How to Delegate a Lattice Basis

David Cash Dennis Hofheinz Eike Kiltz

July 24, 2009

Abstract

We present a technique, which we call *basis delegation*, that allows one to use a short basis of a given lattice to derive a new short basis of a related lattice in a secure way. And since short bases for lattices essentially function like cryptographic trapdoors, basis delegation turns out to be a very powerful primitive. As the main application of our technique, we show how to construct hierarchical identity-based encryption (HIBE) that is secure, without random oracles, under the assumption that certain standard lattice problems are hard in the worst case. This construction and its variants constitute the first HIBE schemes from lattices, as well as the first lattice-based constructions of stateless signatures and identity-based encryption without random oracles.

1 Introduction

Lattice problems are arguably the most compelling candidate for hard problems for cryptography that are not directly related to integer factoring or discrete logarithm problems. Lattice problems provide some significant advantages not found in other types of cryptography. The most prominent (and unique) advantage was first found in a result of Ajtai [Ajt04], which showed that certain lattice problems are hard on average as long as they are hard in the worst case. Lattice problems are also attractive in that, unlike the number-theory problems used in cryptography, no quantum algorithms for lattice problems are known to outperform classical algorithms. Furthermore, schemes based on lattice problems typically scale better asymptotically than number-theory based counterparts.

Recently it was shown by Gentry, Peikert and Vaikuntanathan [GPV08] that lattice problems suffice for constructing a kind of trapdoor primitive called a *preimage-samplable function*. In the same work, this primitive was shown to imply, in the random oracle model, digital signatures and, using specific properties of their construction, identity-based encryption (IBE).

In the present paper we show how to realize a much richer type of trapdoor-like behavior from lattices. We present a technique that we call *basis delegation*, which, at a high level, allows one to use a short basis of a given lattice to derive a short basis of related lattice in secure way. And since short bases for lattices essentially function like cryptographic trapdoors, basis delegation turns out to be a very powerful primitive.

At the heart of our basis delegation technique is an idea that we call *generalized preimage sampling*, which has further applications. This idea essentially shows that, given a trapdoor that allows preimage-sampling in the sense of [GPV08], we can actually use a trapdoor to preimage sample under many *different*, but related, public keys. Interestingly, this property is enabled by the fact that we are sampling preimages, and not computing unique preimages, as is the case with traditional primitives like trapdoor functions. Previously, the idea of preimage sampling was necessary to protect the lattice and somewhat of a hindrance when using the primitive, but here we find that it gives far more flexibility than traditional approaches.

We show how to apply our techniques to give several new constructions of lattice-based primitives. In particular, we get the following results:

- Stateless digital-signatures and IBE schemes that are provably secure, under worst-case lattice hardness assumptions, without random oracles. Specifically, we show how to apply generalized preimage sampling to implement "all but one" style simulations with lattices without random oracles.
- A reasonably efficient construction of hierarchical identity-based encryption (HIBE) that is provably secure, in the random oracle model, under standard worst-case lattice assumptions. The construction uses our basis-delegation technique to implement the key hierarchy.
- A HIBE scheme that is provably secure under the same assumptions, but *without* random oracles. This construction requires further ideas which extend the combinatorial ideas behind the *admissible hash functions* of [BB04b].

Our constructions rely on the same assumptions as recent work (although with different quantitative parameters that depend on the specific properties of instantiations of our schemes). As is always the case with identity based encryption, we get public-key encryption and chosen-ciphertext security from each of our constructions with security and efficiency properties similar to the IBE schemes.

Despite its importance, constructions of secure and efficient IBE have been found only in the last decade [BF03, Coc01, SOK00], and we still have relatively few techniques for constructing them. Prior to the recent lattice-based constructions, all known constructions were based either on pairings or quadratic residuosity. For hierarchical identity-based encryption [GS02, HL02], we have even fewer constructions, and all known HIBE constructions are from problems related pairings, and, in particular, none are known from lattices.

In addition to giving the first signature schemes[1] and identity-based encryption schemes from lattices without random oracles, these results also represent the first construction of *hierarchical* identity-based encryption that does not use pairings (with or without random oracles). Our constructions are also *anonymous* and therefore have applications in searchable encryption [BDOP04, ABC$^+$05].

1.1 Overview of Our Techniques

Here we start with a high-level description of our techniques, beginning with generalized preimage sampling. For now we deal with the techniques abstractly and dispense with some issues related to lattices that complicate the explanation.

Generalized Preimage Sampling. Recall that the work of [GPV08] gave a construction of a keyed function f_{pk}, such that with the corresponding trapdoor sk, one could *sample* preimages from for a given element in the range of the function. Preimage sampling (instead of deterministic

[1]Ajtai's lattice-based one-way function [Ajt04] already implies stateful signatures without random oracles through the generic, tree-based construction by Rompel [Rom90]. See also [LM08] for a more efficient construction. Our proposed signature schemes follow the "hash-and-sign" principle and are therefore stateless. (This is a crucial property in practical applications where it may be required that different systems sign for the same public-key.)

inversion, which may be possible even with a many-to-one function) was necessary because of the security properties of the lattices involved.

In our generalized version of preimage sampling enables the following type of inversion. We deal with the same function (up to parameter resetting) f_{pk}, and start with the same trapdoor sk. Now fix some y in the range of f_{pk}. Instead of enabling sampling[2] from the set $f_{pk}^{-1}(y) = \{x : f_{pk}(x) = y\}$, we show that one can actually sample from $f_{pk'}^{-1}(y)$ *for an entire family of public keys* pk'. In slightly more detail, we show that one can securely pre-image sample whenever pk appears as a "substring" of pk'. Intuitively, we construct a preimage sample by selecting some portions of the preimage element in the "forward" direction, and then we use the trapdoor the compute some in the "backward" direction. We note that this idea *crucially* depends on preimage sampling, as normally we would not be allowed to compute part of the preimage in the forward direction. Using this technique, we are able to carry out security reductions that have the flavor of Noar-Yung encryption, where the trapdoor elements known by the simulator are indistinguishable from non-trapdoor elements.

Basis Delegation. Our basis delegation algorithm is built from the generalized preimage sampler, but it requires one more fundamental observation about an abstract property of the GPV function f_{pk}.

We can understand the situation by an analogy with, say, the RSA trapdoor function with public key N and trapdoor p, q where $N = pq$. There the domain and range of the function are elements of $\mathbb{Z}_N$, and these elements never "mix" with the trapdoor p, q directly, and they seem like different types of objects. But the GPV function seems fundamentally different: its trapdoor elements are actually very much the same type of object as the elements of domain – in particular, they are both "short" vectors in a particular lattice. The difference is only quantitative, in that the trapdoor elements are simply shorter.

This observation allows us to "mix" the trapdoor and domain elements, arriving at a natural and powerful type of primitive. Specifically, we will show that one can generate a new trapdoor by simply sampling preimages several times. The new trapdoor will be quantitatively weaker than the original, but still sufficient for cryptographic purposes.

Now the payoff comes when we combine this idea with generalized preimage sampling. Recall that generalized preimage sampling gives the ability to generate preimage samples for some pk'. But according to the above idea, this actually allows us to generate a trapdoor for pk'! The new trapdoor will again be weaker, but sufficient for our purposes. This is exactly the idea behind basis delegation, as trapdoors for the GPV function are bases. A hierarchy structure also becomes apparent when one observes that the new trapdoor is sufficient for *further* preimage sampling.

1.2 Applications of Our Techniques

In this section we discuss how our techniques can be applied to construct cryptographic primitives. As with the previous discussion, we ignore the technicalities involved with lattices and focus on the "cryptographic" ideas needed.

HIBE with a Random Oracle. Next we present is a HIBE scheme that is secure in the random oracle model under appropriate assumptions. It is implemented using our basis delegation in the following way. The master public/secret key pair is simply a public/secret key pair $(pk_\varepsilon, sk_\varepsilon)$ for

[2] By *sampling*, we mean sampling from a specific distribution defined by the function, but this detail is not important at this point.

the GPV preimage samplable function, which was also the case in the GPV IBE scheme. Given the master secret key, we extract a key (basis) for the identity id_1 by taking $pk_1 = H(id_1)$, and then setting $pk' = pk_\varepsilon \| pk_1$. Since pk_ε is a substring of pk', our basis delegation algorithm can extract a new key for pk'. Now, with a secret key for pk', a user can further delegate by using basis delegation again: for another identity id_2, it computes $pk_2 = H(id_1 \| id_2)$, and then applies basis delegation to get a secret key for the public key $pk_\varepsilon \| pk_1 \| pk_2$. This works because $pk_\varepsilon \| pk_1$ is a substring of the new public key. Finally, we complete the IBE by noting that encryption and decryption algorithms can work as in the GPV IBE, up to resetting parameters.

In the security proof, the simulator is able to derive secret keys for every identity except the challenge because it will always know a trapdoor for *some* substring of the identity, but *not* for the same substring that a real user would know (which will be for a prefix). Therefore the simulator can guess the challenge identity chain as the identities are given to the random oracle and answer key extraction queries for all other identities.

Selective-ID security in the Standard Model. As another application of our basis delegation technique we show how to instantiate the hash function H in the standard model. First we explain a simple IBE scheme that is only selective-ID secure (i.e., in the security experiment the adversary has to decide first on the challenge identity, and then receives the public key). We use a Dolev-Dwork-Naor [DDN91] type idea where the master public-key contains $2k$ many independent public-keys $pk_{i,b}$ ($1 \leq i \leq k$, $0 \leq b \leq 1$) and a user public-key for id is $pk_{1,id_1} \| \dots \| pk_{k,id_k}$, choosen according to the binary representation of the identity. By our generalized preimage sampling technique, a user sceret-key can be computed if (and only if) one knows at least one of the k secret-keys corresponding to pk_{i,id_i}. Since in the security proof the challenge identity is known in advance, we can apply an "all-but-one" simulation strategy where the simulator knows all corresponding secret-keys except the ones for the challenge identity. Our standard-model HIBE is an extension of this idea that uses our basis delegation technique to implement the key hierarchy.

Full security in the Standard Model. The security proof for the latter scheme depends, of course, heavily on the fact that the simulator knows the challenge identity in advance. In the general HIBE security experiment, however, the adversary may choose the challenge identity dynamically and adaptively. To achieve full HIBE security in the standard model, we employ a probabilistic argument, using a variant of *admissible hash functions* (AHFs, [BB04b]). AHFs have been introduced by Boneh and Boyen [BB04b] in a similar setting to achieve full IBE security. As we will explain in Section 5.3, 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. In particular, we use artificial abort techniques (as used by Waters in [Wat05]).

We remark that the resulting HIBE scheme is not very practical, as we incur an efficiency cost similar to that of [BB04b]. It does however serve as a proof-of-concept for a lattice based HIBE with a polynomial security reduction in the standard model.

Further Applications. A HIBE scheme is an extremely powerful primitive, so it directly gives rise to a number of further cryptographic applications. Specifically, we can achieve CCA secure encryption (both for PKE and HIBE schemes) by sacrificing one HIBE level and using techniques from [BCHK06]. Since the transformation of [BCHK06] requires only selective-identity security, this in particular yields a very efficient lattice-based CCA secure PKE scheme. Furthermore, our fully CPA secure HIBE scheme implies a stateless lattice-based signature scheme without random oracles.

1.3 Related Work

Since the work of Ajtai [Ajt04], lattice applications in cryptography have been studied intensely. See the survey by Micciancio and Regev [MR09] for an overview.

In concurrent and independent work, Peikert [Pei09a] proposes the notion of a "bonsai tree" on lattices which, at a technical level, is equivalent to our basis delegation technique. His work also shows how to use bonsai trees to get digital signatures and HIBE without random oracles. Furthermore, Peikert obtains fully secure (as opposed to selective-ID secure) HIBE through a similar implementation of hash functions, although without resolving the complications implicit in prior work that we deal with in Section 5. We remark that he obtains fully secure signatures through chameleon hashes, which results in more efficient schemes than one gets through our fully secure IBE (in addition to avoiding prior complications).

In another concurrent and independent work, Agrawal and Boyen [AB09] also show how to obtain (non-hierarchical) IBE based on LWE without random oracles. Their construction is very similar to a one-level version of our HIBE.

2 Preliminaries

2.1 Notation

We write $[d] = \{1, \dots, d\}$. A probabilistic polynomial-time (PPT) algorithm is a randomized algorithm which runs in strict polynomial time. If A is a probabilistic algorithm, we write $y \leftarrow A(x)$ to denote that the random variable y is defined as the output of A when run on input x and with fresh random coins. On the other hand, if S is a distribution, then $s \leftarrow S$ defines s as being sampled from S. Vectors are denoted by bold lower-case letters (e.g., $\mathbf{a} = (a_i)$) and matrices by bold upper-case letters (e.g., $\mathbf{A} = (a_{i,j})$). For an $n \times m$ matrix $\mathbf{A}$ we write $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m]$, where $\mathbf{a}_i$ denotes the ith column vector of $\mathbf{A}$. We write $\|\mathbf{a}\|$ for the Euclidean norm of $\mathbf{a}$, and $\|\mathbf{A}\| = \max_{i \in [m]} \|\mathbf{a}_i\|$, where $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_m]$.

2.2 Cryptographic Notions

Hierarchical Identity-based Encryption. A (hierarchical) identity of length ℓ is a vector $\mathbf{id} = (id_1, \dots, id_\ell)$, where id_i is from some identity-space (usually $\{0,1\}^*$.). We denote by $\mathbf{id}|i = (id_1, \dots, id_i)$ the i-th prefix of $\mathbf{id}$. An hierarchical identity-based encryption scheme HIBE for hierarchies of depth d consists of the following algorithms.

- A setup algorithm $\mathsf{HIBESetup}$ that, given the security parameter k, outputs a master public key mpk and a master secret key msk.
- A secret key extraction algorithm $\mathsf{HIBEExt}$ that, given msk and an identity $\mathbf{id}$ of length at most d outputs a user secret key $usk_{\mathbf{id}}$ for that identity.
- A secret key delegation algorithm $\mathsf{HIBEDel}$ that, given an identity $\mathbf{id}$ of length at most d and a user secret key $usk_{\mathbf{id}|\ell-1}$ for its parent $\mathbf{id}|\ell - 1$, outputs a user secret key $usk_{\mathbf{id}}$ for $\mathbf{id}$.

- An encryption algorithm $\mathsf{HIBEEnc}$ that, given a message m, an identity $\mathbf{id}$ and the master public key mpk ourputs a ciphertext C.

- A decryption algorithm $\mathsf{HIBEDec}$ that, given a user secret key $usk_{\mathbf{id}}$ and a ciphertext c, returns a message m.

We require the usual completeness properties and that a user secret key computed using the delegation algorithm has the same distribution as one generated by the key extraction algorithm. Note that for $d = 1$ we have an identity-based encryption (IBE) scheme and for $d = 0$ we have a standard public-key encryption (PKE) scheme.

For security we use the notion of chosen-plaintext (IND-CPA) security [GS02] which is defined by the following game. An adversary A is given randomly generated mpk and access to an oracle computing $\mathsf{HIBEExt}$. The adversary produces a challenge identity vector $\mathbf{id}^*$ and two messages m_0, m_1, is given an encryption of m_b under identity vector $\mathbf{id}^*$ for $b \leftarrow \{0,1\}$ chosen uniformly at random, and tries to guess b without querying its oracle on any parent of identity vector $\mathbf{id}^*$. We define A's advantage in the above game as $\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{HIBE},\mathsf{A}}(k)$. We say that HIBE is IND-CPA secure if for all PPT A, $\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{HIBE},\mathsf{A}}(k) - 1/2$ is negligible. For the notion of IND-CCA security we give the adversary additionally a decryption oracle that decrypts all ciphertexts, except the challenge one. For the notion of selective identity (sID-IND-CPA) security [BB04a] we require the adversary to commit to the challenge identity vector $\mathbf{id}^*$ before obtaining mpk.

We also consider the notion of anonymity which says that a ciphertext does not reveal the identity of the recipient [BDOP04, ABC+05]. (A formal definition can be looked up in [BDOP04, ABC+05].)

Collision-resistant hash functions. Let $\mathcal{H} = \{\mathcal{H}_k\}$ be a collection of distributions of functions $H : \mathcal{C}_k \to \mathcal{D}_k = \{0,1\}^\lambda$. For an algorithm B, define

$$\mathbf{Adv}^{\mathsf{crhf}}_{\mathcal{H},\mathsf{B}}(k) := \Pr\big[x' \neq x \wedge H(x') = H(x) \mid H \leftarrow \mathcal{H}_k \; ; \; (x, x') \leftarrow \mathsf{B}(H)\big].$$

We say that $\mathcal{H}$ is *collision resistant* if for all PPT B, the function $\mathbf{Adv}^{\mathsf{crhf}}_{\mathcal{H},\mathsf{B}}(k)$ is negligible in k.

2.3 Lattices

For a set of linear independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \subset \mathbb{R}^{m\times m}$, the *lattice generated by* $\mathbf{B}$ is the set

$$\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{Bc} = \sum_{i=1}^{m} c_i \cdot \mathbf{b}_i : \mathbf{c} \in \mathbb{Z}^m\}.$$

In this case $\mathbf{B}$ is referred to as a *basis* for $\mathcal{L}(\mathbf{B})$. (The notion of $\mathcal{L}(\mathbf{B})$ also makes sense if the vectors in $\mathbf{B}$ are not linear independent.)

We will restrict our attention to a special class of q-ary lattices defined by Ajtai [Ajt99]. Lattices in this family are more easily described by a matrix that functions like a parity check matrix from coding theory. More precisely, for some integers q, m, n and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we define the following m-dimensional q-ary lattice

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{0} \bmod q\},$$

i.e., the lattice that contains all vectors that are orthogonal modulo q to the rows of $\mathbf{A}$.

We will need the following lemma, which shows that if we start with a set of vectors that span a particular lattice, then we can compute a basis of that lattice that does not increase the length of the spanning set's *Gram-Schmidt*[3] vectors $\widetilde{\mathbf{b}}_i$.

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

2.4 Discrete Gaussian

The Gaussian function on $\mathbb{R}^m$ with parameter $r > 0$ is:

$$\rho_r(\mathbf{x}) = \exp\left(\pi \cdot \|\mathbf{x}\|^2 / r^2\right).$$

For a countable set A we define $\rho_{s,\mathbf{c}}(A)$ in the obvious way as $\sum_{x \in A} \rho_r(x)$. Fix a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The discrete Gaussian distribution on lattice $\Lambda_q^\perp(\mathbf{A})$ is defined for $\mathbf{x} \in \Lambda_q^\perp(\mathbf{A})$ by

$$D_{\Lambda_q^\perp(\mathbf{A}),r}(\mathbf{x}) = \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{r,\mathbf{c}}(\Lambda_q^\perp(\mathbf{A}))}.$$

For a fixed vector $\mathbf{y} \in \mathbb{Z}_q^n$ in the span of $\mathbf{A}$ it will also be useful to define the coset of $\Lambda_q^\perp(\mathbf{A})$ as

$$\Lambda_q^{\mathbf{y}}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{A}\mathbf{e} = \mathbf{y} \bmod q\} = \mathbf{t} + \Lambda_q^\perp(\mathbf{A}) \bmod q,$$

where $\mathbf{t}$ is an arbitrary solution (over $\mathbb{Z}$) of the equation $\mathbf{A}\mathbf{t} = \mathbf{y} \bmod q$. The Gaussian on $\Lambda_q^{\mathbf{y}}(\mathbf{A})$, which is the conditional distribution $D_{\mathbb{Z}^m,r}$ conditioned on $\mathbf{A}\mathbf{e} = \mathbf{y} \bmod q$, is given by

$$D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}(\mathbf{x}) = \frac{\rho_r(\mathbf{x})}{\rho_r(\mathbf{t} + \Lambda_q^\perp)}.$$

We collect some central facts regarding the Gaussian on (cosets of) lattices that apply when the Gaussian parameter r exceeds the "smoothing parameter" of the underlying lattice. The latter one will not be further defined here, as we only implicitly use the fact that for almost all matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ the smoothing parameter is sufficiently small [GPV08, Lemma 5.3].

Lemma 2.2. *Let* n, q, m *be integers and* $m \geq 2n \lg q$*. For for all but a* $2q^{-n}$ *fraction of all* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and for* $r \geq \omega(\sqrt{\log m})$*, the following hold:*

1. *[Reg05, Cor. 3.16] The probability that a set of* m^2 *vectors chosen independently from* $D_{\Lambda_q^\perp(\mathbf{A}),r}$ *contains no* m *linearly independent vectors is exponentially small.*
2. *[GPV08, Cor. 5.4] The distribution of* $\mathbf{y} = \mathbf{A}\mathbf{e} \mod q$ *is statistically close to uniform on* $\mathbb{Z}_q^n$ *when* $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,r}$*.*
3. *[MR07] Except with exponentially small probability we have that* $\|\mathbf{e}\| \leq r\sqrt{m}$ *when* $\mathbf{e} \leftarrow D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$*.*
4. *[MR07] For any* $\mathbf{t} \in \mathbb{Z}^m$*,* $\rho_r(\mathbf{t} + \Lambda_q^\perp(\mathbf{A})) \in [\frac{1-\varepsilon}{1+\varepsilon}, 1] \cdot \rho_r(\Lambda_q^\perp(\mathbf{A}))$*, for some negligible function* ε*.*

[3] The Gram-Schmidt vectors of a basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ are defined to be $\widetilde{\mathbf{b}_1} = \mathbf{b}_1$, and then for $i = 2, \ldots, n$ $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to span$(\mathbf{b}_1, \ldots, \mathbf{b}_n)$.

2.5 Hard Average Case Problems on Lattices

To define the hard problems we will need another probability distribution. The *normal (Gaussian) distribution* on $\mathbb{R}$ with mean 0 and variance σ^2 is the distribution with density function $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-x^2/2\sigma^2\right)$. For $\alpha \in \mathbb{R}^+$, let Ψ_α be the distribution, modulo 1, of a normal random variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$. We define the *discretized normal distribution on* $\mathbb{Z}_q$, denoted $\bar{\Psi}_\alpha$ to be the distribution of $\lfloor q \cdot X \rceil \mod q$, where X is a random variable with distribution Ψ_α and $\lfloor x \rceil$ is the closest integer to $x \in \mathbb{R}$.

Let $q \geq 2$ be an integer and let χ be probability distribution on $\mathbb{Z}_q$. For $\mathbf{s} \in \mathbb{Z}_q^n$, define $A_{\mathbf{s},\chi}$ to the be the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ induced by choosing $\mathbf{a}$ uniformly at random from $\mathbb{Z}_q^n$, $x \leftarrow \chi$, and outputting $(\mathbf{a}, \mathbf{a}^T\mathbf{s} + x)$.

Learning with Errors (Decision Version). In $\mathsf{LWE}_{q,\chi}$, the adversary must distinguish between the following oracles. The first oracle selects $\mathbf{s}$ uniformly at random from $\mathbb{Z}_q^n$, which remains fixed. To respond to a query, the oracle draws a sample from $A_{\mathbf{s},\chi}$ and returns it to the adversary. The other oracle draws uniform samples over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ and returns them to the adversary.

Short Integer Solutions. An instance of the $\mathsf{SIS}_{q,m,\gamma}$ problem is a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. The goal is to find a nonzero integer vector $\mathbf{e} \in \Lambda_q^\perp(\mathbf{A})$ such that $\|\mathbf{e}\| \leq \gamma$.

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

The LWE and SIS problems are especially interesting to us because there is great theoretical evidence for their hardness. In particular, solving LWE and SIS, which are randomized problems, is known to be as hard as solving certain lattice problems in the *worst case*. For instance, if we set $q \geq \gamma \cdot \omega(\sqrt{n \log n})$, then solving $\mathsf{SIS}_{q,m,\gamma}$ is as hard as approximately solving, say, the shortest vector problem within a factor $\tilde{O}(\gamma \cdot \sqrt{n})$ [MR07, GPV08]. If we take $q \geq (1/\alpha) \cdot \omega(\sqrt{n \log n})$, then solving $\mathsf{LWE}_{q,\chi}$ with $\chi = \bar{\Psi}_\alpha$ is as hard as approximating the same problems within factor $\tilde{O}(n/\alpha)$ on a quantum machine. Since we do not use these results explicitly, we only refer to [Reg05, Pei09b] for specific calculations with regard to the hardness of LWE and SIS.

2.6 Public-key encryption from learning with errors

We recall the public-key encryption scheme from [GPV08], which is, in some sense, a dual of the scheme of Regev [Reg05].

The scheme is parameterized by some $r \geq \omega(\sqrt{\log m})$, which specifies the discrete Gaussian distribution $D_{\mathbb{Z}^m,r}$ from which the secret keys are chosen and α which specifies the amount of noise in the ciphertexts. Let $\mathsf{LWE\text{-}PKE} := (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be the following PKE scheme:

$\mathsf{Setup}(1^k)$ uniformly picks $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and an error vector $\mathbf{e} \leftarrow D_{\mathbb{Z}^m,r}$, sets $\mathbf{y} := \mathbf{A}\mathbf{e}$ and outputs

$$pk := (\mathbf{A}, \mathbf{y}), \qquad\qquad sk := \mathbf{e}.$$

$\mathsf{Enc}(pk, b \in \{0,1\})$ parses $pk = (\mathbf{A}, \mathbf{y})$, and uniformly picks $\mathbf{s} \in \mathbb{Z}_q^n$ and error vectors $\mathbf{x} \leftarrow \chi^m$, $x \leftarrow \chi$. Enc defines $\mathbf{p} := \mathbf{A}^T\mathbf{s} + \mathbf{x}$ and $c := \mathbf{y}^T\mathbf{s} + x + b\lfloor q/2 \rfloor \in \mathbb{Z}_q$. The ciphertext is

$$C := (\mathbf{p}, c) \in \mathbb{Z}_q^m \times \mathbb{Z}_q.$$

$\mathsf{Dec}(sk, C)$ parses $sk = \mathbf{e}$ and $C = (\mathbf{p}, c)$, and computes $b' = c - \mathbf{ep}$. If b' is closer to 0 than $\lfloor q/2 \rfloor$ modulo q, output 0. Otherwise output 1.

As pointed out in [GPV08], efficiency of the basic scheme can be improved to encrypt messages of length $\ell = \text{poly}(n)$ bits, with ciphertexts of $\tilde{O}(m + \ell)$ bits and public-keys of size $\tilde{O}(\ell n)$ bits. This can be done by including ℓ syndromes $\mathbf{y}_1, \ldots, \mathbf{y}_\ell$ in the public-key and to encrypt to each of them using the same $\mathbf{s}$ and $\mathbf{p} = \mathbf{A}^T\mathbf{s} + \mathbf{x}$. Furthermore, it is also possible to encrypt $O(\log n)$ bits per syndrome, which yields an expansions factor of $O(1)$.

Theorem 2.3 ([GPV08]). *Let $q \geq 5r(m+1)$, $\alpha \leq 1/(r\sqrt{m+1} \cdot \omega(\sqrt{\log n}))$, $\chi = \bar{\Psi}_\alpha$, and $m \geq 2n \lg q$. Then the above public-key encryption scheme is CPA-secure and anonymous, assuming that $\mathsf{LWE}_{q,\chi}$ is hard. Moreover, decryption succeeds with overwhelming probability.*

3 Basis Delegation

In this section we describe and analyze our basis delegation construction, including its main component, generalized preimage sampling. We first recall some existing cryptographic tools that will be used below.

3.1 Trapdoors for Lattices and Preimage Sampling

Ajtai [Ajt99] showed how to sample a nearly uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a relatively short basis $\mathbf{B}$ of $\Lambda^\perp(\mathbf{A})$. We give the following improved version of Ajtai's basis sampling algorithm from [AP09].

Lemma 3.1 ([AP09]). *Let n, q, m be positive integers with $q \geq 2$ and $m \geq 5n \lg q$. There exist a PPT algorithm $\mathsf{TrapGen}$ that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{S} \in \mathbb{Z}_q^{m \times m})$ such that $\mathbf{A}$ is statistically close to uniform on $\mathbb{Z}_q^{n \times m}$ and $\mathbf{B}$ is a basis of $\Lambda^\perp(\mathbf{A})$ such that $\|\mathbf{B}\| \leq m \cdot \omega(\sqrt{\log m})$ with all but $n^{\omega(1)}$ probability.*

It was also shown in [GPV08] that if $\mathsf{ISIS}_{q,m,2r\sqrt{m}}$ is hard then $\mathbf{A}$ defines one-way function $f_\mathbf{A} : S_{m,r} \to \mathbb{Z}_q^n$ with $f_\mathbf{A}(\mathbf{e}) = \mathbf{Ae} \bmod q$, where $S_{m,r} = \{\mathbf{e} \in \mathbb{Z}^m \ : \ \|\mathbf{e}\| \leq r\sqrt{m}\}$. Note that by Lemma 2.2 (2), $f_\mathbf{A}$ is surjective for almost all $\mathbf{A}$. Furthermore, a short basis for $\Lambda^\perp(\mathbf{A})$ can be used as a trapdoor to sample from $f_\mathbf{A}^{-1}(\mathbf{y})$, for any $\mathbf{y} \in \mathbb{Z}_q^n$. Klein's algorithm [Kle00] can be used to efficiently sample short vectors from $f_\mathbf{A}^{-1}(\mathbf{y})$ (distributed according to $D_{\Lambda_q^\mathbf{y}(\mathbf{A}),r}$), without giving away any information about the short basis $\mathbf{B}$ [GPV08].

Lemma 3.2 ([GPV08]). *Let n, q, m be positive integers with $q \geq 2$ and $m \geq 2n \lg q$. There exist a PPT algorithm $\mathsf{SamplePre}$ such that on input of $\mathbf{A} \in \mathbb{Z}_q^{n \times km}$, a basis $\mathbf{B}_S$ for $\Lambda_q^\perp(\mathbf{A})$, a vector $\mathbf{y} \in \mathbb{Z}_q^n$, and an integer $r \geq \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log m})$, the distribution of the output of $\mathbf{e} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{B}, \mathbf{y}, r)$ is within negligible statistical distance of $D_{\Lambda_q^\mathbf{y}(\mathbf{A}),r}$.*

Note that $\mathsf{SamplePre}$ in particular allows one to sample efficiently from the distribution $D_{\mathbb{Z}^m,r}$ for any $r \geq \omega(\sqrt{\log m})$, by taking $\mathbf{B}$ to be the standard basis.

3.2 Our Basis Delegation Algorithm

We now describe our main technical result which is an abstract method that uses a good basis of some lattice Λ' to generate another good basis for a higher-dimensional lattice Λ that contains a sublattice isomorphic to Λ'.

Let $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$ and write $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_k]$, where each $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$. For $S \subseteq [k]$, $S = \{i_1, \ldots, i_j\}$, we will write $\mathbf{A}_S$ to denote $[\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_j}]$, i.e., the components of $\mathbf{A}$ selected according to S, when $\mathbf{A}$ is viewed as a vector over $\mathbb{Z}_q^{n\times m}$. We describe a procedure for generating a short basis of $\Lambda_q^\perp(\mathbf{A})$ using a short basis of $\Lambda_q^\perp(\mathbf{A}_S)$, for some $S \subseteq [k]$. That is, given a trapdoor for $f_{\mathbf{A}_S}$, we can generate a trapdoor for $f_{\mathbf{A}}$.

Theorem 3.3. *Let n, q, m, k be positive integers with $q \geq 2$ and $m \geq 2n \lg q$. There exists a PPT algorithm* $\mathsf{SampleBasis}$*, that on input of $\mathbf{A} \in \mathbb{Z}^{n\times km}$, a set $S \subseteq [k]$, a basis $\mathbf{B}_S$ for $\Lambda_q^\perp(\mathbf{A}_S)$, and an integer $L \geq \|\widetilde{\mathbf{B}}_S\| \cdot \sqrt{km} \cdot \omega(\sqrt{\log km})$ outputs $\mathbf{B} \leftarrow \mathsf{SampleBasis}(\mathbf{A}, \mathbf{B}_S, S, L)$ such that, for an overwhelming fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, $\mathbf{B}$ is a basis of $\Lambda_q^\perp(\mathbf{A})$ with $\|\widetilde{\mathbf{B}}\| \leq L$ (with overwhelming probability). Furthermore, up to a statistical distance the distribution of the basis $\mathbf{B}$ only depends on $\mathbf{A}$ and L (and does not depend on $\mathbf{B}_S$ and S).*

For the proof of Theorem 3.3 we need the following result that generalizes Lemma 3.2 in that it allows pre-image sampling of the function $f_{\mathbf{A}}$ given a short basis for $\Lambda_q^\perp(\mathbf{A}_S)$.

Theorem 3.4. *Let n, q, m, k be as in Theorem 3.3. There exists a PPT algorithm* $\mathsf{GenSamplePre}$*, that on input of $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, a set $S \subseteq [k]$, a basis $\mathbf{B}_S$ for $\Lambda_q^\perp(\mathbf{A}_S)$, a vector $\mathbf{y} \in \mathbb{Z}_q^n$, and an integer $r \geq \|\widetilde{\mathbf{B}}_S\| \cdot \omega(\sqrt{\log km})$ outputs $\mathbf{e} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}, \mathbf{B}_S, S, \mathbf{y}, r)$ which, for an overwhelming fraction of $\mathbf{A} \in \mathbb{Z}_q^{n\times km}$, is within negligible statistical distance of the distribution $D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$.*

Proof. Assume without loss of generality that $S = [s]$ for some $s \in [k]$. Let $S^c = [k] \setminus S$. The sampling algorithm $\mathsf{GenSamplePre}(\mathbf{A}, \mathbf{B}_S, S, \mathbf{y}, r)$ proceeds as follows:

1. Sample $\mathbf{e}_{S^c} \in \mathbb{Z}^{(k-s)\cdot m}$ from the distribution $D_{\mathbb{Z}^{(k-s)m},r}$, and let $\mathbf{z} = \mathbf{y} - \mathbf{A}_{S^c}\mathbf{e}_{S^c}$. Parse $\mathbf{e}_{S^c}$ as $[\mathbf{e}_{s+1}, \ldots, \mathbf{e}_k]$. This defines $\mathbf{e}_i$ for $i \in S^c$.

2. Run $\mathbf{e}_S \leftarrow \mathsf{SamplePre}(\mathbf{A}_S, \mathbf{B}_S, \mathbf{z})$ from Lemma 3.2 to sample a vector $\mathbf{e}_S \in \mathbb{Z}^{sm}$ from the distribution $D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$. Parse $\mathbf{e}_S$ as $[\mathbf{e}_1, \ldots, \mathbf{e}_s] \in (\mathbb{Z}^m)^s$. This defines $\mathbf{e}_i$ for each $i \in S$

3. Output $\mathbf{e} \in \mathbb{Z}^{km}$, as $\mathbf{e} = [\mathbf{e}_1, \ldots, \mathbf{e}_k]$.

First note that by construction the vectors $\mathbf{e}$ output by this algorithm are contained in $\Lambda_q^{\mathbf{y}}(\mathbf{A})$ so it remains to analyze their distribution.

For the analysis we can assume that the distribution of the vectors $\mathbf{e}_S$ and $\mathbf{e}_{S^c}$ sampled in the first two steps is perfect.[4] For any fixed vector defined by $\mathbf{e} = (\mathbf{e}_S, \mathbf{e}_{S^c}) \in \Lambda_q^{\mathbf{y}}(\mathbf{A})$, let $p(\mathbf{e})$ denote the probability that $\mathsf{GenSamplePre}(\mathbf{A}, \mathbf{B}_S, S, \mathbf{y}, r)$ outputs that vector. We have

$$p(\mathbf{e}) = \Pr[\mathbf{e}_{S^c}] \cdot \Pr[\mathbf{e}_S \mid \mathbf{e}_{S^c}] = \frac{\rho_r(\mathbf{e}_{S^c})}{\rho_r(\mathbb{Z}^{(k-s)m})} \cdot \frac{\rho_r(\mathbf{e}_S)}{\rho_r(\{\mathbf{e}_S : \mathbf{A}_S\mathbf{e}_S = \mathbf{y} - \mathbf{A}_{S^c}\mathbf{e}_{S^c} \bmod q\})} \tag{3.1}$$

[4]The distribution of $\mathbf{e}$ can be seen as a function of the two sampled distributions. Hence, the statistical imperfections of the sampling algorithms affect the distribution of $\mathbf{e}$ only negligibly.

For a fixed $\mathbf{e}_{S^c}$, let $\mathbf{t}(\mathbf{e}_{S^c})$ be an arbitrary solution (over $\mathbb{Z}$) to the equation $\mathbf{A}_S\mathbf{t}(\mathbf{e}_{S^c}) = \mathbf{y} - \mathbf{A}_{S^c}\mathbf{e}_{S^c} \bmod q$. Then we have

$$\{\mathbf{e}_S : \mathbf{A}_S\mathbf{e}_S = \mathbf{y} - \mathbf{A}_{S^c}\mathbf{e}_{S^c} \bmod q\} = \mathbf{t}(\mathbf{e}_{S^c}) + \Lambda_q^{\perp}(\mathbf{A}_S)$$

and by Lemma 2.2 (4), we have

$$\rho_r(\mathbf{t}(\mathbf{e}_{S^c}) + \Lambda_q^{\perp}(\mathbf{A}_S)) \in [\frac{1-\varepsilon}{1+\varepsilon}, 1] \cdot \rho_r(\Lambda_q^{\perp}(\mathbf{A}_S)) \ ,$$

for some negligible function ε. Combining this with (3.1) we obtain

$$p(\mathbf{e}) \in \frac{\rho_r(\mathbf{e}_{S^c})}{\rho_r(\mathbb{Z}^{(k-s)m})} \cdot \frac{\rho_r(\mathbf{e}_S)}{[\frac{1-\varepsilon}{1+\varepsilon}, 1] \cdot \rho_r(\Lambda_q^{\perp}(\mathbf{A}_S))} \ . \tag{3.2}$$

Next, we claim that

$$\rho_r(\Lambda_q^{\mathbf{y}}(\mathbf{A})) \in [\frac{1-\varepsilon'}{1+\varepsilon'}, 1] \cdot \rho_r(\mathbb{Z}^{(k-s)m})\rho_r(\Lambda_q^{\perp}(\mathbf{A}_S)) \tag{3.3}$$

for some negligible function ε'. Combining (3.2) with (3.3) we obtain

$$p(\mathbf{e}) \in [\frac{1-\varepsilon'}{1+\varepsilon'}, \frac{1+\varepsilon}{1-\varepsilon}] \cdot \frac{\rho_r(\mathbf{e}_S)\rho_r(\mathbf{e}_S)}{\rho_r(\Lambda_q^{\mathbf{y}}(\mathbf{A}))} \tag{3.4}$$

which is within negligible statistical distance of $D_{\Lambda_q^{\mathbf{y}}(\mathbf{A}),r}$.

It remains to prove (3.3).

$$\begin{aligned}
\rho_r(\Lambda_q^{\mathbf{y}}(\mathbf{A})) = \sum_{\mathbf{e}\in\Lambda_q^{\mathbf{y}}(\mathbf{A})} \rho_r(\mathbf{e}) &= \sum_{\substack{(\mathbf{e}_S,\mathbf{e}_{S^c})\in\mathbb{Z}^{sm}\times\mathbb{Z}^{(k-s)m}:\\ \mathbf{A}_S\mathbf{e}_S=\mathbf{y}-\mathbf{A}_{S^c}\mathbf{e}_{S^c} \bmod q}} \rho_r(\mathbf{e}_S)\rho_r(\mathbf{e}_{S^c}) \\
&= \sum_{\mathbf{e}_{S^c}\in\mathbb{Z}^{(k-s)m}} \rho_r(\mathbf{e}_{S^c}) \sum_{\substack{\mathbf{e}_S\in\mathbb{Z}^{sm}:\\ \mathbf{A}_S\mathbf{e}_S=\mathbf{y}-\mathbf{A}_{S^c}\mathbf{e}_{S^c} \bmod q}} \rho_r(\mathbf{e}_S) \\
&= \sum_{\mathbf{e}_{S^c}\in\mathbb{Z}^{(k-s)m}} \rho_r(\mathbf{e}_{S^c})\rho_r(\mathbf{t}(\mathbf{e}_{S^c}) + \Lambda_q^{\perp}(\mathbf{A}_S)) \\
&\in \sum_{\mathbf{e}_{S^c}\in\mathbb{Z}^{(k-s)m}} \rho_r(\mathbf{e}_{S^c}) \cdot \left[\frac{1-\varepsilon'(\mathbf{e}_{S^c})}{1+\varepsilon'(\mathbf{e}_{S^c})}, 1\right] \rho_r(\Lambda_q^{\perp}(\mathbf{A}_S)) \qquad (3.5) \\
&\subseteq \left[\frac{1-\varepsilon'}{1+\varepsilon'}, 1\right] \cdot \rho_r(\mathbb{Z}^{(k-s)m})\rho_r(\Lambda_q^{\perp}(\mathbf{A}_S))
\end{aligned}$$

where (3.5) used again Lemma 2.2 (4) to obtain negligible functions $\varepsilon'(\mathbf{e}_{S_c})$, for each $\mathbf{e}_{S_c}$. ☐

We can now complete the proof of Theorem 3.3.

Proof. The algorithm $\mathsf{SampleBasis}(\mathbf{A}, \mathbf{B}_S, S, L)$ works as follows. It draws $O((km)^2)$ samples by running $\mathsf{GenSamplePre}(\mathbf{A}, \mathbf{B}_S, S, \mathbf{y} = \mathbf{0}, r := L/\sqrt{km})$ that many times. By Lemma 2.2 (1), we have, with overwhelming probability, that the samples contain km linearly-independent vectors. By Lemma 2.2 (3), they have length at most $r \cdot \sqrt{km} = L$. The algorithm then applies the deterministic procedure from Lemma 2.1 to process the samples into a basis for $\Lambda_q^{\perp}(\mathbf{A})$ without increasing the length of their Gram-Schmidt vectors. ☐

4 HIBE in the random oracle model

As outlined in the introduction, we now use our basis delegation technique from Section 3 in a straightforward way to a HIBE system in the random oracle model. The master public key is the same as the one from LWE-PKE (from Section 2.6), namely a matrix $\mathbf{A}$ and a syndrome $\mathbf{y}$, and the master secret key is a short basis $\mathbf{B}_\varepsilon$ for lattice $\Lambda_q^\perp(\mathbf{A})$. For a hierarchical identity $\mathbf{id} = (id_1, \ldots, id_\ell)$, the user secret key consists of a short basis $\mathbf{B}_{\mathbf{id}}$ for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{id}})$, where $\mathbf{A}_{\mathbf{id}} = [\mathbf{A}, \mathbf{A}_1, \ldots, \mathbf{A}_\ell]$ and $\mathbf{A}_i = H(id_1, \ldots, id_i)$. Delegation of the user secret keys can be done using our basis delegation technique from Theorem 3.3 and the length of the basis grows by a factor of roughly $\sqrt{\ell m}$ upon each delegation. Encryption for $\mathbf{id}$ is done using LWE-PKE with the public key $(\mathbf{A}_{\mathbf{id}}, \mathbf{y})$ and decryption is done using a short vector $\mathbf{e}_{\mathbf{id}} \in \Lambda_q^{\mathbf{y}}(\mathbf{A}_{\mathbf{id}})$ (which can be computed from the short basis $\mathbf{B}_{\mathbf{id}}$ for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{id}})$).

For our actual HIBE scheme we will also make the syndrome $\mathbf{y}$ dependent on the identity $\mathbf{id}$. This way we save one delegation level and the resulting scheme is more efficient.

4.1 The HIBE Scheme RO-HIBE

Below let $H : \{0,1\}^* \to \mathbb{Z}_q^{n \times m}$ and $G : \{0,1\}^* \to \mathbb{Z}_q^n$ be hash functions. Let $\ell \in [d]$, where d is the maximal depth of the HIBE, which we take to be a constant. The scheme is parameterized by d and functions $L(\ell)$, $r(\ell)$, and $\alpha(\ell)$ ($1 \le \ell \le d$), which we fix as follows.

$$L \ge m \cdot \omega(\sqrt{\log n}) \qquad L(\ell) \ge L \cdot m^{\ell/2} \cdot \omega(\lg^{\ell/2} m)$$
$$r(\ell) \ge L(\ell - 1) \cdot \omega(\lg^{1/2} m) \qquad \alpha(\ell) \le 1/(r(\ell) \cdot \sqrt{\ell m + 1} \cdot \omega(\lg^{1/2} n))$$

Intuitively, for an identity $\mathbf{id} = (id_1, \ldots, id_\ell)$ of level ℓ, these parameters serve the following purposes. $L(\ell)$ is the size of the user's secret basis, $r(\ell)$ is the Gaussian parameter for generating the user's key, and $\alpha(\ell)$ is the Gaussian parameter that the encryptor uses when adding noise to the ciphertext.

We now describe RO-HIBE = (HIBESetup, HIBEExt, HIBEDel, HIBEEnc, HIBEDec).

HIBESetup. This algorithm runs the trapdoor algorithm TrapGen from Lemma 3.1 to generate $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with corresponding trapdoor $\mathbf{B}_\varepsilon \in \mathbb{Z}^{m \times m}$ (where $\|\mathbf{B}_\varepsilon\| \le L$) and returns

$$mpk = \mathbf{A}, \quad msk = (mpk, \mathbf{B}_\varepsilon),$$

To simplify the presentation of the scheme, let us first fix some notation for the algorithms below. For an arbitrary identity $\mathbf{id} = (id_1, \ldots, id_\ell)$ we define the associated parity check matrix $\mathbf{A}_{\mathbf{id}}$ and syndrome $\mathbf{y}_{\mathbf{id}}$ as

$$\mathbf{A}_{\mathbf{id}} = [\mathbf{A}, \mathbf{A}_1, \cdots, \mathbf{A}_\ell] \in \mathbb{Z}_q^{n \times (\ell+1)m}, \quad \mathbf{y}_{\mathbf{id}} = G(\mathbf{id}) \in \mathbb{Z}_q^n, \tag{4.1}$$

where $\mathbf{A}_i = H(\mathbf{id}|i) \in \mathbb{Z}_q^{n \times m}$. The user secret keys for an identity $\mathbf{id}$ of length ℓ will consist of a basis part $\mathbf{B}_{\mathbf{id}} \in \mathbb{Z}^{(\ell+1)m \times (\ell+1)m}$ for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{id}})$ and a syndrome part $\mathbf{e}_{\mathbf{id}} \in \mathbb{Z}_q^{\ell m}$ satisfying $\mathbf{A}_{\mathbf{id}|\ell-1}\mathbf{e}_{\mathbf{id}} = \mathbf{y}_{\mathbf{id}}$, where

$$\|\widetilde{\mathbf{B}}_{\mathbf{id}}\| \le L(\ell), \qquad \|\mathbf{e}_{\mathbf{id}}\| \le r(\ell)\ . \tag{4.2}$$

Note that by construction $\mathbf{B}_\varepsilon$ is a short basis for $\mathbf{A}_\epsilon = \mathbf{A}$ satisfying (4.2).

$\mathsf{HIBEExt}(msk, \mathbf{id})$. This algorithm computes a user secret key $usk_{\mathbf{id}} = (\mathbf{B_{id}}, \mathbf{e_{id}})$ for an identity $\mathbf{id} = (id_1, \ldots, id_\ell)$. Here, $\mathbf{B_{id}} \leftarrow \mathsf{SampleBasis}(\mathbf{A_{id}}, \mathbf{B}_\varepsilon, S = \{1\}, L(\ell))$ is a basis for $\Lambda_q^\perp(\mathbf{A_{id}})$ and $\mathbf{e_{id}} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{\mathbf{id}|\ell-1}, \mathbf{B}_\varepsilon, S = \{1\}, \mathbf{y_{id}}, r(\ell))$ is distributed according to $D_{\Lambda^{\mathbf{y_{id}}}(\mathbf{A}_{\mathbf{id}|\ell-1}), r(\ell)}$. Note that by Theorems 3.3 and 3.4, $usk_{\mathbf{id}} = (\mathbf{B_{id}}, \mathbf{e_{id}})$ satisfies (4.2).

$\mathsf{HIBEDel}(mpk, usk_{\mathbf{id}|\ell-1}, \mathbf{id})$. The delegation algorithm derives a user secret key $usk_{\mathbf{id}} = (\mathbf{B_{id}}, \mathbf{e_{id}})$ for an identity $\mathbf{id} = (id_1, \ldots, id_\ell)$ $(1 \le \ell \le d)$ given a user secret key $(\mathbf{B}_{\mathbf{id}|\ell-1}, \mathbf{e}_{\mathbf{id}|\ell-1})$ for $\mathbf{id}|\ell-1$. The short vector $\mathbf{e}_{\mathbf{id}|\ell-1}$ will not be needed for delegation. Inductively, we have $\|\widetilde{\mathbf{B}}_{\mathbf{id}|\ell-1}\| \le L(\ell-1)$. Note that $\mathbf{A_{id}} = [\mathbf{A}, \mathbf{A}_1, \cdots, \mathbf{A}_\ell] = [\mathbf{A}_{\mathbf{id}|\ell-1}, \mathbf{A}_\ell] \in \mathbb{Z}_q^{n \times (\ell+1)m}$. To compute the basis part $\mathbf{B_{id}}$ of $usk_{\mathbf{id}}$, run $\mathbf{B_{id}} \leftarrow \mathsf{SampleBasis}(\mathbf{A_{id}}, \mathbf{B}_{\mathbf{id}|\ell-1}, S = \{1, \ldots, \ell\}, L(\ell))$. Note that since ℓ is constant,

$$L(\ell) = L(\ell-1) \cdot \sqrt{m} \cdot \omega(\sqrt{\log m}) \ge \|\widetilde{\mathbf{B}}_{\mathbf{id}|\ell-1}\| \cdot \sqrt{(\ell+1)m} \cdot \omega(\sqrt{\log{(\ell+1)m}}).$$

Therefore, by Theorem 3.3, we have $\|\widetilde{\mathbf{B}}_{\mathbf{id}}\| \le L(\ell)$. The syndrome part $\mathbf{e_{id}}$ of the user secret key is computed as $\mathbf{e_{id}} \leftarrow \mathsf{SamplePre}(\mathbf{A}_{\mathbf{id}|\ell-1}, \mathbf{B}_{\mathbf{id}|\ell-1}, S = \{1, \ldots, \ell\}, \mathbf{y_{id}}, r(\ell))$. Note that by Theorems 3.3 and 3.4 the user secret key $usk_{\mathbf{id}} = (\mathbf{B_{id}}, \mathbf{e_{id}})$ satisfies (4.2) and furthermore it has a distribution that is statistically close to the one computed by $\mathsf{HIBEExt}$.

$\mathsf{HIBEEnc}(mpk, \mathbf{id}, b)$. Say $\mathbf{id} = (id_1, \ldots, id_\ell)$ is an ℓ-level identity. The ciphertext is computed by running $\mathsf{Enc}(pk = (\mathbf{A}_{\mathbf{id}|\ell-1}, \mathbf{y_{id}}), b)$ with Gaussian parameter $\alpha = \alpha(\ell)$. This yields a ciphertext

$$C = (\mathbf{p}, c) \in \mathbb{Z}_q^{\ell m} \times \mathbb{Z}_q.$$

$\mathsf{HIBEDec}(usk_{\mathbf{id}}, (\mathbf{p}, c))$. Decryption uses only $\mathbf{e_{id}}$; it runs $\mathsf{Dec}(sk = \mathbf{e_{id}}, C = (\mathbf{p}, c))$. That is, it computes $b' = c - \mathbf{e}_{\mathbf{id}}^T \mathbf{p} \in \mathbb{Z}_q$, and outputs 0 if b' is closer to 0 than q, and 1 otherwise.

Correctness follows from our choice of parameters combined with Lemma 2.3. A multi-bit HIBE follows in the same way from the multi-bit PKE scheme by letting hash function G map into multiple uniform syndromes in $\mathbb{Z}_q^n$, one for each bit of the message.

4.2 Security

Theorem 4.1. *Let d be the maximal depth of the HIBE, $q \ge 5r(d)(m+1)$ and $m \ge 2n \lg q$. If G and H are modeled as random oracles, then* RO-HIBE *is CPA secure, assuming that* $\mathsf{LWE}_{q,\chi}$ *is hard, where $\chi = \bar{\Psi}_{\alpha(d)}$.*

Proof. Let d be the maximal depth of the HIBE system. Let A be an adversary attacking the CPA security of RO-HIBE. We assume, without loss of generality, that

- A has distinguishing advantage $\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{RO\text{-}HIBE},\mathsf{A}}(k)$ in the CPA experiment,
- A always makes exactly Q_G different G-queries for some polynomial Q_G,
- for each $i \in [d]$, A always makes exactly Q_H different H-queries of length i (i.e., of the form $(id_1, \ldots, id_i)$) for some polynomial Q_H,
- whenever A makes an H-query $(id_1, \ldots, id_i)$, it has queried $H(id_1, \ldots, id_j)$ for all $j < i$ beforehand,

- whenever A submits a user secret key query or a challenge identity, it has made all relevant G, resp. H-queries beforehand.

We now construct an adversary B that has advantage $\mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{B}}(k)$ in attacking the LWE problem, where

$$\mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{B}}(k) \geq \frac{\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{RO\text{-}HIBE},\mathsf{A}}(k)}{d \cdot Q_G \cdot Q_H^{d-1}} - \mathrm{negl}\,. \tag{4.3}$$

Adversary B first uniformly picks $\ell^* \in [d]$. (ℓ^* is a guess for the length of the challenge identity.) Next, B obtains $(\ell^*+1)(m+1)$ samples from the LWE oracle which get parsed as $(\mathbf{A}_i^*, \mathbf{p}_i^*) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$ $(0 \leq i \leq \ell^* - 1)$ and $(\mathbf{y}^*, c^*) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$. It sets the master public key to be $mpk = \mathbf{A} = \mathbf{A}_0^*$, the master secret key (a short basis for $\Lambda_q^{\perp}(\mathbf{A})$) is unknown to B. Next, adversary B chooses a random vector $\mathbf{j}^* = (j_1^*, \ldots, j_{\ell^*-1}^*) \in \{1, \ldots, Q_H\}^{\ell^*-1}$ and a random index $j \in \{1, \ldots, Q_G\}$, both uniformly distributed.

Adversary B initializes two lists $\mathcal{G}$ and $\mathcal{H}$ and executes adversary A on input mpk.

Queries to $H(\cdot)$. On A's j_i-th distinct query $(id_{j_i,1}, \ldots, id_{j_i,i})$ to $H(\cdot)$ of length i, do the following: if $i \leq \ell^*$ and $j_i = j_i^*$, then return $\mathbf{A}_i^*$ (as obtained during setup from the LWE oracle). Otherwise, if $i > \ell^*$ or $j_i \neq j_i^*$, run the trapdoor algorithm TrapGen to generate $\mathbf{A}_{i,j_i} \in \mathbb{Z}_q^{n\times m}$ with corresponding trapdoor $\mathbf{T}_{i,j_i} \in \mathbb{Z}^{m\times m}$. Return $\mathbf{A}_{i,j_i}$ and store the tuple $((id_{j_i,1}, \ldots, id_{j_i,i}), \mathbf{A}_{i,j_i}, \mathbf{T}_{i,j_i})$ in list $\mathcal{H}$. Note that by Lemma 3.1, $\mathbf{A}_{i,j_i}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$.

Queries to $G(\cdot)$. On A's j-th distinct query $\mathbf{id}_j$ to $G(\cdot)$, do the following: if $j = j^*$ then return $\mathbf{y}^*$ (obtained during setup from the LWE oracle). Otherwise for $j \neq j^*$, pick an error vector $\mathbf{e}_j \leftarrow D_{\mathbb{Z}^{\ell m}, r(\ell)}$ (where ℓ is the length of $\mathbf{id}_j$) and set $\mathbf{y}_j := \mathbf{A}_{\mathbf{id}_j|\ell-1}\mathbf{e}_j \in \mathbb{Z}_q^n$. (Recall that we assumed that A has already made all relevant queries to H that define $\mathbf{A}_{\mathbf{id}_j|\ell-1}$.) Return $\mathbf{y}_j$ and store $(\mathbf{id}_j, \mathbf{y}_j, \mathbf{e}_j)$ in list $\mathcal{G}$. Note that according to Lemma 2.2 (2), $\mathbf{y}_j$ is statistically close to uniform on $\mathbb{Z}_q^n$.

Queries to HIBEExt. When A asks for a user secret key for $\mathbf{id} = (id_1, \ldots, id_\ell)$, we again assume that A has already made all relevant queries to G and H that define $\mathbf{y}_{\mathbf{id}}$ and $\mathbf{A}_{\mathbf{id}}$. If, for one $i \in \{1, \ldots, \ell\}$, $\mathbf{A}_{id_i} = H(\mathbf{id}|i)$ is contained in list $\mathcal{H}$, then B can compute a properly distributed basis $\mathbf{B}_{\mathbf{id}}$ for $\mathbf{A}_{\mathbf{id}}$ by running $\mathbf{B}_{\mathbf{id}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{id}}, \mathbf{B}_{i,id_i}, S = \{i\}, L(\ell))$ from Theorem 3.3. If $\mathbf{y}_{\mathbf{id}}$ is contained in list $\mathcal{G}$, then B can retrieve a properly distributed vector $\mathbf{e}_{\mathbf{id}}$ from $\mathcal{G}$ with $\mathbf{y}_{\mathbf{id}} = \mathbf{A}_{\mathbf{id}|\ell-1}\mathbf{e}_{\mathbf{id}}$. If the generation of $usk_{\mathbf{id}} = (\mathbf{B}_{\mathbf{id}}, \mathbf{e}_{\mathbf{id}})$ was successful, then B returns $usk_{\mathbf{id}}$. In all other cases, B aborts (and returns a random bit).

Challenge. When A submits a challenge identity $\mathbf{id}^* = (id_1^*, \ldots, id_{\ell'}^*)$ (that is not a parent of all its user secret-key queries), assume again that A has already made all relevant queries to G and H that define $\mathbf{y}_{\mathbf{id}^*}$ and $\mathbf{A}_{\mathbf{id}^*|\ell^*-1}$. If $\ell' \neq \ell^*$, then B aborts and returns a random bit. Otherwise, if $\ell^* = \ell'$ consider matrix $\mathbf{A}_{\mathbf{id}^*|\ell^*-1}$ and syndrome $\mathbf{y}_{\mathbf{id}^*}$ that is used for the encryption procedure. If one of the sub-matrices $H(\mathbf{id}^*|i)$ $(1 \leq i \leq \ell^* - 1)$ of $\mathbf{A}_{\mathbf{id}^*|\ell^*-1}$ is contained in list $\mathcal{H}$ or $\mathbf{y}_{\mathbf{id}^*}$ is contained in list $\mathcal{G}$, then B aborts and returns a random bit. Otherwise we have $\mathbf{A}_{\mathbf{id}^*|\ell^*-1} = [A_0^*, \ldots, A_{\ell-1}^*]$ and $\mathbf{y}_{\mathbf{id}}^* = y^*$ and B generates a challenge $C^* = (\mathbf{p}^*, c^*)$ for bit b as $\mathbf{p}^* = (\mathbf{p}_0^*, \ldots, \mathbf{p}_{\ell-1}^*)$ and $c^* = d + b\lfloor q/2 \rceil$.

When A terminates with some output, B also terminates with the same output.

It remains to analyze the reduction. It is easy to see that the probability of an abort is $1 - \frac{1}{dQ_G Q_H^{\ell^*-1}}$, and only depends on ℓ^*. Implementing a straightforward additional artificial abort step, this probability of an abort can be raised to $1 - \frac{1}{dQ_G Q_H^{d-1}}$, independently of A's view. We already showed that if B does not abort during the $\mathsf{HIBEExt}$ queries, then the distribution of its answers is statistically close to the one from the real HIBE CPA experiment. Furthermore, if B does not abort during the challenge query, then the distribution of the challenge ciphertext is distributed as in the real, resp. ideal HIBE CPA experiment, depending on whether the LWE sample is real, resp. random. Therefore, conditioned on B not aborting, A's view is statistically close to the one provided by the real HIBE CPA security experiment. This proves (4.3).

We note that with a different simulation technique from [Cor02] it is furthermore possible to improve the reduction from (4.3) to $\mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{B}}(k) = \Theta(\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{RO\text{-}HIBE},\mathsf{A}}(k)/(dQ_E^d))$, where Q_E is an upper bound on the number of extraction queries.

□

5 The HIBE Scheme in the Standard Model

Overall strategy. Recall that in $\mathsf{RO\text{-}HIBE}$, an identity $\mathbf{id} = (id_1, \ldots, id_\ell)$ gives rise to a "user public key" $(\mathbf{A_{id}}, \mathbf{y_{id}})$. Here, $\mathbf{A_{id}} \in \mathbb{Z}_q^{n\times(\ell+1)m}$ and $\mathbf{y_{id}} \in \mathbb{Z}_q^n$ are derived using H and G, respectively. The user public key $(\mathbf{A_{id}}, \mathbf{y_{id}})$ can then be used to encrypt as in [GPV08]'s public-key encryption scheme $\mathsf{LWE\text{-}PKE}$.

The advantage of this approach is that we can use $\mathsf{LWE\text{-}PKE}$ (and its security proof) almost as a black box. However, a technical difficulty turns up in the security proof. Namely, we have to construct a $\mathsf{LWE\text{-}PKE}$ challenge associated to user public key $(\mathbf{A_{id^*}}, \mathbf{y_{id^*}})$, while at the same time being able to simulate user secret keys for all other queried identities. This is particularly difficult since we do not know the challenge identity $\mathbf{id}^*$ in advance, while setting up a master public key for our simulation.

To prove $\mathsf{RO\text{-}HIBE}$ secure in the random oracle model, we guessed $\mathbf{id}^*$ (by guessing in which H-query it appears), and embedded the $\mathsf{LWE\text{-}PKE}$ challenge into the values $H(\mathbf{id}^*|i)$ and $G(\mathbf{id}^*)$ on the fly. All other H-queries were answered with matrices with known trapdoor. This allowed for a straightforward simulation (provided the guess for $\mathbf{id}^*$ is correct), but requires a programmable random oracle.

Trapdoor and challenge matrices. In the standard model, we hence have to work harder. We present our argument first for the selective-identity case, and then show how to generalize it. Recall that in the selective-identity security experiment, the adversary has to decide first on the challenge identity, and then receives the public key. In the simulation, we can hence set up the public key depending on the challenge identity.

Concretely, we will distinguish two types of matrices $\mathbf{A}$. If we know (or can efficiently derive) a short basis in the sense of Lemma 3.1 for $\mathbf{A}$, we call $\mathbf{A}$ *trapdoor.* If, on the other hand, $\mathbf{A}$ is comprised solely of columns output by the LWE oracle LWE, we call $\mathbf{A}$ *challenge.* In our security proof, all occuring matrices will either be trapdoor or challenge. Obviously, a matrix $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_\ell]$ is challenge iff all $\mathbf{A}_i$ are challenge. Furthermore, by Theorem 3.3, $\mathbf{A}$ is trapdoor iff at least one $\mathbf{A}_i$ is trapdoor.

As with $\mathsf{RO\text{-}HIBE}$, each HIBE identity $\mathbf{id} = (id_1, \ldots, id_\ell)$ will give rise to a "user public key" matrix

$$\mathbf{A}_{\mathbf{id}} = [\mathbf{A}, \mathbf{A}_{1,id_1}, \ldots, \mathbf{A}_{\ell,id_\ell}].$$

Again, the corresponding user secret key will contain a short basis for $\mathbf{A}_{\mathbf{id}}$. The master secret key will be a short basis for $\mathbf{A}$, so thanks to Theorem 3.3, the master secret key allows to compute user secret keys for arbitrary identities. However, during simulation, $\mathbf{A}$ will be challenge, so that a user secret key can be constructed iff at least one of the $\mathbf{A}_{i,id_i}$ is trapdoor. We will set up the $\mathbf{A}_{i,id_i}$ such that

- for all $\mathbf{id}$ that appear in IBEExt queries, $\mathbf{A}_{\mathbf{id}}$ is trapdoor, and
- for the challenge identity $\mathbf{id}^*$, $\mathbf{A}_{\mathbf{id}^*}$ is challenge.

The implementation of the hash function. We will accomplish this by setting up a mapping from identities to matrices as follows:

$$\mathbf{A}_{i,id_i} := [\mathbf{C}_{i,1,t_1}, \ldots, \mathbf{C}_{i,\lambda,t_\lambda}]$$

with $(t_1, \ldots, t_\lambda) := H_i(id_i) \in \{0,1\}^\lambda$ for suitable hash functions H_i, and matrices $\mathbf{C}_{i,u,b}$ ($1 \leq u \leq \lambda$, $0 \leq b \leq 1$) contained in the master public key. Now if the challenge identity $\mathbf{id}^*$ is known at the time of setup, we can choose the $\mathbf{C}_{i,u,b}$ as challenge if and only if the u-th bit of $H_i(id_i^*)$ is b. This way, *only* $\mathbf{A}_{\mathbf{id}^*}$ consists solely of challenge matrices, and hence is challenge. Every user public key matrix $\mathbf{A}_{\mathbf{id}}$ for $\mathbf{id} \neq \mathbf{id}^*$ contains at least one trapdoor matrix, and hence is trapdoor. (Note the similarity to the public key setup from [DDN91].) The setup is completely oblivious to an adversary. This enables a successful simulation.

From selective-identity to full security. The previously sketched setup depends of course heavily on the challenge identity $\mathbf{id}^*$ in advance. In the general IBE security experiment, however, the adversary may choose $\mathbf{id}^*$ dynamically and adaptively. To overcome this obstacle, we will employ a probabilistic argument, along the lines of [BB04b]. Concretely, we will set up the public key such that each $\mathbf{A}_{\mathbf{id}}$ is challenge with a certain probability. A sophisticated construction of the hash functions H_i will ensure that, to a certain degree, these probabilities (resp. the corresponding events) are independent. That is, even an adversary that adaptively asks for user secret keys will not manage to produce an identity $\mathbf{id}$ for which $\mathbf{A}_{\mathbf{id}}$ is *guaranteed* to be challenge or trapdoor. Setting the probabilities in the right way (depending on the adversary's complexity), we can achieve that with non-negligible probability, all $\mathbf{A}_{\mathbf{id}}$ associated with user secret key queries are trapdoor, while $\mathbf{A}_{\mathbf{id}^*}$ is challenge. In this case, a successful simulation will be possible.

Of course, we will have to take care that the event of a successful simulation is (at least approximately) independent of the adversary's view. To achieve independence, we will employ an "artificial abort" strategy similar to the one from [Wat05].

One scheme, two results. We stress that we can get very different results about one single scheme. We can prove our scheme selective-identity secure, given that the hash function is collision-resistant (and, of course, assuming that the learning with error problem is hard). To prove the same scheme fully secure, we need an additional assumption about the hash function. (This additional assumption will enable the probabilistic argument sketched above.)

5.1 The scheme SM-HIBE

Let $d \in \mathbb{N}$ denote the maximal depth of the HIBE. Let $\mathcal{H} = (\mathcal{H}_k)_k$ be a family of hash functions $H : \{0,1\}^k \to \{0,1\}^\lambda$. The security properties of the hash function and the parameter $\lambda = \lambda(k)$ will be selected depending if the scheme is fully secure or only selective-identity secure. Again the scheme is parametrized by L and some functions $L(\ell)$, $r(\ell)$, and $\alpha(\ell)$ ($1 \le \ell \le d$), which we fix as follows.

$$
\begin{array}{ll}
L \ge m \cdot \omega(\sqrt{\log n}) & L_\lambda(\ell) = L \cdot (m\lambda)^{\ell/2} \cdot \omega(\lg^{\ell/2} m) \\
r_\lambda(\ell) = L_\lambda(\ell) & \alpha_\lambda(\ell) \le 1/(r_\lambda(\ell) \cdot \sqrt{(\lambda\ell+1)m+1} \cdot \omega(\lg^{1/2} n))
\end{array}
$$

$\mathsf{IBESetup}$. Using the trapdoor algorithm $\mathsf{TrapGen}$, generate $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and a corresponding short basis $\mathbf{B} \in \mathbb{Z}^{m\times m}$ with $\|\widetilde{\mathbf{B}}\| \le L_\lambda$. Furthermore, sample uniformly and independently matrices $\mathbf{C}_{i,u,b} \in \mathbb{Z}_q^{n\times m}$ (for $1 \le i \le d, 1 \le u \le \lambda$ and $0 \le b \le 1$) and a vector $\mathbf{y} \in \mathbb{Z}_q^n$. Finally, choose $H_1, \ldots, H_d \leftarrow \mathcal{H}_k$. Return

$$
mpk = (\mathbf{A}, \mathbf{y}, (\mathbf{C}_{i,u,b})_{\substack{1\le i\le d\\ 1\le u\le\lambda \\ 0\le b\le 1}}, (H_i)_{i=1}^d) \qquad\qquad msk = (mpk, \mathbf{B}).
$$

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

$$
\mathbf{A}_{\mathbf{id}} := [\mathbf{A}, \mathbf{A}_{1,id_1}, \ldots, \mathbf{A}_{\ell,id_\ell}] \in \mathbb{Z}_q^{n\times(\lambda\ell+1)m} \tag{5.1}
$$

where

$$
\mathbf{A}_{i,id_i} := [\mathbf{C}_{i,1,t_1}, \ldots, \mathbf{C}_{i,\lambda,t_\lambda}] \in \mathbb{Z}_q^{n\times\lambda m} \tag{5.2}
$$

for $(t_1, \ldots, t_\lambda) := H_i(id_i) \in \{0,1\}^\lambda$. The user secret keys for an identity $\mathbf{id}$ will consist of a basis part $\mathbf{B}_{\mathbf{id}}$ for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{id}})$ and a syndrome part $\mathbf{e}_{\mathbf{id}}$ satisfying $\mathbf{A}_{\mathbf{id}}\mathbf{e}_{\mathbf{id}} = \mathbf{y}$, where

$$
\|\|\mathbf{B}\|_{\mathbf{id}}\| \le L_\lambda(\ell), \qquad \|\mathbf{e}_{\mathbf{id}}\| \le r_\lambda(\ell) \ . \tag{5.3}
$$

Note that by construction $\mathbf{B}_\varepsilon$ is a short basis for $\mathbf{A}_\epsilon = \mathbf{A}$ satisfying (5.3).

$\mathsf{HIBEExt}(msk, \mathbf{id})$. This algorithm computes a user secret key for $\mathbf{id} = (id_1, \ldots, id_\ell)$ which consists of $(\mathbf{B}_{\mathbf{id}}, \mathbf{e}_{\mathbf{id}})$ where $\mathbf{B}_{\mathbf{id}} \leftarrow \mathsf{SampleBasis}(\mathbf{A}_{\mathbf{id}}, \mathbf{B}_\varepsilon, S = \{1\}, L_\lambda(\ell))$ is a basis for $\Lambda_q^\perp(\mathbf{A}_{\mathbf{id}})$ and $\mathbf{e}_{\mathbf{id}} \leftarrow \mathsf{GenSamplePre}(\mathbf{A}_{\mathbf{id}}, \mathbf{B}_\varepsilon, S = \{1\}, \mathbf{y}_{\mathbf{id}}, r_\lambda(\ell))$ is distributed according to $D_{\mathbb{Z}^{(\lambda\ell+1)m}, r_\lambda(\ell)}$ conditioned on $\mathbf{A}_{\mathbf{id}}\mathbf{e}_{\mathbf{id}} = \mathbf{y}_{\mathbf{id}}$. Note that by Theorems 3.3 and 3.4, $usk_{\mathbf{id}} = (\mathbf{B}_{\mathbf{id}}, \mathbf{e}_{\mathbf{id}})$ satisfies (5.3).

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

$$
L(\ell) = L(\ell-1) \cdot \sqrt{\lambda m} \cdot \omega(\sqrt{\log \lambda m}) \ge \|\widetilde{\mathbf{B}}_{\mathbf{id}|\ell-1}\| \cdot \sqrt{(\lambda\ell+1)m} \cdot \omega(\sqrt{\log{(\lambda\ell+1)m}})
$$

and therefore by Theorem 3.3 we have $\|\widetilde{\mathbf{B}}_{\mathbf{id}}\| \le L_\lambda(\ell)$. The syndrome part of the user secret key is computed as $\mathbf{e}_{\mathbf{id}} \leftarrow \mathsf{SamplePre}(\mathbf{A}_{\mathbf{id}}, \mathbf{B}_{\mathbf{id}|\ell-1}, S = \{1,\dots,\lambda\ell\}, \mathbf{y}, r_\lambda(\ell))$. Note that by Theorems 3.3 and 3.4 the user secret key $usk_{\mathbf{id}} = (\mathbf{B}_{\mathbf{id}}, \mathbf{e}_{\mathbf{id}})$ satisfies (5.3) and furthermore it has a distribution that is statistically close to the one computed by $\mathsf{HIBEExt}$.

$\mathsf{HIBEEnc}(mpk, \mathbf{id}, b)$. Let $\mathbf{id}$ be at level ℓ. The ciphertext is computed by running $\mathsf{Enc}(pk = (\mathbf{A}_{\mathbf{id}}, \mathbf{y}), b)$ with Gaussian parameter $\alpha = \alpha_\lambda(\ell)$, which outputs the ciphertext

$$C = (\mathbf{p}, c).$$

$\mathsf{HIBEDec}(usk_{\mathbf{id}}, (\mathbf{p}, c))$. Decryption uses only $\mathbf{e}_{\mathbf{id}}$. It simply runs $\mathsf{Dec}(sk = \mathbf{e}_{\mathbf{id}}, C = (\mathbf{p}, c))$. That is, it computes $b' = c - \mathbf{e}_{\mathbf{id}}^T\mathbf{p} \in \mathbb{Z}_q$, and outputs 0 if b' is closer to 0 than q, and 1 otherwise.

The scheme's correctness is inherited by $\mathsf{LWE\text{-}PKE}$.

5.2 Security

We now formally state security of our construction. If the hash function $\mathcal{H}$ is collision-resistant then we can prove the scheme selective-identity CPA secure. This results in $\lambda = 2k$ and hence the scheme is roughly a factor k less efficient than our random-oracle construction. If the hash function fulfils the stronger security requirements of being admissable (to be defined in Section 5.3) then we can prove the scheme (full-identity) CPA secure. Unfortunately, we only know constructions of admissible hash functions that require $\lambda = k^{2+\varepsilon}$ so the resulting scheme is quite unpractical.

Theorem 5.1. *Let $q \ge 5r(d)(m+1)$, $\alpha_\lambda(\ell)$ be as above, and $m \ge 2n \lg q$. Assume $\mathsf{LWE}_{q,\chi}$ is hard, where $\chi = \bar{\Psi}_{\alpha(d)}$ and d is the maximal depth of the scheme. If $\mathcal{H}$ is a family of collision-resistant hash functions, then $\mathsf{SM\text{-}HIBE}$ is selective-ID CPA-secure. If $\mathcal{H}$ is a family of admissible hash functions, then $\mathsf{SM\text{-}HIBE}$ is CPA-secure.*

5.3 Admissible hash functions

We give a variant of the definition from [BB04b]. Let $\mathcal{H} = \{\mathcal{H}_k\}$ be a collection of distributions of functions $H : \mathcal{C}_k \to \mathcal{D}_k = \{0,1\}^\lambda$. For $H \in \mathcal{H}_k$, $K \in \{0,1,\perp\}^\lambda$, and $x \in \mathcal{C}_k$, define

$$F_{K,H}(x) = \begin{cases} \mathtt{B} & \text{if } \exists u \in \{1,\dots,\lambda\} : t_u = K_i \\ \mathtt{R} & \text{if } \forall u \in \{1,\dots,\lambda\} : t_u \ne K_i \end{cases} \quad \text{for } (t_1,\dots,t_\lambda) = H(x). \tag{5.4}$$

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

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

- For every PPT algorithm C that, on input a function $H \in \mathcal{H}_k$, outputs a vector $\mathbf{x} \in \mathcal{C}_k^{Q+1}$, the function
$$\mathbf{Adv}^{\mathsf{adm}}_{\mathcal{H},\mathsf{C}}(k) := \Pr[\mathbf{x} \in \mathsf{bad}_H \mid H \leftarrow \mathcal{H}_k \,;\, \mathbf{x} \leftarrow \mathsf{C}(H)]$$
is negligible in k.

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

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

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

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

Difference to the definition of [BB04b]. Note that our definition of admissibility is conceptually different from that of [BB04b]. The reason for our change is that our definition is better suited for our purposes. Concretely, their definition is based upon indistinguishability from a (biased) random function. However, their construction only achieves asymptotic indistinguishability (i.e., negligible distinguishing success) when the "target" random function is constant. (In their notation, this corresponds to the case when γ is negligible, so that $\Pr[F_{K,H}(x) = 1] \approx 1$.) Such a function is not very useful for our (or their) purposes. In an asymptotic sense, their construction becomes only useful with parameters that cause the distinguishing advantage to become significant (but smaller than the inverse of a given polynomial). With that parameter choice, our definition allows for a conceptually simpler analysis. Namely, it separates certain negligible error probabilities (of $\mathbf{x} \in \mathsf{bad}_H$) from significant, but purely combinatorial bounds on the probability of the "simulation-enabling" event

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

Specifically, we can bound $\Pr[\mathsf{good}]$ for *every* $\mathbf{x} \notin \mathsf{bad}_H$, which simplifies the artificial abort step below. Note that while the original analysis from [BB04b] does not incorporate an artificial abort step, this actually would have been necessary to guarantee sufficient independence of (their version of) event good. This becomes an issue in [BB04b, Claim 2], when the success probability of an adversary conditioned on good is related to its original (unconditioned) success probability.

Constructions. [BB04b] show how to construct admissible hash functions from a given collision-resistant hash function family. Since collision-resistant hash functions can be built from the LWE problem (e.g., [Ajt04]), this does not entail extra assumptions in the encryption context. Specifically, for parameter choices as in [BB04b, Section 5.3], we get a single hash function with output length $\lambda = O(k^{2+\varepsilon})$ (for arbitrary $\varepsilon > 0$) that is Δ-admissible with $\Delta = \Theta(1/Q^2)$.[5]

5.4 Proof of full-identity security

We prove the following lemma.

Lemma 5.2. *Assume an adversary* A *on* $\mathsf{SM\text{-}HIBE}$*'s CPA security that makes at most* $Q(k)$ *user secret key queries. Then, for every polynomial* $S = S(k)$*, there exists an* $\mathsf{LWE}_{q,\chi}$*-distinguisher* D *and an adversary* C *on* $\mathcal{H}$*'s admissibility such that*

$$\mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}(k) \leq d \cdot \mathbf{Adv}^{\mathsf{adm}}_{\mathcal{H},\mathsf{C}}(k) + \frac{\mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{D}}(k)}{\Delta(k,Q)^d} + \frac{1}{S(k)} + \operatorname{negl}. \tag{5.5}$$

[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.

Here, the running time of C *is roughly that of the IBE experiment* $\mathbf{Exp}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}$*, and the running time of* D *is roughly that of* $\mathbf{Exp}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}$ *plus* $O(k^2QS/\Delta^d)$ *steps.*

Note that for the admissible hash function from [BB04b], $\Delta(k,Q)^d = \Theta(1/Q^{2d})$ is significant. Since S in Lemma 5.2 is arbitrary, we obtain:

Corollary 5.3 (SM-HIBE is CPA secure)**.** *If* $\mathcal{H}$ *is admissible, and if the* $\mathsf{LWE}_{q,\chi}$ *problem is hard, then* SM-HIBE *is CPA secure.*

Proof of Lemma 5.2. We proceed in games, with **Game** 0 being the original $\mathbf{Exp}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}$ experiment with adversary A. We assume without loss of generality that A always makes exactly $Q = Q(k)$ user secret key queries. We denote these queries by by $\mathbf{id}^j = (id_1^j, \dots, id_{\ell_j}^j)$ (for $1 \le j \le Q$), and the challenge identity chosen by A as $\mathbf{id}^* = (id_1^*, \dots, id_{\ell^*}^*)$. By out_i, we denote the experiment's output in Game i. By definition,

$$|\Pr[out_0 = 1] - 1/2| = \mathbf{Adv}^{\mathsf{hibe\text{-}cpa}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}(k). \tag{5.6}$$

In the following, let $\mathcal{ID}_i^Q := \bigcup_j \{id_i^j\}$ 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 $1 \le |\mathcal{ID}_i^Q| \le Q$ and $0 \le |\mathcal{ID}_i^*| \le 1$. For our upcoming probabilistic argument we need actually identity sets of a fixed size, so we pad all $\mathcal{ID}_i^Q$ and $\mathcal{ID}_i^*$ in some canonical way with unused identities. Concretely, for all i, let $\mathcal{ID}_i^{\mathsf{R}} \supseteq \mathcal{ID}_i^*$ and $\mathcal{ID}_i^{\mathsf{B}} \supseteq \mathcal{ID}_i^Q \setminus \mathcal{ID}_i^*$ be disjoint, and such that $|\mathcal{ID}_i^{\mathsf{R}}| = 1$ and $|\mathcal{ID}_i^{\mathsf{B}}| = Q$. Let $\overrightarrow{\mathcal{ID}_i} \in (\{0,1\}^k)^{Q+1}$ be the vector whose first component is the (unique) element of $\mathcal{ID}_i^{\mathsf{R}}$, and whose remaining Q components are the elements of $\mathcal{ID}_i^{\mathsf{B}}$ (in some canonical order).

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

$$|\Pr[out_1 = 1] - \Pr[out_0]| \le d \cdot \mathbf{Adv}^{\mathsf{adm}}_{\mathcal{H},\mathsf{C}}(k). \tag{5.7}$$

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

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

$$\Pr[out_2 = 1] - 1/2 = \Pr[\mathsf{good}_2]\left(\Pr[out_1 = 1] - 1/2\right) = \Delta^d\left(\Pr[out_1 = 1] - 1/2\right). \tag{5.8}$$

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

$$F_i(id) = F_{K^i,H_i}(id) = \begin{cases} \mathtt{B} & \text{if } \exists u \in \{1,\dots,\lambda\} : t_u = K_u^i \\ \mathtt{R} & \text{if } \forall u \in \{1,\dots,\lambda\} : t_u \neq K_u^i \end{cases} \quad \text{for } (t_1,\dots,t_\lambda) = H_i(id),$$

associated with H_i. 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 $F_i(id) = \mathtt{B}$, then id is *blue*, and if $F_i(id) = \mathtt{R}$, then id is *red*. Later on, blue will correspond to trapdoor matrices, while red will correspond to challenge matrices.

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

Ideally, we would like to replace event good_2 from Game 2 with event E. Unfortunately, however, E might not be independent of 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 $\lceil kS/\Delta^d \rceil$ samples, we can obtain an approximation $\tilde{p_E} \geq \Delta^d$ of p_E that satisfies

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

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 $1/2^k$,

$$|\Pr[\mathsf{good}_3] - \Pr[\mathsf{good}_2]| = \left|\Delta^d - \Delta^d \frac{p_E}{\tilde{p_E}}\right| = \Delta^d \left|\frac{\tilde{p_E} - p_E}{\tilde{p_E}}\right| \leq \Delta^d \frac{\Delta^d}{S\tilde{p_E}} \leq \frac{\Delta^d}{S}. \tag{5.9}$$

Since (5.9) holds for arbitrary $\overrightarrow{\mathcal{ID}_i}$ except with probability $1/2^k$, we obtain that the statistical distance between the output of Game 2 and Game 3 is bounded by $\Delta^d/S + 2^{-k}$. Hence,

$$|\Pr[out_3 = 1] - \Pr[out_2 = 1]| \leq \frac{\Delta^d}{S} + \frac{1}{2^k}. \tag{5.10}$$

In **Game** 4, we set up the public key differently. We call matrices that are chosen uniformly *challenge,* and matrices that are chosen along with a short basis (using algorithm $\mathsf{TrapGen}$) *trapdoor.* Now in Game 4, we will set up the public key as follows:

- $\mathbf{A}$ as trapdoor (as in the earlier games),
- $\mathbf{C}_{i,u,b}$ as trapdoor if $K_u^i = b$ (and as challenge if $K_u^i \neq b$).

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

$$|\Pr[out_4 = 1] - \Pr[out_3 = 1]| = \text{negl}\,. \tag{5.11}$$

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

$$\Pr[out_5 = 1] = \Pr[out_4 = 1]. \tag{5.12}$$

In **Game** 6, we change the way user secret keys queries $\mathbf{id} = (id_1, \ldots, id_\ell)$ are answered. By the change from Game 5, we may assume that $F_i(id_i) = \mathsf{B}$ for some i. Hence, $t_u = K_u^i$ for $(t_1, \ldots, t_\lambda) = H_i(id_i)$ and some u. Thus, the matrix $\mathbf{C}_{i,u,t_u}$ that appears in the decomposition of $\mathbf{A}_{\mathbf{id}}$ (see (5.1,5.2)) is trapdoor by our public key setup. To generate a user secret key, we have to find a short basis for $\mathbf{A}_{\mathbf{id}}$. In Game 4, this is achieved by algorithms $\mathsf{SampleBasis}$ and $\mathsf{GenSamplePre}$, using the short basis of the first matrix $\mathbf{A}$ in the decomposition of $\mathbf{A}_{\mathbf{id}}$. In Game 6, we instead use the short basis of $\mathbf{C}_{i,u,t_u}$ that we have initially generated. By Theorem 3.3, this results in the same distribution of bases $usk_{\mathbf{id}} = (\mathbf{B}_{\mathbf{id}}, \mathbf{e}_{\mathbf{id}})$, up to negligible statistical distance. Hence

$$|\Pr[out_6 = 1] - \Pr[out_5 = 1]| = \text{negl}\,. \tag{5.13}$$

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

$$|\Pr[out_7 = 1] - \Pr[out_6 = 1]| = \text{negl}\,. \tag{5.14}$$

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

$$\Pr[out_8 = 1] = 1/2. \tag{5.15}$$

We furthermore claim that

$$|\Pr[out_8 = 1] - \Pr[out_7 = 1]| \leq \mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{D}}(k) \tag{5.16}$$

for the following LWE distinguisher D. Recall that D has access to either

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

D will simulate Game 7 and use its oracle to help set up parts of the public key and the challenge ciphertext. First, D samples $(\mathbf{y}, c_{\mathbf{y}})$ to obtain the $\mathbf{y}$ part of the public key. Then, D samples all challenge matrices in the public key by querying its oracle m times respectively (and adjoining the outputs). This way, D obtains $(\mathbf{A}, \mathbf{g}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ from the oracle such that

$$\mathbf{g} = \begin{cases} \mathbf{A}^T\mathbf{s} + \mathbf{x} & \text{if } \mathsf{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathsf{D} \text{ runs with random oracle.} \end{cases} \tag{5.17}$$

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

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

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

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

$$\begin{aligned}
\mathbf{p}^* &:= \mathbf{g} + \sum_{v=1}^{\ell\lambda} \mathbf{g}_{i_v,u_v,b_v} \overset{(5.17)}{=} \begin{cases} \mathbf{A}_{\mathbf{id}^*}^T \mathbf{s} + \mathbf{x} & \text{if } \mathsf{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathsf{D} \text{ runs with random oracle,} \end{cases} \\
c^* &:= c_{\mathbf{y}} + b\lfloor \frac{q}{2} \rfloor \overset{(5.17)}{=} \begin{cases} \mathbf{y}^T \mathbf{s} + x + b\lfloor \frac{q}{2} \rfloor & \text{if } \mathsf{D} \text{ runs with real oracle} \\ \text{uniformly random} & \text{if } \mathsf{D} \text{ runs with random oracle.} \end{cases}
\end{aligned} \tag{5.18}$$

Finally, D outputs whatever the experiment outputs. By (5.18), D's outputs distribution is exactly that of Game 7, resp. Game 8 if it interacts with the real, resp. random oracle. (5.16) follows.

Taking (5.6-5.16) together shows (5.5). □

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 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 did not need this approximation/abort step at all, since (5.9) and hence (5.10) 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 $\mathcal{H}$ queries. Hence, to achieve $|p_E - \Delta^d| < \Delta^d/S$, we would need to consider arbitrary polynomial values of Q and adjust the $\mathcal{H}$ parameters accordingly. Of course, in an asymptotic sense, we already do consider arbitrary polynomial values of Q, because A may make up to Q user secret key queries. However, in a concrete sense, the number of (online) user secret key queries will be much lower than the inverse of A's distinguishing advantage. Hence, when considering concrete parameters, we can work with much smaller $\mathcal{H}$ parameters when implementing our artificial abort step, at the price of a worse reduction. This is why we decided for an artificial abort step.

5.5 Proof of selective-identity security

The use of admissible hash functions makes $\mathsf{SM\text{-}HIBE}$ comparatively inefficient. Recall that we used admissible hash functions solely to embed a challenge without knowing the challenge identity in advance. If we knew the challenge identity in advance (as with selective-identity security), we could set up the IBE public key so that *only* the challenge identity maps to a challenge matrix. In particular, instead of an admissible hash function $\mathcal{H}$, we could do with only a collision-resistant $\mathcal{H}$.[6] Formally:

Lemma 5.4 ($\mathsf{SM\text{-}HIBE}$ is sID secure.)**.** *Assume an adversary* A *on* $\mathsf{SM\text{-}HIBE}$*'s CPA security. Then there exists an* $\mathsf{LWE}_{q,\chi}$*-distinguisher* D *and an adversary* C *on* $\mathcal{H}$*'s collision-resistance (both with roughly the same complexity as* A*) such that*

$$\mathbf{Adv}^{\mathsf{hibe\text{-}sid}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}(k) \le d \cdot \mathbf{Adv}^{\mathsf{crhf}}_{\mathcal{H},\mathsf{C}}(k) + \mathbf{Adv}^{\mathsf{lwe}}_{q,\chi,\mathsf{D}}(k) + \mathrm{negl}\,. \tag{5.19}$$

[6] In fact, if only identities $id_i \in \{0,1\}^k$ are considered, $\mathcal{H}$ could even be the identity mapping.

Proof. The proof is very similar to the proof of Theorem 5.2, and we only detail the differences.

Game 0 is the original $\mathbf{Exp}^{\mathsf{hibe\text{-}sid}}_{\mathsf{SM\text{-}HIBE},\mathsf{A}}$ experiment. In **Game** 1, we let the experiment abort upon $\mathcal{H}$-collisions. (Hence we get the $d \cdot \mathbf{Adv}^{\mathsf{crhf}}_{\mathcal{H},\mathsf{C}}$ term in (5.19) instead of the $d \cdot \mathbf{Adv}^{\mathsf{adm}}_{\mathcal{H},\mathsf{C}}$ term in (5.5).) The abort and coloring steps that appear in **Games** 2 **and** 3 in the proof of Theorem 5.2 can be skipped in our case. For **Game** 4, note that in the selective-identity security experiment, the adversary has to choose the challenge identity $\mathbf{id}^* = (id_1^*, \ldots, id_{\ell^*}^*)$ in advance, before receiving the public key. Hence, in Game 4, we are free to set up the public key in dependence of $\mathbf{id}^*$. Specifically, $\mathbf{C}_{i,u,b}$ is chosen as trapdoor iff the u-th bit of $id_i^* \neq b$ (or if $i > \ell^*$). This way, all user public keys $\mathbf{A}_{\mathbf{id}}$ contain at least one trapdoor $\mathbf{C}_{i,u,b}$ unless $\mathbf{id} \neq \mathbf{id}^*$. Hence, **Games** 5-8 can be implemented as in the proof of Theorem 5.2, and (5.19) follows. $\square$

We stress that the use of only collision-resistant hash functions (or omitting $\mathcal{H}$ altogether in case $id_i \in \{0,1\}^k$) leads to a much more efficient scheme. In particular, public keys and ciphertexts now contain only $O(d \cdot k)$ matrices, resp. vectors.

6 Variants and Optimizations

6.1 Anonymous HIBE

Our two HIBE schemes RO-HIBE and SM-HIBE are not anonymous since their ciphertexts leak the length of the recipiet's identity. A straightforward padding scheme can be used to prevent this attack. That is, one padds the $\mathbf{c}$ part of the ciphertext by uniform random elements from $\mathbb{Z}_q$ to make the ciphertext look as a ciphertext for an identity of maximal length d. Anonymity then follows by the same reduction as CPA security, observing that the ciphertexts are pseudorandom and therefore do not reveal any information about the recipient's identity.

6.2 Chosen-ciphertext security

It is known how to construct CCA-secure PKE schemes from selective-identity secure IBE schemes (see [BCHK06]). Specifically, if we set $d = 1$ and omit the hash function $\mathcal{H}$ in the selective-identity secure HIBE scheme just discussed, we obtain a CCA-secure PKE scheme with the following efficiency characteristics:

- the public key contains $2k + 1$ matrices from $\mathbb{Z}_q^{n \times m}$,
- the secret key contains a matrix from $\mathbb{Z}_q^{m \times m}$,
- ciphertexts contain a vector $\mathbf{p} \in \mathbb{Z}_q^{(k+1)m}$, along with a padded message $c \in \mathbb{Z}_q$, and a verification key and a signature from a one-time signature scheme.

The security of that scheme can be reduced solely to the LWE problem. (Assuming that a suitable one-time signature scheme is chosen.) We remark that the resulting CCA-secure PKE scheme is very similar to the one recently proposed by Peikert [Pei09b].

Similarly, the transformation from [BCHK06] can be used to turn any (fully) CPA-secure d-level HIBE scheme[7] into a (fully) CCA-secure $(d-1)$-level HIBE scheme. Specifically, a 2-level CPA-secure HIBE scheme (such as our SM-HIBE scheme with $d = 2$) can be transformed into a fully CCA-secure IBE scheme.

[7]It is actually sufficient to provide full security at the first $(d-1)$ levels and only selective-ID security at the dth level.

6.3 Signature schemes

As noted by Naor (see [BF03]), our IBE scheme SM-HIBE immediately gives rise to a signature scheme. A signature for a message M is the user secret key for identity M which consist of a short vector $\mathbf{e}_M \in \Lambda_q^\perp(\mathbf{A}_M)$. (Matrix $\mathbf{A}_M$ is defined as in (5.1).) Signing is performed using the master secret key, and verification is performed by encrypting random messages under identity M and checking whether the signature works as a decryption key. We note that in our case the scheme can be further optimized. First, one can set $\mathbf{y} = 0$ in the master public-key and verification of a signature $\mathbf{e}_M$ is simply checking if $\|\mathbf{e}_M\| \leq r\sqrt{m}$ and if $\mathbf{e}_M \in \Lambda_q^\perp(\mathbf{A}_M)$. Furthermore, security of this signature scheme can be reduced to the average-case hardness of $\mathsf{SIS}_{q,2\lambda m,2r\sqrt{\lambda m}}$.

References

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

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, August 2005.

[Ajt99] Miklós Ajtai. Generating Hard Instances of the Short Basis Problem. In *ICALP*, pages 1–9, 1999.

[Ajt04] Miklós Ajtai. Generating Hard Instances of Lattice Problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[AP09] Joël Alwen and Chris Peikert. Generating Shorter Bases for Hard Random Lattices. In *STACS*, pages 75–86, 2009.

[BB04a] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, August 2004.

[BCHK06] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *SIAM Journal on Computing*, 36(5):915–942, 2006.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, May 2004.

[BF03] Dan Boneh and Matthew K. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[Coc01] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer.

[Cor02] Jean-Sébastien Coron. Optimal Security Proofs for PSS and Other Signature Schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer, April / May 2002.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552. ACM Press, May 1991.

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

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, December 2002.

[HL02] Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, April / May 2002.

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

[LM08] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically Efficient Lattice-Based Digital Signatures. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, March 2008.

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

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post Quantum Cryptography*, pages 147–191. Springer, February 2009.

[Pei09a] Chris Peikert. Bonsai Trees: Arboriculture in Lattice-Based Cryptography. In *manuscript*, 2009.

[Pei09b] Chris Peikert. Public Key Cryptosystems from the Worst-Case Shortest Vector Problem. In *STOC*, 2009. To appear.

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

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394. ACM Press, May 1990.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on Pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[Wat05] Brent R. Waters. Efficient Identity-Based Encryption Without Random Oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.