THE KNAPSACK CRYPTOSYSTEM

EZRA FURTADO-TIWARI

Abstract.
We introduce the Merkle-Hellman knapsack cryptosystem, attacks on the cryptosystem, and a more secure knapsack cryptosystem called the Chor-Rivest cryptosystem.

1. The Knapsack Cryptosystem

The knapsack cryptosystem, built around the NP-hard subset sum problem, relies on the fact that Alice can compute subset sums on an "easy" array and then obfuscate it so that Eve cannot compute subset sums in polynomial time. They thus constructed their cryptosystem around superincreasing arrays.

Definition 1.1. Let $a_1, a_2, \ldots, a_n$ be a sequence. We say it is *superincreasing* if $a_i > \sum_{j=1}^{i-1} a_j$ for all $1 < i \leq n$.

The main advantage of using superincreasing arrays is that they allow us to "greedily" solve the knapsack problem in linear time.

Lemma 1.2. *Given a superincreasing sequence $\{a_n\}$ and an integer k, we may find a subsequence of $\{a_n\}$ with sum k (or prove that no such subsequence exists) in polynomial time.*

Proof. Assume that $a_i > k$ for some $1 \leq i \leq n$. Then for $j > i$, it follows that $a_j > k$ as well. Now assume that $a_i < k$ for some i. Either we choose the element a_i to be part of the subsequence, meaning that we must find a subsequence with sum $k - a_i$ from the first $i - 1$ elements of $\{a_n\}$, or we do not choose it, meaning we must find a subsequence with sum $k > a_i > \sum_{j=1}^{i-1} a_j$, so we must choose the element a_i. Once we complete this procedure, we may verify whether the sum of the elements we choose is equal to k. If it is, we have found a valid subsequence, and if it does not, then no such subsequence exists. ■

From this property, Merkle and Hellman constructed the following cryptosystem in 1978 and published it in [MH78].

Definition 1.3 (Merkle-Hellman Cryptosystem)**.** The Merkle-Hellman Cryptosystem constructs a public and private key as follows:

(1) Alice generates a superincreasing array $\{a_n\}$.
(2) Alice randomly generates some $k > \sum_{i=1}^{n} a_i$, and an integer $c < k$ such that $\gcd(c, k) = 1$. This pair (c, k) is the private key.
(3) Alice constructs a new sequence $\{b_n\}$ such that

$$b_i = a_i \cdot c \pmod{k}.$$

She uses this sequence as the public key.

Date: August 16, 2024.

(4) To encrypt a message m with n binary digits, Bob computes the sum

$$x = \sum_{i=1}^{n} m_i b_i$$

and sends it to Alice.

(5) When Alice decrypts the message x, she will compute $x' = x \cdot c^{-1} \pmod{k}$, and solve the easier version of the subset sum problem on her sequence $\{a_n\}$ with sum x'.

Remark 1.4 (Iterative Knapsack Cryptosystem). A seemingly more secure version of the knapsack cryptosystem is the *iterated knapsack cryptosystem*, in which a sequence of sufficiently large c_i and k_i are chosen such that

$$b_i = (((a_i \cdot c_0) \bmod k_0) \cdot c_1) \bmod k_1 \cdot \ldots \cdot c_m \bmod k_m).$$

2. Shamir's Attack

Shortly after, in 1982, Shamir published [Sha82], an outline of an attack on the single-round Merkle-Hellman cryptosystem. Although it did not break the iterative cryptosystem they proposed, it contained ideas that allowed other cryptographers to construct more powerful attacks.

Theorem 2.1 (Shamir's Attack)**.** *Given $b_1, \ldots, b_n$ generated by the procedure described in 1.3, we can solve the knapsack problem on it in polynomial time in n.*

The main idea behind Shamir's attack is that we can construct an algorithm to find some *decryption pair* (c', k') such that the sequence $\{d_n\}$ given by $d_i = b_i \cdot c' \pmod{k'}$ is superincreasing. Note that we do not have to choose $c' = c$ and $k' = k$, as any superincreasing array will allow us to decrypt the message in polynomial time.

Note that the congruence

$$b_i (c')^{-1} \equiv a_i \pmod{k'}$$

is equivalent to the equation

$$b_i (c')^{-1} = q_i k' + a_i$$

for some integer q_i. Note that this implies

$$\frac{(c')^{-1}}{k'} - \frac{q_i}{b_i} = \frac{a_i}{b_i k'},$$

meaning that $\frac{q_i}{b_i}$ is close to $\frac{(c')^{-1}}{k'}$. This is because we expect $a_i \approx b_i$ (as b_i is obtained through a pseudorandom process), and k' must be large in order for the cryptosystem to be secure.

Then we have

$$\frac{q_i}{b_i} - \frac{q_1}{b_1} = \frac{a_1}{k' b_1} - \frac{a_i}{k' b_i},$$

which gives us

$$q_i b_1 - q_1 b_i = \frac{1}{k'}(a_1 b_i - a_i b_1).$$

From here we note that if $M > \sum_{i=1}^{\infty} a_i$ and $\{a_n\}$ is superincreasing, we have $0 \le a_i \le \frac{M}{2^{n-i}}$, so that for $1 \le i \le g$,

$$|q_i b_1 - q_1 b_i| \le \frac{M}{2^{n-g}}.$$

We can approximate M by $\max_{i=1}^{n} b_i$ and construct an integer program to compute the q_is. More precisely, we construct the integer program

$$|q_i b_1 - q_1 b_i| \leq \frac{\tilde{M}}{2^{n-g}}, \quad 2 \leq i \leq g,$$
$$1 \leq q_1 \leq \frac{\tilde{M}}{2^{n-g}} - 1,$$

where $\tilde{M} = \max_{i=1}^{n} b_i$. We choose small g and compute the results of this integer program for all $\binom{n}{g}$ subsets of $\{b_n\}$ that could form the first g elements in sorted order. Due to a result from Lenstra we may find a solution to this integer program in polynomial time, and in fact this particular program is likely to have only one solution. By checking this case and some related ones, we can compute $\frac{(c')^{-1}}{k'}$ in polynomial time, which allows us to recover a decryption pair.

3. More Powerful Attacks

Although Lenstra's algorithm for integer programming can be proven to run in polynomial time, it is theoretically very inefficient due to the high polynomial degree in its runtime. However, the discovery of this algorithm motivated the creation of more efficient ways to solve the knapsack problem in certain cases. The following attack is called a *low-density attack* and relies on a property called the *density* of knapsack problems.

Definition 3.1. We define the *density* of a subset sum problem to be

$$d = \frac{n^2}{\log_2 \max_{i=1}^{n} a_i}.$$

Theorem 3.2. *The subset sum problem can be solved in polynomial time if its density is less than* 0.65.

Definition 3.3. An *integer lattice* L is a subgroup of $\mathbb{Z}^n$ under addition given by n linearly independent basis vectors in $\mathbb{R}^n$ (note that by definition of a group, L must contain all linear combinations of the basis vectors). We can write down an integer lattice using an $n \times n$ matrix

$$V = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}$$

where the $\mathbf{v}_i$s are the basis vectors.

Let $\{b_n\}$ be the public key for a knapsack cryptosystem and let s be the sum required. Consider the $(n+1) \times (n+1)$ lattice L given by the matrix

$$V = \begin{bmatrix} 1 & 0 & \dots & b_1 \\ 0 & 1 & \dots & b_2 \\ \vdots & \ddots & \ddots & b_n \\ 0 & \dots & \dots & -s \end{bmatrix}. \tag{3.1}$$

Now note that if for some set S, we have

$$\sum_{x \in S} b_x = s,$$

then the sum of the corresponding row vectors $\mathbf{v}_x$ will produce the vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_n, 0),$$

where the x_i are the binary digits of the message m we wish to decode. A distinguishing characteristic of such a vector $\mathbf{x}$ is that we expect its magnitude to be very small, considering that it only contains 1s and 0s, and the b_i are assumed to be large to prevent other attacks. In many cases, this vector $\mathbf{x}$ will indeed be the shortest possible nonzero vector that is contained in L.

There is no known polynomial time algorithm to solve the *shortest vector problem* described above, but there exist algorithms to solve a related problem, that of computing a *reduced basis* for a lattice.

Definition 3.4. Let L be an integer lattice. A *reduced basis* of L is a set of linearly independent basis vectors $\mathbf{v}_n$ such that

$$|\mathbf{v}_1|^2 < 2^{n-1} \min_{\mathbf{x} \in L, \mathbf{x} \neq 0} |\mathbf{x}|^2.$$

Lagarias and Odlyzko proved in [LO85] that for almost all integer lattices L formed using the procedure described above, there exists some $\mathbf{v} \in L$ such that if $\mathbf{e}$ is the shortest vector in L and $\sum \mathbf{e}_i \leq \frac{n}{2}$,

$$\mathbf{v}_i = \lambda \mathbf{e}_i$$

for all i.

The final ingredient in our attack is the LLL algorithm, which we will state without proof. Details about this algorithm are provided in [LLL82].

Theorem 3.5 (Lenstra–Lenstra–Lovász Algorithm)**.** *Given an integer lattice V, there exists a polynomial time algorithm to compute a reduced basis for it.*

Finally, we can define the low-density attack in entirety.

Proof of Theorem 3.2. We can use the following procedure, called the low-density attack, to solve the knapsack problem in polynomial time in "almost all" cases if $d < 0.65$:

(1) Let $\{b_n\}$ be the public key and let s be the given sum.
(2) Construct a lattice L from $\{b_n\}$ and s as in (3.1).
(3) Compute a reduced basis U of L using the LLL algorithm.
(4) For each vector $\mathbf{u} \in U$, check for all i if $\mathbf{u}_i = \lambda$ or $\mathbf{u}_i = 0$. For each $\mathbf{u}$ satisfying this condition, compute the vector $\lambda^{-1}\mathbf{u}$, and check if this vector yields a solution to the subset sum problem.
(5) If no solution is found, repeat the procedure with $s' = \sum_{i=1}^{n} b_i - s$. This ensures that $\sum \mathbf{e}_i < \frac{n}{2}$. If that fails, then halt without returning a solution.

■

Due to the procedure by which both the single-round and iterative knapsack cryptosystem are constructed, the knapsack problems created by these procedures almost always have low density, so the low-density attack renders all formulations of the Merkle-Hellman cryptosystem insecure.

4. The Chor-Rivest Cryptosystem

The Chor-Rivest cryptosystem works with higher density subset sum problems and is thus more robust against the low density attack.

Definition 4.1 (Chor-Rivest Cryptosystem)**.** The Chor-Rivest cryptosystem relies on the subset sum problem but works with high density problems. It works as follows:

(1) Let p^h be a prime power such that $p^h - 1$ is B smooth for some small B (and thus it is easy to compute discrete logarithms in $\mathbb{F}_{p^h}$).
(2) Let $f(x)$ be an irreducible monic polynomial over $\mathbb{F}_p$ with degree h. Note that we can represent $\mathbb{F}_{p^h}$ as $\mathbb{F}_p[x]/f(x)$.
(3) Let t be the residue class corresponding to $x \pmod{f(x)}$. Then t is an element of $\mathbb{F}_{p^h}$ and $f(t) = 0$.
(4) Pick a generator g of $(\mathbb{F}_p)^\times$.
(5) For $b \in \mathbb{F}_p$, let a_b be an integer satisfying
$$g^{a_b} \equiv b + t \pmod{p}.$$
(6) Construct an injection $\pi : \{0, \ldots, p-1\} \to \mathbb{F}_p$. Choose a non-negative integer $d < p$, and define a sequence c_i of length p such that
$$c_i \equiv a_{\pi(i)} + d \pmod{p^h - 1}.$$
(7) To encode a message, first transform the message into a p-vector $(m_1, \ldots, m_{p-1})$ such that
$$\sum_{i=0}^{p-1} m_i = h.$$
Then send the message
$$s = \sum_{i=0}^{p-1} m_i c_i.$$
(8) To decrypt a message, compute
$$r = s - dh \pmod{p^h - 1}.$$
We can write
$$g^r = \prod_{i=0}^{p-1} g^{m_i a_{\pi(i)}},$$
which by definition of a is equal to
$$\prod_{i=0}^{p-1} (t + \pi(i))^{m_i}.$$
We may also represent g^r as a polynomial G in x with degree $< h$, which gives us
$$G + f(x) = \prod_{i=0}^{p-1} (t + \pi(i))^{m_i}.$$
We can factor $G + f(x)$ efficiently to recover the message.

Rivest and Chor concluded experimentally that instances of the Chor-Rivest cryptosystem generally have density > 1, rendering the low-density attack unsuccessful. However, more efficient algorithms for solving the subset sum problem have been proposed and can break this cryptosystem as well [IKKS07].

References

[IKKS07] Tetsuya Izu, Jun Kogure, Takeshi Koshiba, and Takeshi Shimoyama. Low-density attack revisited. *Designs, Codes and Cryptography*, 43:47–59, 04 2007.

[LLL82] Arjen Lenstra, Hendrik Lenstra, and Lovász László. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261, 12 1982.

[LO85] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, jan 1985.

[MH78] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

[Sha82] Adi Shamir. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 145–152, 1982.