Cryptosystems Based on Lattices

We present some cryptosystems whose common feature is that they all rely on computational problems in lattices for their security. The subject of lattice based cryptography is very active and there have recently been new ideas that revolutionised the field. It is beyond the scope of this book to survey these recent developments.

19.9 The Goldreich-Goldwasser-Halevi Cryptosystem and Variants

The Goldreich-Goldwasser-Halevi (GGH) cryptosystem relies on the difficulty of the closest vector problem (CVP) in a lattice. The system is reminiscent of the McEliece cryptosystem, which we briefly recall in the next paragraph. Encryption for both systems is randomised.

In the McEliece cryptosystem one chooses an error correcting code (some references for error correcting codes are van Lint [391] and Chapter 18 of [609]) over a finite field $\mathbb{F}_q$ (typically $\mathbb{F}_2$) given by a $k \times n$ generator matrix G (where $k < n$) and publishes a "disguised" version $G' = SGP$ where S and P are suitable invertible matrices (we refer to Section 8.5 of Menezes, van Oorschot and Vanstone [418] for details). The public key is G' and the private key is (S, G, P). To encrypt a message $\underline{m} \in \mathbb{F}_q^k$ one computes $\underline{c} = \underline{m}G' + \underline{e}$ where $\underline{e} \in \mathbb{F}_q^n$ is a randomly chosen error vector of low Hamming weight; note that this computation is over $\mathbb{F}_q$. To decrypt one uses the decoding algorithm for the error correcting code.

The basic GGH public key encryption scheme is similar; we give an informal sketch of the idea now. One chooses a "nice" basis B for a full rank lattice $L \subset \mathbb{Z}^n$ and publishes a "disguised" basis $B' = UB$ for L where U is "random" unimodular matrix. A message $\underline{m} \in \mathbb{Z}^n$ is encrypted as $\underline{c} = \underline{m}B' + \underline{e}$ where $\underline{e}$ is a randomly chosen short error vector; note that this computation is over $\mathbb{Z}$. To decrypt one solves the closest vector problem, using the nice basis B, to obtain the lattice point $\underline{m}B'$ close to $\underline{c}$; one can then obtain $\underline{m}$.

While encryption is superficially the same for the McEliece and GGH cryptosystems, there are significant differences between the security analysis of these schemes. An advantage of the lattice approach is that the error vector is required to have less structure: it is only required to be short, compared with McEliece where the error vector must have low Hamming weight. Both schemes have ciphertexts larger than the messages but an advantage of McEliece is that ciphertexts have a fixed size whereas for GGH the coefficients are integers whose size can vary significantly.

Exercise 19.9.1. Show that any cryptosystem based on the McEliece or GGH idea does not have indistinguishability security under passive attack.

Exercise 19.9.2. A variant of the McEliece or GGH proposal is to swap the roles of the message and the randomness. In other words one encodes the message as a valid error vector $\underline{m}$, chooses a random $\underline{e} \in \mathbb{F}_q^k$ (respectively, $\underline{e} \in \mathbb{Z}^k$) and computes $\underline{c} = \underline{e}G' + \underline{m}$ (resp., $c = \underline{e}B' + \underline{m}$). Show that this variant also does not have indistinguishability security under passive attacks.

Exercise 19.9.3. Show that any cryptosystem based on the McEliece or GGH idea (and without any padding scheme) does not have one way encryption security under a CCA attack.

Exercises 19.9.1 and 19.9.3 show that the 'textbook' (i.e., without a padding scheme) McEliece and GGH cryptosystems should not be used in practice. Using techniques similar to those presented later for Elgamal and RSA one can prevent such attacks as long as the basic scheme is OWE-CPA secure (see Section 1.3.1 for the definition of this security notion). Hence, for the rest of this chapter we focus purely on the textbook versions and mainly consider security under passive attacks.

Exercise 19.9.4. Given a GGH public key B' show that there is more than one private basis matrix B that is suitable for decryption. Show that one can efficiently determine whether a guess B for a GGH private basis matrix can correspond to a given public basis B'.

To give a precise definition of the GGH cryptosystem it is necessary to give the following details:

1. What dimension n should be used?
2. How does one choose the "nice" lattice basis B and what properties should it have?
3. How does one choose the "random" unimodular matrix U?
4. What is the message space and how does one encode information into a vector $\underline{m}$?
5. How does one choose the error vector $\underline{e}$ and in what space does it lie?

We briefly sketch the proposal of Goldreich, Goldwasser and Halevi; the reader should refer to the original paper [256] for complete details. It is suggested to take $n \geq 200$. Two methods to generate the "nice" basis B are given: one is to choose a random matrix B with entries in, say, $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ (so that all vectors are relatively short); another is to choose $B = kI_n + E$ where I_n is the $n \times n$ identity matrix, $k > 1$ is a "medium sized" integer and E is a random matrix with small entries as in the first case above. Two methods to generate U are also given; in both cases the issue is to ensure that the coefficients of $B' = UB$ do not explode in size (we refer to Section 3.2 of [256] for details). The message space is the set of vectors of length n with entries in

$\{-M, -(M-1), \ldots, -1, 0, 1, \ldots, M-1, M\}$ for some $M \in \mathbb{N}$ ([256] actually suggests $M = n$). Finally, the error vector is chosen to be a random vector of length n with entries in $\{-\sigma, \sigma\}$ for some $\sigma \in \mathbb{N}$ (typically, $\sigma = 3$).

As mentioned above, to encrypt a message $\underline{m}$ one computes the ciphertext $\underline{c} = \underline{m}B' + \underline{e}$. To decrypt $\underline{c}$ one uses the Babai rounding technique with respect to the nice basis B for the lattice. More precisely, multiply $\underline{c}$ by B^{-1} to obtain

$$\underline{c}B^{-1} = (\underline{m}B' + \underline{e})B^{-1} = \underline{m}UBB^{-1} + \underline{e}B^{-1} = \underline{m}U + \underline{e}B^{-1} \in \mathbb{Q}^n.$$

The Babai rounding will remove the term $\underline{e}B^{-1}$ as long as it is small enough. One then multiplies by U^{-1} to get the message $\underline{m}$.

Exercise 19.9.5. Write down algorithms for KeyGen, Encrypt and Decrypt.

Example 19.9.6. Let $L \subset \mathbb{R}^2$ be the lattice with basis matrix

$$B = \begin{pmatrix} 17 & 0 \\ 0 & 19 \end{pmatrix}.$$

Let

$$U = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix} \quad \text{giving} \quad B' = UB = \begin{pmatrix} 34 & 57 \\ 51 & 95 \end{pmatrix}.$$

Let the message be $\underline{m} = (2, -5)$ and take $\underline{e} = (1, -1)$ (this is GGH encryption with $\sigma = 1$). Then

$$\underline{c} = \underline{m}B' + \underline{e} = (-186, -362).$$

To decrypt one computes

$$\underline{c}B^{-1} \approx (-10.94, -19.05)$$

(note that $\underline{m}U = (-11, -19)$ and $\underline{e}B^{-1} \approx (0.06, -0.05)$). We round the above to $(-11, -19)$ and recover the message as $\underline{m} = (-11, -19)U^{-1} = (2, -5)$.

Exercise 19.9.7. Show that GGH decryption gives the correct result as long as the entries of $\underline{e}B^{-1}$ are real numbers of absolute value $< 1/2$. Let ρ be the maximum, in the ℓ_1-norm, of the columns of B^{-1}. Show that if $\sigma < 1/(2\rho)$ then decryption gives the correct result.

Exercise 19.9.8. For the public key in Example 19.9.6 decrypt the ciphertext $\underline{c} = (220, 400)$.

As mentioned, the ciphertext in GGH encryption is considerably larger than the message. A precise analysis of this depends on the sizes of entries in B' (which in turns depends on the specific choices for B and U). We do not give any estimates for the ciphertext expansion.

Micciancio [421] proposed a variant of the GGH cryptosystem. The first idea is, instead of choosing the public basis to be $B' = UB$ for a random matrix $U \in \mathrm{SL}_2(\mathbb{Z})$, one can choose B' to be the Hermite normal form (HNF) of B. There is no loss of security by doing this, since anyone can compute the HNF of UB, and get the same result. The second idea is to encode the message in the error vector rather than in the lattice point (this is the same idea as discussed in Exercise 19.9.2) and to reduce it to the orthogonalized parallelepiped (see Exercise 19.9.9). This results in significantly shorter ciphertexts than the original GGH system and makes the encryption process deterministic. We refer to [421] for further details.

Exercise 19.9.9. Let $\underline{b}_1, \ldots, \underline{b}_n$ be an ordered basis for a lattice L and let $\underline{b}_1^*, \ldots, \underline{b}_n^*$ be the corresponding Gram-Schmidt vectors. Define the **orthogonalized parallelepiped**

$$\mathcal{P} = \left\{ \sum_{i=1}^{n} x_i \underline{b}_i^* : 0 \leq x_i \leq 1 \right\}.$$

Given $\underline{v} \in \mathbb{R}^n$ show how to compute $\underline{w} \in \mathcal{P}$ such that $\underline{v} - \underline{w} \in L$. This is called reducing to the orthogonalized parallelepiped.

19.10 Cryptanalysis of GGH Encryption

We now discuss the one-way encryption (OWE) security of the GGH cryptosystem under passive attacks. There are three natural ways to attack the GGH cryptosystem:

1. Try to obtain the private key B from the public key B'.
2. Try to obtain information about the message from the ciphertext, given that the error vector is small.
3. Try to solve the CVP of $\underline{c}$ with respect to the lattice L defined by B'.

We also present a fourth attack, due to Nguyen, which exploits the particular format of the error vectors in the GGH cryptosystem. Lattice basis reduction algorithms have a role to play in the first and third of these attacks.

Computing a Private Key

For the first attack, we simply run a lattice basis reduction algorithm (such as LLL) on the public basis matrix B'. If we are lucky then it will output a basis B'' that is good enough to allow the efficient solution of the required closest vector instances.

Example 19.10.1. Let

$$B = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 23 & 0 \\ 0 & 0 & 99 \end{pmatrix}$$

and define $B' = UB$ where

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 8 & 1 & 0 \\ -11 & 5 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 3 & -10 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix}.$$

Then B' is the public basis matrix

$$B' = \begin{pmatrix} 7 & 69 & -990 \\ 56 & 575 & -8514 \\ -77 & -644 & 8019 \end{pmatrix}.$$

Let $\underline{m} = (2, -1, 3)$ be a message and $\underline{e} = (-1, 1, 1)$ an error vector and define $\underline{c} = \underline{m}B' + \underline{e} = (-274, -2368, 30592)$. Running LLL on B' does yield (up to sign) the matrix B (and hence $U = B'B^{-1}$). From this one can recover $\underline{m}$.

To prevent such an attack it is necessary that the dimension of the lattice be sufficiently large.

Computing Information about the Message

For the second attack we exploit the fact that $\underline{c} = \underline{m}B' + \underline{e}$ where $\underline{e}$ is a vector with small entries. A naive attack is to try all values of the error vector $\underline{e}$ until $\underline{c} - \underline{e}$ lies in the image of $\mathbb{Z}^n B'$. A more subtle idea is to compute $\underline{c}(B')^{-1} = \underline{m} + \underline{e}(B')^{-1}$ and try to deduce possible values for some entries of $\underline{e}(B')^{-1}$. For example, if the j-th column of $(B')^{-1}$ has particularly small norm then one can deduce that the j-th entry of $\underline{e}(B')^{-1}$ is always small and hence get an accurate estimate for the j-th entry of $\underline{m}$. We refer to Section 4.2 of [256] for further discussion. To defeat this attack one should not naively encode the message as a vector $\underline{m} \in \mathbb{Z}^n$. Instead, one should only use some low-order bits of some entries of $\underline{m}$ to carry information, or use an appropriate randomised padding scheme.

Solving the CVP Directly

For the third attack one can consider any of the algorithms listed in Chapter 18 for solving the CVP. For example, one can use the Babai nearest plane algorithm or the embedding technique.

Example 19.10.2. We use the public key and ciphertext from Example 19.10.1 and recover the message using the embedding technique. Construct

$$A = \begin{pmatrix} 7 & 69 & -990 & 0 \\ 56 & 575 & -8514 & 0 \\ -77 & -644 & 8019 & 0 \\ -274 & -2368 & 30592 & 1 \end{pmatrix}.$$

Running LLL on A yields the matrix

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 5 & 2 & 2 & 2 \\ -1 & -15 & 8 & 8 \\ -1 & -2 & 51 & -48 \end{pmatrix}.$$

As desired, the first row is $(-1, 1, 1, 1) = (\underline{e}, 1)$. From this one can compute the message as $\underline{m} = (\underline{c} - \underline{e})(B')^{-1}$.

Exercise 19.10.3. For the public key from Example 19.10.1 use the embedding technique to decrypt the ciphertexts $\underline{c} = (120, 1220, -18017)$ and $\underline{c} = (-83, -714, 9010)$.

To defeat such attacks it is necessary that the lattice dimension is sufficiently large and that the solution to the CVP instance is not too special. In particular, the error vector should not be too short compared with the vectors the lattice.

Nguyen's Attack

Nguyen noted that the choice of the error vector in the original GGH cryptosystem made it extremely vulnerable to attack. Write $\underline{\sigma} = (\sigma, \sigma, \ldots, \sigma) \in \mathbb{Z}^n$. The crucial observation is that if $\underline{c}$ is a GGH ciphertext then $\underline{c} + \underline{\sigma} \equiv \underline{m}B' \pmod{2\sigma}$. If B' is invertible modulo 2σ (or even modulo a factor of 2σ) then one can already extract significant information about the message $\underline{m}$. Furthermore, if one successfully computes $\underline{m}_0 \equiv \underline{m} \pmod{2\sigma}$, then one obtains the simpler closest vector instance

$$\frac{\underline{c} - \underline{m}_0 B'}{2\sigma} = \underline{m}' B' + \frac{\underline{e}}{2\sigma}$$

where $\underline{m} = \underline{m}_0 + 2\sigma\underline{m}'$. Since $\underline{e}/(2\sigma)$ is a much shorter vector than $\underline{e}$ it is possible that algorithms for the closest vector problem that were not successful on the original instance can succeed on the new instance.

Example 19.10.4. Consider the lattice L and ciphertext c from Example 19.9.6. Since $\sigma = 1$ we can add $(1, 1)$ to c and solve

$$c + (1, 1) = (-185, -361) \equiv \underline{m}_0 B' \pmod 2$$

(note that B' is invertible over $\mathbb{F}_2$). One finds $\underline{m}_0 = (0, 1) \equiv (2, -5) \pmod 2$ as expected.

Exercise 19.10.5. Perform Nguyen's attack for the ciphertexts of Exercise 19.10.3.

The natural approach to resist Nguyen's attack is to choose error vectors with a more general range of entries (e.g., $e_j \in \{-\sigma, -(\sigma-1), \ldots, -1, 0, 1, \ldots, \sigma\}$ for $1 \leq j \leq n$). It is then necessary to re-evaluate all the other attacks and parameter choices.

Finally, we remark that none of the above techniques gives an attack with polynomial asymptotic complexity as the dimension n grows. Hence, the GGH encryption scheme and its variants are not broken. On the other hand, in practice one needs to use lattices of rather large dimension and this limits the practicality of the GGH system.

19.11 GGH Signatures

Let B' be a GGH public key corresponding to a lattice L in $\mathbb{Z}^n$. The natural signature scheme is as follows: Given a message m hash it to a "random" element $H(m) \in \mathbb{Z}^n$. Then, using the private key, compute a lattice vector $\underline{s}$ close to $H(m)$. The signature on message m is then $\underline{s}$. To verify the signature one checks that $\underline{s}$ lies in the lattice (i.e., $\underline{s}(B')^{-1} \in \mathbb{Z}^n$) and that $\|\underline{s} - H(m)\|$ is smaller than some threshold that is specified as part of the signature verification key.

We remark that signatures for lattice schemes are somewhat easier than for the McEliece cryptosystem since CVP algorithms work for any point in $\mathbb{R}^n$ whereas decoding algorithms may fail for words that are not within the minimum distance of a code-word (however, see Courtois, Finiasz and Sendrier [451] for a study of McEliece signatures).

To analyse the security (namely, resistance to forgery) of such a signature scheme one must consider all the attacks mentioned above on the encryption scheme. One therefore is required to use lattices of large dimension $n \geq 200$.

Furthermore, as usual with signatures, one must also consider the fact that an adversary could obtain signatures on messages and that this might leak information about the private key. For the GGH signature scheme one sees that $\underline{s} - H(m)$ is a short vector in $\mathbb{R}^n$. Indeed, if the CVP algorithm used by the signer is perfect for the basis B then $\underline{s} - H(m)$ always lies in the parallelepiped

$$\mathcal{P}_{1/2}(B) = \{\underline{x}B : \underline{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n, -1/2 \leq x_i \leq 1/2 \text{ for all } 1 \leq i \leq n\},$$

which is called a **fundamental domain** for the lattice (i.e., for every point $\underline{x} \in \mathbb{R}^n$ there is some $\underline{y}$ in the lattice such that $\underline{x} - \underline{y} \in \mathcal{P}_{1/2}(B)$). The fundamental domain of a lattice is a simplex whose sides are determined by the basis vectors in B. Hence, it is natural to wonder whether seeing a number of random entries in $\mathcal{P}_{1/2}(B)$ allows one to learn something about the vectors in B. Nguyen and Regev [457, 458] have explored this idea and shown that such an approach can be used to cryptanalyse signatures. Adding a "perturbation" to the signature seems to prevent the attack of Nguyen and Regev (see Section 1.3 of [458]). Gentry, Peikert and Vaikuntanathan [253] give a method to sample

from a lattice (when given a sufficiently good basis) such that the output is statistically close to a Gaussian distribution. Hence, their paper gives[3] a secure implementation of the GGH signature concept.

19.12 NTRU

The NTRU[4] cryptosystem was invented by Hoffstein, Pipher and Silverman. The original proposal is phrased in terms of polynomial rings. We refer to Hoffstein, Pipher and Silverman [288], Section 6.10 of [289] or Section 17.4 of [609] for a description of the system in these terms.

The NTRU encryption scheme can also be described as a special case of Micciancio's variant of the GGH encryption scheme. The public key is a $2n \times 2n$ matrix

$$B = \begin{pmatrix} qI_n & 0 \\ H & I_n \end{pmatrix}$$

in Hermite normal form, where I_n is the $n \times n$ identity matrix, q is an integer (typically $q = 2^8$ or 2^{10}) and H is an $n \times n$ matrix with entries in $\{0, 1, \ldots, q-1\}$. The crucial property of NTRU is that the matrix H is a **circulant matrix**, in other words, the rows of H are just cyclic rotations of the first row of H. This means that to specify the NTRU public key one only needs to specify q and the first row of H; the public key requires $O(n \log_2(q))$ bits.

The matrix H is constructed by the user in a special way so that they know a basis for the lattice generated by B consisting of short vectors. Encryption proceeds as in the Micciancio scheme. We refer to Section 5.2 of Micciancio and Regev [423] for further details.

The details of the NTRU scheme have evolved over time. In particular, earlier parameter choices for NTRU had a noticeable probability of decryption failures, and this property was used to develop active (i.e., not passive) attacks [298]. Hence, the currently recommended parameters for NTRU have negligible probability of decryption failures.

The security of the NTRU cryptosystem relies on the difficulty of computing short vectors in the NTRU lattice. One remark is that the NTRU lattice has a number of special properties that can be used to improve the standard algorithms for finding short vectors. In particular, if $\underline{v}$ is a short vector in the NTRU lattice then so are the n "cyclic rotations" of $\underline{v}$. As a sample of the literature on special properties of the NTRU lattice we refer to May and Silverman [412], Gama, Howgrave-Graham and Nguyen [234] and Gentry [251].

19.13 Knapsack Cryptosystems

Knapsack cryptosystems were proposed by Merkle and Hellman in 1978. As with NTRU, the original description of knapsack cryptosystems made no reference to lattices. However there is a general attack on knapsacks using lattices (indeed, this was the first application of lattice basis reduction to public key cryptanalysis) and so it is natural to consider them as a lattice-based cryptosystem. Though not used in practice, we briefly present knapsack cryptosystems as they are an excellent source of exercises in cryptanalysis.

[3] This is only one of the many contributions of [253].

[4] The meaning of the acronym NTRU is not explained in the original paper. One interpretation is that it stands for "Number Theorists are Us". After various successful attacks were discovered on the corresponding signature scheme some individuals in the cryptography community started to refer to it as "Not True".

Definition 19.13.1. Let $b_1, \ldots, b_n$ be distinct positive (i.e., $b_i \geq 1$) integers (sometimes called **weights**). The **subset sum problem** is, given an integer s obtained as a sum of elements b_i, to find $x_i \in \{0, 1\}$ for $i = 1, \ldots, n$ such that

$$s = \sum_{i=1}^{n} x_i b_i.$$

The name **knapsack** is a mis-use of subset sum. It comes from the idea of finding out what is in a knapsack (a type of bag) just from its weight. The subset sum problem is $\mathcal{NP}$-complete.

Exercise 19.13.2. A decisional variant of Definition 19.13.1 is, given $\{b_1, \ldots, b_n\}$ and $s \in \mathbb{N}$ to decide whether or not there are $x_i \in \{0, 1\}$ such that $s = \sum_{i=1}^{n} x_i b_i$. Prove that these two computational problems are equivalent.

Exercise 19.13.3. Let notation be as in Definition 19.13.1 and let $B = \sum_{i=1}^{n} b_i$. Give a time-memory tradeoff algorithm to find the solution $x_i \in \{0, 1\}$, or show none exists, in $O(n2^{n/2} \log(B)^2)$ bit operations and with $O(n2^{n/2} \log(B))$ bits of storage.

The attack of Exercise 19.13.3 has been greatly improved by Shamir and Schroeppel (we do not have space for the details; see Section 8.1.2 of Joux [317]). A further improvement has been given by Howgrave-Graham and Joux [300]. Wagner's algorithm (see Section 13.8) does not seem to be directly applicable to the subset sum problem, though has been used to solve the modular subset sum problem (i.e., given $\{b_i\}$, s and m to find $x_i \in \{0, 1\}$ such that $\sum_{i=1}^{n} x_i b_i \equiv s \pmod{m}$) by Wagner (also see the work of Lyubashevsky and Shallue).

Exercise 19.13.4. Show that every subset sum instance can be reduced to an instance where the weights satisfy $\gcd(b_1, \ldots, b_n) = 1$.

The motivating idea of a knapsack cryptosystem is that computing $s = \sum_{i=1}^{n} x_i b_i$ is a one-way function. The remaining problem is to design subset sum instances that can be efficiently solved using a private key. To do this one first considers easy instances of the subset sum problem.

Definition 19.13.5. A sequence $b_1, \ldots, b_n$ in $\mathbb{N}$ is **superincreasing** if, for each $2 \leq i \leq n$

$$b_i > \sum_{j=1}^{i-1} b_j.$$

There is an efficient greedy algorithm to solve the subset sum problem if the b_i are a superincreasing sequence: Just subtract the largest possible value from s and repeat.

Example 19.13.6. The sequence

$$1, 2, 4, 8, \ldots, 2^{n-1}$$

is a superincreasing sequence. Decomposing an integer s with respect to this sequence is the same as writing it in binary.

Exercise 19.13.7. Consider the superincreasing sequence

$$1,\ 5,\ 7,\ 20,\ 35,\ 80,\ 170.$$

Decompose $s = 112$ with respect to this sequence.

Exercise 19.13.8. Show that if $b_1, \ldots, b_n$ is a superincreasing sequence then $b_i \geq 2^{i-1}$ and $b_{i+j} > 2^{j-1} b_i$ for $1 \leq i \leq n$ and $1 \leq j$.

The following definition gives a rough estimate for the "information rate" of a knapsack cryptosystem (in other words, the ratio of the number of bits to represent the solution of a subset sum instance versus the number of bits in sum itself). This quantity arises in the cryptanalysis of knapsack cryptosystems.

Definition 19.13.9. The **density** of a sequence $b_1, \ldots, b_n$ is

$$d = n / \log_2 \max\{b_i\}.$$

Exercise 19.13.10. What is the density of $1, 2, 4, 8, \ldots, 2^{n-1}$?

Exercise 19.13.11. What is the density of $3, 7, 11, 27, 50, 107, 210, 430$?

Exercise 19.13.12. Show that the density of a superincreasing sequence is at most $1 + 1/(n-1)$.

19.13.1 Public Key Encryption Using Knapsacks

The idea of the Merkle-Hellman knapsack cryptosystem is to have a superincreasing sequence as the private key but to 'disguise' this for the public key. We briefly sketch the algorithms for the "textbook" Merkle-Hellman knapsack cryptosystem (for more details see Section 8.6 of [418]). The length n of the sequence is a security parameter.

- KeyGen(n): Generate a superincreasing sequence $b_1, \ldots, b_n$ in $\mathbb{N}$. Choose a modulus $M > \sum_{i=1}^{n} b_i$ and a random integer W coprime to M. Select a random permutation π of the integers $\{1, \ldots, n\}$. Define $a_i = W b_{\pi(i)} \pmod{M}$. The public key is $\underline{a} = (a_1, \ldots, a_n)$ and the private key is $\pi, W, M, \underline{b} = (b_1, \ldots, b_n)$.
- The message space is $M_n = \{0,1\}^n$ (i.e., binary strings of length n).
- To Encrypt a message $\underline{m} = (m_1, \ldots, m_n)$ where $m_i \in \{0,1\}$ a user computes the integer
$$c = \underline{m} \cdot \underline{a} = \sum_{i=1}^{n} m_i a_i$$
and transmits this.
- To Decrypt, the user with the private key multiplies c by $W^{-1} \pmod{M}$ to obtain $0 \leq s < M$. The user can solve the subset sum problem for s with respect to the superincreasing sequence. (If there is no solution then the decryption algorithm outputs the invalid ciphertext symbol $\perp$.) The message is then obtained by permuting the sequence x_i using π^{-1}.

Exercise 19.13.13. Show that decryption does recover the message.

Example 19.13.14. Consider the superincreasing sequence from Exercise 19.13.7

$$1,\ 5,\ 7,\ 20,\ 35,\ 80,\ 170.$$

We disguise using modulus 503 and multiplier 430 (and taking π to be the identity permutation for simplicity) to get the public key

$$430,\ 138,\ 495,\ 49,\ 463,\ 196,\ 165.$$

Let the message be the binary sequence 1001100. The ciphertext is

$$c = 430 + 49 + 463 = 942.$$

To decrypt we compute $430^{-1}c \equiv 56 \pmod{503}$, which is then easily decomposed as $35 + 20 + 1$ giving the message 1001100.

Exercise 19.13.15. Consider the Merkle-Hellman public and private key from Example 19.13.14. Decrypt the ciphertext 829.

Exercise 19.13.16. What is the density of the public key in Example 19.13.14.

Exercise 19.13.17. Consider the Merkle-Hellman private key $M = 201$, $W = 77$ and $\underline{b} = (2, 5, 11, 27, 46, 100)$. What is the public key? What is the encryption of 101011?

Exercise 19.13.18. Show that the "textbook" Merkle-Hellman knapsack does not have IND-CPA security.

Exercise 19.13.19. Show that the "textbook" Merkle-Hellman knapsack does not have OWE-CCA security.

Example 19.13.20. One can compute a single bit of information about the message from the ciphertext in the the "textbook" Merkle-Hellman system. Suppose that not all $a_1, \ldots, a_n$ in the public key are even (if so, divide all a_i and the challenge ciphertext c by 2 and try again). Then $c \equiv \sum_{i=1}^{n} x_i a_i \pmod 2$ and so one obtains

$$\sum_{i=1, a_i \text{ odd}}^{n} x_i \pmod 2.$$

In general one prefers the cipertext to have a similar size to n. Exercise 19.13.21 shows that it is impossible to have a ciphertext of exactly the same bit-length as the message when using knapsacks.

Exercise 19.13.21. Show that a ciphertext in the "textbook" Merkle-Hellman scheme is expected to require at least $n + \log_2(n) - 2$ bits.

Exercise 19.13.22. It is sometimes stated in the literature that a Merkle-Hellman public key must have density less than 1. Show that this is not the case.

To avoid attacks (to be described in the next section) it was proposed to iterate the Merkle-Hellman procedure t times. In other words, first choose a superincreasing sequence $b_1, \ldots, b_n$, choose (M_1, W_1) such that $M_1 > \sum_{i=1}^{n} b_n$ and compute $a_{1,i} = Wb_i \pmod{M_1}$ for $1 \leq i \leq n$. Then choose (M_2, W_2) such that $M_2 > \sum_{i=1}^{n} a_{1,i}$ and compute $a_{2,i} = Wa_{1,i} \pmod{M_2}$ for $1 \leq i \leq n$ and so on. The public key is $a_{t,1}, \ldots, a_{t,n}$. One can then apply a permutation to the public key if necessary. The original Merkle-Hellman cryptosystem is the case $t = 1$, which is sometimes given the anachronistic name "single-iterated Merkle-Hellman".

Exercise 19.13.23. Give the decryption algorithm for the iterated Merkle-Hellman system.

Exercise 19.13.24. Show that in iterated Merkle-Hellman one expects $M_{i+1} > (n/2)M_i$ for $1 \leq i < n$. Hence, show that the ciphertext in an iterated Merkle-Hellman system is at least $t(\log_2(n) - 1) + n + \log_2(n) - 2$ bits. Determine the expected density of the public key.

It follows that one cannot iterate the Merkle-Hellman construction too many times.

In the next section we will sometimes assume that $b_1 b_2 < M$. Exercise 19.13.25 shows that if this is not the case then ciphertexts are roughly double the length of the message, and hence are less desirable for practical applications.

Exercise 19.13.25. Let $b_1, \ldots, b_n$ be a superincreasing sequence and suppose $M > \sum_{i=1}^{n} b_i$ is such that $b_1 b_2 > M$. Show that the average ciphertext size is at least $2n + \log_2(n) - 6$ bits.

19.13.2 Cryptanalysis of Knapsack Cryptosystems

We now give a number of attacks on the knapsack cryptosystem, some of which are easy exercises. For a thorough discussion of the history of these attacks see Brickell and Odlyzko [104] and Odlyzko [470].

We remark that there is not necessarily a unique private key for a given Merkle-Hellman knapsack public key since $\{W^{-1} a_i \pmod{M} : 1 \leq i \leq n\}$ might be a superincreasing sequence for more than one choice of (M, W).

Exercise 19.13.26. Show that, given a Merkle-Hellman knapsack public key $(a_1, \ldots, a_n)$, one can efficiently determine whether a guess for (M, W) provides a useful private key.

We now show that the scheme is insecure if the first elements of the superincreasing sequence are known.

Example 19.13.27. Let $a_1, \ldots, a_n$ be a Merkle-Hellman knapsack public key. Suppose one knows the first two elements b_1 and b_2 of the superincreasing sequence. We show how to recover the private key.

First, suppose no permutation is used. Then $a_1 \equiv W b_1 \pmod{M}$ and $a_2 \equiv W b_2 \pmod{M}$. It follows that $a_1 b_2 \equiv a_2 b_1 \pmod{M}$ and so M is a factor of $(a_1 b_2 - a_2 b_1)$. Since b_1 and b_2 are small, $a_i \approx 2^n$ (perhaps $n = 256$) and $(a_1 b_2 - a_2 b_1)$ is not expected to have any special form, it is natural to assume that this factoring problem is fairly easy. Furthermore, since $\max\{a_i : 1 \leq i \leq n\} < M$ and we expect $M < 2 \max\{a_i : 1 \leq i \leq n\}$ (i.e., not all $a_i < M/2$) there are few possible values for M.

For each possible value of M one can compute $W = a_1 b_1^{-1} \pmod{M}$ and then test whether the values $W^{-1} a_i \pmod{M}$ look like a superincreasing sequence.

To deal with the permutation just repeat the attack for all triples (a_i, a_j) with $1 \leq i, j \leq n$ distinct. If (i, j) does not correspond to the correct permutation of $(1, 2)$ then probably either $(a_i b_2 - a_j b_1)$ does not have a factor of the right size, or the corresponding W do not yield superincreasing sequences.

Exercise 19.13.28. Perform the attack of Example 19.13.27 for the Merkle-Hellman public key

$$8391588,\ 471287,\ 8625204,\ 906027,\ 8328886$$

given that $b_1 = 44899$ and $b_2 = 1048697$ (with no permutation used).

In practice, due to Example 19.13.27, one would take b_1 and b_2 to have around $2^{\kappa/2}$ bits each for security parameter κ.

We now show why M must be kept secret.

Example 19.13.29. Suppose M is known to the attacker, who wants to compute W and hence the superincreasing sequence $b_1, \ldots, b_n$.

First, assume no permutation is used. Since

$$a_i \equiv b_i W \pmod{M}$$

for $1 \leq i \leq n$ one has $b_2 \equiv b_1(a_2 a_1^{-1}) \pmod{M}$. Let $0 \leq c < M$ be such that $c \equiv a_2 a_1^{-1} \pmod{M}$. Then

$$b_2 = b_1 c + zM$$

where $z < b_1$. Running the extended Euclidean algorithm on (c, M) computes all triples (b_1, z, b_2) such that $b_1 b_2 < M$ in polynomial-time (following Exercise 19.13.25 we assume that the desired solution satisfies this condition). For each candiate pair (b_1, b_2) one checks whether $a_1 b_1^{-1} \equiv a_2 b_2^{-1} \pmod{M}$ and, if so, calls this value W and tests whether $W^{-1} b_i \pmod{N}$ is "small" (at most $M/2$, and usually much smaller) for a few randomly chosen indices i. One expects to easily find the right pair (b_1, b_2) and hence the correct value for W.

When the permutation is used one repeats the above attack for all pairs (a_i, a_j) for distinct $1 \leq i, j \leq n$.

Exercise 19.13.30. Show that W must be kept secret.

Shamir's Attack

We now present an attack using lattices to compute both M and W together (actually, to compute M and $U = W^{-1} \pmod{M}$ where $1 \leq U < M$). This approach originates with Shamir [546] although we follow the presentation of Lagarias [360]. For clarity, we first assume that no permutation is used. The starting point is to note that for $1 \leq i \leq n$ there are integers k_i such that

$$a_i U - k_i M = b_i$$

and $0 \leq k_i < a_i$. Hence,

$$0 \leq \frac{U}{M} - \frac{k_i}{a_i} = \frac{b_i}{a_i M}. \tag{19.7}$$

Since the b_i are superincreasing we have $b_i < M/2^{n-i}$ and so $0 \leq U/M - k_i/a_i < 1/(a_i 2^{n-i})$. In particular, $U/M - k_1/a_1 < 1/(a_1 2^{n-1})$ is very small.

We now observe that to break the Merkle-Hellman knapsack it is sufficient to find any pair (u, m) of positive integers such that $ua_i \pmod{m}$ is a superincreasing sequence (or at least is similar enough to such a sequence that one can solve the subset sum problem). We show in the next paragraph that if k_1/a_1 is close enough to U/M then taking $(u, m) = (k_1, a_1)$ will suffice.

Subtracting the case $i = 1$ of equation (19.7) from the i-th gives

$$\frac{k_1}{a_1} - \frac{k_i}{a_i} = \frac{b_i}{a_i M} - \frac{b_1}{a_1 M} = \frac{a_1 b_i - a_i b_1}{a_1 a_i M}$$

and so, for $2 \leq i \leq n$,

$$|a_i k_1 - a_1 k_i| = |a_1 b_i - a_i b_1|/M < 2Mb_i/M = 2b_i < M/2^{n-i-1}. \tag{19.8}$$

Taking $m = a_1$ and $u = k_1$ we have $ua_i \pmod{m}$ being very close to a superincreasing sequence (in the sense that the numbers grow in a very controlled way).

It remains to compute the integer k_1 such that equation (19.8) holds, given only the integers $a_1, \ldots, a_n$. Another way to write equation (19.8) is $|a_i/a_1 - k_i/k_1| < M/(a_1 k_1 2^{n-i-1})$ and one sees that the problem is precisely simultaneous Diophantine approximation as considered in Section 19.5. We apply the method of Section 19.5.

Hence, consider the following basis matrix (where $0 < \lambda < 1$ is a parameter analogous to ϵ/Q in equation (19.5) and where $1 < l \leq n$)

$$\begin{pmatrix} \lambda & a_2 & a_3 & \cdots & a_l \\ 0 & -a_1 & 0 & \cdots & 0 \\ 0 & 0 & -a_1 & & \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & & \cdots & -a_1 \end{pmatrix}. \tag{19.9}$$

This lattice contains the vector $(\lambda k_1, k_1 a_2 - k_2 a_1, k_1 a_3 - k_3 a_1, \ldots, k_1 a_d - k_d a_1)$. Perfoming lattice basis reduction one obtains a guess for k_1. One now sets $u = k_1$ and $m = a_1$ and computes $ua_i \pmod m$ for $2 \leq i \leq n$. Hopefully this is a superincreasing sequence (or, is at least close enough to one to allow efficient decryption). One then computes $uc \pmod n$ where c is any challenge ciphertext, decrypts using the superincreasing sequence, and therefore recovers the message. One might expect to have to take $l = n$, but the attack actually works for rather small values of l (see below for more discussion).

Example 19.13.31. Consider the superincreasing sequence

$$b_1 = 7, b_2 = 20, b_3 = 35, b_4 = 71, b_5 = 140, b_6 = 307, b_7 = 651, b_8 = 1301.$$

Choose $M = 2609$ and $W = 2525$ (giving $U = 528$). The Merkle-Hellman public key is

$$(2021, 929, 2278, 1863, 1285, 302, 105, 294).$$

The encryption of 10101011 is 5983.

The sequence of values k_i such that $a_i U - k_i M = b_i$ are

$$409, 188, 461, 377, 260, 61, 21, 59$$

and the values of $a_i k_1 - a_1 k_i$ for $i = 2, 3, 4$ are 13, 21 and 50.

Take $\lambda = 1/32$ and $l = 4$ and consider the lattice basis as in equation (19.9). LLL reduction gives

$$\begin{pmatrix} 409/32 & 13 & 21 & 50 \\ 2021/32 & 0 & 0 & 0 \\ -755/32 & -108 & -19 & 51 \\ 205/8 & -137 & 556 & -216 \end{pmatrix}.$$

One recovers $k_1 = 409$ (and the first few values $a_i k_1 - a_1 k_i$). One can even get the result using $\lambda = 1/8$ and $l = 3$. The LLL-reduced basis is

$$\begin{pmatrix} -409/8 & -13 & -21 \\ 63/8 & -82 & 23 \\ 385/8 & -52 & -84 \end{pmatrix}.$$

To complete the cryptanalysis take $u = k_1 = 409$ and $m = a_1 = 2021$. The sequence $a_i u \pmod m$ for $1 \leq i \leq 8$ is $(0, 13, 21, 50, 105, 237, 504, 1007)$, which is a superincreasing sequence. One then computes $5983u \equiv 1637 \pmod m$, which decomposes with respect to the superincreasing sequence as $1637 = 1007 + 504 + 105 + 21$. The corresponding message is $x_1 0101011$ where $x_1 \in \{0, 1\}$. Using the original public key one can determine that $x_1 = 1$ and confirm that the message does encrypt to the given ciphertext.

Exercise 19.13.32. Using the same public key as Example 19.13.31 decrypt the ciphertext 5522 using the key (u, m) determined by the cryptanalysis.

Exercise 19.13.33. Consider the Merkle-Hellman public key 1994, 1966, 1889, 822, 640, 1224, 1402, 1492 and ciphertext 6569. Deduce the message using Shamir's attack.

A formal analysis of this method is given by Lagarias [360]. As with other attacks on knapsack cryptosystems, the results are heuristic in the sense that they are proved by considering a "random" knapsack instance. The first issue is the size of l. Shamir [546] and Lagarias [360] both suggest that one can take $l > 1/d + 1$, where d is the density of the instance. In practice, $l = 4$ seems to be acceptable. This means the computation is only for lattices of very small dimensions and is certainly polynomial-time. So far we have ignored the permutation; in practice the attack is repeated for the $n(n-1)\cdots(n-(l-1))$ choices of l values from $(a_1, \ldots, a_n)$. Since l is constant this still gives a polynomial-time attack. For these reasons the Merkle-Hellman knapsack cryptosystem is considered to be totally broken.

The iterated Merkle-Hellman system is designed to avoid the above attacks, though Lagarias [360] showed how to attack the double iterated knapsack (i.e., the case $t = 2$) and discussed how to generalise the attack to larger t using exactly the same methods. Brickell [103] also discusses a heuristic method to attack the general iterated Merkle-Hellman system.

Direct Lattice Attack on Subset Sum

We now discuss a more direct way to use lattices to solve the subset sum problem and hence break knapsack cryptosystems. This idea originates in the work of Lagarias and Odlyzko [362]. These methods do not rely on any properties of the subset sum instance and so can be applied to iterated Merkle-Hellman. However, they only work when the density is sufficiently small.

Let $(a_1, \ldots, a_n)$ be a sequence of weights and let $s = \sum_{i=1}^{n} x_i a_i$ be a subset sum instance. Note that $s' = \sum_{i=1}^{n} a_i - s$ is the subset sum instance of the complement $\bar{x_1} \cdots \bar{x_n}$ (where $\bar{0} = 1$ and $\bar{1} = 0$). Since one can repeat any attack on s and s' in turn we may always assume that at most half the entries of the solution are non-zero.

The basic method is to consider the lattice L with basis

$$\begin{pmatrix} I_n & \underline{a} \\ \underline{0} & -s \end{pmatrix} \tag{19.10}$$

where I_n is an $n \times n$ identity matrix and $\underline{a}$ is the list of weights represented as a column vector. Then

$$\underline{v} = (x_1, x_2, \ldots, x_n, 0)$$

is a vector in the lattice. Since $\|v\| \leq \sqrt{n}$ this vector is very short and so one could hope to find it using lattice basis reduction.

Example 19.13.34. Consider the subset sum instance from Example 19.13.14. Reducing the basis

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 430 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 138 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 495 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 49 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 463 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 196 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 165 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -942 \end{pmatrix}$$

using LLL gives

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & 1 & 1 \\ 1 & 0 & 1 & 0 & -2 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 2 & 1 & 0 & 1 & 0 & 1 \\ 0 & -2 & 1 & 1 & -1 & 1 & 0 & 1 \\ -1 & 0 & 1 & 2 & 0 & 0 & -1 & -2 \\ 0 & 0 & 1 & 0 & 0 & 0 & -3 & 0 \end{pmatrix}.$$

One sees that the message $(1, 0, 0, 1, 1, 0, 0, 0)$ appears as the first row (smallest vector) in the lattice.

Exercise 19.13.35. Consider the knapsack public key

$$2381, 1094, 2188, 2442, 2280, 1129, 1803, 2259, 1665$$

and ciphertext 7598. Determine the message using the direct lattice method.

Lagarias and Odlyzko analysed the method for "random" subset sum instances of a given size. They showed (Theorem 3.3 of [362], also see Section 2 of [154]) that for randomly chosen weights a_i of size $2^{\beta n}$ with $\beta > 1.54725$ (i.e., random subset sum instances of density at most 0.6463) then with overwhelming probability (as n tends to infinity) the desired solution vector $\underline{x}$ is the shortest non-zero vector in the lattice. If one can solve the shortest vector problem then one therefore can break the cryptosystem.

There are therefore two problems to overcome. First, the statement only holds for randomly generated weights of a given size and so does not say anything concrete about specific instances. Second, there is no known efficient algorithm to solve SVP exactly. This latter point is a serious problem: as seen in Example 19.13.34 there are many very small vectors in the lattice that do not have entries only in $\{0, 1\}$ (these are often called **parasitic solutions** to the subset sum instance). Hence, for large n, it is quite possible that LLL outputs a short basis that does not include the desired solution vector. Nevertheless, the LLL algorithm does work well in practice and can be used to solve subset sum instances when the density is not too high.

Theorem 3.5 of Lagarias and Odlyzko [362] shows (for randomly chosen weights a_i of size $2^{(1/2+\beta)n^2}$ with $\beta > 0$) that with overwhelming probability (as n tends to infinity) the desired solution vector $\underline{x}$ is computed using the LLL algorithm. This is done by showing that the parasitic solutions all have significantly larger size. The problem with this result is that it only applies when the density satisfies $d \leq (1/2 + \beta)^{-1} 1/n$, which is extremely low.

Coster, Joux, LaMacchia, Odlyzko, Schnorr and Stern [154] improved the method by replacing the last row of the lattice in equation (19.10) by $(1/2, 1/2, \ldots, 1/2, s)$. Under the same simplifying assumptions as used by Lagarias and Odlyzko they showed the attack could be applied for instances with density $d < 0.9408$. Again, although their method officially requires an efficient algorithm for SVP, solving the approximate SVP using LLL works well in practice as long as n is not too large. An alternative formulation of this method is given in Section 3.2 of Nguyen and Stern [463].

The direct lattice attacks require lattices of dimension n so can be defeated by choosing n sufficiently large. Hence, the high-density subset sum problem remains hard in general. The problem with knapsack cryptosystems is that one needs to iterate the basic Merkle-Hellman construction sufficiently many times to avoid the attacks presented earlier. Iterating the Merkle-Hellman method lowers the lattice density and this can make

the system vulnerable to the direct lattice attack unless n is rather large. To conclude, it seems that iterated knapsack cryptosystems are completely broken.