Unfortunately, the operating characteristics of standard lattice reduction algorithms such as BKZ-LLL are not nearly as well understood as are the operating characteristics of sieves, the index calculus, or Pollard's ρ method. This makes it difficult to predict theoretically how well a lattice reduction algorithm will perform on any given class of lattices. Thus in practice, the security of a lattice-based cryptosystem such as NTRUEncrypt must be determined experimentally.

Roughly, one takes a sequence of parameters (N, q, d) in which N grows and such that certain ratios involving N, q, and d are held approximately constant. For each set of parameters, one runs many experiments using BKZ-LLL with increasing block size β until the algorithm finds a short vector in $L_{\boldsymbol{h}}^{\text{NTRU}}$. Then one plots the logarithm of the average running time against N, verifies that the points approximately lie on line, and computes the best-fitting line

$$\log(\text{Running Time}) = AN + B. \tag{7.49}$$

After doing this for many values of N up to the point at which the computations become infeasible, one can use the line (7.49) to extrapolate the expected amount of time it would take to find a private key vector in an NTRU lattice $L_{\boldsymbol{h}}^{\text{NTRU}}$ for larger values of N. Such experiments suggest that values of N in the range from 250 to 1000 yield security levels comparable to currently secure implementations of RSA, Elgamal, and ECC. Details of such experiments are described in [102].

Remark 7.62. Proposition 7.61 says that the short target vectors in an NTRU lattice are $\mathcal{O}(\sqrt{N}\,)$ shorter than predicted by the Gaussian heuristic. Theoretically and experimentally, it is true that if a lattice of dimension n has a vector that is extremely small, say $\mathcal{O}(2^n)$ shorter than the Gaussian prediction, then lattice reduction algorithms such as LLL and its variants are very good at finding the tiny vector. It is a natural and extremely interesting question to ask whether vectors that are only $\mathcal{O}(n^\epsilon)$ shorter than the Gaussian prediction might similarly be easier to find. At this time, no one knows the answer to this question.

7.12 Lattice-Based Digital Signature Schemes

We have already seen digital signatures schemes whose security depends on the integer factorization problem (Sect. 4.2) and on the discrete logarithm problem in the multiplicative group (Sect. 4.3) or in an elliptic curve (Sect. 6.4.3). In this section we briefly discuss how digital signature schemes may be constructed from hard lattice problems.

7.12.1 The GGH Digital Signature Scheme

It is easy to convert the CVP idea underlying GGH encryption into a lattice-based digital signature scheme. Samantha knows a good (i.e., short and reasonably orthogonal) private basis $\mathcal{B}$ for a lattice L, so she can use Babai's algorithm (Theorem 7.34) to solve, at least approximately, the closest vector problem in L for a given vector $\boldsymbol{d} \in \mathbb{R}^n$. She expresses her solution $\boldsymbol{s} \in L$ in terms of a bad public basis $\mathcal{B}'$. The vector $\boldsymbol{s}$ is Samantha's signature on the document $\boldsymbol{d}$. Victor can easily check that $\boldsymbol{s}$ is in L and is close to $\boldsymbol{d}$. The GGH digital signature scheme is summarized in Table 7.5.

Samantha	Victor
Key creation	
Choose a good basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ and a bad basis $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ for L. Publish the public key $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$.	
Signing	
Choose document $\boldsymbol{d} \in \mathbb{Z}^n$ to sign. Use Babai's algorithm with the good basis to compute a vector $\boldsymbol{s} \in L$ that is close to $\boldsymbol{d}$. Write $\boldsymbol{s} = a_1\boldsymbol{w}_1 + \cdots + a_n\boldsymbol{w}_n$. Publish the signature $(a_1, \ldots, a_n)$.	
Verification	
	Compute $\boldsymbol{s} = a_1\boldsymbol{w}_1 + \cdots + a_n\boldsymbol{w}_n$. Verify that $\boldsymbol{s}$ is sufficiently close to $\boldsymbol{d}$.

Table 7.5: The GGH digital signature scheme

Notice the tight fit between the digital signature and the underlying hard problem. The signature $\boldsymbol{s} \in L$ is a solution to apprCVP for the vector $\boldsymbol{d} \in \mathbb{R}^n$, so signing a document is equivalent to solving apprCVP.

Remark 7.63. In a lattice-based digital signature scheme, the digital document to be signed is a vector in $\mathbb{R}^n$. Just as with other signature schemes, in practice Samantha applies a hash function to her actual document in order to create a short document of just a few hundred bits, which is then signed. (See Remark 4.2.) For lattice-based signatures, one uses a hash function whose output is a vector in $\mathbb{Z}^n$ having coordinates in some specified range.

Example 7.64. We illustrate the GGH digital signature scheme using the lattice and the good and bad bases from Example 7.36 on page 410. Samantha decides to sign the document

$$\boldsymbol{d} = (678846, 651685, 160467) \in \mathbb{Z}^3.$$

She uses Babai's algorithm to find a vector

$$\boldsymbol{s} = 2213\boldsymbol{v}_1 + 7028\boldsymbol{v}_2 - 6231\boldsymbol{v}_3 = (678835, 651671, 160437) \in L$$

that is quite close to $\boldsymbol{d}$,

$$\|\boldsymbol{s}-\boldsymbol{d}\| \approx 34.89.$$

Samantha next uses linear algebra to express $\boldsymbol{s}$ in terms of the bad basis,

$$\boldsymbol{s} = 1531010\boldsymbol{w}_1 - 553385\boldsymbol{w}_2 - 878508\boldsymbol{w}_3,$$

where $\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3$ are the vectors on page 410. She publishes

$$(1531010, -553385, -878508)$$

as her signature for the document $\boldsymbol{d}$. Victor verifies the signature by using the public basis to compute

$$\boldsymbol{s} = 1531010\boldsymbol{w}_1 - 553385\boldsymbol{w}_2 - 878508\boldsymbol{w}_3 = (678835, 651671, 160437),$$

which is automatically a vector in L, and then verifying that $\|\boldsymbol{s}-\boldsymbol{d}\| \approx 34.89$ is small.

We observe that if Eve attempts to sign $\boldsymbol{d}$ using Babai's algorithm with the bad basis $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3\}$, then the signature that she obtains is

$$\boldsymbol{s}' = (2773584, 1595134, -131844) \in L.$$

This vector is not a good solution to apprCVP, since $\|\boldsymbol{s}'-\boldsymbol{d}\| > 10^6$.

Remark 7.65 (Key Size Issues). The GGH signature scheme suffers the same drawback as the GGH cryptosystem, namely security requires lattices of high dimension, which in turn lead to very large public verification keys; cf. Sect. 7.7. It is thus tempting to use an NTRU lattice L^{NTRU} as the public key, but there is an initial difficulty because L^{NTRU} has dimension $2N$, so the known (secret) short vector $(\boldsymbol{f}, \boldsymbol{g})$ and its rotations $(x^i \star \boldsymbol{f}, x^i \star \boldsymbol{g})$ for $0 \le i < N$ give only half a very short basis for L^{NTRU}. Using a technique described in [55], it is possible to extend the half-basis to a full basis that is short enough to make an NTRU signature scheme feasible. However, both GGH and NTRU signature schemes have a more serious shortcoming which we now describe.

7.12.2 Transcript Analysis

In any digital signature scheme, each document/signature pair $(\boldsymbol{d}, \boldsymbol{s})$ reveals some information about the private signing key $\boldsymbol{v}$, since at the very least, it reveals that the document $\boldsymbol{d}$ signed with the private key $\boldsymbol{v}$ yields the signature $\boldsymbol{s}$. Hence a sufficiently long *transcript* of signed documents

$$(\boldsymbol{d}_1, \boldsymbol{s}_1),\ (\boldsymbol{d}_2, \boldsymbol{s}_2),\ (\boldsymbol{d}_3, \boldsymbol{s}_3),\ \ldots, (\boldsymbol{d}_r, \boldsymbol{s}_r) \tag{7.50}$$

may reveal information about either the signing key or how to sign additional documents.