CHAPTER I

Introduction

We introduce the three main characters in public key cryptography. As in many books on the subject, it is assumed that Alice and Bob wish to perform some form of communication whilst Eve is an eavesdropper who wishes to spy on (or tamper with) the communications between Alice and Bob. Of course there is no assumption that Alice and Bob (or Eve) are actually human. They may (and probably will) be computers on some network such as the Internet.

Modern cryptography, as applied in the commercial world, is concerned with a number of problems. The most important of these are:

1. **Confidentiality:** A message sent from Alice to Bob cannot be read by anyone else.
2. **Authenticity:** Bob knows that only Alice could have sent the message he has just received.
3. **Integrity:** Bob knows that the message from Alice has not been tampered with in transit.
4. **Non-repudiation:** It is impossible for Alice to turn around later and say she did not send the message.

To see why all four properties are important consider the following scenario. Alice wishes to buy some item over the Internet from Bob. She sends her instruction to Bob which contains her credit card number and payment details. She requires that this communication be confidential, since she wants other people to know neither her credit card details nor what she is buying. Bob needs to know that the message is authentic in that it came from Alice and not some impostor. Both Alice and Bob need to be certain that the message's integrity is preserved, for example the amount cannot be altered by some third party whilst it is in transit. Finally Bob requires the non-repudiation property, meaning that Alice should not be able to say she did not send the instruction.

In other words, we require transactions to take place between two mutually distrusting parties over a public network. This is different from conventional private networks, such as those used in banking, where there are key hierarchies and tamper proof hardware which can store symmetric keys.

It is common in the literature to introduce public key techniques in the area of confidentiality protection. Public key techniques are, however, usually infeasible to use directly in this context, being orders of magnitude slower than symmetric techniques. Their use in confidentiality is often limited to

the transmission of symmetric cipher keys. On the other hand *digital signatures*, which give the user the authentication, integrity and non-repudiation properties required in electronic commerce, seem to require the use of public key cryptography.

A computer which is processing payments for a bank or a business may need to verify or create thousands of digital signatures every second. This has led to the demand for public key digital signature schemes which are very efficient. Whilst many schemes are based on the discrete logarithm problem in a finite abelian group, there is some debate as to what type of groups to use. One choice is the group of points on an elliptic curve over a finite field. This choice is becoming increasingly popular, precisely because of efficiency considerations. In this book, we attempt to summarize the latest knowledge available on both theoretical and practical issues related to elliptic curve cryptosystems.

I.1. Cryptography Based on Groups

In this section, some of the standard protocols of public key cryptography are surveyed. A more detailed discussion of all of these protocols and other related areas of cryptography can be found in the books by Menezes, van Oorschot and Vanstone [**99**] and Schneier [**139**], although neither of these books covers the use of elliptic curves in cryptography. The protocols discussed here only require the use of a finite abelian group G, of order $\#G$, which is assumed to be cyclic. The group of interest in this work is the *additive* group of points on an elliptic curve. However, it is convenient for the remainder of this chapter to assume the group is *multiplicative*, with generator g, and that the order, $\#G$, is a prime. If this is not the case, we can always take a prime order subgroup of G as our group, with no loss of security. The additive vs. multiplicative issue is, of course, just one of notation. We will revert to additive notation later on, when the discussion focuses on the elliptic curve groups.

The group G should be presented in such a way as to make multiplication and exponentiation easy, whilst computing discrete logarithms is hard. The reason for this will become clearer below. It should also be possible to generate random elements from the group with an almost uniform distribution.

By the *discrete logarithm problem* (DLP) we mean the problem of determining the least positive integer, x, if it exists, which satisfies the equation

$$h = g^x$$

for two, given, elements h and g in the group G. Note that a common feature of all of the following schemes is that if there is a fast way to solve the DLP in G, then they are all insecure for the group G. Since we have assumed that G is of prime order such a discrete logarithm always exists.

I.1.1. Diffie–Hellman key exchange. Alice and Bob wish to agree on a secret random element in the group, which could be of use as a key for a

higher speed symmetric algorithm like the *Data Encryption Standard* (DES). They wish to make this agreement over an insecure channel, without having exchanged any information previously. The only public items, which can be shared amongst a group of users, are the group G and an element $g \in G$ of large known order.

1. Alice generates a random integer $x_A \in \{1, \ldots, \#G - 1\}$. She sends to Bob the element
$$g^{x_A}.$$
2. Bob generates a random integer $x_B \in \{1, \ldots, \#G - 1\}$. He sends to Alice the element
$$g^{x_B}.$$
3. Alice can then compute
$$g^{x_A x_B} = (g^{x_B})^{x_A}.$$
4. Likewise, Bob can compute
$$g^{x_A x_B} = (g^{x_A})^{x_B}.$$

The only information that Eve knows is G, g, g^{x_A} and g^{x_B}. If Eve can recover $g^{x_A x_B}$ from this data then Eve is said to have solved a *Diffie–Hellman problem* (DHP). It is easy to see that if Eve can find discrete logarithms in G then she can solve the DHP. It is believed for most groups in use in cryptography that the DHP and the DLP are equivalent [**94**], in a complexity-theoretic sense (there is a polynomial time reduction of one problem to the other, and vice versa).

I.1.2. ElGamal encryption [39]. Alice wishes to send a message to Bob. Her message, m, is assumed to be encoded as an element in the group. Bob has a public key consisting of g and $h = g^x$, where x is the private key.

1. Alice generates a random integer $k \in \{1, \ldots, \#G - 1\}$ and computes
$$a = g^k, \ b = h^k m.$$
2. Alice sends the cipher text (a, b) to Bob.
3. Bob can recover the message from the equation
$$ba^{-x} = h^k m g^{-kx} = g^{xk - xk} m = m.$$

I.1.3. ElGamal digital signature [39]. Here, Bob wants to sign a message $m \in (\mathbb{Z}/(\#G)\mathbb{Z})$. He can use the same public and private key pair, h and x, as he used for the encryption scheme. We will need a bijection f from G to $\mathbb{Z}/(\#G)\mathbb{Z}$.

1. Bob generates a random integer $k \in \{1, \ldots, \#G - 1\}$, and computes
$$a = g^k.$$
2. Bob computes a solution, $b \in \mathbb{Z}/(\#G)\mathbb{Z}$, to the congruence
$$m \equiv x f(a) + bk \pmod{\#G}.$$

3. Bob sends the signature, (a, b), and the message, m, to Alice.
4. Alice verifies the signature by checking that the following equation holds:
$$h^{f(a)}a^b = g^{xf(a)+kb} = g^m.$$

I.1.4. Digital Signature Algorithm. A version of ElGamal signatures, called the *Digital Signature Algorithm* (DSA), is the basis of the Digital Signature Standard [**FIPS186**]. An elliptic curve version of DSA (ECDSA) is described in the IEEE P1363 standard draft [**P1363**]. The signature procedure is almost identical to the ElGamal scheme above. It is described here for the sake of completeness, as well as to introduce a slightly different signature verification procedure with some computational advantages.

Bob wants to sign a message $m \in \mathbb{Z}/(\#G)\mathbb{Z}$. He uses the same public and private key pair h and x as before, and both he and Alice use a common bijective mapping, f, from G to $\mathbb{Z}/(\#G)\mathbb{Z}$.

1. Bob generates a random integer $k \in \{1, \ldots, \#G - 1\}$, and computes
$$a = g^k.$$
2. He computes the solution, b, to the congruence
$$m \equiv -xf(a) + kb \pmod{\#G}.$$
3. He sends the signature, (a, b), and the message, m, to Alice.
4. Alice computes
$$u = mb^{-1} \pmod{\#G}\ ,\ v = f(a)b^{-1} \pmod{\#G}.$$
5. She then computes
$$w = g^u h^v$$
and verifies that
$$\begin{aligned} w &= g^u h^v = g^{mb^{-1}} g^{vx} = g^{mb^{-1}+xf(a)b^{-1}} \\ &= g^{(m+xf(a))b^{-1}} = g^{kbb^{-1}} = g^k \\ &= a. \end{aligned}$$

Although the signature verification procedure implemented by Alice appears, at first glance, more complicated than the one described for the ElGamal scheme, it is in fact computationally simpler. Upon closer scrutiny, one notes that the verification procedure described for DSA requires two group exponentiations, while the one described for the ElGamal scheme requires three. The two procedures are, of course, mathematically equivalent.

In its standardized versions, the DSA requires also a secure *hashing function*. This is a many-to-one function that maps the original message to a shorter *digest*, in a way that is infeasible to invert in practice. The message digest is the quantity actually operated on, in lieu of m. See, e.g., [**99**] or [**P1363**] for the details.

I.1.5. Massey–Omura encryption. Here Alice wishes to send a message to Bob. They do not need to have a private or public key. The message is encoded as an element $m \in G$. This protocol is sometimes described as the 'you-to-me, me-to-you' method. It requires Alice and Bob to carry out a conversation rather than just a single transmission of encrypted text.

1. Alice computes a random integer, x_A, coprime to $\#G$, and sends Bob the element
$$a = m^{x_A}.$$
2. Bob computes a random integer, x_B, coprime to $\#G$, and sends back to Alice the element
$$b = a^{x_B} = m^{x_A x_B}.$$
3. Alice can compute $x_A^{-1} \pmod{\#G}$ and so sends back to Bob the element
$$a' = b^{x_A^{-1}} = m^{x_A x_B x_A^{-1}} = m^{x_B}.$$
4. Finally Bob computes $x_B^{-1} \pmod{\#G}$ and can decrypt the message as
$$(a')^{x_B^{-1}} = m^{x_B x_B^{-1}} = m.$$

This algorithm, also referred to as the 'double lock' algorithm, is seldom used in practice but is of historical interest.

I.1.6. Nyberg–Rueppel digital signature [113]. Nyberg and Rueppel present a series of digital signature schemes which allow message recovery. Below we give a variant of one of these schemes, based on a system of Piveteau [**122**]. However, here it is given as a standard signature scheme without any message recovery. For details on how to add message recovery, to this and to other schemes, we refer the reader to [**113**].

Our reason for including the following scheme is that the message to be signed, m, is a member of the group G and not $\mathbb{Z}/(\#G)\mathbb{Z}$. This makes it slightly different from the ElGamal and DSA schemes above.

Once again we assume f is a bijection from G to $\mathbb{Z}/(\#G)\mathbb{Z}$. Alice wishes to sign a message, $m \in G$. She has a private key $x \in \mathbb{Z}$, coprime to $\#G$, and a public key $y = g^x$.

1. She computes a random integer, k, coprime to $\#G$, and computes $r = g^{-k}m$.
2. Alice then computes a solution, s, to the congruence
$$1 \equiv f(r)x + sk \pmod{\#G}.$$
3. She sends the message, m, and the digital signature, (r, s), to Bob.
4. Bob can verify that the message came from Alice by verifying the equation
$$y^{-f(r)}r^s = g^{sk-1-sk}m^s = m^s g^{-1}.$$

I.1.7. Problem reductions. It is not proven that breaking any of the above schemes is equivalent to solving the DLP, but this is believed to be the case. That no proof of this fact has been found is similar to other situations in cryptography: for example there is no proof that breaking the RSA system ([**133**] [**134**]) is equivalent to factoring the modulus, although the recent work of Boneh and Venkatesan [**19**] gives evidence that they may not be equivalent. There are a few public key cryptographic schemes for which one can prove that breaking the system is at least as hard as solving some hard mathematical problem, such as factoring a number or taking discrete logarithms. However, these are not discussed here.

We do note that for some classes of finite abelian groups one can prove that breaking the Diffie–Hellman key exchange protocol is polynomial time equivalent to solving a DLP. What is interesting about this work is that this result uses auxiliary groups which are themselves usually taken to be elliptic curves. The interested reader should consult [**94**], [**95**], [**18**] and Section IX.4.

The requirement in the signature schemes for a bijective function, f, from G to $\mathbb{Z}/(\#G)\mathbb{Z}$ may seem a little restrictive. For the groups, $\mathbb{F}_p^*$, the bijective function to use is obvious. For other groups the condition that f is bijective can be weakened. What is really required is a function

$$f : G \longrightarrow \mathbb{Z}/M\mathbb{Z}$$

for some number M, of the order of magnitude of the size of the group G, which is almost injective. In other words its degree as a map should be 'small'.

For elliptic curve systems the group elements are presented as pairs, (x, y), over some finite field. Such a pair represents a point on an elliptic curve. Over large prime fields, $\mathbb{F}_p$, field elements are naturally represented as integers modulo p, and one usually just uses the x-coordinate of the curve as the map from points (group elements) to integers modulo p (the latter prime turns out to be close to $\#G$, and is thus used for M above). This is a degree two map and will clearly suffice for applications. For large finite fields of characteristic two, one performs a similar method, but a way of converting the x-coordinate into an integer is needed. A simple method, used in practice, is to take the representation of x relative to a given basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, and interpret the same coefficients as the binary digits of an integer. As long as Alice and Bob are using the same internal representation and order conventions, or at least Bob knows how to convert from his internal representation into Alice's, their implementations should be interoperable.

I.2. What Types of Group are Used

All of the above protocols work for a general abelian group, G, so one could consider various other groups to use in such protocols. However, since the protocols are to be implemented in hardware or software, the group operation should be simple to realize. One way of interpreting this condition, but not the only way, is to insist that the group operation be given by simple algebraic

formulae. In other words G must be a commutative finite algebraic group. This then restricts quite considerably the types of such groups which are available.

A commutative finite algebraic group is essentially equivalent to the product of a finite number of copies of the additive and multiplicative groups of finite fields and a finite number of abelian varieties. For all practical purposes, the latter can be taken to be Jacobians of curves. It will be seen in Chapter V that, owing to a general purpose algorithm of Pohlig and Hellman, the group G should have a large subgroup of prime order. Thus we can restrict ourselves to only considering single copies of additive and multiplicative subgroups of finite fields or Jacobians.

The DLP in some additive groups is clearly easy, e.g. the additive group of a finite field. Fortunately, this is not the case, as far as is known, for the group of an elliptic curve. Not surprisingly, all of the above protocols were originally described in terms of the finite (multiplicative) abelian group $\mathbb{F}_q^*$. However, if one uses such groups the choice of q needs to be very large indeed, because there are known sub-exponential methods for solving the DLP in $\mathbb{F}_q^*$ (see [**1**] and [**88**]). These methods are usually based on the ideas behind the well known number field sieve factoring method (see [**77**]).

This situation led Miller [**103**] and Koblitz [**62**] to propose the technique, common in number theory, of replacing a group such as $\mathbb{F}_q^*$ with the group, $E(\mathbb{F}_q)$, of rational points on an elliptic curve, E, defined over $\mathbb{F}_q$ (these concepts will be precisely defined later). This technique will be seen again in the elliptic curve factoring method and the elliptic curve primality proving method. Elliptic curves are Jacobians of dimension one and so are the simplest case of a Jacobian. It turns out that the (additive) DLP in elliptic curve groups is, at present, orders of magnitude harder than the corresponding problem in the multiplicative group of a finite field of a similar size, a fact that is more precisely quantified in the next section.

If one wants to avoid algebraic groups then only one other type of group is known which is secure and almost practical. These are the class groups of orders of number fields. These were originally proposed by Buchmann and Williams [**23**] for class groups of imaginary quadratic orders. The protocols used in this situation differ slightly from those described earlier, but the essential features remain the same. In imaginary quadratic orders the elements of the class group can be represented by reduced binary quadratic forms. These forms can be multiplied using the standard composition and reduction algorithms which date back to Gauss (see [**29**] and [**50**]). We shall see in a later chapter that the arithmetic on an elliptic curve and in the Jacobian of a hyperelliptic curve is closely related to this composition of binary quadratic forms.

Such schemes based on class groups are particularly interesting, as breaking some of the proposed cryptosystems is provably as hard as factoring the

discriminant of the order. However, the protocols are at present very slow owing to the complexity of the group operations. For other work on class group based systems, see [**10**], [**20**], [**22**] and [**52**].

There are cryptosystems based on elliptic curves which are provably as hard as known mathematical problems. For example there are systems based on elliptic curves over $\mathbb{Z}/n\mathbb{Z}$, where n is the product of two primes, for which the ability to break the system is as hard as factoring the modulus n (see the work of Meyer and Müller [**101**]). However, Joye and Quisquater [**57**] pointed out that the system of Meyer and Müller is reducible to the system of Rabin and Williams (see [**129**] and [**163**]). Hence, since the Meyer–Müller system is probably slower than the Rabin–Williams system, we shall not discuss the former system further.

There are other systems based on elliptic curves over $\mathbb{Z}/n\mathbb{Z}$, which are in some sense elliptic curve analogues of the RSA scheme (see for example Koyama et al. [**68**]). However, these are not provably as hard as factoring and they appear to offer no advantage over RSA in terms of security but do give a decrease in performance when compared with RSA. These schemes are not discussed further in this book. The reader is referred instead to [**17**], [**58**], [**70**], [**90**], [**121**] and [**159**].

I.3. What it Means in Practice

In this section we discuss the practical implications of using the group $E(\mathbb{F}_q)$ of a suitably chosen elliptic curve over a finite field to implement a DLP-based cryptosystem, as opposed to the more 'conventional' choice of the multiplicative group $\mathbb{F}_p^*$ of a finite field. Notice that, in the comparison, $\mathbb{F}_q$ and $\mathbb{F}_p$ need not be the same field. The key observation is that, for a well-chosen curve (in a sense to be made clear later in the book), the best known method for solving the DLP on $E(\mathbb{F}_q)$ is of complexity exponential in the size $n = \lceil \log_2 q \rceil$ of the field elements, while algorithms that are sub-exponential in $N = \lceil \log_2 p \rceil$ are available for the DLP in $\mathbb{F}_p^*$.

More specifically, the best known general algorithms for the elliptic curve DLP are of complexity proportional to

$$C_{\mathrm{EC}}(n) = 2^{n/2}$$

(see Chapter V).

Define the function

$$L_p(v, c) = \exp\left(c(\log p)^v (\log\log p)^{(1-v)}\right),$$

where 'log' without base specification denotes real natural logarithms. When $v = 1$, the function L_p is exponential in $\log p$, while for $v = 0$ it is polynomial in $\log p$. When $0 < v < 1$, the behaviour is strictly between polynomial and exponential, and is referred to as *sub-exponential*.

Discrete logarithms in $\mathbb{F}_p$ can be found in time proportional to $L_p(1/3, c_0)$, where $c_0 = (64/9)^{1/3} \approx 1.92$, using a general number field sieve method ([**99**,

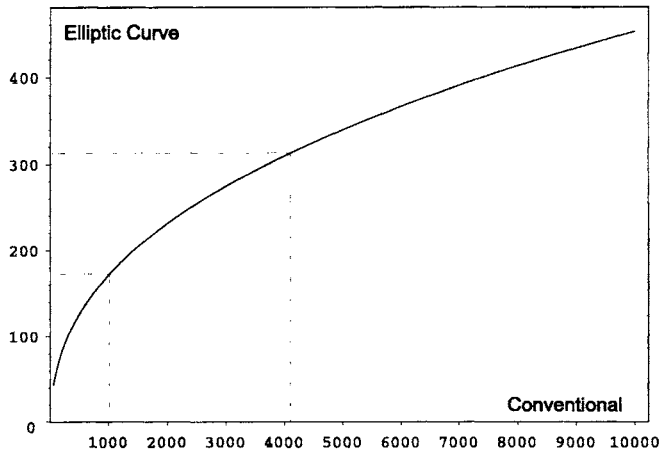

FIGURE I.1. Elliptic curve vs. conventional cryptosystem key sizes (in bits) for similar strength.

Ch. 3] [**114**]). In terms of N, and neglecting constant factors, the complexity is

$$C_{\text{CONV}}(N) = \exp(c_0 N^{1/3}(\log(N \log 2))^{2/3}),$$

where the subscript CONV stands for 'conventional'. Notice that the best known algorithms for *integer factorization* are of roughly the same asymptotic complexity (see [**99**] and [**77**]). Therefore, the discussion and comparisons in what follows apply also to conventional public key cryptosystems based on factorization, e.g., RSA.

Equating C_{EC} and C_{CONV} (and, again, neglecting constant factors in the complexities), it follows that for similar levels of security, we must have

$$n = \beta N^{1/3} (\log(N \log 2))^{2/3}$$

where $\beta = 2c_0/(\log 2)^{2/3} \approx 4.91$. Now, the parameters n and N can be interpreted as the 'key sizes', in bits, for the respective cryptosystems. Therefore, with current algorithmic knowledge, the key size in an elliptic curve cryptosystem grows slightly faster than the *cube root* of the corresponding 'conventional' key size, for similar cryptographic strength.

The relation is plotted in Figure I.1, where the correspondence for 'conventional' key sizes of 1024 and 4096 bits (common values for RSA) has been emphasized with the dotted lines. The equivalent key sizes shown for an elliptic curve cryptosystem are 173 and 313 bits, respectively. Given that various approximations have been used, and various constants neglected, such figures are, of course, approximate and give only general trends. A fair comparison should also take into account the complexity of implementing the cryptosystem. While the implementation of group exponentiation is of about the same

complexity in both cases, in terms of elementary group operations, the group operations themselves are more complex in the elliptic curve case, for the same field size (by a small constant factor – see Chapter IV). Nevertheless, the plot helps explain the recent interest in elliptic curve cryptography as a less expensive alternative to the conventional systems. In practice, shorter key lengths can translate to faster implementations, less power consumption, less silicon area, etc.