1 Digital Signatures

In this lecture we cover constructions of, attacks on, and security proofs for, *digital signature schemes* based on lattices. We first recall the definition of a signature scheme. Informally, it allows a user to "sign" messages using a secret key, so that anyone can verify the validity of the signatures using a corresponding public key. At the same time, it should be hard for an attacker to produce a valid signature for a message that the signer did not actually sign. We now present the formal syntax of a signature scheme and some of the main kinds of security that are considered.

Definition 1.1. A digital signature scheme consists of three algorithms:

- $\mathsf{Gen}()$ outputs a secret *signing key* sk and a public *verification key* vk.
- $\mathsf{Sign}(sk, \mu)$, given a signing key sk and a message μ, outputs a *signature* σ.
- $\mathsf{Ver}(vk, \mu, \sigma)$, given a verification key vk, a message μ, and a signature σ, either accepts or rejects.

For correctness, we require that for any $(vk, sk) \leftarrow \mathsf{Gen}()$, any message μ, and any signature $\sigma \leftarrow \mathsf{Sign}(sk, \mu)$, the verifier $\mathsf{Ver}(vk, \mu, \sigma)$ always accepts. (We could also relax this slightly, requiring that this holds with high probability over all the random choices.)

For security, we want a signature scheme to be *unforgeable*. There are a variety of ways this can be defined, depending on what kind of power the adversary is allowed to have. Two possibilities are as follows:

1. *Key-only attack*: given a verification key vk where $(sk, vk) \leftarrow \mathsf{Gen}$, it should be infeasible for an adversary to produce any (μ^*, σ^*) such that $\mathsf{Ver}(vk, \mu^*, \sigma^*)$ accepts.
2. *Chosen-message attack*: given a verification key vk as above, along with "black-box" oracle access to $\mathsf{Sign}(sk, \cdot)$—i.e., the ability to get valid signatures for any desired messages—it should be infeasible for the adversary to produce any (μ^*, σ^*) such that $\mathsf{Ver}(vk, \mu^*, \sigma^*)$ accepts and μ^* was not a query to the signing oracle.[1]

Unforgeability under key-only attack is not strong enough for real-world use, because the adversary will typically get to see some of the user's legitimate signatures, which could provide helpful information in attacking the scheme.[2] By contrast, unforgeability under a chosen-message attack is a strong security notion, because it says that the attacker, even after getting valid signatures on any messages of its choice, cannot forge a signature on any new message.

2 GGH Signature Scheme

The first proposal for a lattice-based signature scheme was given by Goldreich, Goldwasser, and Halevi in 1997 [GGH97]. (An efficient real-world instantiation of this idea was later given in [HHGP+03].) The key idea is that the secret signing key is a "short" basis of a lattice $\mathcal{L}$ (i.e., consisting of short vectors) whereas the public key is a "bad" (long) basis of the same lattice. Each message corresponds to a "random" coset of the lattice, via a suitably complex hash function. To sign a message, the signer uses its short basis to find a

[1] Note that the second clause of the forgery condition is necessary, because otherwise the adversary could just query its signing oracle on an arbitrary μ^*, receive a valid signature σ^*, and output (μ^*, σ^*) as its "forgery." To rule out this trivial attack, the definition requires the adversary to output a valid signature for a "new" message, for which it has never received a valid signature.

[2] As an exercise, construct a signature scheme that is unforgeable under key-only attack, but breaks completely once the adversary is given a single valid signature. Hint: the signature on any message can just be the secret signing key.

relatively short element of a corresponding lattice coset, which serves as the signature. To verify a purported signature on a message, one checks that the signature belongs to the appropriate coset and is suitably short.

- $\mathsf{Gen}()$: somehow generate a lattice $\mathcal{L}$ along with both a short basis $\mathbf{S}$ and a "bad" basis $\mathbf{B}$ (or some other public representation). Output $sk = \mathbf{S}$ and $vk = \mathbf{B}$.

  For example, this could be done by generating $\mathbf{S}$ as a set of short random vectors, then defining $\mathbf{B} = \mathbf{SU}$ for some random "bad" unimodular matrix $\mathbf{U}$. In particular, $\mathbf{B}$ could be the Hermite normal form basis of $\mathcal{L}$, which is a "hardest possible" basis of $\mathcal{L}$ [Mic01].

- $\mathsf{Sign}(\mathbf{S}, \mu)$: map μ to a lattice coset $\mathbf{y} + \mathcal{L}$ using a cryptographic hash function H, as $\mathbf{y} = H(\mu)$. Using the short basis $\mathbf{S}$, output a relatively short element $\mathbf{x} \in \mathbf{y} + \mathcal{L}$ as the signature.

  This can be done using the "round-off" algorithm, which lets $\mathbf{x} = \mathbf{S} \cdot \mathrm{frac}(\mathbf{S}^{-1} \cdot \mathbf{y}) \in \mathcal{P}(\mathbf{S})$, where $\mathrm{frac}(\mathbf{r}) = \mathbf{r} - \lfloor \mathbf{r} \rceil$ is the "fractional" (non-integral) part of a vector $\mathbf{r}$. Alternatively, one can use the iterative "nearest-plane" algorithm, which can produce a somewhat shorter vector, in $\mathcal{P}(\widetilde{\mathbf{S}})$.

- $\mathsf{Ver}(\mathbf{B}, \mu, \mathbf{x})$: hash the message as $\mathbf{y} = H(\mu)$. If $\mathbf{x} \in \mathbf{y} + \mathcal{L}$ and $\mathbf{x}$ is "sufficiently short"—e.g., if $\|\mathbf{x}\| \leq \frac{n}{2} \max \|\mathbf{s}_i\|$ (or another suitable public bound)—then accept; otherwise, reject.

Note that, to forge a signature on a particular message, one must find a sufficiently short vector in the corresponding lattice coset. So, if this problem is hard—on the average, for lattices generated according to the key-generation procedure—then the signature scheme is plausibly unforgeable—under a key-only attack, at least.

2.1 Instantiation with SIS Lattices

Let us consider instantiating the GGH scheme with the hard SIS lattices we have covered previously. In this case, the public verification key vk would not be a basis, but instead would be a (near-)uniformly random parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, which defines the lattice $\mathcal{L} = \mathcal{L}^\perp(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = \mathbf{0}\}$. The secret signing key sk would be a short basis $\mathbf{S}$ of $\mathcal{L}$, which must somehow be generated together with $\mathbf{A}$ (see below).

Recall that the integral cosets of $\mathcal{L}^\perp(\mathbf{A})$ correspond to syndromes $\mathbf{y} \in \mathbb{Z}_q^n$, as $\mathcal{L}^\perp_{\mathbf{y}} := \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{y}\}$. So, we map each message to a syndrome via some complex cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{Z}_q^n$. To sign a message having syndrome $\mathbf{y}$, the signer outputs the signature $\mathbf{x} = \mathbf{S} \cdot \mathrm{frac}(\mathbf{S}^{-1} \cdot \mathbf{t})$, where $\mathbf{t} \in \mathcal{L}^\perp_{\mathbf{y}}(\mathbf{A})$ is an arbitrary solution to $\mathbf{At} = \mathbf{y} \in \mathbb{Z}_q^m$. To verify, check that $\mathbf{Ax} = \mathbf{y}$ and that $\mathbf{x}$ is sufficiently short.

It is straightforward to see that if we model H as a truly random function and if SIS is hard, then this instantiation is unforgeable under a *key-only attack*. Specifically, the attacker is given uniformly random verification key $\mathbf{A}$ and the uniformly random hash $\mathbf{y} = H(\mu^*)$ of its target message μ^*, and computing a valid signature for m^* means finding a short $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{y}$, or equivalently, $[\mathbf{A} \mid \mathbf{y}] \cdot \binom{\mathbf{x}}{-1} = \mathbf{0}$. So, forging solves SIS on the uniformly random instance $[\mathbf{A} \mid \mathbf{y}]$.

Key generation for SIS lattices. As mentioned above, the key-generation procedure needs to compute a (nearly) uniform parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a short basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ of the lattice $\mathcal{L}^\perp(\mathbf{A}) \subseteq \mathbb{Z}^m$. It is not at all obvious how to do this![3] If we first choose $\mathbf{A}$ uniformly at random, then it is hard to find *any* short nonzero lattice vectors, much less a short basis. On the other hand, if we first choose $\mathbf{S}$

[3] Our goal here is roughly analogous to generating an RSA modulus $N = pq$ together with its prime factorization (p, q), or a discrete logarithm instance $y = g^x$ together with its solution x. However, in these cases the method is straightforward: e.g., choose two large primes p, q randomly and independently, and let $N = pq$ be their product.

to consist of random short vectors, then it will be highly unlikely to be a basis of a q-ary lattice (much less have determinant q^n), as is required.

We recall from a previous lecture a *partial* solution to this problem, which generates a near-uniform $\mathbf{A}$ together with *one or more* short linearly independent vectors in $\mathcal{L}^\perp(\mathbf{A})$—but not a full basis. The method works as follows: choose uniformly random $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ for some $\bar{m} < m$, then choose uniformly random $\mathbf{X} \in \{0,1\}^{\bar{m} \times (m-\bar{m})}$, and output $\mathbf{A} = [\bar{\mathbf{A}} \mid -\bar{\mathbf{A}}\mathbf{X}] \in \mathbb{Z}_q^{n \times m}$. By the "regularity" lemma (a.k.a. leftover hash lemma), $\mathbf{A}$ is very close to uniform when $\bar{m} \gtrsim n \log q$, and by construction, the columns of $\left(\begin{smallmatrix} \mathbf{X} \\ \mathbf{I} \end{smallmatrix}\right) \in \{0,1\}^{m \times (m-\bar{m})}$ are short linearly independent vectors in $\mathcal{L}^\perp(\mathbf{A})$. However, this yields only $m - \bar{m}$ such vectors, whereas we need m of them to have a full basis. Note that adding more columns to $\mathbf{A}$ increases the number of short vectors we have, but also the number we need by the same amount. In the next lecture we will see a way to "close the loop" and get a full short basis of $\mathcal{L}^\perp(\mathbf{A})$.

2.2 Insecurity of GGH Signatures

As shown by Nguyen and Regev [NR06], when the attacker is given a (small) polynomial number of signatures on arbitrary distinct messages, *any* instantiation of the GGH scheme is *completely broken*; the choice of lattice family and its potential hardness do not matter. In particular, the scheme is broken under a chosen-message attack, or even any "any-message attack." The key reason for this insecurity is that signatures turn out to leak a lot of useful information about the secret key: an attacker can efficiently "learn" the short secret basis (or its equivalent, up to order and sign) from a sequence of signatures.

In a bit more detail, the work of [NR06] exploits the fact that the signing procedure produces a signature by shifting the message hash $\mathbf{y} = H(m)$ into the fundamental parallelepiped $\mathcal{P}(\mathbf{S})$ of the short basis $\mathbf{S}$. Because the hash is uniformly random modulo the lattice, these signatures are essentially uniform over the fundamental parallelepiped.[4] It turns out that, given enough such points, the vectors defining the parallelepiped can be recovered using statistical analysis techniques, known as Independent Component Analysis.

3 GPV Signature Scheme

Work of Gentry, Peikert, and Vaikuntanathan [GPV08] gave a *theoretically sound* design of a GGH-like signature scheme, mainly by adjusting how signatures are produced. The GPV scheme has a rigorous proof of unforgeability under chosen-message attack, when modeling the message hash as a "random oracle," and assuming the hardness of an average-case approximate-SVP problem—in particular, assuming that SIS is hard when the scheme is instantiated with SIS lattices.

Recall that the insecurity of the GGH scheme stems from the fact that signatures are essentially uniform over the fundamental parallelepiped of the secret short basis $\mathbf{S}$, and hence reveal that basis. The key idea behind the GPV scheme is to suitably *randomize* the signing procedure, so that each signature is drawn from a distribution that reveals nothing about the short basis, while still being short and belonging to the appropriate coset. Specifically, the signature is drawn from a *discrete Gaussian* distribution over the coset.

More formally, for a given lattice coset $\mathbf{y} + \mathcal{L}$ (corresponding to the hashed message), the signing procedure uses the short basis $\mathbf{S}$ to sample a signature $\mathbf{x}$ from the discrete Gaussian distribution $D_{\mathbf{y}+\mathcal{L},r}$, for some relatively small parameter r that is related to the length of $\mathbf{S}$, and exceeds the smoothing parameter of $\mathcal{L}$.

[4] With slightly more work, the attack also succeeds against the version of GGH that signs using the nearest-plane algorithm, in which signatures are essentially uniform over the fundamental parallelepiped $\mathcal{P}(\tilde{\mathbf{S}})$ of the Gram-Schmidt vectors of the short basis.

Recall that this distribution is defined as

$$D_{\mathbf{y}+\mathcal{L},r}(\mathbf{x}) := \frac{\rho_r(\mathbf{x})}{\rho_r(\mathbf{y}+\mathcal{L})} = \frac{\rho_r(\mathbf{x})}{\sum_{\mathbf{v}\in\mathbf{y}+\mathcal{L}}\rho_r(\mathbf{v})}$$

for all $\mathbf{x} \in \mathbf{y} + \mathcal{L}$, where $\rho_r(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|^2/r^2)$. By previously covered tail bounds for discrete Gaussians, the sample will be relatively short with high probability.

Observe that the definition of the discrete Gaussian does not depend on the secret basis $\mathbf{S}$ in any way (other than in its width parameter r, which is public). So, intuitively, signatures sampled from this distribution do not reveal anything about the secret key. More precisely, the security proof will show that signatures can be "simulated" *without knowing any short lattice vectors*, when the hash function is suitably modeled.

3.1 Signature Scheme Template

We formalize the above with the following GPV signature scheme template. This template can be instantiated with a variety of lattice families (e.g., SIS lattices), key-generation procedures, discrete Gaussian sampling algorithms, verification checks, and the like.

- $\mathsf{Gen}()$: generate an m-dimensional integer lattice $\mathcal{L}$ together with a basis $\mathbf{S}$ that is "short enough" to enable the signing algorithm to be run for a public quantity $r \geq \eta_\varepsilon(\mathcal{L})$, for some negligible ε.

  Let the secret signing key be $sk = \mathbf{S}$, and the public verification key vk be some "hard" representation of $\mathcal{L}$, e.g., its Hermite normal form basis or parity-check matrix.

- $\mathsf{Sign}(\mathbf{S}, \mu)$: if μ has already been signed, then output the same signature as before.[5] Alternatively, augment μ with sufficiently many random bits (which are then included with the signature), so that with high probability the same augmented message is never signed more than once.

  Map μ (including any augmented randomness) to a lattice coset $\mathbf{y} + \mathcal{L} = H(\mu) \in \mathbb{Z}^m/\mathcal{L}$ using a cryptographic hash function H. Using a suitable algorithm (e.g., the one described in Section 3.3 below) with basis $\mathbf{S}$, sample $\mathbf{x} \leftarrow D_{\mathbf{y}+\mathcal{L},r}$ and output it as the signature.

- $\mathsf{Ver}(vk, \mu, \mathbf{x})$: map μ (including any augmented randomness) to a lattice coset $\mathbf{y} + \mathcal{L} = H(\mu)$. If $\mathbf{x} \in \mathbf{y} + \mathcal{L}$ and $\|\mathbf{x}\| \leq r\sqrt{m}$, then accept; otherwise, reject.[6]

3.2 Security

Here we state and (almost entirely) prove the main security theorem about the GPV signature scheme. As described in Section 2.1 above, the scheme can be instantiated with SIS lattices, where the public key is a near-uniformly random parity-check matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and the hash function H maps messages to syndromes $\mathbf{y} \in \mathbb{Z}_q^n$. A signature is a Gaussian-distributed $\mathbf{x} \in \mathcal{L}_{\mathbf{y}}^{\perp}(\mathbf{A})$, i.e., one for which $\mathbf{Ax} = \mathbf{y}$. The hardness assumption from the theorem is then the SIS assumption, i.e., hardness of finding a short nonzero vector in $\mathcal{L}^{\perp}(\mathbf{A})$ for a uniformly random $\mathbf{A}$. As we have previously seen, this holds as long as certain lattice problems are hard in the worst case.

[5] This can be achieved by locally storing all message-signature pairs, or by applying a pseudorandom function to the message to get "repeatable (pseudo)randomness" for the signing process, thereby making it a deterministic function of the message and the secret key.

[6] Alternatively (or in addition), other norm checks can be enforced on $\mathbf{x}$, e.g., on the ℓ_∞ norm. The check(s) should hold with high probability over the choice of a properly generated $\mathbf{x}$; the stricter they are, the harder forgery will be.

Theorem 3.1. *Suppose that it is infeasible to find a nonzero vector of length at most $2r\sqrt{m}$ in a random m-dimensional lattice $\mathcal{L}$ as generated by* Gen. *Then the above signature scheme is (strongly[7]) unforgeable under chosen-message attack, when the hash function is modeled as a "random oracle" that maps messages (including any added randomness) to the quotient group $\mathbb{Z}^m/\mathcal{L}$.*

The heart of the proof is the following "preimage sampling" lemma, which is also used in many subsequent works. Essentially, it says that sampling a uniformly random coset, then sampling from a discrete Gaussian over that coset, can be efficiently simulated given only (the public representation of) the lattice, and not any secret short vectors. (The lemma's name refers to the fact that sampling from $D_{\mathbf{y}+\mathcal{L},r}$ corresponds to sampling a preimage of a given output of the function that maps a Gaussian-distributed $\mathbf{x}$ to $\mathbf{x} \bmod \mathcal{L}$.)

Lemma 3.2 (Preimage Sampling Lemma). *For any $r \geq \eta_\varepsilon(\mathcal{L})$ where ε is negligible, the output distributions of the following two probabilistic experiments are within negligible statistical distance of each other:*

REAL: *choose uniformly random coset $\mathbf{y} + \mathcal{L} \leftarrow \mathbb{Z}^m/\mathcal{L}$, choose $\mathbf{x} \leftarrow D_{\mathbf{y}+\mathcal{L},r}$, and output $(\mathbf{x}, \mathbf{y} + \mathcal{L})$.*

SIM: *choose $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,r}$, define coset $\mathbf{y} + \mathcal{L} = \mathbf{x} \bmod \mathcal{L}$, and output $(\mathbf{x}, \mathbf{y} + \mathcal{L})$.*

Proof. We prove this by showing two facts: first, that the $\mathbf{y} + \mathcal{L}$ components are statistically close, and second, that conditioned on any fixed value of $\mathbf{y} + \mathcal{L}$ in the second component, the first components are identically distributed.

For the first fact, we need to show that in the SIM experiment, where $\mathbf{x} \leftarrow D_{\mathbb{Z}^m,r}$, the coset $\mathbf{y} + \mathcal{L} = \mathbf{x} \bmod \mathcal{L}$ is nearly uniform over $\mathbb{Z}^m/\mathcal{L}$. For any fixed coset $\bar{\mathbf{y}} + \mathcal{L}$, we are interested in the probability that $\mathbf{x} \in \bar{\mathbf{y}} + \mathcal{L}$, which is

$$\sum_{\mathbf{x} \in \bar{\mathbf{y}}+\mathcal{L}} D_{\mathbb{Z}^m,r}(\mathbf{x}) = \frac{\sum_{\mathbf{x} \in \bar{\mathbf{y}}+\mathcal{L}} \rho_r(\mathbf{x})}{\rho_r(\mathbb{Z}^m)} = \frac{\rho_r(\bar{\mathbf{y}} + \mathcal{L})}{\rho_r(\mathbb{Z}^m)} \approx \frac{\rho_r(\mathcal{L})}{\rho_r(\mathbb{Z}^m)},$$

where the approximation follows from the fact that $r \geq \eta_\varepsilon(\mathcal{L})$ and the smoothing theorem, which says that all cosets of $\mathcal{L}$ have essentially the same mass under ρ_r (up to a multiplicative factor $1 \pm \mathrm{negl}$). So, each coset $\bar{\mathbf{y}} + \mathcal{L}$ is essentially equally likely in the SIM experiment.

For the second step, we need to show that in the SIM experiment, conditioned on the second component being any fixed coset $\bar{\mathbf{y}} + \mathcal{L}$, the conditional distribution of the first component $\mathbf{x}$ is $D_{\bar{\mathbf{y}}+\mathcal{L},r}$. This follows immediately by definition: initially, $\mathbf{x}$ is distributed according to $D_{\mathbb{Z}^m,r}$, i.e., each $\mathbf{x} \in \mathbb{Z}^m$ has probability proportional to $\rho_r(\mathbf{x})$. Then we condition on the event $\mathbf{x} \in \bar{\mathbf{y}} + \mathcal{L}$, so each such $\mathbf{x}$ still has probability proportional to $\rho_r(\mathbf{x})$, but with a new constant of proportionality, namely, $\rho_r(\bar{\mathbf{y}} + \mathcal{L})$. This is exactly the definition of $D_{\bar{\mathbf{y}}+\mathcal{L},r}$, as needed. □

Proof of Theorem 3.1. The key idea is that in the chosen-message attack, by Lemma 3.2, hash values and signatures can jointly be efficiently "simulated," given only (the public representation of) the lattice $\mathcal{L}$. So, having access to the hash and signing oracles does not help the adversary produce a forgery, essentially because it could perform the simulation itself. Moreover, finding a forgery allows one to compute a short nonzero lattice vector, in a way we will see.

The proof proceeds via reduction. Let $\mathcal{F}$ be an adversary that attempts to forge a signature using a chosen-message attack, and succeeds with some probability. We construct a reduction $\mathcal{S}$ that attempts finds

[7] *Strong* unforgeability is a strengthening of ordinary unforgeability that means, even for a message that was queried to the signing oracle (possibly more than once), it is infeasible to find a valid signature that is different from the one(s) returned by the oracle.

a short nonzero vector in a given random lattice $\mathcal{L}$ (as distributed according to Gen), and succeeds with essentially the same probability (up to negligible additive error). The reduction simulates the chosen-message attack game to $\mathcal{F}$, and then uses $\mathcal{F}$'s output forgery to compute a short nonzero lattice vector. A key point is that because $\mathcal{S}$ has no oracles of its own, it must simulate both the hash oracle H and the signing oracle to $\mathcal{F}$. At first glance, it is unclear how $\mathcal{S}$ can implement the signing oracle, because it does not not know a valid signing key for the lattice (i.e., a short basis). Instead, $\mathcal{S}$ will use its ability to "program" the outputs of the hash oracle so that it knows a (properly distributed) signature for any queried message.

Reduction. Formally, the reduction $\mathcal{S}$ is given $\mathcal{L}$ as input, and works as follows. It runs $\mathcal{F}$, giving it $\mathcal{L}$ as the public verification key. The reduction then simulates the hashing and signing oracles as described below. Without loss of generality, we assume that $\mathcal{F}$ never queries the hash oracle H on the same input more than once, because repeated queries would yield the same result. The same goes for the signing oracle, in the version of the system where the signer returns the same signature on repeated queries.[8] Finally, we can assume that $\mathcal{F}$, before querying the signing oracle or outputting a forgery on a particular message, first queries the hash oracle H on that message (because the reduction can induce such a query itself).

- Hash queries: on the ith query μ_i, choose $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^m,r}$ and let $\mathbf{y}_i + \mathcal{L} = \mathbf{x}_i \bmod \mathcal{L}$. Store $(\mu_i, \mathbf{x}_i)$ and return $\mathbf{y}_i + \mathcal{L}$ as the hash output.
- Sign queries: on message μ, find the unique stored $(\mu_i, \mathbf{x}_i)$ for which $\mu_i = \mu$, and return $\mathbf{x}_i$ as the signature.

Finally, $\mathcal{F}$ outputs some potential forgery $(\mu^*, \mathbf{x}^*)$. The reduction $\mathcal{S}$ finds the unique stored $(\mu_i, \mathbf{x}_i)$ for which $\mu_i = \mu^*$, and outputs $\mathbf{v} = \mathbf{x}^* - \mathbf{x}_i$ as a potential short nonzero vector in $\mathcal{L}$.

Analysis. We now analyze the reduction. First observe that by Lemma 3.2, the reduction correctly simulates (up to negligible statistical distance) the hashing and signing oracles of the real chosen-message attack, so the probability that $\mathcal{F}$ outputs a valid forgery is essentially the same in the simulation as it is in the real attack game. This is because for each hash query μ_i (which all are distinct by assumption), the real attack and the simulation respectively correspond to the REAL and SIM experiments of Lemma 3.2: in the real attack, the hash output for μ_i is a fresh uniformly random coset $\mathbf{y}_i + \mathcal{L}$, and the corresponding signature (if it is ever queried) is distributed according to $D_{\mathbf{y}_i+\mathcal{L},r}$, whereas in the simulation, the signature $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^m,r}$ is chosen first, and the hash output for μ_i is defined as $\mathbf{y}_i + \mathcal{L} = \mathbf{x}_i \bmod \mathcal{L}$.[9]

Finally, we claim that if $\mathcal{F}$ outputs a valid forgery $(\mu^*, \mathbf{x}^*)$, the reduction $\mathcal{S}$ outputs a nonzero $\mathbf{v} \in \mathcal{L}$ of norm $\|\mathbf{v}\| \leq 2r\sqrt{m}$, with high probability. First, let $(\mu_i = \mu^*, \mathbf{x}_i)$ be as defined in the reduction, and observe that since both $\mathbf{x}_i, \mathbf{x}^*$ belong to the same coset $\mathbf{y}_i + \mathcal{L} = H(\mu^*) = H(\mu_i)$ and $\|\mathbf{x}_i\|, \|\mathbf{x}^*\| \leq r\sqrt{m}$ (because they are valid signatures for $\mu^* = \mu_i$), we have $\mathbf{v} = \mathbf{x}^* - \mathbf{x}_i \in \mathcal{L}$ and $\|\mathbf{v}\| \leq 2r\sqrt{m}$ by the triangle inequality.

It just remains to show that $\mathbf{v} \neq \mathbf{0}$ with high probability. There are two cases, based on whether $\mathcal{F}$ queried the signing oracle on μ^*, or not. In the former case, according to the definition of strong unforgeability we must have $\mathbf{x}^* \neq \mathbf{x}_i$, so $\mathbf{v} = \mathbf{x}^* - \mathbf{x}_i \neq \mathbf{0}$. In the latter case, we use an information-theoretic argument: because $\mathcal{F}$ did not query the signing oracle on $\mu^* = \mu_i$, the only random variable that depends on $\mathbf{x}_i$ that $\mathcal{F}$ has seen is the hash value $\mathbf{y}_i + \mathcal{L} = \mathbf{x}_i \bmod \mathcal{L} = H(\mu_i)$. Therefore, conditioned on $\mathcal{F}$'s view, $\mathbf{x}_i$ is random

[8] In the version where the signer first augments the message with some randomness, $\mathcal{F}$ may call the signing oracle more than once on the same message, because the results on repeated queries can vary. However, observe that with high probability, the same *augmented* message is not signed more than once, which is all we need.

[9] Notice that here we are using the assumption that no (randomness augmented) message is ever signed more than once: in the simulation, the same signature would be returned each time, so we must ensure that the same is true in the real system.

with discrete Gaussian distribution $D_{\mathbf{y}_i+\mathcal{L},r}$. One can show that this distribution has large "min-entropy," i.e., even the most likely outcome is very unlikely. In particular, for any forged signature $\mathbf{x}^*$, we have that $\mathbf{x}_i \neq \mathbf{x}^*$ with high probability, as needed. □

3.3 Discrete Gaussian Sampling Algorithm

It remains to give an efficient algorithm for discrete Gaussian sampling. Specifically, given an m-dimensional lattice coset $\mathbf{y} + \mathcal{L}$, a width r, and a sufficiently short basis $\mathbf{S}$ of $\mathcal{L}$, the algorithm should sample from (a distribution statistically very close to) the discrete Gaussian $D_{\mathbf{y}+\mathcal{L},r}$. The work of [GPV08] showed that a *randomized* version of the *nearest-plane* algorithm fulfills this goal.[10]

More formally, the algorithm works as follows. (Recall that $\widetilde{\mathbf{S}}$ denotes the Gram-Schmidt orthogonalization of $\mathbf{S}$, which can be precomputed before the input coset is known.) For $i = m$ down to 1:

1. Let $c_i = \langle \mathbf{y}, \widetilde{\mathbf{s}}_i \rangle / \langle \widetilde{\mathbf{s}}_i, \widetilde{\mathbf{s}}_i \rangle$.
2. Let $z_i \leftarrow c_i + D_{\mathbb{Z}-c_i, r/\|\widetilde{\mathbf{s}}_i\|}$.

   (This is the only difference with the nearest-plane algorithm: instead of letting $z_i = \lfloor c_i \rceil$, we let it be a random integer from a discrete Gaussian "centered" at c_i, of appropriate width.)
3. Let $\mathbf{y} \leftarrow \mathbf{y} - z_i \mathbf{s}_i$.

Finally, output (the most recent value of) $\mathbf{y}$.

Theorem 3.3. *If $r \geq (\max_i \|\widetilde{\mathbf{s}}_i\|) \cdot \sqrt{\ln(O(m/\varepsilon))} \geq \eta_\varepsilon(\mathcal{L}(\mathbf{S}))$ for some positive $\varepsilon < 1/2$, then the above algorithm samples from $D_{\mathbf{y}+\mathcal{L},r}$, to within ε statistical distance.*

Proof. By construction, the algorithm returns a vector in the coset $\mathbf{y} + \mathcal{L}$, because it changes $\mathbf{y}$ only by subtracting lattice vectors from it. Now let $\mathbf{x} \in \mathbf{y} + \mathcal{L}$ be arbitrary; to establish the claim, we show that the probability that the algorithm outputs $\mathbf{x}$ is proportional to $\rho_r(\mathbf{x})$, up to a factor within $1 \pm \varepsilon$. Trivially, $\mathbf{x}$ is output if and only if the unique sequence of z_i yielding $\mathbf{x}$ is chosen; these satisfy $\mathbf{x} = \sum_{i=1}^m (c_i - z_i)\widetilde{\mathbf{s}}_i$, where the c_i are as defined in the algorithm (and implicitly depend on the chosen values of z_j for $j > i$). The probability that this sequence of z_i is chosen is:

$$\begin{aligned}
\prod_i D_{\mathbb{Z}-c_i, r/\|\widetilde{\mathbf{s}}_i\|}(z_i - c_i) &= \frac{\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(z_i - c_i)}{\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(c_i + \mathbb{Z})} \\
&= \frac{\prod_i \rho_r((z_i - c_i)\|\widetilde{\mathbf{s}}_i\|)}{\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(c_i + \mathbb{Z})} \\
&= \frac{\rho_r(\mathbf{x})}{\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(c_i + \mathbb{Z})} \\
&\approx \frac{\rho_r(\mathbf{x})}{\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(\mathbb{Z})}.
\end{aligned}$$

The final equality follows from the fact that $\|\mathbf{x}\|^2 = \sum_{i=1}^m (z_i - c_i)^2 \|\widetilde{\mathbf{s}}_i\|^2$ by orthogonality of the $\widetilde{\mathbf{s}}_i$, and by definition of ρ_r (over both $\mathbb{R}^m$ and $\mathbb{R}$). The final approximation follows from our assumption that $r/\|\widetilde{\mathbf{s}}_i\| \geq \sqrt{\ln(O(m/\varepsilon))} \geq \eta_\varepsilon(\mathbb{Z})$, which implies that $\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(c_i + \mathbb{Z})$ is sufficiently close to $\prod_i \rho_{r/\|\widetilde{\mathbf{s}}_i\|}(\mathbb{Z})$. Since the denominator in the final expression is just a fixed constant of proportionality, the proof is complete. □

[10] A similarly randomized nearest-plane algorithm was previously considered in [Kle00], but for the very different purpose of solving BDD, and for complementary parameters in relation to the short basis vectors.

References

[GGH97] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *CRYPTO*, pages 112–131. 1997. Page 1.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008. Pages 3 and 7.

[HHGP+03] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140. 2003. Page 1.

[Kle00] P. N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA*, pages 937–941. 2000. Page 7.

[Mic01] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *CaLC*, pages 126–145. 2001. Page 2.

[NR06] P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009. Preliminary version in Eurocrypt 2006. Page 3.