Bonsai Trees
(or, Arboriculture in Lattice-Based Cryptography)

Chris Peikert*
Georgia Institute of Technology

July 20, 2009

Abstract

We introduce *bonsai trees*, a lattice-based cryptographic primitive that we apply 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 [GPV08], 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 [Sha84].)

Abstractly, the GPV schemes are structurally quite similar to Rabin/Rabin-Williams signatures [Rab79] (based on integer factorization) and the Cocks/Boneh-Gentry-Hamburg IBEs [Coc01, BGH07] (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., [LN09]), 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 starting from the work of Boneh and Franklin [BF03]. For example, the more general notion of *hierarchical* IBE [HL02, GS02] permits multiple levels of secret-key authorities. This model is more appropriate for large organizations, isolates damage in the case of secret-key exposure, and has further applications such as such as forward-secure encryption [CHK07] and broadcast encryption [DF02, YFDL04].

*Email: cpeikert@alum.mit.edu. This material is based upon work supported by the National Science Foundation under Grants CNS-0716786 and CNS-0749931. 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.

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: roughly speaking, 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 [GMR88, NY89] and specialized to lattice assumptions [LM08]), and in particular it does not require the signer to keep any state. (This a crucial property in many real-world scenarios, where distinct systems may sign for the same public key.) In our scheme, the signature length and verification time are a factor of k larger than in the random-oracle scheme of [GPV08] (where k is the output length of a *chameleon* hash function), and the signing algorithm is essentially as efficient as the one from [GPV08].[1] The underlying hard problem is the standard *short integer solution* (SIS) problem dating back to the seminal work of Ajtai [Ajt04], which is known to be as hard as several worst-case approximation problems on lattices. The concrete approximation factor underlying our signature scheme is $\tilde{O}(\sqrt{k} \cdot n^{3/2})$, which is only a factor $\sqrt{k}$ looser (roughly) than that of [GPV08].

Our second application is a *hierarchical* identity-based encryption (HIBE) scheme, which is the first HIBE that does not rely on bilinear pairings. The scheme works in the standard model, also making it the first non-pairing-based IBE 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 is also as hard as certain worst-case lattice problems [Reg05, Pei09] and is already the foundation for many other cryptographic schemes (including the plain IBE of [GPV08]).

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 [BBDP01] and searching on encrypted data [BCOP04, ABC$^+$08]. While there are a few anonymous (non-hierarchical) IBEs [BF03, CS07, BGH07, GPV08], only one other HIBE is known to be anonymous [BW06].

1.2 Overview of Bonsai Trees and Applications

The art of arboreal 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, the arborist can 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 intuitive description here is developed technically in Section 3.) The first principle is the tree itself, which 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.

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

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 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 with a system, a random oracle, etc. Their key property is that they naturally form a hierarchy as follows: excepting the root, every lattice in the tree is a *higher-dimensional superlattice* of its parent. More precisely, 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 instance into 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., [GGH97, HPS98, HHGP+03, GPV08, PVW08, Pei09]), 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 usually have less power, though in some applications they might also be given control over certain parts of the tree.

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 [GPV08, PVW08, Pei09], 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 [Ajt99] and recently improved by Alwen and Peikert [AP09], which generates a hard random lattice together with a short basis 'from scratch.'

At first glance, the algorithms of [Ajt99, AP09] seem useful only for controlling a new tree entirely by its root, which is not helpful if we need finer-grained control. However, it is observed in [AP09] that the main technique for generating a lattice with a short basis is actually more general. In our language, the technique allows an arborist to *generate and take control of* a new offshoot from *any uncontrolled* branch of a tree. More formally, it provides a way to *extend* a given random lattice (the parent) into a higher-dimensional random superlattice (the child), while simultaneously producing an *entire short basis* for the child lattice. We stress that no special knowledge about the parent is required to do this, even though the two lattices are related; moreover, the child is distributed (statistically) as if it were an undirected offshoot. The technique, therefore, allows the arborist to achieve a primary bonsai aesthetic: a carefully controlled tree that nonetheless gives the appearance of having grown without any outside intervention.[2]

[2]It is worth noting that in [Ajt99, AP09], even the simple goal of generating one lattice 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 according to the

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 [GMR88]. By a standard, efficient transformation using *chameleon hashes* [KR00] (which have efficient realizations under conventional lattice assumptions), it suffices to construct a *weakly secure* scheme, namely, one that is existentially unforgeable under a static attack in which the adversary submits all its query messages 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 message μ of length k, the signer first 'hashes' μ by walking its associated root-to-leaf path, arriving at the corresponding lattice Λ_μ. The signature is simply a short nonzero vector $\mathbf{v} \in \Lambda_\mu$, chosen at random from a certain 'canonical' 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 covering all of M, yet is unlikely to control the leaf of whatever arbitrary message $\mu^* \notin M$ for which $\mathcal{F}$ eventually produces a forgery. If the latter condition happens to hold true, then the forger has found a short nonzero vector in a 'hard' random 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. The simulator can achieve such control 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 [CHK07]. 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 relatively standard.

main technique. Fittingly, this is analogous to the common 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.

For the security proof, the simulator again prepares a bonsai tree so that it controls certain branches (corresponding to 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. As in several prior (H)IBEs, here the simulator (roughly speaking) needs to guess in advance the location of the target identity, which it can do with probability about $2^{-d}/d$, where d is the depth of the tree. We note that the bonsai framework also appears amenable to certain combinatorial techniques introduced by Boneh and Boyen [BB04b] for dealing with adaptive attacks on (H)IBE schemes, though we do not pursue their application in this work.

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 within it, so it is clearly insecure to reveal the child. We address this issue with an additional bonsai principle of *randomizing control*, using the 'oblivious' sampling algorithm of [GPV08]. 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 [CHK04, BCHK07] 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 [Pei09], 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, 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., [Mic07, PR06, LM06, PR07, SSTX09]). This approach can 'miniaturize' the bonsai trees and most of their associated operations by an almost-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 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 scheme, the size of a ciphertext grows linearly with the depth of the identity to which it is encrypted. Moreover, the (Euclidean) length of an entity's trapdoor basis increases *geometrically* with its depth in the tree (more precisely, with the length of the delegation chain), due to the re-randomization that is performed during 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 even for depths as large as $d = n^c$ for any $c < 1$.

As in some prior pairing-based (H)IBEs (e.g.,[GS02, BB04a]), a basic security reduction for a full *adaptive-identity* attack is somewhat loose, degrading exponentially with the (bit) length of the identities. The reason, roughly speaking, is that the simulation must 'guess' in advance which identity the adversary will attack, and there are exponentially many such identities as a function of the depth. Recent works have achieved tight reductions for pairing-based (H)IBEs under various assumptions [Gen06, GH09, Wat09], and a variant of the GPV IBE 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

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 [HW09b], but there are also some significant structural differences. Most significantly, our scheme does not need to perform a trapdoor inversion operation for every prefix of the message as in [HW09b]. Additionally, in contrast with prior hash-and-sign schemes (e.g., [GHR99, CS00, HW09a, HW09b]), our simulator does not use an 'accumulator' to sign *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 induces the forest-of-subtrees structure as described in Section 1.2.1, which requires a somewhat different simulation strategy from that of [HW09b] to ensure that a forgery is useful.

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 blend between pairing-based techniques and the trapdoor techniques found in [Coc01, BGH07, GPV08].

In a concurrent and independent work, Cash, Hofheinz, and Kiltz [CHK09] have also developed a HIBE scheme in the standard model based on the LWE problem. Though their scheme emerged from an alternative perspective and differs in some technical details, the complexity and underlying machinery are very similar to ours. We have also learned that Agrawal and Boyen [AB09] independently developed a standard-model, but *non-hierarchical*, IBE based on LWE. Their construction also has structure and complexity similar to ours, but it does not address delegation. In both other works [CHK09, AB09], signature schemes follow from the (H)IBEs via a generic transformation, but they are less efficient than our signatures by a factor of a security parameter (and have somewhat looser security reductions), because the (H)IBEs must first be made fully secure against adaptive-identity attacks.

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 $i = 2, \ldots, k$, $\widetilde{\mathbf{s}_i}$ is the component of $\mathbf{s}_i$ orthogonal to $\mathrm{span}(\mathbf{s}_1, \ldots, \mathbf{s}_{i-1})$. 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 $\mathcal{A}$ having binary output, 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 $\mathbf{Adv}^{\mathrm{atk}}_{\mathrm{SCH}}(\mathcal{A})$ to describe the of an adversary $\mathcal{A}$ mounting an atk attack on a cryptographic scheme SCH, where the definition advantage is specified as part of the attack. Similarly, we write $\mathbf{Adv}_{\mathrm{PROB}}(\mathcal{A})$ 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 [KR00]. For our purposes, we need a slight generalization in the spirit of "preimage sampleable" (trapdoor) functions [GPV08].

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). The functions h_i are efficiently computable given their description, and have 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 [GPV08]. 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 SS 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 $\mathcal{A}_{\text{SS}}^{\text{eu-scma}}(\mathcal{F})$ 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., [KR00]. Furthermore, the construction results in an *online/offline* signature scheme; see [ST01]. 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:

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

The correctness requirement is: for $(pk, sk) \leftarrow \mathsf{Gen}$ and $(\kappa, \sigma) \leftarrow \mathsf{Encaps}(pk)$, $\mathsf{Decaps}(sk, \sigma)$ 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 $(pk, sk) \leftarrow \mathsf{Gen}$, $(\kappa, \sigma) \leftarrow \mathsf{Encaps}(pk)$, and $\kappa' \leftarrow \{0,1\}^\ell$ (chosen uniformly and independently of the other values). The advantage $\mathbf{Adv}_{\text{KEM}}^{\text{ind-cpa}}(\mathcal{A})$ of an adversary $\mathcal{A}$ is its distinguishing advantage between (pk, sk, κ) and (pk, sk, 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 secret key sk_ε. (In the following, 1^d and mpk are implicit parameters to every algorithm, and every sk_{id} is assumed to include id itself.)
- $\mathsf{Extract}(sk_{id}, id')$, given a secret key for identity $id \in \mathcal{ID}^{<d}$ that is a prefix of $id' \in \mathcal{ID}^{\leq d}$, outputs a secret key $sk_{id'}$ 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.
- $\mathsf{Decaps}(sk_{id}, \sigma)$ outputs a key κ.

The correctness requirement is: for any identity $id \in \mathcal{ID}^{\leq d}$, generate $(mpk, sk_\varepsilon) \leftarrow \mathsf{Setup}(1^d)$, sk_{id} via any legal sequence of calls to $\mathsf{Extract}$ starting from sk_ε, and $(\kappa, \sigma) \leftarrow \mathsf{Encaps}(id)$. Then $\mathsf{Decaps}(sk_{id}, \sigma)$ 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 are concerned only with 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.

The following lemma will be useful in our constructions.

Lemma 2.1 ([MG02, Lemma 7.1, page 129])**.** *There is a deterministic poly-time algorithm* $\mathsf{ToBasis}(\mathbf{B}, \mathbf{S})$ *that, given an arbitrary basis* $\mathbf{B}$ *of an* m*-dimensional lattice* $\Lambda = \mathcal{L}(\mathbf{B})$ *and a full-rank set of lattice vectors* $\mathbf{S} \subset \Lambda$*, outputs a basis* $\mathbf{T}$ *of* Λ *such that* $\|\widetilde{\mathbf{t}_i}\| \leq \|\widetilde{\mathbf{s}_i}\|$ *for all* $i \in [m]$.

2.3.1 Hard Lattices and Problems

We will work with an certain family of integer lattices whose importance in cryptography was first demonstrated Ajtai [Ajt04]. 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

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

One may check that $\Lambda^{\perp}(\mathbf{A})$ contains the identity $\mathbf{0} \in \mathbb{Z}^m$ and is closed under addition, hence it is a subgroup of (and lattice in) $\mathbb{Z}^m$.

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 χ parameterized by α, which is obtained by drawing $x \in \mathbb{R}$ from a normal distribution of standard deviation α and outputting $\lfloor q \cdot x \rceil \bmod q$.

Definition 2.3. The *learning with errors* problem $\mathsf{LWE}_{q,\chi}$ 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 $\tilde{O}(\beta \cdot \sqrt{n})$ [MR07, GPV08]. 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 [Reg05, Pei09] for precise statements.

2.3.2 Gaussians over Lattices

We briefly recall Gaussian distributions over lattices; for more details see [MR07]. For any $s > 0$ and dimension $m \geq 1$, define the Gaussian function $\rho_s : \mathbb{R}^m \to \mathbb{R}^+$ as $\rho_s(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|^2 / s^2)$. For an m-dimensional lattice Λ, define the *discrete Gaussian distribution* $D_{\Lambda,s}$ over Λ (centered at zero) as

$$\forall\, \mathbf{x} \in \Lambda, D_{\Lambda,s}(\mathbf{x}) \sim \rho_s(\mathbf{x})$$

(where each probability is normalized by $\sum_{\mathbf{v} \in \Lambda} \rho_s(\mathbf{v})$), and $D_{\Lambda,s}(\mathbf{x}) = 0$ elsewhere.

We summarize several standard facts from the literature about discrete Gaussians over lattices.

Lemma 2.4. *Let* $\mathbf{B}$ *be a basis of an* m*-dimensional lattice* Λ *(for* $m \geq n$*), and let* $s \geq \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$.

1. *[MR07, Lemma 4.4]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda,s}}[\|\mathbf{x}\| > s \cdot \sqrt{m}] \leq \mathrm{negl}(n)$.
2. *[PR06, Lemma 2.11]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda,s}}[\mathbf{x} = \mathbf{0}] \leq \mathrm{negl}(n)$.
3. *[Reg05, Corollary 3.16]: a set of* $O(m^2)$ *independent samples from* $D_{\Lambda,s}$ *contains a set of* m *linearly independent vectors, except with* $\mathrm{negl}(n)$ *probability.*
4. *[GPV08, Theorem 4.1]: there is a PPT algorithm* $\mathsf{SampleD}(\mathbf{B}, s)$ *that generates samples from* $D_{\Lambda,s}$ *(up to* $\mathrm{negl}(n)$ *statistical distance).*

3 Principles of Bonsai Trees

In this section we lay out the framework and techniques for the cultivation of bonsai trees by a (cryptographic) arborist. There are four basic principles, which we explore in turn: undirected growth, controlled growth, extending control to a descendant, and randomizing control.

3.1 Undirected Growth

Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and let $\mathbf{A}' = \mathbf{A} \| \bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m'}$ for some $m' > m$ be some 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 $\mathbb{Z}^{m'}$). Indeed, 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 $\mathbf{A}'\mathbf{v}' = \mathbf{A}\mathbf{v} = \mathbf{0} \in \mathbb{Z}_q^n$. As such, undirected growth in a bonsai tree is accomplished simply by concatenating fresh uniformly random vectors from $\mathbb{Z}_q^n$ onto a given parity-check matrix $\mathbf{A}$.

3.2 Controlled Growth

We say that an arborist controls a lattice if it knows a relatively good (short) basis for the lattice. The following lemma shows how to grow a new controlled extension off of any (usually uncontrolled) lattice.

Proposition 3.1 ([AP09, Lemma 3.4])**.** *Let* $\delta > 0$ *be any fixed real constant and let* $q \geq 3$ *be odd. There is a PPT algorithm* $\mathsf{ExtLattice}(\mathbf{A}_1, m_2)$ *that, given uniformly random* $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m_1}$ *for any* $m_1 \geq (1+\delta) n \lg q$ *and* $\mathrm{poly}(n)$*-bounded* $m_2 \geq (4+2\delta) n \lg q$*, outputs* $(\mathbf{A}_2 \in \mathbb{Z}_q^{n \times m_2}, \mathbf{S} \in \mathbb{Z}_q^{m \times m})$*, where* $m = m_1 + m_2$*, such that:*

- $\mathbf{A} = \mathbf{A}_1 \| \mathbf{A}_2 \in \mathbb{Z}_q^{n \times m}$ *is within* $\mathrm{negl}(n)$ *statistical distance of uniform,*
- $\mathbf{S}$ *is a basis of* $\Lambda^\perp(\mathbf{A})$*, and*
- $\|\widetilde{\mathbf{S}}\| \leq O(\sqrt{n \lg q})$ *with overwhelming probability.*

3.3 Extending Control

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

The deterministic algorithm $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}' = \mathbf{A} \| \bar{\mathbf{A}})$ takes a basis $\mathbf{S}$ of $\Lambda^\perp(\mathbf{A}) \subseteq \mathbb{Z}^m$ for any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ that generates $\mathbb{Z}_q^n$ and arbitrary $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$, where m and $\bar{m}$ may be arbitrary. It outputs a basis $\mathbf{S}'$ of $\Lambda^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m'}$, where $m' = m + \bar{m}$, computed as follows:

- For $i = 1, \ldots, m$, let $\mathbf{s}'_i = \mathbf{s}_i \| \mathbf{0} \in \mathbb{Z}^{m'}$.
- For $i = 1, \ldots, \bar{m}$, let $\mathbf{s}'_{m+i} = \mathbf{t}_i \| \mathbf{e}_i \in \mathbb{Z}^{m'}$, where $\mathbf{t}_i \in \mathbb{Z}^m$ is an arbitrary integer solution to the equation $\mathbf{A}\mathbf{t}_i = -\bar{\mathbf{a}}_i = \mathbf{a}'_{i+m} \in \mathbb{Z}_q^n$, and $\mathbf{e}_i \in \mathbb{Z}^{\bar{m}}$ is the ith standard basis vector.

Lemma 3.2. $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}')$ *runs in poly-time and outputs a basis* $\mathbf{S}'$ *of* $\Lambda^\perp(\mathbf{A}')$ *such that* $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$.

Proof. The running time of ExtBasis is polynomial, because integer solutions $\mathbf{t}_i$ exist and can be found by standard linear algebra (they need not be short).

We now verify the desired properties of $\mathbf{S}'$. Observe that for $i = 1, \ldots, m$, we have $\mathbf{A}'\mathbf{s}'_i = \mathbf{A}\mathbf{s}_i = \mathbf{0}$ by assumption on $\mathbf{S}$, and for $i = 1, \ldots, \bar{m}$, we have $\mathbf{A}'\mathbf{s}'_{m+i} = \mathbf{A}\mathbf{t}_i + \bar{\mathbf{a}}_i = \mathbf{0}$ by construction. Thus $\mathbf{S}' \subset \Lambda^\perp(\mathbf{A}')$. To check that $\mathbf{S}'$ is indeed a *basis*, let $\mathbf{x}' = \mathbf{x} \| \bar{\mathbf{x}} \in \Lambda^\perp(\mathbf{A}')$ be arbitrary, where $\mathbf{x} \in \mathbb{Z}^m$, $\bar{\mathbf{x}} \in \mathbb{Z}^{\bar{m}}$. Then we have

$$\mathbf{0} = \mathbf{A}'\mathbf{x}' = \mathbf{A}\mathbf{x} + \bar{\mathbf{A}}\bar{\mathbf{x}} = \mathbf{A}\mathbf{x} - (\mathbf{A}\mathbf{T})\bar{\mathbf{x}} = \mathbf{A}(\mathbf{x} - \mathbf{T}\bar{\mathbf{x}}) \in \mathbb{Z}_q^n.$$

Thus $\mathbf{x} - \mathbf{T}\bar{\mathbf{x}} \in \Lambda^\perp(\mathbf{A})$, so by assumption on $\mathbf{S}$ there exists $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{S}\mathbf{z} = \mathbf{x} - \mathbf{T}\bar{\mathbf{x}}$. Now let $\mathbf{z}' = \mathbf{z} \| \bar{\mathbf{x}} \in \mathbb{Z}^{m'}$. By construction of $\mathbf{S}'$, we have

$$\mathbf{S}'\mathbf{z}' = (\mathbf{S}\mathbf{z} + \mathbf{T}\bar{\mathbf{x}}) \| \bar{\mathbf{x}} = \mathbf{x} \| \bar{\mathbf{x}} = \mathbf{x}'.$$

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

Finally, we confirm that $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$. For $i = 1, \ldots, m$, we clearly have $\|\widetilde{\mathbf{s}'_i}\| = \|\widetilde{\mathbf{s}_i}\|$. Now because $\mathbf{S}$ is full-rank, we have $\mathrm{span}(\mathbf{S}) = \mathrm{span}(\mathbf{e}_1, \ldots, \mathbf{e}_m) \subseteq \mathbb{R}^{m'}$. Therefore for $i = m+1, \ldots, m'$, we have $\widetilde{\mathbf{s}'_i} = \mathbf{e}_i \in \mathbb{R}^{m'}$, so $\|\widetilde{\mathbf{s}'_i}\| = 1 \leq \|\widetilde{\mathbf{s}'_1}\|$, and we are done. □

3.4 Randomizing Control

Finally, we show how an arborist can randomize its control over a lattice, with a slight loss in quality. This operation is typically needed for securely delegating control to another entity.

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

1. For $i = 1, \ldots, m$:
   (a) Choose $\mathbf{v} \leftarrow \mathsf{SampleD}(\mathbf{S}, s)$. If $\mathbf{v}$ is linearly independent of $\{\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}\}$, then let $\mathbf{v}_i = \mathbf{v}$ and go to the next value of i; otherwise, repeat this step.
2. Output $\mathbf{S}' = \mathsf{ToBasis}(\mathbf{V}, \mathbf{S})$.

The following lemma follows immediately by Lemma 2.1 (regarding ToBasis) and the facts about discrete Gaussians listed in Lemma 2.4.

Lemma 3.3. *With overwhelming probability,* $\mathbf{S}' \leftarrow \mathsf{RandBasis}(\mathbf{S}, s)$ *repeats Step 1a at most* $O(m^2)$ *times, and* $\|\widetilde{\mathbf{S}'}\| \leq s \cdot \sqrt{m}$. *Moreover, for any two bases* $\mathbf{S}_0, \mathbf{S}_1$ *of the same lattice and any* $s \geq \|\widetilde{\mathbf{S}_i}\| \cdot \omega(\sqrt{\log n})$ *for* $i = 0, 1$, $\mathsf{RandBasis}(\mathbf{S}_0, s)$ *and* $\mathsf{RandBasis}(\mathbf{S}_1, s)$ *are within* $\mathrm{negl}(n)$ *statistical distance.*

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 a full eu-acma-secure scheme.

Our scheme involves a few parameters:

- dimensions $m_1, m_2 = O(n \lg q)$, and a bound $\widetilde{L} = O(\sqrt{n \lg q})$ (all as per Proposition 3.1),
- a (hashed) message length k and a 'total dimension' $m = m_1 + (k+1)m_2$, and
- a Gaussian parameter $s = \widetilde{L} \cdot \omega(\sqrt{\log n})$.

The scheme SS is defined as follows.

- Gen: generate (via Proposition 3.1) $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times (m_1+m_2)}$ that is (negligibly close to) uniform with a basis $\mathbf{S}$ of $\Lambda^\perp(\mathbf{A}_0)$ such that $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L}$. For each $(b, j) \in \{0,1\} \times [k]$, generate uniform and independent $\mathbf{A}_j^{(b)} \in \mathbb{Z}_q^{n \times m_2}$. Output $vk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\})$ and $sk = (\mathbf{S}, vk)$.
- $\mathsf{Sign}(sk, \mu \in \{0,1\}^k)$: 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
$$\mathbf{v} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}_\mu), s).$$
(In the negligibly rare event that $\mathbf{v} = \mathbf{0}$ or $\|\mathbf{v}\| > s \cdot \sqrt{m}$ (Lemma 2.4), we may re-sample $\mathbf{v}$.)
- $\mathsf{Ver}(vk, \mu, \mathbf{v})$: 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.

4.1 Efficient Implementation

The above signing algorithm is described in a way that is most convenient for the analysis, but it is not very efficient: first it extends a dimension-m_1 basis of $\Lambda^\perp(\mathbf{A}_0)$ into a dimension-m basis of $\Lambda^\perp(\mathbf{A}_\mu)$, and then invokes the $\mathsf{SampleD}$ algorithm on the m-dimensional basis; note that this requires time at least quadratic in m. Fortunately, the signing algorithm may be implemented much more efficiently, due to the special structure of the extended basis and the implementation of $\mathsf{SampleD}$ from [GPV08].

Consider $\mathbf{S}' = \mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}_\mu)$. By the construction in Section 3.3, observe that for all $i > j > m_1 + m_2$, the projection of $\mathbf{s}_i$ in the direction of $\widetilde{\mathbf{s}_j} = \mathbf{e}_j$ is zero. In the iterative "nearest-plane" implementation of $\mathsf{SampleD}$, the coordinates v_i for $i > m_1 + m_2$ are therefore independent and drawn from $D_{\mathbb{Z},s}$. Thus there is no need to compute $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}_\mu)$ at all, and the implementation of $\mathsf{SampleD}$ may be optimized to work as follows: choose $\mathbf{v} \leftarrow D_{\Lambda^\perp(\mathbf{A}_\mu), s}$ by sampling $v_i \leftarrow D_{\mathbb{Z},s}$ independently for each $i > m_1 + m_2$, and then choose the first $m_1 + m_2$ coordinates of $\mathbf{v}$ by sampling from a Gaussian over the appropriate coset of $\Lambda^\perp(\mathbf{A}_0)$ to ensure that $\mathbf{v} \in \Lambda^\perp(\mathbf{A}_\mu)$. Essentially, this optimization corresponds to computing the trapdoor function from [GPV08] about k times in the forward direction, followed by one inversion operation.

4.2 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 SS that makes at most Q queries,*

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

Proof. Let $\mathcal{F}$ be an adversary mounting an eu-scma attack on SS. We construct a reduction $\mathcal{S}$ attacking $\mathsf{SIS}_{q,\beta}$. The reduction $\mathcal{S}$ takes as input $m' = m_1 + (2k+1)m_2$ 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 $\mathbf{A}_0 = \mathbb{Z}_q^{n \times (m_1 + m_2)}$ and $\mathbf{U}_i^{(b)} \in \mathbb{Z}_q^{n \times m_2}$.

$\mathcal{S}$ 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 of all strings $p \in \{0,1\}^{\leq k}$ having the property that p is a shortest string such that no $\mu^{(j)}$ has it as a prefix. Intuitively, each p corresponds to a maximal subtree of the bonsai tree that does not contain any of the queried messages. This set may be computed efficiently via a breadth-first pruned search of $\{0,1\}^{\leq k}$, viewed as a binary tree. Namely, starting from a queue initialized to ε, 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 the set, 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 its set uniformly at random, letting $t = |p|$. It then provides an SS 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 $i \in [t]$, generate $\mathbf{A}_i^{(1-p_i)}$ and basis $\mathbf{S}_i$ such that $\|\widetilde{\mathbf{S}_i}\| \leq \widetilde{L}$ by invoking

$$\mathsf{ExtLattice}(\mathbf{A}_0 \| \mathbf{A}_1^{(p_1)} \| \cdots \| \mathbf{A}_{i-1}^{(p_{i-1})}, m_2).$$

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

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

where $\mathbf{A}_\mu$ is as in the signature scheme. (In the rare case that $\mathbf{v} = \mathbf{0}$ or $\|\mathbf{v}\| > s \cdot \sqrt{m}$, $\mathcal{S}$ can simply re-sample $\mathbf{v}$.)

Finally, if $\mathcal{F}$ produces a valid forgery $(\mu^*, \mathbf{v}^* \neq \mathbf{0})$, then we have $\mathbf{A}_{\mu^*} \cdot \mathbf{v}^* = \mathbf{0} \in \mathbb{Z}_q^n$ for $\mathbf{A}_{\mu^*}$ as defined in SS. 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 of the form $\mathbf{U}_i^{(b)}$. Therefore, $\mathcal{S}$ can generate a nonzero $\mathbf{v} \in \mathbb{Z}^{m'}$ by suitably arranging the entries of $\mathbf{v}^*$ and padding with zero entries so that

$$\mathbf{A}\mathbf{v} = \mathbf{0} \in \mathbb{Z}_q^n.$$

$\mathcal{S}$ outputs $\mathbf{v}$ as a solution to SIS.

We now analyze the reduction. First observe that conditioned on any choice of p, the verification key vk given to $\mathcal{F}$ is negligibly close to uniform. Therefore, the choice of p is statistically hidden from $\mathcal{F}$, so if $\mathcal{F}$ produces a valid forgery, p is a prefix of μ^* with probability at least $1/(k \cdot Q) - \mathrm{negl}(n)$. In such a case, it is immediate that $\|\mathbf{v}\| = \|\mathbf{v}^*\| \leq \beta$, hence $\mathbf{v}$ is a valid solution to SIS, as desired. Finally, observe that 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 $s \geq \|\widetilde{\mathbf{S}_i}\| \cdot \omega(\sqrt{\log n})$. □

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, which is based on the idea of using a short lattice basis as the trapdoor for an injective one-way function, is now standard (see [GPV08, Pei09]). 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 parameterized by the modulus q, dimension m, key length ℓ, and a bound $\widetilde{L}$ 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. KEM also uses an algorithm called Invert, which recovers $\mathbf{s} \in \mathbb{Z}_q^n$ from $\mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s})$, given $\mathbf{A}$ and a suitably short basis of $\Lambda^\perp(\mathbf{A})$. See, e.g., [Pei09] for details on the implementation of Invert.

- Gen: generate (via Proposition 3.1) $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ that is (negligibly close to) uniform with a basis $\mathbf{S}$ of $\Lambda^\perp(\mathbf{A})$ such that $\|\widetilde{\mathbf{S}}\| \le \widetilde{L}$. Also generate $\mathbf{U} \in \mathbb{Z}_q^{n\times\ell}$ uniformly at random. Output public key $pk = (\mathbf{A}, \mathbf{U})$ and secret key $sk = (\mathbf{S}, pk)$.
- $\mathsf{Encaps}(pk = (\mathbf{A}, \mathbf{U}))$: choose key $\mathbf{k} \leftarrow \{0,1\}^\ell$ and $\mathbf{s} \leftarrow \mathbb{Z}_q^n$. Output key $\mathbf{k}$ and ciphertext $(\mathbf{b}, \mathbf{p}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$, where
$$\mathbf{b} \leftarrow \mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s}) \quad \text{and} \quad \mathbf{p} \leftarrow \mathsf{Noisy}_\chi(\mathbf{U}^t\mathbf{s} + \mathbf{k} \cdot \lfloor q/2 \rfloor).$$
- $\mathsf{Decaps}(sk, (\mathbf{b}, \mathbf{p}))$: Let $\mathbf{s} \leftarrow \mathsf{Invert}(\mathbf{S}, \mathbf{A}, \mathbf{b})$. Output the $\mathbf{k} \in \{0,1\}^\ell$ such that $\mathbf{k} \cdot \lfloor q/2 \rfloor$ is closest (modulo q) to $\mathbf{p} - \mathbf{U}^t\mathbf{s} \in \mathbb{Z}_q^\ell$.

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 $q \ge 2\widetilde{L}\sqrt{m}$ and $\chi = \bar{\Psi}_\alpha$ for $1/\alpha \ge \widetilde{L} \cdot \omega(\sqrt{\log n})$. Then* Decaps *is correct with overwhelming probability over the randomness of* 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 Schemes

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 [CHK07]. We mainly focus on the case of *selective-identity*, *chosen-plaintext* attacks, i.e., sid-ind-cpa security.

The BTE scheme is parameterized by dimensions $m_1, m_2 = O(n \lg q)$ 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),

- $m_1 + (i+1)m_2$ 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 (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.

- $\mathsf{Setup}(d)$: Generate (via Proposition 3.1) $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times (m_1+m_2)}$ 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 $\mathbf{A}_j^{(b)} \in \mathbb{Z}_q^{n \times m_2}$. Choose $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$ uniformly at random. Output $mpk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\}, \mathbf{U}, d)$ and $msk = \mathbf{S}_0$.

  All the remaining algorithms implicitly take the master public key mpk as an input. For an identity $id \in \{0,1\}^*$ of length $t = |id| \leq d$, we let $\mathbf{A}_{id} = \mathbf{A}_0 \| \mathbf{A}_1^{(id_1)} \| \cdots \| \mathbf{A}_t^{(id_t)}$, and let $pk_{id} = (\mathbf{A}_{id}, \mathbf{U})$ denote the KEM public key associated with identity id.

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

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

  (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 $\mathsf{RandBasis}$.)

  Output $sk_{id'} = (\mathbf{S}_{id'}, pk_{id'})$.

- $\mathsf{Encaps}(id)$: output $(\kappa, \sigma) \leftarrow \mathsf{KEM.Encaps}(pk_{id})$.
- $\mathsf{Decaps}(sk_{id}, \sigma)$: output $\kappa \leftarrow \mathsf{KEM.Decaps}(sk_{id}, \sigma)$.

Instantiating the parameters. Suppose that BTE is used in a setting in which $\mathsf{Extract}(\mathbf{S}_{id}, id')$ is invoked only on identities id' whose lengths are a multiple of some $k \geq 1$. For example, consider the two main applications of [CHK07]: 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.

We need only define s_i and $\widetilde{L_i}$ for i that are multiples of k. Let

$$\widetilde{L_i} = s_i \cdot \sqrt{m_1 + (i+1)m_2} = s_i \cdot O(\sqrt{d \cdot n \lg q})$$

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

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

Finally, to ensure that the underlying KEM is complete (Lemma 5.1), we let $q \geq 2\widetilde{L_d}\sqrt{m_1 + (d+1)m_2}$ and $\chi = \bar{\Psi}_\alpha$ for $1/\alpha = \widetilde{L_d} \cdot \omega(\sqrt{\log n})$. For any $d = \mathrm{poly}(n)$, invoking the worst-case to average-case reduction for LWE yields an underlying approximation factor of $\tilde{O}(n^{3/2} \cdot (d \cdot \sqrt{n})^{d/k})$.

5.2.1 Variations

Anonymity. With a small modification, BTE may be made *anonymous* across all depths of the hierarchy. That is, ciphertexts hide (computationally) the particular identities to which they are encrypted. The modification is simply to extend the $\mathbf{b}$ component of the KEM ciphertext to have length exactly $m_1 + (d+1)m_2$, 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.

Shorter public key in the random oracle model. In the random oracle model, 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. Moreover, the identity hierarchy need not be a binary tree, but may have arbitrary degree. The security proof for this variant is a straightforward adaptation of the one given below.

Adaptive-identity security. Our proof of security may be adapted to the case of adaptive-identity attacks (aid-ind-cpa), via a reduction that loses a factor of about $2^{-d}/d$ in its advantage relative to the adversary. Techniques such as those found in [BB04b, Wat05] may be applicable to improve the tightness of the reduction; we leave this to future work.

Chosen-ciphertext security. Security under chosen-ciphertext attack (sid-ind-cca or aid-ind-cca) follows directly by a transformation of [BCHK07], from ind-cpa-secure HIBE for depth $d+1$ to ind-cca-secure HIBE for depth d.

5.3 Security

Theorem 5.2 (Security of BTE)**.** *Let atk = sid-ind-cpa. There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking KEM (instantiated with dimension $m = m_1 + (d+1)m_2$ and q, χ as in BTE) such that, for any adversary $\mathcal{A}$ mounting an atk attack on BTE,*

$$\mathbf{Adv}_{KEM}(\mathcal{S}^{\mathcal{A}}) \geq \mathbf{Adv}^{atk}_{BTE}(\mathcal{A}) - \mathrm{negl}(n).$$

Proof of Theorem 5.2. 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 $pk = (\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{U} \in \mathbb{Z}_q^{n \times \ell})$, an encapculation $(\mathbf{b}, \mathbf{p}) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$, and a key $\mathbf{k} \in \{0,1\}^\ell$ which either is encapsulated by $(\mathbf{b}, \mathbf{p})$ 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 HIBE 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 $\mathbf{A} = \mathbf{A}_0 \| \mathbf{A}_1 \| \cdots \| \mathbf{A}_d$, where $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times (m_1 + m_2)}$ and $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m_2}$ for all $i \in [d]$. Similarly, truncate $\mathbf{b}$ to $\mathbf{b}^* \in \mathbb{Z}_q^{m_1 + (t^*+1)m_2}$.
- *Undirected growth.* For each $i \in [t^*]$, let $\mathbf{A}_i^{(id_i^*)} = \mathbf{A}_i$.

- *Controlled growth.* For each $i \in [t^*]$, generate $\mathbf{A}_i^{(1-id_i^*)} \in \mathbb{Z}_q^{n \times m_2}$ and basis $\mathbf{T}_i$ by invoking $\mathsf{ExtLattice}(\mathbf{A}_0 \| \cdots \| \mathbf{A}_{i-1}, m_2)$. If $t^* < d$, for each $b \in \{0,1\}$ generate $\mathbf{A}_{t^*+1}^{(b)}$ and basis $\mathbf{T}_{t^*+1}^{(b)}$ by two independent invocations of $\mathsf{ExtLattice}(\mathbf{A}_0 \| \cdots \| \mathbf{A}_{t^*}, m_2)$. For each $i > t^* + 1$ (if any) and $b \in \{0,1\}$, generate $\mathbf{A}_i^{(b)} \in \mathbb{Z}_q^{n \times m_2}$ uniformly at random.

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

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 as

$$\mathbf{S}_{id} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{T}_i, \mathbf{A}_{id}), s_t).$$

- If $t = |id| > t^*$, answer the query as

$$\mathbf{S}_{id} \leftarrow \mathsf{RandBasis}(\mathsf{ExtBasis}(\mathbf{T}_{t^*+1}^{(id_{t^*+1})}, \mathbf{A}_{id}), s_t).$$

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 $\mathbf{T}_i, \mathbf{T}_{t^*+1}^{(b)}$ are sufficiently short to invoke $\mathsf{RandBasis}$). Finally, the encapsulation $(\mathbf{b}^*, \mathbf{p})$ (for identity id^*) and key $\mathbf{k}$ 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. □

References

[AB09] Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. Manuscript, July 2009.

[ABC+08] 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.

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

[Ajt04] Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582, 2001.

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

[BCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[BF03] 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.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[CHK07] 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.

[CHK09] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. Manuscript, July 2009.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[CS00] 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.

[CS07] Giovanni Di Crescenzo and Vishal Saraswat. Public key encryption with searchable keywords based on Jacobi symbols. In *INDOCRYPT*, pages 282–296, 2007.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *ACM Workshop on Digital Rights Management*, pages 61–80, 2002.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

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

[GH09] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *TCC*, pages 437–456, 2009.

[GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT*, pages 123–139, 1999.

[GMR88] 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.

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

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[HHGP+03] 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.

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[HW09a] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *EUROCRYPT*, pages 333–350, 2009.

[HW09b] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In *CRYTPO*, 2009. To appear.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS*, 2000.

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

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

[LN09] Gaëtan Leurent and Phong Q. Nguyen. How risky is the random-oracle model? In *CRYPTO*, 2009. To appear.

[MG02] 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.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

[MR07] 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.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.

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

[PR07] Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, pages 478–487, 2007.

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

[Rab79] 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.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. Cryptology ePrint Archive, Report 2009/285, 2009. `http://eprint.iacr.org/`.

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

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

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, 2009. To appear.

[YFDL04] 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.