FOUR TYPES OF CRYPTOSYSTEMS AND RELATED ALGORITHMS

ASTRID (ANRAN) LI

Abstract. Before the invention of public-key cryptography in 1976, two parties in a cryptosystem would need to exchange keys by a non-cryptographic method, but the process of key exchange could be easily eavesdropped. Public-key cryptography uses both private keys and public keys, and the construction of public keys is based on the difficulty to invert one-way functions. In this paper, we will introduce four types of cryptosystems: knapsack cryptosystem, RSA, key exchange, and elliptic-curve cryptography. We will examine the mathematical problems that these cryptosystems are based on and learn algorithms to solve these problem in special cases. We assume some knowledge in elementary number theory, linear algebra, field theory and Galois theory.

Contents

1. Knapsack problem and cryptosystem

Given a set of positive integers A and an integer I, we want to choose a subset of A such that the sum of the elements of the subset is equal to I. This is called the knapsack problem. There may be multiple or no solutions. However, if we require the set to be super-increasing, i.e., every element is larger than the sum of all previous elements, then the solution is unique if it exists.

For example, Let A=\{2,3,7,15,28\} and I=25. We start with the largest number in S and construct a binary sequence: 1 means the element is chosen and 0 means it is not chosen. We cannot choose 28 since 28 $>$ 25, so the first binary digit is 0. 15 is chosen. Then we have 25-15=10 left, so we choose 7 and 3 successively and do not choose 2. Therefore, the sequence is (01110). The knapsack cryptosystem is based on the fact that if the set is super-increasing, then the solution is unique if it exists. We will show an example as we describe the algorithm.

Alice wants to construct her public key. She randomly chooses 3 items:

(1) A super-increasing set A of 5 elements: \{2,3,7,15,28\}; 2=s_1, 3=s_2,..., 28=s_5.
(2) An integer a larger than $s_1 + s_2 + ... + s_5$: a=57.
(3) An integer m larger than a and coprime to a: m=70.

Alice can find b, the inverse of a mod m, by the Euclidean Algorithm. In our example, b=43.

Now Alice constructs a new set T by multiplying each s_i with a: $t_i \equiv as_i$ mod m. We have $T = \{44, 31, 49, 15, 56\}$. This is the public key she publishes.

Suppose Bob wants to send the letter S to Alice. In the usual A-Z (with numerical equivalents from 0-25) alphabet, S corresponds to 18, which is equal to (10010) in binary form. Label the binary digits: $\epsilon_1 = 0, \epsilon_2 = 1, \epsilon_3 = \epsilon_4 = 0, \epsilon_5 = 1$. To encode the message using Alice's public key, Bob computes $C = \sum_{i=1}^{5} \epsilon_i t_i = 1 \times 31 + 1 \times 56 = 87$. This is the number that Bob sends to Alice.

Note that $bC = \sum_{i=1}^{5} \epsilon_i(bt_i)$. Since $t_i = as_i$ and b is the inverse of a mod m, we have $s_i = bt_i$. Therefore, $bC = \sum_{i=1}^{5} \epsilon_i s_i$. Now this problem is reduced to a super-increasing knapsack problem: Alice knows bC and each s_i, and she wants to find ϵ_i, which is equivalent to deciding whether or not to choose the element s_i, given the integer bC.

In our example, $bC = 43 \times 87 \equiv 31$ mod 70. She solves the knapsack problem with 31 and her original set $A = \{2, 3, 7, 15, 28\}$. Since 31=28+3, she obtains the binary sequence $(10010)_2$=18, which corresponds to the letter S.

Security analysis. The advantage of this algorithm is its speed: there is only one modular multiplication to solve. However, this cryptosystem can be attacked in several ways. Before we examine one of the attacks, we introduce the order notation that allows us to compare the magnitudes of different parameters.

Definition 1.1. Let $f(x)$ and $g(x)$ be two functions with positive values. We write $f(x) = \mathcal{O}(g(x))$ if there exist positive real numbers m and M such that for all $x > m$, $f(x) \leq Mg(x)$. In analytic number theory, this notation gives us an approximation of an arithmetical function.

The elements of the original super-increasing set S should be large enough to prevent easy attacks. In particular, s_1, the smallest element of S, should be larger than 2^n (where n is the size of S) and each $s_{i+1} > 2s_i$. Therefore,

$$s_n > 2s_{n-1} > 4s_{n-2} > ... > 2^n s_1 > 2^{2n}.$$

Then a, the integer we chose to be larger than the sum of all elements of S, is larger than $2s_n > 2^{2n+1}$. Since $m > a$, then $m > 2^{2n+1}$ as well.

In our description of the cryptosystem, $t_i \equiv as_i \bmod m$, so $t_i < m$, meaning that $t_i = \mathcal{O}(2^{2n})$. The encoded message C sent to Alice is a linear combination of t_i, so $C = \mathcal{O}(2^{2n})$ too.

An attacker Eve wants to find the vector $x = (\epsilon_1, ..., \epsilon_n)$ as defined in the cryptosysten. She constructs the following vectors with $n+1$ coordinates:

$$\begin{aligned} v_1 &= (2, 0, 0, ..., 0, t_1); \\ v_2 &= (0, 2, 0, ..., 0, t_2); \\ &... \\ v_n &= (0, 0, 0, ..., 2, t_n); \\ v_{n+1} &= (1, 1, 1, ..., 1, C). \end{aligned}$$

Then

$$\begin{aligned} \|v_i\| &= \sqrt{4 + t_i^2} = \mathcal{O}(2^{2n}), 1 \leq i \leq n; \\ \|v_{n+1}\| &= \sqrt{n + C^2} = \mathcal{O}(2^{2n}). \end{aligned}$$

Consider the lattice L generated by $v_1, ..., v_{n+1}$: $\{a_1 v_1 + a_2 v_2 + ... + a_{n+1} v_{n+1} : a_1, ...a_{n+1} \in \mathbb{Z}\}$. Therefore, all the vectors in L should have magnitudes $\mathcal{O}(2^{2n})$.

Let $y = \sum_{i=1}^{n} \epsilon_i v_i - v_{n+1} = (2\epsilon_1 - 1, 2\epsilon_2 - 1, ..., 2\epsilon_n - 1, 0)$. Then $y \in L$. Since $\epsilon_i \in \{0, 1\}$, then $2\epsilon_i - 1 \in \{1, -1\}$. Thus, $\|y\| = \sqrt{1 + 1 + ... + 1 + 0} = \sqrt{n}$. However, this tells us that y is a very small vector in L and vectors in L should have much larger magnitude, so y is most likely the only small nonzero vector in L. There are algorithms to find small vectors in lattices known as *lattice reduction algorithms*. We will not describe this algorithm here but a description can be found in [7].

2. Factorization and RSA

All public-key cryptosystems are based on trapdoor functions: functions that are easily evaluated but almost impossible to invert. One example of such a function is the product of two large primes – it's easy to compute the product of two large primes, but trying to factor a large number is very difficult. The RSA cryptosystem is based on this fact.

2.1. **RSA.**

Let P and C denote the set of message units of the plaintext and ciphertext, respectively. The first thing to do is to standardize the system by fixing the type of the message units. We choose positive integers $k < l$, meaning that P consists of k-digit letter blocks and C consists of l-digit letter blocks. Suppose that our alphabet consists of N letters. Therefore, there are in total N^k possible message units in P and N^l possible message units in C. We can then assign numerical equivalents to each message unit in P and C with integers from 0 to N^k-1 and from 0 to N^l-1 respectively.

Now the user Alice wants to construct her public key. Using a random-number generator, she chooses two large primes p and q such that $N^k < n = pq < N^l$. We can compute $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) = n-p-q+1$, where φ is the Euler's totient function. Next, Alice randomly generates $e \in \mathbb{Z}$ such that $1 < e < \varphi(n)$ and $g.c.d(e, \varphi(n))$=1. Since e and $\varphi(n)$ are coprime, we can always find $d \equiv e^{-1} \bmod \varphi(n)$ by the Euclidean Algorithm. Then Alice publishes her key (n, e).

The enciphering transformation is $C \equiv P^e \bmod n$. Since $n = pq$, we have

$$C \equiv P^e \bmod p \Rightarrow C^d \equiv P^{de} \bmod p.$$

Since $de \equiv 1 \bmod \varphi(n)$, we have

$$de \equiv 1 \bmod (p-1) \Rightarrow de = 1 + k(p-1), k \in \mathbb{Z}.$$

Plugging this equation in, we have

$$C^d \equiv P^{1+k(p-1)} \equiv P(P^{p-1})^k \bmod p.$$

If $p|P$, then $C^d \equiv 0 \equiv P \bmod p$. Otherwise, by Fermat's Little Theorem, $P^{p-1} \equiv 1 \bmod p$, so $C^d \equiv P \bmod p$.

By the same argument, $C^d \equiv P \bmod q$. Then by the Chinese Remainder Theorem, $P \equiv C^d \bmod n$. This is the deciphering transformation, and the deciphering key is (n, d), which only Alice knows. When Bob sends a message to Alice using Alice's public key, Alice can recover the message since only she knows d, her private decryption key.

The degree of safety of RSA depends on the difficulty to factor large numbers. In the following subsections we will describe some factorization algorithms and see their implications on the security of RSA.

2.2. **Factor base algorithm.** The factor base algorithm is inspired by Fermat factorization. In Fermat factorization, we are given the information that n is the product of two primes $p > q$ that are close to each other. Then $n = pq = (\frac{p+q}{2})^2 - (\frac{p-q}{2})^2$.

Let $t = \frac{p+q}{2}$ and $q = \frac{p-q}{2}$, then $t^2 - n = s^2$. Since p and q are close to each other, we know that t is only slightly larger than $\sqrt{n}$ and s is very small. Thus, we can start with the smallest integer larger than $\sqrt{n}$ and increment by 1 until we find one t such that $t^2 - n$ is a perfect square. Thus, we should not choose a product of two close primes in the RSA cryptosystem.

The goal of the factor base algorithm is also to find integers t and s such that $t^2 \equiv s^2 \bmod n$ but $t \not\equiv s \bmod n$. In this way, $n \mid t^2 - s^2 = (t+s)(t-s)$ but $n \nmid t+s$ or $t-s$. Then taking the greatest common divisor of n and $t+s$ gives us a proper factor of n.

Definition 2.1. A *factor base* is a set B of small distinct primes $\{p_1, p_2, ...p_m\}$ except that $p_1 = -1$. It is called a factor base since we will reduce the problem to considering only these small prime factors.

Definition 2.2. x^2 is called a *B-number* if x^2 mod n (where n is the composite integer that we want to factor) in its "least absolute residue" can be factored as a product of the primes in B. The least absolute residue r of x is an integer in the range of $[-n/2, n/2]$ and $r \equiv x \bmod n$. For example, $4^2 \equiv 4 \bmod 6$, and because $4 > 3$, the least absolute residue of 4^2 is -2.

Suppose we have a fixed factor base B and a set of B-numbers. For every B-number x^2 mod n, we construct a corresponding vector $\epsilon \in \mathbb{F}_2^m$ in the following way: factor x^2 mod n, say $x^2 = \prod_{k=1}^m p_k^{\alpha_k}$, and set the k-th coordinate of ϵ to be 0 if α_k is even and 1 if α_k is odd. For example, suppose the factor base is $B = \{-1, 2, 3, 5\}$ and $n = 4633$. $68^2 = -9 = -1 \times 3^2$ in the least absolute residue form, so its corresponding vector is $\epsilon = (1, 0, 0, 0)$.

Now, our goal is to find a set of B-numbers x_i^2 such that the sum of corresponding vectors ϵ_i is equal to (0,0,...0). We can achieve this by creating a matrix M whose rows are these vectors and solve $vM = 0$ by Gaussian elimination, where the vector v tells you how to choose the subset. Suppose we have such a set of B-numbers. Then we take the product of all x_i^2. Then for each prime $p \in B$, its power in the product is the sum of powers of p from each x_i^2. Since their vectors add up to zero, then we know that the power of each prime number in the product is an even number, and so the product is altogether an even power.

For each i, let $y_i = \prod_{k=1}^m p_k^{\alpha_{ik}}$, the factorization of x_i^2 mod n. Then

$$\prod y_i = \prod_{k=1}^{m} p_k^{\sum_{i=1} \alpha_{ik}}. \tag{2.3}$$

Then $\sum_i \alpha_{ik}$ is even. Let $\beta_k = \frac{1}{2}\sum_{i=1} \alpha_{ik}$. So we can express the right hand side of (2.3) as $(\prod_{k=1}^m p_k^{\beta_k})^2$. Remember that our final goal is to construct integers t and s such that $t^2 \equiv s^2 \bmod n$. Now, let

$$t = \textstyle\prod_i x_i,\ s = \prod_{k=1}^m p_k^{\beta_k}.$$

Then the left hand side of (2.3) can be expressed as t^2, while the right hand side can be expressed as s^2. Since all the computation is done mod n, we finally have our desired result: $t^2 \equiv s^2 \bmod n$. Of course, we also have to check that $t \not\equiv s$ mod n. If t and s do not satisfy this requirement, we have to choose another set of B-numbers such that their corresponding vectors add up to 0 in $\mathbb{F}_2^m$.

Example 2.4. We want to factor $n = 4633$ and want to construct a factor base using $68^2, 152^2, 153^2$.

$$68^2 \equiv -9 \equiv -1 \times 3^2 \mod 4633;$$
$$152^2 \equiv -61 \equiv -1 \times 61 \mod 4633;$$
$$153^2 \equiv 244 \equiv 2^2 \times 61 \mod 4633.$$

Thus, our factor base is $B = \{-1, 2, 3, 61\}$. Then we construct their corresponding vectors in $\mathbb{F}_2^4$:

$$\epsilon_{68} = (1, 0, 0, 0);$$
$$\epsilon_{152} = (1, 0, 0, 1);$$
$$\epsilon_{153} = (0, 0, 0, 1).$$

Then $\epsilon_{68} + \epsilon_{152} + \epsilon_{153}$=(0,0,0,0). Let $t = 68 \times 152 \times 153 \equiv 1555 \bmod 4633$, and let $s = 2 \times 3 \times 61$=366. So $g.c.d(4633, 1555+366) = 113$, and we have $4633 = 113 \times 41$.

In this algorithm, we did not describe how to come up with a set with B-numbers. An algorithm to achieve this is the quadratic sieve algorithm. A reference can be found in Section 5.5 of [1].

2.3. **Pollard's ρ method.** Let $p(x)$ be a simple polynomial from $\mathbb{Z}/n\mathbb{Z}$ to itself with integer coefficients. For a fixed element $x_0 \in \mathbb{Z}/n\mathbb{Z}$, define $x_1 = p(x_0)$, $x_2 = p(x_1)$ and so on.

Suppose that after some iteration we have a pair of elements x_j and x_i such that $x_j \not\equiv x_i \bmod n$, but $x_j \equiv x_i \bmod d$, where d is some divisor of n. Then $g.c.d(x_j - x_i, n)$ gives us a nontrivial divisor of n.

However, as we compute more x_j, it takes more time to compute $g.c.d(x_j - x_i, n)$ for every $i < j$. To save running time and storage, we want a method of only computing one $g.c.d$ for each x_j: write j in binary form, and let l be 1 smaller than the number of digits of j, so that $2^l \leq j < 2^{l+1}$. Then we compute $g.c.d(x_j - x_{2^l-1}, n)$. If this does not give us a nontrivial factor of n, we repeat the same procedure for $j+1$.

Example 2.5. Factor $n = 2701$ with $p(x) = x^3+x+1$ and $x_0 = 1$. All computations are done modulo 2701.

$$x_1 = p(1) = 3, g.c.d(3-1, 2701) = 1;$$
$$x_2 = p(3) = 31, g.c.d(31-3, 2701) = 1;$$
$$x_3 = p(31) = 112, g.c.d(112-3, 2701) = 1;$$
$$x_4 = p(112) = 521, g.c.d(521-112, 2701) = 1;$$
$$x_5 = p(521) = 2325, g.c.d(2325-112, 2701) = 1;$$
$$x_6 = p(2325) = 630, g.c.d(630-112, 2701) = 37.$$

So $2701 = 37 \times 73$.

We want an estimate of the running time of this algorithm. Specifically, since we chose $p(x)$ and x_0 randomly, we want to calculate the proportion of pairs $(p(x), x_0)$ such that after several iteration we have the first repetition.

Proposition 2.6. *Let S be a set of size h and $p(x)$ be a map from S to S. For any $x_0 \in S$, define $x_1 = p(x_0)$. Suppose we want the first repetition to occur after k iterations where $k = 1 + [\sqrt{2\lambda h}]$ for some positive real number λ, i.e. $x_0, x_1, ...x_k$ are all distinct. Then the proportion of pairs $(p(x), x_0)$ such that $x_0, x_1, ..., x_k$ are distinct is less than $e^{-\lambda}$.*

The proof of this result reduces to counting the number of pairs for which x_0, x_1,..., x_k are distinct. See Proposition V.2.1 of [1].

3. Discrete log problem and key exchange

3.1. **Diffie-Hellman key exchange.** Named after Whitfield Diffie and Martin Hellman, this method allows two users to publicly exchange their secret keys in an insecure environment. Just as the RSA cryptosystem is based on the difficulty to factor large numbers, the Diffie-Hellman key exchange is based on the difficulty to solve the discrete log problem in a finite field. Let g be a primitive root of $\mathbb{F}_q$ and b be a nonzero element of $\mathbb{F}_q$. The discrete log problem requires us to solve $g^x = b$ in $\mathbb{F}_q$.

Suppose Alice and Bob want to secretly agree upon a key in an environment with eavesdroppers. First, they agree on a large finite field $\mathbb{F}_q$ and a primitive root $g \in \mathbb{F}_q$. Both $\mathbb{F}_q$ and g are published. Alice secretly chooses an integer a and Bob secretly chooses an integer b. They compute g^a and g^b respectively and both values are made public. Their secret key will be g^{ab}: since Alice knows the value of g^b and her secret number a, she can compute g^{ab} and the same applies for Bob. An eavesdropper only knows the values of g^a and g^b and cannot compute g^{ab} without solving a discrete log problem. Thus, there is no danger of being eavesdropped if the discrete log problem is hard enough to solve. In the following subsections, we will look at some algorithms to solve some special discrete log problems.

3.2. **Pohlig-Hellman algorithm.**

The Pohlig-Hellman algorithm is an algorithm for finding discrete logs in a finite field, namely, given an element y that is known to be of the form b^x, find x. Let $\mathbb{F}_q$ be the finite field with q elements and g be a generator of $\mathbb{F}_q$.

First, we factor $q-1$ as a product of primes: $q-1=\prod p^{\alpha}$. For every distinct p dividing $q-1$, define

$$x_p \equiv y_0 + y_1 p + y_2 p^2 + \dots + y_{(\alpha-1)} p^{\alpha-1} \bmod p^{\alpha},$$

where $y_i \in \{0,1,\dots,p-1\}$. For each distinct p, we want to find x_p.

For example, let q=181 and 2 be the generator of $\mathbb{F}_{181}$. We want to solve $2^x = 153$ in $\mathbb{F}_{181}$. $181-1 = 2^2 \times 3^2 \times 5$, so we define:

$$\begin{aligned} x_2 &\equiv y_0 + 2y_1 \quad \bmod 2^2, y_0, y_1 \in \{0,1\}; \\ x_3 &\equiv y_0 + 3y_1 \quad \bmod 3^2, y_0, y_1 \in \{0,1,2\}; \\ x_5 &\equiv y_0 \quad \bmod 5, y_0 \in \{0,1,2,3,4\}. \end{aligned}$$

Note that the y_0, y_1 in each congruence are different. We want to solve the following system of congruences:

$$\begin{aligned} x &\equiv x_2 \quad \bmod 2^2; \\ x &\equiv x_3 \quad \bmod 3^2; \\ x &\equiv x_5 \quad \bmod 5. \end{aligned}$$

We first find x_2. $\frac{180}{2} = 90$, and we compute 153^{90} using the repeated squaring method:

$$\begin{aligned} 153^2 = 23409 \equiv 60 &\mod 181; \\ 153^4 \equiv 60^2 \equiv 3600 \equiv 161 &\mod 181; \\ 153^8 \equiv 161^2 \equiv 25921 \equiv 38 &\mod 181; \\ 153^{16} \equiv 38^2 \equiv 1444 \equiv 177 &\mod 181; \\ 153^{32} \equiv 177^2 \equiv 31329 \equiv 16 &\mod 181; \\ 153^{64} \equiv 16^2 \equiv 256 \equiv 75 &\mod 181. \end{aligned}$$

Then, $153^{90} = 153^{64+16+8+2} \equiv -1 \bmod 181$.

Since $2^x \equiv 153 \bmod 181$, we also have $2^{90x} \equiv -1 \bmod 181$.

By definition, $x \equiv x_2 \equiv y_0 + 2y_1 \bmod 2^2$, so we can write x as $4k + y_0 + 2y_1$. So $2^{90x} = 2^{360k+90y_0+180y_1} \equiv -1 \mod 181$.

Since $g.c.d(2, 181) = 1$ and $\phi(181) = 180$, by Euler's Theorem,

$$\begin{aligned} 2^{360k} \equiv 2^{180y_1} \equiv 1 &\mod 181; \\ 2^{90x} \equiv 2^{90y_0} \equiv -1 &\mod 181. \end{aligned}$$

Since y_0 can only be 0 or 1 and y_0=0 does not satisfy the equation, then y_0=1. Now we want to find y_1. We can compute $153^{45} = 153^{32+8+4+1} = 19$ in $\mathbb{F}_{181}$. Using the same approach, we replace 153 by 2^x:

$$153^{45} = 2^{45x} = 2^{45(4k+y_0+2_{y_1})} = 2^{45+90y_1} \equiv 19 \mod 181.$$

Just as y_0, y_1 can only be 0 or 1. We can easily get $2^{45} = 2^{32+8+4+1} \equiv 162 \bmod 181$. Then we find y_1=1. So we have $x_2 = 1 + 2 \times 1 = 3$.

Using the same method, we find $x_3 = 8$ and $x_5 = 2$. So we have

$$\begin{aligned} x &\equiv 3 \bmod 4; \\ x &\equiv 8 \bmod 9; \\ x &\equiv 2 \bmod 5. \end{aligned}$$

By Chinese Remainder Theorem, there is a unique solution: x=107.

This algorithm is efficient if all prime divisors of $q - 1$ are small; otherwise, it takes a long time to compute all the y_i. Thus, when we choose the finite field $\mathbb{F}_q$ in the Diffie-Hellman method, we should make sure that $q - 1$ has some large prime factors – at least about 20 digits.

3.3. **Index-calculus algorithm.**

Suppose $q = p^n$, where p is a small prime and n is a large integer. This algorithm solves $y = b^x$ for any nonzero $y \in \mathbb{F}_q$, where b is a generator of ${\mathbb{F}_q}^*$. Choose an irreducible polynomial $p(x) \in \mathbb{F}_q[x]$ of degree n, then field theory tells us that $\mathbb{F}_q \cong \mathbb{F}_p[x]/p(x)$. Thus, the base $b \in \mathbb{F}_q$ can be regarded as a polynomial in $\mathbb{F}_p[x]$ with degree less than n. In general, all elements in $\mathbb{F}_q$ can be regarded as polynomials in this way.

Lemma 3.1. *If b is a generator of ${\mathbb{F}_q}^*$, then $b^{q-1/q-1}$ is a generator of ${\mathbb{F}_p}^*$.*

Proof. Let $b' = b^{q-1/q-1}$. We first need to show that $b' \in \mathbb{F}_p$. Let σ be the Frobenius map: $\sigma(a) = a^p$. Then $\sigma(b') = b^{(q-1/p-1)p} = b^{p^n+p^{n-1}+\ldots+p}$. Since b is a generator of ${\mathbb{F}_q}^*$, $b^{p^n} = b$, so $\sigma(b') = b^{p^{n-1}+\ldots+p+1} = b'$. b' is fixed by σ, so we know $b' \in \mathbb{F}_p$.

We know that the first $p^n - 1$ powers of b are distinct, so the first $p-1$ powers of b' are distinct. Thus, b' is a generator of ${\mathbb{F}_p}^*$. $\square$

After finding a generator in $\mathbb{F}_p$, we can reduce our problem by considering this smaller field. Now we define some notation.

Definition 3.2. For $a, b \in \mathbb{F}_q$, rewrite them as $a(x), b(x)$ since they can be considered as polynomials in $\mathbb{F}_p[x]$ with degree less than n. Let $index(a(x))$ be the discrete log of $a(x)$ to the base $b(x)$, i.e. $a(x)^{index(a(x))} = b(x)$.

Let B be a subset of $\mathbb{F}_q$ consisting of all irreducible polynomials of $\mathbb{F}_p[x]$ of degree less than n. Therefore, the size of B is between p and p^n. The index-calculus algorithm has two stages. The first stage is to construct a database of the discrete logs of all polynomials in B. Then we can solve our original discrete log problem using the information we obtain in the first stage.

Stage 1. We choose a random integer k between 1 and $q-2$ and compute the polynomial $c(x) \equiv b(x)^k \mod p(x)$ (remember that $p(x)$ is the irreducible polynomial of degree n we chose before). We want to determine whether $c(x)$ can be factored into a product of polynomials in B, i.e. if $c(x) = c_0 \prod_{f\in B} f(x)^{\alpha_f}$, where c_0 is the leading coefficient of $c(x)$. If it cannot be factored this way, we choose another k and repeat this procedure until we obtain $\#B$ such equations. Notice that this is similar to the idea of the factor-base algorithm.

For each factorization, we take the logarithm of both sides:

$$index(c(x)) = index(c_0) + \textstyle\sum_{f\in B} \alpha_f \; index(f(x)).$$

Since $c(x) \equiv b(x)^k \mod p(x)$, $index(c(x)) = k$. For the constant c_0, we can find $index(c_0)$ using the Pohlig-Hellman algorithm described before. Therefore, the only unknowns in this equation are $index(f(x))$ for all $f \in B$. Since there are $\#B$ unknowns and $\#B$ equations, we can solve this system of linear equations. We have now found the discrete logs of all polynomials in our chosen basis B. This completes the first stage of the algorithm.

Stage 2. We want to find $index(y)$. As before, we randomly choose an integer k between 1 and $q-2$ and compute $y' = yb^k$. This is to say $y'(x) \equiv y(x)b(x)^k \mod p(x)$ when we consider them as polynomials. This gives us $index(y') = index(y)+k$. Again, we determine if $y'(x)$ can be factored into a product of polynomials in B. We keep trying different k until we find a $y'(x)$ that can be factored in this way: $y'(x) = y_0 \prod_{f\in B} f(x)^{\alpha_f}$. Taking the logarithm of both sides, we have

$$index(y') = index(y_0) + \textstyle\sum_{f\in B} \alpha_f index(f(x)).$$

In Stage 1, we have computed $index(f(x))$ for all f in B. For the constant y_0, we can find $index(y_0)$ using the Pohlig-Hellman algorithm. Thus, we can write $index(y')$ explicitly, and $index(y) = index(y') - k$. This completes the index-calculus algorithm.

This algorithm tells us that when we choose the finite field $\mathbb{F}_q$ in the Diffie-Hellman key exchange method, we should make sure that q is not a large power of a small prime.

4. Elliptic curves

In this section, we will first study the algebraic structures of elliptic curves and see how elliptic-curve cryptography allows users to choose smaller keys that guarantee security.

Definition 4.1. Let K be a field whose characteristic is not equal to 2 or 3 and $y^2 = x^3 + ax + b$ be a cubic equation over K. We require that this equation has no multiple roots; equivalently, $4a^2 + 27b^2 \neq 0$. An *elliptic curve over K*, denoted by E, is the set of solutions (x, y) $(x, y \in K)$ satisfying the equation together with an extra point O called the "point at infinity"(explained later).

Rules for addition

(1) If $P = (x, y)$, then define $-P = (x, -y)$, the reflection with respect to the x-axis. By ensuring that E has no repeated roots, $-P$ always exists.
(2) If P and Q are two distinct points on E, then the line passing through P and Q intersects E at exactly one more point R. Define $P + Q = -R$.
   Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and suppose $x_1 \neq x_2$. Then the coordinates of $P + Q = (x_3, y_3)$ are given by
$$x_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)^2 - x_1 - x_2;$$
$$y_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x_1 - x_3) - y_1.$$
(3) If $x_1 = x_2$, but $y_1 \neq y_2$, then $Q = -P$. The line passing through P and $-P$ intersects E a third time at O. Thus, we define $P + (-P) = O$, the point at infinity.
(4) If we add P to itself, then $2P$ is defined as the intersection point between E and the tangent line to E at P.
   By replacing x_2 with x_1 and y_2 with y_1 in the result of case (2), we get the coordinates of $2P$:
$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1;$$
$$y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$
(5) Define nP to be P added to itself n times.
(6) Addition is commutative and associative.

Theorem 4.2. *The points of E form an abelian group, where O behaves as the additive identity.*

Definition 4.3. The *order* N of a point P on an elliptic curve E is the smallest positive integer such that $NP = O$. If such N does not exist, then P is said to have infinite order.

There is an algorithm to compute kP, called the successive doubling algorithm. Let the binary expansion of k be $b_n b_{n-1}...b_1 b_0$. We can easily compute $2P$, $4P$,..., $2^n P$ using the 4-th rule for addition. Then $kP = \sum_{i=0}^{n} 2^{b_i} P$.

E can also be defined on a finite field $\mathbb{F}_q$ where q is a prime power. Since the number of pairs (x, y), where $x, y \in \mathbb{F}_q$, is finite, the number of points on E is also finite and the points form a finite abelian group.

4.1. **Hasse's Theorem.**

This theorem gives us a bound for the number of points on an elliptic curve defined over a finite field.

Theorem 4.4. *Let E be an elliptic curve defined on $\mathbb{F}_p$. Then $|\#E(\mathbb{F}_p) - (p+1)| \leq 2\sqrt{p}$.*

We will now prove the first half of this theorem.

Proposition 4.5. *Let E be an elliptic curve defined over $\mathbb{F}_p$ and ϕ be the Frobenius endomorphism from $\mathbb{F}_p$ to itself. Then $\#E(\mathbb{F}_p)$=$\#ker(\phi - I)$, where I is the identity map.*

Proof. We prove this statement by showing that each of the two sets contains the other.

Since ϕ is the Frobenious endomorphism, $\phi(P) = P$ for every $P \in E(\mathbb{F}_p)$. The kernel of $\phi - I$ is the set of points in E such that $\phi(P) - P = 0$. Therefore, every point in $E(\mathbb{F}_p)$ belongs to $ker(\phi - I)$.

Field theory tells us that every finite extension of a finite field is separable. Consider $\bar{\mathbb{F}_p}$, the algebraic closure of $\mathbb{F}_p$. Then the field extension $\bar{\mathbb{F}_p}/\mathbb{F}_p$ is finite, normal, and separable, so it is Galois. Consider the Galois group $Gal(\bar{\mathbb{F}_p}/\mathbb{F}_p)$, which is generated by the Frobenius map ϕ. Then any element of $\bar{\mathbb{F}_p}$ that is fixed by the Galois group must be in $\mathbb{F}_p$. Therefore, every element in $ker(\phi - I)$ is also in $E(\mathbb{F}_p)$.

Since the two sets are equal, they have the same size. □

To prove the second half of this theorem, we need to study isogenies between elliptic curves.

Definition 4.6. An *isogeny* ϕ between two elliptic curves E_1 and E_2 is a homomorphism from E_1 and E_2 such that $\phi(O_1) = O_2$. If $E_1 = E_2$, then ϕ is an endomorphism.

Definition 4.7. Let ϕ be an isogeny of an elliptic curve E defined over the field K and $P = (x, y) \in E$ where x and y are transcendental over $\bar{K}$, the algebraic closure of K. Then the *degree* of ϕ, $deg(\phi)$, is the degree of the field extension $[\bar{K}(P) : \bar{K}(Q)]$, where $Q = \phi(P)$. The notation $\bar{K}(P)$ means the smallest field containing $\bar{K}$, x and y.

Let us introduce some properties of isogenies between elliptic curves. The proofs of these results will be skipped, but they can be found in [4] and Chapter 3 of [3].

Proposition 4.8. *Let φ be an isogeny from E_1 to E_2 and ϕ be an isogeny from E_2 to E_3. Then $deg(\phi \circ \varphi) = deg(\phi)deg(\varphi)$.*

Proof. See Proposition 9 of [4]. □

Definition 4.9. For any $m \in \mathbb{Z}$, $[m]$ denotes the isogeny from E_1 to E_2 that takes P to mP.

Theorem 4.10. *Let φ be a nonconstant isogeny from E_1 to E_2 with degree m. Then there exists a unique isogeny $\hat{\varphi}$ such that $\hat{\varphi} \circ \varphi = [m]$. $\hat{\varphi}$ is called the dual isogeny of φ.*

Proof. See Appendix 1 of [4]. □

Now we will look at some properties of dual isogenies.

Proposition 4.11. *Let φ and ω be two isogenies from E_1 to E_2 and ϕ be an isogeny from E_2 to E_3.*

(1) $\widehat{\phi \circ \varphi} = \hat{\varphi} \circ \hat{\phi}$;
(2) $\widehat{\varphi + \omega} = \hat{\varphi} + \hat{\omega}$;
(3) $\widehat{[m]} = [m]$;
(4) $deg([m]) = m^2$;
(5) $deg(\varphi) = deg(\hat{\varphi})$;
(6) $\varphi = \hat{\hat{\varphi}}$.

Proof. See Theorem 6.2, Chapter 3, of [3]. □

Definition 4.12. Let V be a vector space over K. A *bilinear form* on V is a map $B : V \times V \to K$ such that

$$\begin{aligned} B(u+v,w) &= B(u,w) + B(v,w) \\ B(\lambda u, v) &= \lambda B(u,v) \\ B(u, v+w) &= B(u,v) + B(u,w) \\ B(u, \lambda v) &= \lambda B(u,v) \end{aligned}$$

for every $u, v, w \in V$ and $\lambda \in K$.

Definition 4.13. A function $Q : V \to K$ is called a *quadratic form* if for every $v \in V$, there exists a symmetric bilinear form B such that $Q(v) = B(v,v)$. Moreover, Q is called *positive definite* if $Q(v) \geq 0$ and $Q(v) = 0$ if and only if $v = 0$. B is *symmetric* if $B(u,v) = B(v,u)$.

Theorem 4.14. *The degree of an isogeny is a positive definite quadratic form.*

Proof. See Corollary 6.3, Chapter 3, of [3]. □

Before we continue the proof of Hasse's theorem, we have to assume another important fact: if φ is a separable isogeny, meaning that the field extension $\bar{K}(Q)/\bar{K}(P)$ is separable, then $\#ker(\varphi) = deg(\varphi)$. A proof can be found in Section 3.4, Theorem 4.10, of [3]. We will also assume that $\phi - I$ is separable, where ϕ is the Frobenius endomorphism. A proof can be found in Section 3.5, Corollary 5.3, of [3].

Since $\#E(\mathbb{F}_p)$=$\#ker(\phi - I) = deg(\phi - I)$, it suffices to give $deg(\phi - I)$ some boundary.

Lemma 4.15. $|deg(\phi - \varphi) - deg(\phi) - deg(\varphi)| \leq 2\sqrt{deg(\phi)deg(\varphi)}$.

Proof. Let $L(\phi, \varphi)$ denote the map $deg(\phi-\varphi)-deg(\phi)-deg(\varphi)$, which is a bilinear form. Therefore, for any $m, n \in \mathbb{Z}$,

$$L(m\phi, n\varphi) = mL(\phi, n\varphi) = mnL(\phi, \varphi).$$

On the other hand,

$$\begin{aligned} L(m\phi, n\varphi) &= deg(m\phi - n\varphi) - deg(m\phi) - deg(n\varphi) \\ &= deg(m\phi - n\varphi) - m^2 deg(\phi) - n^2 deg(\varphi). \end{aligned}$$

So $deg(m\phi - n\varphi) = m^2 deg(\phi) + n^2 deg(\varphi) + mnL(\phi, \varphi)$. Take $m = -L(\phi, \varphi)$ and $n = 2deg(\phi)$. Then

$$\begin{aligned} deg(m\phi - n\varphi) &= L(\phi, \varphi)^2 deg(\phi) + 4deg(\phi)^2 deg(\varphi) - 2deg(\phi)L(\phi, \varphi) \\ &= deg(\phi)(L(\phi, \varphi)^2 + 4deg(\phi)deg(\varphi) - 2L(\phi, \varphi)). \end{aligned}$$

Since the degree is positive definite, $deg(m\phi - n\varphi) \geq 0$ and $deg(\phi) \geq 0$, so

$$\begin{aligned} L(\phi, \varphi)^2 + 4deg(\phi)deg(\varphi) - 2L(\phi, \varphi) &\geq 0 \\ 4deg(\phi)deg(\varphi) - L(\phi, \varphi)^2 &\geq 0 \\ 4deg(\phi)deg(\varphi) &\geq L(\phi, \varphi)^2 \\ 2\sqrt{deg(\phi)deg(\varphi)} &\geq |L(\phi, \varphi)| \end{aligned}$$

□

Finally, change ϕ and φ to φ and I respectively, we have $|deg(\varphi - I) - p - 1| \geq 2\sqrt{p}$, and that ends the proof of Hasse's theorem.

4.2. **Key exchange.**

Recall that the Diffie-Hellman key exchange method is based on the difficulty to solve the discrete log problem in a finite field. However, there are algorithms such as Pohlig-Hellman to solve the discrete log problem for small fields. Thus, to make the system safe, the field must be large enough. But that would take more computation time. We will see an analog of the Diffie-Hellman key exchange based on elliptic curves that improves this situation.

Suppose Alice and Bob want to agree upon a key. Given an elliptic curve E defined over a finite field $\mathbb{F}_q$ (both E and $\mathbb{F}_q$ are known), they publicly choose a point B on E which serves as the base. First, Alice secretly chooses a random integer x, computes xB and publishes the value of xB. Similarly, Bob secretly chooses a random integer y and publishes the value of yB. Since xB and yB are public and both Alice and Bob know their own keys x and y, both of them can compute xyB, which will be their secret key. An outsider only knows xB and yB; to find xyB, one needs to find x knowing only the values of xB and B. Although there are algorithms to compute a multiple of a point on E, it's hard to reverse the process. The elliptic curve cryptosystem is considered safer than the Diffie-Hellman exchange, because there are no efficient algorithms for solving the analogous "discrete log problem" on elliptic curves.

4.3. **Lenstra's factorization algorithm using elliptic curves.** Lenstra's algorithm is a factorization method using the addition rules of elliptic cuves. First we will do an example. Let the elliptic curve be given by the equation $y^2 = x^3 + 4x + 9$ and $P = (2, 5)$ be a point on the curve. Take 589, which is not a prime, so $\mathbb{Z}/589\mathbb{Z}$ is not a field. We want to compute $2P$ mod 589.

Using the equations for addition, the x-coordinate of $2P$ is equal to $(\frac{3\times 2^2+4}{2\times 5})^2 - 2\times 2 = (\frac{8}{5})^2 - 4$. To write the result modulo 589, we need to find the inverse of 5 mod 589 using the Euclidean algorithm. Since $g.c.d(5, 589) = 1$, the inverse exists.

$$589 = 5 \times 117 + 4;$$
$$5 = 4 \times 1 + 1.$$
$$1 = 5 - 4 \times 1 = 5 - (589 - 5 \times 117) = 5 \times 118 - 589.$$

So the inverse of 5 is 118. The x-coordinate of $2P$ is $(\frac{8}{5})^2 - 4 = (8\times 118)^2 - 4 = 564$. Then the y-coordinate is equal to $(\frac{3\times 2^2+4}{2\times 5})(2 - 564) - 5 = 156$. So $2P = (564, 156)$.

Next, compute $3P$ mod 589, which is equal to $2P + P$. Again, we use the equations for addition. The x-coordinate of $3P$ is $(\frac{156-5}{564-2})^2 - 2 - 564 = (\frac{151}{562})^2 - 566$. Since $g.c.d(562, 589) = 1$, we can find the inverse of 562 mod 589 by the Euclidean algorithm, which is 349. Then the x-coordinate is equal to $(151\times 349)^2 - 566 = 148$, and the y-coordinate is equal to $(\frac{156-5}{564-2})(564 - 148) - 156 = 278 \times 416 - 156 = 48$. So $3P = (148, 48)$.

Similarly, we compute $4P = (303, 572)$. Then when we want to compute $7P$ by adding $3P$ and $4P$, a problem occurs. According to the equation for $7P$, $303 - 148 = 155$ appears in the denominator, which is not coprime to 589, so we cannot find its inverse. However, this step automatically provides us a nontrivial factor of 589: $g.c.d(155, 589) = 31$. Then we can factor 589 as 31×19.

Of course, when we perform the above computation, we do not know which multiple of P we need to compute so that we can encounter a denominator which is not coprime to the composite number n we want to factor. In the example above, we luckily find $7P$ because 7 is small. Therefore, when we leave this task to a computer, we have to specify what partial sums the computer will compute by choosing some boundaries. To see how we choose the boundary, let's examine the above example more closely.

Let's consider $y^2 = x^3 + 4x + 9$ defined over the field $\mathbb{F}_{31}$. We can compute that $3P = (24, 17)$ and $4P = (24, 14)$, so $7P = 3P + 4P = O$. There is a proposition that generalizes this result.

Proposition 4.16. *Let E be an elliptic curve given by $y^2 = x^3 + ax + b$, and P_1, P_2 be two points on E that are not inverses of each other. If there exists a prime p dividing n such that $(P_1 \text{ mod } p) + (P_2 \text{ mod } p) = O \text{ mod } p$, then $(P_1 + P_2) \text{ mod } n$ does not exist, i.e. $(P_1 + P_2)$ will have a denominator not coprime to n so that its inverse mod n does not exist.*

Proof. Let P_1 mod $p = (x_1, y_1)$ and P_2 mod $p = (x_2, y_2)$. Since p is a prime, the points $E(\mathbb{F}_p)$ form a group. Since $(P_1 \text{ mod } p) + (P_2 \text{ mod } p) = O$ mod p and inverses are unique in a group, P_1 mod p must be $-P_2$ mod p, i.e. $x_1 \equiv x_2$ mod p and $y_1 \equiv -y_2$ mod p.

Let $P_1 = (x_1', y_1')$ and $P_2 = (x_2', y_2')$ where the coordinates are rational numbers. Then $p \mid x_1' - x_1$ and $x_2' - x_2$, i.e. they are fractions with numerators divisible by p written in lowest terms. It follows that $x_1' - x_2'$ has a numerator divisible by p. When we compute $P_1 + P_2$, we get a denominator which is not coprime to n, since p is a common factor. $\square$

Therefore, as long as we have mP mod $p = O$ mod p where p divides n, we get a nontrivial factor of n. Let M be the bound of all prime divisors of this multiple m. It is more likely to get mP mod $p = O$ mod p for larger M, but the computation will take more time. So we also need some limit on the running time, M'. For each prime $p < M$, let α_p be the largest exponent such that $p^{\alpha_p} \leq M'$.

Now we describe the algorithm. First, we randomly generate an elliptic curve E given by $y^2 = x^3 + ax + b$ and a point $P = (x, y)$ on E. Let n be the composite number we want to factor. Thus, we need to check that $4a^3+27b^2$ and n are coprime. Then we choose boundaries M and M' according to the principles described before. Compute $2P$, $4P$,..., $2^{\alpha_2}P$, then $3(2^{\alpha_2})P$, $9(2^{\alpha_2})P$,..., $3^{\alpha_3}(2^{\alpha_2})P$,...and eventually $q^{\alpha_q}P$, where q is the largest prime smaller than M. At any stage, if the addition fails, we get a nontrivial factor of n. If all computations succeed, then we have been very unlucky and must start with another elliptic curve or another point P.

Acknowledgments

It is my pleasure to thank my mentor, Andreea Iorga, who patiently helped me understand hard concepts and algorithms, find resources, and edit this paper. I would also like to thank Professor Daniil Rudenko and Professor Peter May for organizing an instructive program, and Professor Slava Naprienko from Stanford University for discussing advanced topics on Cryptography with me. Lastly, I would like to thank my friends Daohong Xu, Yafei Li, Doctor Ying Xu and my mother for their support and enlightenment as I struggled with math and life.

References

[1] Neal Koblitz. *A Course in Number Theory and Cryptography*, second edition. Springer-Verlag.
[2] Jeffery Hoffstein, Jill Pipher, Joseph H. Silverman. *An Introduction to Mathematical Cryptography.* Springer.
[3] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, second edition. Springer.
[4] Kristen Hendricks. *On the Proof of Hasse's Theorem.*
[5] Eric Landquist. *The Quadratic Sieve Factoring Algorithm.*
[6] Evan Dummit. Northwestern Mathematics, Lecture Notes: Mathematical Cryptography, Spring 2016. URL: https://web.northeastern.edu/dummit/handouts.html.
[7] Adi Shamir. *A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem.*