15.8 Applications of elliptic curves in cryptography

Let G be a finite Abelian group. To be suitable for applications in public-key cryptography, the group G should be such that operations of multiplication and exponentiation in G are simple while taking the logarithm (the inverse operation of exponentiation) is very difficult. It should also be possible to generate random elements of the group in an almost uniform manner. However, the main question is how difficult is the so-called *discrete logarithm problem* in the group G.

> **The discrete logarithm problem:** Let $(G, *)$ be a finite group, $g \in G$, $H = \{g^i : i \geq 0\}$ the subgroup of G generated by g, and $h \in H$. The problem requires to find the smallest non-negative integer x such that $h = g^x$, where $g^x = \underbrace{g * g * ... * g}_{x \text{ times}}$.
>
> That number x is called the *discrete logarithm* and it is denoted by $\log_g h$.

The fact that there are groups in which the discrete logarithm problem is difficult was used by Diffie and Hellman in their solution of the problem of key exchange.

Let us assume that Alice and Bob would like to make an agreement about a secret random element in the group G, which they could, later on, use as the key in some symmetric cryptosystem. They have to manage their agreement through an insecure communication channel without previously exchanging any information. The only information which they have is the publicly available information about the group G and its generator g (let us assume for simplicity that the group G is cyclic).

We will describe the Diffie-Hellman protocol. We denote by $|G|$ the number of elements in a group G.

> **Diffie-Hellman key exchange protocol:**
>
> 1. Alice generates a random integer $a \in \{1, 2, \ldots, |G|-1\}$ and sends the element g^a to Bob.
> 2. Bob generates a random integer $b \in \{1, 2, \ldots, |G| - 1\}$ and sends the element g^b to Alice.
> 3. Alice calculates $(g^b)^a = g^{ab}$.
> 4. Bob calculates $(g^a)^b = g^{ab}$.
>
> Now, their secret key is $K = g^{ab}$.

Their opponent (Eve), which can spy on their communication through the insecure communication channel, knows the following data: G, g, g^a, g^b. Eve needs to calculate g^{ab} from these data (it is said that Eve needs to solve the *Diffie-Hellman problem* (DHP)). If Eve can calculate a from g and g^a (i.e. if she can solve the discrete logarithm problem (DLP)), then she can also, by using a and g^b, calculate g^{ab}. It is believed that these two problems, DHP and DLP, are equivalent for most of the groups that are used in cryptography (i.e. that there are polynomial time algorithms which reduce one problem to the other).

In the original definition of the Diffie-Hellman protocol, for the group G, it is taken the multiplicative group $\mathbb{F}_p^*$ of the finite field $\mathbb{F}_p$, where p is a large enough prime number, while g is a primitive root modulo p (here, we have $\log_g h = \operatorname{ind}_g h$, in the notation of Chapter 3.7).

We will now describe the *ElGamal cryptosystem* from 1985, which is based on the difficulty of computing the discrete logarithm in the group $(\mathbb{F}_p^*, \cdot_p)$.

It is considered that this problem is approximately of the same difficulty as the problem of factorization of a composite number n (if p and n are of the same size), and some of the methods which are used in the best known algorithms for solving those problems are very similar.

ElGamal cryptosystem: Let p be a prime number and $\alpha \in \mathbb{F}_p^*$ a primitive root modulo p. Let $\mathcal{P} = \mathbb{F}_p^*$, $\mathcal{C} = \mathbb{F}_p^* \times \mathbb{F}_p^*$ and

$$\mathcal{K} = \{(p, \alpha, a, \beta) \,:\, \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α, β are public and the value a is secret.

For $K \in \mathcal{K}$ and a (secret) random number $k \in \{0, 1, \ldots, p-1\}$ (the "ephemeral key"), we define

$$e_K(x, k) = (\alpha^k \bmod p,\ x\beta^k \bmod p).$$

For $y_1, y_2 \in \mathbb{Z}_p^*$, we define

$$d_K(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p.$$

We could say that the plaintext x is "masked" by multiplication by β^k. The one who knows the secret exponent a, can calculate β^k from α^k and "remove the mask".

For the exponent a to be really secret, a prime number p has to be large enough so that in $\mathbb{F}_p^*$, the discrete logarithm problem would practically be unsolvable. Therefore, nowadays it is recommended to use prime numbers of at least 1024 bits. Also, due to the reasons we will explain later, the order of the group, i.e. the number $p-1$, should have at least one large prime factor (of at least 160 bits).

The group $\mathbb{F}_p^*$ is not the only group for which exponentiation is much simpler than taking the logarithm. Indeed, there are groups, such as the group of an elliptic curve over a finite field, for which the difference in the complexities of these two problems (exponentiation and taking the logarithm) is even bigger.

The idea that elliptic curves might be useful in constructing public-key cryptosystems was first publicly stated independently by Koblitz and Miller in 1985. We mention [48, 174, 210, 250, 415] as books that deal with the applications of elliptic curves in cryptography.

All cryptosystems which in their original definition use the group $\mathbb{F}_p^*$, such as the ElGamal cryptosystem, can be easily modified so that they use the group $E(\mathbb{F}_p)$ instead. However, the literal translation of the ElGamal cryptosystem into elliptic curves has a few downsides.

The first one is that before encrypting, we need to transfer the elements of the plaintext into the points on an elliptic curve. There is no satisfying deterministic algorithm for that issue. However, there is a probabilistic algorithm, which uses the fact that squares in a finite field represent 50 % of all elements. This means that we can expect that from k tries, we will find a number x such that x^3+ax+b is a square in $\mathbb{F}_p$, with probability $1-\frac{1}{2^k}$. For $k=30$ that is a satisfactory probability.

The second problem is that, in this variant of the ElGamal cryptosystem, the ciphertext of one element of the plaintext is an ordered pair of points on an elliptic curve. This means that, during the encrypting, the message becomes approximately four times longer.

We will describe one variant of the ElGamal cryptosystem which uses elliptic curves. It is called the *Menezes-Vanstone cryptosystem*. It uses elliptic curves only for the "masking", while plaintexts and ciphertexts are arbitrary ordered pairs of elements of the field (and not necessarily pairs which correspond to the points on the elliptic curve). In this cryptosystem, the encrypted message is (only) two time longer than the original message.

Menezes-Vanstone cryptosystem: Let E be an elliptic curve over $\mathbb{F}_p$ ($p > 3$ prime), $\alpha \in E(\mathbb{F}_p)$ and H the cyclic subgroup of E generated by α. Let $\mathcal{P} = \mathbb{F}_p^* \times \mathbb{F}_p^*$, $\mathcal{C} = E \times \mathbb{F}_p^* \times \mathbb{F}_p^*$ and

$$\mathcal{K} = \{(E, \alpha, a, \beta) \,:\, \beta = a\alpha\},$$

where $a\alpha$ denotes $\alpha + \alpha + \cdots + \alpha$ (a times), and $+$ is the addition of points on the elliptic curve.

The values E, α, β are public and the value a is secret.

For $K \in \mathcal{K}$, a secret random number $k \in \{0, 1, \ldots, |H| - 1\}$ (the ephemeral key) and $x = (x_1, x_2) \in \mathbb{F}_p^* \times \mathbb{F}_p^*$, we define

$$e_K(x, k) = (y_0, y_1, y_2),$$

where $y_0 = k\alpha$, $(c_1, c_2) = k\beta$, $y_1 = c_1 x_1 \bmod p$, $y_2 = c_2 x_2 \bmod p$.

For the ciphertext $y = (y_0, y_1, y_2)$, we define

$$d_K(y) = (y_1 c_1^{-1} \bmod p,\ y_2 c_2^{-1} \bmod p),$$

where $ay_0 = (c_1, c_2)$.

As it was already mentioned, the main reason for introducing elliptic curves in public-key cryptography is the fact that the discrete logarithm problem in the group $E(\mathbb{F}_p)$ is more difficult than the discrete logarithm problem in the group $\mathbb{F}_p^*$. This means that the same level of security can be achieved with a smaller key. So, for example, instead of 1024-bit keys, the 160-bit keys are sufficient. This is especially important in applications (such as, e.g. "smart cards") in which the storage space for keys is very limited.

The most efficient known algorithms for the discrete logarithm problem in the group $\mathbb{F}_p^*$ are based on the so-called *index calculus method*. The method itself can be defined for an arbitrary group G, but its efficiency depends on the properties of that group. First of all, we have to be able to choose a relatively small subset $\mathcal{B}$ of the group G (so-called *factor basis*) which has the property that a large number of elements of G can be efficiently expressed as a product of elements of $\mathcal{B}$.

For the efficiency of this method in the group $\mathbb{F}_p^*$, the properties of the distribution of primes are crucial, and first of all, the fact that there are infinitely many of them. More precisely, the number of primes which are less than a real number x is asymptotically equal to $\frac{x}{\ln x}$. We will see that the hardness of the problems of finding elliptic curves of large rank is the most

important limiting factor for the application of this method on groups of elliptic curves over finite fields. This was, in fact, the motivation for introducing elliptic curves into cryptography.

We will now describe the algorithm which for a cyclic group G of order n with a generator g calculates the discrete logarithm $\log_g h$ of an arbitrary element h of the group G. The name of the method originates from the *index*, which is an alternative term for the discrete logarithm.

Algorithm index calculus:

1. *The choice of a factor basis*: We choose a subset $\mathcal{B} = \{p_1, \ldots, p_m\}$ of G such that a relatively large number of elements of G can be expressed as a product of elements of $\mathcal{B}$.
2. *Linear relations in logarithms*: For a random number k, $0 \leq k \leq n-1$, we calculate g^k and try to express it as a product of elements of $\mathcal{B}$

$$g^k = \prod_{i=1}^{m} p_i^{c_i}, \quad c_i \geq 0.$$

   If we managed to do so, we take the logarithms of both sides in the obtained relation and write $k \bmod n$ as a linear combination of the logarithms

$$k \equiv \sum_{i=1}^{m} c_i \log_g p_i \pmod{n}.$$

   We repeat this process until we obtain at least m such independent relations.
3. *Solving the system*: We solve the linear system of m equations with m variables, and in this manner, we obtain the values $\log_g p_i$.
4. *Calculating* $x = \log_g h$: For a random number k, $0 \leq k \leq n-1$, we calculate $h \cdot g^k$ and try to express it as a product of elements of $\mathcal{B}$

$$h \cdot g^k = \prod_{i=1}^{m} p_i^{d_i}, \quad d_i \geq 0.$$

If we did not manage to do that, we choose a new k, and if we were successful, we take the logarithms of both sides of the obtained relation, and we get

$$x = \log_g h = \left(\sum_{i=1}^{m} d_i \log_g p_i - k \right) \bmod n.$$

For applying the index calculus algorithm to the group $\mathbb{F}_p^*$, which is a cyclic group of order $n = p - 1$, we take the first m prime numbers as the factor basis $\mathcal{B}$. Then we try to express numbers of the form $r = g^k \bmod p$ as products of powers of the first m prime numbers. It is clear that the larger m we choose, the greater is the likelihood that we will be able to factorize r as a product of powers of the first m prime numbers. On the other hand, the larger m also means that solving the system in the third step of the algorithm will be more demanding. It can be shown that the optimal choice for the largest element p_m of the factor basis is approximately

$$L(p) = e^{\sqrt{\ln p \ln \ln p}}.$$

In this way, the index calculus algorithm becomes a subexponential time algorithm for calculating the discrete logarithm in the group $\mathbb{F}_p^*$.

Let us now say something about the known algorithms for solving the discrete logarithm problem in the group of an elliptic curve over a finite field (ECDLP).

We will first describe the *Pohlig-Hellman algorithm* which is based on the fact that calculating the discrete algorithm m, reduces to determining the value of m modulo each prime factor of n (the order of the considered group). The direct consequence of the existence of this algorithm is that, if we want a cryptosystem based on ECDLP to be is secure, then n needs to have at least one large prime factor.

The Pohlig-Hellman algorithm can be applied to any Abelian group G. Let the order n of G be divisible by a prime number p, and suppose that we want to solve the discrete logarithm problem $Q = mP$. Let $n' = n/p$, $m \equiv m_0 \pmod p$, and $Q' = n'Q$, $P' = n'P$. Then P' is a point of order p, so $mP' = m_0P'$. Now, the discrete logarithm problem $Q = mP$ in G is reduced to the problem

$$Q' = n'Q = n'mP = m_0P'$$

in a subgroup of G of order p. By solving the new problem, we determine the value m_0, i.e. we determine m modulo p.

The values of m modulo $p^2, p^3, \ldots, p^c$ (where p^c is the largest power of p dividing n) are determined in the following way. Assume that we know that $m \equiv m_{i-1} \pmod{p^i}$. Then $m = m_{i-1} + kp^i$, for an integer k. Thus, we obtain the problem

$$R = Q - m_{i-1}P = k(p^i P) = kS,$$

where R and S are known, and S has the order $s = n/p^i$. The value of $k_{i-1} = k \bmod p$ is determined in the same way as we determined above the value of m modulo p, so we obtain $m \equiv m_i \pmod{p^{i+1}}$, where $m_i = m_{i-1} + k_{i-1}p^i$.

Continuing this process and solving the discrete logarithm problem in subgroups of order p, at the end, we determine the value of m modulo p^c. After calculating this value for all prime divisors of n, the number m, i.e. the solution of the original discrete logarithm problem, is found by applying the Chinese remainder theorem.

Several methods for solving ECDLP (and general DLP in an arbitrary Abelian group of order n), which have the complexity $O(\sqrt{n})$, are known. These are *Pollard's ρ-method* and *Pollard's λ-method* as well as the Shanks *baby step – giant step method* (BSGS), which was already described in the previous section as a part of the Mestre-Shanks method for calculating the order of the group $|E(\mathbb{F}_p)|$. The idea of Pollard's methods is to use "random" mapping for finding equations of the form

$$m_1P + n_1Q = m_2P + n_2Q,$$

which yields that $Q = (m_2 - m_1)(n_1 - n_2)^{-1}P$, under the assumption that there is an inverse of $n_1 - n_2$ modulo n (for details, see [379, Chapter 13]). The estimate $O(\sqrt{n})$ comes from the "birthday paradox": if we randomly choose elements of a set that consists of n elements, the first repetition (with the probability of at least 50 %) can be expected after approximately $\sqrt{n}$ choices (the original birthday paradox states that in a group of at least 23 people there are at least two persons who have birthdays on the same day with the probability greater than 50 %).

Let us mention that for elliptic curves of certain special forms there are more efficient algorithms for the ECDLP, than the ones stated above. Knowing those algorithms is important because they indicate which elliptic curves should be avoided in cryptographic applications that use the ECDLP.

We will now list the types of elliptic curves which should be avoided (for details, see [48]):

- The Pohlig-Hellman algorithm implies that we should avoid elliptic curves for which the order of the group $E(\mathbb{F}_q)$ does not have a large prime factor. More precisely, $|E(\mathbb{F}_q)|$ should have at least one prime factor n greater than 2^{160} because otherwise, we could solve the ECDLP by Pollard's method. Namely, it is considered that 2^{80} is the borderline number of operations which nowadays guarantees satisfactory security (let us mention that in 1998, the DES cryptosystem was broken by a "brute force" attack, i.e. by testing all 2^{56} possible keys (see [147, Chapter 2.3], [389, Chapter 3.5.2])). Usually, the curve E is chosen such that the number $|E(\mathbb{F}_q)|$ is of the form $h \cdot r$, where r is a prime number and $h = 1, 2$ or 4.

- An elliptic curve is called *anomalous* if its trace of Frobenius $t = q + 1 - |E(\mathbb{F}_q)|$ is equal to 1, i.e. if $|E(\mathbb{F}_q)| = q$. For such curves, there is a polynomial time algorithm for the ECDLP which was discovered by Smart, Satoh, Araki and Semaev. Therefore, anomalous curves should not be used in this context.

- We say that an elliptic curve E over $\mathbb{F}_q$, where $q = p^k$, is *supersingular* if p divides t. For curves over $\mathbb{F}_p$ for $p \geq 5$ this means that $t = 0$, i.e. $|E(\mathbb{F}_p)| = p + 1$. For such curves, there is the *MOV-attack* (Menezes, Okamoto, Vanstone), which in polynomial time, reduces the ECDLP in the group $E(\mathbb{F}_q)$ to the (ordinary) DLP in the field $\mathbb{F}_{q^2}$. Therefore, supersingular curves should also be avoided in this context. Furthermore, all curves for which there is a small positive integer k (for instance $k \leq 20$) such that $q^k \equiv 1 \pmod{|E(\mathbb{F}_q)|}$ should be avoided because in that case, the MOV-attack reduces the ECDLP into the DLP in the field $\mathbb{F}_{q^k}$.

We see that it is easy to decide whether the ECDLP on a specific elliptic curve is difficult enough for applications in cryptography if we know the order of the group $E(\mathbb{F}_q)$.

The primary motivation for using elliptic curves in cryptography originates in the nonexistence of a subexponential time algorithm for solving the discrete logarithm problem for elliptic curves, while the discrete logarithm problem in the multiplicative group of a finite field can be solved in subexponential time by the index calculus method. The main reasons why the index calculus method is not applicable to elliptic curves lie in the facts that:

- it is difficult to find an elliptic curve over $\mathbb{Q}$ with large rank;
- it is difficult to find an elliptic curve with many independent points with small numerators and denominators;
- it is difficult to "lift" the points in $E(\mathbb{F}_p)$ to the points in $E(\mathbb{Q})$.

If it were possible to solve these difficult problems, then we would be able to apply the analogue of the index calculus method in which we would replace the set of prime numbers by generators of an elliptic curve over $\mathbb{Q}$ of large rank. Let us mention that it is estimated that for applying this idea for p approximately equal to 2^{160}, we should use a curve of rank greater than 180 (see [374]). Since no curve of rank greater than 28 is known, it is clear that this idea is not realistic.

Because of that, we can expect that with cryptosystems based on elliptic curves, we will achieve satisfactory security with a shorter key than with cryptosystems based on the factorization or the ordinary discrete logarithm problem. So, for achieving the same level of security as with the RSA cryptosystem (the same holds for the ElGamal cryptosystem) with a 1024 bit key (which is nowadays the standard value), with elliptic curves it is sufficient to take a 160-bit key. We conclude that cryptosystems based on elliptic curves provide the same level of security with seven times shorter key length as the RSA cryptosystem and it can be expected that the ratio will be even larger in the future.

A serious potential threat for the security of all cryptosystems based on the discrete logarithm problem (and also on the factorization) is connected to the question what could happen with their security if efficient quantum computers would be constructed. Unlike the classical computers, for which the basic unit of information is one bit (which can be 0 or 1), quantum computers use ideas from the quantum mechanic and for them, the basic unit of information, a *qubit*, could carry a lot more information. Such computers are still not realized in practice in a form which would come close to the classical computers, but such realization is not excluded in the foreseeable future. Therefore, in the last 20 years, there has also been progress on the algorithms designed specifically for such computers. One of the most famous algorithms for quantum computers is Shor's algorithm which uses the fact that quantum methods enable very fast calculation of the period of periodic functions. This gives polynomial quantum algorithms for the problems of factorization and discrete logarithm (see [422]).

One of the cryptosystems which might be secure even in the era of quantum computers (such cryptosystems are called "post-quantum") is NTRU,

which was proposed in 1997 by Hoffstein, Pipher and Silverman. This cryptosystem uses polynomials, more precisely, the ring $R = \mathbb{Z}[x]/\langle x^n - 1\rangle$ (of "n-truncated" polynomials). On the elements of R, the operation of cyclic convolution is defined. Then the reduction of such obtained polynomials over two relatively prime moduli p and q is used. Let us mention that there are some potential attacks to the NTRU cryptosystem that use the LLL algorithm for finding the smallest vector in the corresponding lattice (see [219, Chapter 6]).

There are also the post-quantum cryptographic algorithms based on the properties of isogenies of supersingular elliptic curves, out of which, one of the most interesting is the protocol for key exchange SIDH (supersingular isogeny Diffie-Hellman key exchange) (see [235]).

15.9 Primality proving using elliptic curves

If a positive integer n passes several good primality tests (e.g. the Miller-Rabin test from Chapter 3.9 with respect to a few different bases), then we can be quite confident that n is prime. However, those tests do not provide a *proof* that n is prime. Concerning the relevance of this problem for applications in cryptography, two different ways in which the need for large prime numbers occurs in cryptography should be distinguished. When choosing secret prime numbers p and q for each user in the RSA cryptosystem, we want to generate such numbers as fast as possible, and then it is satisfactory if there is a high probability that those numbers are prime. On the other hand, when choosing a finite field which will be used in the ElGamal cryptosystem, we are dealing with the prime number which might be recommended as a standard and used for a few years, so in this case, we want to be sure (to have a proof) that the number is indeed prime. Now, we will say something about the methods with which it can be proved that a given (large) positive integer is prime.

Theorem 15.21 (Pocklington, 1914)**.** *Let n be a positive integer and let s be a divisor of $n-1$ which is greater than $\sqrt{n}$. Assume that there is a positive integer a such that*

$$a^{n-1} \equiv 1 \pmod{n},$$

$$\gcd(a^{(n-1)/q} - 1, n) = 1, \quad \textit{for each prime divisor } q \textit{ of } s.$$

Then n is prime.