Chapter 6

Elliptic Curve Cryptography

In this chapter, we'll discuss several cryptosystems based on elliptic curves, especially on the discrete logarithm problem for elliptic curves. We'll also treat various related ideas, such as digital signatures.

One might wonder why elliptic curves are used in cryptographic situations. The reason is that elliptic curves provide security equivalent to classical systems while using fewer bits. For example, it is estimated in [12] that a key size of 4096 bits for RSA gives the same level of security as 313 bits in an elliptic curve system. This means that implementations of elliptic curve cryptosystems require smaller chip size, less power consumption, etc. Daswani and Boneh [14] performed experiments using 3Com's PalmPilot, which is a small hand-held device that is larger than a smart card but smaller than a laptop computer. They found that generating a 512-bit RSA key took 3.4 minutes, while generating a 163-bit ECC-DSA key to 0.597 seconds. Though certain procedures, such as signature verifications, were slightly faster for RSA, the elliptic curve methods such as ECC-DSA clearly offer great increases in speed in many situations.

6.1 The Basic Setup

Alice wants to send a message, often called the **plaintext**, to Bob. In order to keep the eavesdropper Eve from reading the message, she encrypts it to obtain the **ciphertext**. When Bob receives the ciphertext, he decrypts it and reads the message. In order to encrypt the message, Alice uses an **encryption key**. Bob uses a **decryption key** to decrypt the ciphertext. Clearly, the decryption key must be kept secret from Eve.

There are two basic types of encryption. In **symmetric encryption**, the encryption key and decryption key are the same, or one can be easily deduced from the other. Popular symmetric encryption methods include the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES, often referred to by its original name *Rijndael*). In this case, Alice and Bob

need to have some way of establishing a key. For example, Bob could send a messenger to Alice several days in advance. Then, when it is time to send the message, they both will have the key. Clearly this is impractical in many situations.

The other type of encryption is **public key encryption**, or asymmetric encryption. In this case, Alice and Bob do not need to have prior contact. Bob publishes a public encryption key, which Alice uses. He also has a private decryption key that allows him to decrypt ciphertexts. Since everyone knows the encryption key, it should be infeasible to deduce the decryption key from the encryption key. The most famous public key system is known as RSA and is based on the difficulty of factoring integers into primes. Another well-known system is due to ElGamal and is based on the difficulty of the discrete logarithm problem.

Generally, public key systems are slower than good symmetric systems. Therefore, it is common to use a public key system to establish a key that is then used in a symmetric system. The improvement in speed is important when massive amounts of data are being transmitted.

6.2 Diffie-Hellman Key Exchange

Alice and Bob want to agree on a common key that they can use for exchanging data via a symmetric encryption scheme such as DES or AES. For example, Alice and Bob could be banks that want to transmit financial data. It is impractical and time-consuming to use a courier to deliver the key. Moreover, we assume that Alice and Bob have had no prior contact and therefore the only communication channels between them are public. One way to establish a secret key is the following method, due to Diffie and Hellman (actually, they used multiplicative groups of finite fields).

1. Alice and Bob agree on an elliptic curve E over a finite field $\mathbf{F}_q$ such that the discrete logarithm problem is hard in $E(\mathbf{F}_q)$. They also agree on a point $P \in E(\mathbf{F}_q)$ such that the subgroup generated by P has large order (usually, the curve and point are chosen so that the order is a large prime).

2. Alice chooses a secret integer a, computes $P_a = aP$, and sends P_a to Bob.

3. Bob chooses a secret integer b, computes $P_b = bP$, and sends P_b to Alice.

4. Alice computes $aP_b = abP$.

5. Bob computes $bP_a = baP$.

6. Alice and Bob use some publicly agreed on method to extract a key from abP. For example, they could use the last 256 bits of the x-coordinate of abP as the key. Or they could evaluate a hash function at the x-coordinate.

The only information that the eavesdropper Eve sees is the curve E, the finite field $\mathbf{F}_q$, and the points P, aP, and bP. She therefore needs to solve the following:

DIFFIE-HELLMAN PROBLEM
Given P, aP, and bP in $E(\mathbf{F}_q)$, compute abP.

If Eve can solve discrete logs in $E(\mathbf{F}_q)$, then she can use P and aP to find a. Then she can compute $a(bP)$ to get abP. However, it is not known whether there is some way to compute abP without first solving a discrete log problem.

A related question is the following:

DECISION DIFFIE-HELLMAN PROBLEM
Given P, aP, and bP in $E(\mathbf{F}_q)$, and given a point $Q \in E(\mathbf{F}_q)$ determine whether or not $Q = abP$.

In other words, if Eve receives an anonymous tip telling her abP, can she verify that the information is correct?

The Diffie-Hellman problem and the Decision Diffie-Hellman problem can be asked for arbitrary groups. Originally, they appeared in the context of multiplicative groups $\mathbf{F}_q^\times$ of finite fields.

For elliptic curves, the Weil pairing can be used to solve the Decision Diffie-Hellman problem in some cases. We give one such example.

Let E be the curve $y^2 = x^3 + 1$ over $\mathbf{F}_q$, where $q \equiv 2 \pmod 3$. By Proposition 4.33, E is supersingular. Let $\omega \in \mathbf{F}_{q^2}$ be a primitive third root of unity. Note that $\omega \notin \mathbf{F}_q$ since the order of $\mathbf{F}_q^\times$ is $q-1$, which is not a multiple of 3. Define a map

$$\beta : E(\overline{\mathbf{F}}_q) \to E(\overline{\mathbf{F}}_q), \quad (x, y) \mapsto (\omega x, y), \quad \beta(\infty) = \infty.$$

It is straightforward to show, using the formulas for the addition law, that β is an isomorphism (Exercise 6.1).

Suppose $P \in E(\overline{\mathbf{F}}_q)$ has order n. Then $\beta(P)$ also has order n. Define the modified Weil pairing

$$\tilde{e}_n(P_1, P_2) = e_n(P_1, \beta(P_2)),$$

where e_n is the usual Weil pairing and $P_1, P_2 \in E[n]$.

LEMMA 6.1
Assume $3 \nmid n$. If $P \in E(\mathbf{F}_q)$ has order exactly n, then $\tilde{e}_n(P,P)$ is a primitive nth root of unity.

PROOF Suppose $uP = v\beta(P)$ for some integers u, v. Then

$$\beta(vP) = v\beta(P) = uP \in E(\mathbf{F}_q).$$

If $vP = \infty$, then $uP = \infty$, so $u \equiv 0 \pmod{n}$. If $vP \neq \infty$, write $vP = (x,y)$ with $x, y \in \mathbf{F}_q$. Then

$$(\omega x, y) = \beta(vP) \in E(\mathbf{F}_q).$$

Since $\omega \notin \mathbf{F}_q$, we must have $x = 0$. Therefore $vP = (0, \pm 1)$, which has order 3. This is impossible since we have assumed that $3 \nmid n$. It follows that the only relation of the form $uP = v\beta(P)$ has $u, v \equiv 0 \pmod{n}$, so P and $\beta(P)$ form a basis of $E[n]$. By Corollary 3.10, $\tilde{e}_n(P,P) = e_n(P, \beta(P))$ is a primitive nth root of unity. ∎

Suppose now that we know P, aP, bP, Q and we want to decide whether or not $Q = abP$. First, use the usual Weil pairing to decide whether or not Q is a multiple of P. By Lemma 5.1, Q is a multiple of P if and only if $e_n(P,Q) = 1$. Assume this is the case, so $Q = tP$ for some t. We have

$$\tilde{e}_n(aP, bP) = \tilde{e}_n(P,P)^{ab} = \tilde{e}_n(P, abP) \quad \text{and} \quad \tilde{e}_n(Q,P) = \tilde{e}_n(P,P)^t.$$

Assume $3 \nmid n$. Then $\tilde{e}_n(P,P)$ is a primitive nth root of unity, so

$$Q = abP \iff t \equiv ab \pmod{n} \iff \tilde{e}_n(aP, bP) = \tilde{e}_n(Q,P).$$

This solves the Decision Diffie-Hellman problem in this case. Note that we did not need to compute any discrete logs, even in finite fields. All that was needed was to compute the Weil pairing.

The above method was pointed out by Joux and Nguyen. For more on the Decision Diffie-Hellman problem, see [13].

Joux [56] (see also [124]) has given another application of the modified Weil pairing to what is known as **tripartite Diffie-Hellman** key exchange. Suppose Alice, Bob, and Chris want to establish a common key. The standard Diffie-Hellman procedure requires two rounds of interaction. The modified Weil pairing allows this to be cut to one round. As above, let E be the curve $y^2 = x^3 + 1$ over $\mathbf{F}_q$, where $q \equiv 2 \pmod{3}$. Let P be a point of order n. Usually, n should be chosen to be a large prime. Alice, Bob, and Chris do the following.

1. Alice, Bob, and Chris choose secret integers $a, b, c \bmod n$, respectively.
2. Alice broadcasts aP, Bob broadcasts bP, and Chris broadcasts cP.

3. Alice computes $\tilde{e}_n(bP, cP)^a$, Bob computes $\tilde{e}_n(aP, cP)^b$, and Chris computes $\tilde{e}_n(aP, bP)^c$.

4. Since each of the three users has computed the same number, they use this number to produce a key, using some publicly prearranged method.

Recall that, since E is supersingular, the discrete log problem on E can be reduced to a discrete log problem for $\mathbf{F}_{q^2}^\times$ (see Section 5.3.1). Therefore, q should be chosen large enough that this discrete log problem is hard.

For more on cryptographic applications of pairings, see [57].

6.3 Massey-Omura Encryption

Alice wants to send a message to Bob over public channels. They have not yet established a private key. One way to do this is the following. Alice puts her message in a box and puts her lock on it. She sends the box to Bob. Bob puts his lock on it and sends it back to Alice. Alice then takes her lock off and sends the box back to Bob. Bob then removes his lock, opens the box, and reads the message.

This procedure can be implemented mathematically as follows.

1. Alice and Bob agree on an elliptic curve E over a finite field $\mathbf{F}_q$ such that the discrete log problem is hard in $E(\mathbf{F}_q)$. Let $N = \#E(\mathbf{F}_q)$.

2. Alice represents her message as a point $M \in E(\mathbf{F}_q)$. (We'll discuss how to do this below.)

3. Alice chooses a secret integer m_A with $\gcd(m_A, N) = 1$, computes $M_1 = m_A M$, and sends M_1 to Bob.

4. Bob chooses a secret integer m_B with $\gcd(m_B, N) = 1$, computes $M_2 = m_B M_1$, and sends M_2 to Alice.

5. Alice computes $m_A^{-1} \in \mathbf{Z}_N$. She computes $M_3 = m_A^{-1} M_2$ and sends M_3 to Bob.

6. Bob computes $m_B^{-1} \in \mathbf{Z}_N$. He computes $M_4 = m_B^{-1} M_3$. Then $M_4 = M$ is the message.

Let's show that M_4 is the original message M. Formally, we have

$$M_4 = m_B^{-1} m_A^{-1} m_B m_A M = M,$$

but we need to justify the fact that m_A^{-1}, which is an integer representing the inverse of m_A mod N, and m_A cancel each other. We have $m_A^{-1} m_A \equiv 1$

(mod N), so $m_A^{-1}m_A = 1 + kN$ for some k. The group $E(\mathbf{F}_q)$ has order N, so Lagrange's theorem implies that $NR = \infty$ for any $R \in E(\mathbf{F}_q)$. Therefore,

$$m_A^{-1}m_A R = (1 + kN)R = R + k\infty = R.$$

Applying this to $R = m_B M$, we find that

$$M_3 = m_A^{-1}m_B m_A M = m_B M.$$

Similarly, m_B^{-1} and m_B cancel, so

$$M_4 = m_B^{-1}M_3 = m_B^{-1}m_B M = M.$$

The eavesdropper Eve knows $E(\mathbf{F}_q)$ and the points $m_A M$, $m_B m_A M$, and $m_B M$. Let $a = m_A^{-1}, b = m_B^{-1}, P = m_A m_B M$. Then we see that Eve knows P, bP, aP and wants to find abP. This is the Diffie-Hellman problem (see Section 6.2).

The above procedure works in any finite group. It seems that the method is rarely used in practice.

It remains to show how to represent a message as a point on an elliptic curve. We use a method proposed by Koblitz. Suppose E is an elliptic curve given by $y^2 = x^3 + Ax + B$ over $\mathbf{F}_p$. The case of an arbitrary finite field $\mathbf{F}_q$ is similar. Let m be a message, expressed as a number $0 \le m < p/100$. Let $x_j = 100m + j$ for $0 \le j < 100$. For $j = 0, 1, 2, \ldots, 99$, compute $s_j = x_j^3 + Ax_j + B$. If $s_j^{(p-1)/2} \equiv 1 \pmod{p}$, then s_j is a square mod p, in which case we do not need to try any more values of j. When $p \equiv 3 \pmod 4$, a square root of s_j is then given by $y_j \equiv s_j^{(p+1)/4} \pmod{p}$ (see Exercise 6.7). When $p \equiv 1 \pmod 4$, a square root of s_j can also be computed, but the procedure is more complicated (see [25]). We obtain a point (x_j, y_j) on E. To recover m from (x_j, y_j), simply compute $[x_j/100]$ (= the greatest integer less than or equal to $x_j/100$). Since s_j is essentially a random element of $\mathbf{F}_p^\times$, which is cyclic of even order, the probability is approximately $1/2$ that s_j is a square. So the probability of not being able to find a point for m after trying 100 values is around 2^{-100}.

6.4 ElGamal Public Key Encryption

Alice wants to send a message to Bob. First, Bob establishes his public key as follows. He chooses an elliptic curve E over a finite field $\mathbf{F}_q$ such that the discrete log problem is hard for $E(\mathbf{F}_q)$. He also chooses a point P on E (usually, it is arranged that the order of P is a large prime). He chooses a secret integer s and computes $B = sP$. The elliptic curve E, the finite field

$\mathbf{F}_q$, and the points P and B are Bob's public key. They are made public. Bob's private key is the integer s.

To send a message to Bob, Alice does the following:

1. Downloads Bob's public key.
2. Expresses her message as a point $M \in E(\mathbf{F}_q)$.
3. Chooses a secret random integer k and computes $M_1 = kP$.
4. Computes $M_2 = M + kB$.
5. Sends M_1, M_2 to Bob.

Bob decrypts by calculating

$$M = M_2 - sM_1.$$

This decryption works because

$$M_2 - sM_1 = (M + kB) - s(kP) = M + k(sP) - skP = M.$$

The eavesdropper Eve knows Bob's public information and the points M_1 and M_2. If she can calculate discrete logs, she can use P and B to find s, which she can then use to decrypt the message as $M_2 - sM_1$. Also, she could use P and M_1 to find k. Then she can calculate $M = M_2 - kB$. If she cannot calculate discrete logs, there does not appear to be a way to find M.

It is important for Alice to use a different random k each time she sends a message to Bob. Suppose Alice uses the same k for both M and M'. Eve recognizes this because then $M_1 = M_1'$. She then computes $M_2' - M_2 = M' - M$. Suppose M is a sales announcement that is made public a day later. Then Eve finds out M, so she calculates $M' = M - M_2 + M_2'$. Therefore, knowledge of one plaintext M allows Eve to deduce another plaintext M' in this case.

The ElGamal Public Key system, in contrast to the ElGamal signature scheme of the next section, does not appear to be widely used.

6.5 ElGamal Digital Signatures

Alice wants to sign a document. The classical way is to write her signature on a piece of paper containing the document. Suppose, however, that the document is electronic, for example, a computer file. The naive solution would be to digitize Alice's signature and append it to the file containing the document. In this case, evil Eve can copy the signature and append it to another document. Therefore, steps must be taken to tie the signature to

the document in such a way that it cannot be used again. However, it must be possible for someone to verify that the signature is valid, and it should be possible to show that Alice must have been the person who signed the document. One solution to the problem relies on the difficulty of discrete logs. Classically, the algorithm was developed for the multiplicative group of a finite field. In fact, it applies to any finite group. We'll present it for elliptic curves.

Alice first must establish a public key. She chooses an elliptic curve E over a finite field $\mathbf{F}_q$ such that the discrete log problem is hard for $E(\mathbf{F}_q)$. She also chooses a point $A \in E(\mathbf{F}_q)$. Usually the choices are made so that the order N of A is a large prime. Alice also chooses a secret integer a and computes $B = aA$. Finally, she chooses a function

$$f : E(\mathbf{F}_q) \to \mathbf{Z}.$$

For example, if $\mathbf{F}_q = \mathbf{F}_p$, then she could use $f(x, y) = x$, where x is regarded as an integer, $0 \leq x < p$. The function f needs no special properties, except that its image should be large and only a small number of inputs should produce any given output (for example, for $f(x, y) = x$, at most two points (x, y) yield a given output x).

Alice's public information is E, $\mathbf{F}_q$, f, A, and B. She keeps a private. The integer N does not need to be made public. Its secrecy does not affect our analysis of the security of the system. To sign a document, Alice does the following:

1. Represents the document as an integer m (if $m > N$, choose a larger curve, or use a hash function (see below)).
2. Chooses a random integer k with $\gcd(k, N) = 1$ and computes $R = kA$.
3. Computes $s \equiv k^{-1}(m - af(R)) \pmod{N}$.

The signed message is (m, R, s). Note that m, s are integers, while R is a point on E. Also, note that Alice is not trying to keep the document m secret. If she wants to do that, then she needs to use some form of encryption. Bob verifies the signature as follows:

1. Downloads Alice's public information.
2. Computes $V_1 = f(R)B + sR$ and $V_2 = mA$.
3. If $V_1 = V_2$, he declares the signature valid.

If the signature is valid, then $V_1 = V_2$ since

$$V_1 = f(R)B + sR = f(R)aA + skA = f(R)aA + (m - af(R))A = mA = V_2.$$

We have used the fact that $sk \equiv m - af(R)$, hence $sk = m - af(R) + zN$ for some integer z. Therefore,

$$skA = (m - af(R))A + zNA = (m - af(R))A + \infty = (m - af(R))A.$$

This is why the congruence defining s was taken mod N.

If Eve can calculate discrete logs, then she can use A and B to find a. In this case, she can put Alice's signature on any message. Alternatively, Eve can use A and R to find k. Since she knows $s, f(R), m$, she can then use $ks \equiv m - af(R) \pmod N$ to find a. If $d = \gcd(f(R), N) \neq 1$, then $af(R) \equiv m - ks \pmod N$ has d solutions for a. As long as d is small, Eve can try each possibility until she obtains $B = aA$. Then she can use a, as before, to forge Alice's signature on arbitrary messages.

As we just saw, Alice must keep a and k secret. Also, she must use a different random k for each signature. Suppose she signs m and m' using the same k to obtain signed messages (m, R, s) and (m', R, s'). Eve immediately recognizes that k has been used twice since R is the same for both signatures. The equations for s, s' yield the following:

$$\begin{aligned} ks &\equiv m - af(R) \pmod N \\ ks' &\equiv m' - af(R) \pmod N. \end{aligned}$$

Subtracting yields $k(s-s') \equiv m-m' \pmod N$. Let $d = \gcd(s-s', N)$. There are d possible values for k. Eve tries each one until $R = kA$ is satisfied. Once she knows k, she can find a, as above.

It is perhaps not necessary for Eve to solve discrete log problems in order to forge Alice's signature on another message m. All Eve needs to do is produce R, s such that the verification equation $V_1 = V_2$ is satisfied. This means that she needs to find $R = (x, y)$ and s such that

$$f(R)B + sR = mA.$$

If she chooses some point R (there is no need to choose an integer k), she needs to solve the discrete log problem $sR = mA - f(R)B$ for the integer s. If, instead, she chooses s, then she must solve an equation for $R = (x, y)$. This equation appears to be at least as complex as a discrete log problem, though it has not been analyzed as thoroughly. Moreover, no one has been able to rule out the possibility of using some procedure that finds R and s simultaneously. There are ways of using a valid signed message to produce another valid signed message (see Exercise 6.2). However, the messages produced are unlikely to be meaningful messages.

The general belief is that the security of the ElGamal system is very close to the security of discrete logs for the group $E(\mathbf{F}_q)$.

A disadvantage of the ElGamal system is that the signed message (m, R, s) is approximately three times as long as the original message (it is not necessary to store the full y-coordinate of R since there are only two choices for y for a given x). A more efficient method is to choose a public hash function H and sign $H(m)$. A **cryptographic hash function** is a function that takes inputs of arbitrary length, sometimes a message of billions of bits, and outputs values of fixed length, for example, 160 bits. A hash function H should have the following properties:

1. Given a message m, the value $H(m)$ can be calculated very quickly.

2. Given y, it is computationally infeasible to find m with $H(m) = y$. (This says that H is **preimage resistant**.)

3. It is computationally infeasible to find distinct messages m_1 and m_2 with $H(m_1) = H(m_2)$. (This says that H is **strongly collision-free**.)

The reason for (2) and (3) is to prevent Eve from producing messages with a desired hash value, or two messages with the same hash value. This helps prevent forgery. There are several popular hash functions available, for example, MD5 (due to Rivest; it produces a 128-bit output) and the Secure Hash Algorithm (from NIST; it produces a 160-bit output). We won't discuss these here. For details, see [81]. Recent work of Wang, Yin, and Yu [127] has found weaknesses in them, so the subject is somewhat in a state of flux.

If Alice uses a hash function, the signed message is then

$$(m, R_H, s_H),$$

where $(H(m), R_H, s_H)$ is a valid signature. To verify that the signature (m, R_H, s_H) is valid, Bob does the following:

1. Downloads Alice's public information.

2. Computes $V_1 = f(R_H)B + s_H R_H$ and $V_2 = H(m)A$.

3. If $V_1 = V_2$, he declares the signature valid.

The advantage is that a very long message m containing billions of bits has a signature that requires only a few thousand extra bits. As long as the discrete log problem is hard for $E(\mathbf{F}_q)$, Eve will be unable to put Alice's signature on another message. The use of a hash function also guards against certain other forgeries (see Exercise 6.2).

A recent variant of the ElGamal signature scheme due to van Duin is very efficient in certain aspects. For example, it avoids the computation of k^{-1}, and its verification procedure requires only two computations of an integer times a point. As before, Alice has a document m that she wants to sign. To set up the system, she chooses an elliptic curve E over a finite field $\mathbf{F}_q$ and a point $A \in E(\mathbf{F}_q)$ of large prime order N. She also chooses a cryptographic hash function H. She chooses a secret integer a and computes $B = aA$. The public information is (E, q, N, H, A, B). The secret information is a. To sign m, Alice does the following:

1. Chooses a random integer $k \bmod N$ and computes $R = kA$.

2. Computes $t = H(R, m)k + a \pmod N$.

The signed document is (m, R, t).

To verify the signature, Bob downloads Alice's public information and checks whether

$$tA = H(R, m)R + B$$

is true. If it is, the signature is declared valid; otherwise, it is invalid.

6.6 The Digital Signature Algorithm

The Digital Signature Standard [1],[86] is based on the Digital Signature Algorithm (DSA). The original version used multiplicative groups of finite fields. A more recent elliptic curve version (ECDSA) uses elliptic curves. The algorithm is a variant on the ElGamal signature scheme, with some modifications. We sketch the algorithm here.

Alice wants to sign a document m, which is an integer (actually, she usually signs the hash of the document, as in Section 6.5). Alice chooses an elliptic curve over a finite field $\mathbf{F}_q$ such that $\#E(\mathbf{F}_q) = fr$, where r is a large prime and f is a small integer, usually 1,2, or 4 (f should be small in order to keep the algorithm efficient). She chooses a base point G in $E(\mathbf{F}_q)$ of order r. Finally, Alice chooses a secret integer a and computes $Q = aG$. Alice makes public the following information:

$$\mathbf{F}_q, \quad E, \quad r, \quad G, \quad Q.$$

(There is no need to keep f secret; it can be deduced from q and r using Hasse's theorem by the technique in Examples 4.6 and 4.7.) To sign the message m Alice does the following:

1. Chooses a random integer k with $1 \leq k < r$ and computes $R = kG = (x, y)$.
2. Computes $s = k^{-1}(m + ax) \pmod{r}$.

The signed document is

$$(m,\ R,\ s).$$

To verify the signature, Bob does the following.

1. Computes $u_1 = s^{-1}m \pmod{r}$ and $u_2 = s^{-1}x \pmod{r}$.
2. Computes $V = u_1G + u_2Q$.
3. Declares the signature valid if $V = R$.

If the message is signed correctly, the verification equation holds:

$$V = u_1G + u_2Q = s^{-1}mG + s^{-1}xQ = s^{-1}(mG + xaG) = kG = R.$$

The main difference between the ECDSA and the ElGamal system is the verification procedure. In the ElGamal system, the verification equation $f(R)B + sR = mA$ requires three computations of an integer times a point. These are the most expensive parts of the algorithm. In the ECDSA, only two computations of an integer times a point are needed. If many verifications are going to be made, then the improved efficiency of the ECDSA is valuable. This is the same type of improvement as in the van Duin system mentioned at the end of the previous section.

6.7 ECIES

The Elliptic Curve Integrated Encryption Scheme (ECIES) was invented by Bellare and Rogaway [2]. It is a public key encryption scheme.

Alice wants to send a message m to Bob. First, Bob establishes his public key. He chooses an elliptic curve E over a finite field $\mathbf{F}_q$ such that the discrete log problem is hard for $E(\mathbf{F}_q)$, and he chooses a point A on E, usually of large prime order N. He then chooses a secret integer s and computes $B = sA$. The public key is (q, E, N, A, B). The private key is s.

The algorithm also needs two cryptographic hash functions, H_1 and H_2, and a symmetric encryption function E_k (depending on a key k) that are publicly agreed upon.

To encrypt and send her message, Alice does the following:

1. Downloads Bob's public key.
2. Chooses a random integer k with $1 \leq k \leq N - 1$.
3. Computes $R = kA$ and $Z = kB$.
4. Writes the output of $H_1(R, Z)$ as $k_1 \| k_2$ (that is, k_1 followed by k_2), where k_1 and k_2 have specified lengths.
5. Computes $C = E_{k_1}(m)$ and $t = H_2(C, k_2)$.
6. Sends (R, C, t) to Bob.

To decrypt, Bob does the following:

1. Computes $Z = sR$, using his knowledge of the secret key s.
2. Computes $H_1(R, Z)$ and writes the output as $k_1 \| k_2$.

3. Computes $H_2(C, k_2)$. If it does not equal t, Bob stops and rejects the ciphertext. Otherwise, he continues.

4. Computes $m = D_{k_1}(C)$, where D_{k_1} is the decryption function for E_{k_1}.

An important feature is the authentication procedure in step (3) of the decryption. In many cryptosystems, an attacker can choose various ciphertexts and force Bob to decrypt them. These decryptions are used to attack the system. In the present system, the attacker can generate ciphertexts by choosing C and k_2' and then letting $t' = H_2(C, k_2')$. But the attacker does not know Z, so he cannot use the same value k_2 that Bob obtains from $H_1(R, Z)$. Therefore, it is very unlikely that $t' = H_2(C, k_2')$ will equal $t = H_2(C, k_2)$. With very high probability, Bob simply rejects the ciphertext and does not return a decryption.

In our description of the procedure, we used hash functions for the authentication. There are other message authentication methods that could be used.

An advantage of ECIES over the Massey-Omura and ElGamal public key methods is that the message is not represented as a point on the curve. Moreover, since a keyed symmetric method is used to send the message, we do not need to do a new elliptic curve calculation for each block of the message.

6.8 A Public Key Scheme Based on Factoring

Most cryptosystems using elliptic curves are based on the discrete log problem, in contrast to the situation for classical systems, which are sometimes based on discrete logs and sometimes based on the difficulty of factorization. The most famous public key cryptosystem is called RSA (for Rivest-Shamir-Adleman) and proceeds as follows. Alice wants to send a message to Bob. Bob secretly chooses two large primes p, q and multiplies them to obtain $n = pq$. Bob also chooses integers e and d with $ed \equiv 1 \pmod{(p-1)(q-1)}$. He makes n and e public and keeps d secret. Alice's message is a number $m \pmod{n}$. She computes $c \equiv m^e \pmod{n}$ and sends c to Bob. Bob computes $m \equiv c^d \pmod{n}$ to obtain the message. If Eve can find p and q, then she can solve $ed \equiv 1 \pmod{(p-1)(q-1)}$ to obtain d. It can be shown (by methods similar to those used in the elliptic curve scheme below; see [121]) that if Eve can find the decryption exponent d, then she probably can factor n. Therefore, the difficulty of factoring n is the key to the security of the RSA system.

A natural question is whether there is an elliptic curve analogue of RSA. In the following, we present one such system, due to Koyama-Maurer-Okamoto-Vanstone. It does not seem to be used much in practice.

Alice want to send a message to Bob. They do the following.

1. Bob chooses two distinct large primes p, q with $p \equiv q \equiv 2 \pmod 3$ and computes $n = pq$.
2. Bob chooses integers e, d with $ed \equiv 1 \pmod{\text{lcm}(p+1, q+1)}$. (He could use $(p+1)(q+1)$ in place of $\text{lcm}(p+1, q+1)$.)
3. Bob makes n and e public (they form his public key) and he keeps d, p, q private.
4. Alice represents her message as a pair of integers $(m_1, m_2) \pmod n$. She regards (m_1, m_2) as a point M on the elliptic curve E given by
$$y^2 = x^3 + b \mod n,$$
where $b = m_2^2 - m_1^3 \pmod n$ (she does not need to compute b).
5. Alice adds M to itself e times on E to obtain $C = (c_1, c_2) = eM$. She sends C to Bob.
6. Bob computes $M = dC$ on E to obtain M.

We'll discuss the security of the system shortly. But, first, there are several points that need to be discussed.

1. Note that the formulas for the addition law on E never use the value of b. Therefore, Alice and Bob never need to compute it. Eve can compute it, if she wants, as $b = c_2^2 - c_1^3$.
2. The computation of eM and dC on E are carried out with the formulas for the group law on an elliptic curve, with all of the computations being done mod n. Several times during the computation, expressions such as $(y_2 - y_1)/(x_2 - x_1)$ are encountered. These are changed to integers mod n by finding the multiplicative inverse of $(x_2 - x_1)$ mod n. This requires $\gcd(x_2 - x_1, n) = 1$. If the gcd is not 1, then it is p, q, or n. If we assume it is very hard to factor n, then we regard the possibility of the gcd being p or q as very unlikely. If the gcd is n, then the slope is infinite and the sum of the points in question is ∞. The usual rules for working with ∞ are followed. For technical details of working with elliptic curves mod n, see Section 2.11.

   By the Chinese Remainder Theorem, an integer mod n may be regarded as a pair of integers, one mod p and one mod q. Therefore, we can regard a point on E in $\mathbf{Z}_n$ as a pair of points, one on E mod p and the other on E mod q. In this way, we have
$$E(\mathbf{Z}_n) = E(\mathbf{F}_p) \oplus E(\mathbf{F}_q). \tag{6.1}$$
   For example, the point $(11, 32)$ on $y^2 = x^3 + 8 \mod 35$ can be regarded as the pair of points
$$(1, 2) \mod 5, \quad (4, 4) \mod 7.$$

Any such pair of points can be combined to obtain a point mod n. There is a technicality with points at infinity, which is discussed in Section 2.11.

3. Using (6.1), we see that the order of $E(\mathbf{Z}_n)$ is $\#E(\mathbf{F}_p) \cdot \#E(\mathbf{F}_q)$. By Proposition 4.33, E is supersingular mod p and mod q, so we find (by Corollary 4.32) that

$$\#E(\mathbf{F}_p) = p+1 \text{ and } \#E(\mathbf{F}_q) = q+1.$$

Therefore, $(p+1)M = \infty \pmod{p}$ and $(q+1)M = \infty \pmod{q}$. This means that the decryption works: Write $de = 1 + k(p+1)$ for some integer k. Then

$$dC = deM = (1+k(p+1))M = M+k(p+1)M = M+\infty = M \pmod{p},$$

and similarly mod q. Therefore, $dC = M$.

4. A key point of the procedure is that the group order is independent of b. If Bob chooses a random elliptic curve $y^2 = x^3 + Ax + B$ over $\mathbf{Z}_n$, then he has to compute the group order, perhaps by computing it mod p and mod q. This is infeasible if p and q are chosen large enough to make factoring n infeasible. Also, if Bob fixes the elliptic curve, Alice will have difficulty finding points M on the curve. If she does the procedure of first choosing the x-coordinate as the message, then solving $y^2 \equiv m^3 + Am + B \pmod{n}$ for y, she is faced with the problem of computing square roots mod n. This is computationally equivalent to factoring n (see [121]). If Bob fixes only A (the formulas for the group operations depend only on A) and allows Alice to choose B so that her point lies on the curve, then his choice of e, d requires that the group order be independent of B. This is the situation in the above procedure.

If Eve factors n as pq, then she knows $(p+1)(q+1)$, so she can find d with $ed \equiv 1 \pmod{(p+1)(q+1)}$. Therefore, she can decrypt Alice's message.

Suppose that Eve does not yet know the factorization of n, but she finds out the decryption exponent d. We claim that she can, with high probability, factor n. She does the following:

1. Writes $ed-1 = 2^k v$ with v odd and with $k \geq 1$ ($k \neq 0$ since $p+1$ divides $ed-1$).

2. Picks a random pair of integers $R = (r_1, r_2) \mod n$, lets $b' = r_2^2 - r_1^3$, and regards R as a point on the elliptic curve E' given by $y^2 = x^3 + b'$.

3. Computes $R_0 = vR$. If $R_0 = \infty \bmod n$, start over with a new R. If R_0 is ∞ mod exactly one of p, q, then Eve has factored n (see below).

4. For $i = 0, 1, 2, \ldots, k$, computes $R_{i+1} = 2R_i$.

5. If for some i, the point R_{i+1} is ∞ mod exactly one of p, q, then $R_i = (x_i, y_i)$ with $y_i \equiv 0$ mod one of p, q. Therefore, $\gcd(y_i, n) = p$ or q. In this case, Eve stops, since she has factored n.

6. If for some i, $R_{i+1} = \infty$ mod n, then Eve starts over with a new random point.

In a few iterations, this should factor n. Since $ed-1$ is a multiple of $\#E(\mathbf{Z}_n)$,

$$R_k = (ed-1)R = edR - R = \infty.$$

Therefore, each iteration of the procedure will eventually end with a point R_j that is ∞ mod at least one of p, q. Let $2^{k'}$ be the highest power of 2 dividing $p+1$. If we take a random point P in $E(\mathbf{F}_p)$, then the probability is $1/2$ that the order of P is divisible by $2^{k'}$. This follows easily from the fact that $E(\mathbf{F}_p)$ is cyclic (see Exercise 6.6). In this case, $R_{k'-1} = 2^{k'-1}vP \neq \infty \pmod{p}$, while $R_{k'} = 2^{k'}vP = \infty \pmod{p}$. If the order is not divisible by $2^{k'}$, then $R_{k'-1} = \infty \pmod{p}$. Similarly, if $2^{k''}$ is the highest power of 2 dividing $q+1$, then $R_{k''-1} = \infty \pmod{q}$ half the time, and $\neq \infty \pmod{q}$ half the time. Since mod p and mod q are independent, it is easy to see that the sequence $R_0, R_1, R_2, \ldots$ reaches ∞ mod p and mod q at different indices i at least half the time. This means that for at least half of the choices of random starting points R, we obtain a factorization of n.

If $R_0 = \infty$ mod p, but not mod q, then somewhere in the calculation of R_0 there was a denominator of a slope that was infinite mod p but not mod q. The gcd of this denominator with n yields p. A similar situation occurs if p and q are switched. Therefore, if R_0 is infinite mod exactly one of the primes, Eve obtains a factorization, as claimed in step (3).

We conclude that knowledge of the decryption exponent d is computationally equivalent to knowledge of the factorization of n.

6.9 A Cryptosystem Based on the Weil Pairing

In Chapter 5, we saw how the Weil pairing could be used to reduce the discrete log problem on certain elliptic curves to the discrete log problem for the multiplicative group of a finite field. In the present section, we'll present a method, due to Boneh and Franklin, that uses the Weil pairing on these curves to obtain a cryptosystem (other pairings could also be used). The reader may wonder why we use these curves, since the discrete log problem is easier on these curves. The reason is that the properties of the pairing are used in an essential way. The fact that the pairing can be computed quickly is vital for the present algorithm. This fact was also important in reducing the discrete log problem to finite fields. However, note that the discrete log

problem in the finite field is still not trivial as long as the finite field is large enough.

For simplicity, we'll consider a specific curve, namely the one discussed in Section 6.2. Let E be defined by $y^2 = x^3 + 1$ over $\mathbf{F}_p$, where $p \equiv 2 \pmod 3$. Let $\omega \in \mathbf{F}_{p^2}$ be a primitive third root of unity. Define a map

$$\beta : E(\mathbf{F}_{p^2}) \to E(\mathbf{F}_{p^2}), \quad (x, y) \mapsto (\omega x, y), \quad \beta(\infty) = \infty.$$

Suppose P has order n. Then $\beta(P)$ also has order n. Define the modified Weil pairing

$$\tilde{e}_n(P_1, P_2) = e_n(P_1, \beta(P_2)),$$

where e_n is the usual Weil pairing and $P_1, P_2 \in E[n]$. We showed in Lemma 6.1 that if $3 \nmid n$ and if $P \in E(\mathbf{F}_p)$ has order exactly n, then $\tilde{e}_n(P, P)$ is a primitive nth root of unity.

Since E is supersingular, by Proposition 4.33, $E(\mathbf{F}_p)$ has order $p+1$. We'll add the further assumption that $p = 6\ell - 1$ for some prime ℓ. Then $6P$ has order ℓ or 1 for each $P \in E(\mathbf{F}_p)$.

In the system we'll describe, each user has a public key based on her or his identity, such as an email address. A central trusted authority assigns a corresponding private key to each user. In most public key systems, when Alice wants to send a message to Bob, she looks up Bob's public key. However, she needs some way of being sure that this key actually belongs to Bob, rather than someone such as Eve who is masquerading as Bob. In the present system, the authentication happens in the initial communication between Bob and the trusted authority. After that, Bob is the only one who has the information necessary to decrypt messages that are encrypted using his public identity.

A natural question is why RSA cannot be used to produce such a system. For example, all users could share the same common modulus n, whose factorization is known only to the trusted authority (TA). Bob's identity, call it *bobid*, would be his encryption exponent. The TA would then compute Bob's secret decryption exponent and communicate it to him. When Alice sends Bob a message m, she encrypts it as $m^{bobid} \pmod n$. Bob then decrypts using the secret exponent provided by the TA. However, anyone such as Bob who knows an encryption and decryption exponent can find the factorization of n (using a variation of the method of Section 6.8), and thus read all messages in the system. Therefore, the system would not protect secrets. If, instead, a different n is used for each user, some type of authentication procedure is needed for a communication in order to make sure that the n is the correct one. This brings us back to the original problem.

The system described in the following gives the basic idea, but is not secure against certain attacks. For ways to strengthen the system, see [15].

To set up the system, the trusted authority does the following:

1. Chooses a large prime $p = 6\ell - 1$ as above.
2. Chooses a point P of order ℓ in $E(\mathbf{F}_p)$.

3. Chooses hash functions H_1 and H_2. The function H_1 takes a string of bits of arbitrary length and outputs a point of order ℓ on E (see Exercise 6.8). The function H_2 inputs an element of order ℓ in $\mathbf{F}_{p^2}^\times$ and outputs a binary string of length n, where n is the length of the messages that will be sent.

4. Chooses a secret random $s \in \mathbf{F}_\ell^\times$ and computes $P_{pub} = sP$.

5. Makes $p, H_1, H_2, n, P, P_{pub}$ public, while keeping s secret.

If a user with identity ID wants a private key, the trusted authority does the following:

1. Computes $Q_{ID} = H_1(ID)$. This is a point on E.

2. Lets $D_{ID} = sQ_{ID}$.

3. After verifying that ID is the identification for the user with whom he is communicating, sends D_{ID} to this user.

If Alice wants to send a message M to Bob, she does the following:

1. Looks up Bob's identity, for example, $ID =$*bob@computer.com* (written as a binary string) and computes $Q_{ID} = H_1(ID)$.

2. Chooses a random $r \in \mathbf{F}_\ell^\times$.

3. Computes $g_{ID} = \tilde{e}_\ell(Q_{ID}, P_{pub})$.

4. Lets the ciphertext be the pair

$$c = (rP,\ M \oplus H_2(g_{ID}^r)),$$

where $\oplus$ denotes XOR (= bitwise addition mod 2).

Bob decrypts a ciphertext (u, v) as follows:

1. Uses his private key D_{ID} to compute $h_{ID} = \tilde{e}_\ell(D_{ID}, u)$.

2. Computes $m = v \oplus H_2(h_{ID})$.

The decryption works because

$$\tilde{e}_\ell(D_{ID}, u) = \tilde{e}_\ell(sQ_{ID}, rP) = \tilde{e}_\ell(Q_{ID}, P)^{sr} = \tilde{e}_\ell(Q_{ID}, P_{pub})^r = g_{ID}^r.$$

Therefore,

$$m = v \oplus H_2(\tilde{e}_\ell(D_{ID}, u)) = (M \oplus H_2(g_{ID}^r)) \oplus H_2(g_{ID}^r) = M.$$

Exercises

6.1 Show that the map β in Section 6.2 is an isomorphism (it is clearly bijective; the main point is that it is a homomorphism).

6.2 (a) Suppose that the ElGamal signature scheme is used to produce the valid signed message (m, R, s), as in Section 6.5. Let h be an integer with $\gcd(h, N) = 1$. Assume $\gcd(f(R), N) = 1$. Let

$$R' = hR, \quad s' \equiv sf(R')f(R)^{-1}h^{-1} \pmod{N},$$
$$m' \equiv mf(R')f(R)^{-1} \pmod{N}.$$

Show that (m', R', s') is a valid signed message (however, it is unlikely that m' is a meaningful message, so this procedure does not affect the security of the system).

(b) Suppose a hash function is used, so the signed messages are of the form (m, R_H, s_H). Explain why this prevents the method of (a) from working.

6.3 Use the notation of Section 6.5. Let u, v be two integers with $\gcd(v, N) = 1$ and let $R = uA + vB$. Let $s \equiv -v^{-1}f(R) \pmod{N}$ and $m \equiv su \pmod{N}$.

(a) Show that (m, R, s) is a valid signed message for the ElGamal signature scheme. (However, it is unlikely that m is a meaningful message.)

(b) Suppose a hash function is used, so the signed messages are of the form (m, R_H, s_H). Explain why this prevents the method of (a) from working.

6.4 Let E be an elliptic curve over $\mathbf{F}_q$ and let $N = \#E(\mathbf{F}_q)$. Alice has a message that she wants to sign. She represents the message as a point $M \in E(\mathbf{F}_q)$. Alice has a secret integer a and makes public points A and B in $E(\mathbf{F}_q)$ with $B = aA$, as in the ElGamal signature scheme. There is a public function $f : E(\mathbf{F}_q) \to \mathbf{Z}/N\mathbf{Z}$. Alice performs the following steps.

(a) She chooses a random integer k with $\gcd(k, N) = 1$.

(b) She computes $R = M - kA$.

(c) She computes $s \equiv k^{-1}(1 - f(R)a) \pmod{N}$.

(d) The signed message is (M, R, s).

Bob verifies the signature as follows.

(a) He computes $V_1 = sR - f(R)B$ and $V_2 = sM - A$.

(b) He declares the signature valid if $V_1 = V_2$.

Show that if Alice performs the required steps correctly, then the verification equation $V_1 = V_2$ holds. (This signature scheme is a variant of one due to Nyberg and Rueppel (see [12]). An interesting feature is that the message appears as an element of the group $E(\mathbf{F}_q)$ rather than as an integer.)

6.5 Let p, q be prime numbers and suppose you know the numbers $m = (p+1)(q+1)$ and $n = pq$. Show that p, q are the roots of the quadratic equation

$$x^2 - (m - n - 1)x + n = 0$$

(so p, q can be found using the quadratic formula).

6.6 Let E be the elliptic curve $y^2 = x^3 + b \bmod p$, where $p \equiv 2 \pmod 3$.

(a) Suppose $E[n] \subseteq E(\mathbf{F}_p)$ for some $n \not\equiv 0 \pmod p$. Show that $n|p-1$ and $n^2|p+1$. Conclude that $n \leq 2$.

(b) Show that $E[2] \not\subseteq E(\mathbf{F}_p)$.

(c) Show that $E(\mathbf{F}_p)$ is cyclic (of order $p+1$).

6.7 Let $p \equiv 3 \pmod 4$ be a prime number. Suppose $x \equiv y^2 \pmod p$.

(a) Show that $(y^{(p+1)/2})^2 \equiv y^2 \pmod p$.

(b) Show that $y^{(p+1)/2} \equiv \pm y \pmod p$.

(c) Show that $x^{(p+1)/4}$ is a square root of $x \pmod p$.

(d) Suppose z is not a square mod p. Using the fact that -1 is not a square mod p, show that $-z$ is a square mod p.

(e) Show that $z^{(p+1)/4}$ is a square root of $-z \pmod p$.

6.8 Let $p = 6\ell - 1$ and E be as in Section 6.9. The hash function H_1 in that section inputs a string of bits of arbitrary length and outputs a point of order ℓ on E. One way to do this is as follows.

(a) Choose a hash function H that outputs integers mod p. Input a binary string B. Let the output of H be the y coordinate of a point: $y = H(B)$. Show that there is a unique x mod p such that (x, y) lies on E.

(b) Let $H_1(B) = 6(x, y)$. Show that $H_1(B)$ is a point of order ℓ or 1 on E. Why is it very unlikely that $H_1(B)$ has order 1?