1 The Subset-Sum Problem

We begin by recalling the definition of the *subset-sum* problem—sometimes also called the "knapsack" problem—in its search form.

Definition 1.1 (Subset-Sum). Given positive integer weights $\mathbf{a} = (a_1, \ldots, a_n)$ and $s = \sum_{i=1}^{n} a_i x_i = \langle \mathbf{a}, \mathbf{x} \rangle \in \mathbb{Z}$ for some bits $x_i \in \{0, 1\}$, find $\mathbf{x} = (x_1, \ldots, x_n)$.

The subset-sum problem (in its natural decision variant) is NP-complete. However, recall that NP-completeness is a *worst-case* notion, i.e., there does not appear to be an efficient algorithm that solves *every* instance of subset-sum. Whether or not "most instances" can be solved efficiently, and what "most instances" even means, is a separate question. As we will see below, certain "structured" instances of subset-sum are easily solved. Moreover, we will see that if the bit length of the a_i is large enough relative to n, then subset-sum is easy to solve for almost every choice of $\mathbf{a}$, using LLL.

2 Knapsack Cryptography

Motivated by the simplicity and NP-completeness of subset-sum, in the late 1970's there were proposals to use it as the basis of public-key encryption schemes; see [Odl90] for a survey. In these systems, the public key consists of weights $\mathbf{a} = (a_1, \ldots a_n)$ chosen from some specified distribution, and to encrypt a message $\mathbf{x} \in \{0, 1\}^n$ one computes the ciphertext

$$s = \mathsf{Enc}_{\mathbf{a}}(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle.$$

A major advantage of this kind of encryption algorithm is its efficiency: encrypting involves just summing up n integers, which is much faster than operations like modular exponentiation, as used in other cryptosystems. As for security, recovering the message $\mathbf{x}$ from the ciphertext is equivalent to solving the subset-sum instance $(\mathbf{a}, s)$, which we would like to be hard.[1] Of course, the receiver who generated the public key needs to have a way of recovering the message, i.e., decrypting the ciphertext. This is achieved by embedding a secret "trapdoor" into the weights, which allows the receiver to convert the ciphertext into a different, easily solvable subset-sum instance.

One class of easily solved subset-sum instances involves weights of the following type.

Definition 2.1. A sequence $\mathbf{a} = (a_1, \ldots, a_n)$ is *superincreasing* if $a_i > \sum_{j=1}^{i-1} a_j$ for all i.

Given any superincreasing sequence $\mathbf{a}$ and $s = \langle \mathbf{a}, \mathbf{x} \rangle$, it is easy to find $\mathbf{x}$: observe that $x_n = 1$ if and only if $s > \sum_{j=1}^{n-1} a_i$. Having found x_n, we can then recursively solve the instance $(\mathbf{a}' = (a_1, \ldots, a_{n-1}), s' = s - a_n x_n)$, which still involves superincreasing weights.

Of course, we cannot use a superincreasing sequence as the public key, or it would be trivial for an eavesdropper to decrypt. The final idea is to embed a superincreasing sequence into a "random-looking" public key, along with a trapdoor that lets us convert the latter back to the former. The original method of doing so, proposed by Merkle and Hellman [MH78], permutes and multiplies the weights by a random secret (modulo another random modulus), as follows:

[1] For this lecture, we ignore the fact that accepted notions of security for encryption require much more than hardness of recovering the entire message. However, such hardness is clearly necessary: if the message is indeed easy to recover by an eavesdropper, then the scheme is insecure.

1. Start with some superincreasing sequence $\mathbf{b} = (b_1, \ldots, b_n)$.
2. Choose some modulus $m > \sum_{i=1}^n b_i$, a uniformly random multiplier $w \leftarrow \mathbb{Z}_m^*$, and a uniformly random permutation π on $\{1, \ldots, n\}$.
3. Let $a_i = w \cdot b_{\pi(i)} \bmod m$. The public key is $\mathbf{a} = (a_1, \ldots, a_n)$, and the trapdoor is (m, w, π).

The encryption of a message $\mathbf{x} \in \{0,1\}^n$ is then

$$s = \mathsf{Enc}_{\mathbf{a}}(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle = w \cdot \sum_{i=1}^n b_{\pi(i)} x_i.$$

Given the trapdoor (m, w, π), we can decrypt s as follows: simply compute

$$s' := w^{-1} s = \sum_{i=1}^n b_{\pi(i)} x_i \bmod m,$$

and then solve the subset-sum problem for the (permuted) superincreasing $\mathbf{b}$ and s', where we treat s' as its integer representative in $\{0, \ldots, m-1\}$. This works because $\sum_{i=1}^n b_{\pi(i)} x_i < m$, so s' is the true subset-sum (not modulo anything).

It turns out that some care is needed in choosing the superincreasing sequence $b_1, \ldots, b_n$. For example, the natural choice of $b_i = 2^{i-1}$ ends up admitting some simple attacks. We won't discuss this issue in any detail, because it turns out that the Merkle–Hellman scheme (and almost all of its subsequent variants) can be broken using tools like LLL, regardless of what superincreasing sequence is used.

3 Lattice Attacks on Knapsack Cryptography

In 1982, Shamir [Sha82] showed how to break the basic Merkle–Hellman class of schemes in polynomial time. His attack uses Lenstra's polynomial-time algorithm for fixed-dimension integer programming, which uses LLL as a subroutine. (Shamir's attack has been extended to break many subsequent versions of the Merkle–Hellman system.) Shortly thereafter, Lagarias and Odlyzko [LO83] gave an incomparable attack, later simplified by Frieze [Fri86], that solves almost all instances of "low-density" subset-sum problems.

Definition 3.1. The *density* of a subset-sum instance is $n / \max_i \log a_i$.

Theorem 3.2 (Lagarias–Odlyzko, Frieze). *There is an efficient algorithm that, given uniformly random and independent weights* $a_1, \ldots, a_n \in \{1, \ldots, X\}$*, where* $X \geq 2^{n^2(1/2+\epsilon)}$ *for some arbitrary constant* $\varepsilon > 0$*, and* $s = \langle \mathbf{a}, \mathbf{x} \rangle$ *for some arbitrary* $\mathbf{x} \in \{0,1\}^n$*, outputs* $\mathbf{x}$ *with probability* $1 - 2^{-n^2(\varepsilon - o(1))}$ *over the choice of the* a_i*.*

Notice that the density of the above subset-sum instances is roughly $2/n$.

Proof. We are given a subset-sum instance $(\mathbf{a} = (a_1, \ldots, a_n), s = \langle \mathbf{a}, \mathbf{x} \rangle)$ for some $\mathbf{x} \in \{0,1\}^n$. Without loss of generality, we may assume that $s \geq (\sum_i a_i)/2$: if not, we replace s by $(\sum_{i=1}^n a_i) - s$, which corresponds to flipping all the bits of $\mathbf{x}$. Note that this assumption implies that $\mathbf{x} \neq \mathbf{0}$.

The main idea is to define a lattice where not only is $\mathbf{x}$ a shortest nonzero lattice vector, but all lattice vectors not parallel to $\mathbf{x}$ are much longer, by a factor of $2^{n/2}$ or more. Then because LLL gives a $2^{n/2}$-factor approximation to the shortest lattice vector, it must yield $\mathbf{x}$.

Let $B = \lceil \sqrt{n \cdot 2^n} \rceil$, and define the lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ using the basis

$$\mathbf{B} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ Ba_1 & Ba_2 & \cdots & Ba_n & -Bs \end{pmatrix} \in \mathbb{Z}^{(n+1)\times(n+1)}.$$

Clearly, $\binom{\mathbf{x}}{0} \in \mathcal{L}$. As we will see in a moment, the B factor in the last row serves to amplify the norms of lattice vectors that do not correspond to *exact* equalities $\langle \mathbf{a}, \mathbf{z} \rangle = z_{n+1}s$ for integral $\mathbf{z}, z_{n+1}$.

The algorithm simply runs LLL on the above basis $\mathbf{B}$ to obtain a nonzero lattice vector whose length is within a $2^{n/2}$ factor of $\lambda_1(\mathcal{L})$. The following analysis shows that with high probability, the obtained vector is of the form $k\binom{\mathbf{x}}{0}$ for some nonzero integer k, which reveals the solution $\mathbf{x} \in \{0,1\}^n$.

Notice that $\mathbf{B}\binom{\mathbf{x}}{1} = \binom{\mathbf{x}}{0} \in \mathcal{L}$ is a nonzero lattice vector, and has norm at most $\sqrt{n}$. Also, any lattice vector has a final coordinate divisible by B, and if this coordinate is nonzero, then the vector has length at least $B > 2^{n/2} \cdot \|\mathbf{x}\| \geq 2^{n/2} \cdot \lambda_1(\mathcal{L})$. Therefore, LLL always yields some nonzero lattice vector whose final coordinate is zero, and whose norm is at most $2^{n/2}\sqrt{n}$. We next show that with high probability, nonzero integer multiples of $\binom{\mathbf{x}}{0}$ are the *only* such lattice vectors; therefore, LLL must return one of these.

Fix an arbitrary nonzero vector $\binom{\mathbf{z}}{0} \in \mathbb{Z}^{n+1}$, where $\|\mathbf{z}\| \leq 2^{n/2}\sqrt{n}$ and $\mathbf{z}$ is not an integer multiple of $\mathbf{x}$. We want to bound the probability that this vector is in $\mathcal{L}$, i.e., that $\binom{\mathbf{z}}{0} = \mathbf{B}\binom{\mathbf{z}}{z_{n+1}}$ for some $z_{n+1} \in \mathbb{Z}$. In such an event, we have

$$s \cdot |z_{n+1}| = |s \cdot z_{n+1}| = |\langle \mathbf{a}, \mathbf{z} \rangle| \leq \|\mathbf{z}\| \sum_{i=1}^{n} a_i \leq 2\|\mathbf{z}\|s,$$

so $|z_{n+1}| \leq 2\|\mathbf{z}\|$. Fix any such z_{n+1}. In order for $\binom{\mathbf{z}}{0}$ to be in $\mathcal{L}$, it must be the case that

$$\langle \mathbf{a}, \mathbf{z} \rangle = z_{n+1} \cdot s = z_{n+1}\langle \mathbf{a}, \mathbf{x} \rangle,$$

which implies that $\langle \mathbf{a}, \mathbf{y} \rangle = 0$ where $\mathbf{y} = \mathbf{z} - z_{n+1}\mathbf{x}$. Since $\mathbf{z}$ is not an integer multiple of $\mathbf{x}$, some $y_i \neq 0$, and we can assume that without loss of generality that $i = 1$. Therefore, we must have $a_1 = -(\sum_{i=2}^{n} a_i y_i)/y_1$.

With these observations, for any fixed $\mathbf{z}, z_{n+1}$ satisfying the above constraints, the probability that $\binom{\mathbf{z}}{0} \in \mathcal{L}$ is bounded by

$$\Pr_{\mathbf{a}}[\langle \mathbf{a}, \mathbf{y} \rangle = 0] = \Pr_{a_1}\left[a_1 = -\Big(\sum_{i=2}^{n} a_i y_i\Big)/y_1\right] \leq 1/X,$$

because the a_i are chosen uniformly from $\{1, \ldots, X\}$.

Finally, we apply the union bound over all relevant choices of $\mathbf{z}, z_{n+1}$. Because $\|\mathbf{z}\| \leq 2^{n/2}\sqrt{n} \leq B$, each coordinate of $\mathbf{z}$ has magnitude at most B, and similarly, $|z_{n+1}| \leq 2\|\mathbf{z}\| \leq 2B$. Therefore, the number of choices for $\mathbf{z}, z_{n+1}$ is (crudely) upper bounded by

$$(2B+1)^n \cdot (4B+1) \leq (5B)^{n+1} \leq 2^{n^2(1/2+o(1))}.$$

Because $X = 2^{n^2(1/2+\epsilon)}$ for some constant $\epsilon > 0$, the probability that there exists any $\binom{\mathbf{z}}{0} \in \mathcal{L}$ satisfying the above constraints is at most $2^{-n^2(\varepsilon - o(1))}$, as claimed. $\square$

Variants. We have shown that, except for integer multiples of $\binom{\mathbf{x}}{0}$, no lattice vector has length less than $2^{n/2}\sqrt{n}$ (with high probability). So, LLL's approximation factor of $2^{n/2}$ guarantees that it returns $k\binom{\mathbf{x}}{0}$ for some nonzero integer k. Inspecting the analysis, the $2^{n/2}$ approximation factor corresponds to the density bound of $2/n$. More generally, for an approximation factor γ, to dispense with lattice vectors having a nonzero final coordinate we set $B \approx \gamma$, so the union bound is taken over $\approx B^n$ vectors, so we set $X \approx B^n$, giving a density of $n/\log X \approx 1/\log B \approx 1/\log \gamma$.

What if we had an algorithm that achieves a better approximation factor, e.g., one that solves SVP *exactly*, or to within a $\operatorname{poly}(n)$ factor? For a density of about $1/1.6$, following the same kind of argument, but with tighter bounds on the number of allowed $\mathbf{z}$, one can show that $\pm\binom{\mathbf{x}}{0}$ are the *only* shortest vectors in the lattice (with high probability). Similarly, for density $1/\Theta(\log n)$, one can show that all lattice vectors not parallel to $\binom{\mathbf{x}}{0}$ are some $\operatorname{poly}(n)$ factor longer than it. However, at densities above $2/3$ or so, $\binom{\mathbf{x}}{0}$ may no longer be a shortest nonzero vector in the lattice, so even an exact-SVP oracle might not reveal a subset-sum solution.

References

[Fri86] A. M. Frieze. On the Lagarias-Odlyzko algorithm for the subset sum problem. *SIAM J. Comput.*, 15(2):536–539, 1986. Page 2.

[LO83] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985. Preliminary version in FOCS 1983. Page 2.

[MH78] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inf. Theory*, 24(5):525–530, 1978. Page 1.

[Odl90] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In C. Pomerance, editor, *Cryptology and Computational Number Theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*, pages 75–88. 1990. Page 1.

[Sha82] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Trans. Inf. Theory*, 30(5):699–704, 1984. Preliminary version in CRYPTO 1982. Page 2.