Chapter 9
Elliptic Curve Cryptosystems

Elliptic Curve Cryptography (ECC) is the newest member of the three families of established public-key algorithms of practical relevance introduced in Sect. 6.2.3. However, ECC has been around since the mid-1980s.

ECC provides the same level of security as RSA or discrete logarithm systems with considerably shorter operands (approximately 160–256 bit vs. 1024–3072 bit). ECC is based on the generalized discrete logarithm problem, and thus DL-protocols such as the Diffie–Hellman key exchange can also be realized using elliptic curves. In many cases, ECC has performance advantages (fewer computations) and bandwidth advantages (shorter signatures and keys) over RSA and Discrete Logarithm (DL) schemes. However, RSA operations which involve short public keys as introduced in Sect. 7.5.1 are still much faster than ECC operations.

The mathematics of elliptic curves are considerably more involved than those of RSA and DL schemes. Some topics, e.g., counting points on elliptic curves, go far beyond the scope of this book. Thus, the focus of this chapter is to explain the basics of ECC in a clear fashion without too much mathematical overhead, so that the reader gains an understanding of the most important functions of cryptosystems based on elliptic curves.

In this chapter, you will learn:

- The basic pros and cons of ECC vs. RSA and DL schemes.
- What an elliptic curve is and how to compute with it.
- How to build a DL problem with an elliptic curve.
- Protocols that can be realized with elliptic curves.
- Current security estimations of cryptosystems based on elliptic curves.

Christof Paar et al., *Understanding Cryptography*

© Springer-Verlag Berlin Heidelberg 2010

9.1 How to Compute with Elliptic Curves

We start by giving a short introduction to the mathematical concept of elliptic curves, independent of their cryptographic applications. ECC is based on the generalized discrete logarithm problem. Hence, what we try to do first is to find a cyclic group on which we can build our cryptosystem. Of course, the mere existence of a cyclic group is not sufficient. The DL problem in this group must also be computationally hard, which means that it must have good one-way properties.

We start by considering certain polynomials (e.g., functions with sums of exponents of x and y), and we plot them over the real numbers.

Example 9.1. Let's look at the polynomial equation $x^2 + y^2 = r^2$ over the real numbers $\mathbb{R}$. If we plot all the pairs (x,y) which fulfill this equation in a coordinate system, we obtain a circle as shown in Fig. 9.1.

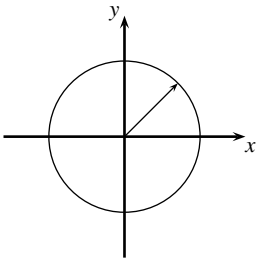

Fig. 9.1 Plot of all points (x,y) which fulfill the equation $x^2 + y^2 = r^2$ over $\mathbb{R}$

◇

We now look at other polynomial equations over the real numbers.

Example 9.2. A slight generalization of the circle equation is to introduce coefficients to the two terms x^2 and y^2, i.e., we look at the set of solutions to the equation $a \cdot x^2 + b \cdot y^2 = c$ over the real numbers. It turns out that we obtain an ellipse, as shown in Figure 9.2.

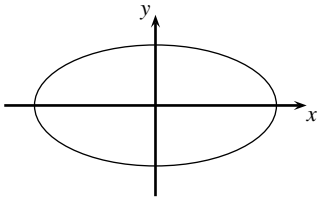

Fig. 9.2 Plot of all points (x,y) which fulfill the equation $a \cdot x^2 + b \cdot y^2 = c$ over $\mathbb{R}$

◇

9.1.1 Definition of Elliptic Curves

From the two examples above, we conclude that we can form certain types of curves from polynomial equations. By "curves", we mean the set of points (x,y) which are solutions of the equations. For example, the point $(x = r, y = 0)$ fulfills the equation of a circle and is, thus, in the set. The point $(x = r/2, y = r/2)$ is not a solution to the polynomial $x^2 + y^2 = r^2$ and is, thus, not a set member. An *elliptic curve* is a special type of polynomial equation. For cryptographic use, we need to consider the curve not over the real numbers but over a finite field. The most popular choice is prime fields $GF(p)$ (cf. Sect. 4.2), where all arithmetic is performed modulo a prime p.

Definition 9.1.1 Elliptic Curve
The elliptic curve *over* $\mathbb{Z}_p$, $p > 3$, *is the set of all pairs* $(x,y) \in \mathbb{Z}_p$ *which fulfill*

$$y^2 \equiv x^3 + a \cdot x + b \bmod p \tag{9.1}$$

together with an imaginary point of infinity $\mathcal{O}$, *where*

$$a, b \in \mathbb{Z}_p$$

and the condition $4 \cdot a^3 + 27 \cdot b^2 \neq 0 \bmod p$.

The definition of elliptic curve requires that the curve is nonsingular. Geometrically speaking, this means that the plot has no self-intersections or vertices, which is achieved if the discriminant of the curve $-16(4a^3 + 27b^2)$ is nonzero.

For cryptographic use we are interested in studying the curve over a prime field as in the definition. However, if we plot such an elliptic curve over $\mathbb{Z}_p$, we do not get anything remotely resembling a curve. However, nothing prevents us from taking an elliptic curve equation and plotting it over the set of real numbers.

Example 9.3. In Figure 9.3 the elliptic curve $y^2 = x^3 - 3x + 3$ is shown over the real numbers.

◇

We notice several things from this elliptic curve plot.[1] First, the elliptic curve is symmetric with respect to the x-axis. This follows directly from the fact that for all values x_i which are on the elliptic curve, both $y_i = \sqrt{x_i^3 + a \cdot x_i + b}$ and $y_i' = -\sqrt{x_i^3 + a \cdot x_i + b}$ are solutions. Second, there is one intersection with the x-axis. This follows from the fact that it is a cubic equation if we solve for $y = 0$ which has one real solution (the intersection with the x-axis) and two complex solutions (which do not show up in the plot). There are also elliptic curves with three intersections with the x-axis.

[1] Note that elliptic curves are not ellipses. They play a role in determining the circumference of ellipses, hence the name.

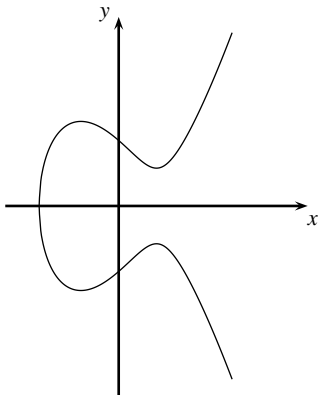

Fig. 9.3 $y^2 = x^3 - 3x + 3$ over $\mathbb{R}$

We now return to our original goal of finding a curve with a large cyclic group, which is needed for constructing a discrete logarithm problem. The first task for finding a group is done, namely identifying a set of elements. In the elliptic curve case, the group elements are the points that fulfill Eq. (9.1). The next question at hand is: How do we define a group operation with those points? Of course, we have to make sure that the group laws from Definition 4.3.1 in Sect. 4.2 hold for the operation.

9.1.2 Group Operations on Elliptic Curves

Let's denote the group operation with the addition symbol[2] "+". "Addition" means that given two points and their coordinates, say $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, we have to compute the coordinates of a third point R such that:

$$P + Q = R$$
$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

As we will see below, it turns out that this addition operation looks quite arbitrary. Luckily, there is a nice geometric interpretation of the addition operation if we consider a curve defined over the real numbers. For this geometric interpretation, we have to distinguish two cases: the addition of two distinct points (named point addition) and the addition of one point to itself (named point doubling).

Point Addition $\mathbf{P + Q}$ This is the case where we compute $R = P + Q$ and $P \neq Q$. The construction works as follows: Draw a line through P and Q and obtain a third point of intersection between the elliptic curve and the line. Mirror this third

[2] Note that the choice of naming the operation "addition" is completely arbitrary; we could have also called it multiplication.

intersection point along the x-axis. This mirrored point is, by definition, the point R. Figure 9.4 shows the point addition on an elliptic curve over the real numbers.

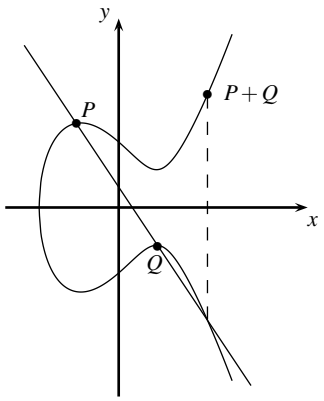

Fig. 9.4 Point addition on an elliptic curve over the real numbers

Point Doubling $\mathbf{P + P}$ This is the case where we compute $P + Q$ but $P = Q$. Hence, we can write $R = P + P = 2P$. We need a slightly different construction here. We draw the tangent line through P and obtain a second point of intersection between this line and the elliptic curve. We mirror the point of the second intersection along the x-axis. This mirrored point is the result R of the doubling. Figure 9.5 shows the

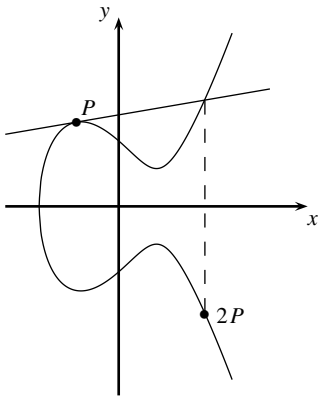

Fig. 9.5 Point doubling on an elliptic curve over the real numbers

doubling of a point on an elliptic curve over the real numbers.

You might wonder why the group operations have such an arbitrary looking form. Historically, this *tangent-and-chord* method was used to construct a third point if two points were already known, while only using the four standard algebraic operations add, subtract, multiply and divide. It turns out that if points on the elliptic

curve are *added* in this very way, the set of points also fulfill most conditions necessary for a group, that is, closure, associativity, existence of an identity element and existence of an inverse.

Of course, in a cryptosystem we cannot perform geometric constructions. However, by applying simple coordinate geometry, we can express both of the geometric constructions from above through analytic expressions, i.e., formulae. As stated above, these formulae only involve the four basic algebraic operations. These operations can be performed in any field, not only over the field of the real numbers (cf. Sect. 4.2). In particular, we can take the curve equation from above, but we now consider it over prime fields $GF(p)$ rather than over the real numbers. This yields the following analytical expressions for the group operation.

Elliptic Curve Point Addition and Point Doubling

$$x_3 = s^2 - x_1 - x_2 \mod p$$
$$y_3 = s(x_1 - x_3) - y_1 \mod p$$

where

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p & ; \text{if } P \neq Q \text{ (point addition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p & ; \text{if } P = Q \text{ (point doubling)} \end{cases}$$

Note that the parameter s is the slope of the line through P and Q in the case of point addition, or the slope of the tangent through P in the case of point doubling.

Even though we made major headway towards the establishment of a finite group, we are not there yet. One thing that is still missing is an identity (or neutral) element $\mathcal{O}$ such that:

$$P + \mathcal{O} = P$$

for all points P on the elliptic curve. It turns out that there isn't any point (x, y) that fulfills the condition. Instead we **define** an abstract *point at infinity* as the neutral element $\mathcal{O}$. This point at infinity can be visualized as a point that is located towards "plus" infinity along the y-axis or towards "minus" infinity along the y-axis.

According the group definition, we can now also define the inverse $-P$ of any group element P as:

$$P + (-P) = \mathcal{O}.$$

The question is how do we find $-P$? If we apply the tangent-and-chord method from above, it turns out that the inverse of the point $P = (x_p, y_p)$ is the point $-P = (x_p, -y_p)$, i.e., the point that is reflected along the x-axis. Figure 9.6 shows the point P together with its inverse. Note that finding the inverse of a point $P = (x_p, y_p)$ is now trivial. We simply take the negative of its y coordinate. In the case of elliptic curves over a prime field $GF(p)$ (the most interesting case in cryptography), this is easily achieved since $-y_p \equiv p - y_p \mod p$, hence

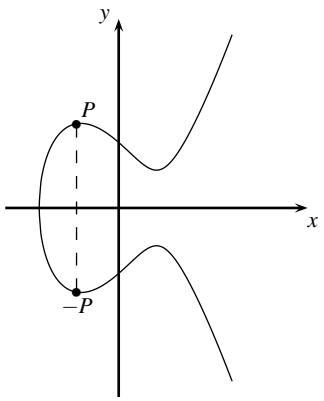

Fig. 9.6 The inverse of a point P on an elliptic curve

$$-P = (x_p, p - y_p).$$

Now that we have defined all group properties for elliptic curves, we now look at an example for the group operation.

Example 9.4. We consider a curve over the small field $\mathbb{Z}_{17}$:

$$E : y^2 \equiv x^3 + 2x + 2 \bmod 17.$$

We want to double the point $P = (5,1)$.

$$\begin{aligned}
2P &= P + P = (5,1) + (5,1) = (x_3, y_3) \\
s &= \frac{3x_1^2 + a}{2y_1} = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 \equiv 9 \cdot 9 \equiv 13 \bmod 17 \\
x_3 &= s^2 - x_1 - x_2 = 13^2 - 5 - 5 = 159 \equiv 6 \bmod 17 \\
y_3 &= s(x_1 - x_3) - y_1 = 13(5-6) - 1 = -14 \equiv 3 \bmod 17 \\
2P &= (5,1) + (5,1) = (6,3)
\end{aligned}$$

For illustrative purposes we check whether the result $2P = (6,3)$ is actually a point on the curve by inserting the coordinates into the curve equation:

$$\begin{aligned}
y^2 &\equiv x^3 + 2 \cdot x + 2 \bmod 17 \\
3^2 &\equiv 6^3 + 2 \cdot 6 + 2 \bmod 17 \\
9 &= 230 \equiv 9 \bmod 17
\end{aligned}$$

◇

9.2 Building a Discrete Logarithm Problem with Elliptic Curves

What we have done so far is to establish the group operations (point addition and doubling), we have provided an identity element, and we have shown a way of finding the inverse for any point on the curve. Thus, we now have all necessary requirements in place to motivate the following theorem:

> **Theorem 9.2.1** *The points on an elliptic curve together with $\mathcal{O}$ have cyclic subgroups. Under certain conditions* all *points on an elliptic curve form a cyclic group.*

Please note that we have not proved the theorem. This theorem is extremely useful because we have a good understanding of the properties of cyclic groups. In particular, we know that by definition a primitive element must exist such that its powers generate the entire group. Moreover, we know quite well how to build cryptosystems from cyclic groups. Here is an example for the cyclic group of an elliptic curve.

Example 9.5. We want to find all points on the curve:

$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \bmod 17.$$

It happens that all points on the curve form a cyclic group and that the order is $\#E = 19$. For this specific curve the group order is a prime and, according to Theorem 8.2.4, every element is primitive.

As in the previous example we start with the primitive element $P = (5,1)$. We compute now all "powers" of P. More precisely, since the group operation is addition, we compute $P, 2P, \ldots, (\#E)P$. Here is a list of the elements that we obtain:

$$\begin{aligned}
2P &= (5,1) + (5,1) = (6,3) & 11P &= (13,10)\\
3P &= 2P + P = (10,6) & 12P &= (0,11)\\
4P &= (3,1) & 13P &= (16,4)\\
5P &= (9,16) & 14P &= (9,1)\\
6P &= (16,13) & 15P &= (3,16)\\
7P &= (0,6) & 16P &= (10,11)\\
8P &= (13,7) & 17P &= (6,14)\\
9P &= (7,6) & 18P &= (5,16)\\
10P &= (7,11) & 19P &= \mathcal{O}
\end{aligned}$$

From now on, the cyclic structure becomes visible since:

$$\begin{aligned}
20P &= 19P + P = \mathcal{O} + P = P\\
21P &= 2P\\
&\vdots
\end{aligned}$$

It is also instructive to look at the last computation above, which yielded:

$$18P + P = \mathcal{O}.$$

This means that $P = (5,1)$ is the inverse of $18P = (5,16)$, and vice versa. This is easy to verify. We have to check whether the two x coordinates are identical and that the two y coordinates are each other's additive inverse modulo 17. The first condition obviously hold and the second one too, since

$$-1 \equiv 16 \bmod 17.$$

$\diamond$

To set up DL cryptosystems it is important to know the order of the group. Even though knowing the exact number of points on a curve is an elaborate task, we know the approximate number due to *Hasse's theorem*.

Theorem 9.2.2 Hasse's theorem
Given an elliptic curve E modulo p, the number of points on the curve is denoted by $\#E$ and is bounded by:

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}.$$

Hasse's theorem, which is also known as *Hasse's bound*, states that the number of points is roughly in the range of the prime p. This has major practical implications. For instance, if we need an elliptic curve with 2^{160} elements, we have to use a prime of length of about 160 bit.

Let's now turn our attention to the details of setting up the discrete logarithm problem. For this, we can strictly proceed as described in Chapter 8.

Definition 9.2.1 Elliptic Curved Discrete Logarithm Problem (ECDLP)
Given is an elliptic curve E. We consider a primitive element P and another element T. The DL problem is finding the integer d, where $1 \leq d \leq \#E$, such that:

$$\underbrace{P + P + \cdots + P}_{d \ \ times} = dP = T. \tag{9.2}$$

In cryptosystems, d is the private key which is an integer, while the public key T is a point on the curve with coordinates $T = (x_T, y_T)$. In contrast, in the case of the DL problem in $\mathbb{Z}_p^*$, both keys were integers. The operation in Eq. (9.2) is called *point multiplication*, since we can formally write $T = dP$. This terminology can be misleading, however, since we cannot directly multiply the integer d with a curve

point P. Instead, dP is merely a convenient notation for the repeated application of the group operation in Equation (9.2).[3] Let's now look at an example for an ECDLP.

Example 9.6. We perform a point multiplication on the curve $y^2 \equiv x^3 + 2x + 2 \bmod 17$ that was also used in the previous example. We want to compute

$$13P = P + P + \ldots + P$$

where $P = (5,1)$. In this case, we can simply use the table that was compiled earlier:

$$13P = (16,4).$$

$\diamond$

Point multiplication is analog to exponentiation in multiplicative groups. In order to do it efficiently, we can directly adopt the square-and-multiply algorithm. The only difference is that squaring becomes doubling and multiplication becomes addition of P. Here is the algorithm:

Double-and-Add Algorithm for Point Multiplication
Input: elliptic curve E together with an elliptic curve point P
a scalar $d = \sum_{i=0}^{t} d_i 2^i$ with $d_i \in 0,1$ and $d_t = 1$
Output: $T = dP$
Initialization:
$T = P$
Algorithm:
1 FOR $i = t-1$ DOWNTO 0
1.1 $T = T + T \bmod n$
IF $d_i = 1$
1.2 $T = T + P \bmod n$
2 RETURN (T)

For a random scalar of length of $t+1$ bit, the algorithm requires on average $1.5t$ point doubles and additions. Verbally expressed, the algorithm scans the bit representation of the scalar d from left to right. It performs a doubling in every iteration, and only if the current bit has the value 1 does it perform an addition of P. Let's look at an example.

Example 9.7. We consider the scalar multiplication $26P$, which has the following binary representation:

$$26P = (11010_2)P = (d_4 d_3 d_2 d_1 d_0)_2 P.$$

[3] Note that the symbol "+" was chosen arbitrarily to denote the group operation. If we had chosen a multiplicative notation instead, the ECDLP would have had the form $P^d = T$, which would have been more consistent with the conventional DL problem in $\mathbb{Z}_p^*$.

The algorithm scans the scalar bits starting on the left with d_4 and ending with the rightmost bit d_0.

Step		
#0	$P = \mathbf{1}_2 P$	inital setting, bit processed: $d_4 = 1$
#1*a*	$P + P = 2P = \mathbf{10}_2 P$	DOUBLE, bit processed: d_3
#1*b*	$2P + P = 3P = 10_2 P + 1_2 P = \mathbf{11}_2 P$	ADD, since $d_3 = 1$
#2*a*	$3P + 3P = 6P = 2(11_2 P) = \mathbf{110}_2 P$	DOUBLE, bit processed: d_2
#2*b*		no ADD, since $d_2 = 0$
#3*a*	$6P + 6P = 12P = 2(110_2 P) = \mathbf{1100}_2 P$	DOUBLE, bit processed: d_1
#3*b*	$12P + P = 13P = 1100_2 P + 1_2 P = \mathbf{1101}_2 P$	ADD, since $d_1 = 1$
#4*a*	$13P + 13P = 26P = 2(1101_2 P) = \mathbf{11010}_2 P$	DOUBLE, bit processed: d_0
#4*b*		no ADD, since $d_0 = 0$

It is instructive to observe how the binary representation of the exponent evolves. We see that doubling results in a left shift of the scalar, with a 0 put in the rightmost position. By performing addition with P, a 1 is inserted into the rightmost position of the scalar. Compare how the highlighted exponents change from iteration to iteration.

◇

If we go back to elliptic curves over the real numbers, there is a nice geometric interpretation for the ECDLP: given a starting point P, we compute $2P$, $3P$, …, $dP = T$, effectively hopping back and forth on the elliptic curve. We then publish the starting point P (a public parameter) and the final point T (the public key). In order to break the cryptosystem, an attacker has to figure out how often we "jumped" on the elliptic curve. The number of hops is the secret d, the private key.

9.3 Diffie–Hellman Key Exchange with Elliptic Curves

In complete analogy to the conventional Diffie–Hellman key exchange (DHKE) introduced in Sect. 8.1, we can now realize a key exchange using elliptic curves. This is referred to as elliptic curve Diffie–Hellman key exchange, or ECDH. First we have to agree on domain parameters, that is, a suitable elliptic curve over which we can work and a primitive element on this curve.

ECDH Domain Parameters

1. Choose a prime p and the elliptic curve

$$E : y^2 \equiv x^3 + a \cdot x + b \bmod p$$

2. Choose a primitive element $P = (x_P, y_P)$

The prime p, the curve given by its coefficients a, b, and the primitive element P are the domain parameters.

Note that in practice finding a suitable elliptic curve is a relatively difficult task. The curves have to show certain properties in order to be secure. More about this is said below. The actual key exchange is done the same way it was done for the conventional Diffie–Hellman protocol.

Elliptic Curve Diffie–Hellman Key Exchange (ECDH)

Alice		**Bob**
choose $k_{prA} = a \in \{2, 3, \ldots, \#E - 1\}$		choose $k_{prB} = b \in \{2, 3, \ldots, \#E - 1\}$
compute $k_{pubA} = aP = A = (x_A, y_A)$		compute $k_{pubB} = bP = B = (x_B, y_B)$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
compute $aB = T_{AB}$		compute $bA = T_{AB}$

Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$.

The correctness of the protocol is easy to prove.

Proof. Alice computes

$$aB = a(bP)$$

while Bob computes

$$bA = b(aP).$$

Since point addition is associative (remember that associativity is one of the group properties), both parties compute the same result, namely the point $T_{AB} = abP$. □

As can be seen in the protocol, Alice and Bob choose the private keys a and b, respectively, which are two large integers. With the private keys both generate their respective public keys A and B, which are points on the curve. The public keys are computed by point multiplication. The two parties exchange these public parameters with each other. The joint secret T_{AB} is then computed by both Alice and Bob by performing a second point multiplication involving the public key they received and their own secret parameter. The joint secret T_{AB} can be used to derive a session key, e.g., as input for the AES algorithm. Note that the two coordinates (x_{AB}, y_{AB}) are not independent of each other: Given x_{AB}, the other coordinate can be computed by simply inserting the x value in the elliptic curve equation. Thus, only one of the two coordinates should be used for the derivation of a session key. Let's look at an example with small numbers:

Example 9.8. We consider the ECDH with the following domain parameters. The elliptic curve is $y^2 \equiv x^3 + 2x + 2 \bmod 17$, which forms a cyclic group of order $\#E = 19$. The base point is $P = (5,1)$. The protocol proceeds as follows:

Alice		**Bob**
choose $a = k_{pr,A} = 3$		choose $b = k_{pr,B} = 10$
$A = k_{pub,A} = 3P = (10,6)$		$B = k_{pub,B} = 10P = (7,11)$
	$\xrightarrow{A}$	
	$\xleftarrow{B}$	
$T_{AB} = aB = 3(7,11) = (13,10)$		$T_{AB} = bA = 10(10,6) = (13,10)$

The two scalar multiplications that each Alice and Bob perform require the Double-and-Add algorithm.

◇

One of the coordinates of the joint secret T_{AB} can now be used as session key. In practice, often the x-coordinate is hashed and then used as a symmetric key. Typically, not all bits are needed. For instance, in a 160-bit ECC scheme, hashing the x-coordinate with SHA-1 results in a 160-bit output of which only 128 would be used as an AES key.

Please note that elliptic curves are not restricted to the DHKE. In fact, almost all other discrete logarithm protocols, in particular digital signatures and encryption, e.g., variants of Elgamal, can also be realized. The widely used elliptic curve digital signature algorithms (ECDSA) will be introduced in Sect. 10.5.1.

9.4 Security

The reason we use elliptic curves is that the ECDLP has very good one-way characteristics. If an attacker Oscar wants to break the ECDH, he has the following information: E, p, P, A, and B. He wants to compute the joint secret between Alice and Bob $T_{AB} = a \cdot b \cdot P$. This is called the elliptic curve Diffie–Hellman problem (ECDHP). There appears to be only one way to compute the ECDHP, namely to solve either of the discrete logarithm problems:

$$a = \log_P A$$

or

$$b = \log_P B$$

If the elliptic curve is chosen with care, the best known attacks against the ECDLP are considerably weaker than the best algorithms for solving the DL problem modulo p, and the best factoring algorithms which are used for RSA attacks. In particular, the index-calculus algorithms, which are powerful attacks against the DLP modulo p, are not applicable against elliptic curves. For carefully selected elliptic curves, the only remaining attacks are generic DL algorithms, that is Shanks' baby-step giant-step method and Pollard's rho method, which were described in Sect. 8.3.3. Since the number of steps required for such an attack is roughly equal

to the square root of the group cardinality, a group order of at least 2^{160} should be used. According to Hasse's theorem, this requires that the prime p used for the elliptic curve must be roughly 160-bit long. If we attack such a group with generic algorithms, we need around $\sqrt{2^{160}} = 2^{80}$ steps. A security level of 80 bit provides medium-term security. In practice, elliptic curve bit lengths up to 256 bit are commonly used, which provide security levels of up to 128 bit.

It should be stressed that this security is only achieved if cryptographically strong elliptic curves are used. There are several families of curves that possess cryptographic weaknesses, e.g., supersingular curves. They are relatively easy to spot, however. In practice, often standardized curves such as ones proposed by the National Institute of Standards and Technology (NIST) are being used.

9.5 Implementation in Software and Hardware

Before using ECC, a curve with good cryptographic properties needs to be identified. In practice, a core requirement is that the cyclic group (or subgroup) formed by the curve points has prime order. Moreover, certain mathematical properties that lead to cryptographic weaknesses must be ruled out. Since assuring all these properties is a nontrivial and computationally demanding task, often standardized curves are used in practice.

When implementing elliptic curves it is useful to view an ECC scheme as a structure with four layers. On the bottom layer modular arithmetic, i.e., arithmetic in the prime field $GF(p)$, is performed. We need all four field operations: addition, subtraction, multiplication and inversion. On the next layer, the two group operations, point doubling and point addition, are realized. They make use of the arithmetic provided in the bottom layer. On the third layer, scalar multiplication is realized, which uses the group operations of the previous layer. The top layer implements the actual protocol, e.g., ECDH or ECDSA. It is important to note that two entirely different finite algebraic structures are involved in an elliptic curve cryptosystem. There is a *finite field* $GF(p)$ over which the curve is defined, and there is the *cyclic group* which is formed by the points on the curve.

In software, a highly optimized 256-bit ECC implementation on a 3-GHz, 64-bit CPU can take approximately 2 ms for one point multiplication. Slower throughputs due to smaller microprocessors or less optimized algorithms are common with performances in the range of 10 ms. For high-performance applications, e.g., for Internet servers that have to perform a large number of elliptic curve signatures per second, hardware implementations are desirable. The fastest implementations can compute a point multiplication in the range of 40 μs, while speeds of several 100 μs are more common.

On the other side of the performance spectrum, ECC is the most attractive public-key algorithm for lightweight applications such as RFID tags. Highly compact ECC engines are possible which need as little as 10,000 gate equivalences and run at a speed of several tens of milliseconds. Even though ECC engines are much larger

than implementations of symmetric ciphers such as 3DES, they are considerably smaller than RSA implementations.

The computational complexity of ECC is cubic in the bit length of the prime used. This is due to the fact that modular multiplication, which is the main operation on the bottom layer, is quadratic in the bit length, and scalar multiplication (i.e., with the Double-and-Add algorithm) contributes another linear dimension, so that we have, in total, a cubic complexity. This implies that doubling the bit length of an ECC implementation results in performance degradation by a factor of roughly $2^3 = 8$. RSA and DL systems show the same cubic runtime behavior. The advantage of ECC over the other two popular public-key families is that the parameters have to be increased much more slowly to enhance the security level. For instance, doubling the effort of an attacker for a given ECC system requires an increase in the length of the parameter by 2 bits, whereas RSA or DL schemes require an increase of 20–30 bits. This behavior is due to the fact that only generic attacks (cf. Sect. 8.3.3) are known ECC cryptosystems, whereas more powerful algorithms are available for attacking RSA and DL schemes.

9.6 Discussion and Further Reading

History and General Remarks ECC was independently invented in 1987 by Neal Koblitz and in 1986 by Victor Miller. During the 1990s there was much speculation about the security and practicality of ECC, especially if compared to RSA. After a period of intensive research, they appear nowadays very secure, just like RSA and DL schemes. An important step for building confidence in ECC was the issuing of two ANSI banking standards for elliptic curve digital signature and key establishment in 1999 and 2001, respectively [6, 7]. Interestingly, in Suite B—a collection of crypto algorithms selected by the NSA for use in US government systems—only ECC schemes are allowed as asymmetric algorithms [130]. Elliptic curves are also widely used in commercial standards such as IPsec or Transport Layer Security (TLS).

At the time of writing, there still exist far more fielded RSA and DL applications than elliptic curve ones. This is mainly due to historical reasons and due to the quite complex patent situation of some ECC variants. Nevertheless, in many new applications with security needs, especially in embedded systems such as mobile devices, ECC is often the preferred public-key scheme. For instance, ECC is used in the most popular business handheld devices. Most likely, ECC will become more widespread in the years to come. Reference [100] describes the historical development of ECC with respect to scientific and commercial aspects, and makes excellent reading.

For readers interested in a deeper understanding of ECC, the books [25, 24, 90, 44] are recommended. The overview article [103], even though a bit dated now, provides a good state-of-the-art summary as of the year 2000. For more recent developments, the annual *Workshop on Elliptic Curve Cryptography (ECC)* is recommended as an excellent resource [166]. The workshop includes both theoretical and

applied topics related to ECC and related crypto schemes. There is also a rich literature that deals with the mathematics of elliptic curves [154, 101, 155], regardless of their use in cryptography.

Implementation and Variants In the first few years after the invention of ECC, these algorithms were believed to be computationally more complex than existing public-key schemes, especially RSA. This assumption is somewhat ironic in hindsight, given that ECC tends to be often faster than most other public-key schemes. During the 1990s, fast implementation techniques for ECC was intensively researched, which resulted in considerable performance improvements.

In this chapter, elliptic curves over prime fields $GF(p)$ were introduced. These are currently in practice somewhat more widely used than over other finite fields, but curves over binary Galois fields $GF(2^m)$ are also popular. For efficient implementations, improvements are possible at the finite field arithmetic layer, at the group operation layer and at the point multiplication layer. There is a wealth of techniques and in the following is a summary of the most common acceleration techniques in practice. For curves over $GF(p)$, generalized Mersenne primes are often used at the arithmetic level. These are primes such as $p = 2^{192} - 2^{64} - 1$. Their major advantage is that modulo reduction is extremely simple. If general primes are used, methods similar to those described in Sect. 7.10 are applicable. With respect to ECC over fields $GF(2^m)$, efficient arithmetic algorithms are described in [90]. On the group operation layer, several optimizations are possible. A popular one is to switch from the affine coordinates that were introduced here to projective coordinates, in which each point is represented as a triple (x, y, z). Their advantage is that no inversion is required within the group operation. The number of multiplications increases, however. On the next layer, fast scalar multiplication techniques are applicable. Improved versions of the Double-and-Add algorithm which make use of the fact that adding or subtracting a point come at almost identical costs are commonly being applied. An excellent compilation of efficient computation techniques for ECC is the book [90].

A special type of elliptic curve that allows for particularly fast point multiplication is the *Koblitz curve* [158]. These are curves over $GF(2^m)$ where the coefficients have the values 0 or 1. There have also been numerous other suggestions for elliptic curves with good implementation properties. One such proposal involves elliptic curves over optimum extension fields, i.e., fields of the form $GF(p^m)$, $p > 2$ [10].

As mentioned in Sect. 9.5, standardized curves are often being used in practice. A widely used set of curves is provided in the FIPS Standard [126, Appendix D]. Alternatives are curves specified by the ECC Brainpool consortium or the Standards for Efficient Cryptography Group (SECG) [34, 9] .

Elliptic curves also allow for many variants and generalization. They are a special case of hyperelliptic curves, which can also be used to build discrete logarithm cryptosystems [44]. A summary of implementation techniques for hyperelliptic curves is given in [175]. A completely different type of public-key scheme which also makes use of elliptic curves is identity-based cryptosystems [30], which have drawn much attention over the last few years.

9.7 Lessons Learned

- Elliptic Curve Cryptography (ECC) is based on the discrete logarithm problem. It requires arithmetic modulo a prime or in a Galois field $GF(2^m)$.
- ECC can be used for key exchange, for digital signatures and for encryption.
- ECC provides the same level of security as RSA or discrete logarithm systems over $\mathbb{Z}_p^*$ with considerably shorter operands (approximately 160–256 bit vs. 1024–3072 bit), which results in shorter ciphertexts and signatures.
- In many cases ECC has performance advantages over other public-key algorithms. However, signature verification with short RSA keys is still considerably faster than ECC.
- ECC is slowly gaining popularity in applications, compared to other public-key schemes, i.e., many new applications, especially on embedded platforms, make use of elliptic curve cryptography.

Problems

9.1. Show that the condition $4a^3 + 27b^2 \neq 0 \bmod p$ is fulfilled for the curve

$$y^2 \equiv x^3 + 2x + 2 \bmod 17 \tag{9.3}$$

9.2. Perform the additions

1. $(2,7) + (5,2)$
2. $(3,6) + (3,6)$

in the group of the curve $y^2 \equiv x^3 + 2x + 2 \bmod 17$. Use only a pocket calculator.

9.3. In this chapter the elliptic curve $y^2 \equiv x^3 + 2x + 2 \bmod 17$ is given with $\#E = 19$. Verify Hasse's theorem for this curve.

9.4. Let us again consider the elliptic curve $y^2 \equiv x^3 + 2x + 2 \bmod 17$. Why are *all* points primitive elements?

Note: In general it is not true that all elements of an elliptic curve are primitive.

9.5. Let E be an elliptic curve defined over $\mathbb{Z}_7$:

$$E : y^2 = x^3 + 3x + 2.$$

1. Compute all points on E over $\mathbb{Z}_7$.
2. What is the order of the group? (Hint: Do not miss the neutral element $\mathcal{O}$.)
3. Given the element $\alpha = (0,3)$, determine the order of α. Is α a primitive element?

9.6. In practice, a and k are both in the range $p \approx 2^{150} \cdots 2^{250}$, and computing $T = a \cdot P$ and $y_0 = k \cdot P$ is done using the Double-and-Add algorithm as shown in Sect. 9.2.

1. Illustrate how the algorithm works for $a = 19$ and for $a = 160$. Do *not* perform elliptic curve operations, but keep P a variable.
2. How many (i) point additions and (ii) point doublings are required on average for one "multiplication"? Assume that all integers have $n = \lceil \log_2 p \rceil$ bit.
3. Assume that all integers have $n = 160$ bit, i.e., p is a 160-bit prime. Assume one group operation (addition or doubling) requires 20 μsec. What is the time for one double-and-add operation?

9.7. Given an elliptic curve E over $\mathbb{Z}_{29}$ and the base point $P = (8,10)$:

$$E : y^2 = x^3 + 4x + 20 \bmod 29.$$

Calculate the following point multiplication $k \cdot P$ using the Double-and-Add algorithm. Provide the intermediate results after each step.

1. $k = 9$
2. $k = 20$

9.8. Given is the same curve as in 9.7. The order of this curve is known to be $\#E = 37$. Furthermore, an additional point $Q = 15 \cdot P = (14, 23)$ on this curve is given. Determine the result of the following point multiplications by using as few group operations as possible, i.e., make smart use of the known point Q. Specify *how* you simplified the calculation each time.

Hint: In addition to using Q, use the fact that it is easy to compute $-P$.

1. $16 \cdot P$
2. $38 \cdot P$
3. $53 \cdot P$
4. $14 \cdot P + 4 \cdot Q$
5. $23 \cdot P + 11 \cdot Q$

You should be able to perform the scalar multiplications with considerably fewer steps than a straightforward application of the double-and-add algorithm would allow.

9.9. Your task is to compute a session key in a DHKE protocol based on elliptic curves. Your private key is $a = 6$. You receive Bob's public key $B = (5, 9)$. The elliptic curve being used is defined by

$$y^2 \equiv x^3 + x + 6 \bmod 11.$$

9.10. An example for an elliptic curve DHKE is given in Sect. 9.3. Verify the two scalar multiplications that Alice performs. Show the intermediate results within the group operation.

9.11. After the DHKE, Alice and Bob possess a mutual secret point $R = (x, y)$. The modulus of the used elliptic curve is a 64-bit prime. Now, we want to derive a session key for a 128-bit block cipher. The session key is calculated as follows:

$$K_{\mathrm{AB}} = h(x||y)$$

Describe an *efficient* brute-force attack against the symmetric cipher. How many of the key bits are truly random in this case? (Hint: You do not need to describe the mathematical details. Provide a list of the necessary steps. Assume you have a function that computes square roots modulo p.)

9.12. Derive the formula for addition on elliptic curves. That is, given the coordinates for P and Q, find the coordinates for $R = (x_3, y_3)$.

Hint: First, find the equation of a line through the two points. Insert this equation in the elliptic curve equation. At some point you have to find the roots of a cubic polynomial $x^3 + a_2x^2 + a_1x + a_0$. If the three roots are denoted by x_0, x_1, x_2, you can use the fact that $x_0 + x_1 + x_2 = -a_2$.