Bonsai Trees, or How to Delegate a Lattice Basis

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

September 6, 2011

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 [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.

[*]University of California, San Diego. Email: `cdc@ucsd.edu`. Part of this work was 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. Email: `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, CNS-0749931, and CAREER award CCF-1054495, by the US Department of Homeland Security under Contract Number HSHQDC-07-C-00006, and the Alfred P. Sloan Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the US Department of Homeland Security, or the Sloan Foundation.

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 and Franklin [BF01]. For example, the more general notion of *hierarchical* IBE [HL02, GS02] 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 [CHK03] and broadcast encryption [DF02, YFDL04].

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 [GMR84, NY89] and specialized to lattice assumptions [LM08]), 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 [GPV08], 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 [GPV08].[1] The underlying hard problem is the standard *short integer solution* (SIS) problem dating back to the seminal work of Ajtai [Ajt96], which is known to be as hard as several worst-case approximation problems on lattices (see also [MR04, GPV08]). 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 [GPV08].

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 [Reg05, Pei09b]; LWE is also the foundation for the plain IBE of [GPV08], 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 [BBDP01] and searching on encrypted data [BCOP04, ABC$^+$05]. While there are a few anonymous (non-hierarchical) IBEs [BF01, CS07, BGH07, GPV08], only one other HIBE is known to be anonymous [BW06].

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

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

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 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., [GGH97, HPS98, HHGP$^+$03, GPV08, PVW08, Pei09b]), 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. As described in [GPV08], these two operations are effectively the inversion algorithms for certain families of injective and 'preimage sampleable' trapdoor functions, respectively. The reader may view a short basis for a lattice as roughly analogous to the factorization of an RSA modulus, 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 [GPV08, PVW08, Pei09b], 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. 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]

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 [GMR84]. By a standard, efficient transformation using *chameleon hashes* [KR00] (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 of 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 for whatever arbitrary message $\mu^* \notin M$ that $\mathcal{F}$ eventually produces in its 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. (This combinatorial game is closely related to the recent "prefix technique" introduced by Hohenberger and Waters [HW09]; see Section 1.4 for discussion.)

To satisfy the conflicting constraints of the game, $\mathcal{S}$ 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 can be achieved 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.

[2]It is worth noting that in [Ajt99, AP09], 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.

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 [CHK03]. 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 security against a *fully adaptive* attack ("full security"), where the adversary may choose its target identity after making its secret-key queries. (See Section 2.2 for a precise definition.) 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 approach of Boneh and Boyen [BB04a] to 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 [BB04b] 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 [BB04b]. However, as we will explain in Section 5.4.4, 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 solve this issue with an additional bonsai principle of *randomizing control*, using the Gaussian sampling algorithm of [GPV08]. This produces a new basis under a 'canonical' distribution, 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 [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

Application	Model	Public key	Secret key	Signature	Assumption
Hash-and-sign [GPV08]	RO	$\tilde{O}(n^2)$	$\tilde{O}(n^2)$	$\tilde{O}(n)$	$\tilde{O}(n)$-SIS
Hash-and-sign	Standard	$\tilde{O}(kn^2)$	$\tilde{O}(n^2)$	$\tilde{O}(kn)$	$\tilde{O}(\sqrt{k} \cdot n)$-$\mathsf{SIS}$

Application	Model	Public key	Secret key	Ciphertext	$1/\alpha$ for LWE
IBE [GPV08]	RO	$\tilde{O}(n^2)$	$\tilde{O}(n^2)$	$\tilde{O}(n)$	$\tilde{O}(n)$
IBE	Standard	$\tilde{O}(kn^2)$	$\tilde{O}(n^2)$	$\tilde{O}(kn)$	$\tilde{O}(\sqrt{k} \cdot n)$
HIBE (depth d)	RO	$\tilde{O}(d^2n^2)$	$\tilde{O}(d^5n^2)$	$\tilde{O}(d^3n)$	$\sqrt{dn} \cdot \tilde{O}(d\sqrt{n})^d$
HIBE (depth d)	Standard	$\tilde{O}(d^3kn^2)$	$\tilde{O}(d^5k^2n^2)$	$\tilde{O}(d^3kn)$	$\sqrt{dn} \cdot \tilde{O}(d\sqrt{kn})^d$

Figure 1: Complexity of our bonsai tree applications in comparison with prior work; n is the main security parameter and k is the output length of a chameleon hash function (for the signature scheme) or collision-resistant hash function (for the IBEs). "RO" means random oracle model under fully adaptive attack, while "Standard" means standard model under selective-identity attack (standard-model schemes with adaptive security are omitted). "Public key," "secret key," etc. refer to their lengths in bits; for the HIBEs, these quantities are the maximum possible over all levels of the hierarchy. The column labelled "$1/\alpha$ for LWE" refers to the inverse error rate of the underlying LWE problem used in the proofs of security.

of exactly the same 'type' as those in the recent CCA-secure cryptosystem of [Pei09b], 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 [GPV08], 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., [Mic02, PR06, LM06, PR07, SSTX09, LPR10]), with little to no essential changes to the proofs. 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 linearly with the depth of the identity to which it is encrypted. Moreover, the Euclidean length of a 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 asymptotically 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 correspond 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 [Wat09]) 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 [Gen06, GH09, Wat09], 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 knowing at least one secret key for every 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 [CHK09] and the fourth author [Pei09a], which contained some overlapping results and were accepted to Eurocrypt 2010 under the condition that they be merged.

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

The analysis of our signature scheme may be seen as a refinement of the recent "prefix technique" of Hohenberger and Waters [HW09] for standard-model signatures. Our scheme and its proof appear quite different from their main RSA-based construction, but is closely related (at least at a combinatorial level) to their construction based on bilinear pairings. Interestingly, in both of these constructions the scheme itself does not deal with message prefixes at all; they only show up in the security proofs. The main difference between our simulation strategies is that, in the prefix technique, the simulator in a static chosen-message attack sets up the public key by guessing a uniformly random prefix of a random queried message, in the hope that the adversary's forged message will start with that prefix (but its next bit will be different). With this strategy, some of the simulator's potential guesses may prevent it from being able to sign all the queried messages, thus causing the simulation to fail. (Fortunately, a noticeable fraction of the simulator's guesses avoid this outcome, so the reduction goes through correctly.) In contrast, our simulator looks at *all* the queried messages, and sets up the public key so that it always knows how to sign all of them (and potentially many others). This is done by constructing a forest of subtrees as described in Section 1.2.1.

The structure of our HIBE is also similar, at a combinatorial level, 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 key 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 [Coc01, BGH07, GPV08].

Following the initial dissemination of our results in [CHK09, Pei09a], several extensions and additional applications have been found. Rückert [Rüc10] modified our signature scheme to make it *strongly* unforgeable, and constructed hierarchical identity-based signatures. Agrawal and Boyen [AB09] 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 [ABB10] improved the efficiency of our (H)IBE schemes (under a quantitatively stronger LWE assumption), and Boyen [Boy10] 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. For an alphabet X and nonnegative integer n, define $X^{\leq n} = \bigcup_{i=0}^{n} X^i$, and $X^{<n}$ similarly. We say that a function in n is *negligible*, written $\operatorname{negl}(n)$, if it vanishes faster than the inverse of any polynomial in n. We say that a probability $p(n)$ is *overwhelming* if $1 - p(n)$ is negligible.

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 $\operatorname{span}(\mathbf{s}_1, \ldots, \mathbf{s}_{i-1})$. 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 $\widetilde{\mathbf{S}} = \mathbf{Q} \cdot \operatorname{diag}(r_{1,1}, \ldots, r_{k,k})$. 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 $\mathbf{Adv}^{\mathsf{atk}}_{\mathsf{SCH}}(\mathcal{A})$ 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 $\mathbf{Adv}_{\mathsf{PROB}}(\mathcal{A})$ 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). 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 $h_i(m, r) = h_i(m', r')$.

In Section 4 we realize chameleon hash functions (in the above sense) using lattices, via bonsai techniques.

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 $\mathcal{A}^{\text{eu-scma}}_{\mathsf{SIG}}(\mathcal{F})$ 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., [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 $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{KEM}}(\mathcal{A})$ of an adversary $\mathcal{A}$ is its distinguishing advantage between (pk, σ^*, κ^*) and (pk, σ^*, κ').

Hierarchical Identity-Based Encryption (HIBE). In HIBE, identities are strings (tuples) over some alphabet $\mathcal{ID}$. Hence hierarchical identities of maximal depth $d \in \mathbb{N}$ are of the form $\mathcal{ID}^{\leq d} = \bigcup_{i=0}^{d} \mathcal{ID}^i$. A HIBE for keys of bit length $\ell = \ell(n)$ 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. Here d is the maximal depth of the hierarchy, specified in unary. (In the following, d and mpk are implicit parameters to every algorithm, and every user secret key sk_{id} is assumed to include id itself.)
- $\mathsf{Extract}(sk_{id}, id')$, given a user secret key for identity $id \in \mathcal{ID}^{<d} = \bigcup_{i=0}^{d-1} \mathcal{ID}^i$ that is a proper prefix of $id' \in \mathcal{ID}^{\leq d}$, outputs a user 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 $id \in \mathcal{ID}^{\leq d}$, first generate $(mpk, sk_\varepsilon) \leftarrow \mathsf{Setup}(1^d)$, then generate sk_{id} via any legal sequence of calls to $\mathsf{Extract}$ starting from sk_ε, then generate $(\kappa, \sigma) \leftarrow \mathsf{Encaps}(id)$. Then $\mathsf{Decaps}(sk_{id}, \sigma)$ should output κ with overwhelming probability (over all the randomness in the experiment).

There are several attack notions for HIBE. One simple notion is 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, $(mpk, sk_\varepsilon) \leftarrow \mathsf{Setup}(1^d)$, $(\kappa, \sigma^*) \leftarrow \mathsf{Encaps}(id^*)$, 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 adversary's advantage $\mathbf{Adv}^{\text{sid-ind-cpa}}_{\mathsf{HIBE}}(\mathcal{A})$ is its distinguishing advantage between the two cases $\kappa^* = \kappa$ and $\kappa^* = \kappa'$.

A related notion is security against an *adaptive-identity* attack, known as aid-ind-cpa or "full" security. Here the adversary is first given mpk and has oracle access to $\mathsf{Extract}(sk_\varepsilon, \cdot)$ both before and after choosing its target identity id^*; as above, the adversary may never query any identity that is a prefix of the id^* it eventually names. Finally, both of the above notions may be extended to *chosen-ciphertext* attacks in the natural way; since our focus is on chosen-plaintext security, 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 $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{Z}^m$:

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

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 [MW01] 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 ([MG02, 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)* $\mathbf{S}$ *of lattice vectors in* $\Lambda = \mathcal{L}(\mathbf{B})$*, 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 a certain family of integer lattices whose importance in cryptography was first demonstrated by Ajtai [Ajt96]. 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 $q\mathbb{Z}^m$ (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

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

where $\bar{\mathbf{x}} \in \mathbb{Z}^m$ is an arbitrary solution to $\mathbf{A}\bar{\mathbf{x}} = \mathbf{y}$.

It is known (see, e.g., [Reg05, 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 $m = \operatorname{poly}(n)$ 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 $\tilde{O}(\beta \cdot \sqrt{n})$ [MR04, 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, Pei09b] 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 [MR04, GPV08]. 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 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. *[MR04, Lemma 4.4]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}}[\|\mathbf{x}\| > s \cdot \sqrt{m}] \leq \operatorname{negl}(n)$.
2. *[PR06, Lemma 2.11]:* $\Pr_{\mathbf{x} \leftarrow D_{\Lambda^\perp(\mathbf{A}),s}}[\mathbf{x} = \mathbf{0}] \leq \operatorname{negl}(n)$.
3. *[Reg05, 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* $\operatorname{negl}(n)$ *probability.*
4. *[GPV08, 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* $\operatorname{negl}(n)$ *statistical distance), and the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}$.*
5. *[GPV08, Theorem 4.1]: there is a PPT algorithm* $\mathsf{SampleD}(\mathbf{S}, \mathbf{A}, \mathbf{y}, s)$ *that generates a sample from $D_{\Lambda_{\mathbf{y}}^\perp(\mathbf{A}),s}$ (up to* $\operatorname{negl}(n)$ *statistical distance).*

For Item 5 above, a recent work of Peikert [Pei10] gives an alternative $\mathsf{SampleD}$ 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 smaller than $\|\widetilde{\mathbf{S}}\|$, but is also not much larger in most important cases; see [Pei10] 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 addition, 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,

$$\Lambda^\perp(\mathbf{A}' \cdot \mathbf{P}) = \mathbf{P} \cdot \Lambda^\perp(\mathbf{A}')$$

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 ([AP09]). *There is a fixed constant $C > 1$ and a probabilistic polynomial-time algorithm* $\mathsf{GenBasis}(1^n, 1^m, q)$ *that, for* $\operatorname{poly}(n)$*-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* $\operatorname{negl}(n)$ *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.

Lemma 3.2. *There is a deterministic polynomial-time algorithm* $\mathsf{ExtBasis}$ *with the following properties: given an arbitrary* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *whose columns generate the entire group* $\mathbb{Z}_q^n$*, an arbitrary basis* $\mathbf{S} \in \mathbb{Z}^{m\times m}$ *of* $\Lambda^\perp(\mathbf{A})$*, and an arbitrary* $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times \bar{m}}$*,* $\mathsf{ExtBasis}(\mathbf{S}, \mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}})$ *outputs a basis* $\mathbf{S}'$ *of* $\Lambda^\perp(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$

[3] An earlier version of this paper [Pei09a] used a core lemma from [AP09] to directly extend a given random parity-check matrix $\mathbf{A}$ (*without* known good basis) into a random matrix $\mathbf{A}' = \mathbf{A}\|\bar{\mathbf{A}}$ *with* known good basis. While using that method can improve our applications' key sizes by a small constant factor, it also makes the applications more cumbersome to describe, so we opt to use the above proposition instead.

such that $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$. *Moreover, the same holds even for any given permutation of the columns of* $\mathbf{A}'$ *(e.g., if columns of* $\bar{\mathbf{A}}$ *are both appended and prepended to* $\mathbf{A}$*).*

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

$$\mathbf{S}' = \begin{pmatrix} \mathbf{S} & \mathbf{W} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \in \mathbb{Z}^{m' \times m'},$$

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 $\mathbf{A}\mathbf{W} = -\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$. Note that $\mathbf{W}$ exists by the hypothesis that $\mathbf{A}$ generates $\mathbb{Z}_q^n$, and it may be computed efficiently using, e.g., Gaussian elimination.

We now analyze $\mathbf{S}'$. First,

$$\mathbf{A}'\mathbf{S}' = \mathbf{A}\mathbf{S} \| (\mathbf{A}\mathbf{W} + \bar{\mathbf{A}}) = \mathbf{0} \in \mathbb{Z}_q^{n \times m'}$$

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

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

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 $\mathbf{S}\mathbf{z} = \mathbf{v} - \mathbf{W}\bar{\mathbf{v}}$. Now let $\mathbf{z}' = \mathbf{z} \| \bar{\mathbf{v}} \in \mathbb{Z}^{m + \bar{m}}$. By construction, we have

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

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 $\|\widetilde{\mathbf{S}'}\| = \|\widetilde{\mathbf{S}}\|$. By the design of $\mathbf{S}'$ and the definition of Gram-Schmidt orthogonalization, it is easy to check that

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

because $\mathbf{S}$ is full-rank. It follows that $\|\widetilde{\mathbf{S}'}\| = \max\{\|\widetilde{\mathbf{S}}\|, 1\} = \|\widetilde{\mathbf{S}}\|$, because $\widetilde{\mathbf{s}_1} = \mathbf{s}_1 \in \mathbb{Z}^m$ is nonzero.

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 $\mathbf{S}'' = \mathbf{P}\mathbf{S}'$ as a basis for $\Lambda^{\perp}(\mathbf{A}' \cdot \mathbf{P})$, where $\mathbf{P}$ is the given 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 in the QR decompositions of $\mathbf{S}'$ and $\mathbf{P}\mathbf{S}'$ are exactly the same. $\square$

3.3.1 An Optimization: Gaussian Sampling via Implicit Extension

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 [GPV08], 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, and $\widetilde{\mathbf{S}'} = \left(\begin{smallmatrix} \widetilde{\mathbf{S}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$. By inspection, it can be verified that a recursive execution of $\mathbf{v}' \leftarrow \mathsf{SampleD}(\mathbf{S}', \mathbf{y}', s)$ simply chooses all the entries of $\bar{\mathbf{v}} \in \mathbb{Z}^{\bar{m}}$ *independently* from $D_{\mathbb{Z},s}$, then chooses $\mathbf{v} \leftarrow \mathsf{SampleD}(\mathbf{S}, \mathbf{y}' - \bar{\mathbf{A}}\bar{\mathbf{v}}, s)$, and outputs $\mathbf{v}' = \mathbf{v} \| \bar{\mathbf{v}}$. 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 [GPV08], 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 Gaussian 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 (essentially) statistically independent 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 efficiently 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, to convert this into an (offline / online) signature scheme that is unforgeable under adaptive chosen-message attack (eu-acma-secure), we need a suitable notion of chameleon hash, which we also construct here.

[4] An earlier version of this paper [CHK09] explicitly defined a sampling procedure using this algorithm, 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 execution of $\mathsf{SampleD}$ on the structured basis $\mathbf{S}'$.

4.1 Chameleon Hash

Let $n \geq 1$, $q \geq 2$ be integers, let $m = O(n \log q)$ and $\widetilde{L}$ be as in Proposition 3.1, let $s = \widetilde{L} \cdot \omega(\sqrt{\log n})$ be a Gaussian parameter, and let $k \geq 1$ be an integer message length. Define a message space $\mathcal{M} = \{0,1\}^k$, randomness space $\mathcal{R} = \{\mathbf{r} \in \mathbb{Z}^m : \|\mathbf{r}\| \leq s \cdot \sqrt{m}\}$ with distribution $D_{\mathbb{Z}^m,s}$ (restricted to $\mathcal{R}$, which by Item 1 of Lemma 2.4 changes the distribution by only $\mathrm{negl}(n)$ statistical distance), and range $\mathcal{Y} = \mathbb{Z}_q^n$.

For $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times k}$, $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{A} = \mathbf{A}_0 \| \mathbf{A}_1$, define the hash function $h_{\mathbf{A}} \colon \mathcal{M} \times \mathcal{R} \to \mathcal{Y}$ as

$$h_{\mathbf{A}}(\mathbf{m}; \mathbf{r}) = \mathbf{A} \cdot (\mathbf{m} \| \mathbf{r}) = \mathbf{A}_0 \mathbf{m} + \mathbf{A}_1 \mathbf{r}.$$

Lemma 4.1. *The family $\mathcal{H} = \{h_{\mathbf{A}}\}$ (under the uniform distribution over $\mathcal{H}$) is a family of chameleon hash functions, assuming that $\mathsf{SIS}_{q,\beta}$ is hard for $\beta = \sqrt{k + 4s^2 \cdot m}$.*

Proof. It is easy to verify that all of the required properties described in Section 2.2 hold. First, each $h_{\mathbf{A}}$ is efficiently computable given $\mathbf{A}$, and for every $\mathbf{m} \in \mathcal{M}$, the distribution of $(\mathbf{A}, h_{\mathbf{A}}(\mathbf{m}; \mathbf{r}))$ for $\mathbf{r} \leftarrow D_{\mathbb{Z}^m,s}$ is within $\mathrm{negl}(n)$ statistical distance of uniform over $\mathcal{H} \times \mathcal{Y}$ by Item 4 of Lemma 2.4.

For the trapdoor property, a uniformly random (up to $\mathrm{negl}(n)$ statistical distance) instance $h_{\mathbf{A}} = \mathbf{A}_0 \| \mathbf{A}_1$ may be selected by choosing $\mathbf{A}_0$ at random, and generating $(\mathbf{A}_1, \mathbf{S}) \leftarrow \mathsf{GenBasis}(1^n, 1^m, q)$, where $\mathbf{S}$ is a basis of $\Lambda^\perp(\mathbf{A}_1)$ and $\|\widetilde{\mathbf{S}}\| \leq \widetilde{L}$, as described in Proposition 3.1. Then for any $\mathbf{y} \in \mathbb{Z}_q^n$ and $\mathbf{m} \in \mathcal{M}$, by Item 5 of Lemma 2.4 one can use the basis $\mathbf{S}$ to efficiently sample randomness $\mathbf{r}$ from (a distribution within $\mathrm{negl}(n)$ statistical distance of) $D_{\Lambda^\perp_{\mathbf{u}}(\mathbf{A}_1),s}$, where $\mathbf{u} = \mathbf{y} - \mathbf{A}_0 \mathbf{m} \in \mathbb{Z}_q^n$. By Item 4 of Lemma 2.4, this is the distribution of $\mathbf{r} \in \mathcal{R}$ conditioned on $h_{\mathbf{A}}(\mathbf{m}; \mathbf{r}) = \mathbf{y}$.

Finally, for the collision-resistance property, note that any collision $\mathbf{m} \| \mathbf{r} \neq \mathbf{m}' \| \mathbf{r}' \in \mathcal{M} \times \mathcal{R}$ for $h_{\mathbf{A}}$ (for uniformly random $\mathbf{A}$) immediately yields a solution $\mathbf{z} = (\mathbf{m} - \mathbf{m}') \| (\mathbf{r} - \mathbf{r}') \neq \mathbf{0}$ to the instance $\mathbf{A}$ of $\mathsf{SIS}_{q,\beta}$, because

$$\mathbf{A}\mathbf{z} = h_{\mathbf{A}}(\mathbf{m}; \mathbf{r}) - h_{\mathbf{A}}(\mathbf{m}'; \mathbf{r}') = \mathbf{0},$$

and $\|\mathbf{z}\|^2 \leq k + 4s^2 \cdot m = \beta^2$ by the triangle inequality. □

4.2 Signature Scheme

In addition to the main SIS parameters n and q, our signature scheme also involves:

- a dimension $m = O(n \log q)$ and a bound $\widetilde{L} = O(\sqrt{n \log q})$, as per Proposition 3.1;
- a (hashed) message length $k \geq 1$, 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: generate $(\mathbf{A}_0, \mathbf{S}_0) \leftarrow \mathsf{GenBasis}(1^n, 1^m, q)$ (see Proposition 3.1), where $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ is negligibly close to uniform and $\mathbf{S}_0$ is a basis of $\Lambda^\perp(\mathbf{A}_0)$ with $\|\widetilde{\mathbf{S}_0}\| \leq \widetilde{L}$. (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 $\|\mathbf{v}\| > s \cdot \sqrt{m'}$ or $\mathbf{v} = \mathbf{0}$ (Lemma 2.4, items 1 and 2, respectively), 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 public key (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 and its security proof are easy to modify accordingly.

4.3 Security

Theorem 4.2. *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,*

$$\mathbf{Adv}_{\mathsf{SIS}_{q,\beta}}(\mathcal{S}^{\mathcal{F}}) \geq \mathbf{Adv}^{\textit{eu-scma}}_{\mathsf{SIG}}(\mathcal{F}) / ((k-1) \cdot Q + 1) - \operatorname{negl}(n).$$

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

The reduction $\mathcal{S}$ simulates the static chosen-message attack to $\mathcal{F}$ as follows. First, $\mathcal{S}$ invokes $\mathcal{F}$ to receive $Q' \leq Q$ messages $\mu^{(1)}, \mu^{(2)}, \ldots, \mu^{(Q')} \in \{0,1\}^k$. Then $\mathcal{S}$ computes the nonempty set P of all strings $p \in \{0,1\}^{\leq k}$ having the property that p is a shortest string for which no $\mu^{(i)}$ has p as a prefix. Equivalently, P is the set of maximal subtrees of $\{0,1\}^{\leq k}$ (viewed as a tree) that do not contain any of the queried messages. Such a set may be computed iteratively as follows. Start with $P = \{\varepsilon\}$. Then for $i = 1, \ldots, Q'$, do the following: if any element $p \in P$ is a prefix of $\mu^{(i)}$ (there will be at most one such p), then remove p from P, and add to P all the $k - |p|$ elements $p' \in \{0,1\}^{\leq k}$ of length $|p'| \geq |p| + 1$ that agree with the length-$(|p'|-1)$ prefix of $\mu^{(i)}$ on all but the last bit of p'. By induction, it is straightforward to see that after the ith iteration (the one dealing with $\mu^{(i)}$), the set P is correct for the message set $\{\mu^{(1)}, \ldots, \mu^{(i)}\}$, so the algorithm is correct. Finally, the size of the output set P is at most $(k-1) \cdot Q + 1$, because initially $|P| = 1$, and in each iteration the size of P can grow by at most $k - 1$ elements (up to k can be added, and only if one is removed).

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 exactly as in the signature scheme, and has the form required by 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 $\mathbf{Adv}^{\text{eu-scma}}_{\mathsf{SIG}}(\mathcal{F}) - \text{negl}(n)$. 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 $1/|P| - \text{negl}(n) \geq 1/((k-1) \cdot Q + 1) - \text{negl}(n)$. 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 [GPV08] (which is dual to the scheme of Regev [Reg05]) is well-known. To progress to our HIBE systems, it is enough to simply understand the KEM interface (i.e., the forms of the public/secret keys and ciphertexts) and the conditions on the parameters needed for correct decapsulation.

The scheme KEM is parametrized by a modulus q, dimension m, 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 KEM is used.

- Gen: Choose $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ uniformly at random, $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,s}$ and set $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$. Output public key $pk = \mathbf{A}\|\mathbf{y} \in \mathbb{Z}_q^{n \times (m+1)}$ and secret key $sk = \mathbf{x}$.
- Encaps($pk = \mathbf{A}\|\mathbf{y}$): Choose $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and let
$$\mathbf{b} \leftarrow \mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s}) \quad \text{and} \quad b' \leftarrow \mathsf{Noisy}_\chi(\mathbf{y}^t\mathbf{s} + \kappa \cdot \lfloor q/2 \rfloor),$$
where $\kappa \in \{0, 1\}$ is a uniformly random bit. Output κ and encapsulation $\sigma = \mathbf{b}\|b' \in \mathbb{Z}_q^{m+1}$.
- Decaps($sk = \mathbf{x}, \sigma = \mathbf{b}\|b'$): Compute $b' - \mathbf{x}^t\mathbf{b} \bmod q$ and output 0 if the result is closer to 0 than to $\lfloor q/2 \rfloor$ modulo q, and 1 otherwise.

Observe that, conditioned on the public key $\mathbf{A}\|\mathbf{y}$, the secret decapsulation key $\mathbf{x}$ has distribution $D_{\Lambda^\perp_{\mathbf{y}}(\mathbf{A}),s}$. Later on, our HIBE schemes will generate a decapsulation key for a given $\mathbf{A}$ and $\mathbf{y}$ using $\mathbf{x} \leftarrow \mathsf{SampleD}(\mathbf{S}, \mathbf{A}, \mathbf{y}, s)$, where $\mathbf{S}$ is a suitable basis of $\Lambda^\perp(\mathbf{A})$. By Item 5 of Lemma 2.4, the behavior of Decaps is the same (up to negligible error probability) using such an $\mathbf{x}$.

The following lemma is standard from prior work, e.g., [GPV08].

Lemma 5.1 (Correctness and Security)**.** *Let $m \geq Cn \lg q$ for any fixed constant $C > 1$, let $q \geq 4s(m+1)$, and let χ be the discretized Gaussian with parameter α, where $1/\alpha \geq s\sqrt{m+1} \cdot \omega(\sqrt{\log n})$. Then* Decaps *is correct with overwhelming probability over all the randomness of* Gen *and* 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*,*

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

As explained in [GPV08], the basic single-bit scheme can be amortized to allow for keys of length $\ell = \mathrm{poly}(n)$ bits, with ciphertexts in $\mathbb{Z}_q^{m+\ell}$ and public keys in $\mathbb{Z}_q^{n\times(m+\ell)}$. Here the secret key consists of ℓ independent samples $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^m,s}$ for $i \in [\ell]$, and the public key includes the corresponding syndromes $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$ (along with $\mathbf{A}$ itself). Each bit of the KEM key is concealed using a different syndrome, but using the same $\mathbf{s}$ and 'preamble' $\mathbf{b} \leftarrow \mathsf{Noisy}_\chi(\mathbf{A}^t\mathbf{s})$. Furthermore, it is also possible to conceal $\Omega(\log n)$ encapsulated bits per syndrome by using a smaller error rate α, which yields an amortized expansion factor of $O(1)$. For simplicity, in this work we deal only with encapsulation of single bits, but all of our schemes can be amortized in the manner described above.

We point out one nice property of KEM, which is convenient for the security proof of our HIBE schemes: for any dimensions $m \leq m'$ (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 any m out of m' of their components (i.e., columns of $\mathbf{A}$ and corresponding entries of $\mathbf{b}$).

5.2 A Generalized Hierarchical Scheme

In this section we describe a generalized hierarchical identity-based encryption scheme, called $\mathsf{GENHIBE}$, which we later instantiate in different ways to yield concrete constructions in both the random oracle and standard models. The generalized system captures the essential features of all of our HIBE systems, which are the ability to encrypt to any node in an (implicit) authority tree, and any node's ability to securely delegate a secret key to any of its descendants.

In $\mathsf{GENHIBE}$ there is an identity tree of depth d over some alphabet $\mathcal{ID}$, which is identified with the set $\mathcal{ID}^{\leq d}$ in the natural way: the empty string $\varepsilon \in \mathcal{ID}^{\leq d}$ is the root of the tree, and an identity id' is a descendant of id if and only if id is a proper prefix of id'. Each identity id is implicitly assigned a public key $pk_{id} = \mathbf{A}_{id} \| \mathbf{y}_{id}$ for the KEM scheme described above in Section 5.1. These public keys must have the property that if id' is a descendant of id, then $\mathbf{A}_{id'}$ is an extension of $\mathbf{A}_{id}$ (that is, $\mathbf{A}_{id'} = \mathbf{A}_{id} \| \bar{\mathbf{A}}$ for some $\bar{\mathbf{A}}$). The secret key for an identity id consists of a decapsulation key $\mathbf{x}_{id}$ for its KEM public key $\mathbf{A}_{id} \| \mathbf{y}_{id}$, as well as a relatively short basis $\mathbf{S}_{id}$ for the lattice $\Lambda^\perp(\mathbf{A}_{id})$. (In our instantiations, the short basis is used only for delegation, and so it can be omitted for users at the leaves of the hierarchy.)

In $\mathsf{GENHIBE}$ we deliberately leave out the Setup algorithm, which must be provided by an instantiation. The algorithm should output a master public key mpk and the root-level secret key $(\mathbf{S}_\varepsilon, \mathbf{x}_\varepsilon)$, as well a description of the mapping from an identity id to a KEM public key pk_{id}. These are the only aspects of $\mathsf{GENHIBE}$ that need to be defined when instantiating it. (The $\mathsf{Extract}$, Encaps, and Decaps algorithms are common to all the instantiations, and rely on the assignment of KEM public keys to identities.) Looking ahead, an instantiation of $\mathsf{GENHIBE}$ will assign public keys to identities either via a random oracle, or (in the standard model) by piecing together different parts of the master public key.

The $\mathsf{GENHIBE}$ scheme is parametrized by a few quantities that are indexed by depth within the hierarchy. For an identity at depth $t \geq 0$ (where $t = 0$ corresponds to the root),

- m_t is the maximum possible dimension of the identity's associated lattice, where we require $m_j \leq m_t$ for all $j < t$. (In our instantiations, all identities at a given depth have the same lattice dimension);
- $\widetilde{L_t}$ is an upper bound on the Gram-Schmidt lengths of the user's secret short basis;
- s_t is the Gaussian parameter used to generate the decapsulation key and (when $t \geq 1$) the secret basis, which must exceed $\widetilde{L_j} \cdot \omega(\sqrt{\log n})$ for all $j < t$.

These parameters, along with the maximum length of any possible delegation chain (which is always at most d, but can be less in instantiations), determine the modulus q and error rate α used in the underlying KEM. We instantiate all the parameters after defining the GENHIBE scheme.

Definition of GENHIBE. The algorithms of GENHIBE are as follows:

- $\mathsf{Extract}(sk_{id} = (\mathbf{S}_{id}, \mathbf{x}_{id}), id' = id \| \bar{id})$: if $t' = |id'| > d$, output $\perp$. Otherwise, let $pk_{id'} = \mathbf{A}_{id'} \| \mathbf{y}_{id'}$, where $\mathbf{A}_{id'}$ is an extension of the $\mathbf{A}_{id}$ component of pk_{id} by assumption. Sample a decapsulation key

  $$\mathbf{x}_{id'} \leftarrow \mathsf{SampleD}(\mathsf{ExtBasis}(\mathbf{S}_{id}, \mathbf{A}_{id'}), \mathbf{A}_{id'}, \mathbf{y}_{id'}, s_{t'}),$$

  and a basis

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

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

  (Note that as required by RandBasis, we have $s_{t'} \geq \widetilde{L_t} \cdot \omega(\sqrt{\log n}) \geq \|\widetilde{\mathbf{S}_{id}}\| \cdot \omega(\sqrt{\log n})$, where $t = |id| < t'$. Note also that if $|id'| = t' = d$, we need not generate $\mathbf{S}_{id'}$, because it is only needed for extracting keys for descendants, not decapsulation.)
- $\mathsf{Encaps}(id)$: Output $(\kappa, \sigma) \leftarrow \mathsf{KEM.Encaps}(pk_{id})$.
- $\mathsf{Decaps}(sk_{id} = (\mathbf{S}_{id}, \mathbf{x}_{id}), \sigma)$: Output $\kappa \leftarrow \mathsf{KEM.Decaps}(\mathbf{x}_{id}, \sigma)$.

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

Instantiating the parameters. Suppose that GENHIBE is instantiated in a setting in which Extract is invoked only on identities id' whose lengths are a multiple of some integer $k \geq 1$. (Looking forward, our random-oracle instantiation will use $k = 1$, and in our standard-model schemes k will be the output length of some hash function.) Then it is enough to define s_t and $\widetilde{L_t}$ only for values of t that are multiples of k.

Let $\widetilde{L_0}$ be determined by a concrete instantiation, and let $s_0 = \widetilde{L_0} \cdot \omega(\sqrt{\log n})$. For $t > 0$ that is a multiple of k, let

$$s_t = \widetilde{L_{t-k}} \cdot \omega(\sqrt{\log n}) \qquad \text{and} \qquad \widetilde{L_t} = s_t \cdot \sqrt{m_t} \leq s_t \cdot \sqrt{m_d}.$$

Note that as required, with overwhelming probability $\widetilde{L_t} \geq \|\widetilde{\mathbf{S}_{id}}\|$ when $|id| = t$, by Lemma 3.3. Now unwind the above recurrences to obtain the bounds

$$s_t \leq s_0 \cdot \left(\sqrt{m_d} \cdot \omega(\sqrt{\log n})\right)^{t/k-1} \qquad \text{and} \qquad \widetilde{L_t} \leq \widetilde{L_0} \cdot \left(\sqrt{m_d} \cdot \omega(\sqrt{\log n})\right)^{t/k}.$$

Finally, to satisfy the hypotheses of Lemma 5.1 that are needed for completeness, we can let

$$\begin{aligned} 1/\alpha &\geq s_d \cdot \sqrt{m_d+1} \cdot \omega(\sqrt{\log n}) \\ &= \widetilde{L_0} \cdot \tilde{O}(\sqrt{m_d})^{d/k}, \end{aligned} \tag{5.1}$$

$$\begin{aligned} q &\geq 4s_d \cdot (m_d+1) \\ &= \widetilde{L_0} \cdot \tilde{O}(\sqrt{m_d})^{d/k+1}. \end{aligned} \tag{5.2}$$

(Note also that for each level of the hierarchy it is possible to use a different error rate α_t, which varies inversely with $s_t \cdot m_t$.)

Extensions: Anonymity and chosen-ciphertext security. With a small modification, GENHIBE (or more accurately, our concrete instantiations of it) may be made *anonymous* over all levels of the hierarchy. That is, a ciphertext computationally hides 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 uniformly random and independent elements of $\mathbb{Z}_q$. (The decapsulation algorithm simply ignores the padding.) Anonymity then follows immediately from the pseudorandomness of the LWE distribution.

Security under chosen-ciphertext attack also 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 Full Security in the Random Oracle Model

Here we construct a fully (i.e., aid-ind-cpa) secure HIBE in the random oracle model. (It is also selectively secure under a tighter security reduction.) The scheme can be seen as a generalization of the GPV IBE [GPV08] to hierarchical identities. Essentially, we hash (via a random oracle H) the components of an identity tuple $id = (id_1, \ldots, id_\ell)$ to dimension-m matrices $\mathbf{A}_{i,id}$, which are then concatenated to obtain a KEM public key $\mathbf{A}_{id}$ for the user. In the security proof, the main technical difficulty is that we have to embed a given KEM challenge public key as the user public key $\mathbf{A}_{id^*}$ of the eventual challenge identity id^*, while at the same time being able to simulate user secret keys for all identities on which Extract is queried. The problem is that the simulator does not know the challenge identity id^* in advance when answering random oracle queries. Therefore, our simulator "guesses" id^* (by guessing in which H-query it appears), and "programs" H's outputs on the fly, so that it returns parts of the KEM challenge when queried on the corresponding parts of id^*, and returns controlled matrices with known trapdoors on all other queries. This allows for a straightforward simulation, provided that we guess id^* correctly, but needs a programmable random oracle. We remark that due to this guessing strategy, the security reduction loses a factor of roughly Q_H^d in its advantage, where Q_H is the number of random oracle queries and d is the depth of the hierarchy. (In a selective-identity attack, of course, the simulator knows exactly at which query it should embed the KEM challenge, which leads to a tight reduction.)

We remark that to obtain a fully secure HIBE in the random oracle model, it is also possible to use a generic transformation of Boneh and Boyen [BB04a], which starts from a selectively secure HIBE and applies hash functions to the identities. (We give a standard-model selectively secure HIBE in Section 5.4.1 below.) Compared to this generic transformation, our specific construction is more efficient, mainly because using the random oracle we can directly encode each identity component as a matrix of dimension m, instead of dimension $k \cdot m$ where k is the output length of the random oracle. Overall, then, the dimension of the lattice associated to an identity id at depth ℓ is ℓm, instead of $(\ell+1)km$ when using the generic transformation. (We also remark that in the generic transformation, the security reduction also loses a factor of roughly Q_H^d in advantage, relative to the selectively secure system.)

Definition of ROHIBE**.** Our random-oracle scheme ROHIBE is an instantiation of $\mathsf{GENHIBE}$ from Section 5.2, so we need only specify the Setup algorithm and define how an identity id maps to a KEM public key $\mathbf{A}_{id}\|\mathbf{y}_{id}$. Let $H : \{0,1\}^* \to \mathbb{Z}_q^{n\times m}$ and $G : \{0,1\}^* \to \mathbb{Z}_q^n$ be hash functions modelled as random oracles.

- $\mathsf{Setup}(1^d)$: Generate $(\mathbf{A}_0, \mathbf{S}_0) \leftarrow \mathsf{GenBasis}(1^n, 1^m, q)$, where $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m}$ is negligibly far from uniform, and $\mathbf{S}_0$ is a basis of $\Lambda^\perp(\mathbf{A}_0)$ with $\|\widetilde{\mathbf{S}}\| \le \widetilde{L_0}$. Output $mpk = (\mathbf{A}_0, d)$ and $msk = \mathbf{S}_0$.

For an identity vector $id = (id_1, \ldots, id_t) \in (\{0,1\}^*)^t$ of length $t \le d$, the KEM public key $pk_{id} = \mathbf{A}_{id}\|\mathbf{y}_{id}$ for id is defined using mpk and the hash functions H, G as:

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

where $\mathbf{A}_i = H(id_1, \ldots, id_i) \in \mathbb{Z}_q^{n\times m}$.

For technical reasons related to the proof, we also need to modify the $\mathsf{Extract}$ algorithm (from $\mathsf{GENHIBE}$) so that the same $\mathbf{x}_{id'}$ is returned every time identity id' is queries. This means that the actual algorithm should either be made stateful, or should use a PRF in the standard way to get repeatable randomness. Alternatively, $\mathbf{x}_{id'}$ can itself be computed using $\mathbf{S}_{id'}$, at the expense of using a slightly larger parameter in the $\mathsf{SampleD}$ algorithm, i.e., by running $\mathbf{x}_{id'} \leftarrow \mathsf{SampleD}(\mathbf{S}_{id'}, \mathbf{A}_{id'}, \mathbf{y}_{id'}, \|\widetilde{\mathbf{S}_{id'}}\| \cdot \omega(\sqrt{\log n}))$.

Instantiating the parameters. In this instantiation of $\mathsf{GENHIBE}$, the lattice dimension for an identity at depth t is $m_t = (t+1)\cdot m$, and the initial Gram-Schmidt bound $\widetilde{L_0} = O(\sqrt{n \log q})$. Using the bound $m = O(n \log q)$, we can therefore set the parameters q, α using Equations (5.1) and (5.2) so that

$$\log q = \tilde{O}(d) \qquad \text{and} \qquad 1/\alpha = \sqrt{dn} \cdot \tilde{O}(d\sqrt{n})^d.$$

Theorem 5.2 (Security of ROHIBE)**.** *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking* KEM *(instantiated with dimension dm and q, α as above) 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,*

$$\mathbf{Adv}_{\mathsf{KEM}}(\mathcal{S}^{\mathcal{A}}) \ge \mathbf{Adv}^{\textit{aid-ind-cpa}}_{\mathsf{ROHIBE}}(\mathcal{A})/(dQ_H^{d-1}Q_G) - \mathrm{negl}(n).$$

Proof. Let $\mathcal{A}$ be an adversary mounting an 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 $\mathbf{b}\|b' \in \mathbb{Z}_q^{dm} \times \mathbb{Z}_q$, and a bit κ which either is encapsulated by $\mathbf{b}\|b'$ or is uniform and independent of everything else; the goal of $\mathcal{S}$ is to determine which is the case.

Let Q_G and Q_H be the numbers of 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 for each length of the input tuple. 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 game 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. Choose a vector $\mathbf{j}^* = (j_1^*, \ldots, j_{t^*-1}^*) \leftarrow \{1, \ldots, Q_H\}^{t^*-1}$ and an index $j^* \leftarrow \{1, \ldots, Q_G\}$. (These guesses means that $\mathcal{S}$ hopes that the j_i^*-th distinct query of length $1 \le i \le t^*-1$ to $H(\cdot)$ is a prefix of the challenge identity and that the j^*-th distinct query to $G(\cdot)$ is the challenge identity.)

- *Parse 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 answers to oracle queries to H and G, queries for user secret keys, and it must also generate the challenge encapsulation. To do this, it maintains 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. On $\mathcal{A}$'s j_i-th distinct query $(id_{j_i,1}, \ldots, id_{j_i,i})$ of length i to $H(\cdot)$, 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. 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 $\mathbf{x}_j \leftarrow D_{\mathbb{Z}^m, s_t}$ (where t is the depth of id_j) and set $\mathbf{y}_j := \mathbf{A}_{(id_{j,1}, \ldots, id_{j,t-1})} \cdot \mathbf{x}_j \in \mathbb{Z}_q^n$. (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 $(id_j, \mathbf{y}_j, \mathbf{x}_j)$ in list $\mathcal{G}$.

Queries to $\mathsf{Extract}$. 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 some $i \in [t-1]$, $\mathbf{A}_{id_i} = H(id_1, \ldots, id_i)$ is contained in list $\mathcal{H}$, then $\mathcal{S}$ 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 $\mathcal{S}$ can retrieve a properly distributed vector $\mathbf{x}_{id}$ from $\mathcal{G}$ satisfying $\mathbf{A}_{(id_1, \ldots, id_{t-1})} \mathbf{x}_{id} = \mathbf{y}_{id}$. If the generation of $sk_{id} = (\mathbf{S}_{id}, \mathbf{x}_{id})$ was successful, then $\mathcal{S}$ returns sk_{id}. In all other cases, $\mathcal{S}$ 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 $\mathbf{b}^* \| b'$ and the key bit κ.

$\mathcal{S}$ 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 far from 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 Item 4 of Lemma 2.4. Due to the truncation property of KEM, 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 $\mathcal{S}$ does not abort; this is at least $1/(dQ_G Q_H^{d-1}) - \operatorname{negl}(n)$, because its choice of t^*, j^*, and $\mathbf{j}^*$ are perfectly hidden conditioned on the view of $\mathcal{A}$ (up to the $\operatorname{negl}(n)$ statistical error of $\mathcal{S}$'s simulation). □

5.4 Standard-Model Constructions

Here we give HIBE constructions in the standard model that are secure under both selective and fully adaptive attacks. In Section 5.4.1 we describe a binary tree encryption (BTE) system which is at the heart of all our various HIBE constructions, and prove its selective security in Section 5.4.2. In Section 5.4.3 and beyond we extend our HIBE to full adaptive security.

5.4.1 Binary Tree Encryption

Here we instantiate $\mathsf{GENHIBE}$ to obtain a standard-model HIBE for a restricted identity hierarchy in the form of a binary tree, i.e., with identity alphabet $\mathcal{ID} = \{0,1\}$. This is essentially the notion of "binary tree encryption" (BTE) from [CHK03]. Later on we realize selectively and adaptively secure HIBE schemes for large identity alphabets $\mathcal{ID}$, by mapping them onto a binary tree.

Definition of BTE**.** We need only define the Setup algorithm and specify how it defines a public mapping from identities in $\mathcal{ID}^{\leq d} = \{0,1\}^{\leq d}$ to KEM public keys.

- $\mathsf{Setup}(d)$: Generate $(\mathbf{A}_0, \mathbf{S}_0) \leftarrow \mathsf{GenBasis}(1^n, 1^m, q)$ (see Proposition 3.1), where $\mathbf{A}_0 \in \mathbb{Z}_q^{n\times m}$ is negligibly close to uniform and $\mathbf{S}_0$ is a basis of $\Lambda^\perp(\mathbf{A}_0)$ with $\|\widetilde{\mathbf{S}_0}\| \leq \widetilde{L_0}$.

  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, and choose $\mathbf{x}_\varepsilon \leftarrow \mathsf{SampleD}(\mathbf{S}_0, \mathbf{A}_0, \mathbf{y}, s_0)$. Output $mpk = (\mathbf{A}_0, \{\mathbf{A}_j^{(b)}\}, \mathbf{y}, d)$ and $sk_\varepsilon = (\mathbf{S}_0, \mathbf{x}_\varepsilon)$.

For an identity $id = (id_1, \ldots, id_t) \in \{0,1\}^t$ of length $t = |id| \leq d$, the KEM public key $pk_{id} = \mathbf{A}_{id} \| \mathbf{y}_{id}$ for id is defined relative to mpk as

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

Instantiating the parameters. For this instantiation of $\mathsf{GENHIBE}$, the lattice dimension for an identity at depth t is $m_t = (t+1)m$, and the initial Gram-Schmidt bound $\widetilde{L_0} = O(\sqrt{n\log q})$. As in Section 5.2, suppose that BTE is used in a setting where $\mathsf{Extract}$ is called only on identities of length divisible by some integer k. Then letting $d' = \lfloor d/k \rfloor$ and using the bound $m = O(n\log q)$, using Equations (5.1) and (5.2) we can set the parameters q, α so that

$$\log q = \tilde{O}(d') \qquad \text{and} \qquad 1/\alpha = \sqrt{d'n} \cdot \tilde{O}\left(d'\sqrt{kn}\right)^{d'}.$$

HIBE from BTE. We recall from [CHK03] how to obtain HIBE for any identity space $\mathcal{ID}$ from any BTE. The HIBE master public key is simply a BTE public key, plus a hash function $h\colon \mathcal{ID} \to \{0,1\}^k$ drawn at random from a family of universal one-way (or collision resistant) hash functions. Each HIBE identity $id = (id_1, \ldots, id_\ell) = \mathcal{ID}^{\leq d}$ is mapped to the BTE identity $(h(id_1), \ldots, h(id_\ell)) \in (\{0,1\}^k)^\ell$. Therefore, to emulate a HIBE hierarchy of depth d' requires a BTE hierarchy of depth $d = d'k$, and $\mathsf{Extract}$ in the BTE is only ever called on identities of length divisible by k. It is easy to verify that if the BTE is selectively (sid-ind-cpa) secure, then so is the resulting HIBE.

5.4.2 Selective Security of BTE

Recall that in a selective-identity (sid-ind-cpa) attack, the adversary first has to declare its challenge identity before receiving the master public key mpk. In the security proof, the simulator can therefore set up mpk to depend on the challenge identity. In our simulation, all matrices in mpk will either be undirected, or controlled by the simulator. As we have seen, a matrix $\mathbf{A} = \mathbf{A}_1 \| \ldots \| \mathbf{A}_\ell$ is undirected if all $\mathbf{A}_i$ are undirected, while $\mathbf{A}$ is controlled if and only if at least one $\mathbf{A}_i$ is controlled. In the simulation, $\mathbf{A}_0$ will be undirected, so a user secret key for identity id can be constructed if and only if at least one of the $\mathbf{A}_i^{(id_i)}$ is controlled. We will set up these matrices so that: (1) for all id that can appear in $\mathsf{Extract}$ queries (i.e., those which differ from id^* in at least one position), $\mathbf{A}_{id}$ is controlled, and (2) for the challenge identity id^*, $\mathbf{A}_{id^*}$ is undirected.

Theorem 5.3 (Security of BTE)**.** *There exists a PPT oracle algorithm (a reduction) $\mathcal{S}$ attacking* KEM *(instantiated with dimension $(d+1)m$ and q, α as above) such that, for any adversary $\mathcal{A}$ mounting a sid-ind-cpa attack on* BTE*,*

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

Proof. We construct a reduction $\mathcal{S}$ attacking KEM, which has oracle access to an adversary $\mathcal{A}$ mounting a sid-ind-cpa attack on BTE. The reduction is given a uniformly random KEM public key $pk = \mathbf{A} \| \mathbf{y} \in \mathbb{Z}_q^{n \times (d+1)m} \times \mathbb{Z}_q^n$, an encapsulation $\mathbf{b} \| b' \in \mathbb{Z}_q^{(d+1)m} \times \mathbb{Z}_q$, and a bit κ which either is encapsulated by $\mathbf{b} \| b'$ or is uniform and independent of everything else; 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 and receives its challenge identity id^* of length $t^* = |id^*| \in [d]$. Then $\mathcal{S}$ produces a master public key mpk, encapsulation, 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 $t^* < d$, then 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 $\mathbf{b}^* \| b'$, and the key bit κ.

Then $\mathcal{S}$ answers each $\mathsf{Extract}$ 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 $(\mathbf{S}_{id}, \mathbf{x}_{id})$, which are computed by

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

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

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

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, the answers to $\mathsf{Extract}$ queries are negligibly close to those in the real attack, 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 $\mathsf{RandBasis}$ and $\mathsf{SampleD}$ as above. Finally, the encapsulation $\mathbf{b}^* \| b'$ (for identity id^*) and key bit κ 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. $\square$

5.4.3 Fully Secure HIBE

Our fully secure HIBE is essentially our BTE scheme, with identity components mapped to binary strings via a suitable kind of hash function (not just a UOWHF, which was sufficient for selectively secure HIBE). An identity $id = (id_1, \ldots, id_\ell) \in \mathcal{ID}^{\leq d}$ maps to a BTE identity $(H_1(id_1), \ldots, H_\ell(id_\ell)) \in (\{0,1\}^\lambda)^\ell$ for suitable hash functions H_i specified in the public key.

As before, we need the simulator to embed an undirected KEM challenge into the master public key so that it corresponds to the public key $\mathbf{A}_{id^*}$ of the target identity, while also having enough controlled matrices to allow for answering Extract queries. In a fully adaptive attack, the adversary may choose the challenge identity id^* dynamically and adaptively, making this a difficult balance to strike. To overcome this obstacle, we employ a probabilistic argument along the lines of the one in [BB04a]. Concretely, we set up the master public key such that each $\mathbf{A}_{id}$ is totally undirected with a certain probability. A sophisticated construction of the hash functions H_i will ensure that the corresponding events are sufficiently independent. That is, even an adversary that adaptively asks for user secret keys cannot manage to produce an identity id for which $\mathbf{A}_{id}$ is *guaranteed* to be either undirected or controlled. By setting the probabilities in the right way (depending on the adversary's complexity), we can ensure that with significant probability, all $\mathbf{A}_{id}$ associated with Extract queries are controlled, while $\mathbf{A}_{id^*}$ is undirected. In this event, 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 this independence, we will employ an "artificial abort" strategy similar to the one from [Wat05].

Definition of SMHIBE. Let $\mathcal{H} = (\mathcal{H}_n)_n$ be a family of hash functions $H \colon \{0,1\}^n \to \{0,1\}^\lambda$, whose properties are described below in Section 5.4.4.

Our standard-model HIBE scheme SMHIBE is just the BTE scheme from Section 5.4.1, implemented with a suitable mapping from HIBE to BTE identities. Concretely, to obtain a HIBE scheme that supports identities $id = (id_1, \ldots, id_\ell) \in (\{0,1\}^n)^\ell$ of up to d levels, we will add to BTE's master public key d hash functions $H_1, \ldots, H_d$. A HIBE identity $(id_1, \ldots, id_\ell)$ then maps to the binary string (i.e., BTE identity)

$$(H_1(id_1), \ldots, H_\ell(id_\ell)) \in \{0,1\}^{\lambda \cdot \ell}.$$

For convenience, we restate the entire modified Setup algorithm; we also slightly modify the notation so it is easier to see how the pieces of mpk are assembled to generate user public keys.

- $\mathsf{Setup}(1^d)$: Using Proposition 3.1, generate $\mathbf{A}_0 \in \mathbb{Z}_q^{n \times m}$ and a corresponding short basis $\mathbf{S}_0 \in \mathbb{Z}^{m \times m}$ with $\|\widetilde{\mathbf{S}_0}\| \leq \widetilde{L_0}$. Also sample uniformly random and independent matrices $\mathbf{B}_{i,u,b} \in \mathbb{Z}_q^{n \times m}$ for $i \in [d]$, $u \in [\lambda]$, $b \in \{0,1\}$, and a vector $\mathbf{y} \in \mathbb{Z}_q^n$. Finally, choose $H_1, \ldots, H_d \leftarrow \mathcal{H}_n$. Output

$$mpk = (\mathbf{A}_0, \mathbf{y}, \{\mathbf{B}_{i,u,b}\}_{(i,u,b) \in [d] \times [\lambda] \times \{0,1\}}, \{H_i\}_{i \in [d]}) \quad \text{and} \quad msk = (mpk, \mathbf{S}_0).$$

For a HIBE identity $id = (id_1, \ldots, id_\ell) \in (\{0,1\}^n)^\ell$, we define

$$\begin{aligned} \mathbf{A}_{id} &:= \mathbf{A}_0 \| \mathbf{A}_{1,id_1} \| \ldots \| \mathbf{A}_{\ell,id_\ell} \in \mathbb{Z}_q^{n \times (\lambda\ell+1)m} \\ \text{where} \quad \mathbf{A}_{i,id_i} &:= \mathbf{B}_{i,1,t_1} \| \ldots \| \mathbf{B}_{i,\lambda,t_\lambda} \in \mathbb{Z}_q^{n \times \lambda m}, \end{aligned} \tag{5.3}$$

where $(t_1, \ldots, t_\lambda) := H_i(id_i) \in \{0,1\}^\lambda$. Note that the user secret key for an identity id therefore consists of a basis part $\mathbf{S}_{id}$ for $\Lambda^\perp(\mathbf{A}_{id})$ and a decapsultion key $\mathbf{x}_{id}$ satisfying $\mathbf{A}_{id}\mathbf{x}_{id} = \mathbf{y}$. For brevity, we will write $id|\ell := (id_1, \ldots, id_\ell)$ for $\ell \leq |id|$.

Instantiating the parameters. In our scheme SMHIBE, each resulting BTE identity component id_i is a λ-bit string, and thus leads to $k = \lambda$ matrices of dimension m, which are appended to form $\mathbf{A}_{id}$. Furthermore, the root matrix $\mathbf{A}_\varepsilon$ is of dimension m. With a maximal depth of d, each individual KEM public key matrix $\mathbf{A}_{id}$ is therefore of dimension at most $(d\lambda + 1)m$. In particular, we can use parameters as in Section 5.4.1, with $k = \lambda$.

5.4.4 Admissible hash functions

Definition. To achieve full security, we will require a hash function (mapping identities to bit strings) with certain special properties. Concretely, we use a variant of the admissible hash functions from [BB04b]. 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

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

For $\mu \in \{0,\dots,\lambda\}$, denote by $\mathcal{K}_\mu$ the uniform distribution on all keys $K \in \{0,1,\perp\}^\lambda$ having exactly μ non-$\perp$ components. For a function $\Delta : \mathbb{N}^2 \to \mathbb{R}$, we say that $\mathcal{H}$ is Δ-*admissible* if for every polynomially bounded $Q = Q(n)$, 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

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

  is negligible in n.

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

  $$\Pr[F_{K,H}(x_0) = \mathtt{UN} \wedge F_{K,H}(x_1) = \dots = F_{K,H}(x_Q) = \mathtt{CO}] \geq \Delta(n,Q),$$

  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 $\Delta = \Delta(n,Q)$ which is significant for every polynomial $Q = Q(n)$.

Intuition. We will eventually use H on input an identity component $id \in \mathcal{C}_n$. The output $t = H(id) \in \{0,1\}^\lambda$ will determine a selection $\mathbf{B}_{1,t_1},\dots,\mathbf{B}_{\lambda,t_\lambda}$ out of 2λ matrices $\mathbf{B}_{u,b}$ (for $(u,b) \in [\lambda] \times \{0,1\}$). The vector $K \in \{0,1,\perp\}^\lambda$ indicates for which $\mathbf{B}_{u,b}$ we know a trapdoor during our security proof. (Concretely, we will know a trapdoor for $\mathbf{B}_{u,K_u}$; if $K_u = \perp$, then we will not know a trapdoor for either $\mathbf{B}_{u,0}$ or $\mathbf{B}_{u,1}$.) Using this setup, $F_{K,H}(id) = \mathtt{CO}$ means that we know a trapdoor for at least one of the matrices $\mathbf{B}_{u,t_u}$ appearing in

$$\mathbf{B} := \mathbf{B}_{1,t_1} \| \dots \| \mathbf{B}_{\lambda,t_\lambda},$$

where $t = H(id)$. Hence, we can derive a trapdoor for $\mathbf{B}$ using Lemma 3.2, and we can say that $\mathbf{B}$ is *controlled*. If $F_{K,H}(id) = \mathtt{UN}$, then we can hope to embed an *undirected* KEM challenge into this matrix. (This also explains the notation of F-images.)

In the definition of admissibility, we consider adversarially chosen identities $x_0,\dots,x_Q$, where x_0 is the challenge identity, and the x_i (for $i \geq 1$) are identities from user secret key queries. We will be interested in the case that x_0 is mapped to an undirected matrix (i.e., $F_{K,H}(x_0) = \mathtt{UN}$), while the other x_i are mapped to controlled matrices (i.e., $F_{K,H}(x_i) = \mathtt{CO}$). The second condition of admissibility requires that the probability that this happens is at least Δ for a suitable choice of keys K and "good" query vectors $\mathbf{x}$. The first condition states that it is computationally infeasible to find "bad" query vectors. (In the concrete construction mentioned below, "bad" query vectors correspond to collisions of an underlying cryptographic hash function.) Note that the first condition ("bad" vectors are infeasible to find) is computational, while the second condition is purely statistical.

Taken together, admissibility of $\mathcal{H}$ hence simply means that with significant probability, (1) all user secret key queries of an adversary attacking SMHIBE refer to controlled matrices $\mathbf{B}$, while (2) the (adversarially selected) challenge identity is mapped to an undirected matrix $\mathbf{B}$.

We stress that while the *security reduction* depends on the parameter Δ (and thus on the number Q of user secret key queries), the hash function itself does not. In particular, the resulting HIBE scheme does not make any assumptions about Q (except, of course, that Q is bounded by *some* a-priori unknown polynomial).

Difference from the definition of [BB04b]. Note that our definition of admissibility is conceptually different from that of [BB04b]. The reason for the 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 advantage) 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 useful only with parameters that cause the distinguishing advantage to become non-negligible (but still 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, by purely combinatorial bounds on the probability of the 'simulation-enabling' event

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

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 [BB04b] does not incorporate an artificial abort step, this actually would have been necessary to guarantee sufficient independence in (their version of) the event good. This becomes an issue in [BB04b, Claim 2], when the success probability of an adversary conditioned on good is related to its original (unconditioned) success probability.

Constructions. In [BB04b] it is shown how to construct admissible hash functions from a given collision-resistant hash function family. And since collision-resistant hash functions can be built from the SIS problem (which is no easier than the LWE problem), this does not entail extra assumptions in the encryption context. While the construction from [BB04b] is proved to satisfy the definition of admissibility from that work, the proof actually transfers directly to our definition as well. For parameter choices as in [BB04b, Section 5.3], we get a single hash function with output length $\lambda = O(n^{2+\varepsilon})$ (for arbitrary $\varepsilon > 0$) that is Δ-admissible with $\Delta = \Theta(1/Q^2)$.[5]

[5] In the notation of [BB04b], 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.

5.4.5 Full security of SMHIBE

If the hash function family $\mathcal{H}$ is admissible, then we can prove full security. Unfortunately, we only know constructions of admissible hash functions that require $\lambda = O(n^{2+\varepsilon})$, so the resulting scheme is not very practical. We base the security of SMHIBE on that of the KEM scheme of Section 5.1 (with $q' = q$, $n' = n$, and $m' = 2d\lambda m$).

Theorem 5.4. *Let $\mathcal{A}$ be an adversary mounting an aid-ind-cpa attack on* SMHIBE *that makes at most $Q(n)$* Extract *queries. Then for every polynomial $S = S(n)$, there exists an adversary $\mathcal{S}^{\mathcal{A}}$ on* KEM*'s ind-cpa security, and an adversary $\mathcal{C}$ on $\mathcal{H}$'s admissibility such that*

$$\mathbf{Adv}^{\textit{aid-ind-cpa}}_{\mathsf{SMHIBE}}(\mathcal{A}) \leq d \cdot \mathbf{Adv}^{\textit{adm}}_{\mathcal{H}}(\mathcal{C}) + \frac{\mathbf{Adv}^{\textit{ind-cpa}}_{\mathsf{KEM}}(\mathcal{S}^{\mathcal{A}})}{\Delta(n,Q)^d} + \frac{1}{S(n)} + \mathrm{negl}(n). \tag{5.4}$$

The running time of $\mathcal{C}$ is roughly that of the aid-ind-cpa experiment with $\mathcal{A}$, and the running time of $\mathcal{S}^{\mathcal{A}}$ 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 [BB04b], $\Delta(n,Q)^d = \Theta(1/Q^{2d})$ is inverse polynomial for any fixed d. Since S in Theorem 5.4 is arbitrary, we obtain:

Corollary 5.5. *If $\mathcal{H}$ is admissible, and if* KEM *is ind-cpa secure, then* SMHIBE *is aid-ind-cpa-secure.*

5.4.6 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 $id^j = (id^j_1, \dots, id^j_{\ell_j})$ (for $1 \leq j \leq Q$), and the challenge identity chosen by $\mathcal{A}$ as $id^* = (id^*_1, \dots, id^*_{\ell^*})$. By out_i, we denote the experiment's output in Game i. By definition,

$$|\Pr[out_0 = 1] - 1/2| = \mathbf{Adv}^{\text{aid-ind-cpa}}_{\mathsf{SMHIBE}}(\mathcal{A}). \tag{5.5}$$

In the following, for every $i \in [d]$, 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 (but arbitrary) way with unused identities.

Concretely, for all i, let $\mathcal{ID}^{\mathtt{UN}}_i \supseteq \mathcal{ID}^*_i$ and $\mathcal{ID}^{\mathtt{CO}}_i \supseteq \mathcal{ID}^Q_i \setminus \mathcal{ID}^*_i$ be disjoint, and such that $|\mathcal{ID}^{\mathtt{UN}}_i| = 1$ and $|\mathcal{ID}^{\mathtt{CO}}_i| = Q$. Intuitively, the (singleton) sets $\mathcal{ID}^{\mathtt{UN}}_i$ contain those identities which we need to map to undirected matrices, while the $\mathcal{ID}^{\mathtt{CO}}_i$ should map to controlled matrices.

Furthermore, let $\overrightarrow{\mathcal{ID}_i} \in (\{0,1\}^n)^{Q+1}$ be the vector whose first component is the (unique) element of $\mathcal{ID}^{\mathtt{UN}}_i$, and whose remaining Q components are the elements of $\mathcal{ID}^{\mathtt{CO}}_i$ (in some canonical order). Intuitively, $\overrightarrow{\mathcal{ID}_i} \in (\{0,1\}^n)^{Q+1}$ plays the role of the input vector $\mathbf{x}$ in the definition of admissibility. (That is, we will be interested in the case that its first component is mapped to an undirected matrix, while the other components are mapped to controlled matrices.)

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. Here, bad_{H_i} denotes the set of "bad $\mathcal{H}$-queries" that comes with the admissibility of $\mathcal{H}$. In the concrete admissible hash function mentioned above, e.g., the set bad_{H_i} corresponds to collisions of an underlying cryptographic hash function. A straightforward reduction shows

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

for a suitable adversary $\mathcal{C}$ on $\mathcal{H}$'s admissibility. The factor of d comes from the fact that we are actually using d different admissible hash functions, one for each level.

In **Game** 2, after the adversary has terminated, we let an event good_2 occur independently with probability Δ^d. (That is, we independently toss a coin with bias Δ^d, and let good_2 be the event that the coin comes out 1.) We abort the experiment (and output a uniformly random bit) if $\neg\mathsf{good}_2$ occurs. We get

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

Intuitively, this seemingly meaningless and unforced artificial abort step paves the way for our later refinements. In particular, we will later on have to deal with a (forced) abort event which can happen with a probability *up to* (but not necessarily exactly) Δ^d. However, the adversary's output distribution should be (at least approximately) independent of the abort event. Hence, we will fix the probability for an abort to Δ^d already at this point, and later on take care that the abort always happens with (roughly) the same probability.

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

$$F_i(id) = F_{K^i,H_i}(id) = \begin{cases} \mathtt{CO} & \text{if } \exists u \in \{1,\dots,\lambda\} : t_u = K_u^i \\ \mathtt{UN} & \text{if } \forall u \in \{1,\dots,\lambda\} : t_u \neq K_u^i \end{cases}$$

for $(t_1, \dots, t_\lambda) = H_i(id)$. Here, for every i, $K^i = (K_1^i, \dots, K_\lambda^i) \in \{0, 1, \perp\}^\lambda$ is initially chosen by the experiment uniformly among all $K \in \{0, 1, \perp\}^\lambda$ with exactly μ non-$\perp$ components. If $F_i(id) = \mathtt{CO}$, then id is *controlled*, and if $F_i(id) = \mathtt{UN}$, then id is *undirected*. Later on, controlled identities will correspond to controlled matrices, while undirected identities will correspond to undirected matrices.

Let E_i denote the event that in Game 3, $F_i(id) = \mathtt{CO}$ for all $id \in \mathcal{ID}_i^{\mathtt{CO}}$ and $F_i(id) = \mathtt{UN}$ for all $id \in \mathcal{ID}_i^{\mathtt{UN}}$. 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 $1/2^n$,

$$|\Pr[\mathsf{good}_3] - \Pr[\mathsf{good}_2]| = \left|\Delta^d - \Delta^d \frac{p_E}{\tilde{p_E}}\right| = \Delta^d \left|\frac{\tilde{p_E} - p_E}{\tilde{p_E}}\right| \leq \Delta^d \frac{\Delta^d}{S\tilde{p_E}} \leq \frac{\Delta^d}{S}. \tag{5.6}$$

Since (5.6) 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,

$$|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leq \frac{\Delta^d}{S} + \frac{1}{2^n}. \tag{5.7}$$

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:

- $\mathbf{A}_0$ 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

$$|\Pr[out_4 = 1] - \Pr[out_3 = 1]| = \mathrm{negl}(n). \tag{5.8}$$

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 $F_i(id_i) = \mathtt{UN}$ for all i. This change is purely conceptual: since id is not a prefix of the challenge identity id^*, there is an i with $id_i \in \mathcal{ID}_i^{\mathtt{CO}} \supseteq \mathcal{ID}_i^{Q} \setminus \mathcal{ID}_i^*$. But since $F_i(id_i) = \mathtt{UN}$ for this i, there is a partial identity $id_i \in \mathcal{ID}^{\mathtt{CO}}$ which *should* map to a controlled matrix but does not (since actually $F_i(id_i) = \mathtt{UN}$). This implies $\neg E_i$ according to the definition of E_i in Game 3, and thus $\neg E$. But $\neg E$ implies that we would have aborted anyway, so our additional abort criterion from Game 5 is purely conceptual. We get

$$\Pr[out_5 = 1] = \Pr[out_4 = 1]. \tag{5.9}$$

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 $F_i(id_i) = \mathtt{CO}$ 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 (5.3)) 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 $\mathbf{A}_0$ 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 $usk_{id} = (\mathbf{S}_{id}, \mathbf{x}_{id})$, up to negligible statistical distance. Hence

$$|\Pr[out_6 = 1] - \Pr[out_5 = 1]| = \mathrm{negl}(n). \tag{5.10}$$

In **Game** 7, we set up $\mathbf{A}_0$ as undirected instead of controlled. (Note that since Game 6, we do not need a short basis of $\mathbf{A}_0$ anymore to generate user secret keys.) Again, by Proposition 3.1, this change affects the distribution of the public key only negligibly. We get

$$|\Pr[out_7 = 1] - \Pr[out_6 = 1]| = \mathrm{negl}(n). \tag{5.11}$$

We finally claim that

$$\Pr[out_7 = 1] - 1/2 = \mathbf{Adv}_{\mathsf{KEM}}^{\text{ind-cpa}}(\mathcal{S}^{\mathcal{A}}) \tag{5.12}$$

for our LWE-based KEM KEM with parameters of $q' = q$, $n' = n$, and $m' = (2d\lambda + 1)m$, and the following adversary $\mathcal{S}^{\mathcal{A}}$. Recall that $\mathcal{S}^{\mathcal{A}}$ is given a KEM public key $\overline{pk}$, which it parses as

$$\overline{pk} \;=\; \overline{\mathbf{A}} \| \overline{\mathbf{y}} \;=\; \overline{\mathbf{A}_0} \| \overline{\mathbf{B}_{1,1,0}} \| \ldots \| \overline{\mathbf{B}_{d,\lambda,1}} \| \overline{\mathbf{y}} \;\in\; \mathbb{Z}_q^{n \times (2d\lambda+1)m+1}$$

with $\overline{\mathbf{A}_0}, \overline{\mathbf{B}_{i,u,b}} \in \mathbb{Z}_q^{n \times m}$ and $\overline{\mathbf{y}} \in \mathbb{Z}_q^m$. Furthermore, $\mathcal{S}^{\mathcal{A}}$ receives an encapsulation $\overline{\sigma^*} = \overline{\mathbf{b}^*} \| \overline{b'}$ with

$$\begin{pmatrix} \overline{\mathbf{b}_0^*} \\ \hline \mathbf{b}_{1,1,0}^* \\ \vdots \\ \hline \mathbf{b}_{d,\lambda,1}^* \end{pmatrix} := \overline{\mathbf{b}^*} \;\leftarrow\; \mathsf{Noisy}_\chi \begin{pmatrix} \overline{\mathbf{A}_0}^t \mathbf{s} \\ \hline \mathbf{B}_{1,1,0}^t \mathbf{s} \\ \vdots \\ \hline \mathbf{B}_{d,\lambda,1}^t \mathbf{s} \end{pmatrix}, \qquad \overline{b'} \;\leftarrow\; \mathsf{Noisy}_\chi(\overline{\mathbf{y}}\mathbf{s} + \overline{\kappa^*} \cdot \lfloor q/2 \rfloor) \tag{5.13}$$

for a uniformly chosen KEM key $\overline{\kappa^*} \in \{0,1\}$. Finally, $\mathcal{S}^\mathcal{A}$ receives a challenge $\overline{\kappa}$, which is either equal to $\overline{\kappa^*}$, or an independently and uniformly random bit.

Now $\mathcal{S}^\mathcal{A}$ sets up an environment for $\mathcal{A}$ as in Game 7. First, $\mathcal{S}^\mathcal{A}$ prepares an SMHIBE master public key mpk. In this, $\mathcal{S}^\mathcal{A}$ embeds its own challenge matrices $\overline{\mathbf{A}_0}$ and $\overline{\mathbf{B}_{i,u,b}}$ as the undirected matrices of mpk. Concretely, $\mathcal{S}^\mathcal{A}$ sets $\mathbf{A}_0 := \overline{\mathbf{A}_0}$, and $\mathbf{B}_{i,u,b} := \overline{\mathbf{B}_{i,u,b}}$ whenever $b \neq K^i_u$. For $b = K^i_u$, $\mathcal{S}^\mathcal{A}$ chooses $\mathbf{B}_{i,u,b}$ controlled as in Game 7. With this setup, $\mathcal{S}^\mathcal{A}$ can perfectly play Game 7 with $\mathcal{A}$; in particular, $\mathcal{S}^\mathcal{A}$ can answer all of $\mathcal{A}$'s user secret key queries (or abort if necessary as in Game 7).

It remains to embed $\mathcal{S}^\mathcal{A}$'s own challenge $\overline{\sigma^*}$ into the SMHIBE challenge encapsulation $\sigma^* = \mathbf{b}^* \| b'$ for identity $id^* = (id^*_1, \ldots, id^*_{\ell^*})$. Since otherwise Game 7 aborts with uniform output, we may assume that E occurs. Hence, for all i, for $(t^i_1, \ldots, t^i_\lambda) = H_i(id^*_i)$, and all u, we have $K^i_u \neq t^i_u$. Thus, we can write

$$\mathbf{A}_{id^*} \;=\; \mathbf{A}_0 \| \mathbf{B}_{1,1,t^1_1} \| \ldots \| \mathbf{B}_{\ell^*,\lambda,t^{\ell^*}_\lambda} \;=\; \overline{\mathbf{A}_0} \| \overline{\mathbf{B}_{1,1,t^1_1}} \| \ldots \| \overline{\mathbf{B}_{\ell^*,\lambda,t^{\ell^*}_\lambda}}$$

as a concatenation undirected matrices that are part of $\overline{\mathbf{A}}$. This allows to construct σ^* from $\overline{\sigma^*}$ (as given by (5.13)) through $b' := \overline{b'}$ and

$$\mathbf{b}^* := \begin{pmatrix} \overline{\mathbf{b}^*_0} \\ \overline{\mathbf{b}^*_{1,1,t^1_1}} \\ \vdots \\ \overline{\mathbf{b}^*_{\ell^*,\lambda,t^{\ell^*}_\lambda}}. \end{pmatrix}$$

Observe that this makes σ^* an SMHIBE encapsulation of $\overline{\kappa^*}$ (from (5.13)). Hence, we have a direct correspondence between SMHIBE and KEM challenges. Thus, we can set the SMHIBE challenge key as $\kappa := \overline{\kappa}$, and let $\mathcal{S}^\mathcal{A}$ output whatever $\mathcal{A}$ eventually outputs. This finally yields (5.12).

Taking (5.5-5.12) together shows (5.4).

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 [Wat05].) 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 would not need this approximation/abort step at all, since (5.6) and hence (5.7) followed directly by these better bounds. The good news is that the analysis of $\mathcal{H}$ from [BB04b] provides such better bounds for $\Pr[E]$, resp. p_E. The bad news is that this comes at a price: using the analysis of [BB04b], $|p_E - \Delta^d|/\Delta^d$ depends in an inversely polynomial way on Q, the number of user secret key queries. Hence, to achieve $|p_E - \Delta^d| < \Delta^d/S$, we would need to choose fixed $\mathcal{H}$ parameters that work for *arbitrary* polynomial values of Q. 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 typically 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 slower reduction. This is why we decided for an artificial abort step.

Acknowledgments

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

References

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

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

[ABC+05] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptology*, 21(3):350–391, 2008. Preliminary version in CRYPTO 2005.

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

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

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

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

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

[BBDP01] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, pages 566–582. 2001.

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

[BCOP04] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522. 2004.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Preliminary version in CRYPTO 2001.

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

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

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

[CHK03] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *J. Cryptology*, 20(3):265–294, 2007. Preliminary version in EUROCRYPT 2003.

[CHK09] D. Cash, D. Hofheinz, and E. Kiltz. How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351, July 2009. `http://eprint.iacr.org/`.

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

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

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

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

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

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

[GMR84] S. Goldwasser, S. Micali, and R. 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] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

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

[HHGP$^+$03] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140. 2003.

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

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

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

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

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

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

[LN09] G. Leurent and P. Q. Nguyen. How risky is the random-oracle model? In *CRYPTO*, pages 445–464. 2009.

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

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

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

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

[MW01] D. Micciancio and B. Warinschi. A linear space algorithm for computing the Hermite normal form. In *ISSAC*, pages 231–236. 2001.

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

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

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

[Pei10] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*. 2010. To appear.

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

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

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

[Rab79] M. 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] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005.

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

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

[SSTX09] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635. 2009.

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

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

[Wat09] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636. 2009.

[YFDL04] D. Yao, N. Fazio, Y. Dodis, and A. 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.