A Gentle Introduction to Elliptic Curve Cryptography

Jeffrey L. Vagle
BBN Technologies

November 21, 2000

1 Introduction

Cryptography is the study of hidden message passing. It is also the story of Alice and Bob, their shady friends, their numerous and crafty enemies, and their dubious relationship.

One uses cryptography to mangle a message sufficiently such that only intended recipients of that message can "unmangle" the message and read it. We will call messages to be sent *plaintext* and the mangled message *ciphertext*. The process of converting plaintext to ciphertext is called *encryption*, and the reverse process is called *decryption*.

Cryptographers often use the characters of Alice and Bob to illustrate the methods behind cryptographic systems. A fantastic after-dinner speech was given at the Zurich Seminar regarding the the private lives of Alice and Bob, the text of which may be found at

`http://www.conceptlabs.co.uk/alicebob.html`,

but that's another story.

Our purpose here is twofold: First, to give a brief overview of the nature and mechanics of cryptography, elliptic curves, and how the two manage to fit together. Secondly, and perhaps more importantly, we will be relating the spicy details behind Alice and Bob's decidedly nonlinear relationship.

2 Algebra Refresher

In order to speak about cryptography and elliptic curves, we must treat ourselves to a bit of an algebra refresher. We will concentrate on the algebraic structures of groups, rings, and fields.

2.1 Groups

A group G is a finite or infinite set of elements together with a binary operation which together satisfy the four fundamental properties of closure, associativity, the identity property, and the inverse property. The operation with respect to which a group is defined is often called the "group operation," and a set is said to be a group "under" this operation. Elements $A, B, C, \dots$ with binary operation between A and B denoted AB form a group if:

1. Closure: If $A, B \in G$, then $AB \in G$.

2. Associativity: For all $A, B, C \in G, (AB)C = A(BC)$.

3. Identity: There exists an element I, such that $AI = IA = A$ for all $A \in G$.

4. Inverse: For every $A \in G$ there exists an element $B = A^{-1}$, such that $AB = BA = I$.

2.1.1 Abelian Groups

Abelian groups are groups where all elements in that group commute. That is, for group G, $AB = BA$ for all $A, B \in G$.

Another important type of group is the *cyclic group*. Cyclic groups have at least one element g, the powers of which run through all the elements in the group. The element g is called a *generator*, and will prove quite important later in this paper. We note here that all cyclic groups are abelian, but not all abelian groups are cyclic.

2.2 Rings

A ring is a set S together with two binary operators $+$ and $*$ (addition and multiplication, respectively) satisfying the following conditions:

1. Additive associativity: For all $a, b, c \in S, (a + b) + c = a + (b + c)$.

2. Additive commutativity: For all $a, b \in S, a + b = b + a$.

3. Additive identity: There exists an element $0 \in S$ such that for all $a \in S, 0 + a = a + 0 = a$.

4. Additive inverse: For every $a \in s$ there exists $-a \in S$ such that $a + (-a) = (-a) + a = 0$.

5. Multiplicative associativity: For all $a, b, c \in S, (a * b) * c = a * (b * c)$.

6. Left and right distributivity: For all $a, b, c \in S, a*(b+c) = (a*b)+(a*c)$ and $(b + c) * a = (b * a) + (c * a)$.

This means that a ring is an *abelian group* under addition.

2.3 Fields

A *field* is any set of elements which satisfies the field axioms for both addition and multiplication and is a commutative division algebra

The field axioms are:

Axiom	Addition	Multiplication
Commutativity	$a + b = b + a$	$ab = ba$
Associativity	$(a + b) + c = a + (b + c)$	$(ab)c = a(bc)$
Distributivity	$a(b + c) = ab + ac$	$(a + b)c = ac + bc$
Identity	$a + 0 = a = 0 + a$	$a \cdot 1 = a = 1 \cdot a$
Inverses	$a + (-a) = 0 = (-a) + a$	$aa^{-1} = 1 = a^{-1}a, a \neq 0$

2.3.1 Division Algebras

A division algebra is a ring in which every nonzero element has a multiplicative inverse, but multiplication is not commutative. Like rings, a division algebra is a set S together with two binary operators $+$ and $*$, satisfying the following:

1. Associativity over $+$ and $*$.
2. Commutativity over $+$.
3. Identity elements for $+$ and $*$.
4. Inverse for $+$ and $*$.
5. Left and right distributivity.

As we mentioned earlier, fields are division algebras with the additional feature of being commutative over $*$.

2.3.2 Finite Fields

A finite field is a field with a finite number of elements, also called a Galois field. The order of a finite field is always a prime or a power of a prime. For each prime power, there exists exactly one (with the usual caveat that "exactly one" means "exactly one up to an isomorphism") finite field GF(p^n), often written as $\mathbb{F}_{p^n}$, or simply $\mathbb{F}_q$. That is, for every prime power $q = p^n$, there is a field of q elements that is unique (up to isomorphism).

These finite fields form an abelian group with respect to $*$, and therefore, as we saw earlier, every finite field has a generator g. That is, an integer g exists such that the powers of g exhaust all non-zero residue classes mod p.

2.4 Modular Arithmetic

Modular arithmetic (sometimes referred to as clock arithmetic) is much easier to work with than the "standard" arithmetic we generally use. This system is based in the notion of *congruence* and *residue classes*, which we'll briefly address later (the interesting details of which are beyond the scope of this paper). The main thing to remember is that, rather than using a number "line," (Figure 1) modular arithmetic uses a number "circle" (Figure 2).

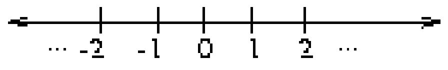

Figure 1: A Number Line

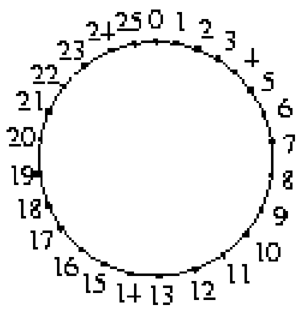

Figure 2: A Number Circle

Therefore, in our 26 element example above, we would start counting at $0, 1, \ldots$ and continue until we get to 25. The next integer after 25 is 0, using modular arithmetic (we've gone completely around our number circle).

Addition in modular arithmetic works the same way as on the number line. For example, to add 15 and 18 using "conventional" arithmetic, we start at 0, count 15 along the number line, then count 18 more, ending up at 33 (Figure 3). If we use modular addition, we "wrap around" our number circle, and end up at 7 (Figure 4).

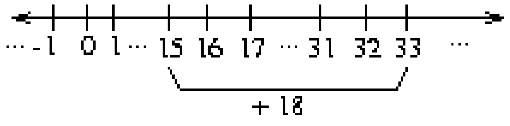

Figure 3: Addition on a Number Line

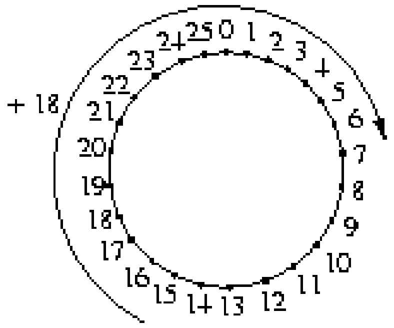

Figure 4: Addition on a Number Circle

Multiplication works in the same fashion (multiplication is, after all, repeated addition). Therefore, $7 * 4 = 28$ using the number line, and is 2 using our modular system.

2.5 Modular Arithmetic, Continued

What's especially cool about modular arithmetic is that it complies with the same axioms as the standard arithmetic we're used to. To explain this, we must briefly touch on the notion of *congruences* and *residues.*

Two numbers a and b are congruent modulo m iff $m|(a - b)$. We can represent this as $a \equiv b \bmod m$.

The set of numbers congruent to a modulo m is denoted $[a]_m$. Therefore, if $b \in [a]_m$, then $m|(a-b)$, as a and b have the same remainder after dividing by m. There are exactly m different sets $[a]_m$, called residue classes (or equivalence classes), which we will refer to as $\mathbb{Z}/m\mathbb{Z}$.

3 Cryptography Basics

Now that we've recovered our mathematical sea legs, let's dive right in to the cryptographic basics.

3.1 Enciphering

In order for a message to go from plaintext to ciphertext, it must be passed through an *enciphering transformation* function, f. These functions take a plaintext message unit and return a ciphertext message unit. I say "unit" here, because cryptosystems will often break up messages into easily digestible chunks for handling. We will assume here that $f : \mathcal{P} \to C$ is injective; that is, there is one (and only one) plaintext message which is the encryption of any given ciphertext message. If this were not so, one could potentially decrypt a given ciphertext into more than one plaintext, resulting in potentially embarassing (and career-threatening) gaffes such as mistaking `BEGIN THE ATTACK` with `EIGHT FOR DINNER`.

3.2 Deciphering

To go from ciphertext to plaintext, we pass the message through a *deciphering transformation*, f^{-1}, the inverse of the enciphering transformation function.

Therefore, a cryptosystem can be represented using the notation:

$$\mathcal{P} \xrightarrow{f} \mathcal{C} \xrightarrow{f^{-1}} \mathcal{P}.$$

3.3 Simple Cryptosystems

In order to begin setting up a cryptosystem, we must "enumerate" the set of all plaintext message units and the set of all ciphertext message units. Obviously, we don't go through and count every one of these; we rely on mathematics to do this for us.

For example, if we choose the *alphabet* of our plaintext and ciphertext message units to be the 26 letter alphabet from $A - Z$, we can use the integers from $0, 1, \ldots, 25$ as their "numerical equivalents." Or, we may wish to include a blank space. We can then use $0, 1, \ldots, 26$, with 26 representing the extra character (or lack thereof).

We could also choose to enumerate message units using other mathematical objects, say, points on a curve. But we are getting ahead of ourselves. Let's stick to integers for now.

3.3.1 An Example

By now, you're probably asking yourself, "So where are all the salacious details about Alice and Bob I've been promised?" Well, we're finally getting to the good stuff now. However, because we're going to start out with some relatively simple examples, our Alice and Bob stories will begin rather tamely.

Alice and Bob wish to devise a method of passing hidden messages back and forth to each other. They begin by defining a standard message unit to be a single letter in our 26 letter alphabet. They can therefore enumerate their alphabet using the integers $0, 1, \ldots, 25$. Their enciphering transformation will be some sort of rearranging of these 26 integers.

N.B. In order to make the computation of enciphering and deciphering a bit more palatable, we can think of the integers above as $\mathbb{F}_{26}$, or $\mathbb{Z}/q\mathbb{Z}$, where $q = 26$ in this case, as we are really interested in the set of residue classes of modulo q. When we do this, we can make use of the incredibly useful properties of finite fields (and their kin) that we saw earlier. Therefore, our cryptographic operations become the operations of addition and multiplication mod q, *i.e.* using modular arithmetic.

3.3.2 Example, Continued

Alice and Bob, using their 26 letter alphabet as above, define $P \in \mathcal{P} = \{0, 1, \ldots, 25\}$ to be the plaintext message unit, where $\mathcal{P}$ is the set of all possible plaintext message units. They now define their enciphering transformation function $f : \mathcal{P} \to \mathcal{P}$ to be

$$f(P) = \begin{cases} P + 5 & \text{if } x < 21 \\ P - 21 & \text{if } x \geq 21 \end{cases}$$

This is just an easier way of representing a function that adds 5 mod 26:

$$f(P) \equiv P + 5 \bmod 26.$$

Usually, the general form of the transformation function looks more like $f : \mathcal{P} \to \mathcal{C}$, where $\mathcal{C}$ is the set of all possible ciphertext message units. However, in this simple example, $\mathcal{P} = \mathcal{C}$.

Therefore, in order to encipher their (already cryptic) message `FOOBAZ`, Alice and Bob first convert from letters to numbers:

$$\texttt{FOOBAZ} \to \texttt{5 14 14 1 0 25},$$

then apply f:

$$\texttt{5 14 14 1 0 25} \to \texttt{10 19 19 6 5 4},$$

and convert back to letters, resulting in the "encrypted" message `KTTGFE`.

To decipher the message, they simply apply f^{-1}, or subtract 3 mod 26, using the same method as above.

This simple cryptosystem is often referred to as the "Caesar cipher," as it was apparently invented by (and used by) Julius Caesar in ancient Rome.

3.4 More Complex Cryptosystems

Obviously, the Caesar cipher would protect Alice and Bob's messages against only the most dimwitted (or uninterested) of opponents. They could improve their simple cryptosystemn by using a transformation of $\mathbb{Z}/N\mathbb{Z}$ called an *affine map*.

3.4.1 Affine Maps

An affine map is any transformation preserving collinearity; that is, all points originally on a line remain on a line after transformation. Affine maps also preserve ratios of distances between points; that is, the midpoint of an original line segment remains the midpoint after transformation. An affine transformation of $\mathbb{Z}/N\mathbb{Z}$ is a map $F : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$ of the general form:

$$F(p) = Ap + q,$$

for all $p \in \mathbb{Z}/N\mathbb{Z}$ where A is a linear transformation of $\mathbb{Z}/N\mathbb{Z}$.

Using modular arithmetic, we define our map as

$$C \equiv aP + b \bmod N,$$

where a and b are fixed integers, together forming the enciphering key.

3.4.2 Example Redux

In order to encrypt the message `FOOBAZ` using our affine transformation, we must first choose an enciphering key. Let's pick $a = 9$ and $b = 24$. Our letter to number conversion yields:

$$\texttt{FOOBAZ} \to \texttt{5 14 14 1 00 25}.$$

When we apply our affine map f, we get

$$\texttt{5 14 14 1 00 25} \to \texttt{17 20 20 7 24 15},$$

which gives us the message `RUUHYP`.

Although this presents a potentially more complex problem for the code-breaker, this cryptosystem is still not all that hard to break. Further complications include the use of *digraphs*, or treating the message as pairs of characters, rather than singletons.

As a matter of fact, if we deal with our message in k-graphs, where $k > 2$, we make the job of breaking the system quite difficult for the attacker, since the number of possible k-letter blocks is very large. Of course, their are time- and space-related issues to be dealt with when deriving these cryptosystems, but we won't concern ourselves with that here.

3.5 Classical Cryptosystems

The cryptosystems we have been describing thus far are referred to as *classical* (or *symmetric* or *private key*) cryptosystems. The sets $\mathcal{P}$ and $\mathcal{C}$ were fixed because N was fixed, and the enciphering key K_E was the pair (a, b). The deciphering key K_D was also obtained by that same affine map inverted:

$$P \equiv a^{-1}C - a^{-1}b \bmod N.$$

That is, they are cryptosystems in which, once the enciphering algorithm is known, the deciphering algorithm can be derived in roughly the same order of magnitude of time as the enciphering algorithm, as all encipherings and decipherings are based around the same key. Of course, deciphering can sometimes take a bit longer because of the problems of finding an inverse modulo N (using the Euclidean algorithm, for instance). But this is a one-time-only expense, as it is necessary only in order to find the decryption key.

We now move into the realm of *public key* cryptography, where, if the enciphering time were polynomial in N, and the deciphering time based on knowledge of the encryption key were 2^N (a considerable difference).

4 Public Key Cryptography

An important factor in the security of cryptosystems is the assumption that all enciphering and deciphering algorithms are publicly known. This means that the secrecy comes from the keys themselves: we are free to change keys, revoke keys, and keep keys secret.

As we've already seen, using classical (private key) cryptosystems, once a user knows K_E, it is rather easy to derive K_D. Using public key cryptography, a user who knows how to encrypt a message cannot use K_E to find K_D. Well, cannot implies impossibility, and that's not entirely true. Public key cryptography relies on *trapdoor functions*: functions that are fairly easy to compute on their own, but f^{-1} would take a prohibitive amount of computation to derive without K_D. Sounds dubious, doesn't it?

Let's take a closer look at the mathematics behind public key cryptosystems.

4.1 Trapdoor Functions

Trapdoor functions are closely related to *one-way* functions, in that they are easy to compute, but f^{-1} is not without some sort of additional information.

An early example of such a function had the passwords and their encrypted forms were integers modulo a large prime p, with the "one-way" function $f : \mathbb{F}_p \to \mathbb{F}_p$ given by a polynomial $f(x)$ which was easy enough to evaluate, but took an unreasonably long time to invert.

N.B. The word "unreasonable" doesn't often show up as a basis for rigorous mathematical proof. This brings up the dark secret of modern cryptography: what is empirically considered "realistically computable" is key. This is not, of course, some sort of absolute truth, but is affected by advances in technology and science. Therefore, what is considered *strong cryptography* today, may be child's play to break in 10 years (see *quantum computing*).

That said, let's look on the bright side: the possibility exists that some trapdoor function may someday be *provably* one-way. Until that day, however, we must face the fact that there are no provable public keys, and Alice and Bob must rely on empirical evidence, only (a fact which seems to bother natural scientists not at all).

4.2 Public Key Cryptography: the Basics

The reason this sort of cryptography is called "public key" is because key required to encrypt message units, K_E, can be made publicly available. Because of the computational infeasability of decrypting a message with knowledge of K_E only, this idea works. Therefore, when Alice wishes to pass a hidden message to Bob without their nemesis Charles being able to read the message, she simply obtains Bob's encryption (public) key K_{E_B} and uses the enciphering function on the message with this key. Now, only Bob will be able to (easily) read this message, as he has kept his decryption (private) key K_{D_B} close to his vest.

Without this information, Charles has a considerable task in front of him if he wants to compute the inverse of the encryption function (and obtain the plaintext), unless he has knowledge of some new mathematical method of computing inverses. If this is the case, all bets are off. However, we will go on assuming that he got Cs in math and physics in college, and is really not all that clever after all.

But why is all this necessary in the first place? Couldn't Alice and Bob simply meet and agree upon some shared (symmetric) key, and use conventional methods to hide their messages? This is, of course, possible, but certain facts about their lifestyles may prevent this. For instance, what if Alice and Bob have never actually met? What if they have never even heard each other's voice over the telephone? What if they really don't trust each other all that much? These sorts of problems, apart from what they might say about Alice and Bob's odd relationship, are just the sorts of things public key cryptography is good for.

4.3 Why Public Key Cryptography?

Until recently, most users of cryptography were military and/or diplomatic organizations who, by their very nature, were a small, finite number of individuals who would share a system of keys distributed internally.

The relatively recent advent of computer network communication has changed the nature of the average user of cryptography. Now, every time you order that book from some .com, do your banking online, or electronically sign your email, your are using some sort of cryptography. Because we may require secure communications with many different parties, these parties constantly changing, the use of classical cryptography quickly becomes unweildy in all but the smallest of networks.

Therefore, new requirements are made of cryptosystems, such as *authentication*, *non-repudiation*, *message integrity*, and *distributed trust*, which go beyond mere message hiding.

4.3.1 Authentication

Using public key cryptography, one can identify oneself in such a way as to eliminate the possibility that someone has usurped your identity. To illustrate this, we again turn to Alice and Bob.

Alice and Bob have never met, yet somehow (the details are not important here) have managed to have safely obtained each other's public key. They have previously agreed upon an enciphering transformation f, where Alice must use f_B (the enciphering transformation using Bob's public key) to encrypt a message to Bob, and Bob must use f_A to send a message to Alice. As in our earlier examples, we will assume that $\mathcal{P} = \mathcal{C}$ and are the same for both Alice and Bob.

Let's say Alice has come up with some sort of signature S. If Alice wishes to identify herself to Bob, simply sending Bob the message $f_B(S)$ would not do, since K_B is publicly available, and anyone could send such a message. However if Alice includes in that same message the sub-message created by the composite function $f_B f_A^{-1}(S)$, when Bob decrypts the message, he will find an unencrypted part remains $(f_A^{-1}(S))$. Since the person claiming to have sent the message was Alice, Bob can apply Alice's public key, f_A, to the remainder and end up with S. It is impossible for anyone but Alice to have sent this message, as Alice is the only person with access to f_A^{-1}. This bit, in effect, becomes Alice's unforgeable *digital signature.*

4.3.2 Non-Repudiation

Since Alice and Bob have never met, we can safely assume that their level of trust is tenuous, at best. Alice wants to be assured that any message sent by Bob (identified through his digital signature) could not later be falsly denied by Bob when the Secret Police (or the bank, IRS, their spouses, etc.)

somehow find the message and wish to hold someone accountable for its content. This is called non-repudiation and is extremely important in the use of public key cryptography.

For instance, if two parties enter into some sort of electronic contract, both parties have a vested interest in the binding of such a document, and should things go south, would want to prove the identity of the other. Also, the use of anonymous, digital cash has been studied recently, that is, money that has no physical counterpart, but only exists as bits on a machine. Before accepting this digital money from a customer, a merchant would certainly want some method of verifying that the government actually did issue this electronic note.

The same authentication mechanism described in Section 4.3.1 also works for this purpose. Since no one but Bob could have possibly signed his messages, it could certainly be proven, given some additional infrastructure, that the message did indeed originate from Bob.

4.3.3 Message Integrity

Another concern of Alice's (and probably Bob's, too, since the spectre of accountability has raised its ugly head) is the possibility that someone (probably Charles) could be intercepting their messages, altering them to serve his purposes, and passing them on as if nothing had happened. If this task proved too difficult for the not-too-bright Charles, he could simply intercept the message and send along his own replacement, instead. How can Alice and Bob verify that the messages that they are receiving have passed over the network unmolested?

Again, the notion of authentication stands in Charles's way. Any message passed back and forth between Alice and Bob could first be checked for authenticity. If Alice receives a message from Bob, and she verifies it is indeed from him by checking his digital signature, there is no way it could have come from anyone else, so we've eliminated the possibility that Charles has intercepted Bob's message and passed along one of his own with a forged signature.

However, what can Alice do to assure herself that Charles has not simply twiddled a few bits in the original message, retaining Bob's signature throughout? This requires more information about the message to be contained within the signature, and we do this through the use of *hash functions*

4.3.4 Hash Functions

The hash function plays a key role in modern cryptography. It is a map $h : x \to y$ going from a variable-length input x to a fixed-length (and generally smaller) output y. This function has the property where it is computationally infeasible to find two different inputs x and x' such that $h(x) = h(x')$. That is not to say that the function must be injective, necessarily, only that finding multiple unique inputs for the same output is a hard problem.

If, when Alice signs her message to Bob, she includes in her signature S the value of $h(x)$, where x is the entire text of her message, Bob can verify not only that the message in question was not only sent by Alice, but that the contents of which were not tampered with by Charles, as Charles would not have been able to change x without changing the value of $h(x)$.

5 The Diffie-Hellman Key Exchange System

The first public key cryptosystem ever invented was Diffie-Hellman. Because the computation behind public key cryptosystems takes a relatively long time when compared to classical cryptosystems, public key cryptography is often used in a modified role along with a symmetric cryptosystem to transmit hidden messages. In 1976, Diffie and Hellman [2] provided a detailed method of agreeing upon a key for a symmetric system using public key methods, the mathematics of which were based on the *discrete log problem*.

5.1 Discrete Log

As we discussed earlier, modern cryptography is based around the idea of the one-way function. Finding a good example of one of these functions is key in the creation of a successful cryptosystem. One of the more important examples of a one-way function is the exponential function in a large finite field.

As anyone who has ever done some advanced arithmetic can attest, it does not seem any easier to find x^y than it is to find the inverse, $log_x y$, when working in $\mathbb{R}$. This is not a good candidate for a one-way function. However, if we find a finite field like $(\mathbb{Z}/q\mathbb{Z})^*$ or $\mathbb{F}_q^*$ (the $*$ adds the group multiplication operation - an abelian group), there are known methods (such as repeated-squaring) for computing x^y for large x fairly quickly. However, given an element z of the form x^y, computing $y = log_x z$ is considerably more difficult. This problem is called "discrete" because of the use of finite fields (as opposed to $\mathbb{R}$, which is continuous).

Therefore, given a finite group G, $x, z \in G$, z a power of x, the discrete logarithm of z to the base x is any integer y such that $x^y = z$.

5.2 Diffie-Hellman

The algorithm behind Diffie-Hellman is fairly straightforward. The idea is to generate a private key that can later be used for communication, and sharing it in a secure fashion.

The first step in the algorithm is for Alice and Bob to choose a random integer over some interval defined by the affine transformation provided by the symmetric key (don't worry too much about the details). It turns out Alice and Bob can choose this element from a finite field of the same size as the interval prescribed. For example, if we need a random integer $0 < i < N^{13}$, and we have a finite field $\mathbb{F}_p$ of prime p elements, we can let an element of $\mathbb{F}_p$ correspond to an integer from 0 to $p - 1$ (using modular arithmetic mod N^{13}). There is a similar mechanism for finding a random element in $\mathbb{F}_q$, $q = p^k$.

In order to generate the Diffie-Hellman random element in a large finite field $\mathbb{F}_q$, we begin by assuming the value of q is publicly available. We also have some fixed $g \in \mathbb{F}_q$, where g is a generator of $\mathbb{F}_q^*$, and is also public. It is important that g be a generator, as when we generate a key from $\mathbb{F}_q$, it will be only from the powers of g. If g is a generator, a random element of $\mathbb{F}_q^*$ could be *any* element in $\mathbb{F}_q^*$.

5.3 Diffie-Hellman, Continued

We return now to Alice and Bob and their hidden message dilemma. They have decided to use Diffie-Hellman to arrive at a symmetric key with which to transact their private business.

As discussed earlier, they key in question will be a random element of $\mathbb{F}_q^*$, where q is public knowledge. Alice begins by secretly choosing an arbitrary integer $1 < a < q - 1$, and then publicly computing $g^a \in \mathbb{F}_q$. Bob also secretly chooses $1 < b < q - 1$ and publicly computes $g^b \in \mathbb{F}_q$. Thus, the secret Diffie-Hellman key becomes g^{ab}, a key which both Alice and Bob can easily compute using the information they hold privately along with their contemporary's public information.

The only information Charles has to work with is g^a and g^b. Diffie-Hellman works based on the assumption that it is an intractible problem to compute g^{ab} if one only knows g^a and g^b.

N.B. You can see why it was so important to choose g a generator for $\mathbb{F}_q^*$, as powers of g can be any element in the field, increasing the size of the keyspace, and making life difficult for Charles.

5.4 Scratching the Surface

Obviously, Diffie-Hellman is only one particular use of public key cryptography (albeit a modified one), chosen here for its relatively straighforward algorithm. There are many others that approach the one-way function problem in varying ways.

RSA [1], for instance, relies on the careful choice of two very large prime numbers p and q, and computes $n = pq$ which is used to create (using the Euler phi-function) the user's public and private keys. The word "careful" is used here because it is important that the primes p and q are not easy to guess (*e.g.* by looking them up in a table of primes), or relatively easy to derive (*e.g.* by looking for Mersenne primes, or factors of $b^k \pm 1$ for small b and k).

The security of RSA is based on the rather old problem of finding the complete factorization of a large composite integer, not knowing the prime factors in advance. RSA as a cryptosystem lends itself to more of the problems addressed in Section 4.3, and is used as such in many current applications.

For more information on RSA and other public key cryptosystems, I highly recommend further reading in [7], and for a more advanced treatment of the mathematics see [4].

6 Elliptic Curves

Here we finally begin to delve into the algebra and arithmetic of elliptic curves, and what makes them so useful in the realm of cryptography.

We'll begin our exploration by taking a look at the elementary facts about elliptic curves, and follow up with their cryptographic applications.

6.1 Elliptic Curve Basics

An elliptic curve over a field K is a cubic curve in two variables, $f(X, Y) = 0$, creating the set of points $(x, y) \in K$ satisfying the equation

$$y^2 = x^3 + ax + b.$$

These points, along with a single element O called the "point at infinity," make up an elliptic curve.

For the purposes of our discussion here, K will be either the field $\mathbb{R}$ (real numbers) or the finite field $\mathbb{F}_q$ of $q = p^r$ elements.

The general (Weierstrass) form of the elliptic curve is:

$$y^2 + ay = x^3 + bx^2 + cxy + dx + e; a, b, c, d, e \in K.$$

6.2 Arithmetic of Elliptic Curves

What makes elliptic curves so darned important with respect to cryptography is that the set of points on an elliptic curve form an abelian group. To show how elliptic curves satisfy the fundamental properties of groups, along with commutativity, we will explore the geometry of these curves.

6.3 Elliptic Curves and Abelian Groups

Let's begin by letting E be an elliptic curve over $\mathbb{R}$, with $P, Q \in E$. We can now define the arithmetic of these curves using a few rules.

6.3.1 Additive Identity

If the point P is the "point at infinity" O, then $-P = 0$ and $P + Q = Q$. This makes O the additive identity (like 0) for the group of points on E.

6.3.2 Additive Inverse

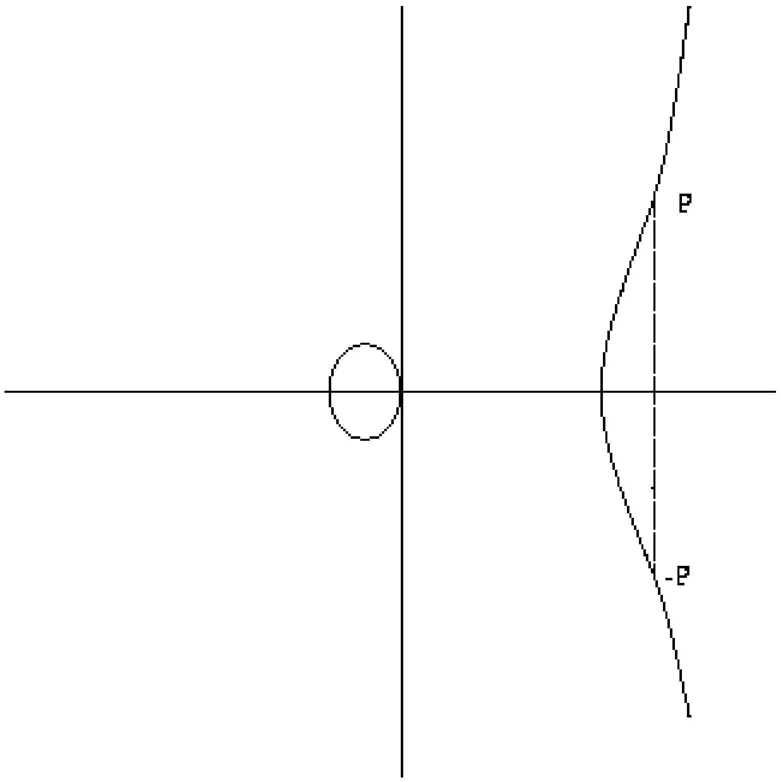

Figure 5: Elliptic Curves: Additive Inverse

Assuming $P \neq O$ (as we will for the remainder of these definitions), we define $-P$, where $P = (x, y)$ to be $(x, -y)$. A quick look at the general formula for elliptic curves verifies that $(x, -y) \in E$ iff $(x, y) \in E$.

6.3.3 Addition of Points on E

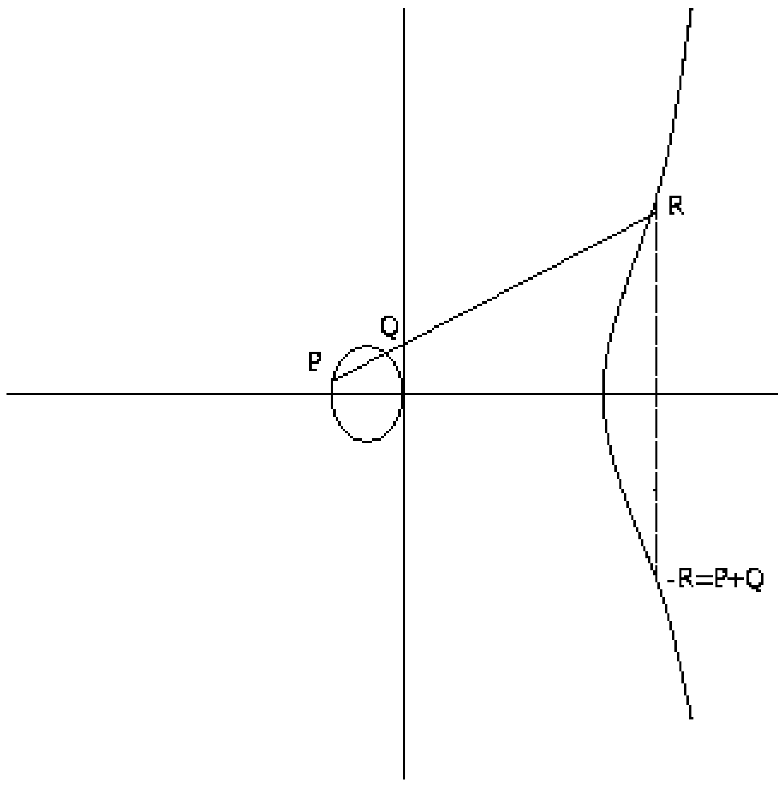

Figure 6: Elliptic Curves: Addition

When P and Q have different x-coordinates, then there is a line $l = \overline{PQ}$ that intersects the curve at exactly one point R. If l is tangent to the curve at P or Q, then $R = P$ or $R = Q$, respectively. $P + Q$ is therefore defined to be $-R$ (Figure 6).

6.3.4 Addition of Points, Case 2

If $Q = -P$, then we define $P + Q$ to be O, the "point at infinity."

6.3.5 Addition of Points, Case 3

If $P = Q$, then l is the tangent line to the curve at P, and R is the only other point of intersection of l with E, and we define $P + Q = -R$. If P is a point of inflection, then $P = R$.

6.3.6 E Forms an Abelian Group

The geometric argument is just one way to prove that the definitions we have just laid out for $P + Q$ makes the points on E an abelian group. One could also use real analysis or an algebraic argument, as well. For further reading on the topic, along with the complete proofs, see [9].

6.4 Elliptic Curves over Finite Fields

Now, let K be the finite field $\mathbb{F}_q$, where $q = p^r$, and E is an elliptic curve defined over K.

As we have seen in earlier discussions, it is important to cryptographers that a group have a finite number of points. Finding these points therefore becomes an important task.

E has at most $2q + 1$ $\mathbb{F}_q$ points: $2q$ pairs of (x, y) along with O, the point at infinity. Counting points on elliptic curves is important to cryptographers using these curves, as it's nice to know the structure of the abelian group; *i.e.* is it cyclic. Hasse's Theorem deals with the size of N, the number of $\mathbb{F}_q$ points on E, but is beyond the scope of this paper - don't worry too much about it. Again, if you wish to explore the mathematics a bit further, [9] is an excellent source.

7 Elliptic Curve Cryptosystems

We now have a multiplicative group of a finite field $\mathbb{F}_q$, the finite abelian group $\mathbb{F}_q^*$. As with our conventional public key cryptographic examples, we can use this abelian group to form a public key cryptosystem.

As you recall, the discrete log problem formed a nice one-way function for use in our cryptosystem. It would be nice if we could find an analogous problem using elliptic curves over finite fields.

7.1 Multiplication of points on E

Earlier, we would multiply elements of $\mathbb{F}_q^*$ in our discrete log problem. Using elliptic curves, we now add points on an elliptic curve over a finite field. Therefore, as we would raise an element to a power k through repeated multiplication k times, we now use repeated addition of an element k times (multiplication). The method of repeated doubling works quite nicely for this.

7.1.1 Repeated Doubling

Let P be a point on the elliptic curve E defined over $\mathbb{F}_q^*$, and let $k = 200$. In order to find kP, we can use the method of repeated doubling to do this:

$$200P = 2(2(2(P + 2(2(2(P + 2P)))))).$$

Using this method we only end up performing 7 doublings and 2 additions of P.

7.1.2 Optimization

This is where knowing the number of points N on our curve E can be important. If we know N, we can reduce the time estimates for repeated addition of points on E by orders of magnitude through certain clever algorithms. For details on this, see [4].

7.2 Labeling Plaintext Messages in E

As in our earlier examples, we need a method for encoding plaintext message units in E. Our method earlier involved simply translating the 26 letter alphabet to the integers $0, 1, \dots, 25$. We want a method like this.

Unfortunately, there are no known polynomial time algorithms for finding a large number of points on an arbitrary E over $\mathbb{F}_q^*$. We are not simply looking for random points on E, here. As in our previous examples, we want a systematic way of finding points on E relating somehow to the plaintext message p, so that going back and forth between them is made tolerable (to a computer, at least). Therefore, we are forced to use probablistic algorithms to do this, where the chance of failure is acceptably small.

7.3 Elliptic Curves and Discrete Log

Using the definitions above, we can now describe an analog to the discrete log problem using elliptic curves over finite fields.

Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$, and let B be a (base) point on E. The discrete log problem on E to the base B becomes: given a point $P \in E$, it is an intractable problem to find an integer $x \in \mathbb{Z}$ such that $xB = P$, if x exists.

It turns out that the elliptic curve analog of the discrete log problem may be a harder problem than the original. The standard techniques used on ordinary finite fields to solve the discrete log problem do not work over elliptic curves.

This issue becomes especially important when considering the reality of implementation issues. Because of software and hardware constraints, it is often easiest to construct the discrete log problem over finite fields $\mathbb{F}_{2^r}$, *extension fields* of $\mathbb{F}_2$. There are a number of specialized methods for solving the discrete log problem in $\mathbb{F}^*_{2^r}$, making the job of Charles easier unless r is chosen to be quite large. Because these methods do not work for solving the elliptic curve analog of discrete log, r can be chosen to be significantly smaller, allowing for easier implementation.

8 Elliptic Curve Diffie-Hellman Analog

Again, we return to the problems of Alice and Bob in order to illustrate how the elliptic curve discrete log analog allows us to implement the Diffie-Hellman key exchange protocol using elliptic curve cryptography.

The first step, as in the original, is for Alice and Bob to choose a finite field $\mathbb{F}_q$, which will be made public. They also choose an elliptic curve E defined over $\mathbb{F}_q$, which is also public information. In order to construct the key to be used with a symmetric cryptosystem, Alice and Bob must arrive at a random point P on E. Using P, they can take its x-coordinate, which in itself is a random element of $\mathbb{F}_q$. Since $q = p^r$, Alice and Bob can convert

q into a random r-digit base-p integer. This will be the key they will use in their symmetric cryptosystem.

How will Alice and Bob choose P?

8.1 ECC Diffie-Hellman, Step One

The first step in arriving at P is for Alice and Bob to publicly choose a point B on E which will serve as their base. B will serve as a "generator" in our field, as g served in our earlier example. The word "generator" is in quotes here, because we do not require that B be an actual generator for $\mathbb{F}_q^*$. As a matter of fact, $\mathbb{F}_q^*$ may not even be cyclic. We would, however, prefer that B generate a sufficiently large subgroup, whose order is very large (N or a large divisor of N).

8.2 ECC Diffie-Hellman, Step Two

Alice now begins the generation of her secret key by choosing a random $a \in \mathbb{Z}$, of approximately the same order of magnitude as N. Then, she can publicly compute $aB \in E$. Bob also privately chooses a random $b \in \mathbb{Z}$, and publicly computes $bB \in E$.

Therefore, the secret key is computed as

$$P = abB \in E,$$

which can be easily computed by either Alice or Bob, since they are in possession of their private key and the public key of their partner. Charles, on the other hand, only has knowledge of aB and bB. Without being able to solve the elliptic curve discrete log problem, it is impossible for him to derive abB without knowing a or b.

9 Further Study

There has been a considerable amount of handwaving here in the interests of scope. However, in order to obtain more detail and clarity to this picture, I suggest some further exploration.

For a (very) good introduction to cryptography as a whole, please see [7]. It has a nice, well-rounded treatment of the subject, ranging from the math to actual code examples to the politics of encryption.

If you require more mathematical detail in your diet, you would be well-served to read either [4] (for an introduction to the mathematics of cryptography) or [9] (for a detailed look at the mathematics of elliptic curves). Or both.

Whichever texts you choose to explore, you will be happy to know that the adventures of Alice and Bob continue to be chronicled therein, along with their contemporaries Aniuta and Björn (and Aïda and Bernardo, and …).

References

[1] L.M. Adleman, R.L. Rivest, and A. Shamir. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[2] W. Diffie and M.E. Hellmann. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[3] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48, 1987.

[4] N. Koblitz. *A Course in Number Theory and Cryptography.* Number 114 in Graduate Texts in Mathematics. Springer-Verlag, second edition, 1994.

[5] S. Lang. *Algebra.* Addison-Wesley, 1984.

[6] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications.* Cambridge Univ. Press, 1986.

[7] B. Schneier. *Applied Cryptography.* John Wiley and Sons, second edition, 1996.

[8] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44, 1985.

[9] J. Silverman. *The Arithmetic of Elliptic Curves.* Number 106 in Graduate Texts in Mathematics. Springer-Verlag, 1986.