Bonsai Trees, or How to Delegate a Lattice Basis

David Cash* Dennis Hofheinz† Eike Kiltz‡ Chris Peikert§

March 19, 2010

Abstract

We introduce a new *lattice-based* cryptographic structure called a *bonsai tree*, and use it to resolve some important open problems in the area. Applications of bonsai trees include:

- An efficient, stateless 'hash-and-sign' signature scheme in the *standard model* (i.e., no random oracles), and
- The first *hierarchical* identity-based encryption (HIBE) scheme (also in the standard model) that does not rely on bilinear pairings.

Interestingly, the abstract properties of bonsai trees seem to have no known realization in conventional number-theoretic cryptography.

1 Introduction

Lattice-based cryptographic schemes have undergone rapid development in recent years, and are attractive due to their low asymptotic complexity and potential resistance to quantum-computing attacks. One notable recent work in this area is due to Gentry, Peikert, and Vaikuntanathan [25], who constructed an efficient 'hash-and-sign' signature scheme and an identity-based encryption (IBE) scheme. (IBE is a powerful cryptographic primitive in which *any string* can serve as a public key [53].)

Abstractly, the GPV schemes are structurally quite similar to Rabin/Rabin-Williams signatures [50] (based on integer factorization) and the Cocks/Boneh-Gentry-Hamburg IBEs [18, 13] (based on the quadratic residuosity problem), in that they all employ a so-called "preimage sampleable" trapdoor function as a basic primitive. As a result, they have so far required the random oracle model (or similar heuristics) for their security analysis. This is both a theoretical drawback and also a practical concern (see, e.g., [35]), so avoiding such heuristics is an important goal.

Another intriguing open question is whether any of these IBE schemes can be extended to deliver richer levels of functionality, as has been done in pairing-based cryptography since the work of Boneh

*University of California, San Diego. Email: cdc@ucsd.edu. Part of work performed while at Georgia Institute of Technology.

†Karlsruhe Institute of Technology. Email: Dennis.Hofheinz@kit.edu. Part of work performed while at CWI and supported by an NWO Veni grant.

‡Cryptology & Information Security Group, CWI, Amsterdam, The Netherlands. kiltz@cwi.nl. Supported by the research program Sentinels

§Georgia Institute of Technology. Email: cpeikert@cc.gatech.edu. This material is based upon work supported by the National Science Foundation under Grants CNS-0716786 and CNS-0749931, and by the US Department of Homeland Security under Contract Number HSHQDC-07-C-00006. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the US Department of Homeland Security.

and Franklin [12]. For example, the more general notion of *hierarchical* IBE [33, 26] permits multiple levels of secret-key authorities. This notion is more appropriate than standard IBE for large organizations, can isolate damage in the case of secret-key exposure, and has further applications such as forward-secure encryption [16] and broadcast encryption [21, 58].

1.1 Our Results

We put forward a new cryptographic notion called a *bonsai tree*, and give a realization based on hard lattice problems. (Section 1.2 gives an intuitive overview of bonsai trees, and Section 1.4 discusses their relation to other primitives and techniques.) We then show that bonsai trees resolve some central open questions in lattice-based cryptography: to summarize, they remove the need for random oracles in many important applications, and facilitate delegation for purposes such as hierarchical IBE.

Our first application of bonsai trees is an efficient, stateless signature scheme that is secure in the *standard model* (no random oracles) under conventional lattice assumptions. Our scheme has a 'hash-and-sign' flavor that does not use the key-refresh/authentication-tree paradigm of many prior constructions (both generic [28, 43] and specialized to lattice assumptions [37]), and in particular it does not require the signer to keep any state. (Statelessness is a crucial property in many real-world scenarios, where distinct systems may sign relative to the same public key.) In our scheme, the verification key, signature length, and verification time are all an $O(k)$ factor larger than in the random-oracle scheme of [25], where k is the output length of a *chameleon* hash function, and the $O(\cdot)$ notation hides only a 1 or 2 factor. The signing algorithm is essentially as efficient as the one from [25].[1] The underlying hard problem is the standard *short integer solution* (SIS) problem dating back to the seminal work of Ajtai [5], which is known to be as hard as several worst-case approximation problems on lattices (see also [41, 25]). Via SIS, the security of our signature scheme rests upon the hardness of approximating worst-case problems on n-dimensional lattices to within an $\tilde{O}(\sqrt{k} \cdot n^{3/2})$ factor; this is only a $\sqrt{k}$ factor looser than that of [25].

Our second application is a collection of various *hierarchical* identity-based encryption (HIBE) schemes, which are the first HIBEs that do not rely on bilinear pairings. Our main scheme works in the standard model, also making it the first non-pairing-based IBE (hierarchical or not) that does not use random oracles or qualitatively similar heuristics. The underlying hard problem is the standard *learning with errors* (LWE) problem as defined by Regev, which may be seen as the 'dual' of SIS and is also as hard as certain worst-case lattice problems [51, 45]; LWE is also the foundation for the plain IBE of [25], among many other recent cryptographic schemes.

Additionally, our HIBE is *anonymous* across all levels of the hierarchy, i.e., a ciphertext conceals (computationally) the identity to which is was encrypted. Anonymity is a useful property in many applications, such as fully private communication [7] and searching on encrypted data [11, 1]. While there are a few anonymous (non-hierarchical) IBEs [12, 20, 13, 25], only one other HIBE is known to be anonymous [15].

1.2 Overview of Bonsai Trees and Applications

The ancient art of bonsai is centered around a *tree* and the selective *control* thereof by an arborist, the tree's cultivator and caretaker. By combining natural, *undirected* growth with *controlled* propagation techniques such as wiring and pruning, arborists cultivate trees according to a variety of aesthetic forms.

Similarly, cryptographic bonsai is not so much a precise definition as a collection of principles and techniques, which can be employed in a variety of ways. (The informal description here is developed

[1] Our signing algorithm performs about k *forward* computations of a trapdoor function, plus one inversion (which dominates the running time).

technically in Section 3.) The first principle is the tree itself, which in our setting is a *hierarchy of trapdoor functions* having certain properties. The arborist can be any of several entities in the system — e.g., the signer in a signature scheme or a simulator in a security proof — and it can exploit both kinds of growth, undirected and controlled. Briefly stated, *undirected* growth of a branch means that the arborist has no privileged information about the associated function, whereas the arborist *controls* a branch if it knows a trapdoor for the function. Moreover, control automatically extends down the hierarchy, i.e., knowing a trapdoor for a parent function implies knowing a trapdoor for any of its children.

In our concrete lattice-based instantiation, the functions in the tree are indexed by a hierarchy of public lattices chosen at random from a certain 'hard' family (i.e., one having a connection to worst-case problems). The lattices may be specified by a variety of means, e.g., a public key, interaction via a protocol, a random oracle, etc. Their key property is that they naturally form a hierarchy as follows: every lattice in the tree (excepting the root) is a *higher-dimensional superlattice* of its parent. Specifically, a parent lattice in $\mathbb{R}^m$ is simply the restriction of its child(ren) in $\mathbb{R}^{m'}$ (where $m' > m$) to the first m dimensions. As we shall see shortly, this hierarchical relationship means that a parent lattice naturally 'subsumes' its children (and more generally, all its descendants).

Undirected growth in our realization is technically straightforward, emerging naturally from the underlying hard average-case lattice problems (SIS and LWE). This growth is useful primarily for letting a simulator embed a challenge problem into one or more branches of the tree (but it may have other uses as well).

To explain *controlled growth*, we first need a small amount of technical background. As explored in prior works on lattice-based cryptography (e.g., [27, 30, 29, 25, 49, 45]), a lattice has a 'master trapdoor' in the form of a *short basis*, i.e., a basis made up of relatively short lattice vectors. Knowledge of such a trapdoor makes it easy to solve a host of seemingly hard problems relative to the lattice, such as decoding within a bounded distance, or randomly sampling short lattice vectors. The reader may view a short basis for a lattice as roughly analogous to the factorization of an integer, though we emphasize that there are in general *many distinct* short bases that convey roughly 'equal power' with respect to the lattice.

In light of the above, we say that an arborist *controls* a branch of a bonsai tree if it knows a short basis for the associated lattice. The hierarchy of lattices is specially designed so that any short basis of a parent lattice can be easily *extended* to a short basis of any higher-dimensional child lattice, with no loss in quality. This means that control of a branch implicitly comes with control over all its offshoots. In a typical application, the privileged entity in the system (e.g., the signer in a signature scheme) will know a short basis for the root lattice, thus giving it control over the entire tree. Other entities, such as an attacker, will generally have less power, though in some applications they might even be given control over entire subtrees.

So far, we have deliberately avoided the question of *how* an arborist comes to control a (sub)tree by acquiring a short basis for the associated lattice. A similar issue arises in other recent cryptographic schemes [25, 49, 45], but in a simpler setting involving only a single lattice and short basis (not a hierarchy). In these schemes, one directly applies a special algorithm, originally conceived by Ajtai [4] and recently improved by Alwen and Peikert [6], which generates a hard random lattice together with a short basis 'from scratch.' At first glance, the algorithms of [4, 6] seem useful only for controlling a new tree entirely by its root, which is not helpful if we need finer-grained control. Fortunately, we observe that the same technique used for extending an already-controlled lattice also allows us to 'graft' a solitary controlled lattice onto an *uncontrolled* branch.[2]

[2]It is worth noting that in [4, 6], even the simple goal of generating a solitary lattice together with a short basis actually proceeds in two steps: first start with a sufficient amount of random undirected growth, then produce a single controlled offshoot by way of a certain linear algebraic technique. Fittingly, this is analogous to the traditional bonsai practice of growing a new specimen from a cutting of an existing tree, which is generally preferred to growing a new plant 'from scratch' with seeds.

This whole collection of techniques, therefore, allows an arborist to achieve a primary bonsai aesthetic: a carefully controlled tree that nonetheless gives the appearance of having grown without any outside intervention. As we shall see next, bonsai techniques can reduce the construction of complex cryptographic schemes to the design of simple *combinatorial* games between an arborist and an adversary.

1.2.1 Application 1: Hash-and-Sign without Random Oracles

Our end goal is a signature scheme that meets the *de facto* notion of security, namely, existential unforgeability under adaptive chosen-message attack [28]. By a standard, efficient transformation using *chameleon hashes* [34] (which have efficient realizations under conventional lattice assumptions, as we show), it suffices to construct a *weakly secure* scheme, namely, one that is existentially unforgeable under a static attack in which the adversary non-adaptively makes all its queries before seeing the public key.

Our weakly secure scheme signs messages of length k, the output length of the chameleon hash. The public key represents a *binary* bonsai tree T of depth k in a compact way, which we describe in a moment. The secret key is a short basis for the lattice Λ_ε at the root of the tree, which gives the signer control over all of T. To sign a string $\mu \in \{0,1\}^k$ (which is the chameleon hash of the 'true' message m), the signer first derives the lattice Λ_μ from T by walking the root-to-leaf path specified by μ. The signature is simply a short nonzero vector $\mathbf{v} \in \Lambda_\mu$, chosen at random from the 'canonical' Gaussian distribution (which can be sampled efficiently using the signer's control over Λ_μ). A verifier can check the signature $\mathbf{v}$ simply by deriving Λ_μ itself from the public key, and checking that $\mathbf{v}$ is a sufficiently short nonzero vector in Λ_μ.

The bonsai tree T is represented compactly by the public key in the following way. First, the root lattice Λ_ε is specified completely. Then, for each level $i = 0, \ldots, k-1$, the public key includes two blocks of randomness that specify how a parent lattice at level i branches into its two child lattices. We emphasize that all nodes at a given depth use the *same* two blocks of randomness to derive their children.

The proof of security is at heart a combinatorial game on the tree between the simulator $\mathcal{S}$ and forger $\mathcal{F}$, which goes roughly as follows. The forger gives the simulator a set $M = \{\mu_1, \ldots, \mu_Q\}$ of messages, and $\mathcal{S}$ needs to cultivate a bonsai tree (represented by pk) so that it controls some set of subtrees that *cover* all of M, yet is unlikely to control the leaf of whatever arbitrary message $\mu^* \notin M$ that $\mathcal{F}$ eventually produces as a forgery. If the latter condition happens to hold true, then the forger has found a short nonzero vector in an uncontrolled lattice, in violation of the underlying assumption.

To satisfy these conflicting constraints, $\mathcal{S}$ colors red all the edges on the root-to-leaf paths of the messages in M, and lets all the other edges implicitly be colored blue. The result is a forest of at most $Q \cdot k$ distinct blue subtrees $\{B_\ell\}$, each growing off of some red path by a single blue edge. The simulator chooses one of these subtrees B_ℓ uniformly at random (without regard to its size), guessing that the eventual forgery will lie in B_ℓ. It then cultivates a bonsai tree so that all the growth on the path up to and throughout B_ℓ is *undirected* (by embedding its given challenge instance as usual), while all the remaining growth in $T \setminus B_\ell$ is controlled. This goal can be achieved within the confines of the public key by controlling one branch at each level leading up to B_ℓ (namely, the branch growing off of the path to B_ℓ), and none thereafter.

1.2.2 Application 2: Hierarchical Identity-Based Encryption

Bonsai trees also provide a very natural and flexible approach for realizing HIBE. For simplicity, consider an authority hierarchy that is a *binary tree*, which suffices for forward-secure encryption and general HIBE itself [16]. The master public key of the scheme describes a binary bonsai tree, which mirrors the authority hierarchy. The root authority starts out by controlling the entire tree, i.e., it knows a trapdoor short basis for the lattice at the root. Each authority is entitled to control its corresponding branch of the tree. Any entity

in the hierarchy can delegate control over an offshoot branch to the corresponding sub-authority, simply by computing and revealing a short basis of the associated child lattice. In this framework, encryption and decryption algorithms based on the LWE problem are standard.

For the security proof, the simulator again prepares a bonsai tree so that it controls certain branches (which should cover the adversary's queries), while allowing the undirected growth of others (corresponding to the adversary's target identity). This can be accomplished in a few ways, with different advantages and drawbacks in terms of the security notion achieved and the tightness of the reduction. One notion is security against a *selective-identity* attack, where the adversary must declare its target identity before seeing the public key, but may adaptively query secret keys afterward. In this model, the simulator can cultivate a bonsai tree whose growth toward the (known) target identity is undirected, while controlling each branch off of that path; this setup makes it easy for the simulator to answer any legal secret-key query.

A stronger notion is a *fully adaptive* attack, where the adversary may choose its target identity after making its secret-key queries. There are generic combinatorial techniques for converting selective-identity-secure (H)IBE schemes into fully secure ones; we show how to apply and optimize these techniques to our HIBE. First, we use the techniques of Boneh and Boyen [8] construct a fully secure HIBE scheme in the random oracle model. The basic idea is to hash all identities; this way, the target identity can be dynamically embedded as the answer to a random oracle query. Secondly, we demonstrate that other tools of Boneh and Boyen [9] can be adapted to our setting to yield a fully secure HIBE scheme *without* random oracles. This works by hashing identities to branches of a bonsai tree, where a probabilistic argument guarantees that any given identity hashes to a controlled branch with a certain probability. We can adjust this probability in the right way, so that with non-negligible probability, all queried identities hash to controlled branches, while the target identity hashes to an uncontrolled branch. In our probabilistic argument, we employ *admissible hash functions (AHFs)*, as introduced by [9]. However, as we will explain in Section 5.4.1, their original AHF definition and proof strategy do not take into consideration the statistical dependence of certain crucial events. We circumvent this with a different AHF definition and a different proof.

Based on the above description, the reader may still wonder whether secret-key delegation is actually secure, i.e., whether the real and simulated bases are drawn from the same probability distribution. In fact, they may not be! For example, under the most straightforward method of extending a basis, the child basis actually contains the parent basis as a submatrix, so it is clearly insecure to reveal the child. We address this issue with an additional bonsai principle of *randomizing control*, using the 'oblivious' Gaussian sampling algorithm of [25]. This produces a new basis under a 'canonical' distribution, regardless of the original input basis, which ensures that the real system and simulation coincide. The randomization increases the length of the basis by a small factor — which accumulates geometrically with each delegation from parent to child — but for reasonable depths, the resulting bases are still short enough to be useful when all the parameters are set appropriately. (See Section 1.3 for more details.)

For achieving security under chosen-ciphertext attacks (CCA security), a transformation due to Boneh, Canetti, Halevi, and Katz [10] gives a CCA-secure HIBE for depth d from any chosen plaintext-secure HIBE for depth $d + 1$. Alternatively, we observe that the public and secret keys in our HIBE scheme are of exactly the same 'type' as those in the recent CCA-secure cryptosystem of [45], so we can simply plug that scheme into our bonsai tree/HIBE framework. Interestingly, the two approaches result in essentially identical schemes.

1.2.3 Variations

This paper focuses almost entirely on bonsai trees that are related, via worst- to average-case reductions, to *general* lattices. Probably the main drawback is that the resulting public and secret keys are rather large. For

example, the public key in our signature scheme is larger by a factor of k (the output length of a chameleon hash function) than that of its random-oracle analogue [25], which is already at least quadratic in the security parameter. Fortunately, the principles of bonsai trees may be applied equally well using analogous hard problems and tools for *cyclic/ideal lattices* (developed in, e.g., [39, 47, 36, 48, 55, 38]). This approach can 'miniaturize' the bonsai trees and most of their associated operations by about a linear factor in the security parameter. The resulting schemes are still not suitable for practice, but their asymptotic behavior is attractive.

1.3 Complexity and Open Problems

Here we discuss some additional quantitative details of our schemes, and describe some areas for further research.

Several important quantities in our bonsai tree constructions and applications depend upon the depth of the tree. The dimension of a lattice in the tree grows linearly with its depth, and the size of the trapdoor basis grows roughly quadratically with the dimension.

Accordingly, in our HIBE schemes, the dimension of a ciphertext vector grows (at least) linearly with the depth of the identity to which it is encrypted. Moreover, the (Euclidean) length of an user's trapdoor basis increases *geometrically* with its depth in the tree (more precisely, with the length of the delegation chain), due to the basis randomization that is performed with each delegation. To ensure correct decryption, the inverse noise parameter $1/\alpha$ in the associated LWE problem, and hence the approximation factor of the underlying worst-case lattice problems, must grow with the basis length. In particular, a hierarchy of depth d corresponds (roughly) to an $n^{d/2}$ approximation factor for worst-case lattice problems, where n is the dimension. Because lattice problems are conjectured to be hard to approximate to within even subexponential factors, the scheme may remain secure for depths as large as $d = n^c$, where $c < 1$.

Our HIBE scheme that enjoys security under a full *adaptive-identity* attack requires large keys and has a somewhat loose security reduction. In particular, the attack simulation partitions an (implicit) bonsai tree into controlled and undirected branches. This is done in the hope that all user secret key queries refer to controlled branches (so the simulation can derive the corresponding secret key), and that the target identity refers to an undirected branch (so the attack can be converted into one on the LWE problem). This simulation approach (dubbed 'partitioning strategy' in [57]) involves, to a certain extent, guessing the adversary's user secret key and challenge queries. The result is a rather loose security reduction.

In contrast, recent works have achieved tight reductions (and even small keys, in some cases) for pairing-based (H)IBEs under various assumptions [23, 24, 57], and a variant of the GPV IBE (in the random oracle model) also has a tight reduction, but their approaches do not seem to translate to our setting. The issue, essentially, is that our simulator is required to produce a 'master trapdoor' for each queried identity, which makes it difficult to embed the challenge problem into the adversary's view. In prior systems with tight reductions, secret keys are less 'powerful,' so the simulator can embed a challenge while still producing secret keys for any identity (even the targeted one).

A final very interesting (and challenging) question is whether bonsai trees can be instantiated based on other mathematical foundations, e.g., integer factorization. At a very fundamental level, our lattice-based construction seems to rely upon a kind of random self-reducibility that the factorization problem is not known to enjoy.

1.4 Related Techniques and Works

This paper represents a combination of two concurrent and independent works by the first three authors [17] and the fourth author [44], which contained some overlapping results and were accepted to Eurocrypt 2010

under the condition that they be merged.

The abstract properties of bonsai trees appear to have no known realization in conventional number-theoretic cryptography. However, our applications use combinatorial techniques that are similar to those from prior works.

The analysis of our signature scheme is reminiscent of (and influenced by) the recent RSA-based signatures of Hohenberger and Waters [32], but there are some notable structural differences. Most significantly, our scheme does not implicitly 'sign' every prefix of the message as in [32]. Additionally, in contrast with prior hash-and-sign schemes based on RSA [22, 19, 31, 32], our simulator cannot use an 'accumulator' to produce signatures for *exactly* the queried messages, but instead sets up the public key so that it knows enough trapdoors to *cover* all the messages (and potentially many others). This requires the simulator to cultivate a tree whose structure crucially depends on the *global* properties of the entire query set, thus inducing the forest of subtrees as described in Section 1.2.1.

The structure of our HIBE is also similar, at a combinatorial level at least, to that of prior pairing-based HIBEs, in that the simulator can 'control' certain edges of an (implicit) tree by choosing certain random exponents itself. However, there are no *trapdoor functions* per se in pairing-based constructions; instead, the pairing is used to facilitate secret agreement between the encrypter and decrypter. Our approach, therefore, may be seen as a blending of pairing-based techniques and the trapdoor techniques found in [18, 13, 25].

Following the initial dissemination of our results in [17, 44], several extensions and additional applications have been found. Rückert [52] modified our signature scheme to make it *strongly* unforgeable, and constructed hierarchical identity-based signatures. Agrawal and Boyen [3] constructed a standard-model IBE based on LWE, which is secure under a selective-identity attack; their construction has structure similar to ours, but it does not address delegation, nor does it give an efficient signature scheme. Agrawal, Boneh, and Boyen [2] improved the efficiency of our (H)IBE schemes (under a somewhat stronger LWE assumption), and Boyen [14] used similar techniques to obtain shorter signatures (under a stronger SIS assumption).

2 Preliminaries

2.1 Notation

For a positive integer k, $[k]$ denotes the set $\{1, \ldots, k\}$; $[0]$ is the empty set. We denote the set of integers modulo an integer $q \geq 1$ by $\mathbb{Z}_q$. For a string x over some alphabet, $|x|$ denotes the length of x. We say that a function in n is *negligible*, written $\mathrm{negl}(n)$, if it vanishes faster than the inverse of any polynomial in n.

The *statistical distance* between two distributions $\mathcal{X}$ and $\mathcal{Y}$ (or two random variables having those distributions), viewed as functions over a countable domain D, is defined as $\max_{A \subseteq D} |\mathcal{X}(A) - \mathcal{Y}(A)|$.

Column vectors are named by lower-case bold letters (e.g., $\mathbf{x}$) and matrices by upper-case bold letters (e.g., $\mathbf{X}$). We identify a matrix $\mathbf{X}$ with the ordered set $\{\mathbf{x}_j\}$ of its column vectors, and let $\mathbf{X} \| \mathbf{X}'$ denote the (ordered) concatenation of the sets $\mathbf{X}, \mathbf{X}'$. For a set $\mathbf{X}$ of real vectors, we define $\|\mathbf{X}\| = \max_j \|\mathbf{x}_j\|$, where $\|\cdot\|$ denotes the Euclidean norm.

For any (ordered) set $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\} \subset \mathbb{R}^m$ of linearly independent vectors, let $\widetilde{\mathbf{S}} = \{\widetilde{\mathbf{s}_1}, \ldots, \widetilde{\mathbf{s}_k}\}$ denote its *Gram-Schmidt orthogonalization*, defined iteratively as follows: $\widetilde{\mathbf{s}_1} = \mathbf{s}_1$, and for each $i = 2, \ldots, k$, the vector $\widetilde{\mathbf{s}_i}$ is the component of $\mathbf{s}_i$ orthogonal to $\mathrm{span}(\mathbf{s}_1, \ldots, \mathbf{s}_{i-1})$. In matrix notation, there is a unique QR decomposition $\mathbf{S} = \mathbf{QR}$ where the columns of $\mathbf{Q} \in \mathbb{R}^{m \times k}$ are orthonormal (i.e., $\mathbf{Q}^t\mathbf{Q} = \mathbf{I} \in \mathbb{R}^{k \times k}$) and $\mathbf{R} \in \mathbb{R}^{k \times k}$ is right-triangular with positive diagonal entries; the Gram-Schmidt orthogonalization is $\widetilde{\mathbf{S}} = \mathbf{Q} \cdot \mathrm{diag}(r_{1,1}, \ldots, r_{k,k})$. Clearly, $\|\widetilde{\mathbf{s}_i}\| \leq \|\mathbf{s}_i\|$ for all i.

2.2 Cryptographic Definitions

The main cryptographic security parameter through the paper is n, and all algorithms (including the adversary) are implicitly given the security parameter n in unary.

For a (possibly interactive) algorithm $\mathcal{A}$, we define its *distinguishing advantage* between two distributions $\mathcal{X}$ and $\mathcal{Y}$ to be $|\Pr[\mathcal{A}(\mathcal{X}) = 1] - \Pr[\mathcal{A}(\mathcal{Y}) = 1]|$. We use the general notation $\mathbf{Adv}^{\text{atk}}_{\text{SCH}}(\mathcal{A})$ to describe the advantage of an adversary $\mathcal{A}$ mounting an atk attack on a cryptographic scheme SCH, where the definition of advantage is specified as part of the attack. Similarly, we write $\mathbf{Adv}_{\text{PROB}}(\mathcal{A})$ for the advantage of an adversary $\mathcal{A}$ against a computational problem PROB (where again the meaning of advantage is part of the problem definition).

Chameleon hash functions. Chameleon hashing was introduced by Krawczyk and Rabin [34]. For our purposes, we need a slight generalization in the spirit of "preimage sampleable" (trapdoor) functions [25].

A family of chameleon hash functions is a collection $\mathcal{H} = \{h_i : \mathcal{M} \times \mathcal{R} \to \mathcal{Y}\}$ of functions h_i mapping a message $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$ to a range $\mathcal{Y}$. The randomness space $\mathcal{R}$ is endowed with some efficiently sampleable distribution (which may not be uniform). A function h_i is efficiently computable given its description, and the family has the property that for any $m \in \mathcal{M}$, for $h_i \leftarrow \mathcal{H}$ and $r \leftarrow \mathcal{R}$, the pair $(h_i, h_i(m, r))$ is uniform over $(\mathcal{H}, \mathcal{Y})$ (up to negligible statistical distance). The chameleon property is that a random $h_i \leftarrow \mathcal{H}$ may be generated together with a trapdoor t, such that for any output $y \in \mathcal{Y}$ and message $m \in \mathcal{M}$, it is possible (using t) to efficiently sample $r \in \mathcal{R}$ (under the $\mathcal{R}$'s distribution) conditioned on the requirement that $h_i(m, r) = y$. Finally, the family has the standard collision-resistance property, i.e., given $h_i \leftarrow \mathcal{H}$ it should be hard for an adversary to find distinct $(m, r), (m', r') \in \mathcal{M} \times \mathcal{R}$ such that $h_i(m, r) = h_i(m', r')$.

A realization under conventional lattice assumptions of chameleon hash functions (in the above sense) for $\mathcal{M} = \{0, 1\}^\ell$ is straightforward, using the particular preimage sampleable functions (PSFs) from [25]. Briefly, the chameleon hash function is simply a PSF applied to $m \| r$, which may also be viewed as the sum of two independent PSFs applied to m and r, respectively. We omit the details.

Signatures. A signature scheme SIG for a message space $\mathcal{M}$ is a tuple of PPT algorithms as follows:

- Gen 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 $(vk, sk) \leftarrow \mathsf{Gen}$ 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. The first, existential unforgeability under *static* chosen-message attack, or eu-scma security, is defined as follows: first, the forger $\mathcal{F}$ outputs a list of query messages $\mu_1, \ldots, \mu_Q$ for some Q. Next, $(vk, sk) \leftarrow \mathsf{Gen}$ and $\sigma_i \leftarrow \mathsf{Sign}(sk, \mu_i)$ are generated for each $i \in [Q]$, then vk and σ_i (for each $i \in [Q]$) are given to $\mathcal{F}$. Finally, $\mathcal{F}$ outputs an attempted forgery (μ^*, σ^*). The advantage $\mathcal{A}^{\text{eu-scma}}_{\text{SIG}}(\mathcal{F})$ of $\mathcal{F}$ is the probability that $\mathsf{Ver}(vk, \mu^*, \sigma^*)$ accepts and $\mu^* \neq \mu_i$ for all $i \in [Q]$, taken over all the randomness of the experiment.

Another notion, called existential unforgeability under *adaptive* chosen-message attack, or eu-acma security, is defined similarly, except that $\mathcal{F}$ is first given vk and may adaptively choose the messages μ_i.

Using a family of chameleon hash functions (as defined above), there is a generic construction of eu-acma-secure signatures from eu-scma-secure signatures; see, e.g., [34]. Furthermore, the construction results in an *online/offline* signature scheme; see [54]. The basic idea behind the construction is that the signer chameleon hashes the message to be signed, then signs the hashed message using the eu-scma-secure scheme (and includes the randomness used in the chameleon hash with the final signature).

Key-Encapsulation Mechanism (KEM). We present all of our encryption schemes in the framework of *key encapsulation*, which simplifies the definitions and leads to more modular constructions. A KEM for keys of length $\ell = \ell(n)$ is a triple of PPT algorithms as follows:

- $\mathsf{KEM.Gen}$ outputs a public key pk and a secret key sk.
- $\mathsf{KEM.Encaps}(pk)$ outputs a key $\kappa \in \{0,1\}^\ell$ and its encapsulation as $\sigma \in \{0,1\}^*$.
- $\mathsf{KEM.Decaps}(sk, \sigma)$ outputs a key κ.

The correctness requirement is: for $(pk, sk) \leftarrow \mathsf{KEM.Gen}$ and $(\kappa, \sigma) \leftarrow \mathsf{KEM.Encaps}(pk)$, $\mathsf{KEM.Decaps}(sk, \sigma)$ should output κ with all but $\mathrm{negl}(n)$ probability.

In this work we are mainly concerned with indistinguishability under chosen-plaintext attack, or ind-cpa security. The attack is defined as follows: generate $(pk, sk) \leftarrow \mathsf{KEM.Gen}$, $(\kappa^*, \sigma^*) \leftarrow \mathsf{KEM.Encaps}(pk)$, and $\kappa' \leftarrow \{0,1\}^\ell$ (chosen uniformly and independently of the other values). The advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\text{KEM}}(\mathcal{A})$ of an adversary $\mathcal{A}$ is its distinguishing advantage between (pk, σ^*, κ^*) and (pk, σ^*, κ').

Hierarchical Identity-Based Encryption (HIBE) and Binary Tree Encryption (BTE). In HIBE, identities are strings over some alphabet $\mathcal{ID}$; BTE is the special case of HIBE with identity space $\mathcal{ID} = \{0,1\}$. A HIBE is a tuple of PPT algorithms as follows:

- $\mathsf{Setup}(1^d)$ outputs a master public key mpk and root-level user secret key usk_ε. (In the following, 1^d and mpk are implicit parameters to every algorithm, and every usk_{id} is assumed to include id itself.)
- $\mathsf{Extract}(usk_{id}, id')$, given an user secret key for identity $id \in \mathcal{ID}^{<d}$ that is a prefix of $id' \in \mathcal{ID}^{\leq d}$, outputs a user secret key $usk_{id'}$ for identity id'.
- $\mathsf{Encaps}(id)$ outputs a key $\kappa \in \{0,1\}^\ell$ and its encapsulation as $\sigma \in \{0,1\}^*$, to identity id.
- $\mathsf{Decaps}(usk_{id}, \sigma)$ outputs a key κ.

The correctness requirement is: for any identity $id \in \mathcal{ID}^{\leq d}$, generate $(mpk, usk_\varepsilon) \leftarrow \mathsf{Setup}(1^d)$, usk_{id} via any legal sequence of calls to $\mathsf{Extract}$ starting from usk_ε, and $(\kappa, \sigma) \leftarrow \mathsf{Encaps}(id)$. Then $\mathsf{Decaps}(usk_{id}, \sigma)$ should output κ with all but $\mathrm{negl}(n)$ probability (over all the randomness in the experiment).

There are several attack notions for HIBE. We are mainly concerned with the simple notion of indistinguishability under a chosen-plaintext, *selective-identity* attack, or sid-ind-cpa security. The attack is defined as follows: first, the adversary $\mathcal{A}$ is given 1^d and names a target identity $id^* \in \mathcal{ID}^{\leq d}$. Next, $(mpk, msk) \leftarrow \mathsf{Setup}(1^d)$, $(\kappa, \sigma^*) \leftarrow \mathsf{Encaps}(id^*)$, and $\kappa' \leftarrow \{0,1\}^\ell$ are generated. Then $\mathcal{A}$ is given $(mpk, \kappa^*, \sigma^*)$, where κ^* is either κ or κ'. Finally, $\mathcal{A}$ may make extraction queries, i.e., it is given oracle access to $\mathsf{Extract}(sk_\varepsilon, \cdot)$, subject to the constraint that it may not query any identity that is a prefix of (or equal to) the target identity id^*. The advantage $\mathbf{Adv}^{\text{sid-ind-cpa}}_{\text{HIBE}}(\mathcal{A})$ of $\mathcal{A}$ is its distinguishing advantage between the two cases $\kappa^* = \kappa$ and $\kappa^* = \kappa'$.

Another notion is an *adaptive-identity* attack, in which the adversary is first given mpk and oracle access to $\mathsf{Extract}(sk_\varepsilon, \cdot)$ before choosing its target identity id^* (as before, under the constraint that no query identity be a prefix of id^*). Finally, both notions may be extended to *chosen-ciphertext* attacks in the natural way; we omit precise definitions.

2.3 Lattices

In this work, we use m-dimensional *full-rank integer* lattices, which are discrete additive subgroups of $\mathbb{Z}^m$ having finite index, i.e., the quotient group $\mathbb{Z}^m/\Lambda$ is finite. A lattice $\Lambda \subseteq \mathbb{Z}^m$ can equivalently be defined as the set of all integer linear combinations of m linearly independent *basis* vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{Z}^m$:

$$\Lambda = \mathcal{L}(\mathbf{B}) = \left\{ \mathbf{Bc} = \sum\nolimits_{i \in [m]} c_i \mathbf{b}_i \ : \ \mathbf{c} \in \mathbb{Z}^m \right\}.$$

When $m \geq 2$, there are infinitely many bases that generate the same lattice.

Every lattice $\Lambda \subseteq \mathbb{Z}^m$ has a *unique* canonical basis $\mathbf{H} = \mathrm{HNF}(\Lambda) \in \mathbb{Z}^{m \times m}$ called its *Hermite normal form* (HNF). The only facts about the HNF that we require are that it is unique, and that it may be computed efficiently given an arbitrary basis $\mathbf{B}$ of the lattice (see [42] and references therein). We write $\mathrm{HNF}(\mathbf{B})$ to denote the Hermite normal form of the lattice generated by basis $\mathbf{B}$.

The following lemma will be useful in our constructions.

Lemma 2.1 ([40, Lemma 7.1, page 129])**.** *There is a deterministic poly-time algorithm* $\mathsf{ToBasis}(\mathbf{S}, \mathbf{B})$ *that, given a full-rank set (not necessarily a basis) of lattice vectors* $\mathbf{S} \subset \Lambda = \mathcal{L}(\mathbf{B})$*, outputs a basis* $\mathbf{T}$ *of* Λ *such that* $\|\widetilde{\mathbf{t}_i}\| \leq \|\widetilde{\mathbf{s}_i}\|$ *for all* i*.*

2.3.1 Hard Lattices and Problems

We will work with an certain family of integer lattices whose importance in cryptography was first demonstrated by Ajtai [5]. Let $n \geq 1$ and modulus $q \geq 2$ be integers; the dimension n is the main cryptographic security parameter throughout this work, and all other parameters are implicitly functions of n. An m-dimensional lattice from the family is specified relative to the additive group $\mathbb{Z}_q^n$ by a *parity check* (more accurately, "arity check") matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The associated lattice is defined as

$$\Lambda^\perp(\mathbf{A}) = \left\{ \mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \sum\nolimits_{j \in [m]} x_j \cdot \mathbf{a}_j = \mathbf{0} \in \mathbb{Z}_q^n \right\} \subseteq \mathbb{Z}^m.$$

One may check that $\Lambda^\perp(\mathbf{A})$ contains $q\mathbb{Z}^m$ (and in particular, the identity $\mathbf{0} \in \mathbb{Z}^m$) and is closed under addition, hence it is a full-rank subgroup of (and lattice in) $\mathbb{Z}^m$. For any $\mathbf{y}$ in the subgroup of $\mathbb{Z}_q^n$ generated by the columns of $\mathbf{A}$, we also define the coset

$$\Lambda_{\mathbf{y}}^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{y}\} = \Lambda^\perp(\mathbf{A}) + \bar{\mathbf{x}},$$

where $\bar{\mathbf{x}} \in \mathbb{Z}^m$ is an arbitrary element of $\Lambda_{\mathbf{x}}^\perp$.

It is known (see, e.g., [51, Claim 5.3]) that for any fixed constant $C > 1$ and any $m \geq Cn \lg q$, the columns of a *uniformly random* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ generate all of $\mathbb{Z}_q^n$, except with $2^{-\Omega(n)} = \mathrm{negl}(n)$ probability. (Moreover, the subgroup generated by $\mathbf{A}$ can be computed efficiently.) Therefore, throughout the paper we sometimes implicitly assume that such a uniform $\mathbf{A}$ generates $\mathbb{Z}_q^n$.

We recall the *short integer solution* (SIS) and *learning with errors* (LWE) problems, which may be seen as average-case problems related to the family of lattices described above.

Definition 2.2 (Short Integer Solution)**.** An instance of the $\mathsf{SIS}_{q,\beta}$ problem (in the ℓ_2 norm) is a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for any desired $m = \mathrm{poly}(n)$. The goal is to find a *nonzero* integer vector $\mathbf{v} \in \mathbb{Z}^m$ such that $\|\mathbf{v}\|_2 \leq \beta$ and $\mathbf{A}\mathbf{v} = \mathbf{0} \in \mathbb{Z}_q^n$, i.e., $\mathbf{v} \in \Lambda^\perp(\mathbf{A})$.

Let χ be some distribution over $\mathbb{Z}_q$. For a vector $\mathbf{v} \in \mathbb{Z}_q^\ell$ of any dimension $\ell \geq 1$, $\mathsf{Noisy}_\chi(\mathbf{v}) \in \mathbb{Z}_q^\ell$ denotes the vector obtained by adding (modulo q) independent samples drawn from χ to each entry of $\mathbf{v}$ (one sample per entry). For a vector $\mathbf{s} \in \mathbb{Z}_q^n$, $A_{\mathbf{s},\chi}$ is the distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and outputting $(\mathbf{a}, \mathsf{Noisy}_\chi(\langle \mathbf{a}, \mathbf{s} \rangle))$. In this work (and most others relating to LWE), χ is always a discretized normal error distribution parameterized by some $\alpha \in (0, 1)$, which is obtained by drawing $x \in \mathbb{R}$ from the Gaussian distribution of width α (i.e., x is chosen with probability proportional to $\exp(-\pi x^2/\alpha^2)$) and outputting $\lfloor q \cdot x \rceil \bmod q$.

Definition 2.3 (Learning with Errors)**.** The $\mathsf{LWE}_{q,\chi}$ problem is to distinguish, given oracle access to any desired $m = \mathrm{poly}(n)$ samples, between the distribution $A_{\mathbf{s},\chi}$ (for uniformly random and secret $\mathbf{s} \in \mathbb{Z}_q^n$) and the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

We write $\mathbf{Adv}_{\mathsf{SIS}_{q,\beta}}(\mathcal{A})$ and $\mathbf{Adv}_{\mathsf{LWE}_{q,\chi}}(\mathcal{A})$ to denote the success probability and distinguishing advantage of an algorithm $\mathcal{A}$ for the SIS and LWE problems, respectively.

For appropriate parameters, solving SIS and LWE (on the average, with non-negligible advantage) is known to be as hard as approximating certain lattice problems, such as the (decision) shortest vector problem, in the worst case. Specifically, for $q \geq \beta \cdot \omega(\sqrt{n \log n})$, solving $\mathsf{SIS}_{q,\beta}$ yields approximation factors of $\tilde{O}(\beta \cdot \sqrt{n})$ [41, 25]. For $q \geq (1/\alpha) \cdot \omega(\sqrt{n \log n})$, solving $\mathsf{LWE}_{q,\chi}$ yields approximation factors of $\tilde{O}(n/\alpha)$ (in some cases, via a quantum reduction); see [51, 45] for precise statements.

2.3.2 Gaussians over Lattices

We briefly recall Gaussian distributions over lattices, specialized to the family described above; for more details see [41, 25]. For any $s > 0$ and dimension $m \geq 1$, the Gaussian function $\rho_s \colon \mathbb{R}^m \to (0, 1]$ is defined as $\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2/s^2)$. For any coset $\Lambda_{\mathbf{y}}^\perp(\mathbf{A})$, the *discrete Gaussian distribution* $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}$ (centered at zero) over the coset assigns probability proportional to $\rho_s(\mathbf{x})$ to each $\mathbf{x} \in \Lambda_{\mathbf{y}}^\perp(\mathbf{A})$, and probability zero elsewhere.

We summarize several standard facts from the literature about discrete Gaussians over lattices, again specialized to our family of interest.

Lemma 2.4. *Let $\mathbf{S}$ be any basis of $\Lambda^\perp(\mathbf{A})$ for some $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ whose columns generate $\mathbb{Z}_q^n$, let $\mathbf{y} \in \mathbb{Z}_q^n$ be arbitrary, and let $s \geq \|\widetilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$.*

1. *[41, Lemma 4.4]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}}[\|\mathbf{x}\| > s \cdot \sqrt{m}] \leq \mathrm{negl}(n)$.
2. *[47, Lemma 2.11]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda^\perp(\mathbf{A}),s}}[\mathbf{x} = \mathbf{0}] \leq \mathrm{negl}(n)$.
3. *[51, Corollary 3.16]: a set of $O(m^2)$ independent samples from $D_{\Lambda^\perp(\mathbf{A}),s}$ contains a set of m linearly independent vectors, except with $\mathrm{negl}(n)$ probability.*
4. *[25, Theorem 3.1]: For $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$, the marginal distribution of $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$ is uniform (up to $\mathrm{negl}(n)$ statistical distance), and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}$.*
5. *[25, Theorem 4.1]: there is a PPT algorithm $\mathsf{SampleD}(\mathbf{S}, \mathbf{y}, s)$ that generates a sample from $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}$ (up to $\mathrm{negl}(n)$ statistical distance).*

For Item 5 above, a recent work [46] gives an alternative SampleD algorithm that is more efficient and fully parallelizable; it works for any $s \geq \sigma_1(\mathbf{S}) \cdot \omega(\sqrt{\log n})$, where $\sigma_1(\mathbf{S})$ is the largest *singular value* of $\mathbf{S}$ (which is never less than $\|\widetilde{\mathbf{S}}\|$, but is also not much larger in most important cases; see [46] for details).

3 Principles of Bonsai Trees

In this section we lay out the framework and main techniques for the cultivation of bonsai trees. There are four basic principles: undirected growth, controlled growth, extending control over arbitrary new growth, and randomizing control.

3.1 Undirected Growth

Undirected growth is useful primarily for allowing a simulator to embed an underlying challenge problem (i.e., SIS or LWE) into a tree. This is done simply by drawing fresh uniformly random and independent samples $\mathbf{a}_i \in \mathbb{Z}_q^n$ from the problem distribution, and grouping them into (or appending them onto) a parity-check matrix $\mathbf{A}$.

More formally, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be arbitrary for some $m \geq 0$, and let $\mathbf{A}' = \mathbf{A} \| \bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m'}$ for some $m' > m$ be an arbitrary extension of $\mathbf{A}$. Then it is easy to see that $\Lambda^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m'}$ is a *higher-dimensional superlattice* of $\Lambda^\perp(\mathbf{A}) \subseteq \mathbb{Z}^m$, when the latter is lifted to $\mathbb{Z}^{m'}$. Specifically, for any $\mathbf{v} \in \Lambda^\perp(\mathbf{A})$, the vector $\mathbf{v}' = \mathbf{v} \| \mathbf{0} \in \mathbb{Z}^{m'}$ is in $\Lambda^\perp(\mathbf{A}')$ because $\mathbf{A}'\mathbf{v}' = \mathbf{A}\mathbf{v} = \mathbf{0} \in \mathbb{Z}_q^n$.

In fact, the columns of $\mathbf{A}'$ may be ordered arbitrarily (e.g., the columns of $\bar{\mathbf{A}}$ may be both appended and *prepended* to $\mathbf{A}$), which simply results in the entries of the vectors in $\Lambda^\perp(\mathbf{A}')$ being permuted in the corresponding manner. That is, $\Lambda^\perp(\mathbf{A}'\mathbf{P}) = \mathbf{P} \cdot \Lambda^\perp(\mathbf{A}')$ for any permutation matrix $\mathbf{P} \in \{0,1\}^{m' \times m'}$, because $(\mathbf{A}'\mathbf{P})\mathbf{x} = \mathbf{A}'(\mathbf{P}\mathbf{x}) \in \mathbb{Z}_q^n$ for all $\mathbf{x} = \mathbb{Z}^{m'}$.

3.2 Controlled Growth

We say that an arborist *controls* a lattice if it knows a relatively good (i.e., short) basis for the lattice. The following lemma says that a random lattice from our family of interest can be generated under control.[3]

Proposition 3.1 ([6])**.** *There is a fixed constant $C > 1$ and a probabilistic polynomial-time algorithm* $\mathsf{GenBasis}(1^n, 1^m, q)$ *that, for* $\mathrm{poly}(n)$*-bounded $m \geq Cn \lg q$, outputs $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{S} \in \mathbb{Z}^{m \times m}$ such that:*

- *the distribution of $\mathbf{A}$ is within* $\mathrm{negl}(n)$ *statistical distance of uniform,*
- *$\mathbf{S}$ is a basis of $\Lambda^\perp(\mathbf{A})$, and*
- $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L} = O(\sqrt{n \log q})$.

3.3 Extending Control

Here we describe how an arborist may extend its control of a lattice to an arbitrary higher-dimensional extension, without any loss of quality in the resulting basis.

[3] An earlier version of this paper [44] used an underlying lemma from [6] to *directly* extend a random parity-check matrix $\mathbf{A}$ (*without* known good basis) into a random $\mathbf{A}' = \mathbf{A} \| \bar{\mathbf{A}}$ *with* known good basis. While that method saves a small constant factor in key sizes, the applications become somewhat more cumbersome to describe; moreover, our present approach is more general.

Lemma 3.2. *Let $\mathbf{S} \in \mathbb{Z}^{m \times m}$ be an arbitrary basis of $\Lambda^\perp(\mathbf{A})$ for some $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ whose columns generate the entire group $\mathbb{Z}_q^n$, and let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ be arbitrary. There is a deterministic polynomial-time algorithm $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}})$ that outputs a basis $\mathbf{S}'$ of $\Lambda^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$ such that $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$. Moreover, the statement holds even if the columns of $\mathbf{A}'$ are permuted arbitrarily (e.g., if columns of $\bar{\mathbf{A}}$ are both appended and prepended to $\mathbf{A}$).*

Proof. The $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}')$ algorithm computes and outputs an $\mathbf{S}'$ of the form

$$\mathbf{S}' = \begin{pmatrix} \mathbf{S} & \mathbf{W} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

where $\mathbf{I} \in \mathbb{Z}^{\bar{m} \times \bar{m}}$ is the identity matrix, and $\mathbf{W} \in \mathbb{Z}^{m \times \bar{m}}$ is an *arbitrary* (not necessarily short) solution to $\mathbf{AW} = -\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$. Note that $\mathbf{W}$ exists by hypothesis on $\mathbf{A}$, and may be computed efficiently using Gaussian elimination (for example).

We analyze $\mathbf{S}'$. First, $\mathbf{A}'\mathbf{S}' = \mathbf{0}$ by assumption on $\mathbf{S}$ and by construction, so $\mathbf{S}' \subset \Lambda^\perp(\mathbf{A}')$. Moreover, $\mathbf{S}'$ is a *basis* of $\Lambda^\perp(\mathbf{A}')$: let $\mathbf{v}' = \mathbf{v}\|\bar{\mathbf{v}} \in \Lambda^\perp(\mathbf{A}')$ be arbitrary, where $\mathbf{v} \in \mathbb{Z}^m$, $\bar{\mathbf{v}} \in \mathbb{Z}^{\bar{m}}$. Then we have

$$\mathbf{0} = \mathbf{A}'\mathbf{v}' = \mathbf{A}\mathbf{v} + \bar{\mathbf{A}}\bar{\mathbf{v}} = \mathbf{A}\mathbf{v} - (\mathbf{AW})\bar{\mathbf{v}} = \mathbf{A}(\mathbf{v} - \mathbf{W}\bar{\mathbf{v}}) \in \mathbb{Z}_q^n.$$

Thus $\mathbf{v} - \mathbf{W}\bar{\mathbf{v}} \in \Lambda^\perp(\mathbf{A})$, so by assumption on $\mathbf{S}$ there exists some $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Sz} = \mathbf{v} - \mathbf{W}\bar{\mathbf{v}}$. Now let $\mathbf{z}' = \mathbf{z}\|\bar{\mathbf{v}} \in \mathbb{Z}^{m+\bar{m}}$. By construction, we have

$$\mathbf{S}'\mathbf{z}' = (\mathbf{Sz} + \mathbf{W}\bar{\mathbf{v}})\|\bar{\mathbf{v}} = \mathbf{v}\|\bar{\mathbf{v}} = \mathbf{v}'.$$

Because $\mathbf{v}' \in \Lambda^\perp(\mathbf{A}')$ was arbitrary, $\mathbf{S}'$ is therefore a basis of $\Lambda^\perp(\mathbf{A}')$.

We next confirm that $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$. For every $i \in [m]$, we clearly have $\|\widetilde{\mathbf{s}'_i}\| = \|\widetilde{\mathbf{s}_i}\|$. Now because $\mathbf{S}$ is full-rank, we have $\mathrm{span}(\mathbf{S}) = \mathrm{span}(\mathbf{e}_1, \ldots, \mathbf{e}_m) \subseteq \mathbb{R}^{m+\bar{m}}$. Therefore, for $i = m+1, \ldots, m+\bar{m}$ we have $\widetilde{\mathbf{s}'_i} = \mathbf{e}_i \in \mathbb{R}^{m+\bar{m}}$, so $\|\widetilde{\mathbf{s}'_i}\| = 1 \leq \|\widetilde{\mathbf{s}'_1}\|$, as desired.

For the final part of the lemma, we simply compute $\mathbf{S}'$ for $\mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}}$ as described above, and output $\mathbf{S}'' = \mathbf{PS}'$ as a basis for $\Lambda^\perp(\mathbf{A}'\mathbf{P})$, where $\mathbf{P}$ is the desired permutation matrix. The Gram-Schmidt lengths remain unchanged, i.e., $\|\widetilde{\mathbf{s}''_i}\| = \|\widetilde{\mathbf{s}'_i}\|$, because $\mathbf{P}$ is orthogonal and hence the right-triangular matrices are exactly the same in the QR decompositions of $\mathbf{S}'$ and $\mathbf{PS}'$. □

3.3.1 An Optimization

In many of our cryptographic applications, a common design pattern is to extend a basis $\mathbf{S}$ of an m-dimensional lattice $\Lambda^\perp(\mathbf{A})$ to a basis $\mathbf{S}'$ of a dimension-m' superlattice $\Lambda^\perp(\mathbf{A}')$, and then immediately sample (one or more times) from a discrete Gaussian over the superlattice. For the construction and analysis of our schemes, it is more convenient and modular to treat these operations separately; however, a naive implementation would be rather inefficient, requiring at least $(m')^2$ space and time (where m' can be substantially larger than m). Fortunately, the special structure of the extended basis $\mathbf{S}'$, together with the recursive "nearest-plane" operation of the $\mathsf{SampleD}$ algorithm from [25], can be exploited to avoid any *explicit* computation of $\mathbf{S}'$, thus saving a significant amount of time and space over the naive approach.

Let $\mathbf{S} \in \mathbb{Z}^{m \times m}$ be a basis of $\Lambda^\perp(\mathbf{A})$, and let $\mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}}$ for some $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$, where $m' = m + \bar{m}$. Consider a hypothetical execution of $\mathsf{SampleD}(\mathbf{S}', \mathbf{y}', s)$, where $\mathbf{S}' = \left(\begin{smallmatrix} \mathbf{S} & \mathbf{W} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$ is the extended basis as described in the proof of Lemma 3.2. Recall that for all $i = m+1, \ldots, m'$, the vectors $\mathbf{s}'_i$ are integral and have unit Gram-Schmidt vectors $\widetilde{\mathbf{s}'_i} = \mathbf{e}_i$. By inspection, it can be verified that a recursive execution

of $\mathsf{SampleD}(\mathbf{S}', \mathbf{y}', s)$ simply ends up choosing all the entries of $\bar{\mathbf{v}} \in \mathbb{Z}^{\bar{m}}$ *independently* from $D_{\mathbb{Z},s}$, then choosing $\mathbf{v} \leftarrow \mathsf{SampleD}(\mathbf{S}, \mathbf{y}' - \bar{\mathbf{A}}\bar{\mathbf{v}}, s)$, and outputting $\mathbf{v}' = \mathbf{v} \| \bar{\mathbf{v}}$. Therefore, the optimized algorithm can perform exactly the same steps, thus avoiding any need to compute and store $\mathbf{W}$ itself. A similar optimization also works for any permutation of the columns of $\mathbf{A}'$.

In the language of the "preimage sampleable" function $f_{\mathbf{A}}(\mathbf{v}) = \mathbf{A}\mathbf{v} \in \mathbb{Z}_q^n$ defined in [25], the process described above corresponds to sampling a preimage from $f_{\mathbf{A}'}^{-1}(\mathbf{y}')$ by first computing $\bar{\mathbf{y}} = f_{\bar{\mathbf{A}}}(\bar{\mathbf{v}}) = \bar{\mathbf{A}}\bar{\mathbf{v}} \in \mathbb{Z}_q^n$ in the "forward" direction (for random $\bar{\mathbf{v}} \leftarrow \mathbb{D}_{\mathbb{Z}^{\bar{m}},s}$), then choosing a random preimage $\mathbf{v} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{y}' - \bar{\mathbf{y}})$ under the appropriate distribution, and outputting $\mathbf{v}' = \mathbf{v} \| \bar{\mathbf{v}}$.[4]

3.4 Randomizing Control

Finally, we show how an arborist can randomize its lattice basis, with a slight loss in quality. This operation is useful for securely delegating control to another entity, because the resulting basis is still short, but is statistically independent (essentially) of the original basis.

The probabilistic polynomial-time algorithm $\mathsf{RandBasis}(\mathbf{S}, s)$ takes a basis $\mathbf{S}$ of an m-dimensional integer lattice Λ and a parameter $s \geq \|\widetilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$, and outputs a basis $\mathbf{S}'$ of Λ, generated as follows.

1. Let $i \leftarrow 0$. While $i < m$,
   (a) Choose $\mathbf{v} \leftarrow \mathsf{SampleD}(\mathbf{S}, s)$. If $\mathbf{v}$ is linearly independent of $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$, then let $i \leftarrow i + 1$ and let $\mathbf{v}_i = \mathbf{v}$.
2. Output $\mathbf{S}' = \mathsf{ToBasis}(\mathbf{V}, \mathrm{HNF}(\mathbf{S}))$.

In the final step, the (unique) Hermite normal form basis $\mathrm{HNF}(\mathbf{S})$ of Λ is used to ensure that no information particular to $\mathbf{S}$ is leaked by the output; any other publicly available (or publicly computable) basis of the lattice could also be used in its place.

Lemma 3.3. *With overwhelming probability, $\mathbf{S}' \leftarrow \mathsf{RandBasis}(\mathbf{S}, s)$ repeats Step 1a at most $O(m^2)$ times, and $\|\mathbf{S}'\| \leq s \cdot \sqrt{m}$. Moreover, for any two bases $\mathbf{S}_0, \mathbf{S}_1$ of the same lattice and any $s \geq \max\{\|\widetilde{\mathbf{S}_0}\|, \|\widetilde{\mathbf{S}_1}\|\} \cdot \omega(\sqrt{\log n})$, the outputs of $\mathsf{RandBasis}(\mathbf{S}_0, s)$ and $\mathsf{RandBasis}(\mathbf{S}_1, s)$ are within $\mathrm{negl}(n)$ statistical distance.*

Proof. The bound on $\|\mathbf{S}'\|$ and on the number of iterations follow immediately from Lemma 2.4, items 1 and 3, respectively. The claim on the statistical distance follows from the fact that each sample $\mathbf{v}$ drawn in Step 1a has the same distribution (up to $\mathrm{negl}(n)$ statistical distance) whether $\mathbf{S}_0$ or $\mathbf{S}_1$ is used, and the fact that $\mathrm{HNF}(\mathbf{S}_0) = \mathrm{HNF}(\mathbf{S}_1)$ because the Hermite normal form of a lattice is unique. □

4 Signatures

Here we use bonsai tree principles to construct a signature scheme that is existentially unforgeable under a *static* chosen-message attack (i.e., eu-scma-secure). As discussed in Section 2.2, this suffices (using chameleon hashing) for the construction of an (offline / online) signature scheme that is unforgeable under adaptive chosen-message attack (eu-acma-secure).

Our scheme involves a few parameters:

[4] An earlier version of this paper [17] explicitly defined a sampling procedure using this perspective, and gave a (somewhat involved) proof that it correctly samples from a discrete Gaussian over $\Lambda^{\perp}(\mathbf{A}')$. Here, correctness follows directly by examining the operation of $\mathsf{SampleD}$ on the structured basis $\mathbf{S}'$.

- a dimension $m = O(n \lg q)$ and a bound $\widetilde{L} = O(\sqrt{n \lg q})$, as per Proposition 3.1;
- a (hashed) message length k, which induces a 'total dimension' $m' = m \cdot (k+1)$;
- a Gaussian parameter $s = \widetilde{L} \cdot \omega(\sqrt{\log n})$.

The scheme SIG is defined as follows.

- Gen: using Proposition 3.1, generate $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ that is (negligibly close to) uniform, together with a basis $\mathbf{S}_0$ of $\Lambda^\perp(\mathbf{A}_0)$ such that $\|\widetilde{\mathbf{S}_0}\| \leq \widetilde{L}$. (Recall that the columns of $\mathbf{A}_0$ generate all of $\mathbb{Z}_q^n$, with overwhelming probability.)

  Then for each $(b, j) \in \{0,1\} \times [k]$, choose uniformly random and independent $\mathbf{A}_j^{(b)} \in \mathbb{Z}_q^{n \times m}$. Output $vk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\})$ and $sk = (\mathbf{S}_0, vk)$.

- $\mathsf{Sign}(sk, \mu \in \{0,1\}^k)$: let $\mathbf{A}_\mu = \mathbf{A}_0 \| \mathbf{A}_1^{(\mu_1)} \| \cdots \| \mathbf{A}_k^{(\mu_k)} \in \mathbb{Z}_q^{n \times m'}$. Output $\mathbf{v} \leftarrow D_{\Lambda^\perp(\mathbf{A}_\mu), s}$, via

$$\mathbf{v} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_0, \mathbf{A}_\mu), \mathbf{0}, s).$$

  (In the rare event that $\mathbf{v} = \mathbf{0}$ or $\|\mathbf{v}\| > s \cdot \sqrt{m'}$ (Lemma 2.4, items 2 and 2), resample $\mathbf{v}$. Note also that the optimization of Section 3.3.1 applies here.)

- $\mathsf{Ver}(vk, \mu, \mathbf{v})$: let $\mathbf{A}_\mu$ be as above. Accept if $\mathbf{v} \neq \mathbf{0}$, $\|\mathbf{v}\| \leq s \cdot \sqrt{m'}$, and $\mathbf{v} \in \Lambda^\perp(\mathbf{A}_\mu)$; else, reject.

Completeness is by inspection. Note that the matrix $\mathbf{A}_0$ can be omitted from the above scheme (thus making the total dimension $m \cdot k$), at the expense of a secret key that contains *two* short bases $\mathbf{S}_1^{(b)}$ of $\Lambda^\perp(\mathbf{A}_1^{(b)})$, for $b = 0, 1$. The scheme's algorithms and security proof are easy to modify accordingly.

4.1 Security

Theorem 4.1. *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking the $\mathsf{SIS}_{q,\beta}$ problem for $\beta = s \cdot \sqrt{m'}$ such that, for any adversary $\mathcal{F}$ mounting an eu-scma attack on SIG and making at most Q queries,*

$$\mathbf{Adv}_{\mathsf{SIS}_{q,\beta}}(\mathcal{S}^{\mathcal{F}}) \geq \mathbf{Adv}_{SIG}^{\textit{eu-scma}}(\mathcal{F})/(k \cdot Q) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{F}$ be an adversary mounting an eu-scma attack on SIG. We construct a reduction $\mathcal{S}$ attacking $\mathsf{SIS}_{q,\beta}$. The reduction $\mathcal{S}$ takes as input $m'' = m \cdot (2k+1)$ uniformly random and independent samples from $\mathbb{Z}_q^n$ in the form of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m''}$, parsing $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{A}_0 \| \mathbf{U}_1^{(0)} \| \mathbf{U}_1^{(1)} \| \cdots \| \mathbf{U}_k^{(0)} \| \mathbf{U}_k^{(1)}$$

for matrices $\mathbf{A}_0, \mathbf{U}_i^{(b)} \in \mathbb{Z}_q^{n \times m}$.

$\mathcal{S}$ simulates the static chosen-message attack to $\mathcal{F}$ as follows. First, $\mathcal{S}$ invokes $\mathcal{F}$ to receive Q messages $\mu^{(1)}, \ldots, \mu^{(Q)} \in \{0,1\}^k$. (We may assume without loss of generality that $\mathcal{F}$ makes exactly Q queries.) Then $\mathcal{S}$ computes the set P of all strings $p \in \{0,1\}^{\leq k}$ having the property that p is a shortest string for which no $\mu^{(j)}$ has p as a prefix. In brief, each p corresponds to a maximal subtree of $\{0,1\}^{\leq k}$ (viewed as a tree) that does not contain any of the queried messages. The set P may be computed efficiently via a breadth-first pruned search of $\{0,1\}^{\leq k}$. Namely, starting from a queue initialized to $\{\varepsilon\}$, repeat the following until the queue is empty: remove the next string p from the queue and test whether it is the prefix of any $\mu^{(j)}$; if not,

add p to P, else if $|p| < k$, add $p\|0, p\|1 \in \{0,1\}^{\leq k}$ to the queue. Note that this algorithm runs in polynomial time because the only strings ever placed in the queue are prefixes of $\mu^{(j)}$, and hence there are at most $k \cdot Q$ strings in the set.

Next, $\mathcal{S}$ chooses some p from P uniformly at random, letting $t = |p|$. It then provides an SIG verification key $vk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\})$ to $\mathcal{F}$, generated as follows:

- *Uncontrolled growth:* for each $i \in [t]$, let $\mathbf{A}_i^{(p_i)} = \mathbf{U}_i^{(0)}$. For $i = t+1, \ldots, k$, and $b \in \{0,1\}$, let $\mathbf{A}_i^{(b)} = \mathbf{U}_i^{(b)}$.
- *Controlled growth:* for each $i \in [t]$, invoke Proposition 3.1 to generate $\mathbf{A}_i^{(1-p_i)}$ and basis $\mathbf{S}_i$ of $\Lambda^{\perp}(\mathbf{A}_i^{1-p_i})$ such that $\|\widetilde{\mathbf{S}}_i\| \leq \widetilde{L}$.

Next, $\mathcal{S}$ generates signatures for each queried message $\mu = \mu^{(j)}$ as follows: let $i \in [t]$ be the first position at which $\mu_i \neq p_i$ (such i exists by construction of p). Then $\mathcal{S}$ generates the signature $\mathbf{v} \leftarrow D_{\Lambda^{\perp}(\mathbf{A}_\mu),s}$ as

$$\mathbf{v} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_i, \mathbf{A}_\mu), \mathbf{0}, s),$$

where $\mathbf{A}_\mu = \mathbf{A}_L \| \mathbf{A}_i^{(1-p_i)} \| \mathbf{A}_R$ (for some matrices $\mathbf{A}_L, \mathbf{A}_R$) is as in the signature scheme, and has the form required by $\mathsf{ExtBasis}$. (In the event that $\mathbf{v} = \mathbf{0}$ or $\|\mathbf{v}\| > \beta = s \cdot \sqrt{m'}$, resample $\mathbf{v}$.)

Finally, if $\mathcal{F}$ produces a valid forgery $(\mu^*, \mathbf{v}^* \neq \mathbf{0})$, then we have $\mathbf{v}^* \in \Lambda^{\perp}(\mathbf{A}_{\mu^*})$, for $\mathbf{A}_{\mu^*}$ as defined in the scheme. First, $\mathcal{S}$ checks whether p is a prefix of μ^*. If not, $\mathcal{S}$ aborts; otherwise, note that $\mathbf{A}_{\mu^*}$ is the concatenation of $\mathbf{A}_0$ and k blocks $\mathbf{U}_i^{(b)}$. Therefore, by inserting zeros into $\mathbf{v}^*$, $\mathcal{S}$ can generate a nonzero $\mathbf{v} \in \mathbb{Z}^{m''}$ so that $\mathbf{A}\mathbf{v} = \mathbf{0} \in \mathbb{Z}_q^n$. Finally, $\mathcal{S}$ outputs $\mathbf{v}$ as a solution to SIS.

We now analyze the reduction. First observe that conditioned on any choice of $p \in P$, the verification key vk given to $\mathcal{F}$ is negligibly close to uniform, and the signatures given to $\mathcal{F}$ are distributed exactly as in the real attack (up to negligible statistical distance), by Lemma 2.4 and the fact that $s \geq \|\widetilde{\mathbf{S}}_i\| \cdot \omega(\sqrt{\log n})$. Therefore, $\mathcal{F}$ outputs a valid forgery $(\mu^*, \mathbf{v}^* \neq \mathbf{0})$ with probability at least $\mathbf{Adv}_{\mathrm{SIG}}^{\text{eu-scma}}(\mathcal{F}) - \mathrm{negl}(n)$. Finally, conditioned on the forgery, the choice of $p \in P$ is still negligibly close to uniform, so p is a prefix of μ^* with probability at least $1/(k \cdot Q) - \mathrm{negl}(n)$. In such a case, $\mathbf{A}\mathbf{v} = \mathbf{0}$ and $\|\mathbf{v}\| = \|\mathbf{v}^*\| \leq \beta$ by construction, hence $\mathbf{v}$ is a valid solution to the given SIS instance, as desired. □

5 Hierarchical ID-Based Encryption

5.1 Key Encapsulation Mechanism

For our HIBE schemes, it is convenient and more modular to abstract away the encryption and decryption processes into a key-encapsulation mechanism (KEM). The following LWE-based KEM from [25] (which is dual to the scheme of Regev [51]) is now standard. The reader need not be concerned with the details in order to progress to the HIBE schemes; it is enough simply to understand the KEM interface (i.e., the public/secret keys and ciphertext).

KEM is parametrized by a modulus q, dimension m, key length ℓ, and Gaussian parameter s that determines the error distribution χ used for encapsulation. As usual, all these parameters are functions of the LWE dimension n, and are instantiated based on the particular context in which the KEM is used.

- $\mathsf{KEM.Gen}$: Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ uniformly at random, $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,s}$ and set $\mathbf{y} = \mathbf{A}\mathbf{e} \in \mathbb{Z}_q^n$. Output public key $pk = (\mathbf{A}, \mathbf{y}) \in \mathbb{Z}_q^{n \times (m+1)}$ and secret key $sk = \mathbf{e}$.

- $\mathsf{KEM.Encaps}(pk = (\mathbf{A}, \mathbf{y}))$: Choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and let

$$\mathbf{b} \leftarrow \mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s}) \quad \text{and} \quad p \leftarrow \mathsf{Noisy}_\chi(\mathbf{y}^t\mathbf{s} + k \cdot \lfloor q/2 \rfloor),$$

  where $k \in \{0, 1\}$ is a random bit. Output the key bit k and ciphertext $(\mathbf{b}, p) \in \mathbb{Z}_q^{m+1}$.

- $\mathsf{KEM.Decaps}(sk = \mathbf{e}, (\mathbf{b}, p))$: Compute $p - \mathbf{e}^t\mathbf{b} \bmod q$ and output 0 if the result is closer to 0 than $\lfloor q/2 \rfloor$ modulo q, and 1 otherwise.

As explained in [25], the basic scheme can be amortized to allow for KEM keys of length $\ell = \mathrm{poly}(n)$ bits, with ciphertexts in $\mathbb{Z}_q^{m+\ell}$ and public keys in $\mathbb{Z}_q^{n\times(m+\ell)}$. This is done by including ℓ syndromes $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$ (where $\mathbf{y}_i = \mathbf{A}\mathbf{e}_i$ for independent $\mathbf{e}_i \leftarrow D_{\mathbb{Z}^m,s}$) in the public key, and concealing one KEM bit with each of them using the same $\mathbf{s}$ and $\mathbf{b} \leftarrow \mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s})$. Furthermore, it is also possible to conceal $\Omega(\log n)$ KEM bits per syndrome, which yields an amortized expansion factor of $O(1)$. For simplicity, in this work we deal only with the case of single-bit encapsulation, but all of our schemes can be amortized in a manner similar to the above.

We point out one nice property of KEM, which is convenient for the security proof of our BTE/HIBE schemes: for any dimensions $m \le m'$ (and leaving all other parameters the same), the adversary's view for dimension m may be produced by taking a view for dimension m', and truncating the values $\mathbf{A} \in \mathbb{Z}_q^{n\times m'}$ and $\mathbf{b} \in \mathbb{Z}_q^{m'}$ to their first m (out of m') components.

The following lemma is standard from prior work.

Lemma 5.1 (Correctness and Security)**.** *Let $m \ge Cn \lg q$ for any fixed constant $C > 1$, let $q \ge 4s(m+1)$, and let χ be the discretized Gaussian of parameter α for $1/\alpha \ge s\sqrt{m+1} \cdot \omega(\sqrt{\log n})$. Then* $\mathsf{KEM.Decaps}$ *is correct with overwhelming probability over all the randomness of* $\mathsf{KEM.Gen}$ *and* $\mathsf{KEM.Encaps}$*. Moreover, there exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking the* $\mathsf{LWE}_{q,\chi}$ *problem such that, for any adversary $\mathcal{A}$ mounting an ind-cpa attack on KEM,*

$$\mathbf{Adv}_{\mathsf{LWE}_{q,\chi}}(\mathcal{S}^{\mathcal{A}}) \ge \mathbf{Adv}^{\textit{ind-cpa}}_{\textit{KEM}}(\mathcal{A}) - \mathrm{negl}(n).$$

5.2 BTE and HIBE Scheme

Our main construction in this section is a binary tree encryption (BTE) scheme, which suffices for full HIBE by hashing the components of the identities with a universal one-way or collision-resistant hash function [16]. We mainly focus on the case of *selective-identity*, *chosen-plaintext* attacks, i.e., sid-ind-cpa security.

The BTE scheme is parametrized by a dimension $m = O(n \lg q)$ as per Proposition 3.1, as well as a few quantities that are indexed by depth within the hierarchy. For an identity at depth $i \ge 0$ (where $i = 0$ corresponds to the root),

- $(i+1)m$ is the dimension of a lattice associated with the identity;
- $\widetilde{L_i}$ is an upper bound on the Gram-Schmidt lengths of its secret short basis;
- for $i \ge 1$, s_i is the Gaussian parameter used to generate that secret basis, which must exceed $\widetilde{L_j} \cdot \omega(\sqrt{\log n})$ for all $j < i$.

These parameters, along with the total depth d of the hierarchy (or more accurately, the maximum number of delegations down any chain of authority), determine the modulus q and error distribution χ used in the cryptosystem. We instantiate all the parameters after describing the scheme.

- $\mathsf{BTE.Setup}(d)$: Generate (via Proposition 3.1) $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ that is (negligibly close to) uniform with a basis $\mathbf{S}_0$ of $\Lambda^\perp(\mathbf{A}_0)$ such that $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L_0}$. For each $(b, j) \in \{0,1\} \times [d]$, generate uniform and independent $\mathbf{A}_j^{(b)} \in \mathbb{Z}_q^{n \times m}$. Choose $\mathbf{y} \in \mathbb{Z}_q^n$ uniformly at random. Output $mpk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\}, \mathbf{y}, d)$ and $msk = \mathbf{S}_0$.

  All remaining algorithms implicitly take the master public key mpk as input. For an identity $id = (id_1, \ldots, id_t)$ of length $t = |id| \leq d$, let

$$\mathbf{A}_{id} = \mathbf{A}_0 \| \mathbf{A}_1^{(id_1)} \| \cdots \| \mathbf{A}_t^{(id_t)} \in \mathbb{Z}_q^{n \times (t+1)m},$$

  and let $pk_{id} = (\mathbf{A}_{id}, \mathbf{y})$ denote the KEM public key associated with identity id.

- $\mathsf{BTE.Extract}(sk_{id} = (\mathbf{S}_{id}, \mathbf{e}_{id}), id' = id \| \bar{id})$: if $t' = |id'| > d$, output $\perp$. Else, let $t = |id|$ and $\bar{t} = |\bar{id}|$, and choose

$$\mathbf{S}_{id'} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{S}_{id}, \mathbf{A}_{id'}), s_{t'}).$$

  (Note that $s_{t'} \geq \widetilde{L_t} \cdot \omega(\sqrt{\log n}) \geq \|\widetilde{\mathbf{S}_{id}}\| \cdot \omega(\sqrt{\log n})$, as required by $\mathsf{RandBasis}$.) Sample $\mathbf{e}_{id'} \leftarrow D_{\Lambda^\perp_{\mathbf{y}}(\mathbf{A}_{id'}), s_{t'}}$ using $\mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_{id}, \mathbf{A}_{id'}), \mathbf{y}_{id'}, s_{t'})$ and output $sk_{id'} = (\mathbf{S}_{id'}, \mathbf{e}_{id'})$.

- $\mathsf{BTE.Encaps}(id)$: Output $(k, C) \leftarrow \mathsf{KEM.Encaps}(pk_{id})$.

- $\mathsf{BTE.Decaps}(sk_{id} = (\mathbf{S}_{id}, \mathbf{e}_{id}), C)$: Output $k \leftarrow \mathsf{KEM.Decaps}(\mathbf{e}_{id}, C)$.

A multi-bit BTE follows in the same way from the multi-bit KEM scheme by using multiple uniform syndromes $\mathbf{y}_i \in \mathbb{Z}_q^n$, one for each bit of the KEM key.

Instantiating the parameters. Suppose that BTE is employed in a setting in which $\mathsf{BTE.Extract}(sk_{id}, id')$ is invoked only on identities id' whose lengths are a multiple of some $k \geq 1$. For example, consider the two main applications of [16]: in the forward-secure encryption scheme we have $k = 1$, while in the generic BTE-to-HIBE transformation, k is the output length of some UOWHF.

It is enough to define s_i and $\widetilde{L_i}$ for i that are multiples of k. Let

$$\widetilde{L_i} = s_i \cdot \sqrt{(i+1)m} = s_i \cdot O(\sqrt{d \cdot n \lg q})$$

be the bound on the Gram-Schmidt lengths of the secret bases (and note that this bound is satisfied with overwhelming probability by Lemma 2.4). Define $s_i = \widetilde{L_{i-k}} \cdot \omega(\sqrt{\log n})$, and unwind the recurrence to obtain

$$\widetilde{L_t} = \widetilde{L_0} \cdot O(\sqrt{d \cdot n \lg q})^{t/k} \cdot \omega(\sqrt{\log n})^{t/k},$$

with $\widetilde{L_0} = O(\sqrt{n \lg q})$ by Proposition 3.1.

Finally, to ensure that the underlying KEM is complete (Lemma 5.1), we let $q \geq 4 s_d \cdot (d+2)m$ and $1/\alpha = s_d \cdot \sqrt{(d+2)m} \cdot \omega(\sqrt{\log n})$. (It is also possible to use a different noise parameter for each level of the hierarchy.) For any $d = \mathrm{poly}(n)$, invoking known worst-case to average-case reductions for LWE yields an underlying approximation factor of $\tilde{O}(n/\alpha) = n \cdot \tilde{O}((d/k) \cdot \sqrt{nk}))^{d/k}$ for worst-case lattice problems.

Extensions: Anonymity and chosen-ciphertext security. With a small modification, BTE may be made *anonymous* across all depths of the hierarchy. That is, a ciphertext hides (computationally) the particular identity to which it was encrypted. The modification is simply to extend the $\mathbf{b}$ component of the KEM ciphertext to have length exactly $(d+1)m$, by padding it with enough uniformly random and independent elements of $\mathbb{Z}_q$. (The decryption algorithm simply ignores the padding.) Anonymity then follows immediately by the pseudorandomness of the LWE distribution.

Security under chosen-ciphertext attack (sid-ind-cca or aid-ind-cca) follows directly by a transformation of [10], from ind-cpa-secure HIBE for depth $d+1$ to ind-cca-secure HIBE for depth d.

Theorem 5.2 (Security of BTE)**.** *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking KEM (instantiated with dimension $(d+1)m$ and q, χ as in BTE) such that, for any adversary $\mathcal{A}$ mounting an atk attack on BTE,*

$$\mathbf{Adv}_{\mathit{KEM}}(\mathcal{S}^{\mathcal{A}}) \geq \mathbf{Adv}^{\textit{sid-ind-cpa}}_{\mathit{BTE}}(\mathcal{A}) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{A}$ be an adversary mounting a sid-ind-cpa-attack on BTE. We construct a reduction $\mathcal{S}$ attacking KEM. It is given a uniformly random public key $pk = (\mathbf{A}, \mathbf{y}) \in \mathbb{Z}_q^{n \times (d+1)m} \times \mathbb{Z}_q^n$, an encapsulation $(\mathbf{b}, p) \in \mathbb{Z}_q^{(d+1)m} \times \mathbb{Z}_q$, and a bit k which either is encapsulated by $(\mathbf{b}, p)$ or is uniform and independent; the goal of $\mathcal{S}$ is to determine which is the case.

$\mathcal{S}$ simulates the (selective-identity) attack on BTE to $\mathcal{A}$ as follows. First, $\mathcal{S}$ invokes $\mathcal{A}$ on 1^d to receive its challenge identity id^* of length $t^* = |id^*| \in [d]$. Then $\mathcal{S}$ produces a master public key mpk, encapsulated key, and some secret internal state as follows:

- *Parsing the KEM inputs.* Parse $\mathbf{A}$ as $\mathbf{A} = \mathbf{A}_0 \| \mathbf{A}_1 \| \cdots \| \mathbf{A}_d \in \mathbb{Z}_q^{n \times (d+1)m}$ for $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ for all $i \in \{0, \ldots, d\}$. Similarly, truncate $\mathbf{b}$ to $\mathbf{b}^* \in \mathbb{Z}_q^{(t^*+1)m}$.
- *Undirected growth.* For each $i \in [t^*]$, let $\mathbf{A}_i^{(id_i^*)} = \mathbf{A}_i$.
- *Controlled growth.* For each $i \in [t^*]$, generate $\mathbf{A}_i^{(1-id_i^*)} \in \mathbb{Z}_q^{n \times m}$ and basis $\mathbf{S}_i$ by invoking $\mathsf{GenBasis}(1^n, 1^m, q)$. If $t^* < d$, for each $b \in \{0,1\}$ generate $\mathbf{A}_{t^*+1}^{(b)}$ and basis $\mathbf{S}_{t^*+1}^{(b)}$ by two independent invocations of $\mathsf{GenBasis}(1^n, 1^m, q)$. For each $i > t^*+1$ (if any) and $b \in \{0,1\}$, generate $\mathbf{A}_i^{(b)} \in \mathbb{Z}_q^{n \times m}$ uniformly at random.

$\mathcal{S}$ gives to $\mathcal{A}$ the master public key $mpk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\}, \mathbf{y}, d)$, the encapsulation $(\mathbf{b}^*, p)$, and the key bit k.

Then $\mathcal{S}$ answers each secret-key query on an identity id that is not a prefix of (or equal to) id^* as follows:

- If $t = |id| \leq t^*$, then let $i \geq 1$ be the first position at which $id_i \neq id_i^*$. Answer the query with $(\mathbf{S}_{id}, \mathbf{e}_{id})$, which are computed by

$$\begin{aligned} \mathbf{S}_{id} &\leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{S}_i, \mathbf{A}_{id}), s_t) \\ \mathbf{e}_{id} &\leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_i, \mathbf{A}_{id}), \mathbf{y}_{id}, s_t). \end{aligned}$$

- If $t = |id| > t^*$, answer the query $(\mathbf{S}_{id}, \mathbf{e}_{id})$, which are computed by

$$\begin{aligned} \mathbf{S}_{id} &\leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{S}_{t^*+1}^{(id_{t^*+1})}, \mathbf{A}_{id}), s_t) \\ \mathbf{e}_{id} &\leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_{t^*+1}^{(id_{t^*+1})}, \mathbf{A}_{id}), \mathbf{y}_{id}, s_t). \end{aligned}$$

Finally, $\mathcal{S}$ outputs whatever bit $\mathcal{A}$ outputs.

We now analyze the reduction. First, observe that the master public key given to $\mathcal{A}$ is negligibly close to uniform (hence properly distributed), by hypothesis on KEM and by Proposition 3.1. Next, one can check that secret-key queries are distributed as in the real attack (to within $\mathrm{negl}(n)$ statistical distance), by Lemma 3.3 (note that the Gram-Schmidt vectors of each basis $\mathbf{S}_i, \mathbf{S}_{t^*+1}^{(b)}$ are sufficiently short to invoke RandBasis and SampleD). Finally, the encapsulation $(\mathbf{b}^*, p)$ (for identity id^*) and key bit k are distributed as in the real attack, by the truncation property of KEM. Therefore, $\mathcal{S}$'s overall advantage is within $\mathrm{negl}(n)$ of $\mathcal{A}$'s advantage, as desired. □

5.3 Full security in the Random Oracle Model

To obtain a fully secure HIBE in the random oracle model we can use a generic transformation by Boneh and Boyen [8]. It starts from a selective-id secure HIBE and applies hash functions to the identities. The resulting HIBE is fully secure, in the random oracle model, losing roughly a factor of Q_H^d in security, where Q_H is the number of random oracle queries. Furthermore, the $\{\mathbf{A}_j^{(b)}\}$ component of the master public key may be omitted, because each $\mathbf{A}_{id}$ can instead be constructed by querying the random oracle on, say, each prefix of the identity id.

We now give a more efficient fully-secure HIBE scheme, ROHIBE, in the random oracle model. It can be seen as a generalization of the GPV IBE scheme [25]. Compared to the fully-secure scheme obtained by the generic transformation, the efficiency improvement stems from the fact that $\mathbf{y}$ from pk_{id} now also depends on the identity id (via a hash function G). This way the dimension of the lattice associated to id can be decreased. The scheme is again parametrized by a dimension $m = O(n \lg q)$ and the following parameters. For an identity at depth $i \geq 1$,

- $i \cdot m$ is the dimension of a lattice associated with the identity;
- $\widetilde{L_i}$ is an upper bound on the Gram-Schmidt lengths of its secret short basis;
- for $i \geq 1$, s_i is the Gaussian parameter used to generate that secret basis, which must exceed $\widetilde{L_j} \cdot \omega(\sqrt{\log n})$ for all $j < i$.

These parameters, along with the total depth d of the hierarchy, determine the modulus q and error distribution χ used in the cryptosystem. As before, we instantiate all the parameters after describing the scheme. Let $H : \{0,1\}^* \to \mathbb{Z}_q^{n \times m}$ and $G : \{0,1\}^* \to \mathbb{Z}_q^n$ be hash functions.

- ROHIBE.Setup(d): This is the same as BTE.Setup(d), except that we only generate $\mathbf{A}_0$ and $\mathbf{S}_0$. More precisely, using Proposition 3.1, select $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ that is (negligibly close to) uniform and a basis $\mathbf{S}_0$ of $\Lambda^\perp(\mathbf{A}_0)$ such that $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L_0}$. Output $mpk = (\mathbf{A}_0, d)$ and $msk = \mathbf{S}_0$.

  All the remaining algorithms implicitly take the master public key mpk as an input. For an identity vector id of length $t \leq d$, we let

  $$\mathbf{A}_{id} = \mathbf{A}_0 \| \mathbf{A}_1 \| \cdots \| \mathbf{A}_{t-1} \in \mathbb{Z}_q^{n \times tm}, \quad \mathbf{y}_{id} = G(id) \in \mathbb{Z}_q^n,$$

  where $\mathbf{A}_i = H(id_1, \ldots, id_i) \in \mathbb{Z}_q^{n \times m}$. We let $pk_{id} = (\mathbf{A}_{id}, \mathbf{y}_{id})$ denote the KEM public key associated with identity vector id.

- $\mathsf{ROHIBE.Extract}(\mathbf{S}_{id}, id' = id\|\bar{id})$: if $t' = |id'| > d$, output $\perp$. Else, let $t = |id|$ and $\bar{t} = |\bar{id}|$, and choose
$$\mathbf{S}_{id'} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{S}_{id}, \mathbf{A}_{id'}), s_{t'}).$$
Sample $\mathbf{e}_{id'} \leftarrow D_{\Lambda^{\perp}_{\mathbf{y}_{id'}}(\mathbf{A}_{id'}), s_{t'}}$ using $\mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_{id}, \mathbf{A}_{id'}), \mathbf{y}_{id'}, s_{t'})$ and output $sk_{id'} = (\mathbf{S}_{id'}, \mathbf{e}_{id'})$.
For technical reasons, we assume that the same $\mathbf{e}_{id'}$ is drawn every time this identity is used. This means that the actual algorithm should be stateless or use standard techniques like PRFs to get repeated randomness.
- $\mathsf{ROHIBE.Encaps}(id)$: Output $(k, C) \leftarrow \mathsf{KEM.Encaps}(pk_{id})$.
- $\mathsf{ROHIBE.Decaps}(sk_{id} = (\mathbf{S}_{id}, \mathbf{e}_{id}), C)$: Output $k \leftarrow \mathsf{KEM.Decaps}(\mathbf{e}_{id}, C)$.

Instantiating the parameters. A similar computation as in the last subsection shows that we can set
$$\widetilde{L_t} = \widetilde{L_0} \cdot O(\sqrt{d \cdot n \lg q})^{t-1} \cdot \omega(\sqrt{\log n})^{t-1},$$
with $\widetilde{L_0} = O(\sqrt{n \lg q})$. To ensure that the underlying KEM is complete (Lemma 5.1), we let $q \geq 4s_d \cdot (d+1)m$ and $1/\alpha = s_d \cdot \sqrt{(d+1)m} \cdot \omega(\sqrt{\log n})$. For any $d = \mathrm{poly}(n)$, invoking the worst-case to average-case reduction for LWE yields an underlying approximation factor of $n \cdot \tilde{O}(d \cdot \sqrt{n})^d$.

Theorem 5.3 (Security of ROHIBE)**.** *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking KEM (instantiated with dimension dm and q, χ as in ROHIBE) such that, for any adversary $\mathcal{A}$ mounting an aid-ind-cpa attack on ROHIBE making Q_H queries to the random oracle H and Q_G queries to the random oracle G,*
$$\mathbf{Adv}_{KEM}(\mathcal{S}^{\mathcal{A}}) \geq \mathbf{Adv}^{\textit{aid-ind-cpa}}_{\textit{ROHIBE}}(\mathcal{A})/(dQ_H^{d-1}Q_G) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{A}$ be an adversary mounting a aid-ind-cpa-attack on ROHIBE. We construct a reduction $\mathcal{S}$ attacking KEM. It is given a uniformly random public key $pk = (\mathbf{A}, \mathbf{y}) \in \mathbb{Z}_q^{n \times dm} \times \mathbb{Z}_q^n$, an encapsulation $(\mathbf{b}, p) \in \mathbb{Z}_q^{dm} \times \mathbb{Z}_q$, and a bit k which either is encapsulated by $(\mathbf{b}, p)$ or is uniform and independent; the goal of $\mathcal{S}$ is to determine which is the case.

Let Q_G and Q_H be the number or queries that $\mathcal{A}$ issues to H and G, respectively. In our analysis, we will actually be more generous and let the adversary issue at most $d \cdot Q_H$ total queries, where it is allowed Q_H queries to H at each input length. To simplify the analysis, we also assume without loss of generality that (1) whenever $\mathcal{A}$ queries $H(id_1, \ldots, id_i)$, it has already issued the queries $H(id_1, \ldots, id_j)$ for $j < i$, and (2) that when $\mathcal{A}$ asks for sk_{id}, it has already queried $H(id)$ and $G(id)$.

$\mathcal{S}$ simulates the attack on ROHIBE to $\mathcal{A}$ as follows. First, $\mathcal{S}$ produces a master public key mpk, encapsulated key, and some secret internal state as follows:

- *Guess length of challenge identity and random oracle queries.* Choose $t^* \leftarrow [d]$, a guess for the length of the challenge identity. Pick a vector $\mathbf{j}^* = (j_1^*, \ldots, j_{t^*-1}^*) \leftarrow \{1, \ldots, Q_H\}^{t^*-1}$ and index $j \leftarrow \{1, \ldots, Q_G\}$.
- *Parsing the KEM inputs.* Parse $\mathbf{A}$ as $\mathbf{A} = \mathbf{A}_0\|\mathbf{A}_1\|\cdots\|\mathbf{A}_{d-1} \in \mathbb{Z}_q^{n \times dm}$ for $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ for all $i \in [d-1]$. Similarly, truncate $\mathbf{b}$ to $\mathbf{b}^* \in \mathbb{Z}_q^{t^* m}$.

$\mathcal{S}$ gives to $\mathcal{A}$ the master public key $mpk = (\mathbf{A}_0, d)$. To simulate the attack game for $\mathcal{A}$, $\mathcal{S}$ must simulate oracle queries to H and G, queries for user secret keys, and it must also generate the challenge encapsulation. To do this, it will maintain two lists, denoted $\mathcal{H}$ and $\mathcal{G}$, which are initialized to be empty and will store tuples of values. $\mathcal{S}$ processes queries as follows.

Queries to $H(\cdot)$. On $\mathcal{A}$'s j_i-th distinct query $(id_{j_i,1}, \ldots, id_{j_i,i})$ to $H(\cdot)$ of length i, do the following: if $i \leq t^* - 1$ and $j_i = j_i^*$, then return $\mathbf{A}_i$ (i.e., this branch undergoes *undirected growth*). Otherwise, if $i \geq t^*$ or $j_i \neq j_i^*$, run $\mathsf{GenBasis}(1^n, 1^m, q)$ to generate $\mathbf{A}_{i,j_i} \in \mathbb{Z}_q^{n \times m}$ with corresponding short basis $\mathbf{S}_{i,j_i}$ (i.e., this branch undergoes *controlled growth*). Store the tuple $((id_{j_i,1}, \ldots, id_{j_i,i}), \mathbf{A}_{i,j_i}, \mathbf{S}_{i,j_i})$ in list $\mathcal{H}$, and return $\mathbf{A}_{i,j_i}$.

Queries to $G(\cdot)$. On $\mathcal{A}$'s j-th distinct query id_j to $G(\cdot)$, do the following: if $j = j^*$ then return $\mathbf{y}$. (Recall that $\mathbf{y}$ was obtained from the KEM input.) Otherwise for $j \neq j^*$, sample $\mathbf{e}_j \leftarrow D_{\Lambda^\perp(\mathbb{Z}^m), s_t}$ (where t is the depth of id_j) and set $\mathbf{y}_j := \mathbf{A}_{(id_{j,1}, \ldots, id_{j,t-1})} \mathbf{e}_j \in \mathbb{Z}_q^n$. (Recall that we assumed $\mathcal{A}$ has already made all relevant queries to H that in particular define $\mathbf{A}_{(id_{j,1}, \ldots, id_{j,t-1})} = H(id_{j,1}, \ldots, id_{j,t-1})$.) Return $\mathbf{y}_j$ and store $(id_j, \mathbf{y}_j, \mathbf{e}_j)$ in list $\mathcal{G}$.

Queries to $\mathsf{ROHIBE.Extract}$. When $\mathcal{A}$ asks for a user secret key for $id = (id_1, \ldots, id_t)$, we again assume that $\mathcal{A}$ has already made all relevant queries to G and H that define $\mathbf{y}_{id}$ and $\mathbf{A}_{id}$. If, for one $i \in [t-1]$, $\mathbf{A}_{id_i} = H(id_1, \ldots, id_i)$ is contained in list $\mathcal{H}$, then $\mathcal{B}$ can compute a properly distributed short basis $\mathbf{S}_{id}$ for $\mathbf{A}_{id}$ by running $\mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{S}_{i,id_i}, \mathbf{A}_{id}), s_t)$, where $\mathbf{S}_{i,id_i}$ is obtained from $\mathcal{H}$. If $\mathbf{y}_{id}$ is contained in list $\mathcal{G}$, then $\mathcal{B}$ can retrieve a properly distributed vector $\mathbf{e}_{id}$ from $\mathcal{G}$ satisfying $\mathbf{A}_{(id_1, \ldots, id_{t-1})} \mathbf{e}_{id} = \mathbf{y}_{id}$. If the generation of $sk_{id} = (\mathbf{B}_{id}, \mathbf{e}_{id})$ was successful, then $\mathcal{B}$ returns sk_{id}. In all other cases, $\mathcal{B}$ aborts (and returns a random bit).

Challenge query for id^*. Let t be the depth of id^*. If $t \neq t^*$, or $G(id^*) \neq \mathbf{y}$, or $H(id_1^*, \ldots, id_i^*) \neq \mathbf{A}_i$ (for one $1 \leq i \leq t^* - 1$), then abort. Otherwise, return the encapsulation $(\mathbf{b}^*, p)$, and the key-bit k.

$\mathcal{S}$ runs until $\mathcal{A}$ halts, and it outputs whatever bit $\mathcal{A}$ outputs.

It remains to analyze the reduction. The master public key given to $\mathcal{A}$ is negligibly close to uniform by the construction of KEM and Proposition 3.1. By the same proposition, we have that oracle queries to H are properly simulated, up to negligible statistical distance. Oracle responses for G are negligibly far from uniform by Lemma 2.4 (4). It can also be verified using the truncation property of KEM that the challenge encapsulation is properly distributed, conditioned on the event that $\mathcal{S}$ does not abort. Thus all that remains to check is the probability that $\mathcal{S}$ aborts; this is at most $1/(dQ_G Q_H^{d-1})$. This completes the proof.

□

5.4 Full Security in the Standard Model

To achieve aid-ind-cca security in the standard model, we will essentially try to implement the random oracle from our scheme ROHIBE with a suitable hash function. We will employ a probabilistic argument, along the lines of [9]. Concretely, we will set up the public key such that in the simulation, we will know a short basis for $\mathbf{A}_{id}$ with a certain probability. A sophisticated construction of the hash function will ensure that, to a certain degree, these probabilities (resp. the corresponding events) are independent. That is, even an adversary that adaptively asks for user secret keys will not manage to produce an identity id for which the simulation is *guaranteed* to know a trapdoor. In this case, a successful simulation will be possible.

Of course, we will have to take care that the event of a successful simulation is (at least approximately) independent of the adversary's view. To achieve independence, we will employ an "artificial abort" strategy similar to the one from [56].

5.4.1 Admissible hash functions.

We give a variant of the definition from [9]. Let $\mathcal{H} = \{\mathcal{H}_n\}$ be a collection of distributions of functions $H : \mathcal{C}_n \to \mathcal{D}_n = \{0,1\}^\lambda$. For $H \in \mathcal{H}_n$, $K \in \{0,1,\perp\}^\lambda$, and $x \in \mathcal{C}_n$, define

$$F_{K,H}(x) = \begin{cases} \mathtt{B} & \text{if } \exists u \in \{1,\ldots,\lambda\} : t_u = K_i \\ \mathtt{R} & \text{if } \forall u \in \{1,\ldots,\lambda\} : t_u \neq K_i \end{cases} \quad \text{for } (t_1,\ldots,t_\lambda) = H(x).$$

For $\mu \in \{0,\ldots,\lambda\}$, denote by $\mathcal{K}_\mu$ the uniform distribution on all keys $K \in \{0,1,\perp\}^\lambda$ with exactly μ non-$\perp$ components.

We say that $\mathcal{H}$ is Δ-*admissible* (for $\Delta : \mathbb{N}^2 \to \mathbb{R}$) if for every polynomial $Q = Q(n)$, there exists an efficiently computable function $\mu = \mu(n)$, and efficiently recognizable sets $\mathsf{bad}_H \subseteq (\mathcal{C}_n)^*$ ($H \in \mathcal{H}_n$), so that the following holds:

- For every PPT algorithm $\mathcal{C}$ that, on input a function $H \in \mathcal{H}_n$, outputs a vector $\mathbf{x} \in \mathcal{C}_n^{Q+1}$, the function

  $$\mathbf{Adv}^{adm}_{\mathcal{H}}(\mathcal{C}) := \Pr[\mathbf{x} \in \mathsf{bad}_H \mid H \leftarrow \mathcal{H}_n \; ; \; \mathbf{x} \leftarrow \mathcal{C}(H)]$$

  is negligible in n.

- For every $H \in \mathcal{H}_n$ and every $\mathbf{x} = (x_0,\ldots,x_Q) \in \mathcal{C}_n^{Q+1} \setminus \mathsf{bad}_H$, we have that

  $$\Pr[F_{K,H}(x_0) = \mathtt{R} \wedge F_{K,H}(x_1) = \cdots = F_{K,H}(x_Q) = \mathtt{B}] \geq \Delta(n,Q),$$

  where the probability is over uniform $K \in \mathcal{K}_{\mu(n,Q)}$.

We say that $\mathcal{H}$ is *admissible* if $\mathcal{H}$ is admissible for some Δ, such that $\Delta(n,Q)$ is significant for every polynomial $Q = Q(n)$.

Difference to the definition of [9]. Note that our definition of admissibility is conceptually different from that of [9]. The reason for our change is that our definition is better suited for our purposes. Concretely, their definition is based upon indistinguishability from a (biased) random function. However, their construction only achieves asymptotic indistinguishability (i.e., negligible distinguishing success) when the "target" random function is constant. (In their notation, this corresponds to the case when γ is negligible, so that $\Pr[F_{K,H}(x) = 1] = 1 - \mathrm{negl}(n)$.) Such a function is not very useful for aymptotic purposes. In an asymptotic sense, their construction becomes only useful with parameters that cause the distinguishing advantage to become significant (but smaller than the inverse of a given polynomial). With that parameter choice, our definition allows for a conceptually simpler analysis. Namely, it separates certain negligible error probabilities (of $\mathbf{x} \in \mathsf{bad}_H$) from significant, but purely combinatorial bounds on the probability of the "simulation-enabling" event

$$\mathsf{good} := [F_{K,H}(x_0) = \mathtt{R} \wedge F_{K,H}(x_1) = \cdots = F_{K,H}(x_Q) = \mathtt{B}].$$

Specifically, we can bound $\Pr[\mathsf{good}]$ for *every* $\mathbf{x} \notin \mathsf{bad}_H$, which simplifies the artificial abort step below. Note that while the original analysis from [9] does not incorporate an artificial abort step, this actually would have been necessary to guarantee sufficient independence of (their version of) event good. This becomes an issue in [9, Claim 2], when the success probability of an adversary conditioned on good is related to its original (unconditioned) success probability.

Constructions. [9] show how to construct admissible hash functions from a given collision-resistant hash function family. Since collision-resistant hash functions can be built from the LWE problem (e.g., [5]), this does not entail extra assumptions in the encryption context. Specifically, for parameter choices as in [9, Section 5.3], we get a single hash function with output length $\lambda = O(k^{2+\varepsilon})$ (for arbitrary $\varepsilon > 0$) that is Δ-admissible with $\Delta = \Theta(1/Q^2)$.[5]

5.4.2 The scheme SMHIBE.

Let $d \in \mathbb{N}$ denote the maximal depth of the HIBE, and fix a dimension m, as well as $\widetilde{L}_i$, s_i. Let $\mathcal{H} = (\mathcal{H}_n)_n$ be an admissible family of hash functions $H : \{0,1\}^n \to \{0,1\}^\lambda$.

SMHIBE.Setup(d) Using Proposition 3.1, generate $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a corresponding short basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ with $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L_0}$. Furthermore, sample uniformly and independently matrices $\mathbf{B}_{i,u,b} \in \mathbb{Z}_q^{n \times m}$ (for $1 \leq i \leq d, 1 \leq u \leq \lambda$ and $0 \leq b \leq 1$) and a vector $\mathbf{y} \in \mathbb{Z}_q^n$. Finally, choose $H_1, \ldots, H_d \leftarrow \mathcal{H}_n$. Return

$$mpk = (\mathbf{A}, \mathbf{y}, (\mathbf{B}_{i,u,b})_{(i,u,b) \in [d] \times [\lambda] \times \{0,1\}}, (H_i)_{i=1}^d), \qquad msk = (mpk, \mathbf{S}).$$

For an identity $id = (id_1, \ldots, id_\ell)$ we define

$$\begin{aligned} \mathbf{A}_{id} &:= \mathbf{A} || \mathbf{A}_{1,id_1} || \ldots || \mathbf{A}_{\ell,id_\ell} \in \mathbb{Z}_q^{n \times (\lambda \ell + 1)m} \\ \text{for} \quad \mathbf{A}_{i,id_i} &:= \mathbf{B}_{i,1,t_1} || \ldots || \mathbf{B}_{i,\lambda,t_\lambda} \in \mathbb{Z}_q^{n \times \lambda m}, \end{aligned} \tag{5.1}$$

where $(t_1, \ldots, t_\lambda) := H_i(id_i) \in \{0,1\}^\lambda$. The user secret keys for an identity id will consist of a basis part $\mathbf{S}_{id}$ for $\Lambda^\perp(\mathbf{A}_{id})$ and a syndrome part $\mathbf{e}_{id}$ satisfying $\mathbf{A}_{id}\mathbf{e}_{id} = \mathbf{y}$. For brevity, we will write $id|\ell := (id_1, \ldots, id_\ell)$ for $\ell \leq |id|$.

SMHIBE.Extract(msk, id): This algorithm computes a user secret key $(\mathbf{S}_{id}, \mathbf{e}_{id})$ for $id = (id_1, \ldots, id_\ell)$, where $\mathbf{S}_{id} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{A}_{id}, \mathbf{S}_\varepsilon), s_\ell)$ is a basis for $\Lambda^\perp(\mathbf{A}_{id})$ and $\mathbf{e}_{id} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{A}_{id}, \mathbf{S}_\varepsilon), \mathbf{y}_{id}, s$ is distributed according to $D_{\mathbb{Z}^{(\lambda\ell+1)m}, s_\ell}$ conditioned on $\mathbf{A}_{id}\mathbf{e}_{id} = \mathbf{y}_{id}$.

SMHIBE.HIBEDel$(usk_{id|\ell-1}, id)$: The delegation algorithm derives a user secret key for an identity $id = (id_1, \ldots, id_\ell)$ $(1 \leq \ell \leq d)$ given a user secret key for $id|\ell - 1$ which contains a basis $\mathbf{S}_{id|\ell-1}$ for $\Lambda^\perp(\mathbf{A}_{id|\ell-1})$ with $\|\widetilde{\mathbf{S}}_{id|\ell-1}\| \leq L(\ell - 1)$. (We note that the short vector $\mathbf{e}_{id|\ell-1}$ is not needed for delegation.) Note that $\mathbf{A}_{id} = \mathbf{A}||\mathbf{A}_{1,id_1}|| \cdots ||\mathbf{A}_{\ell,id_\ell} = \mathbf{A}_{1,id|\ell-1}||\mathbf{A}_{\ell,id_\ell} \in \mathbb{Z}_q^{n \times (\lambda\ell+1)m}$. To compute the basis part, run $\mathbf{S}_{id} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{A}_{id}, \mathbf{S}_{id|\ell-1}), s_\ell)$. Note that since ℓ is constant,

$$\begin{aligned} L(\ell) = L(\ell - 1) \cdot \sqrt{\lambda m} \cdot \omega(\sqrt{\log \lambda m}) \\ \geq \|\widetilde{\mathbf{S}}_{id|\ell-1}\| \cdot \sqrt{(\lambda\ell + 1)m} \cdot \omega(\sqrt{\log{(\lambda\ell + 1)m}}). \end{aligned}$$

The syndrome part of the user secret key is computed as

$$\mathbf{e}_{id} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{A}_{id}, \mathbf{S}_{id|\ell-1}), \mathbf{y}, s_\ell).$$

By Lemma 3.3, the user secret key $usk_{id} = (\mathbf{S}_{id}, \mathbf{e}_{id})$ has a distribution that is statistically close to the one computed by Extract.

[5] In the notation of [9], we replace the output length β_H of the original hash function with k, and bound the number Q of hash function queries by $2^{k^{\varepsilon/2}}$. Note that Q will later correspond to the number of (online) user secret key queries, so we bound Q by a comparatively small exponential function.

$\mathsf{SMHIBE.Encaps}(id, b)$: Output $C = (k, \mathbf{p}) \leftarrow \mathsf{KEM.Encaps}(pk = (\mathbf{A}_{id}, \mathbf{y}))$.
$\mathsf{SMHIBE.Decaps}(sk_{id}, (\mathbf{S}_{id}, \mathbf{e}_{id}), C)$: Output $k \leftarrow \mathsf{KEM.Decaps}(\mathbf{e}_{id}, C)$.
The scheme's correctness is inherited by that of KEM.

5.4.3 Security of SMHIBE.

We now formally state security of our construction. If the hash function $\mathcal{H}$ is admissible, then we can prove the scheme aid-ind-cpa secure. Unfortunately, we only know constructions of admissible hash functions that require $\lambda = n^{2+\varepsilon}$ so the resulting scheme is not very practical.

Theorem 5.4. *Assume an adversary $\mathcal{A}$ on* SM-HIBE*'s aid-ind-cpa security that makes at most $Q(n)$ user secret key queries. Then, for every polynomial $S = S(n)$, there exists an $\mathsf{LWE}_{q,\chi}$-distinguisher $\mathcal{D}$ and an adversary $\mathcal{C}$ on $\mathcal{H}$'s admissibility such that*

$$\mathbf{Adv}^{\textit{aid-ind-cpa}}_{\text{SM-HIBE}}(\mathcal{A}) \leq d \cdot \mathbf{Adv}^{adm}_{\mathcal{H}}(\mathcal{C}) + \frac{\mathbf{Adv}_{\mathsf{LWE}_{q,\chi}}(\mathcal{D})}{\Delta(n,Q)^d} + \frac{1}{S(n)} + \mathrm{negl}(n). \tag{5.2}$$

Here, the running time of $\mathcal{C}$ is roughly that of the aid-ind-cpa experiment with $\mathcal{A}$, and the running time of $\mathcal{D}$ is roughly that of the aid-ind-cpa experiment with $\mathcal{A}$ plus $O(n^2QS/\Delta^d)$ steps.

Note that for the admissible hash function from [9], $\Delta(n,Q)^d = \Theta(1/Q^{2d})$ is significant. Since S in Theorem 5.4 is arbitrary, we obtain:

Corollary 5.5 (SM-HIBE is aid-ind-cpa secure)**.** *If $\mathcal{H}$ is admissible, and if the $\mathsf{LWE}_{q,\chi}$ problem is hard, then* SM-HIBE *is CPA secure.*

5.4.4 Proof of Theorem 5.4.

We proceed in games, with **Game** 0 being the original aid-ind-cpa experiment with adversary $\mathcal{A}$. We assume without loss of generality that $\mathcal{A}$ always makes exactly $Q = Q(n)$ user secret key queries. We denote these queries by by $id^j = (id^j_1, \ldots, id^j_{\ell_j})$ (for $1 \leq j \leq Q$), and the challenge identity chosen by $\mathcal{A}$ as $id^* = (id^*_1, \ldots, id^*_{\ell^*})$. By out_i, we denote the experiment's output in Game i. By definition,

$$|\Pr[out_0 = 1] - 1/2| = \mathbf{Adv}^{\text{aid-ind-cpa}}_{\text{SM-HIBE}}(\mathcal{A}). \tag{5.3}$$

In the following, let $\mathcal{ID}^Q_i := \bigcup_j \{id^j_i\}$ be the set of all level-i identities contained in user secret key queries. Let $\mathcal{ID}^*_i := \{id^*_i\}$ be the level-i challenge identity (or the empty set if $\ell^* < i$). Note that $1 \leq |\mathcal{ID}^Q_i| \leq Q$ and $0 \leq |\mathcal{ID}^*_i| \leq 1$. For our upcoming probabilistic argument we need actually identity sets of a fixed size, so we pad all $\mathcal{ID}^Q_i$ and $\mathcal{ID}^*_i$ in some canonical way with unused identities. Concretely, for all i, let $\mathcal{ID}^{\mathtt{R}}_i \supseteq \mathcal{ID}^*_i$ and $\mathcal{ID}^{\mathtt{B}}_i \supseteq \mathcal{ID}^Q_i \setminus \mathcal{ID}^*_i$ be disjoint, and such that $|\mathcal{ID}^{\mathtt{R}}_i| = 1$ and $|\mathcal{ID}^{\mathtt{B}}_i| = Q$. Let $\overrightarrow{\mathcal{ID}_i} \in (\{0,1\}^n)^{Q+1}$ be the vector whose first component is the (unique) element of $\mathcal{ID}^{\mathtt{R}}_i$, and whose remaining Q components are the elements of $\mathcal{ID}^{\mathtt{B}}_i$ (in some canonical order).

In **Game** 1, we eliminate the "bad $\mathcal{H}$-queries." Namely, Game 1 aborts (and outputs a uniform bit) whenever $\overrightarrow{\mathcal{ID}_i} \in \mathsf{bad}_{H_i}$ for some i. A straightforward reduction shows

$$|\Pr[out_1 = 1] - \Pr[out_0]| \leq d \cdot \mathbf{Adv}^{adm}_{\mathcal{H}}(\mathcal{C}).$$

for a suitable adversary $\mathcal{C}$ on $\mathcal{H}$'s admissibility.

In **Game** 2, after the adversary has terminated, we throw an event good_2 independently with probability Δ^d. We abort the experiment (and output a uniformly random bit) if $\neg\mathsf{good}_2$ occurs. We get

$$\Pr[out_2 = 1] - 1/2 = \Pr[\mathsf{good}_2]\,(\Pr[out_1 = 1] - 1/2) = \Delta^d\,(\Pr[out_1 = 1] - 1/2)\,.$$

In **Game** 3, we change the abort policy. Namely, after the adversary has terminated, we first attach colors to the identities in all $\overrightarrow{\mathcal{ID}_i}$. To this end, let

$$F_i(id) = F_{K^i,H_i}(id) = \begin{cases} \mathtt{B} & \text{if } \exists u \in \{1,\ldots,\lambda\} : t_u = K^i_u \\ \mathtt{R} & \text{if } \forall u \in \{1,\ldots,\lambda\} : t_u \neq K^i_u \end{cases}$$

for $(t_1,\ldots,t_\lambda) = H_i(id)$. Here, for every i, $K^i = (K^i_1,\ldots,K^i_\lambda) \in \{0,1,\bot\}^\lambda$ is initially chosen by the experiment uniformly among all $K \in \{0,1,\bot\}^\lambda$ with exactly μ non-$\bot$ components. If $F_i(id) = \mathtt{B}$, then id is *blue*, and if $F_i(id) = \mathtt{R}$, then id is *red*. Later on, blue will correspond to trapdoor matrices, while red will correspond to challenge matrices.

Let E_i denote the event that in Game 3, $F_i(id) = \mathtt{B}$ for all $id \in \mathcal{ID}_i^{\mathtt{B}}$ and $F_i(id) = \mathtt{R}$ for all $id \in \mathcal{ID}_i^{\mathtt{R}}$. Let $E := \bigwedge_{i=1}^d E_i$. By assumption about $\mathcal{H}$, we know that $\Pr[E] \geq \Delta^d$.

Ideally, we would like to replace event good_2 from Game 2 with event E. Unfortunately, however, E might not be independent of $\mathcal{A}$'s view, so we cannot (directly) proceed that way. Instead, we use artificial abort techniques. That is, given the identities in all $\overrightarrow{\mathcal{ID}_i}$, we approximate $p_E := \Pr\left[E \mid (\overrightarrow{\mathcal{ID}_i})_i\right]$ by sufficiently often sampling values of K and attaching colors. Hoeffding's inequality yields that with $\lceil nS/\Delta^d \rceil$ samples, we can obtain an approximation $\tilde{p_E} \geq \Delta^d$ of p_E that satisfies

$$\Pr\left[|p_E - \tilde{p_E}| \geq \frac{\Delta^d}{S}\right] \leq \frac{1}{2^n}.$$

Now we finally abort if E does not occur. But even if E occurs (which might be with probability $p_E > \Delta^d$), we artificially enforce an abort with probability $1 - \Delta^d/\tilde{p_E}$. Call good_3 the event that we do not abort. We always have

$$\Pr[\mathsf{good}_3] = p_E \cdot \frac{\Delta^d}{\tilde{p_E}} = \Delta^d \frac{p_E}{\tilde{p_E}}.$$

Hence, except with probability $1/2^n$,

$$|\Pr[\mathsf{good}_3] - \Pr[\mathsf{good}_2]| = \left|\Delta^d - \Delta^d \frac{p_E}{\tilde{p_E}}\right| = \Delta^d \left|\frac{\tilde{p_E} - p_E}{\tilde{p_E}}\right| \leq \Delta^d \frac{\Delta^d}{S\tilde{p_E}} \leq \frac{\Delta^d}{S}. \tag{5.4}$$

Since (5.4) holds for arbitrary $\overrightarrow{\mathcal{ID}_i}$ except with probability $1/2^n$, we obtain that the statistical distance between the output of Game 2 and Game 3 is bounded by $\Delta^d/S + 2^{-n}$. Hence,

$$|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leq \frac{\Delta^d}{S} + \frac{1}{2^n}. \tag{5.5}$$

In **Game** 4, we set up the public key differently. We call matrices that are chosen uniformly *undirected,* and matrices that are chosen along with a short basis (using Proposition 3.1) *controlled.* Now in Game 4, we will set up the public key as follows:

- $\mathbf{A}$ as controlled (as in the earlier games),

- $\mathbf{B}_{i,u,b}$ as controlled if $K_u^i = b$ (and as undirected if $K_u^i \neq b$).

By Proposition 3.1, this change affects the distribution of the public key only negligibly. (Note that bases for the controlled $\mathbf{B}_{i,u,b}$ are generated, but never used in Game 4.) We obtain

$$|\Pr[out_4 = 1] - \Pr[out_3 = 1]| = \mathrm{negl}(n). \tag{5.6}$$

In **Game** 5, we make the following conceptual change regarding user secret key queries. Namely, upon receiving a user secret key request for $id = (id_1, \ldots, id_\ell)$, the experiment immediately aborts (with uniform output) if $F_i(id_i) = \mathtt{R}$ for all i. This change is purely conceptual: since id is not a prefix of the challenge identity id^*, there is an i with $id_i \in \mathcal{ID}_i^{\mathtt{B}} \supseteq \mathcal{ID}_i^Q \setminus \mathcal{ID}_i^*$. But since $F_i(id_i) = \mathtt{R}$, event E cannot occur, so the experiment from Game 5 would eventually abort as well. We get

$$\Pr[out_5 = 1] = \Pr[out_4 = 1]. \tag{5.7}$$

In **Game** 6, we change the way user secret keys queries $id = (id_1, \ldots, id_\ell)$ are answered. By the change from Game 5, we may assume that $F_i(id_i) = \mathtt{B}$ for some i. Hence, $t_u = K_u^i$ for $(t_1, \ldots, t_\lambda) = H_i(id_i)$ and some u. Thus, the matrix $\mathbf{B}_{i,u,t_u}$ that appears in the decomposition of $\mathbf{A}_{id}$ (see (5.1)) is controlled by our public key setup. To generate a user secret key, we have to find a short basis for $\mathbf{A}_{id}$. In Game 4, this is achieved by algorithms $\mathsf{ExtBasis}$ and $\mathsf{RandBasis}$, using the short basis of the first matrix $\mathbf{A}$ in the decomposition of $\mathbf{A}_{id}$. In Game 6, we instead use the short basis of $\mathbf{B}_{i,u,t_u}$ that we have initially generated. By Lemma 3.3, this results in the same distribution of bases $usk_{id} = (\mathbf{S}_{id}, \mathbf{e}_{id})$, up to negligible statistical distance. Hence

$$|\Pr[out_6 = 1] - \Pr[out_5 = 1]| = \mathrm{negl}(n). \tag{5.8}$$

In **Game** 7, we set up $\mathbf{A}$ as undirected instead of controlled. (Note that since Game 6, we do not need a short basis of $\mathbf{A}$ anymore to generate user secret keys.) Again, by Proposition 3.1, this change affects the distribution of the public key only negligibly. We get

$$|\Pr[out_7 = 1] - \Pr[out_6 = 1]| = \mathrm{negl}(n). \tag{5.9}$$

In **Game** 8, we generate the challenge ciphertext $(\mathbf{p}^*, c^*) \in \mathbb{Z}_q^{(\ell\lambda+1)m} \times \mathbb{Z}_q$ uniformly at random. Obviously, $\mathcal{A}$'s view is then independent of the challenge bit b, so

$$\Pr[out_8 = 1] = 1/2. \tag{5.10}$$

We furthermore claim that

$$|\Pr[out_8 = 1] - \Pr[out_7 = 1]| \leq \mathbf{Adv}_{\mathsf{LWE}_{q,\chi}}(\mathcal{D}) \tag{5.11}$$

for the following LWE distinguisher $\mathcal{D}$. Recall that $\mathcal{D}$ has access to either

- a "real" oracle that gives out samples $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + x)$ for a fixed $\mathbf{s} \in \mathbb{Z}_q^n$, uniform $\mathbf{a} \in \mathbb{Z}_q^n$, and an error x sampled from χ, or
- a "random" oracle that gives out samples $(\mathbf{a}, r)$ for uniform $\mathbf{a} \in \mathbb{Z}_q^n$ and $r \in \mathbb{Z}_q^n$.

$\mathcal{D}$ will simulate Game 7 and use its oracle to help set up parts of the public key and the challenge ciphertext. First, $\mathcal{D}$ samples $(\mathbf{y}, c_{\mathbf{y}})$ to obtain the $\mathbf{y}$ part of the public key. Then, $\mathcal{D}$ samples all undirected matrices in

the public key by querying its oracle m times respectively (and adjoining the outputs). This way, $\mathcal{D}$ obtains $(\mathbf{A}, \mathbf{g}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ from the oracle such that

$$\mathbf{g} = \begin{cases} \mathbf{A}^t\mathbf{s} + \mathbf{x} & \text{if } \mathcal{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathcal{D} \text{ runs with random oracle.} \end{cases} \tag{5.12}$$

(Similarly for values $(\mathbf{B}_{i,u,b}, \mathbf{g}_{i,u,b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ obtained from the oracle for $b \neq K_u^i$.) $\mathcal{D}$ then simulates Game 7 with these sampled values of $\mathbf{y}$, $\mathbf{A}$, and $\mathbf{B}_{i,u,b}$ in place. Since the first part of the oracle's output is always uniformly random, this yields the same distribution as in Game 7, resp. Game 8.

The crucial change is the way $\mathcal{D}$ puts together the challenge ciphertext $(\mathbf{p}^*, c^*)$ for identity $id^* = (id_1^*, \ldots, id_{\ell^*}^*)$. Since otherwise the experiment aborts with uniform output, we may assume that E occurs. Hence, for all i, for $(t_1, \ldots, t_\lambda) = H_i(id_i^*)$ and all u, $K_u^i \neq t_u$. Thus, we can write

$$\mathbf{A}_{id^*} = \mathbf{A}||\mathbf{B}_{i_1,u_1,b_1}|| \ldots ||\mathbf{B}_{i_{\ell\lambda},u_{\ell\lambda},b_{\ell\lambda}},$$

as a concatenation solely of undirected matrices, so $\mathcal{D}$ knows the corresponding values $\mathbf{g}$, resp. $\mathbf{g}_{i_v,u_v,b_v}$. Now $\mathcal{D}$ sets up the challenge encryption of a message $b \in \{0,1\}$ as

$$\begin{aligned} \mathbf{p}^* &:= \mathbf{g} + \sum_{v=1}^{\ell\lambda} \mathbf{g}_{i_v,u_v,b_v} \overset{(5.12)}{=} \begin{cases} \mathbf{A}_{id^*}^t\mathbf{s} + \mathbf{x} & \text{if } \mathcal{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathcal{D} \text{ runs with random oracle,} \end{cases} \\ c^* &:= c_{\mathbf{y}} + b\lfloor\frac{q}{2}\rfloor \overset{(5.12)}{=} \begin{cases} \mathbf{y}^t\mathbf{s} + x + b\lfloor\frac{q}{2}\rfloor & \text{if } \mathcal{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathcal{D} \text{ runs with random oracle.} \end{cases} \end{aligned} \tag{5.13}$$

Finally, $\mathcal{D}$ outputs whatever the experiment outputs. By (5.13), $\mathcal{D}$'s outputs distribution is exactly that of Game 7, resp. Game 8 if it interacts with the real, resp. random oracle. (5.11) follows.

Taking (5.3-5.11) together shows (5.2).

Doing without artificial abort. The reason why we needed an artificial abort step in Game 3 is that a certain event E (that determines whether we can carry through the simulation) is not independent of $\mathcal{A}$'s view. In Game 3, we changed the abort policy to make good_3 (the event that we do not abort) sufficiently independent. (This strategy resembles Waters' strategy from [56].) Unfortunately, this results in a rather large computational overhead, since we need to approximate the probability $p_E = \Pr\left[E \mid (\overrightarrow{\mathcal{ID}_i})_i\right]$ on the fly. Observe that if we had better (i.e., tight lower *and* upper) bounds on p_E in the first place (e.g., $|p_E - \Delta^d| < \Delta^d/S$ always), we did not need this approximation/abort step at all, since (5.4) and hence (5.5) followed directly by these better bounds. The good news is that the analysis of $\mathcal{H}$ from [9] provides such better bounds for $\Pr[E]$, resp. p_E. The bad news is that this comes at a price: Using the analysis of [9], $|p_E - \Delta^d|/\Delta^d$ depends in an inversely polynomial way on Q, the number of $\mathcal{H}$ queries. Hence, to achieve $|p_E - \Delta^d| < \Delta^d/S$, we would need to consider arbitrary polynomial values of Q and adjust the $\mathcal{H}$ parameters accordingly. Of course, in an asymptotic sense, we already do consider arbitrary polynomial values of Q, because $\mathcal{A}$ may make up to Q user secret key queries. However, in a concrete sense, the number of (online) user secret key queries will be much lower than the inverse of $\mathcal{A}$'s distinguishing advantage. Hence, when considering concrete parameters, we can work with much smaller $\mathcal{H}$ parameters when implementing our artificial abort step, at the price of a worse reduction. This is why we decided for an artificial abort step.

Acknowledgments

We thank the anonymous Eurocrypt reviewers for their helpful comments, and for pointing out a small error in an earlier formulation of RandBasis.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptology*, 21(3):350–391, 2008. Preliminary version in CRYPTO 2005.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010. To appear.

[3] Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. Manuscript, July 2009.

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

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

[6] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[7] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582, 2001.

[8] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[9] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[10] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[11] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[12] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Preliminary version in CRYPTO 2001.

[13] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[14] Xavier Boyen. Of lettuces of lattices: a framework for short signatures and IBE with full security. In *Public Key Cryptography*, 2010. To appear.

[15] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *J. Cryptology*, 20(3):265–294, 2007. Preliminary version in EUROCRYPT 2003.

[17] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351, July 2009. `http://eprint.iacr.org/`.

[18] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[19] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000. Preliminary version in CCS 1999.

[20] Giovanni Di Crescenzo and Vishal Saraswat. Public key encryption with searchable keywords based on Jacobi symbols. In *INDOCRYPT*, pages 282–296, 2007.

[21] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *ACM Workshop on Digital Rights Management*, pages 61–80, 2002.

[22] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT*, pages 123–139, 1999.

[23] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[24] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *TCC*, pages 437–456, 2009.

[25] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[26] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[27] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131, 1997.

[28] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988. Preliminary version in FOCS 1984.

[29] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140, 2003.

[30] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[31] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *EUROCRYPT*, pages 333–350, 2009.

[32] Susan Hohenberger and Brent Waters. Short and stateless signatures from the rsa assumption. In *CRYPTO*, pages 654–670, 2009.

[33] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.

[34] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000.

[35] Gaëtan Leurent and Phong Q. Nguyen. How risky is the random-oracle model? In *CRYPTO*, pages 445–464, 2009.

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

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

[38] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010. To appear.

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

[40] Daniele Micciancio and Shafi 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.

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

[42] Daniele Micciancio and Bogdan Warinschi. A linear space algorithm for computing the Hermite normal form. In *ISSAC*, pages 231–236, 2001.

[43] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

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

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

[46] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. Manuscript, 2010.

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

[48] Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, pages 478–487, 2007.

[49] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[50] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[51] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. Preliminary version in STOC 2005.

[52] Markus Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. In *PQCrypto*, 2010. To appear.

[53] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[54] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *CRYPTO*, pages 355–367, 2001.

[55] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635, 2009.

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

[57] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[58] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM Conference on Computer and Communications Security*, pages 354–363, 2004.