] All parameters in this discussion assume a message length of $\tilde{\Theta}(n)$ bits.

(using a variant of the "prefix technique" [HW09] as in [CHKP10]), where the reduction's advantage degrades only linearly in the number of signature queries; and *(v)* removing all additional constraints on the parameters n and q (aside from those needed to ensure hardness of the SIS problem). We stress that the scheme itself is essentially the same (up to the improved and generalized parameters, and chameleon hashing) as that of [Boy10]; only the security proof and underlying assumption are improved. Note that in comparison with [CHKP10], there is still a trade-off between the bit length of the signatures and the bound β in the underlying SIS assumption; this appears to be inherent to the style of the security reduction. Note also that the public keys in all of these schemes are still rather large at $\tilde{O}(n^3)$ bits (or $\tilde{O}(n^2)$ bits using the ring analogue of SIS), so they are still mainly of theoretical interest. Improving the key sizes of standard-model signatures is an important open problem.

Chosen ciphertext-secure encryption. We give a new construction of CCA-secure public-key encryption (in the standard model) from the learning with errors (LWE) problem with error rate $\alpha = 1/\operatorname{poly}(n)$, where larger α corresponds to a harder concrete problem. Existing schemes exhibit various incomparable tradeoffs between key size and error rate. The first such scheme is due to [PW08]: it has public keys of size $\tilde{O}(n^2)$ bits (with somewhat large hidden factors) and relies on a quite small LWE error rate of $\alpha = \tilde{O}(1/n^4)$. The next scheme, from [Pei09b], has larger public keys of $\tilde{O}(n^3)$ bits, but uses a better error rate of $\alpha = \tilde{O}(1/n)$. Finally, using the generic conversion from selectively secure ID-based encryption to CCA-secure encryption [BCHK07], one can obtain from [ABB10a] a scheme having key size $\tilde{O}(n^2)$ bits and using error rate $\alpha = \tilde{O}(1/n^2)$. (Here decryption is randomized, since the IBE key-derivation algorithm is.) In particular, the public key of the scheme from [ABB10b] consists of 3 matrices in $\mathbb{Z}_q^{n \times m}$ where m is large enough to embed a (strong) trapdoor, plus essentially one vector in $\mathbb{Z}_q^n$ per message bit.

We give a CCA-secure system that enjoys the best of all prior constructions, which has $\tilde{O}(n^2)$-bit public keys, uses error rate $\alpha = \tilde{O}(1/n)$ (both with small hidden factors), and has deterministic decryption. To achieve this, we need to go beyond just plugging our improved trapdoor generator as a black box into prior constructions. Our scheme relies on the particular structure of the trapdoor instances; in effect, we directly construct a "tag-based adaptive trapdoor function" [KMO10]. The public key consists of only 1 matrix with an embedded (strong) trapdoor, rather than 3 as in the most compact scheme to date [ABB10a]; moreover, we can encrypt up to $n \log q$ message bits per ciphertext without needing any additional public key material. Combining these design changes with the improved dimension of our trapdoor generator, we obtain more than a 7.5-fold improvement in the public key size as compared with [ABB10a]. (This figure does not account for removing the extra public key material for the message bits, nor the other parameter improvements implied by our weaker concrete LWE assumption, which would shrink the keys even further.)

(Hierarchical) identity-based encryption. Just as with signatures, our constructions plug directly into the random-oracle IBE of [GPV08]. In the standard-model depth-d hierarchical IBEs of [CHKP10, ABB10a], our techniques can shrink the public parameters by an additional factor of about $\frac{2+4d}{1+d} \in [3, 4]$, relative to just plugging our improved trapdoor generator as a "black box" into the schemes. This is because for each level of the hierarchy, the public parameters only need to contain one matrix of the same dimension as $\mathbf{G}$ (i.e., about $n \lg q$), rather than two full trapdoor matrices (of dimension about $2n \lg q$ each).[3] Because the adaptation is straightforward given the tools developed in this work, we omit the details.

[3] We note that in [Pei09a] (an earlier version of [CHKP10]) the schemes are defined in a similar way using lower-dimensional extensions, rather than full trapdoor matrices at each level.

1.4 Other Related Work

Concrete parameter settings for a variety "strong" trapdoor applications are given in [RS10]. Those parameters are derived using the previous suboptimal generator of [AP09], and using the methods from this work would yield substantial improvements. The recent work of [LP11] also gives improved key sizes and concrete security for LWE-based cryptosystems; however, that work deals only with IND-CPA-secure encryption, and not at all with strong trapdoors or the further applications they enable (CCA security, digital signatures, (H)IBE, etc.).

2 Preliminaries

We denote the real numbers by $\mathbb{R}$ and the integers by $\mathbb{Z}$. For a nonnegative integer k, we let $[k] = \{1, \ldots, k\}$. Vectors are denoted by lower-case bold letters (e.g., $\mathbf{x}$) and are always in column form ($\mathbf{x}^t$ is a row vector). We denote matrices by upper-case bold letters, and treat a matrix $\mathbf{X}$ interchangeably with its ordered set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots\}$ of column vectors. For convenience, we sometimes use a scalar s to refer to the scaled identity matrix $s\mathbf{I}$, where the dimension will be clear from context.

The statistical distance between two distributions X, Y over a finite or countable domain D is $\Delta(X, Y) = \frac{1}{2}\sum_{w \in D} |X(w) - Y(w)|$. Statistical distance is a metric, and in particular obeys the triangle inequality. We say that a distribution over D is ϵ-uniform if its statistical distance from the uniform distribution is at most ϵ.

Throughout the paper, we use a "randomized-rounding parameter" r that we let be a fixed function $r(n) = \omega(\sqrt{\log n})$ growing asymptotically faster than $\sqrt{\log n}$. By "fixed function" we mean that $r = \omega(\sqrt{\log n})$ always refers to the very same function, and no other factors will be absorbed into the $\omega(\cdot)$ notation. This allows us to keep track of the precise multiplicative constants introduced by our constructions. Concretely, we take $r \approx \sqrt{\ln(2/\epsilon)/\pi}$ where ϵ is a desired bound on the statistical error introduced by each randomized-rounding operation for $\mathbb{Z}$, because the error is bounded by $\approx 2\exp(-\pi r^2)$ according to Lemma 2.3 below. For example, for $\epsilon = 2^{-54}$ we have $r \leq 3.5$, and for $\epsilon = 2^{-71}$ we have $r \leq 4$.

2.1 Linear Algebra

A unimodular matrix $\mathbf{U} \in \mathbb{Z}^{m \times m}$ is one for which $|\det(\mathbf{U})| = 1$; in particular, $\mathbf{U}^{-1} \in \mathbb{Z}^{m \times m}$ as well. The *Gram-Schmidt orthogonalization* of an ordered set of vectors $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \in \mathbb{R}^n$, is $\widetilde{\mathbf{V}} = \{\widetilde{\mathbf{v}}_1, \ldots, \widetilde{\mathbf{v}}_k\}$ where $\widetilde{\mathbf{v}}_i$ is the component of $\mathbf{v}_i$ orthogonal to $\mathrm{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{i-1})$ for all $i = 1, \ldots, k$. (In some cases we orthogonalize the vectors in a different order.) In matrix form, $\mathbf{V} = \mathbf{QDU}$ for some orthogonal $\mathbf{Q} \in \mathbb{R}^{n \times k}$, diagonal $\mathbf{D} \in \mathbb{R}^{k \times k}$ with nonnegative entries, and upper unitriangular $\mathbf{U} \in \mathbb{R}^{k \times k}$ (i.e., $\mathbf{U}$ is upper triangular with 1s on the diagonal). The decomposition is unique when the $\mathbf{v}_i$ are linearly independent, and we always have $\|\widetilde{\mathbf{v}}_i\| = d_{i,i}$, the ith diagonal entry of $\mathbf{D}$.

For any basis $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of $\mathbb{R}^n$, its origin-centered parallelepiped is defined as $\mathcal{P}_{1/2}(\mathbf{V}) = \mathbf{V} \cdot [-\frac{1}{2}, \frac{1}{2})^n$. Its dual basis is defined as $\mathbf{V}^* = \mathbf{V}^{-t} = (\mathbf{V}^{-1})^t$. If we orthogonalize $\mathbf{V}$ and $\mathbf{V}^*$ in forward and reverse order, respectively, then we have $\widetilde{\mathbf{v}_i^*} = \widetilde{\mathbf{v}}_i / \|\widetilde{\mathbf{v}}_i\|^2$ for all i. In particular, $\|\widetilde{\mathbf{v}_i^*}\| = 1/\|\widetilde{\mathbf{v}}_i\|$.

For any square real matrix $\mathbf{X}$, the *(Moore-Penrose) pseudoinverse*, denoted $\mathbf{X}^+$, is the unique matrix satisfying $(\mathbf{X}\mathbf{X}^+)\mathbf{X} = \mathbf{X}$, $\mathbf{X}^+(\mathbf{X}\mathbf{X}^+) = \mathbf{X}^+$, and such that both $\mathbf{X}\mathbf{X}^+$ and $\mathbf{X}^+\mathbf{X}$ are symmetric. We always have $\mathrm{span}(\mathbf{X}) = \mathrm{span}(\mathbf{X}^+)$, and when $\mathbf{X}$ is invertible, we have $\mathbf{X}^+ = \mathbf{X}^{-1}$.

A symmetric matrix $\Sigma \in \mathbb{R}^{n \times n}$ is *positive definite* (respectively, positive *semi*definite), written $\Sigma > \mathbf{0}$ (resp., $\Sigma \geq \mathbf{0}$), if $\mathbf{x}^t \Sigma \mathbf{x} > 0$ (resp., $\mathbf{x}^t \Sigma \mathbf{x} \geq 0$) for all nonzero $\mathbf{x} \in \mathbb{R}^n$. We have $\Sigma > \mathbf{0}$ if and only if Σ is invertible and $\Sigma^{-1} > \mathbf{0}$, and $\Sigma \geq \mathbf{0}$ if and only if $\Sigma^+ \geq \mathbf{0}$. Positive (semi)definiteness defines a partial

ordering on symmetric matrices: we say that $\Sigma_1 > \Sigma_2$ if $(\Sigma_1 - \Sigma_2) > \mathbf{0}$, and similarly for $\Sigma_1 \geq \Sigma_2$. We have $\Sigma_1 \geq \Sigma_2 \geq \mathbf{0}$ if and only if $\Sigma_2^+ \geq \Sigma_1^+ \geq \mathbf{0}$, and likewise for the analogous strict inequalities.

For any matrix $\mathbf{B}$, the symmetric matrix $\Sigma = \mathbf{B}\mathbf{B}^t$ is positive semidefinite, because

$$\mathbf{x}^t \Sigma \mathbf{x} = \langle \mathbf{B}^t\mathbf{x}, \mathbf{B}^t\mathbf{x} \rangle = \|\mathbf{B}^t\mathbf{x}\|^2 \geq 0$$

for any nonzero $\mathbf{x} \in \mathbb{R}^n$, where the inequality is always strict if and only if $\mathbf{B}$ is nonsingular. We say that $\mathbf{B}$ is a *square root* of $\Sigma > \mathbf{0}$, written $\mathbf{B} = \sqrt{\Sigma}$, if $\mathbf{B}\mathbf{B}^t = \Sigma$. Every $\Sigma \geq \mathbf{0}$ has a square root, which can be computed efficiently, e.g., via the Cholesky decomposition.

For any matrix $\mathbf{B} \in \mathbb{R}^{n \times k}$, there exists a *singular value decomposition* $\mathbf{B} = \mathbf{Q}\mathbf{D}\mathbf{P}^t$, where $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{P} \in \mathbb{R}^{k \times k}$ are orthogonal matrices, and $\mathbf{D} \in \mathbb{R}^{n \times k}$ is a diagonal matrix with nonnegative entries $s_i \geq 0$ on the diagonal, in non-increasing order. The s_i are called the *singular values* of $\mathbf{B}$. Under this convention, $\mathbf{D}$ is uniquely determined (though $\mathbf{Q}, \mathbf{P}$ may not be), and $s_1(\mathbf{B}) = \max_{\mathbf{u}} \|\mathbf{B}\mathbf{u}\| = \max_{\mathbf{u}} \|\mathbf{B}^t\mathbf{u}\| \geq \|\mathbf{B}\|, \|\mathbf{B}^t\|$, where the maxima are taken over all unit vectors $\mathbf{u} \in \mathbb{R}^k$.

2.2 Lattices and Hard Problems

Generally defined, an m-dimensional *lattice* Λ is a discrete additive subgroup of $\mathbb{R}^m$. For some $k \leq m$, called the *rank* of the lattice, Λ is generated as the set of all $\mathbb{Z}$-linear combinations of some k linearly independent *basis* vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$, i.e., $\Lambda = \{\mathbf{B}\mathbf{z} : \mathbf{z} \in \mathbb{Z}^k\}$. In this work, we are mostly concerned with full-rank integer lattices, i.e., $\Lambda \subseteq \mathbb{Z}^m$ with $k = m$. (We work with non-full-rank lattices only in the analysis of our Gaussian sampling algorithm in Section 5.4.) The dual lattice Λ^* is the set of all $\mathbf{v} \in \mathrm{span}(\Lambda)$ such that $\langle \mathbf{v}, \mathbf{x} \rangle \in \mathbb{Z}$ for every $\mathbf{x} \in \Lambda$. If $\mathbf{B}$ is a basis of Λ, then $\mathbf{B}^* = \mathbf{B}(\mathbf{B}^t\mathbf{B})^{-1}$ is a basis of Λ^*. Note that when Λ is full-rank, $\mathbf{B}$ is invertible and hence $\mathbf{B}^* = \mathbf{B}^{-t}$.

Many cryptographic applications use a particular family of so-called q*-ary* integer lattices, which contain $q\mathbb{Z}^m$ as a sublattice for some (typically small) integer q. For positive integers n and q, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary and define the following full-rank m-dimensional q-ary lattices:

$$\begin{aligned} \Lambda^\perp(\mathbf{A}) &= \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \bmod q\} \\ \Lambda(\mathbf{A}^t) &= \{\mathbf{z} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{z} = \mathbf{A}^t\mathbf{s} \bmod q\}. \end{aligned}$$

It is easy to check that $\Lambda^\perp(\mathbf{A})$ and $\Lambda(\mathbf{A}^t)$ are dual lattices, up to a q scaling factor: $q \cdot \Lambda^\perp(\mathbf{A})^* = \Lambda(\mathbf{A}^t)$, and vice-versa. For this reason, it is sometimes more natural to consider the non-integral, "1-ary" lattice $\frac{1}{q}\Lambda(\mathbf{A}^t) = \Lambda^\perp(\mathbf{A})^* \supseteq \mathbb{Z}^m$. For any $\mathbf{u} \in \mathbb{Z}_q^n$ admitting an integral solution to $\mathbf{A}\mathbf{x} = \mathbf{u} \bmod q$, define the coset (or "shifted" lattice)

$$\Lambda_{\mathbf{u}}^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \bmod q\} = \Lambda^\perp(\mathbf{A}) + \mathbf{x}.$$

Here we recall some basic facts about these q-ary lattices.

Lemma 2.1. *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary and let $\mathbf{S} \in \mathbb{Z}^{m \times m}$ be any basis of $\Lambda^\perp(\mathbf{A})$.*

1. *For any unimodular $\mathbf{T} \in \mathbb{Z}^{m \times m}$, we have $\mathbf{T} \cdot \Lambda^\perp(\mathbf{A}) = \Lambda^\perp(\mathbf{A} \cdot \mathbf{T}^{-1})$, with $\mathbf{T} \cdot \mathbf{S}$ as a basis.*
2. *[ABB10a, implicit] For any invertible $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, we have $\Lambda^\perp(\mathbf{H} \cdot \mathbf{A}) = \Lambda^\perp(\mathbf{A})$.*
3. *[CHKP10, Lemma 3.2] Suppose that the columns of $\mathbf{A}$ generate all of $\mathbb{Z}_q^n$, let $\mathbf{A}' \in \mathbb{Z}_q^{n \times m'}$ be arbitrary, and let $\mathbf{W} \in \mathbb{Z}^{m \times m'}$ be an arbitrary solution to $\mathbf{A}\mathbf{W} = -\mathbf{A}' \bmod q$. Then $\mathbf{S}' = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{smallmatrix}\right]$ is a basis of $\Lambda^\perp([\mathbf{A}' \mid \mathbf{A}])$, and when orthogonalized in appropriate order, $\widetilde{\mathbf{S}'} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{S}} \end{smallmatrix}\right]$. In particular, $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$.*

Cryptographic problems. For $\beta > 0$, the *short integer solution* problem $\mathsf{SIS}_{q,\beta}$ is an average-case version of the approximate shortest vector problem on $\Lambda^{\perp}(\mathbf{A})$. The problem is: given uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for any desired $m = \mathrm{poly}(n)$, find a relatively short nonzero $\mathbf{z} \in \Lambda^{\perp}(\mathbf{A})$, i.e., output a nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{z} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\| \leq \beta$. When $q \geq \beta\sqrt{n} \cdot \omega(\sqrt{\log n})$, solving this problem (with any non-negligible probability over the random choice of $\mathbf{A}$) is at least as hard as (probabilistically) approximating the Shortest Independent Vectors Problem (SIVP, a classic problem in the computational study of point lattices [MG02]) on n-dimensional lattices to within $\tilde{O}(\beta\sqrt{n})$ factors in the *worst case* [Ajt96, MR04, GPV08].

For $\alpha > 0$, the *learning with errors* problem $\mathsf{LWE}_{q,\alpha}$ may be seen an average-case version of the bounded-distance decoding problem on the dual lattice $\frac{1}{q}\Lambda(\mathbf{A}^t)$. Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, the additive group of reals modulo 1, and let D_α denote the Gaussian probability distribution over $\mathbb{R}$ with parameter α (see Section 2.3 below). For any fixed $\mathbf{s} \in \mathbb{Z}_q^n$, define $A_{\mathbf{s},\alpha}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ obtained by choosing $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow D_\alpha$, and outputting $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle / q + e \bmod 1)$. The search-$\mathsf{LWE}_{q,\alpha}$ problem is: given any desired number $m = \mathrm{poly}(n)$ of independent samples from $A_{\mathbf{s},\alpha}$ for some arbitrary $\mathbf{s}$, find $\mathbf{s}$. The decision-$\mathsf{LWE}_{q,\alpha}$ problem is to distinguish, with non-negligible advantage, between samples from $A_{\mathbf{s},\alpha}$ for uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$, and uniformly random samples from $\mathbb{Z}_q^n \times \mathbb{T}$. There are a variety of (incomparable) search/decision reductions for LWE under certain conditions on the parameters (e.g., [Reg05, Pei09b, ACPS09]); in Section 3 we give a reduction that essentially subsumes them all. When $q \geq 2\sqrt{n}/\alpha$, solving search-$\mathsf{LWE}_{q,\alpha}$ is at least as hard as *quantumly* approximating SIVP on n-dimensional lattices to within $\tilde{O}(n/\alpha)$ factors in the worst case [Reg05]. For a restricted range of parameters (e.g., when q is exponentially large) a *classical* (non-quantum) reduction is also known [Pei09b], but only from a potentially easier class of problems like the decisional Shortest Vector Problem (GapSVP) and the Bounded Distance Decoding Problem (BDD) (see [LM09]).

Note that the m samples $(\mathbf{a}_i, b_i)$ and underlying error terms e_i from $A_{\mathbf{s},\alpha}$ may be grouped into a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vectors $\mathbf{b} \in \mathbb{T}^m$, $\mathbf{e} \in \mathbb{R}^m$ in the natural way, so that $\mathbf{b} = (\mathbf{A}^t\mathbf{s})/q + \mathbf{e} \bmod 1$. In this way, $\mathbf{b}$ may be seen as an element of $\Lambda^{\perp}(\mathbf{A})^* = \frac{1}{q}\Lambda(\mathbf{A}^t)$, perturbed by Gaussian error. By scaling $\mathbf{b}$ and discretizing its entries using a form of randomized rounding (see [Pei10]), we can convert it into $\mathbf{b}' = \mathbf{A}^t\mathbf{s} + \mathbf{e}' \bmod q$ where $\mathbf{e}' \in \mathbb{Z}^m$ has *discrete* Gaussian distribution with parameter (say) $\sqrt{2}\alpha q$.

2.3 Gaussians and Lattices

The n-dimensional Gaussian function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as

$$\rho(\mathbf{x}) \stackrel{\Delta}{=} \exp(-\pi \cdot \|\mathbf{x}\|^2) = \exp(-\pi \cdot \langle \mathbf{x}, \mathbf{x} \rangle).$$

Applying a linear transformation given by a (not necessarily square) matrix $\mathbf{B}$ with linearly independent columns yields the (possibly degenerate) Gaussian function

$$\rho_{\mathbf{B}}(\mathbf{x}) \stackrel{\Delta}{=} \begin{cases} \rho(\mathbf{B}^+\mathbf{x}) = \exp\left(-\pi \cdot \mathbf{x}^t\Sigma^+\mathbf{x}\right) & \text{if } \mathbf{x} \in \mathrm{span}(\mathbf{B}) = \mathrm{span}(\Sigma) \\ 0 & \text{otherwise} \end{cases}$$

where $\Sigma = \mathbf{B}\mathbf{B}^t \geq \mathbf{0}$. Because $\rho_{\mathbf{B}}$ is distinguished only up to Σ, we usually refer to it as $\rho_{\sqrt{\Sigma}}$.

Normalizing $\rho_{\sqrt{\Sigma}}$ by its total measure over $\mathrm{span}(\Sigma)$, we obtain the probability distribution function of the (continuous) *Gaussian distribution* $D_{\sqrt{\Sigma}}$. By linearity of expectation, this distribution has *covariance* $\mathbb{E}_{\mathbf{x} \leftarrow D_{\sqrt{\Sigma}}}[\mathbf{x} \cdot \mathbf{x}^t] = \frac{\Sigma}{2\pi}$. (The $\frac{1}{2\pi}$ factor is the variance of the Gaussian D_1, due to our choice of normalization.) For convenience, we implicitly ignore the $\frac{1}{2\pi}$ factor, and refer to Σ as the covariance matrix of $D_{\sqrt{\Sigma}}$.

Let $\Lambda \subset \mathbb{R}^n$ be a lattice, let $\mathbf{c} \in \mathbb{R}^n$, and let $\Sigma \geq \mathbf{0}$ be a positive semidefinite matrix such that $(\Lambda + \mathbf{c}) \cap \mathrm{span}(\Sigma)$ is nonempty. The *discrete Gaussian distribution* $D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}$ is simply the Gaussian distribution $D_{\sqrt{\Sigma}}$ restricted to have support $\Lambda + \mathbf{c}$. That is, for all $\mathbf{x} \in \Lambda + \mathbf{c}$,

$$D_{\Lambda+\mathbf{c},\sqrt{\Sigma}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x})}{\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c})} \propto \rho_{\sqrt{\Sigma}}(\mathbf{x}).$$

We recall the definition of the *smoothing parameter* from [MR04], generalized to non-spherical (and potentially degenerate) Gaussians. It is easy to see that the definition is consistent with the partial ordering of positive semidefinite matrices, i.e., if $\Sigma_1 \geq \Sigma_2 \geq \eta_\epsilon(\Lambda)$, then $\Sigma_1 \geq \eta_\epsilon(\Lambda)$.

Definition 2.2. Let $\Sigma \geq \mathbf{0}$ and $\Lambda \subset \mathrm{span}(\Sigma)$ be a lattice. We say that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ if $\rho_{\sqrt{\Sigma^+}}(\Lambda^*) \leq 1 + \epsilon$.

The following is a bound on the smoothing parameter in terms of any orthogonalized basis. Note that for practical choices like $n \leq 2^{14}$ and $\epsilon \geq 2^{-80}$, the multiplicative factor attached to $\|\widetilde{\mathbf{B}}\|$ is bounded by 4.6.

Lemma 2.3 ([GPV08, Theorem 3.1])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\epsilon > 0$. We have*

$$\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \cdot \sqrt{\ln(2n(1 + 1/\epsilon))/\pi}.$$

In particular, for any $\omega(\sqrt{\log n})$ function, there is a negligible $\epsilon(n)$ for which $\eta_\epsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$.

For appropriate parameters, the smoothing parameter of a random lattice $\Lambda^\perp(\mathbf{A})$ is small, with very high probability. The following bound is a refinement and strengthening of one from [GPV08], which allows for a more precise analysis of the parameters and statistical errors involved in our constructions.

Lemma 2.4. *Let n, m, $q \geq 2$ be positive integers. For $\mathbf{s} \in \mathbb{Z}_q^n$, let the subgroup $\mathbb{G}_\mathbf{s} = \{\langle \mathbf{a}, \mathbf{s} \rangle : \mathbf{a} \in \mathbb{Z}_q^n\} \subseteq \mathbb{Z}_q$, and let $g_\mathbf{s} = |\mathbb{G}_\mathbf{s}| = q/\gcd(s_1, \ldots, s_n, q)$. Let $\epsilon > 0$, $\eta \geq \eta_\epsilon(\mathbb{Z}^m)$, and $s > \eta$ be reals. Then for uniformly random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$,*

$$\mathop{\mathbb{E}}_{\mathbf{A}}\left[\rho_{1/s}(\Lambda^\perp(\mathbf{A})^*)\right] \leq (1 + \epsilon) \sum_{\mathbf{s} \in \mathbb{Z}_q^n} \max\{1/g_\mathbf{s}, \eta/s\}^m. \tag{2.1}$$

In particular, if $q = p^e$ is a power of a prime p, and

$$m \geq \max\left\{n + \frac{\log(3 + 2/\epsilon)}{\log p}, \frac{n \log q + \log(2 + 2/\epsilon)}{\log(s/\eta)}\right\}, \tag{2.2}$$

then $\mathbb{E}_\mathbf{A}\left[\rho_{1/s}(\Lambda^\perp(\mathbf{A})^*)\right] \leq 1 + 2\epsilon$, and so by Markov's inequality, $s \geq \eta_{2\epsilon/\delta}(\Lambda^\perp(\mathbf{A}))$ except with probability at most δ.

Proof. We will use the fact (which follows from the Poisson summation formula; see [MR04, Lemma 2.8]) that $\rho_t(\Lambda) \leq \rho_r(\Lambda) \leq (r/t)^m \cdot \rho_t(\Lambda)$ for any rank-m lattice Λ and $r \geq t > 0$.

For any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, one can check that $\Lambda^\perp(\mathbf{A})^* = \mathbb{Z}^m + \{\mathbf{A}^t\mathbf{s}/q : \mathbf{s} \in \mathbb{Z}_q^n\}$. Note that $\mathbf{A}^t\mathbf{s}$ is uniformly

random over $\mathbb{G}_{\mathbf{s}}^m$, for uniformly random $\mathbf{A}$. Then we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{A}}\left[\rho_{1/s}(\Lambda^\perp(\mathbf{A})^*)\right] &\le \sum_{\mathbf{s}\in\mathbb{Z}_q^n} \mathop{\mathbb{E}}_{\mathbf{A}}\left[\rho_{1/s}(\mathbb{Z}^m + \mathbf{A}^t\mathbf{s}/q)\right] && \text{(lin. of } \mathbb{E}) \\
&= \sum_{\mathbf{s}\in\mathbb{Z}_q^n} g_{\mathbf{s}}^{-m} \cdot \rho_{1/s}(g_{\mathbf{s}}^{-1}\cdot\mathbb{Z}^m) && \text{(avg. over } \mathbf{A}) \\
&\le \sum_{\mathbf{s}\in\mathbb{Z}_q^n} g_{\mathbf{s}}^{-m} \cdot \max\{1, g_{\mathbf{s}}\eta/s\}^m \cdot \rho_{1/\eta}(\mathbb{Z}^m), && \text{(above fact)} \\
&\le (1+\epsilon)\sum_{\mathbf{s}\in\mathbb{Z}_q^n} \max\{1/g_{\mathbf{s}}, \eta/s\}^m, && (\eta \ge \eta_\epsilon(\mathbb{Z}^m)).
\end{aligned}$$

To prove the second part of the claim, observe that $g_{\mathbf{s}} = p^i$ for some $i \ge 0$, and that there are at most g^n values of $\mathbf{s}$ for which $g_{\mathbf{s}} = g$, because each entry of $\mathbf{s}$ must be in $\mathbb{G}_{\mathbf{s}}$. Therefore,

$$\sum_{\mathbf{s}\in\mathbb{Z}_q^n} 1/g_{\mathbf{s}}^m \le \sum_{i\ge 0} p^{i(n-m)} = \frac{1}{1-p^{n-m}} \le 1 + \frac{\epsilon}{2(1+\epsilon)}.$$

(More generally, for arbitrary q we have $\sum_{\mathbf{s}} 1/g_{\mathbf{s}}^m \le \zeta(m-n)$, where $\zeta(\cdot)$ is the Riemann zeta function.) Similarly, $\sum_{\mathbf{s}}(\eta/s)^m = q^n(s/\eta)^{-m} \le \frac{\epsilon}{2(1+\epsilon)}$, and the claim follows. □

We need a number of standard facts about discrete Gaussians.

Lemma 2.5 ([MR04, Lemmas 2.9 and 4.1])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice. For any $\Sigma \ge \mathbf{0}$ and $\mathbf{c} \in \mathbb{R}^n$, we have $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \le \rho_{\sqrt{\Sigma}}(\Lambda)$. Moreover, if $\sqrt{\Sigma} \ge \eta_\epsilon(\Lambda)$ for some $\epsilon > 0$ and $\mathbf{c} \in \mathrm{span}(\Lambda)$, then $\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \ge \frac{1-\epsilon}{1+\epsilon}\cdot\rho_{\sqrt{\Sigma}}(\Lambda)$.*

Combining the above lemma with a bound of Banaszczyk [Ban93], we have the following tail bound on discrete Gaussians.

Lemma 2.6 ([Ban93, Lemma 1.5])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $r \ge \eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$. For any $\mathbf{c} \in \mathrm{span}(\Lambda)$, we have*

$$\Pr\left[\|D_{\Lambda+\mathbf{c},r}\| \ge r\sqrt{n}\right] \le 2^{-n}\cdot\tfrac{1+\epsilon}{1-\epsilon}.$$

Moreover, if $\mathbf{c} = \mathbf{0}$ then the bound holds for any *$r > 0$, with $\epsilon = 0$.*

The next lemma bounds the predictability (i.e., probability of the most likely outcome or equivalently, min-entropy) of a discrete Gaussian.

Lemma 2.7 ([PR06, Lemma 2.11])**.** *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $r \ge 2\eta_\epsilon(\Lambda)$ for some $\epsilon \in (0,1)$. For any $\mathbf{c} \in \mathbb{R}^n$ and any $\mathbf{y} \in \Lambda + \mathbf{c}$, we have $\Pr[D_{\Lambda+\mathbf{c},r} = \mathbf{y}] \le 2^{-n}\cdot\frac{1+\epsilon}{1-\epsilon}$.*

2.4 Subgaussian Distributions and Random Matrices

For $\delta \ge 0$, we say that a random variable X (or its distribution) over $\mathbb{R}$ is *δ-subgaussian* with parameter $s > 0$ if for all $t \in \mathbb{R}$, the (scaled) moment-generating function satisfies

$$\mathbb{E}\left[\exp(2\pi t X)\right] \le \exp(\delta)\cdot\exp(\pi s^2 t^2).$$

Notice that the $\exp(\pi s^2 t^2)$ term on the right is precisely the (scaled) moment-generating function of the Gaussian distribution D_s. So, our definition differs from the usual definition of subgaussian only in the additional factor of $\exp(\delta)$; we need this relaxation when working with discrete Gaussians, usually taking $\delta = \ln(\frac{1+\epsilon}{1-\epsilon}) \approx 2\epsilon$ for the same small ϵ as in the smoothing parameter η_ϵ.

If X is δ-subgaussian, then its tails are dominated by a Gaussian of parameter s, i.e., $\Pr[|X| \geq t] \leq 2\exp(\delta)\exp(-\pi t^2/s^2)$ for all $t \geq 0$.[4] This follows by Markov's inequality: by scaling X we can assume $s = 1$, and we have

$$\Pr[X \geq t] = \Pr[\exp(2\pi t X) \geq \exp(2\pi t^2)] \leq \exp(\delta)\exp(\pi t^2)/\exp(2\pi t^2) = \exp(\delta)\exp(-\pi t^2).$$

The claim follows by repeating the argument with $-X$, and the union bound. Using the Taylor series expansion of $\exp(2\pi t X)$, it can be shown that any B-bounded symmetric random variable X (i.e., $|X| \leq B$ always) is 0-subgaussian with parameter $B\sqrt{2\pi}$.

More generally, we say that a random vector $\mathbf{x}$ or its distribution (respectively, a random matrix $\mathbf{X}$) is δ-subgaussian (of parameter s) if all its one-dimensional marginals $\langle \mathbf{u}, \mathbf{v} \rangle$ (respectively, $\mathbf{u}^t \mathbf{X} \mathbf{v}$) for unit vectors $\mathbf{u}, \mathbf{v}$ are δ-subgaussian (of parameter s). It follows immediately from the definition that the concatenation of independent δ_i-subgaussian vectors with common parameter s, interpreted as either a vector or matrix, is $(\sum \delta_i)$-subgaussian with parameter s.

Lemma 2.8. *Let $\Lambda \subset \mathbb{R}^n$ be a lattice and $s \geq \eta_\epsilon(\Lambda)$ for some $0 < \epsilon < 1$. For any $\mathbf{c} \in \mathrm{span}(\Lambda)$, $D_{\Lambda+\mathbf{c},s}$ is $\ln(\frac{1+\epsilon}{1-\epsilon})$-subgaussian with parameter s. Moreover, it is 0-subgaussian for* any *$s > 0$ when $\mathbf{c} = \mathbf{0}$.*

Proof. By scaling Λ we can assume that $s = 1$. Let $\mathbf{x}$ have distribution $D_{\Lambda+\mathbf{c}}$, and let $\mathbf{u} \in \mathbb{R}^n$ be any unit vector. We bound the scaled moment-generating function of the marginal $\langle \mathbf{x}, \mathbf{u} \rangle$ for any $t \in \mathbb{R}$:

$$\begin{aligned}
\rho(\Lambda + \mathbf{c}) \cdot \mathbb{E}\left[\exp(2\pi \langle \mathbf{x}, t\mathbf{u} \rangle)\right] &= \sum_{\mathbf{x} \in \Lambda + \mathbf{c}} \exp(-\pi(\langle \mathbf{x}, \mathbf{x} \rangle - 2\langle \mathbf{x}, t\mathbf{u} \rangle)) \\
&= \exp(\pi t^2) \cdot \sum_{\mathbf{x} \in \Lambda + \mathbf{c}} \exp(-\pi \langle \mathbf{x} - t\mathbf{u}, \mathbf{x} - t\mathbf{u} \rangle) \\
&= \exp(\pi t^2) \cdot \rho(\Lambda + \mathbf{c} - t\mathbf{u}).
\end{aligned}$$

Both claims then follow by Lemma 2.5. □

Here we recall a standard result from the non-asymptotic theory of random matrices; for further details, see [Ver11]. (The proof for δ-subgaussian distributions is a trivial adaptation of the 0-subgaussian case.)

Lemma 2.9. *Let $\mathbf{X} \in \mathbb{R}^{n \times m}$ be a δ-subgaussian random matrix with parameter s. There exists a universal constant $C > 0$ such that for any $t \geq 0$, we have $s_1(\mathbf{X}) \leq C \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $2\exp(\delta)\exp(-\pi t^2)$.*

Empirically, for discrete Gaussians the universal constant C in the above lemma is very close to $1/\sqrt{2\pi}$. In fact, it has been proved that $C \leq 1/\sqrt{2\pi}$ for matrices with independent identically distributed *continuous* Gaussian entries.

[4] The converse also holds (up to a small constant factor in the parameter s) when $\mathbb{E}[X] = 0$, but this will frequently not quite be the case in our applications, which is why we define subgaussian in terms of the moment-generating function.

3 Search to Decision Reduction

Here we give a new search-to-decision reduction for LWE that essentially subsumes all of the (incomparable) prior ones given in [BFKL93, Reg05, Pei09b, ACPS09].[5] Most notably, it handles moduli q that were not covered before, specifically, those like $q = 2^k$ that are divisible by powers of very small primes. The only known reduction that ours does not subsume is a different style of *sample-preserving* reduction recently given in [MM11], which works for a more limited class of moduli and error distributions; extending that reduction to the full range of parameters considered here is an interesting open problem. In what follows, $\omega(\sqrt{\log n})$ denotes some fixed function that grows faster than $\sqrt{\log n}$, asymptotically.

Theorem 3.1. *Let q have prime factorization $q = p_1^{e_1} \cdots p_k^{e_k}$ for pairwise distinct $\mathrm{poly}(n)$-bounded primes p_i with each $e_i \geq 1$, and let $0 < \alpha \leq 1/\omega(\sqrt{\log n})$. Let ℓ be the number of prime factors $p_i < \omega(\sqrt{\log n})/\alpha$. There is a probabilistic polynomial-time reduction from solving search-$\mathsf{LWE}_{q,\alpha}$ (in the worst case, with overwhelming probability) to solving decision-$\mathsf{LWE}_{q,\alpha'}$ (on the average, with non-negligible advantage) for any $\alpha' \geq \alpha$ such that $\alpha' \geq \omega(\sqrt{\log n})/p_i^{e_i}$ for every i, and $(\alpha')^\ell \geq \alpha \cdot \omega(\sqrt{\log n})^{1+\ell}$.*

For example, when every $p_i \geq \omega(\sqrt{\log n})/\alpha$ we have $\ell = 0$, and any $\alpha' \geq \alpha$ is acceptable. (This special case, with the additional constraint that every $e_i = 1$, is proved in [Pei09b].) As a qualitatively new example, when $q = p^e$ is a prime power for some (possibly small) prime p, then it suffices to let $\alpha' \geq \alpha \cdot \omega(\sqrt{\log n})^2$. (A similar special case where $q = p^e$ for sufficiently large p and $\alpha' = \alpha \ll 1/p$ is proved in [ACPS09].)

Proof. We show how to recover each entry of $\mathbf{s}$ modulo a large enough power of each p_i, given access to the distribution $A_{\mathbf{s},\alpha}$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and to an oracle $\mathcal{O}$ solving $\mathsf{DLWE}_{q,\alpha'}$. For the parameters in the theorem statement, we can then recover the remainder of $\mathbf{s}$ in polynomial time by rounding and standard Gaussian elimination.

First, observe that we can transform $A_{\mathbf{s},\alpha}$ into $A_{\mathbf{s},\alpha'}$ simply by adding (modulo 1) an independent sample from $D_{\sqrt{\alpha'^2 - \alpha^2}}$ to the second component of each $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle/q + D_\alpha \bmod 1) \in \mathbb{Z}_q^n \times \mathbb{T}$ drawn from $A_{\mathbf{s},\alpha}$.

We now show how to recover each entry of $\mathbf{s}$ modulo (powers of) any prime $p = p_i$ dividing q. Let $e = e_i$, and for $j = 0, 1, \ldots, e$ define $A^j_{\mathbf{s},\alpha'}$ to be the distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ obtained by drawing $(\mathbf{a}, b) \leftarrow A_{\mathbf{s},\alpha'}$ and outputting $(\mathbf{a}, b + r/p^j \bmod 1)$ for a fresh uniformly random $r \leftarrow \mathbb{Z}_q$. (Clearly, this distribution can be generated efficiently from $A_{\mathbf{s},\alpha'}$.) Note that when $\alpha' \geq \omega(\sqrt{\log n})/p^j \geq \eta_\epsilon((1/p^j)\mathbb{Z})$ for some $\epsilon = \mathrm{negl}(n)$, $A^j_{\mathbf{s},\alpha'}$ is negligibly far from $U = U(\mathbb{Z}_q^n \times \mathbb{T})$, and this holds at least for $j = e$ by hypothesis. Therefore, by a hybrid argument there exists some minimal $j \in [e]$ for which $\mathcal{O}$ has a non-negligible advantage in distinguishing between $A^{j-1}_{\mathbf{s},\alpha'}$ and $A^j_{\mathbf{s},\alpha'}$, over a random choice of $\mathbf{s}$ and all other randomness in the experiment. (This j can be found efficiently by measuring the behavior of $\mathcal{O}$.) Note that when $p_i \geq \omega(\sqrt{\log n})/\alpha \geq \omega(\sqrt{\log n})/\alpha'$, the minimal j must be 1; otherwise it may be larger, but there are at most ℓ of these by hypothesis. Now by a standard random self-reduction and amplification techniques (e.g., [Reg05, Lemma 4.1]), we can in fact assume that $\mathcal{O}$ accepts (respectively, rejects) with *overwhelming* probability given $A^{j-1}_{\mathbf{s},\alpha'}$ (resp., $A^j_{\mathbf{s},\alpha'}$), for *any* $\mathbf{s} \in \mathbb{Z}_q^n$.

Given access to $A^{j-1}_{\mathbf{s},\alpha'}$ and $\mathcal{O}$, we can test whether $s_1 = 0 \bmod p$ by invoking $\mathcal{O}$ on samples from $A^{j-1}_{\mathbf{s},\alpha'}$ that have been transformed as follows (all of what follows is analogous for $s_2, \ldots, s_n$): take each sample $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle/q + e + r/p^{j-1} \bmod 1) \leftarrow A^{j-1}_{\mathbf{s},\alpha'}$ to

$$(\mathbf{a}' = \mathbf{a} - r' \cdot (q/p^j) \cdot \mathbf{e}_1 \quad , \quad b' = b = \langle \mathbf{a}', \mathbf{s} \rangle/q + e + (pr + r's_1)/p^j \bmod 1) \tag{3.1}$$

[5] We say "essentially subsumes" because our reduction is not very meaningful when q is itself a very small prime, whereas those of [BFKL93, Reg05] are meaningful. This is only because our reduction deals with the *continuous* version of LWE. If we discretize the problem, then for very small prime q our reduction specializes to those of [BFKL93, Reg05].

for a fresh $r' \leftarrow \mathbb{Z}_q$ (where $\mathbf{e}_1 = (1, 0, \ldots, 0) \in \mathbb{Z}_q^n$). Observe that if $s_1 = 0 \bmod p$, the transformed samples are also drawn from $A^{j-1}_{\mathbf{s},\alpha'}$, otherwise they are drawn from $A^{j}_{\mathbf{s},\alpha'}$ because $r's_1$ is uniformly random modulo p. Therefore, $\mathcal{O}$ tells us which is the case.

Using the above test, we can efficiently recover $s_1 \bmod p$ by 'shifting' s_1 by each of $0, \ldots, p-1 \bmod p$ using the standard transformation that maps $A_{\mathbf{s},\alpha'}$ to $A_{\mathbf{s}+\mathbf{t},\alpha'}$ for any desired $\mathbf{t} \in \mathbb{Z}_q^n$, by taking $(\mathbf{a}, b)$ to $(\mathbf{a}, b + \langle \mathbf{a}, \mathbf{t} \rangle / q \bmod 1)$. (This enumeration step is where we use the fact that every p_i is $\mathrm{poly}(n)$-bounded.) Moreover, we can iteratively recover $s_1 \bmod p^2, \ldots, p^{e-j+1}$ as follows: having recovered $s_1 \bmod p^i$, first 'shift' $A_{\mathbf{s},\alpha'}$ to $A_{\mathbf{s}',\alpha'}$ where $s'_1 = 0 \bmod p^i$, then apply a similar procedure as above to recover $s'_1 \bmod p^{i+1}$: specifically, just modify the transformation in (3.1) to let $\mathbf{a}' = \mathbf{a} - r' \cdot (q/p^{j+i}) \cdot \mathbf{e}_1$, so that $b' = b = \langle \mathbf{a}', \mathbf{s} \rangle / q + e + (pr + r'(s'_1/p^i))/p^j$. This procedure works as long as p^{j+i} divides q, so we can recover $s_1 \bmod p^{e-j+1}$.

Using the above reductions and the Chinese remainder theorem, and letting j_i be the above minimal value of j for $p = p_i$ (of which at most ℓ of these are greater than 1), from $A_{\mathbf{s},\alpha}$ we can recover $\mathbf{s}$ modulo

$$P = \prod_i p_i^{e_i - (j_i - 1)} = q / \prod_i p_i^{j_i - 1} \geq q \cdot \left(\frac{\alpha'}{\omega(\sqrt{\log n})} \right)^{\ell} \geq q \cdot \alpha \cdot \omega(\sqrt{\log n}),$$

because $\alpha' < \omega(\sqrt{\log n})/p_i^{j_i - 1}$ for all i by definition of j_i and by hypothesis on α'. By applying the 'shift' transformation to $A_{\mathbf{s},\alpha}$ we can assume that $\mathbf{s} = 0 \bmod P$. Now every $\langle \mathbf{a}, \mathbf{s}' \rangle / q$ is an integer multiple of $P/q \geq \alpha \cdot \omega(\sqrt{\log n})$, and since every noise term $e \leftarrow D_\alpha$ has magnitude $< (\alpha/2) \cdot \omega(\sqrt{\log n})$ with overwhelming probability, we can round the second component of every $(\mathbf{a}, b) \leftarrow A_{\mathbf{s},\alpha}$ to the exact value of $\langle \mathbf{a}, \mathbf{s} \rangle / q \bmod 1$. From these we can solve for $\mathbf{s}$ by Gaussian elimination, and we are done. □

4 Primitive Lattices

At the heart of our new trapdoor generation algorithm (described in Section 5) is the construction of a very special family of lattices which have excellent geometric properties, and admit very fast and parallelizable decoding algorithms. The lattices are defined by means of what we call a *primitive matrix*. We say that a matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is primitive if its columns generate all of $\mathbb{Z}_q^n$, i.e., $\mathbf{G} \cdot \mathbb{Z}^m = \mathbb{Z}_q^n$.[6]

The main results of this section are summarized in the following theorem.

Theorem 4.1. *For any integers $q \geq 2$, $n \geq 1$, $k = \lceil \log_2 q \rceil$ and $m = nk$, there is a primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ such that*

- *The lattice $\Lambda^{\perp}(\mathbf{G})$ has a known basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ with $\|\widetilde{\mathbf{S}}\| \leq \sqrt{5}$ and $\|\mathbf{S}\| \leq \max\{\sqrt{5}, \sqrt{k}\}$. Moreover, when $q = 2^k$, we have $\widetilde{\mathbf{S}} = 2\mathbf{I}$ (so $\|\widetilde{\mathbf{S}}\| = 2$) and $\|\mathbf{S}\| = \sqrt{5}$.*
- *Both $\mathbf{G}$ and $\mathbf{S}$ require little storage. In particular, they are sparse (with only $O(m)$ nonzero entries) and highly structured.*
- *Inverting $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \bmod q$ can be performed in quasilinear $O(n \cdot \log^c n)$ time for any $\mathbf{s} \in \mathbb{Z}_q^n$ and any $\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$, where $\mathbf{B}$ can denote either $\mathbf{S}$ or $\widetilde{\mathbf{S}}$. Moreover, the algorithm is perfectly parallelizable, running in polylogarithmic $O(\log^c n)$ time using n processors. When $q = 2^k$, the polylogarithmic term $O(\log^c n)$ is essentially just the cost of k additions and shifts on k-bit integers.*

[6] We do not say that $\mathbf{G}$ is "full-rank," because $\mathbb{Z}_q$ is not a field when q is not prime, and the notion of rank for matrices over $\mathbb{Z}_q$ is not well defined.

- *Preimage sampling for $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\mathbf{x} \bmod q$ with Gaussian parameter $s \geq \|\widetilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$ can be performed in quasilinear $O(n \cdot \log^c n)$ time, or parallel polylogarithmic $O(\log^c n)$ time using n processors. When $q = 2^k$, the polylogarithmic term is essentially just the cost of k additions and shifts on k-bit integers, plus the (offline) generation of about m random integers drawn from $D_{\mathbb{Z},s}$.*

More generally, for any integer $b \geq 2$, all of the above statements hold with $k = \lceil \log_b q \rceil$, $\|\widetilde{\mathbf{S}}\| \leq \sqrt{b^2+1}$, and $\|\mathbf{S}\| \leq \max\{\sqrt{b^2+1}, (b-1)\sqrt{k}\}$; and when $q = b^k$, we have $\widetilde{\mathbf{S}} = b\mathbf{I}$ and $\|\mathbf{S}\| = \sqrt{b^2+1}$.

The rest of this section is dedicated to the proof of Theorem 4.1. In the process, we also make several important observations regarding the implementation of the inversion and sampling algorithms associated with $\mathbf{G}$, showing that our algorithms are not just asymptotically fast, but also quite practical.

Let $q \geq 2$ be an integer modulus and $k \geq 1$ be an integer dimension. Our construction starts with a *primitive vector* $\mathbf{g} \in \mathbb{Z}_q^k$, i.e., a vector such that $\gcd(g_1, \ldots, g_k, q) = 1$. The vector $\mathbf{g}$ defines a k-dimensional lattice $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$ having determinant $|\mathbb{Z}^k / \Lambda^\perp(\mathbf{g}^t)| = q$, because the residue classes of $\mathbb{Z}^k / \Lambda^\perp(\mathbf{g}^t)$ are in bijective correspondence with the possible values of $\langle \mathbf{g}, \mathbf{x} \rangle \bmod q$ for $\mathbf{x} \in \mathbb{Z}^k$, which cover all of $\mathbb{Z}_q$ since $\mathbf{g}$ is primitive. Concrete primitive vectors $\mathbf{g}$ will be described in the next subsections. Notice that when $q = \mathrm{poly}(n)$, we have $k = O(\log q) = O(\log n)$ and so $\Lambda^\perp(\mathbf{g}^t)$ is a very low-dimensional lattice. Let $\mathbf{S}_k \in \mathbb{Z}^{k \times k}$ be a basis of $\Lambda^\perp(\mathbf{g}^t)$, that is, $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \in \mathbb{Z}_q^{1 \times k}$ and $|\det(\mathbf{S}_k)| = q$.

The primitive vector $\mathbf{g}$ and associated basis $\mathbf{S}_k$ are used to define the parity-check matrix $\mathbf{G}$ and basis $\mathbf{S} \in \mathbb{Z}_q$ as $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times nk}$ and $\mathbf{S} := \mathbf{I}_n \otimes \mathbf{S}_k \in \mathbb{Z}^{nk \times nk}$. That is,

$$\mathbf{G} := \begin{bmatrix} \cdots \mathbf{g}^t \cdots & & & \\ & \cdots \mathbf{g}^t \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^t \cdots \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}, \qquad \mathbf{S} := \begin{bmatrix} \mathbf{S}_k & & & \\ & \mathbf{S}_k & & \\ & & \ddots & \\ & & & \mathbf{S}_k \end{bmatrix} \in \mathbb{Z}^{nk \times nk}.$$

Equivalently, $\mathbf{G}$, $\Lambda^\perp(\mathbf{G})$, and $\mathbf{S}$ are the direct sums of n copies of $\mathbf{g}^t$, $\Lambda^\perp(\mathbf{g}^t)$, and $\mathbf{S}_k$, respectively. It follows that $\mathbf{G}$ is a primitive matrix, the lattice $\Lambda^\perp(\mathbf{G}) \subset \mathbb{Z}^{nk}$ has determinant q^n, and $\mathbf{S}$ is a basis for this lattice. It also follows (and is clear by inspection) that $\|\mathbf{S}\| = \|\mathbf{S}_k\|$ and $\|\widetilde{\mathbf{S}}\| = \|\widetilde{\mathbf{S}_k}\|$.

By this direct sum construction, it is immediate that inverting $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e})$ and sampling preimages of $f_{\mathbf{G}}(\mathbf{x})$ can be accomplished by performing the same operations n times in parallel for $g_{\mathbf{g}^t}$ and $f_{\mathbf{g}^t}$ on the corresponding portions of the input, and concatenating the results. For preimage sampling, if each of the $f_{\mathbf{g}^t}$ preimages has Gaussian parameter $\sqrt{\Sigma}$, then by independence, their concatenation has parameter $\mathbf{I}_n \otimes \sqrt{\Sigma}$. Likewise, inverting $g_{\mathbf{G}}$ will succeed whenever all the n independent $g_{\mathbf{g}^t}$-inversion subproblems are solved correctly.

In the next two subsections we study concrete instantiations of the primitive vector $\mathbf{g}$, and give optimized algorithms for inverting $g_{\mathbf{g}^t}$ and sampling preimages for $f_{\mathbf{g}^t}$. In both subsections, we consider primitive lattices $\Lambda^\perp(\mathbf{g}^t) \subset \mathbb{Z}^k$ defined by the vector

$$\mathbf{g}^t := \begin{bmatrix} 1 & 2 & 4 & \cdots & 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times k}, \qquad k = \lceil \log_2 q \rceil, \tag{4.1}$$

whose entries form a geometrically increasing sequence. (We focus on powers of 2, but all our results trivially extend to other integer powers, or even mixed-integer products.) The only difference between the two subsections is in the form of the modulus q. We first study the case when the modulus $q = 2^k$ is a power of 2, which leads to especially simple and fast algorithms. Then we discuss how the results can be generalized to arbitrary moduli q. Notice that in both cases, the syndrome $\langle \mathbf{g}, \mathbf{x} \rangle \in \mathbb{Z}_q$ of a binary

vector $\mathbf{x} = (x_0, \ldots, x_{k-1}) \in \{0,1\}^k$ is just the positive integer with binary expansion $\mathbf{x}$. In general, for arbitrary $\mathbf{x} \in \mathbb{Z}^k$ the syndrome $\langle \mathbf{g}, \mathbf{x} \rangle \in \mathbb{Z}_q$ can be computed very efficiently by a sequence of k additions and binary shifts, and a single reduction modulo q, which is also trivial when $q = 2^k$ is a power of 2. The syndrome computation is also easily parallelizable, leading to $O(\log k) = O(\log \log n)$ computation time using $O(k) = O(\log n)$ processors.

4.1 Power-of-Two Modulus

Let $q = 2^k$ be a power of 2, and let $\mathbf{g}$ be the geometric vector defined in Equation (4.1). Define the matrix

$$\mathbf{S}_k := \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k \times k}.$$

This is a basis for $\Lambda^\perp(\mathbf{g}^t)$, because $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \bmod q$ and $\det(\mathbf{S}_k) = 2^k = q$. Clearly, all the basis vectors are short. Moreover, by orthogonalizing $\mathbf{S}_k$ in reverse order, we have $\widetilde{\mathbf{S}_k} = 2 \cdot \mathbf{I}_k$. This construction is summarized in the following proposition. (It generalizes in the obvious way to any integer base, not just 2.)

Proposition 4.2. *For $q = 2^k$ and $\mathbf{g} = (1, 2, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$, the lattice $\Lambda^\perp(\mathbf{g}^t)$ has a basis $\mathbf{S}$ such that $\widetilde{\mathbf{S}} = 2\mathbf{I}$ and $\|\mathbf{S}\| \le \sqrt{5}$. In particular, $\eta_\epsilon(\Lambda^\perp(\mathbf{g}^t)) \le 2r = 2 \cdot \omega(\sqrt{\log n})$ for some $\epsilon(n) = \mathrm{negl}(n)$.*

Using Proposition 4.2 and known generic algorithms [Bab85, Kle00, GPV08], it is possible to invert $g_{\mathbf{g}^t}(\mathbf{s}, \mathbf{e})$ correctly whenever $\mathbf{e} \in \mathcal{P}_{1/2}((q/2) \cdot \mathbf{I})$, and sample preimages under $f_{\mathbf{g}^t}$ with Gaussian parameter $s \ge 2r = 2 \cdot \omega(\sqrt{\log n})$. In what follows we show how the special structure of the basis $\mathbf{S}$ leads to simpler, faster, and more practical solutions to these general lattice problems.

Inversion. Here we show how to efficiently find an unknown scalar $s \in \mathbb{Z}_q$ given $\mathbf{b}^t = [b_0, b_1, \ldots, b_{k-1}] = s \cdot \mathbf{g}^t + \mathbf{e}^t = [s + e_0, 2s + e_1, \ldots, 2^{k-1}s + e_{k-1}] \bmod q$, where $\mathbf{e} \in \mathbb{Z}^k$ is a short error vector.

An iterative algorithm works by recovering the binary digits $s_0, s_1, \ldots, s_{k-1} \in \{0,1\}$ of $s \in \mathbb{Z}_q$, from least to most significant, as follows: first, determine s_0 by testing whether

$$b_{k-1} = 2^{k-1}s + e_{k-1} = (q/2)s_0 + e_{k-1} \bmod q$$

is closer to 0 or to $q/2$ (modulo q). Then recover s_1 from $b_{k-2} = 2^{k-2}s + e_{k-2} = 2^{k-1}s_1 + 2^{k-2}s_0 + e_{k-2} \bmod q$, by subtracting $2^{k-2}s_0$ and testing proximity to 0 or $q/2$, etc. It is easy to see that the algorithm produces correct output if every $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right)$, i.e., if $\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{I}_k/2) = \mathcal{P}_{1/2}(q \cdot (\widetilde{\mathbf{S}_k})^{-t})$. It can also be seen that this algorithm is exactly Babai's "nearest-plane" algorithm [Bab85], specialized to the scaled dual $q(\mathbf{S}_k)^{-t}$ of the basis $\mathbf{S}_k$ of $\Lambda^\perp(\mathbf{g}^t)$, which is a basis for $\Lambda(\mathbf{g})$.

Formally, the iterative algorithm is: given a vector $\mathbf{b}^t = [b_0, \ldots, b_{k-1}] \in \mathbb{Z}_q^{1 \times k}$, initialize $s \leftarrow 0$.

1. For $i = k-1, \ldots, 0$: let $s \leftarrow s + 2^{k-1-i} \cdot \left[b_i - 2^i \cdot s \notin \left[-\frac{q}{4}, \frac{q}{4}\right) \bmod q\right]$, where $[E] = 1$ if expression E is true, and 0 otherwise. Also let $e_i \leftarrow b_i - 2^i \cdot s \in \left[-\frac{q}{4}, \frac{q}{4}\right)$.

2. Output $s \in \mathbb{Z}_q$ and $\mathbf{e} = (e_0, \ldots, e_{k-1}) \in \left[-\frac{q}{4}, \frac{q}{4}\right)^k \subset \mathbb{Z}^k$.

Note that for $x \in \{0, \ldots, q-1\}$ with binary representation $(x_{k-1}x_{k-2}\cdots x_0)_2$, we have

$$\left[x \notin \left[-\tfrac{q}{4}, \tfrac{q}{4}\right) \bmod q\right] = x_{k-1} \oplus x_{k-2}.$$

There is also a non-iterative approach to decoding using a lookup table, and a hybrid approach between the two extremes. Notice that rounding each entry b_i of $\mathbf{b}$ to the nearest multiple of 2^i (modulo q, breaking ties upward) before running the above algorithm does not change the value of s that is computed. This lets us precompute a lookup table that maps the $2^{k(k+1)/2} = q^{O(\lg q)}$ possible rounded values of $\mathbf{b}$ to the correct values of s. The size of this table grows very rapidly for $k > 3$, but in this case we can do better if we assume slightly smaller error terms $e_i \in \left[-\frac{q}{8}, \frac{q}{8}\right)$: simply round each b_i to the nearest multiple of $\max\{\frac{q}{8}, 2^i\}$, thus producing one of exactly $8^{k-1} = q^3/8$ possible results, whose solutions can be stored in a lookup table. Note that the result is correct, because in each coordinate the total error introduced by e_i and rounding to a multiple of $\frac{q}{8}$ is in the range $\left[-\frac{q}{4}, \frac{q}{4}\right)$. A hybrid approach combining the iterative algorithm with table lookups of ℓ bits of s at a time is potentially the most efficient option in practice, and is easy to devise from the above discussion.

Gaussian sampling. We now consider the preimage sampling problem for function $f_{\mathbf{g}^t}$, i.e., the task of Gaussian sampling over a desired coset of $\Lambda^\perp(\mathbf{g}^t)$. More specifically, we want to sample a vector from the set $\Lambda_u^\perp(\mathbf{g}^t) = \{\mathbf{x} \in \mathbb{Z}^k : \langle \mathbf{g}, \mathbf{x}\rangle = u \bmod q\}$ for a desired syndrome $u \in \mathbb{Z}_q$, with probability proportional to $\rho_s(\mathbf{x})$. We wish to do so for any fixed Gaussian parameter $s \geq \|\widetilde{\mathbf{S}_k}\| \cdot r = 2 \cdot \omega(\sqrt{\log n})$, which is an optimal bound on the smoothing parameter of $\Lambda^\perp(\mathbf{G})$.

As with inversion, there are two main approaches to Gaussian sampling, which are actually opposite extremes on a spectrum of storage/parallelism trade-offs. The first approach is essentially to precompute and store many independent samples $\mathbf{x} \leftarrow D_{\mathbb{Z}^k, s}$, 'bucketing' them based on the value of $\langle \mathbf{g}, \mathbf{x}\rangle \in \mathbb{Z}_q$ until there is at least one sample per bucket. Because each $\langle \mathbf{g}, \mathbf{x}\rangle$ is statistically close to uniform over $\mathbb{Z}_q$ (by the smoothing parameter bound for $\Lambda^\perp(\mathbf{g}^t)$), a coupon-collecting argument implies that we need to generate about $q \log q$ samples to occupy every bucket. The online part of the sampling algorithm for $\Lambda^\perp(\mathbf{g}^t)$ is trivial, merely taking a fresh $\mathbf{x}$ from the appropriate bucket. The downside is that the storage and precomputation requirements are rather high: in many applications, q (while polynomial in the security parameter) can be in the many thousands or more.

The second approach exploits the niceness of the orthogonalized basis $\widetilde{\mathbf{S}_k} = 2\mathbf{I}_k$. Using this basis, the randomized nearest-plane algorithm of [Kle00, GPV08] becomes very simple and efficient, and is equivalent to the following: given a syndrome $u \in \{0, \ldots, q-1\}$ (viewed as an integer),

1. For $i = 0, \ldots, k-1$: choose $x_i \leftarrow D_{2\mathbb{Z}+u, s}$ and let $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$.
2. Output $\mathbf{x} = (x_0, \ldots, x_{k-1})$.

Observe that every Gaussian x_i in the above algorithm is chosen from one of only two possible cosets of $2\mathbb{Z}$, determined by the least significant bit of u at that moment. Therefore, we may precompute and store several independent Gaussian samples from each of $2\mathbb{Z}$ and $2\mathbb{Z}+1$, and consume one per iteration when executing the algorithm. (As above, the individual samples may be generated by choosing several $x \leftarrow D_{\mathbb{Z}, s}$ and bucketing each one according to its least-significant bit.) Such presampling makes the algorithm deterministic during its online phase, and because there are only two cosets, there is almost no wasted storage or precomputation. Notice, however, that this algorithm requires $k = \lg(q)$ sequential iterations.

Between the extremes of the two algorithms described above, there is a hybrid algorithm that chooses $\ell \geq 1$ entries of $\mathbf{x}$ at a time. (For simplicity, we assume that ℓ divides k exactly, though this is not

strictly necessary.) Let $\mathbf{h}^t = [1, 2, \ldots, 2^{\ell-1}] \in \mathbb{Z}_{2^\ell}^{1\times\ell}$ be a parity-check matrix defining the 2^ℓ-ary lattice $\Lambda^\perp(\mathbf{h}^t) \subseteq \mathbb{Z}^\ell$, and observe that $\mathbf{g}^t = [\mathbf{h}^t, 2^\ell \cdot \mathbf{h}^t, \ldots, 2^{k-\ell} \cdot \mathbf{h}^t]$. The hybrid algorithm then works as follows:

1. For $i = 0, \ldots, k/\ell - 1$, choose $(x_{i\ell}, \ldots, x_{(i+1)\ell-1}) \leftarrow D_{\Lambda^\perp_{u \bmod 2^\ell}(\mathbf{h}^t), s}$ and let $u \leftarrow (u - x)/2^\ell$, where $x = \sum_{j=0}^{\ell-1} x_{i\ell+j} \cdot 2^j \in \mathbb{Z}$.
2. Output $\mathbf{x} = (x_0, \ldots, x_{k-1})$.

As above, we can precompute samples $\mathbf{x} \leftarrow D_{\mathbb{Z}^\ell, s}$ and store them in a lookup table having 2^ℓ buckets, indexed by the value $\langle \mathbf{h}, \mathbf{x} \rangle \in \mathbb{Z}_{2^\ell}$, thereby making the algorithm deterministic in its online phase.

4.2 Arbitrary Modulus

For a modulus q that is not a power of 2, most of the above ideas still work, with slight adaptations. Let $k = \lceil \lg(q) \rceil$, so $q < 2^k$. As above, define $\mathbf{g}^t := [1, 2, \ldots, 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$, but now define the matrix

$$\mathbf{S}_k := \begin{bmatrix} 2 & & & & q_0 \\ -1 & 2 & & & q_1 \\ & -1 & & & q_2 \\ & & \ddots & & \vdots \\ & & & 2 & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix} \in \mathbb{Z}^{k\times k}$$

where $(q_0, \ldots, q_{k-1}) \in \{0,1\}^k$ is the binary expansion of $q = \sum_i 2^i \cdot q_i$. Again, $\mathbf{S}$ is a basis of $\Lambda^\perp(\mathbf{g}^t)$ because $\mathbf{g}^t \cdot \mathbf{S}_k = \mathbf{0} \bmod q$, and $\det(\mathbf{S}_k) = q$. Moreover, the basis vectors have squared length $\|\mathbf{s}_i\|^2 = 5$ for $i < k$ and $\|\mathbf{s}_k\|^2 = \sum_i q_i \le k$. The next lemma shows that $\mathbf{S}_k$ also has a good Gram-Schmidt orthogonalization.

Lemma 4.3. *With $\mathbf{S} = \mathbf{S}_k$ defined as above and orthogonalized in forward order, we have $\|\widetilde{\mathbf{s}_i}\|^2 = \frac{4-4^{-i}}{1-4^{-i}} \in (4, 5]$ for $1 \le i < k$, and $\|\widetilde{\mathbf{s}_k}\|^2 = \frac{3q^2}{4^k-1} < 3$.*

Proof. Notice that the the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_{k-1}$ are all orthogonal to $\mathbf{g}_k = (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}^k$. Thus, the orthogonal component of $\mathbf{s}_k$ has squared length

$$\|\widetilde{\mathbf{s}_k}\|^2 = \frac{\langle \mathbf{s}_k, \mathbf{g}_k \rangle^2}{\|\mathbf{g}_k\|^2} = \frac{q^2}{\sum_{j<k} 4^j} = \frac{3q^2}{4^k - 1}.$$

Similarly, the squared length of $\widetilde{\mathbf{s}_i}$ for $i < k$ can be computed as

$$\|\widetilde{\mathbf{s}_i}\|^2 = 1 + \frac{4^i}{\sum_{j<i} 4^j} = \frac{4 - 4^{-i}}{1 - 4^{-i}}.$$ □

This concludes the description and analysis of the primitive lattice $\Lambda^\perp(\mathbf{g}^t)$ when q is not a power of 2. Specialized inversion algorithms can also be adapted as well, but some care is needed. Of course, since the lattice dimension $k = O(\log n)$ is very small, one could simply use the general methods of [Bab85, Kle00, GPV08, Pei10] without worrying too much about optimizations, and satisfy all the claims made in Theorem 4.1. Below we briefly discuss alternatives for Gaussian sampling.

The offline 'bucketing' approach to Gaussian sampling works without any modification for arbitrary modulus, with just slighly larger Gaussian parameter $s \geq \sqrt{5} \cdot r$, because it relies only on the smoothing parameter bound of $\eta_\epsilon(\Lambda^\perp(\mathbf{g}^t)) \leq \|\widetilde{\mathbf{S}_k}\| \cdot \omega(\sqrt{\log n})$ and the fact that the number of buckets is q. The randomized nearest-plane approach to sampling does not admit a specialization as simple as the one we have described for $q = 2^k$. The reason is that while the basis $\mathbf{S}$ is sparse, its orthogonalization $\widetilde{\mathbf{S}}$ is not sparse in general. (This is in contrast to the case when $q = 2^k$, for which orthogonalizing in reverse order leads to the sparse matrix $\widetilde{\mathbf{S}} = 2\mathbf{I}$.) Still, $\widetilde{\mathbf{S}}$ is "almost triangular," in the sense that the off-diagonal entries decrease geometrically as one moves away from the diagonal. This may allow for optimizing the sampling algorithm by performig "truncated" scalar product computations, and still obtain an almost-Gaussian distribution on the resulting samples. An interesting alternative is to use a hybrid approach, where one first performs a single iteration of randomized nearest-plane algorithm to take care of the last basis vector $\mathbf{s}_k$, and then performs some variant of the convolution algorithm from [Pei10] to deal with the first $k-1$ basis vectors $[\mathbf{s}_1, \ldots, \mathbf{s}_{k-1}]$, which have very small lengths and singular values. Notice that the orthogonalized component of the last vector $\mathbf{s}_k$ is simply a scalar multiple of the primitive vector $\mathbf{g}$, so the scalar product $\langle \mathbf{s}_k, \mathbf{t} \rangle$ (for any vector $\mathbf{t}$ with syndrome $u = \langle \mathbf{g}, \mathbf{t} \rangle$) can be immediately computed from u as u/q (see Lemma 4.3).

4.3 The Ring Setting

The above constructions and algorithms all transfer easily to compact lattices defined over polynomial rings (i.e., number rings), as used in the representative works [Mic02, PR06, LM06, LPR10]. A commonly used example is the cyclomotic ring $R = \mathbb{Z}[x]/(\Phi_m(x))$ where $\Phi_m(x)$ denotes the mth cyclotomic polynomial, which is a monic, degree-$\varphi(m)$, irreducible polynomial whose zeros are all the *primitive* mth roots of unity in $\mathbb{C}$. The ring R is a $\mathbb{Z}$-module of rank n, i.e., it is generated as the additive integer combinations of the "power basis" elements $1, x, x^2, \ldots, x^{\varphi(m)-1}$. We let $R_q = R/qR$, the ring modulo the ideal generated by an integer q. For geometric concepts like error vectors and Gaussian distributions, it is usually nicest to work with the "canonical embedding" of R, which roughly (but not exactly) corresponds with the "coefficient embedding," which just considers the vector of coefficients relative to the power basis.

Let $\mathbf{g} \in R_q^k$ be a primitive vector modulo q, i.e., one for which the ideal generated by $q, g_1, \ldots, g_k$ is the full ring R. As above, the vector $\mathbf{g}$ defines functions $f_{\mathbf{g}^t} \colon R^k \to R_q$ and $g_{\mathbf{g}^t} \colon R_q \times R^k \to R_q^{1 \times k}$, defined as $f_{\mathbf{g}^t}(\mathbf{x}) = \langle \mathbf{g}, \mathbf{x} \rangle = \sum_{i=1}^k g_i \cdot x_i \bmod q$ and $g_{\mathbf{g}^t}(s, \mathbf{e}) = s \cdot \mathbf{g}^t + \mathbf{e}^t \bmod q$, and the related R-module

$$qR^k \quad \subseteq \quad \Lambda^\perp(\mathbf{g}^t) := \{\mathbf{x} \in R^k : f_{\mathbf{g}^t}(\mathbf{x}) = \langle \mathbf{g}, \mathbf{x} \rangle = 0 \bmod q\} \quad \subsetneq \quad R^k,$$

which has index (determinant) $q^n = |R_q|$ as an additive subgroup of R^k because $\mathbf{g}$ is primitive. Concretely, we can use the exact same primitive vector $\mathbf{g}^t = [1, 2, \ldots, 2^{k-1}] \in R_q^k$ as in Equation (4.1), interpreting its entries in the ring R_q rather than $\mathbb{Z}_q$.

Inversion and preimage sampling algorithms for $g_{\mathbf{g}^t}$ and $f_{\mathbf{g}^t}$ (respectively) are relatively straightforward to obtain, by adapting the basic approaches from the previous subsections. These algorithms are simplest when the power basis elements $1, x, x^2, \ldots, x^{\varphi(m)-1}$ are *orthogonal* under the canonical embedding (which is the case exactly when m is a power of 2, and hence $\Phi_m(x) = x^{m/2} + 1$), because the inversion operations reduce to parallel operations relative to each of the power basis elements. We defer the details to the full version.

5 Trapdoor Generation and Operations

In this section we describe our new trapdoor generation, inversion and sampling algorithms for hard random lattices. Recall that these are lattices $\Lambda^\perp(\mathbf{A})$ defined by an (almost) uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and that the standard notion of a "strong" trapdoor for these lattices (put forward in [GPV08] and used in a large number of subsequent applications) is a short lattice basis $\mathbf{S} \in \mathbb{Z}^{m\times m}$ for $\Lambda^\perp(\mathbf{A})$. There are several measures of quality for the trapdoor $\mathbf{S}$, the most common ones being (in nondecreasing order): the maximal Gram-Schmidt length $\|\widetilde{\mathbf{S}}\|$; the maximal Euclidean length $\|\mathbf{S}\|$; and the maximal singular value $s_1(\mathbf{S})$. Algorithms for generating random lattices together with high-quality trapdoor bases are given in [Ajt99, AP09]. In this section we give much simpler, faster and tighter algorithms to generate a hard random lattice with a trapdoor, and to use a trapdoor for performing standard tasks like inverting the LWE function $g_{\mathbf{A}}$ and sampling preimages for the SIS function $f_{\mathbf{A}}$. We also give a new, simple algorithm for delegating a trapdoor, i.e., using a trapdoor for $\mathbf{A}$ to obtain one for a matrix $[\mathbf{A} \mid \mathbf{A}']$ that extends $\mathbf{A}$, in a secure and non-reversible way.

The following theorem summarizes the main results of this section. Here we state just one typical instantiation with only asymptotic bounds. More general results and exact bounds are presented throughout the section.

Theorem 5.1. *There is an efficient randomized algorithm* $\mathsf{GenTrap}(1^n, 1^m, q)$ *that, given any integers* $n \geq 1$, $q \geq 2$, *and sufficiently large* $m = O(n \log q)$, *outputs a parity-check matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and a 'trapdoor'* $\mathbf{R}$ *such that the distribution of* $\mathbf{A}$ *is* $\mathrm{negl}(n)$*-far from uniform. Moreover, there are efficient algorithms* Invert *and* SampleD *that with overwhelming probability over all random choices, do the following:*

- *For* $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$, *where* $\mathbf{s} \in \mathbb{Z}_q^n$ *is arbitrary and either* $\|\mathbf{e}\| < q/O(\sqrt{n \log q})$ *or* $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,\alpha q}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega(\sqrt{\log n})$, *the deterministic algorithm* $\mathsf{Invert}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ *outputs* $\mathbf{s}$ *and* $\mathbf{e}$.
- *For any* $\mathbf{u} \in \mathbb{Z}_q^n$ *and large enough* $s = O(\sqrt{n \log q})$, *the randomized algorithm* $\mathsf{SampleD}(\mathbf{R}, \mathbf{A}, \mathbf{u}, s)$ *samples from a distribution within* $\mathrm{negl}(n)$ *statistical distance of* $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}),s\cdot\omega(\sqrt{\log n})}$.

Throughout this section, we let $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$ denote some fixed primitive matrix that admits efficient inversion and preimage sampling algorithms, as described in Theorem 4.1. (Recall that typically, $w = n\lceil \log q \rceil$ for some appropriate base of the logarithm.) All our algorithms and efficiency improvements are based on the primitive matrix $\mathbf{G}$ and associated algorithms described in Section 4, and a new notion of trapdoor that we define next.

5.1 A New Trapdoor Notion

We begin by defining the new notion of trapdoor, establish some of its most important properties, and give a simple and efficient algorithm for generating hard random lattices together with high-quality trapdoors.

Definition 5.2. Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$ be matrices with $m \geq w \geq n$. A $\mathbf{G}$-*trapdoor* for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}^{(m-w)\times w}$ such that $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{H}\mathbf{G}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$. We refer to $\mathbf{H}$ as the *tag* or *label* of the trapdoor. The *quality* of the trapdoor is measured by its largest singular value $s_1(\mathbf{R})$.

We remark that, by definition of $\mathbf{G}$-trapdoor, if $\mathbf{G}$ is a primitive matrix and $\mathbf{A}$ admits a $\mathbf{G}$ trapdoor, then $\mathbf{A}$ is primitive as well. In particular, $\det(\Lambda^\perp(\mathbf{A})) = q^n$. Since the primitive matrix $\mathbf{G}$ is typically fixed and public, we usually omit references to it, and refer to $\mathbf{G}$-trapdoors simply as trapdoors. We remark that since

$\mathbf{G}$ is primitive, the tag $\mathbf{H}$ in the above definition is uniquely determined by (and efficiently computable from) $\mathbf{A}$ and the trapdoor $\mathbf{R}$.

The following lemma says that a good basis for $\Lambda^\perp(\mathbf{A})$ may be obtained from knowledge of $\mathbf{R}$. We do not use the lemma anywhere in the rest of the paper, but include it here primarily to show that our new definition of trapdoor is at least as powerful as the traditional one of a short basis. Our algorithms for Gaussian sampling and LWE inversion do not need a full basis, and make direct (and more efficient) use of our new notion of trapdoor.

Lemma 5.3. *Let $\mathbf{S} \in \mathbb{Z}^{w \times w}$ be any basis for $\Lambda^\perp(\mathbf{G})$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ have trapdoor $\mathbf{R} \in \mathbb{Z}^{(m-w) \times w}$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. Then the lattice $\Lambda^\perp(\mathbf{A})$ is generated by the basis*

$$\mathbf{S_A} = \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{bmatrix},$$

where $\mathbf{W} \in \mathbb{Z}^{w \times \bar{m}}$ is an arbitrary solution to $\mathbf{GW} = -\mathbf{H}^{-1}\mathbf{A}[\mathbf{I} \mid \mathbf{0}]^T \bmod q$. Moreover, the basis $\mathbf{S_A}$ satisfies $\|\widetilde{\mathbf{S_A}}\| \leq s_1(\left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]) \cdot \|\widetilde{\mathbf{S}}\| \leq (s_1(\mathbf{R}) + 1) \cdot \|\widetilde{\mathbf{S}}\|$, when $\mathbf{S_A}$ is orthogonalized in suitable order.

Proof. It is immediate to check that $\mathbf{A} \cdot \mathbf{S_A} = \mathbf{0} \bmod q$, so $\mathbf{S_A}$ generates a sublattice of $\Lambda^\perp(\mathbf{A})$. In fact, it generates the entire lattice because $\det(\mathbf{S_A}) = \det(\mathbf{S}) = q^n = \det(\Lambda^\perp(\mathbf{A}))$.

The bound on $\|\widetilde{\mathbf{S_A}}\|$ follows by simple linear algebra. Recall by Item 3 of Lemma 2.1 that $\|\widetilde{\mathbf{B}}\| = \|\widetilde{\mathbf{S}}\|$ when the columns of $\mathbf{B} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W} & \mathbf{S} \end{smallmatrix}\right]$ are reordered appropriately. So it suffices to show that $\|\widetilde{\mathbf{TB}}\| \leq s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$ for any $\mathbf{T}$, $\mathbf{B}$. Let $\mathbf{B} = \mathbf{QDU}$ and $\mathbf{TB} = \mathbf{Q'D'U'}$ be Gram-Schmidt decompositions of $\mathbf{B}$ and $\mathbf{TB}$, respectively, with $\mathbf{Q}, \mathbf{Q'}$ orthogonal, $\mathbf{D}, \mathbf{D'}$ diagonal with nonnegative entries, and $\mathbf{U}, \mathbf{U'}$ upper unitriangular. We have

$$\mathbf{TQDU} = \mathbf{Q'D'U'} \Longrightarrow \mathbf{T'D} = \mathbf{D'U''},$$

where $\mathbf{T} = \mathbf{Q'T'Q}^{-1} \Rightarrow s_1(\mathbf{T'}) = s_1(\mathbf{T})$, and $\mathbf{U''}$ is upper unitriangular because such matrices form a multiplicative group. Now every row of $\mathbf{T'D}$ has Euclidean norm at most $s_1(\mathbf{T}) \cdot \|\mathbf{D}\| = s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$, while the ith row of $\mathbf{D'U''}$ has norm at least $d'_{i,i}$, the ith diagonal of $\mathbf{D'}$. We conclude that $\|\widetilde{\mathbf{TB}}\| = \|\mathbf{D}\| \leq s_1(\mathbf{T}) \cdot \|\widetilde{\mathbf{B}}\|$, as desired. □

We also make the following simple but useful observations:

- The rows of $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$ in Definition 5.2 can appear in any order, since this just induces a permutation of $\mathbf{A}$'s columns.
- If $\mathbf{R}$ is a trapdoor for $\mathbf{A}$, then it can be made into an equally good trapdoor for any extension $[\mathbf{A} \mid \mathbf{B}]$, by padding $\mathbf{R}$ with zero rows; this leaves $s_1(\mathbf{R})$ unchanged.
- If $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H}$, then $\mathbf{R}$ is also a trapdoor for $\mathbf{A'} = \mathbf{A} - [\mathbf{0} \mid \mathbf{H'G}]$ with tag $(\mathbf{H} - \mathbf{H'})$ for any $\mathbf{H'} \in \mathbb{Z}_q^{n \times n}$, as long as $(\mathbf{H} - \mathbf{H'})$ is invertible modulo q. This is the main idea behind the compact IBE of [ABB10a], and can be used to give a family of "tag-based" trapdoor functions [KMO10]. In Section 6 we give explicit families of matrices $\mathbf{H}$ having suitable properties for applications.

5.2 Trapdoor Generation

We now give an algorithm to generate a (pseudo)random matrix $\mathbf{A}$ together with a $\mathbf{G}$-trapdoor. The algorithm is straightforward, and in fact it can be easily derived from the definition of $\mathbf{G}$-trapdoor itself. A random

lattice is built by first extending the primitive matrix $\mathbf{G}$ into a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{HG}]$ (where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ is chosen at random, and $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ is the desired tag), and then applying a random transformation $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right] \in \mathbb{Z}^{m \times m}$ to the semi-random lattice $\Lambda^\perp(\mathbf{A}')$. Since $\mathbf{T}$ is unimodular with inverse $\mathbf{T}^{-1} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$, by Lemma 2.1 this yields the lattice $\mathbf{T} \cdot \Lambda^\perp(\mathbf{A}') = \Lambda^\perp(\mathbf{A}' \cdot \mathbf{T}^{-1})$ associated with the parity-check matrix $\mathbf{A} = \mathbf{A}' \cdot \mathbf{T}^{-1} = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}]$. Moreover, the distribution of $\mathbf{A}$ is close to uniform (either statistically, or computationally) as long as the distribution of $[\bar{\mathbf{A}} \mid \mathbf{0}]\mathbf{T}^{-1} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$ is. For details, see Algorithm 1, whose correctness is immediate.

Algorithm 1 Efficient algorithm $\mathsf{GenTrap}^{\mathcal{D}}(\bar{\mathbf{A}}, \mathbf{H})$ for generating a parity-check matrix $\mathbf{A}$ with trapdoor $\mathbf{R}$.

Input: Matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ for some $\bar{m} \geq 1$, invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$, and distribution $\mathcal{D}$ over $\mathbb{Z}^{\bar{m} \times w}$.
(If no particular $\bar{\mathbf{A}}$, $\mathbf{H}$ are given as input, then the algorithm may choose them itself, e.g., picking $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random, and setting $\mathbf{H} = \mathbf{I}$.)

Output: A parity-check matrix $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n \times m}$, where $m = \bar{m} + w$, and trapdoor $\mathbf{R}$ with tag $\mathbf{H}$.

1: Choose a matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$ from distribution $\mathcal{D}$.
2: Output $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$ and trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$.

We next describe two types of $\mathsf{GenTrap}$ instantiations. The first type generates a trapdoor $\mathbf{R}$ for a statistically near-uniform output matrix $\mathbf{A}$ using dimension $\bar{m} \approx n \log q$ or less (there is a trade-off between $\bar{m}$ and the trapdoor quality $s_1(\mathbf{R})$). The second types generates a computationally *pseudorandom* $\mathbf{A}$ (under the LWE assumption) using dimension $\bar{m} = 2n$; this pseudorandom construction is the first of its kind in the literature. Certain applications allow for an optimization that decreases $\bar{m}$ by an additive n term; this is most significant in the computationally secure construction because it yields $\bar{m} = n$.

Statistical instantiation. This instantiation works for any parameter $\bar{m}$ and distribution $\mathcal{D}$ over $\mathbb{Z}^{\bar{m} \times w}$ having the following two properties:

1. *Subgaussianity*: $\mathcal{D}$ is subgaussian with some parameter $s > 0$ (or δ-subgaussian for some small δ). This implies by Lemma 2.9 that $\mathbf{R} \leftarrow \mathcal{D}$ has $s_1(\mathbf{R}) = s \cdot O(\sqrt{\bar{m}} + \sqrt{w})$, except with probability $2^{-\Omega(\bar{m}+w)}$. (Recall that the constant factor hidden in the $O(\cdot)$ expression is $\approx 1/\sqrt{2\pi}$.)
2. *Regularity*: for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, $\mathbf{A} = [\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is δ-uniform for some $\delta = \mathrm{negl}(n)$.

   In fact, there is no loss in security if $\bar{\mathbf{A}}$ contains an identity matrix $\mathbf{I}$ as a submatrix and is otherwise uniform, since this corresponds with the Hermite normal form of the SIS and LWE problems. See, e.g., [MR09, Section 5] for further details.

For example, let $\mathcal{D} = \mathcal{P}^{\bar{m} \times w}$ where $\mathcal{P}$ is the distribution over $\mathbb{Z}$ that outputs 0 with probability $1/2$, and ± 1 each with probability $1/4$. Then $\mathcal{P}$ (and hence $\mathcal{D}$) is 0-subgaussian with parameter $\sqrt{2\pi}$, and satisfies the regularity condition (for any q) for $\delta \leq \frac{w}{2}\sqrt{q^n/2^{\bar{m}}}$, by a version of the leftover hash lemma (see, e.g., [AP09, Section 2.2.1]). Therefore, we can use any $\bar{m} \geq n \lg q + 2 \lg \frac{w}{2\delta}$.

As another important example, let $\mathcal{D} = D_{\mathbb{Z},s}^{\bar{m} \times w}$ be a discrete Gaussian distribution for some $s \geq \eta_\epsilon(\mathbb{Z})$ and $\epsilon = \mathrm{negl}(n)$. Then $\mathcal{D}$ is 0-subgaussian with parameter s by Lemma 2.8, and satisfies the regularity condition when $\bar{m}$ satisfies the bound (2.2) from Lemma 2.4. For example, letting $s = 2\eta_\epsilon(\mathbb{Z})$ we can use any $\bar{m} = n \lg q + \omega(\log n)$. (Other tradeoffs between s and $\bar{m}$ are possible, potentially using a different choice of $\mathbf{G}$, and more exact bounds on the error probabilities can be worked out from the lemma statements.) Moreover, by Lemmas 2.4 and 2.8 we have that with overwhelming probability over the choice of $\bar{\mathbf{A}}$, the

conditional distribution of $\mathbf{R}$ given $\mathbf{A} = [\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R}]$ is $\mathrm{negl}(n)$-subgaussian with parameter s. We will use this fact in some of our applications in Section 6.

Computational instantiation. Let $\bar{\mathbf{A}} = [\mathbf{I} \mid \hat{\mathbf{A}}] \in \mathbb{Z}_q^{n \times \bar{m}}$ for $\bar{m} = 2n$, and let $\mathcal{D} = D_{\mathbb{Z},s}^{\bar{m} \times w}$ for some $s = \alpha q$, where $\alpha > 0$ is an LWE relative error rate (and typically $\alpha q > \sqrt{n}$). Clearly, $\mathcal{D}$ is 0-subgaussian with parameter αq. Also, $[\bar{\mathbf{A}} \mid \bar{\mathbf{A}}\mathbf{R} = \hat{\mathbf{A}}\mathbf{R}_2 + \mathbf{R}_1]$ for $\mathbf{R} = \left[\begin{smallmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{smallmatrix}\right] \leftarrow \mathcal{D}$ is exactly an instance of decision-$\mathsf{LWE}_{n,q,\alpha}$ (in its normal form), and hence is pseudorandom (ignoring the identity submatrix) assuming that the problem is hard.

Further optimizations. If an application only uses a single tag $\mathbf{H} = \mathbf{I}$ (as is the case with, for example, GPV signatures [GPV08]), then we can save an additive n term in the dimension $\bar{m}$ (and hence in the total dimension m): instead of putting an identity submatrix in $\bar{\mathbf{A}}$, we can instead use the identity submatrix from $\mathbf{G}$ (which exists without loss of generality, since $\mathbf{G}$ is primitive) and conceal the remainder of $\mathbf{G}$ using either of the above methods.

All of the above ideas also translate immediately to the ring setting (see Section 4.3), using an appropriate regularity lemma (e.g., the one in [LPR10]) for a statistical instantiation, and the ring-LWE problem for a computationally secure instantiation.

5.3 LWE Inversion

Algorithm 2 below shows how to use a trapdoor to solve LWE relative to $\mathbf{A}$. Given a trapdoor $\mathbf{R}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and an LWE instance $\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$ for some short error vector $\mathbf{e} \in \mathbb{Z}^m$, the algorithm recovers $\mathbf{s}$ (and $\mathbf{e}$). This naturally yields an inversion algorithm for the injective trapdoor function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \bmod q$, which is hard to invert (and whose output is pseudorandom) if LWE is hard.

Algorithm 2 Efficient algorithm $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}, \mathbf{b})$ for inverting the function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$.

Input: An oracle $\mathcal{O}$ for inverting the function $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ when $\hat{\mathbf{e}} \in \mathbb{Z}^w$ is suitably small.

- parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$;
- $\mathbf{G}$-trapdoor $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times kn}$ for $\mathbf{A}$ with invertible tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$;
- vector $\mathbf{b}^t = g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$ for any $\mathbf{s} \in \mathbb{Z}_q^n$ and suitably small $\mathbf{e} \in \mathbb{Z}^m$.

Output: The vectors $\mathbf{s}$ and $\mathbf{e}$.

1: Compute $\hat{\mathbf{b}}^t = \mathbf{b}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right]$.
2: Get $(\hat{\mathbf{s}}, \hat{\mathbf{e}}) \leftarrow \mathcal{O}(\hat{\mathbf{b}})$.
3: **return** $\mathbf{s} = \mathbf{H}^{-t}\hat{\mathbf{s}}$ and $\mathbf{e} = \mathbf{b} - \mathbf{A}^t\mathbf{s}$ (interpreted as a vector in $\mathbb{Z}^m$ with entries in $[-\frac{q}{2}, \frac{q}{2})$).

Theorem 5.4. *Suppose that oracle $\mathcal{O}$ in Algorithm 2 correctly inverts $g_{\mathbf{G}}(\hat{\mathbf{s}}, \hat{\mathbf{e}})$ for any error vector $\hat{\mathbf{e}} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ for some $\mathbf{B}$. Then for any $\mathbf{s}$ and $\mathbf{e}$ of length $\|\mathbf{e}\| < q/(2\|\mathbf{B}\|s)$ where $s = \sqrt{s_1(\mathbf{R})^2 + 1}$, Algorithm 2 correctly inverts $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e})$. Moreover, for any $\mathbf{s}$ and random $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ where $1/\alpha \geq 2\|\mathbf{B}\|s \cdot \omega(\sqrt{\log n})$, the algorithm inverts successfully with overwhelming probability over the choice of $\mathbf{e}$.*

Note that using our constructions from Section 4, we can implement $\mathcal{O}$ so that either $\|\mathbf{B}\| = 2$ (for q a power of 2, where $\mathbf{B} = \widetilde{\mathbf{S}} = 2\mathbf{I}$) or $\|\mathbf{B}\| = \sqrt{5}$ (for arbitrary q).

Proof. Let $\bar{\mathbf{R}} = [\,\mathbf{R}^t\ \mathbf{I}\,]$, and note that $s = s_1(\bar{\mathbf{R}})$. By the above description, the algorithm works correctly when $\bar{\mathbf{R}}\mathbf{e} \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$; equivalently, when $(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q \in [-\frac{1}{2}, \frac{1}{2})$ for all i. By definition of s, we have $\|\mathbf{b}_i^t\bar{\mathbf{R}}\| \le s\|\mathbf{B}\|$. If $\|\mathbf{e}\| < q/(2\|\mathbf{B}\|s)$, then $|(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q| < 1/2$ by Cauchy-Schwarz. Moreover, if $\mathbf{e}$ is chosen at random from $D_{\mathbb{Z}^m,\alpha q}$, then by the fact that $\mathbf{e}$ is 0-subgaussian (Lemma 2.8) with parameter αq, the probability that $|(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q| \ge 1/2$ is negligible, and the second claim follows by the union bound. □

5.4 Gaussian Sampling

Here we show how to use a trapdoor for efficient Gaussian preimage sampling for the function $f_{\mathbf{A}}$, i.e., sampling from a discrete Gaussian over a desired coset of $\Lambda^\perp(\mathbf{A})$. Our precise goal is, given a $\mathbf{G}$-trapdoor $\mathbf{R}$ (with tag $\mathbf{H}$) for matrix $\mathbf{A}$ and a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$, to sample from the spherical discrete Gaussian $D_{\Lambda_{\mathbf{u}}^\perp(\mathbf{A}),s}$ for relatively small parameter s. As we show next, this task can be reduced, via some efficient pre- and post-processing, to sampling from any sufficiently narrow (not necessarily spherical) Gaussian over the primitive lattice $\Lambda^\perp(\mathbf{G})$.

The main ideas behind our algorithm, which is described formally in Algorithm 3, are as follows. For simplicity, suppose that $\mathbf{R}$ has tag $\mathbf{H} = \mathbf{I}$, so $\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$, and suppose we have a subroutine for Gaussian sampling from any desired coset of $\Lambda^\perp(\mathbf{G})$ with some small, fixed parameter $\sqrt{\Sigma_{\mathbf{G}}} \ge \eta_\epsilon(\Lambda^\perp(\mathbf{G}))$. For example, Section 4 describes algorithms for which $\sqrt{\Sigma_{\mathbf{G}}}$ is either 2 or $\sqrt{5}$. (Throughout this summary we omit the small rounding factor $r = \omega(\sqrt{\log n})$ from all Gaussian parameters.) The algorithm for sampling from a coset $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ follows from two main observations:

1. If we sample a Gaussian $\mathbf{z}$ with parameter $\sqrt{\Sigma_{\mathbf{G}}}$ from $\Lambda_{\mathbf{u}}^\perp(\mathbf{G})$ and produce $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$, then $\mathbf{y}$ is Gaussian over the (non-full-rank) set $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Lambda_{\mathbf{u}}^\perp(\mathbf{G}) \subsetneq \Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ with parameter $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\sqrt{\Sigma_{\mathbf{G}}}$ (i.e., covariance $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Sigma_{\mathbf{G}}[\,\mathbf{R}^t\ \mathbf{I}\,]$). The (strict) inclusion holds because for any $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ where $\mathbf{z} \in \Lambda_{\mathbf{u}}^\perp(\mathbf{G})$, we have

$$\mathbf{A}\mathbf{y} = (\mathbf{A}\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])\mathbf{z} = \mathbf{G}\mathbf{z} = \mathbf{u}.$$

   Note that $s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_{\mathbf{G}}}) \le s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]) \cdot s_1(\sqrt{\Sigma_{\mathbf{G}}}) \le \sqrt{s_1(\mathbf{R})^2 + 1} \cdot s_1(\sqrt{\Sigma_{\mathbf{G}}})$, so $\mathbf{y}$'s distribution is only about an $s_1(\mathbf{R})$ factor wider than that of $\mathbf{z}$ over $\Lambda_{\mathbf{u}}^\perp(\mathbf{G})$. However, $\mathbf{y}$ lies in a non-full-rank subset of $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$, and its distribution is 'skewed' (non-spherical). This leaks information about the trapdoor $\mathbf{R}$, so we cannot just output $\mathbf{y}$.

2. To sample from a *spherical* Gaussian over all of $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$, we use the 'convolution' technique from [Pei10] to correct for the above-described problems with the distribution of $\mathbf{y}$. Specifically, we first choose a Gaussian perturbation $\mathbf{p} \in \mathbb{Z}^m$ having covariance $s^2 - \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Sigma_{\mathbf{G}}\left[\,\mathbf{R}^t\ \mathbf{I}\,\right]$, which is well-defined as long as $s \ge s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_{\mathbf{G}}})$. We then sample $\mathbf{y} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$ as above for an adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A}\mathbf{p}$, and output $\mathbf{x} = \mathbf{p} + \mathbf{y}$. Now the support of $\mathbf{x}$ is all of $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$, and because the covariances of $\mathbf{p}$ and $\mathbf{y}$ are additive (subject to some mild hypotheses), the overall distribution of $\mathbf{x}$ is spherical with Gaussian parameter s that can be as small as $s \approx s_1(\mathbf{R}) \cdot s_1(\sqrt{\Sigma_{\mathbf{G}}})$.

Quality analysis. Algorithm 3 can sample from a discrete Gaussian with parameter $s \cdot \omega(\sqrt{\log n})$ where s can be as small as $\sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{s_1(\Sigma_{\mathbf{G}}) + 2}$. We stress that this is only very slightly larger — a factor of at most $\sqrt{6/4} \le 1.23$ — than the bound $(s_1(\mathbf{R}) + 1) \cdot \|\widetilde{\mathbf{S}}\|$ from Lemma 5.3 on the largest Gram-Schmidt norm of a lattice basis derived from the trapdoor $\mathbf{R}$. (Recall that our constructions from Section 4 give $s_1(\Sigma_{\mathbf{G}}) = \|\widetilde{\mathbf{S}}\|^2 = 4$ or 5.) In the iterative "randomized nearest-plane" sampling algorithm of [Kle00, GPV08], the Gaussian parameter s is lower-bounded by the largest Gram-Schmidt norm of the orthogonalized input basis (times the same $\omega(\sqrt{\log n})$ factor used in our algorithm). Therefore, the efficiency

and parallelism of Algorithm 3 comes at almost no cost in quality versus slower, iterative algorithms that use high-precision arithmetic. (It seems very likely that the corresponding small loss in security can easily be mitigated with slightly larger parameters, while still yielding a significant net gain in performance.)

Runtime analysis. We now analyze the computational cost of Algorithm 3, with a focus on optimizing the online runtime and parallelism (sometimes at the expense of the offline phase, which we do not attempt to optimize).

The offline phase is dominated by sampling from $D_{\mathbb{Z}^m, r\sqrt{\Sigma}}$ for some fixed (typically non-spherical) covariance matrix $\Sigma > \mathbf{I}$. By [Pei10, Theorem 3.1], this can be accomplished (up to any desired statistical distance) simply by sampling a continuous Gaussian $D_{r\sqrt{\Sigma - \mathbf{I}}}$ with sufficient precision, then independently randomized-rounding each entry of the sampled vector to $\mathbb{Z}$ using Gaussian parameter $r \geq \eta_\epsilon(\mathbb{Z})$.

Naively, the online work is dominated by the computation of $\mathbf{H}^{-1}(\mathbf{u} - \bar{\mathbf{w}})$ and $\mathbf{Rz}$ (plus the call to $\mathcal{O}(\mathbf{v})$, which as described in Section 4 requires only $O(\log^c n)$ work, or one table lookup, by each of n processors in parallel). In general, the first computation takes $O(n^2)$ scalar multiplications and additions in $\mathbb{Z}_q$, while the latter takes $O(\bar{m} \cdot w)$, which is typically $\Theta(n^2 \log^2 q)$. (Obviously, both computations are perfectly parallelizable.) However, the special form of $\mathbf{z}$, and often of $\mathbf{H}$, allow for some further asymptotic and practical optimizations: since $\mathbf{z}$ is typically produced by concatenating n independent dimension-k subvectors that are sampled offline, we can precompute much of $\mathbf{Rz}$ by pre-multiplying each subvector by each of the n blocks of k columns in $\mathbf{R}$. This reduces the online computation of $\mathbf{Rz}$ to the summation of n dimension-$\bar{m}$ vectors, or $O(n^2 \log q)$ scalar additions (and no multiplications) in $\mathbb{Z}_q$. As for multiplication by $\mathbf{H}^{-1}$, in some applications (like GPV signatures) $\mathbf{H}$ is always the identity $\mathbf{I}$, in which case multiplication is unnecessary; in all other applications we know of, $\mathbf{H}$ actually represents multiplication in a certain extension field/ring of $\mathbb{Z}_q$, which can be computed in $O(n \log n)$ scalar operations and depth $O(\log n)$. In conclusion, the asymptotic cost of the online phase is still dominated by computing $\mathbf{Rz}$, which takes $\tilde{O}(n^2)$ work, but the hidden constants are small and many practical speedups are possible.

Theorem 5.5. *Algorithm 3 is correct.*

To prove the theorem we need the following fact about products of Gaussian functions.

Fact 5.6 (Product of degenerate Gaussians)**.** *Let $\Sigma_1, \Sigma_2 \in \mathbb{R}^{m \times m}$ be symmetric positive semidefinite matrices, let $V_i = \mathrm{span}(\Sigma_i)$ for $i = 1, 2$ and $V_3 = V_1 \cap V_2$, let $\mathbf{P} = \mathbf{P}^t \in \mathbb{R}^{m \times m}$ be the symmetric matrix that projects orthogonally onto V_3, and let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^m$ be arbitrary. Supposing it exists, let $\mathbf{v}$ be the unique point in $(V_1 + \mathbf{c}_1) \cap (V_2 + \mathbf{c}_2) \cap V_3^\perp$. Then*

$$\rho_{\sqrt{\Sigma_1}}(\mathbf{x} - \mathbf{c}_1) \cdot \rho_{\sqrt{\Sigma_2}}(\mathbf{x} - \mathbf{c}_2) = \rho_{\sqrt{\Sigma_1 + \Sigma_2}}(\mathbf{c}_1 - \mathbf{c}_2) \cdot \rho_{\sqrt{\Sigma_3}}(\mathbf{x} - \mathbf{c}_3),$$

where Σ_3 and $\mathbf{c}_3 \in \mathbf{v} + V_3$ are such that

$$\begin{aligned} \Sigma_3^+ &= \mathbf{P}(\Sigma_1^+ + \Sigma_2^+)\mathbf{P} \\ \Sigma_3^+(\mathbf{c}_3 - \mathbf{v}) &= \Sigma_1^+(\mathbf{c}_1 - \mathbf{v}) + \Sigma_2^+(\mathbf{c}_2 - \mathbf{v}). \end{aligned}$$

Proof of Theorem 5.5. We adopt the notation from the algorithm, let $V = \mathrm{span}(\left[\begin{smallmatrix}\mathbf{R} \\ \mathbf{I}\end{smallmatrix}\right]) \subset \mathbb{R}^m$, let $\mathbf{P}$ be the matrix that projects orthogonally onto V, and define the lattice $\Lambda = \mathbb{Z}^m \cap V = \mathcal{L}(\left[\begin{smallmatrix}\mathbf{R} \\ \mathbf{I}\end{smallmatrix}\right])$, which spans V. We analyze the output distribution of SampleD. Clearly, it always outputs an element of $\Lambda_{\mathbf{u}}^\perp(\mathbf{A})$, so let $\bar{\mathbf{x}} \in \Lambda_{\mathbf{u}}^\perp(\mathbf{A})$ be arbitrary. Now SampleD outputs $\bar{\mathbf{x}}$ exactly when it chooses in Step 1 some $\bar{\mathbf{p}} \in V + \bar{\mathbf{x}}$, followed in

Algorithm 3 Efficient algorithm $\mathsf{SampleD}^{\mathcal{O}}(\mathbf{R}, \bar{\mathbf{A}}, \mathbf{H}, \mathbf{u}, s)$ for sampling a discrete Gaussian over $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$.

Input: An oracle $\mathcal{O}(\mathbf{v})$ for Gaussian sampling over a desired coset $\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})$ with fixed parameter $r\sqrt{\Sigma_{\mathbf{G}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}))$, for some $\Sigma_{\mathbf{G}} \geq 2$ and $\epsilon \leq 1/2$.

Offline phase:

- partial parity-check matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$;
- trapdoor matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$;
- positive definite $\Sigma \geq \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right](2 + \Sigma_{\mathbf{G}})[\,\mathbf{R}^t \ \mathbf{I}\,]$, e.g., any $\Sigma = s^2 \geq (s_1(\mathbf{R})^2 + 1)(s_1(\Sigma_{\mathbf{G}}) + 2)$.

Online phase:

- invertible tag $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ defining $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{H}\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n \times m}$, for $m = \bar{m} + w$ ($\mathbf{H}$ may instead be provided in the offline phase, if it is known then);
- syndrome $\mathbf{u} \in \mathbb{Z}_q^n$.

Output: A vector $\mathbf{x}$ drawn from a distribution within $O(\epsilon)$ statistical distance of $D_{\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}), r \cdot \sqrt{\Sigma}}$.

Offline phase:

1: Choose a fresh perturbation $\mathbf{p} \leftarrow D_{\mathbb{Z}^m, r\sqrt{\Sigma_{\mathbf{p}}}}$, where $\Sigma_{\mathbf{p}} = \Sigma - \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \Sigma_{\mathbf{G}} \left[\,\mathbf{R}^t \ \mathbf{I}\,\right] \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\,\mathbf{R}^t \ \mathbf{I}\,]$.

2: Let $\mathbf{p} = \left[\begin{smallmatrix}\mathbf{p}_1\\ \mathbf{p}_2\end{smallmatrix}\right]$ for $\mathbf{p}_1 \in \mathbb{Z}^{\bar{m}}$, $\mathbf{p}_2 \in \mathbb{Z}^w$, and compute $\bar{\mathbf{w}} = \bar{\mathbf{A}}(\mathbf{p}_1 - \mathbf{R}\mathbf{p}_2) \in \mathbb{Z}_q^n$ and $\mathbf{w} = \mathbf{G}\mathbf{p}_2 \in \mathbb{Z}_q^n$.

Online phase:

3: Let $\mathbf{v} \leftarrow \mathbf{H}^{-1}(\mathbf{u} - \bar{\mathbf{w}}) - \mathbf{w} = \mathbf{H}^{-1}(\mathbf{u} - \mathbf{A}\mathbf{p}) \in \mathbb{Z}_q^n$, and choose $\mathbf{z} \leftarrow D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}), r\sqrt{\Sigma_{\mathbf{G}}}}$ by calling $\mathcal{O}(\mathbf{v})$.

4: **return** $\mathbf{x} \leftarrow \mathbf{p} + \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\mathbf{z}$.

Step 3 by the unique $\bar{\mathbf{z}} \in \Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})$ such that $\bar{\mathbf{x}} - \bar{\mathbf{p}} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\bar{\mathbf{z}}$. It is easy to check that $\rho_{\sqrt{\Sigma_{\mathbf{G}}}}(\bar{\mathbf{z}}) = \rho_{\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{x}} - \bar{\mathbf{p}})$, where

$$\Sigma_{\mathbf{y}} = \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \Sigma_{\mathbf{G}} \left[\,\mathbf{R}^t \ \mathbf{I}\,\right] \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\,\mathbf{R}^t \ \mathbf{I}\,]$$

is the covariance matrix with $\mathrm{span}(\Sigma_{\mathbf{y}}) = V$. Note that $\Sigma_{\mathbf{p}} + \Sigma_{\mathbf{y}} = \Sigma$ by definition of $\Sigma_{\mathbf{p}}$, and that $\mathrm{span}(\Sigma_{\mathbf{p}}) = \mathbb{R}^m$ because $\Sigma_{\mathbf{p}} > \mathbf{0}$. Therefore, we have (where C denotes a normalizing constant that may vary from line to line, but does not depend on $\bar{\mathbf{x}}$):

$$\begin{aligned}
p_{\bar{\mathbf{x}}} &= \Pr[\mathsf{SampleD} \text{ outputs } \bar{\mathbf{x}}] \\
&= \sum_{\bar{\mathbf{p}} \in \mathbb{Z}^m \cap (V + \bar{\mathbf{x}})} D_{\mathbb{Z}^m, r\sqrt{\Sigma_{\mathbf{p}}}}(\bar{\mathbf{p}}) \cdot D_{\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G}), r\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{z}}) && \text{(def. of SampleD)} \\
&= C \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_{\mathbf{p}}}}(\bar{\mathbf{p}}) \cdot \rho_{r\sqrt{\Sigma_{\mathbf{y}}}}(\bar{\mathbf{p}} - \bar{\mathbf{x}}) / \rho_{r\sqrt{\Sigma_{\mathbf{G}}}}(\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})) && \text{(def. of } D) \\
&= C \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_3}}(\bar{\mathbf{p}} - \mathbf{c}_3) / \rho_{r\sqrt{\Sigma_{\mathbf{G}}}}(\Lambda^{\perp}_{\mathbf{v}}(\mathbf{G})) && \text{(Fact 5.6)} \\
&\in C[1, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \sum_{\bar{\mathbf{p}}} \rho_{r\sqrt{\Sigma_3}}(\bar{\mathbf{p}} - \mathbf{c}_3) && \text{(Lemma 2.5 and } r\sqrt{\Sigma_{\mathbf{G}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}))) \\
&= C[1, \tfrac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}}) \cdot \rho_{r\sqrt{\Sigma_3}}(\mathbb{Z}^m \cap (V + \bar{\mathbf{x}}) - \mathbf{c}_3), && (5.1)
\end{aligned}$$

where $\Sigma_3^+ = \mathbf{P}(\Sigma_{\mathbf{p}}^+ + \Sigma_{\mathbf{y}}^+)\mathbf{P}$ and $\mathbf{c}_3 \in \mathbf{v} + V = \bar{\mathbf{x}} + V$, because the component of $\bar{\mathbf{x}}$ orthogonal to V is the unique point $\mathbf{v} \in (V + \bar{\mathbf{x}}) \cap V^{\perp}$. Therefore,

$$\mathbb{Z}^m \cap (V + \bar{\mathbf{x}}) - \mathbf{c}_3 = (\mathbb{Z}^m \cap V) + (\bar{\mathbf{x}} - \mathbf{c}_3) \subset V$$

is a coset of the lattice $\Lambda = \mathcal{L}(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right])$. It remains to show that $r\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda)$, so that the rightmost term in (5.1) above is essentially a constant (up to some factor in $[\frac{1-\epsilon}{1+\epsilon}, 1]$) independent of $\bar{\mathbf{x}}$, by Lemma 2.5. Then we can conclude that $p_{\bar{\mathbf{x}}} \in [\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}] \cdot \rho_{r\sqrt{\Sigma}}(\bar{\mathbf{x}})$, from which the theorem follows.

To show that $r\sqrt{\Sigma_3} \geq \eta_\epsilon(\Lambda)$, note that since $\Lambda^* \subset V$, for any covariance Π we have $\rho_{\mathbf{P}\sqrt{\Pi}}(\Lambda^*) = \rho_{\sqrt{\Pi}}(\Lambda^*)$, and so $\mathbf{P}\sqrt{\Pi} \geq \eta_\epsilon(\Lambda)$ if and only if $\sqrt{\Pi} \geq \eta_\epsilon(\Lambda)$. Now because both $\Sigma_\mathbf{p}, \Sigma_\mathbf{y} \geq 2\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\mathbf{R}^t \ \mathbf{I}]$, we have

$$\Sigma_\mathbf{p}^+ + \Sigma_\mathbf{y}^+ \leq (\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right][\mathbf{R}^t \ \mathbf{I}])^+.$$

Because $r\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \geq \eta_\epsilon(\Lambda)$ for $\epsilon = \mathrm{negl}(n)$ by Lemma 2.3, we have $r\sqrt{\Sigma_3} = r\sqrt{(\Sigma_\mathbf{p}^+ + \Sigma_\mathbf{y}^+)^+} \geq \eta_\epsilon(\Lambda)$, as desired. □

5.5 Trapdoor Delegation

Here we describe very simple and efficient mechanism for securely delegating a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ to a trapdoor for an extension $\mathbf{A}' \in \mathbb{Z}_q^{n\times m'}$ of $\mathbf{A}$. Our method has several advantages over the previous basis delegation algorithm of [CHKP10]: first and most importantly, the size of the delegated trapdoor grows only linearly with the dimension m' of $\Lambda^\perp(\mathbf{A}')$, rather than quadratically. Second, the algorithm is much more efficient, because it does not require testing linear independence of Gaussian samples, nor computing the expensive ToBasis and Hermite normal form operations. Third, the resulting trapdoor $\mathbf{R}$ has a 'nice' Gaussian distribution that is easy to analyze and may be useful in applications. We do note that while the delegation algorithm from [CHKP10] works for *any* extension $\mathbf{A}'$ of $\mathbf{A}$ (including $\mathbf{A}$ itself), ours requires $m' \geq m + w$. Fortunately, this is frequently the case in applications such as HIBE and others that use delegation.

Algorithm 4 Efficient algorithm $\mathsf{DelTrap}^{\mathcal{O}}(\mathbf{A}' = [\mathbf{A} \mid \mathbf{A}_1], \mathbf{H}', s')$ for delegating a trapdoor.

Input: an oracle $\mathcal{O}$ for discrete Gaussian sampling over cosets of $\Lambda = \Lambda^\perp(\mathbf{A})$ with parameter $s' \geq \eta_\epsilon(\Lambda)$.

- parity-check matrix $\mathbf{A}' = [\mathbf{A} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times w}$;
- invertible matrix $\mathbf{H}' \in \mathbb{Z}_q^{n\times n}$;

Output: a trapdoor $\mathbf{R}' \in \mathbb{Z}^{m\times w}$ for $\mathbf{A}'$ with tag $\mathbf{H} \in \mathbb{Z}_q^{n\times n}$.

1: Using $\mathcal{O}$, sample each column of $\mathbf{R}'$ independently from a discrete Gaussian with parameter s' over the appropriate coset of $\Lambda^\perp(\mathbf{A})$, so that $\mathbf{A}\mathbf{R}' = \mathbf{H}'\mathbf{G} - \mathbf{A}_1$.

Usually, the oracle $\mathcal{O}$ needed by Algorithm 4 would be implemented (up to $\mathrm{negl}(n)$ statistical distance) by Algorithm 3 above, using a trapdoor $\mathbf{R}$ for $\mathbf{A}$ where $s_1(\mathbf{R})$ is sufficiently small relative to s'. The following is immediate from Lemma 2.9 and the fact that the columns of $\mathbf{R}'$ are independent and $\mathrm{negl}(n)$-subgaussian. A relatively tight bound on the hidden constant factor can also be derived from Lemma 2.9.

Lemma 5.7. *For any valid inputs $\mathbf{A}'$ and $\mathbf{H}'$, Algorithm 4 outputs a trapdoor $\mathbf{R}'$ for $\mathbf{A}'$ with tag $\mathbf{H}'$, whose distribution is the same for any valid implementation of $\mathcal{O}$, and $s_1(\mathbf{R}') \leq s' \cdot O(\sqrt{m} + \sqrt{w})$ except with negligible probability.*

6 Applications

The main applications of "strong" trapdoors have included digital signature schemes in both the random-oracle and standard models, encryption secure under chosen-ciphertext attack (CCA), and (hierarchical) identity-based encryption. Here we focus on signature schemes and CCA-secure encryption, where our techniques lead to significant new improvements (beyond what is obtained by plugging in our trapdoor generator as a "black box"). Where appropriate, we also briefly mention the improvements that are possible in the remaining applications.

6.1 Algebraic Background

In our applications we need a special collection of elements from a certain ring $\mathcal{R}$, which induce invertible matrices $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ as required by our trapdoor construction. We construct such a ring using ideas from the literature on secret sharing over groups and modules, e.g., [DF94, Feh98]. Define the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ for some monic degree-n polynomial $f(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_0 \in \mathbb{Z}[x]$ that is irreducible modulo every prime p dividing q. (Such an $f(x)$ can be constructed by finding monic irreducible degree-n polynomials in $\mathbb{Z}_p[x]$ for each prime p dividing q, and using the Chinese remainder theorem on their coefficients to get $f(x)$.) Recall that $\mathcal{R}$ is a free $\mathbb{Z}_q$-module of rank n, i.e., the elements of $\mathcal{R}$ can be represented as vectors in $\mathbb{Z}_q^n$ relative to the standard basis of monomials $1, x, \ldots, x^{n-1}$. Multiplication by any fixed element of $\mathcal{R}$ then acts as a linear transformation on $\mathbb{Z}_q^n$ according to the rule $x \cdot (a_0, \ldots, a_{n-1})^t = (0, a_0, \ldots, a_{n-2})^t - a_{n-1}(f_0, f_1, \ldots, f_{n-1})^t$, and so can be represented by an (efficiently computable) matrix in $\mathbb{Z}_q^{n \times n}$ relative to the standard basis. In other words, there is an injective ring homomorphism $h \colon \mathcal{R} \to \mathbb{Z}_q^{n \times n}$ that maps any $a \in \mathcal{R}$ to the matrix $\mathbf{H} = h(a)$ representing multiplication by a. In particular, $\mathbf{H}$ is invertible if and only if $a \in \mathcal{R}^*$, the set of units in $\mathcal{R}$. By the Chinese remainder theorem, and because $\mathbb{Z}_p[x]/(f(x))$ is a field by construction of $f(x)$, an element $a \in \mathcal{R}$ is a unit exactly when it is nonzero (as a polynomial residue) modulo every prime p dividing q. We use this fact quite essentially in the constructions that follow.

6.2 Signature Schemes

6.2.1 Definitions

A signature scheme SIG for a message space $\mathcal{M}$ (which may depend on the security parameter n) is a tuple of PPT algorithms as follows:

- $\mathsf{Gen}(1^n)$ outputs a verification key vk and a signing key sk.
- $\mathsf{Sign}(sk, \mu)$, given a signing key sk and a message $\mu \in \mathcal{M}$, outputs a signature $\sigma \in \{0,1\}^*$.
- $\mathsf{Ver}(vk, \mu, \sigma)$, given a verification key vk, a message μ, and a signature σ, either accepts or rejects.

The correctness requirement is: for any $\mu \in \mathcal{M}$, generate $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $\sigma \leftarrow \mathsf{Sign}(sk, \mu)$. Then $\mathsf{Ver}(vk, \mu, \sigma)$ should accept with overwhelming probability (over all the randomness in the experiment).

We recall two standard notions of security for signatures. An intermediate notion is strong unforgeability under *static* chosen-message attack, or su-scma security, is defined as follows: first, the forger $\mathcal{F}$ outputs a list of *distinct* query messages $\mu^{(1)}, \ldots, \mu^{(Q)}$ for some Q. (The distinctness condition simplifies our construction, and does not affect the notion's usefulness.) Next, we generate $(vk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $\sigma^{(i)} \leftarrow \mathsf{Sign}(sk, \mu^{(i)})$ for each $i \in [Q]$, then give vk and each $\sigma^{(i)}$ to $\mathcal{F}$. Finally, $\mathcal{F}$ outputs an attempted forgery (μ^*, σ^*). The forger's advantage $\mathbf{Adv}^{\text{su-scma}}_{\text{SIG}}(\mathcal{F})$ is the probability that $\mathsf{Ver}(vk, \mu^*, \sigma^*)$ accepts and $(\mu^*, \sigma^*) \neq (\mu^{(i)}, \sigma^{(i)})$ for all $i \in [Q]$, taken over all the randomness of the experiment. The

scheme is su-scma-secure if $\mathbf{Adv}^{\text{su-scma}}_{\text{SIG}}(\mathcal{F}) = \text{negl}(n)$ for every nonuniform probabilistic polynomial-time algorithm $\mathcal{F}$.

Another notion, called strong existential unforgeability under *adaptive* chosen-message attack, or su-acma security, is defined similarly, except that $\mathcal{F}$ is first given vk and may adaptively choose the messages $\mu^{(i)}$ to be signed, which need not be distinct.

Using a family of *chameleon* hash functions, there is a generic transformation from eu-scma- to eu-acma-security; see, e.g., [KR00]. Furthermore, the transformation results in an *offline/online* scheme in which the Sign algorithm can be precomputed before the message to be signed is known; see [ST01]. The basic idea is that the signer chameleon hashes the true message, then signs the hash value using the eu-scma-secure scheme (and includes the randomness used in the chameleon hash with the final signature). A suitable type of chameleon hash function has been constructed under a weak hardness-of-SIS assumption; see [CHKP10].

6.2.2 Standard Model Scheme

Here we give a signature scheme that is statically secure in the standard model. The scheme itself is essentially identical (up to the improved and generalized parameters) to the one of [Boy10], which is a lattice analogue of the pairing-based signature of [Wat05]. We give a new proof with an improved security reduction that relies on a weaker assumption. The proof uses a variant of the "prefix technique" [HW09] also used in [CHKP10].

Our scheme involves a number of parameters. For simplicity, we give some exemplary asymptotic bounds here. (Other slight trade-offs among the parameters are possible, and more precise values can be obtained using the more exact bounds from earlier in the paper and the material below.) In what follows, $\omega(\sqrt{\log n})$ represents a fixed function that asymptotically grows faster than $\sqrt{\log n}$.

- $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for large enough $q = \text{poly}(n)$ and $k = \lceil \log q \rceil = O(\log n)$, with the ability to sample from cosets of $\Lambda^\perp(\mathbf{G})$ with Gaussian parameter $O(1) \cdot \omega(\sqrt{\log n}) \geq \eta_\epsilon(\Lambda^\perp(\mathbf{G}))$. (See for example the constructions from Section 4.)
- $\bar{m} = O(nk)$ and $\mathcal{D} = D^{\bar{m} \times nk}_{\mathbb{Z}, \omega(\sqrt{\log n})}$ so that $(\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{R})$ is $\text{negl}(n)$-far from uniform for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, and $m = \bar{m} + 2nk$ is the total dimension of the signatures.
- ℓ is a suitable message length (see below), and $s = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ is a sufficiently large Gaussian parameter.

The legal values of ℓ are influenced by the choice of q and n. Our security proof requires a special collection of units in the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ as constructed in Section 6.1 above. We need a sequence of ℓ units $u_1, \ldots, u_\ell \in \mathcal{R}^*$, not necessarily distinct, such that any nontrivial subset-sum is also a unit, i.e., for any nonempty $S \subseteq [\ell]$, $\sum_{i \in S} u_i \in \mathcal{R}^*$. By the characterization of units in $\mathcal{R}$ described in Section 6.1, letting p be the smallest prime dividing q, we can allow any $\ell \leq (p-1) \cdot n$ by taking $p-1$ copies of each of the monomials $x^i \in \mathcal{R}^*$ for $i = 0, \ldots, n-1$.

The signature scheme has message space $\{0,1\}^\ell$, and is defined as follows.

- $\mathsf{Gen}(1^n)$: choose $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$, choose $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times nk}$ from distribution $\mathcal{D}$, and let $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

  For $i = 0, 1, \ldots, \ell$, choose $\mathbf{A}_i \leftarrow \mathbb{Z}_q^{n \times nk}$. Also choose a syndrome $\mathbf{u} \leftarrow \mathbb{Z}_q^n$.

  The public verification key is $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$. The secret signing key is $sk = \mathbf{R}$.

- $\mathsf{Sign}(sk, \mu \in \{0,1\}^\ell)$: let $\mathbf{A}_\mu = \left[\mathbf{A} \mid \mathbf{A}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{A}_i\right] \in \mathbb{Z}_q^{n \times m}$, where $\mu_i \in \{0,1\}$ is the ith bit of μ, interpreted as an integer. Output $\mathbf{v} \in \mathbb{Z}^m$ sampled from $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu), s}$, using $\mathsf{SampleD}$ with trapdoor $\mathbf{R}$ for $\mathbf{A}$ (which is also a trapdoor for its extension $\mathbf{A}_\mu$).

- $\mathsf{Ver}(vk, \mu, \mathbf{v})$: let $\mathbf{A}_\mu$ be as above. Accept if $\|\mathbf{v}\| \le s \cdot \sqrt{m}$ and $\mathbf{A}_\mu \cdot \mathbf{v} = \mathbf{u}$; otherwise, reject.

Notice that the signing process takes $O(\ell n^2 k)$ scalar operations (to add up the $\mathbf{A}_i$s), but after transforming the scheme to a fully secure one using chameleon hashing, these computations can be performed offline before the message is known.

Theorem 6.1. *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking the $\mathsf{SIS}_{q,\beta}$ problem for large enough $\beta = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$ such that, for any adversary $\mathcal{F}$ mounting an su-scma attack on SIG and making at most Q queries,*

$$\mathbf{Adv}_{\mathsf{SIS}_{q,\beta}}(\mathcal{S}^{\mathcal{F}}) \ge \mathbf{Adv}^{\textit{su-scma}}_{\textit{SIG}}(\mathcal{F})/(2(\ell-1)Q+2) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{F}$ be an adversary mounting an su-scma attack on SIG, having advantage $\delta = \mathbf{Adv}^{\text{su-scma}}_{\text{SIG}}(\mathcal{F})$. We construct a reduction $\mathcal{S}$ attacking $\mathsf{SIS}_{q,\beta}$. The reduction $\mathcal{S}$ takes as input $\bar{m} + nk + 1$ uniformly random and independent samples from $\mathbb{Z}_q^n$, parsing them as a matrix $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{B}] \in \mathbb{Z}_q^{n \times (\bar{m}+nk)}$ and syndrome $\mathbf{u}' \in \mathbb{Z}_q^n$. It will use $\mathcal{F}$ either to find some $\mathbf{z} \in \mathbb{Z}^m$ of length $\|\mathbf{z}\| \le \beta - 1$ such that $\mathbf{Az} = \mathbf{u}'$ (from which it follows that $[\mathbf{A} \mid \mathbf{u}'] \cdot \mathbf{z}' = \mathbf{0}$, where $\mathbf{z}' = \left[\begin{smallmatrix}\mathbf{z}\\ -1\end{smallmatrix}\right]$ is nonzero and of length at most β), or a nonzero $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0}$ (from which is follows that $[\mathbf{A} \mid \mathbf{u}'] \cdot \left[\begin{smallmatrix}\mathbf{z}\\ 0\end{smallmatrix}\right] = \mathbf{0}$).

We distinguish between two types of forger $\mathcal{F}$: one that produces a forgery on an unqueried message (a violation of standard existential unforgeability), and one that produces a new signature on a queried message (a violation of strong unforgeability). Clearly any $\mathcal{F}$ with advantage δ has probability at least $\delta/2$ of succeeding in at least one of these two tasks.

First we consider $\mathcal{F}$ that forges on an unqueried message (with probability at least $\delta/2$). Our reduction $\mathcal{S}$ simulates the static chosen-message attack to $\mathcal{F}$ as follows:

- Invoke $\mathcal{F}$ to receive up to Q messages $\mu^{(1)}, \mu^{(2)}, \ldots \in \{0,1\}^\ell$. Compute the set P of all strings $p \in \{0,1\}^{\le \ell}$ having the property that p is a shortest string for which no $\mu^{(j)}$ has p as a prefix. Equivalently, P represents the set of maximal subtrees of $\{0,1\}^{\le \ell}$ (viewed as a tree) that do not contain any of the queried messages. The set P has size at most $(\ell-1) \cdot Q + 1$, and may be computed efficiently. (See, e.g., [CHKP10] for a precise description of an algorithm.) Choose some p from P uniformly at random, letting $t = |p| \le \ell$.
- Construct a verification key $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u} = \mathbf{u}')$: for $i = 0, \ldots, \ell$, choose $\mathbf{R}_i \leftarrow \mathcal{D}$, and let

$$\mathbf{A}_i = \mathbf{H}_i\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}_i, \text{ where } \mathbf{H}_i = \begin{cases} h(0) = \mathbf{0} & i > t \\ (-1)^{p_i} \cdot h(u_i) & i \in [t] \\ -\sum_{j \in [t]} p_j \cdot \mathbf{H}_j & i = 0 \end{cases}.$$

  (Recall that $u_1, \ldots, u_\ell \in \mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ are units whose nontrivial subset-sums are also units.)

  Note that by hypothesis on $\bar{m}$ and $\mathcal{D}$, for any choice of p the key vk is only $\mathrm{negl}(n)$-far from uniform in statistical distance. Note also that by our choice of the $\mathbf{H}_i$, for any message $\mu \in \{0,1\}^\ell$ having p as a prefix, we have $\mathbf{H}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{H}_i = \mathbf{0}$. Whereas for any $\mu \in \{0,1\}^\ell$ having $p' \ne p$ as its t-bit prefix, we have

$$\mathbf{H}_0 + \sum_{i \in [\ell]} \mu_i \mathbf{H}_i = \sum_{i \in [t]} (p'_i - p_i) \cdot \mathbf{H}_i = \sum_{i \in [t], p'_i \ne p_i} (-1)^{p_i} \cdot \mathbf{H}_i = h\Big(\sum_{i \in [t], p'_i \ne p_i} u_i\Big),$$

  which is invertible by hypothesis on the u_is. Finally, observe that with overwhelming probability over any fixed choice of vk and the $\mathbf{H}_i$, each column of each $\mathbf{R}_i$ is still independently distributed as

a discrete Gaussian with parameter $\omega(\sqrt{\log n}) \geq \eta_\epsilon(\bar{\mathbf{A}})$ over some fixed coset of $\Lambda^\perp(\bar{\mathbf{A}})$, for some negligible $\epsilon = \epsilon(n)$.

- Generate signatures for the queried messages: for each message $\mu = \mu^{(i)}$, compute

$$\mathbf{A}_\mu = \big[\mathbf{A} \mid \mathbf{A}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{A}_i\big] = \big[\bar{\mathbf{A}} \mid \mathbf{B} \mid \mathbf{HG} - \bar{\mathbf{A}}\big(\mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{R}_i\big)\big],$$

  where $\mathbf{H}$ is invertible because the t-bit prefix of μ is not p. Therefore, $\mathbf{R} = (\mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i \mathbf{R}_i)$ is a trapdoor for $\mathbf{A}_\mu$. By the conditional distribution on the $\mathbf{R}_i$s, concatenation of subgaussian random variables, and Lemma 2.9, we have

$$s_1(\mathbf{R}) = \sqrt{\ell+1} \cdot O(\sqrt{\bar{m}} + \sqrt{nk}) \cdot \omega(\sqrt{\log n}) = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})$$

  with overwhelming probability. Since $s = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ is sufficiently large, we can generate a properly distributed signature $\mathbf{v}_\mu \leftarrow D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu),s}$ using $\mathsf{SampleD}$ with trapdoor $\mathbf{R}$.

Next, $\mathcal{S}$ gives vk and the generated signatures to $\mathcal{F}$. Because vk and the signatures are distributed within $\mathrm{negl}(n)$ statistical distance of those in the real attack (for any choice of the prefix p), with probability at least $\delta/2 - \mathrm{negl}(n)$, $\mathcal{F}$ outputs a forgery $(\mu^*, \mathbf{v}^*)$ where μ^* is different from all the queried messages, $\mathbf{A}_{\mu^*}\mathbf{v}^* = \mathbf{u}$, and $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m}$. Furthermore, conditioned on this event, μ^* has p as a prefix with probability at least $1/((\ell-1)Q+1) - \mathrm{negl}(n)$, because p is still essentially uniform in P conditioned on the view of $\mathcal{F}$. Therefore, all of these events occur with probability at least $\delta/(2(\ell-1)Q+2) - \mathrm{negl}(n)$.

In such a case, $\mathcal{S}$ extracts a solution to its SIS challenge instance from the forgery $(\mu^*, \mathbf{v}^*)$ as follows. Because μ^* starts with p, we have $\mathbf{A}_{\mu^*} = \big[\bar{\mathbf{A}} \mid \mathbf{B} \mid -\bar{\mathbf{A}}\mathbf{R}^*\big]$ for $\mathbf{R}^* = \mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i^* \mathbf{R}_i$, and so

$$\underbrace{[\bar{\mathbf{A}} \mid \mathbf{B}]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{I}_{\bar{m}} & & -\mathbf{R}^* \\ & \mathbf{I}_{nk} & \end{bmatrix} \mathbf{v}^*}_{\mathbf{z}} = \mathbf{u} \bmod q,$$

as desired. Because $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m} = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})^2$ and $s_1(\mathbf{R}^*) = \sqrt{\ell+1} \cdot O(\sqrt{\bar{m}} + \sqrt{nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability (conditioned on the view of $\mathcal{F}$ and any fixed $\mathbf{H}_i$), we have $\|\mathbf{z}\| = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$, which is at most $\beta - 1$, as desired.

Now we consider an $\mathcal{F}$ that forges on one of its queried messages (with probability at least $\delta/2$). Our reduction $\mathcal{S}$ simulates the attack to $\mathcal{F}$ as follows:

- Invoke $\mathcal{F}$ to receive up to Q *distinct* messages $\mu^{(1)}, \mu^{(2)}, \ldots \in \{0,1\}^\ell$. Choose one of these messages $\mu = \mu^{(i)}$ uniformly at random, "guessing" that the eventual forgery will be on μ.
- Construct a verification key $vk = (\mathbf{A}, \mathbf{A}_0, \ldots, \mathbf{A}_\ell, \mathbf{u})$: generate $\mathbf{A}_i$ exactly as above, using $p = \mu$. Then choose $\mathbf{v} \leftarrow D_{\mathbb{Z}^m,s}$ and let $\mathbf{u} = \mathbf{A}_\mu \mathbf{v}$, where $\mathbf{A}_\mu$ is defined in the usual way.
- Generate signatures for the queried messages: for all the queries except μ, proceed exactly as above (which is possible because all the queries are distinct and hence do not have $p = \mu$ as a prefix). For μ, use $\mathbf{v}$ as the signature, which has the required distribution $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_\mu),s}$ by construction.

When $\mathcal{S}$ gives vk and the signatures to $\mathcal{F}$, with probability at least $\delta/2 - \mathrm{negl}(n)$ the forger must output a forgery $(\mu^*, \mathbf{v}^*)$ where μ^* is one of its queries, $\mathbf{v}^*$ is different from the corresponding signature it received, $\mathbf{A}_{\mu^*}\mathbf{v}^* = \mathbf{u}$, and $\|\mathbf{v}^*\| \leq s \cdot \sqrt{m}$. Because vk and the signatures are appropriately distributed for any

choice μ that $\mathcal{S}$ made, conditioned on the above event the probability that $\mu^* = \mu$ is at least $1/Q - \mathrm{negl}(n)$. Therefore, all of these events occur with probability at least $\delta/(2Q) - \mathrm{negl}(n)$.

In such a case, $\mathcal{S}$ extracts a solution to its SIS challenge from the forgery as follows. Because $\mu^* = \mu$, we have $\mathbf{A}_{\mu^*} = \left[\bar{\mathbf{A}} \mid \mathbf{B} \mid -\bar{\mathbf{A}}\mathbf{R}^*\right]$ for $\mathbf{R}^* = \mathbf{R}_0 + \sum_{i\in[\ell]} \mu_i^* \mathbf{R}_i$, and so

$$\underbrace{[\bar{\mathbf{A}} \mid \mathbf{B}]}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{I}_{\bar{m}} & & -\mathbf{R}^* \\ & \mathbf{I}_{nk} & \end{bmatrix} (\mathbf{v}^* - \mathbf{v})}_{\mathbf{z}} = \mathbf{0} \bmod q.$$

Because both $\|\mathbf{v}^*\|, \|\mathbf{v}\| \le s \cdot \sqrt{m} = O(\sqrt{\ell}nk) \cdot \omega(\sqrt{\log n})^2$ and $s_1(\mathbf{R}^*) = O(\sqrt{\ell nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability (conditioned on the view of $\mathcal{F}$ and any fixed $\mathbf{H}_i$), we have $\|\mathbf{z}\| = O(\ell(nk)^{3/2}) \cdot \omega(\sqrt{\log n})^3$ with overwhelming probability, as needed. It just remains to show that $\mathbf{z} \neq \mathbf{0}$ with overwhelming probability. To see this, write $\mathbf{w} = \mathbf{v}^* - \mathbf{v} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \in \mathbb{Z}^{\bar{m}} \times \mathbb{Z}^{nk} \times \mathbb{Z}^{nk}$, with $\mathbf{w} \neq \mathbf{0}$. If $\mathbf{w}_2 \neq \mathbf{0}$ or $\mathbf{w}_3 = \mathbf{0}$, then $\mathbf{z} \neq \mathbf{0}$ and we are done. Otherwise, choose some entry of $\mathbf{w}_3$ that is nonzero; without loss of generality say it is w_m. Let $\mathbf{r} = (\mathbf{R}_0)_{nk}$. Now for any fixed values of $\mathbf{R}_i$ for $i \in [\ell]$ and fixed first $nk - 1$ columns of $\mathbf{R}_0$, we have $\mathbf{z} = \mathbf{0}$ only if $\mathbf{r} \cdot w_m = \mathbf{y} \in \mathbb{R}^{\bar{m}}$ for some fixed $\mathbf{y}$. Conditioned on the adversary's view (specifically, $(\mathbf{A}_0)_{nk} = \bar{\mathbf{A}}\mathbf{r}$), $\mathbf{r}$ is distributed as a discrete Gaussian of parameter $\ge 2\eta_\epsilon(\Lambda^\perp(\bar{\mathbf{A}}))$ for some $\epsilon = \mathrm{negl}(n)$ over a coset of $\Lambda^\perp(\bar{\mathbf{A}})$. Then by Lemma 2.7, we have $\mathbf{r} = \mathbf{y}/w_m$ with only $2^{-\Omega(n)}$ probability, and we are done. □

6.3 Chosen Ciphertext-Secure Encryption

Definitions. A public-key cryptosystem for a message space $\mathcal{M}$ (which may depend on the security parameter) is a tuple of algorithms as follows:

- $\mathsf{Gen}(1^n)$ outputs a public encryption key pk and a secret decryption key sk.
- $\mathsf{Enc}(pk, m)$, given a public key pk and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \{0,1\}^*$.
- $\mathsf{Dec}(sk, c)$, given a decryption key sk and a ciphertext c, outputs some $m \in \mathcal{M} \cup \{\perp\}$.

The correctness requirement is: for any $m \in \mathcal{M}$, generate $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and $c \leftarrow \mathsf{Enc}(pk, m)$. Then $\mathsf{Dec}(sk, c)$ should output m with overwhelming probability (over all the randomness in the experiment).

We recall the two notions of security under chosen-ciphertext attacks. We start with the weaker notion of CCA1 (or "lunchtime") security. Let $\mathcal{A}$ be any nonuniform probabilistic polynomial-time algorithm. First, we generate $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$ and give pk to $\mathcal{A}$. Next, we give $\mathcal{A}$ oracle access to the decryption procedure $\mathsf{Dec}(sk, \cdot)$. Next, $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$ and is given a challenge ciphertext $c \leftarrow \mathsf{Enc}(pk, m_b)$ for either $b = 0$ or $b = 1$. The scheme is CCA1-secure if the views of $\mathcal{A}$ (i.e., the public key pk, the answers to its oracle queries, and the ciphertext c) for $b = 0$ versus $b = 1$ are computationally indistinguishable (i.e., $\mathcal{A}$'s acceptance probabilities for $b = 0$ versus $b = 1$ differ by only $\mathrm{negl}(n)$). In the stronger CCA2 notion, after receiving the challenge ciphertext, $\mathcal{A}$ continues to have access to the decryption oracle $\mathsf{Dec}(sk, \cdot)$ for any query not equal to the challenge ciphertext c; security it defined similarly.

Construction. To highlight the main new ideas, here we present a public-key encryption scheme that is CCA1-secure. Full CCA2 security can be obtained via relatively generic transformations using either strongly unforgeable one-time signatures [DDN00], or a message authentication code and weak form of commitment [BCHK07]; we omit these details.

Our scheme involves a number of parameters, for which we give some exemplary asymptotic bounds. In what follows, $\omega(\sqrt{\log n})$ represents a fixed function that asymptotically grows faster than $\sqrt{\log n}$.

- $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ is a gadget matrix for large enough prime power $q = p^e = \text{poly}(n)$ and $k = O(\log q) = O(\log n)$. We require an oracle $\mathcal{O}$ that solves LWE with respect to $\Lambda(\mathbf{G}^t)$ for any error vector in some $\mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ where $\|\mathbf{B}\| = O(1)$. (See for example the constructions from Section 4.)
- $\bar{m} = O(nk)$ and $\mathcal{D} = D_{\mathbb{Z},\omega(\sqrt{\log n})}^{\bar{m} \times nk}$ so that $(\bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{R})$ is $\text{negl}(n)$-far from uniform for $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, and $m = \bar{m} + nk$ is the total dimension of the public key and ciphertext.
- α is an error rate for LWE, for sufficiently large $1/\alpha = O(nk) \cdot \omega(\sqrt{\log n})$.

Our scheme requires a special collection of elements in the ring $\mathcal{R} = \mathbb{Z}_q[x]/(f(x))$ as constructed in Section 6.1 (recall that here $q = p^e$). We need a very large set $\mathcal{U} = \{u_1, \ldots, u_\ell\} \subset \mathcal{R}$ with the "unit differences" property: for any $i \neq j$, the difference $u_i - u_j \in \mathcal{R}^*$, and hence $h(u_i - u_j) = h(u_i) - h(u_j) \in \mathbb{Z}_q^{n \times n}$ is invertible. (Note that the u_is need not all be units themselves.) Concretely, by the characterization of units in $\mathcal{R}$ given above, we take $\mathcal{U}$ to be all linear combinations of the monomials $1, x, \ldots, x^{n-1}$ with coefficients in $\{0, \ldots, p-1\}$, of which there are exactly p^n. Since the difference between any two such distinct elements is nonzero modulo p, it is a unit.

The system has message space $\{0,1\}^{nk}$, which we map bijectively to the cosets of $\Lambda/2\Lambda$ for $\Lambda = \Lambda(\mathbf{G}^t)$ via some function encode that is efficient to evaluate and invert. Concretely, letting $\mathbf{S} \in \mathbb{Z}^{nk \times nk}$ be any basis of Λ, we can map $\mathbf{m} \in \{0,1\}^{nk}$ to $\mathsf{encode}(\mathbf{m}) = \mathbf{S}\mathbf{m} \in \mathbb{Z}^{nk}$.

- $\mathsf{Gen}(1^n)$: choose $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and $\mathbf{R} \leftarrow \mathcal{D}$, letting $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R} \bmod q$. The public key is $pk = \mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] \in \mathbb{Z}_q^{n \times m}$ and the secret key is $sk = \mathbf{R}$.
- $\mathsf{Enc}(pk = [\bar{\mathbf{A}} \mid \mathbf{A}_1], \mathbf{m} \in \{0,1\}^{nk})$: choose nonzero $u \leftarrow \mathcal{U}$ and let $\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}]$. Choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\bar{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$, and $\mathbf{e}_1 \leftarrow D_{\mathbb{Z},s}^{nk}$ where $s^2 = (\|\bar{\mathbf{e}}\|^2 + \bar{m}(\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$.

  Let
  $$\mathbf{b}^t = 2(\mathbf{s}^t \mathbf{A}_u \bmod q) + \mathbf{e}^t + (\mathbf{0}, \mathsf{encode}(\mathbf{m}))^t \bmod 2q,$$
  where $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1) \in \mathbb{Z}^m$ and $\mathbf{0}$ has dimension $\bar{m}$. (Note the use of mod-$2q$ arithmetic: $2(\mathbf{s}^t \mathbf{A}_u \bmod q)$ is an element of the lattice $2\Lambda(\mathbf{A}_u^t) \supseteq 2q\mathbb{Z}^m$.) Output the ciphertext $c = (u, \mathbf{b}) \in \mathcal{U} \times \mathbb{Z}_{2q}^m$.
- $\mathsf{Dec}(sk = \mathbf{R}, c = (u, \mathbf{b}) \in \mathcal{U} \times \mathbb{Z}_{2q}^m)$: Let $\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}] = [\bar{\mathbf{A}} \mid h(u)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$.

  1. If c does not parse or $u = 0$, output $\perp$. Otherwise, call $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}_u, \mathbf{b} \bmod q)$ to get values $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1) \in \mathbb{Z}^{\bar{m}} \times \mathbb{Z}^{nk}$ for which $\mathbf{b}^t = \mathbf{z}^t \mathbf{A}_u + \mathbf{e}^t \bmod q$. (Note that $h(u) \in \mathbb{Z}_q^{n \times n}$ is invertible, as required by Invert.) If the call to Invert fails for any reason, output $\perp$.
  2. If $\|\bar{\mathbf{e}}\| \geq \alpha q \sqrt{\bar{m}}$ or $\|\mathbf{e}_1\| \geq \alpha q \sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$, output $\perp$.
  3. Let $\mathbf{v} = \mathbf{b} - \mathbf{e} \bmod 2q$, parsed as $\mathbf{v} = (\bar{\mathbf{v}}, \mathbf{v}_1) \in \mathbb{Z}_{2q}^{\bar{m}} \times \mathbb{Z}_{2q}^{nk}$. If $\bar{\mathbf{v}} \notin 2\Lambda(\bar{\mathbf{A}}^t)$, output $\perp$. Finally, output $\mathsf{encode}^{-1}(\mathbf{v}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] \bmod 2q) \in \{0,1\}^{nk}$ if it exists, otherwise output $\perp$.

  (In practice, to avoid timing attacks one would perform all of the Dec operations first, and only then finally output $\perp$ if any of the validity tests failed.)

Lemma 6.2. *The above scheme has only $2^{-\Omega(n)}$ probability of decryption error.*

The error probability can be made zero by changing Gen and Enc so that they resample $\mathbf{R}$, $\bar{\mathbf{e}}$, and/or $\mathbf{e}_1$ in the rare event that they violate the corresponding bounds given in the proof below.

Proof. Let $(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{Gen}(1^n)$. By Lemma 2.9, we have $s_1(\mathbf{R}) \le O(\sqrt{nk}) \cdot \omega(\sqrt{\log n})$ except with probability $2^{-\Omega(n)}$. Now consider the random choices made by $\mathsf{Enc}(\mathbf{A}, \mathbf{m})$ for arbitrary $\mathbf{m} \in \{0,1\}^{nk}$. By Lemma 2.6, we have both $\|\bar{\mathbf{e}}\| < \alpha q \sqrt{\bar{m}}$ and $\|\mathbf{e}_1\| < \alpha q \sqrt{2\bar{m}nk} \cdot \omega(\sqrt{\log n})$, except with probability $2^{-\Omega(n)}$. Letting $\mathbf{e} = (\bar{\mathbf{e}}, \mathbf{e}_1)$, we have

$$\left\|\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\right\| \le \|\bar{\mathbf{e}}^t \mathbf{R}\| + \|\mathbf{e}_1\| < \alpha q \cdot O(nk) \cdot \omega(\sqrt{\log n}).$$

In particular, for large enough $1/\alpha = O(nk) \cdot \omega(\sqrt{\log n})$ we have $\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$. Therefore, the call to Invert made by $\mathsf{Dec}(\mathbf{R}, (u, \mathbf{b}))$ returns $\mathbf{e}$. It follows that for $\mathbf{v} = (\bar{\mathbf{v}}, \mathbf{v}_1) = \mathbf{b} - \mathbf{e} \bmod 2q$, we have $\bar{\mathbf{v}} \in 2\Lambda(\bar{\mathbf{A}}^t)$ as needed. Finally,

$$\mathbf{v}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = 2(\mathbf{s}^t h(u) \mathbf{G} \bmod q) + \mathsf{encode}(\mathbf{m}) \bmod 2q,$$

which is in the coset $\mathsf{encode}(\mathbf{m}) \in \Lambda(\mathbf{G}^t)/2\Lambda(\mathbf{G}^t)$, and so Dec outputs $\mathbf{m}$ as desired. □

Theorem 6.3. *The above scheme is CCA1-secure assuming the hardness of decision-*$\mathsf{LWE}_{q,\alpha'}$ *for* $\alpha' = \alpha/3 \ge 2\sqrt{n}/q$.

Proof. We start by giving a particular form of discretized LWE that we will need below. Given access to an LWE distribution $A_{\mathbf{s},\alpha'}$ over $\mathbb{Z}_q^n \times \mathbb{T}$ for any $\mathbf{s} \in \mathbb{Z}_q^n$ (where recall that $\mathbb{T} = \mathbb{R}/\mathbb{Z}$), by [Pei10, Theorem 3.1] we can transform its samples $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle/q + e \bmod 1)$ to have the form $(\mathbf{a}, 2(\langle \mathbf{s}, \mathbf{a} \rangle \bmod q) + \mathbf{e}' \bmod 2q)$ for $e' \leftarrow D_{\mathbb{Z},\alpha q}$, by mapping $b \mapsto 2qb + D_{\mathbb{Z}-2qb,s} \bmod 2q$ where $s^2 = (\alpha q)^2 - (2\alpha' q)^2 \ge 4n \ge \eta_\epsilon(\mathbb{Z})^2$. This transformation maps the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_{2q}$, so the discretized distribution is pseudorandom under the hypothesis of the theorem.

We proceed via a sequence of hybrid games. The game H_0 is exactly the CCA1 attack with the system described above.

In game H_1, we change how the public key $\mathbf{A}$ and challenge ciphertext $c^* = (u^*, \mathbf{b}^*)$ are constructed, and the way that decryption queries are answered (slightly), but in a way that introduces only $\mathrm{negl}(n)$ statistical difference with H_0. At the start of the experiment we choose nonzero $u^* \leftarrow \mathcal{U}$ and let the public key be $\mathbf{A} = [\bar{\mathbf{A}} \mid \mathbf{A}_1] = [\bar{\mathbf{A}} \mid -h(u^*)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}]$, where $\bar{\mathbf{A}}$ and $\mathbf{R}$ are chosen in the same way as in H_0. (In particular, we still have $s_1(\mathbf{R}) \le O(\sqrt{nk}) \cdot \omega(\sqrt{\log n})$ with overwhelming probability.) Note that $\mathbf{A}$ is still $\mathrm{negl}(n)$-uniform for any choice of u^*, so conditioned on any fixed choice of $\mathbf{A}$, the value of u^* is statistically hidden from the attacker. To aid with decryption queries, we also choose an arbitrary (not necessarily short) $\hat{\mathbf{R}} \in \mathbb{Z}^{\bar{m} \times nk}$ such that $\mathbf{A}_1 = -\bar{\mathbf{A}}\hat{\mathbf{R}} \bmod q$.

To answer a decryption query on a ciphertext $(u, \mathbf{b})$, we use an algorithm very similar to Dec with trapdoor $\mathbf{R}$. After testing whether $u = 0$ (and outputting $\perp$ if so), we call $\mathsf{Invert}^{\mathcal{O}}(\mathbf{R}, \mathbf{A}_u, \mathbf{b} \bmod q)$ to get some $\mathbf{z} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}^m$, where

$$\mathbf{A}_u = [\bar{\mathbf{A}} \mid \mathbf{A}_1 + h(u)\mathbf{G}] = [\bar{\mathbf{A}} \mid h(u - u^*)\mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

(If Invert fails, we output $\perp$.) We then perform steps 2 and 3 on $\mathbf{e} \in \mathbb{Z}^m$ and $\mathbf{v} = \mathbf{b} - \mathbf{e} \bmod 2q$ exactly as in Dec, except that we use $\hat{\mathbf{R}}$ in place of $\mathbf{R}$ when decoding the message in step 3.

We now analyze the behavior of this decryption routine. Whenever $u \ne u^*$, which is the case with overwhelming probability because u^* is statistically hidden, by the "unit differences" property on $\mathcal{U}$ we have that $h(u - u^*) \in \mathbb{Z}_q^{n \times n}$ is invertible, as required by the call to Invert. Now, either there exists an $\mathbf{e}$ that satisfies the validity tests in step 2 and such that $\mathbf{b}^t = \mathbf{z}^t \mathbf{A}_u + \mathbf{e}^t \bmod q$ for some $\mathbf{z} \in \mathbb{Z}_q^n$, or there does not. In the latter case, no matter what Invert does in H_0 and H_1, step 2 will return $\perp$ in both games. Now consider the former case: by the constraints on $\mathbf{e}$, we have $\mathbf{e}^t \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \in \mathcal{P}_{1/2}(q \cdot \mathbf{B}^{-t})$ in both games, so the call to Invert

must return this $\mathbf{e}$ (but possibly different $\mathbf{z}$) in both games. Finally, the result of decryption is the same in both games: if $\bar{\mathbf{v}} \in 2\Lambda(\bar{\mathbf{A}}^t)$ (otherwise, both games return $\perp$), then we can express $\mathbf{v}$ as

$$\mathbf{v}^t = 2(\mathbf{s}^t \mathbf{A}_u \bmod q) + (\mathbf{0}, \mathbf{v}')^t \bmod 2q$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{v}' \in \mathbb{Z}_{2q}^{nk}$. Then for *any* solution $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times nk}$ to $\mathbf{A}_1 = -\bar{\mathbf{A}}\mathbf{R} \bmod q$, we have

$$\mathbf{v}^t \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix}\right] = 2(\mathbf{s}^t h(u)\mathbf{G} \bmod q) + (\mathbf{v}')^t \bmod 2q.$$

In particular, this holds for the $\mathbf{R}$ in H_0 and the $\hat{\mathbf{R}}$ in H_1 that are used for decryption. It follows that both games output $\mathsf{encode}^{-1}(\mathbf{v}')$, if it exists (and $\perp$ otherwise).

Finally, in H_1 we produce the challenge ciphertext $(u, \mathbf{b})$ on a message $\mathbf{m} \in \{0,1\}^{nk}$ as follows. Let $u = u^*$, and choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\bar{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$ as usual, but do not choose $\mathbf{e}_1$. Note that $\mathbf{A}_u = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{R}]$. Let $\bar{\mathbf{b}}^t = 2(\mathbf{s}^t \bar{\mathbf{A}} \bmod q) + \bar{\mathbf{e}}^t \bmod 2q$. Let

$$\mathbf{b}_1^t = -\bar{\mathbf{b}}^t \mathbf{R} + \hat{\mathbf{e}}^t + \mathsf{encode}(\mathbf{m}) \bmod 2q,$$

where $\hat{\mathbf{e}} \leftarrow D_{\mathbb{Z},\alpha q\sqrt{m}\cdot\omega(\sqrt{\log n})}^{nk}$, and output $(u, \mathbf{b} = (\bar{\mathbf{b}}, \mathbf{b}_1))$. We now show that the distribution of $(u, \mathbf{b})$ is within $\mathrm{negl}(n)$ statistical distance of that in H_0, given the attacker's view (i.e., pk and the results of the decryption queries). Clearly, u and $\bar{\mathbf{b}}$ have essentially the same distribution as in H_0, because u is $\mathrm{negl}(n)$-uniform given pk, and by construction of $\bar{\mathbf{b}}$. By substitution, we have

$$\mathbf{b}_1^t = 2(\mathbf{s}^t(-\bar{\mathbf{A}}\mathbf{R}) \bmod q) + (\bar{\mathbf{e}}^t\mathbf{R} + \hat{\mathbf{e}}^t) + \mathsf{encode}(m).$$

Therefore, it suffices to show that for fixed $\bar{\mathbf{e}}$, each $\langle \bar{\mathbf{e}}, \mathbf{r}_i \rangle + \hat{e}_i$ has distribution $\mathrm{negl}(n)$-far from $D_{\mathbb{Z},s}$, where $s^2 = (\|\bar{\mathbf{e}}\|^2 + m(\alpha q)^2) \cdot \omega(\sqrt{\log n})^2$, over the random choice of $\mathbf{r}_i$ (conditioned on the value of $\bar{\mathbf{A}}\mathbf{r}_i$ from the public key) and of $\hat{e}_i$. Because each $\mathbf{r}_i$ is an independent discrete Gaussian over a coset of $\Lambda^{\perp}(\bar{\mathbf{A}})$, the claim follows essentially by [Reg05, Corollary 3.10], but adapted to discrete random variables using [Pei10, Theorem 3.1] in place of [Reg05, Claim 3.9].

In game H_2, we only change how the $\bar{\mathbf{b}}$ component of the challenge ciphertext is created, letting it be uniformly random in $\mathbb{Z}_{2q}^{\bar{m}}$. We construct pk, answer decryption queries, and construct $\mathbf{b}_1$ in exactly the same way as in H_1. First observe that under our (discretized) LWE hardness assumption, games H_1 and H_2 are computationally indistinguishable by an elementary reduction: given $(\bar{\mathbf{A}}, \bar{\mathbf{b}}) \in \mathbb{Z}_q^{n \times \bar{m}} \times \mathbb{Z}_{2q}^{\bar{m}}$ where $\bar{\mathbf{A}}$ is uniformly random and either $\bar{\mathbf{b}}^t = 2(\mathbf{s}^t\bar{\mathbf{A}} \bmod q) + \mathbf{e}^t \bmod 2q$ (for $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},\alpha q}^{\bar{m}}$) or $\bar{\mathbf{b}}$ is uniformly random, we can efficiently emulate either game H_1 or H_2 (respectively) by doing everything exactly as in the two games, except using the given $\bar{\mathbf{A}}$ and $\bar{\mathbf{b}}$ when constructing the public key and challenge ciphertext.

Now by the leftover hash lemma, $(\bar{\mathbf{A}}, \bar{\mathbf{b}}^t, \bar{\mathbf{A}}\mathbf{R}, -\bar{\mathbf{b}}^t\mathbf{R})$ is $\mathrm{negl}(n)$-uniform when $\mathbf{R}$ is chosen as in H_2. Therefore, the challenge ciphertext has the same distribution (up to $\mathrm{negl}(n)$ statistical distance) for any encrypted message, and so the adversary's advantage is negligible. This completes the proof. □

References

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115. 2010.

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618. 2009.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9. 1999.

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48(3):535–553, April 2011. Preliminary version in STACS 2009.

[Bab85] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986. Preliminary version in STACS 1985.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BCHK07] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BFKL93] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO*, pages 278–291. 1993.

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. `http://eprint.iacr.org/`.

[Boy10] X. Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517. 2010.

[BV11a] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*. 2011. To appear.

[BV11b] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524. 2011.

[CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552. 2010.

[CN11] Y. Chen and P. Q. Nguyen. BKZ 2.0: Simulation and better lattice security estimates. In *ASIACRYPT*. 2011. To appear.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DF94] Y. Desmedt and Y. Frankel. Perfect homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM J. Discrete Math.*, 7(4):667–679, 1994.

[Feh98] S. Fehr. *Span Programs over Rings and How to Share a Secret from a Module*. Master's thesis, ETH Zurich, Institute for Theoretical Computer Science, 1998.

[Gen09a] C. Gentry. *A fully homomorphic encryption scheme*. Ph.D. thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[Gen09b] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. 2009.

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131. 1997.

[GH11] C. Gentry and S. Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *FOCS*. 2011. To appear.

[GHV10] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522. 2010.

[GKV10] S. D. Gordon, J. Katz, and V. Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT*, pages 395–412. 2010.

[GN08] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, pages 31–51. 2008.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

[HW09] S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO*, pages 654–670. 2009.

[Kle00] P. N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. 2000.

[KMO10] E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT*, pages 673–692. 2010.

[KR00] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*. 2000.

[LM06] V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155. 2006.

[LM08] V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54. 2008.

[LM09] V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, pages 577–594. 2009.

[LMPR08] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72. 2008.

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339. 2011.

[LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23. 2010.

[Lyu08] V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, pages 162–179. 2008.

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[Mic02] D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

[MM11] D. Micciancio and P. Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484. 2011.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[MR09] D. Micciancio and O. Regev. Lattice-based cryptography. In *Post Quantum Cryptography*, pages 147–191. Springer, February 2009.

[Pei09a] C. Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359, July 2009. `http://eprint.iacr.org/`.

[Pei09b] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342. 2009.

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97. 2010.

[PR06] C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166. 2006.

[PV08] C. Peikert and V. Vaikuntanathan. Noninteractive statistical zero-knowledge proofs for lattice problems. In *CRYPTO*, pages 536–553. 2008.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571. 2008.

[PW08] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196. 2008.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[RS10] M. Rückert and M. Schneider. Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.

[Rüc10] M. Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In *PQCrypto*, pages 182–200. 2010.

[ST01] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *CRYPTO*, pages 355–367. 2001.

[Ver11] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices, January 2011. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`, last accessed 4 Feb 2011.

[vGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43. 2010.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127. 2005.