Trapdoors for Lattices:
Simpler, Tighter, Faster, Smaller

Daniele Micciancio[1]* and Chris Peikert[2]**

[1] University of California, San Diego
[2] Georgia Institute of Technology

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,

* This material is based on research sponsored by NSF under Award CNS-1117936 and 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.

** 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.

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.

– 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 $q = \text{poly}(n)$, 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 $\mathbf{x}$, $\mathbf{e}$.[3] (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 [20, 13, 1, 2], to fully homomorphic encryption schemes [17, 16, 45, 12, 11, 18, 10], and much more (e.g., [29, 40, 26, 38, 39, 35, 6, 42, 9, 19, 22]).

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 $\mathbf{AS} = \mathbf{0} \bmod q$. (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 [20]).

Theoretical solutions for all the above tasks (generating $\mathbf{A}$ with strong trapdoor $\mathbf{S}$ [3, 5], trapdoor inversion of $g_{\mathbf{A}}$ and preimage sampling for $f_{\mathbf{A}}$ [20]) 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 [5] is conceptually and algorithmically complex, and involves costly computations of Hermite normal forms

[3] Inverting these functions corresponds to solving the "short integer solution" (SIS) problem [4] for $f_{\mathbf{A}}$, and the "learning with errors" (LWE) problem [41] for $g_{\mathbf{A}}$, both of which are widely used in lattice cryptography and enjoy provable worst-case hardness.

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 $f_{\mathbf{A}}$ [7, 24, 20] 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 [36], 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 changes in the main parameters, and improvements by even small constant factors can have a significant impact on concrete security. (See, e.g., [15, 34], and the full version for a more detailed discussion.)

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 [3, 5], 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 [7, 24, 20, 36].

Our methods yield large constant-factor improvements, and in some cases even small asymptotic improvements, in the lattice dimension m, trapdoor quality[4] 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. In the full version we detail several improvements to existing applications. We now give a detailed comparison of our results with the most relevant prior works [3, 5, 20, 36]. 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 construc-

[4] There are several notions quality for lattice trapdoors, of varying strength. For now, the reader can think of the quality as a measure of the norm of the vectors in $\mathbf{S}$, where smaller values are better.

tions [3, 5]. 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 [4] 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 [3, 5], 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 [7, 24, 20, 36] (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 [7]), and our preimage sampling algorithm for $f_{\mathbf{A}}$ works with smaller integers and requires much less offline storage than the one from [36].

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 $m > 5n \lg q$ [5] 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 [5] 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 [7, 20] that use high-precision real arithmetic. For the more efficient, parallel preimage sampling algorithm of [36] that uses small-integer arithmetic, the parameters guaranteed by [5] 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 [36].

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 [36] when using a basis generated according to [5].) 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 [13, 1] 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 [13] 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 [32, 37, 28, 30, 44, 31] for more details on these functions and their security foundations.

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. In the full version we describe new and improved applications, with a focus on signature schemes and chosen ciphertext-secure encryption.

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 [20], 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

	[3, 5] constructions	This work (fast $f_{\mathbf{A}}^{-1}$)	Impr. Factor
Dimension m	slow $f_{\mathbf{A}}^{-1}$ [24, 20]: $> 5n \lg q$ fast $f_{\mathbf{A}}^{-1}$ [36]: $> n \lg^2 q$	$\approx 2n \lg q$ $(\overset{s}{\approx})$ $n(1 + \lg q)$ $(\overset{c}{\approx})$	2.5 to $\lg q$
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 to $10\sqrt{\lg q}$
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 to $7 \lg q$

Fig. 1. Summary of parameters for our constructions versus prior ones. The symbols $\overset{s}{\approx}$ and $\overset{c}{\approx}$ denote constructions producing public keys $\mathbf{A}$ that are statistically and computationally close to uniform, respectively. All quality terms s and length bounds β omit the same "smoothing" factor for $\mathbb{Z}$, which is about 4–5 in practice.

	[5] with fast $f_{\mathbf{A}}^{-1}$	This work	Improvement Factor
Sec param n	436	284	1.53
Modulus logarithm $\log_2(q)$	32	24	1.33
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	**67.5**
Key size (ring-based)	$\approx 16 \times 10^6$	$\approx 361 \times 10^3$	$\approx$ **44.3**

Fig. 2. Representative parameters for GPV signatures (using fast inversion algorithms) estimated using the methodology from [34] with $\delta \leq 1.007$, which is estimated to require about 2^{46} core-years on a 64-bit 1.86GHz Xeon [15, 14]. We used $\omega_n = 4.5$ for $\mathbb{Z}$, which corresponds to statistical error $< 2^{-90}$ for each randomized-rounding operation during signing. Key sizes for *ring-based* GPV signatures are approximated to be smaller by a factor of about $0.9n$.

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 no LWE search/decision reduction for such q was known till recently, despite its obvious practical utility. The only such result we are aware of is the recent sample preserving reduction of [33], which applies to arbitrary q (including powers of 2), but requires the error distribution to be polynomially bounded. In the full version we provide a different and very general reduction (generalizing and extending [8, 41, 35, 6],) that also covers the $q = 2^k$ case and others, and is incomparable to [33], as it requires all prime factors of q to be polynomially bounded, but does not impose this restriction on the errors.

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 $\bar{m}$. (Concretely, $\bar{m} \approx n \lg q$ for the statistically secure construction, and $\bar{m} = 2n$ or n for computational security.) As shown in [13], 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

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

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 $\mathbf{R}\mathbf{z}_1, \mathbf{R}\mathbf{z}_2, \ldots, \mathbf{R}\mathbf{z}_n$ 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 in subcubic time. Batch operations can be exploited in applications like the multi-bit IBE of [20] and its extensions to HIBE [13, 1, 2].

Related techniques. At the surface, our trapdoor generator appears similar to the original "GGH" approach of [21] 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: in our case, 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 one. 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. Moreover, it is crucial in our setting that the transformation matrix $\mathbf{T}$ has small entries, while with GGH the transformation matrix can be arbitrary.

A more appropriate comparison is to Ajtai's original method [4] 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 $\mathbf{G}$. We remark that Ajtai's method to generate strong trapdoors [3] and follow-up work [5] are quite different and much more complex.

Our constructions and inversion algorithms also draw upon several other techniques from throughout the literature. The trapdoor basis generator of [5] and the LWE-based "lossy" injective trapdoor function of [40] 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 [40], and is identical to the method used in [1] for constructing compact (H)IBE. Finally, in our preimage sampling algorithm for $f_{\mathbf{A}}$, we use the "convolution" technique from [36] 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 $\mathbf{R}$.

Other Related Work. Concrete parameter settings for a variety "strong" trapdoor applications are given in [43]. Those parameters are derived using the previous suboptimal generator of [5], and using the methods from this work would yield substantial improvements. The recent work of [25] 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.). In a concurrent and independent work, Lyubashevsky [27] constructs a signature scheme in the random oracle model "without (strong) trapdoors," i.e., without relying on short bases or a gadget matrix $\mathbf{G}$. The form and sizes of his public and secret keys are very similar to ours, but the schemes and their security proofs work entirely differently.

2 Primitive Lattices

At the heart of our new trapdoor generation algorithm (described in Section 3) 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 $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ is primitive $\mathbf{G} \cdot \mathbb{Z}^w = \mathbb{Z}_q^n$.[5] The main results of this section are summarized in the following theorem.

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

- *The lattice $\Lambda^\perp(\mathbf{G})$ has a known basis $\mathbf{S} \in \mathbb{Z}^{w \times w}$ 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 $O(w)$ 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.*
- *Preimage sampling for $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\mathbf{x} \bmod q$ with Gaussian parameter $s \geq \|\widetilde{\mathbf{S}}\| \cdot \omega_n$ 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 w 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 $\|\mathbf{S}\| = \sqrt{b^2+1}$.

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. 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. In the full version, we prove that the vector $\mathbf{g} = (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$ for $k = \lceil \lg q \rceil$ admits a short basis for the lattice $\Lambda^\perp(\mathbf{g}^t)$, and we also describe specialized inversion and sampling algorithms that are both very simple and more efficient than generic solutions.

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 $\mathbf{S} := \mathbf{I}_n \otimes \mathbf{S}_k \in \mathbb{Z}^{nk \times nk}$. Equivalently, $\mathbf{G}$, $\Lambda^\perp(\mathbf{G})$, and $\mathbf{S}$ are the direct sums of n

[5] 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.

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. Theorem 1 follows by substituting appropriate primitive vectors $\mathbf{g}$ and bases $\mathbf{S}_k$ into the definitions of $\mathbf{G}$ and $\mathbf{S}$.

3 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 [20] 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 [3, 5] (and in [44], for the ring setting). 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 2. *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* $\mathsf{SampleD}$ *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* $\|\mathbf{e}\| < q/O(\sqrt{n \log q})$ *or* $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ *for* $1/\alpha \geq \sqrt{n \log q} \cdot \omega_n$, *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* $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}), s \cdot \omega_n}$.

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 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 2, and a new notion of trapdoor that we define next.

3.1 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 1. *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. 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 $\mathbf{R}$.

In the full version we show that a good basis for $\Lambda^{\perp}(\mathbf{A})$ may be obtained from knowledge of the trapdoor $\mathbf{R}$. This is not used anywhere in the rest of the paper, but it establishes 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 the new type of trapdoor.

We also make the following simple but useful observations: (1) The rows of $\left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right]$ in Definition 1 can appear in any order, since this just induces a permutation of $\mathbf{A}$'s columns. (2) 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. (3) If $\mathbf{R}$ is a trapdoor for $\mathbf{A}$ with tag $\mathbf{H}$, then $\mathbf{R}$ is also a trapdoor for $\mathbf{A}' = \mathbf{A} - [\mathbf{0} \mid \mathbf{H}'\mathbf{G}]$ 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 [1], and can be used to give a family of "tag-based" trapdoor functions [23]. In the full version we recall explicit families of matrices $\mathbf{H}$ having suitable properties for applications.

3.2 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 $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times (m-w)}$ 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 $\mathbf{T}^{-1} = \left[\begin{smallmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right]$, 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.

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 $\mathbf{H}$.
Choose a matrix $\mathbf{R} \in \mathbb{Z}^{\bar{m} \times w}$ from distribution $\mathcal{D}$.
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 type 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). Some applications allow for an optimization that additionally 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 impliesthat $\mathbf{R} \leftarrow \mathcal{D}$ 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 $\delta = \operatorname{negl}(n)$. 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., [34, 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 $\delta \le \frac{w}{2}\sqrt{q^n/2^{\bar{m}}}$, by a version of the leftover hash lemma (see, e.g., [5, Section 2.2.1]). Therefore, we can use any $\bar{m} \ge n \lg q + 2 \lg \frac{w}{2\delta}$. Other statistical instantiations are presented in the full version.

Computational instantiation. Let $\bar{\mathbf{A}} = [\mathbf{I}_n \mid \hat{\mathbf{A}}] \in \mathbb{Z}_q^{n\times\bar{m}}$ 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. In applications that use only a single tag $\mathbf{H} = \mathbf{I}$ (e.g., GPV signatures [20]), 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, using an appropriate regularity lemma (e.g., the ones from [44] or [31]) for a statistical instantiation, and the ring-LWE problem for a computationally secure instantiation.

3.3 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.

Theorem 3. *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 $1/\alpha \ge 2\|\mathbf{B}\|s\cdot\omega_n$, the algorithm inverts successfully with overwhelming probability over the choice of $\mathbf{e}$.*

Note that using our constructions from Section 2, 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).

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

Proof. Let $\bar{\mathbf{R}} = [\mathbf{R}^t \mid \mathbf{I}]$, 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 $\|\mathbf{b}_i^t\bar{\mathbf{R}}\| \leq s\|\mathbf{B}\|$. 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 with parameter αq, the probability that $|(\mathbf{b}_i^t\bar{\mathbf{R}})\mathbf{e}/q| \geq 1/2$ is negligible, and the second claim follows by the union bound.

3.4 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 the full version, 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 $\sqrt{\Sigma_{\mathbf{G}}} \geq \eta_{\epsilon}(\Lambda^{\perp}(\mathbf{G}))$. For example, Section 2 describes algorithms for which $\sqrt{\Sigma_{\mathbf{G}}}$ is either 2 or $\sqrt{5}$. (Throughout this summary we omit the small smoothing factor ω_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 $\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]\Sigma_{\mathbf{G}}[\mathbf{R}^t \mid \mathbf{I}]$). 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 $s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_{\mathbf{G}}}) \leq s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right]) \cdot s_1(\sqrt{\Sigma_{\mathbf{G}}}) \leq \sqrt{s_1(\mathbf{R})^2 + 1} \cdot s_1(\sqrt{\Sigma_{\mathbf{G}}})$, 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 $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$, 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 $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$, we use the 'convolution' technique from [36] 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 $s^2 - \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \Sigma_{\mathbf{G}} \left[\mathbf{R}^t \mid \mathbf{I}\right]$, which is well-defined as long as $s \geq s_1(\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] \cdot \sqrt{\Sigma_{\mathbf{G}}})$. 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 $\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A})$, 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. Our algorithm can sample from a discrete Gaussian with parameter $s \cdot \omega_n$ 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 $\sqrt{6/4} \leq 1.23$ — than the bound $(s_1(\mathbf{R}) + 1) \cdot \|\widetilde{\mathbf{S}}\|$ on the largest Gram-Schmidt norm of a lattice basis derived from the trapdoor $\mathbf{R}$. (Recall that our constructions from Section 2 give $s_1(\Sigma_{\mathbf{G}}) = \|\widetilde{\mathbf{S}}\|^2 = 4$ or 5.) In the iterative "randomized nearest-plane" sampling algorithm of [24, 20], the Gaussian parameter s is bounded from below by the largest Gram-Schmidt norm of the orthogonalized input basis (times the same ω_n factor used in our algorithm). Therefore, the efficiency and parallelism of our algorithm 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 the sampling algorithm, 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 $D_{\mathbb{Z}^m, \sqrt{\Sigma} \cdot \omega_n}$ for some fixed (typically non-spherical) covariance matrix $\Sigma > \mathbf{I}$. By [36, Theorem 3.1], this can be accomplished (up to any desired statistical distance) simply by sampling a continuous Gaussian $D_{\sqrt{\Sigma - \mathbf{I}} \cdot \omega_n}$ with sufficient precision, then independently randomized-rounding each entry of the sampled vector to $\mathbb{Z}$ using Gaussian parameter $\omega_n \geq \eta_\epsilon(\mathbb{Z})$.

Naively, the online work is dominated by the computation of $\mathbf{H}^{-1}(\mathbf{u} - \bar{\mathbf{w}})$ and $\mathbf{R}\mathbf{z}$ (plus the call to $\mathcal{O}(\mathbf{v})$, which as described in Section 2 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 $\mathbf{R}\mathbf{z}$ by pre-multiplying each

Algorithm 3 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 $\mathbf{A}\mathbf{R}' = \mathbf{H}'\mathbf{G} - \mathbf{A}_1$.
```

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.

3.5 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 [13]: 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 $\mathsf{ToBasis}$ 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 [13] 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.

Usually, the oracle $\mathcal{O}$ needed by Algorithm 3 would be implemented (up to negl(n) statistical distance) by our Gaussian sampling algorithm 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 the fact that the columns of $\mathbf{R}'$ are independent and negl(n)-subgaussian.

Lemma 1. *For any valid inputs $\mathbf{A}'$ and $\mathbf{H}'$, Algorithm 3 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.*

References

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

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

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

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

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[27] V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, 2012.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[44] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635, 2009.

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