Securing Data Transfer with Elliptic Curves

To what extent can elliptic curves be used to establish a shared secret over an insecure channel?

Mathematics

Word count: 3809

Contents

Introduction

As Alice tries to talk to Bob through her laptop, an adversary named Eve tries to eavesdrop their communication and steal sensitive information. Alice and Bob decide to use Diffie-Hellman Key Exchange, which allows them to establish a password for future communication, even if Eve can intercept all of their communication.

Diffie-Hellman is an important element within the collection of cryptographic methods that together bullet-proof Internet connections. 99.3% of the top 1 million websites prefer using Diffie-Hellman over others when it comes to establishing a shared secret [1].

To understand the effectiveness of Diffie-Hellman, we can compare it to a different scheme named *public key cryptography* which can also help Alice secure her messages with Bob.

With public key cryptography, Alice can use Bob's public key to encrypt a message that only Bob can decrypt using his private key. Bob can then use Alice's public key to send a message that only Alice can decrypt with her own private key, which facilitates secure communication [2].

There are some disadvantages to this method. One is that encryption using public keys is often slower than *symmetric cryptography*, where if Alice and Bob share a password, they can both encrypt their messages with the same password. Because using a password is faster, Alice can first use Bob's public key to send a password, then together opt to use password based encryption instead.[2]

Suppose Eve kept a record of all encrypted messages sent between Alice and Bob, and Eve obtains Alice's private key in some way in the future. Eve will be able to decrypt all the passwords, and therefore all the messages. On cryptographic terms, we say that this method does not have *forward secrecy*.[3]

With Diffie-Hellman Key Exchange, Alice and Bob can quickly establish a password without using any public and private keys, and Eve would still be unable to obtain the password just from inspecting the communication. This is great for Alice, as she can generate a password each time she talks to Bob. If Eve ever finds out the password for one of their messages, she would not be able to decrypt the other messages.[3], [4]

Diffie-Hellman Key Exchange is designed specifically so that people can establish a shared secret (the password between Alice and Bob) over an insecure channel (a communication method that Eve can eavesdrop).

Diffie-Hellman takes on different forms. There is Finite Field Diffie-Hellman and Elliptic Curve Diffie-Hellman. To answer our research question, we'll look at both techniques and compare the two methods in terms of how efficient they are (how much data does Alice and Bob need to send to each other?) and how fast they are (how quickly can Alice and Bob calculate the shared password in an exchange?)

Group Theory

$\mathbb{Z}_p^\times$: The Multiplicative Group Over a Prime

Fermat's Little Theorem suggests the following to be true for any integer a and prime p:

$$a^{p-1} \equiv 1 \pmod p$$

Extracting a factor of a, we get

$$a \cdot a^{p-2} \equiv 1 \pmod p$$

Thus, under multiplication modulo p, any non-zero integer a multiplied by a^{p-2} results in 1. As 1 is the multiplicative identity $(1 \cdot x = x)$, a^{p-2} is said to be a's *multiplicative inverse*. Consider the set of numbers from 1 to $p-1$. Every number a has a multiplicative inverse a^{p-2} modulo p; The set contains an identity element (1); Multiplication is associative $(a \cdot (b \cdot c) = (a \cdot b) \cdot c)$; And each multiplication will always result in a number between 1 to $p-1$ since it is performed modulo p (closure). These properties, existence of an identity element and inverses, associativity, and the closure of operations, are exactly the properties that define a group.

A group is, at its core, a set. We'll use some of the same language with sets, for example the $\in$ symbol and the word *element*. We will refer to the specific group we discussed above as $\mathbb{Z}_p^\times$. The subscript is the *modulus* of operations, while the superscript specifies the operation. In a similar vein, $\mathbb{Z}_p^+$ refers to the same set of numbers (though also including the number 0, unlike multiplication), but specifies addition as its group operation.

The *order* of a group refers to the number of elements in that group. For $\mathbb{Z}_p^\times$, the order is $p-1$ since the elements are $1, 2, ..., p-1$. Using notation, we write $|\mathbb{Z}_p^\times| = p-1$. As per above, the additive group includes zero, therefore $|\mathbb{Z}_p^+| = p$.

The *order* of a specific element x, refers to the smallest integer k such that $x^k = 1$, where 1 is the identity element. For example, the order of 17 in $\mathbb{Z}_{1009}^\times$ is 1008, because $17^{1008} = 1$ and 1008 is the smallest smallest exponent to give 1, whereas the order of 2 in the same group is 504, since $2^{504} = 1$ and 504 is the smallest exponent. Therefore, we have $|17| = 1008$ and $|2| = 504$.

The Discrete Log Problem

Under a specific group $\mathbb{Z}_{1009}^\times$, we are asked to find an integer n for which $17^n = 24$. In this case,

$$17^{456} \equiv 24 \pmod{1009}$$

Therefore $n = 456$ is the solution to this question. More generally, the discrete log problem (DLP) asks for a smallest exponent n for a given group g and its elements a and b such that

$$a^n = b$$

Given this problem, one might take the brute-force approach, repeatedly performing the group multiplication, calculating a^2, a^3, a^4 and and comparing each with b. In the example problem, it would take 455 multiplications before finally arriving at the answer. Assume the algorithm is tasked to solve questions of this kind repeatedly with the exponent n taken at random. This algorithm would take on average $\frac{1}{2}|a|$ operations. As the order $|a|$ gets big (towards numbers as big as 2^{200}), this approach quickly becomes infeasible.

The assumption that the discrete log cannot be solved trivially in specific groups is the core of cryptographic protocols and algorithms. There are known techniques better than a brute-force search which can solve the discrete log problem either for specific groups or for all groups in general, which are described in later sections. Modern cryptography uses larger, more secure groups to make those attacks infeasible.

Finite Field Cryptography and Attacks

The term *finite field* refers to the fact that the set of numbers from 0 to $p-1$ with p prime, forms a group under addition modulo p while 1 to $p-1$ forms a group under multiplication modulo p. Moreover, multiplication is distributive over addition: $a(b+c) = ab + ac$ mod p. A field in general is a set, whether finite or infinite, that satisfies these three conditions [5], [6].

The field of integers modulo p (a finite field) is written as $\mathbb{F}_p$. We use $\mathbb{F}_p^\times$ to explicitly refer to the multiplicative subgroup (equivalent to $\mathbb{Z}_p^\times$ used above) [7].

Diffie-Hellman Key Exchange

Building off the previous example, suppose we're given the numbers 407 and 24, which are both exponents of a known base 17 in $\mathbb{F}_{1009}^\times$. Let's let

$$17^a \equiv 407 \pmod{1009}$$
$$17^b \equiv 24 \pmod{1009}$$

Is it possible for us to find 17^{ab}? If we know the value of $a = 123$, we can raise 24 to 123.

$$\begin{aligned} 17^b &= 24 \qquad\qquad a = 123 \\ 17^{ab} &= \left(17^b\right)^a = 24^{123} = 578 \end{aligned}$$

More generally, if we know 17^a and the exponent b, or if we know 17^b and the exponent a, it would be possible for us to know 17^{ab}. But just being given 17^a and 17^b in this case would make it less trivial.

This problem of finding x^{ab} when just given x, x^a, and x^b is named the Diffie-Hellman problem, and it is not hard to see that the difficulty of this problem relates to the difficulty of the Discrete Log Problem, since solving the Discrete Log would give us the answer. Assuming the Diffie-Hellman problem is difficult, we can use this to setup a cryptographic exchange.

Consider the following case where anything sent between Alice and Bob can be seen by Eve. Alice knows that $17^{123} \equiv 407$, but only sends Bob 407. Bob knows that $17^{456} \equiv 24$, but only sends Alice 24. After exchanging their information, Alice can compute $24^{123} \equiv 578$, and Bob can also compute $407^{456} \equiv 578$. Eve, only intercepting the numbers $17, 407, 24$ in their communication, is unable to calculate the secret number 578 without solving the Diffie-Hellman problem.

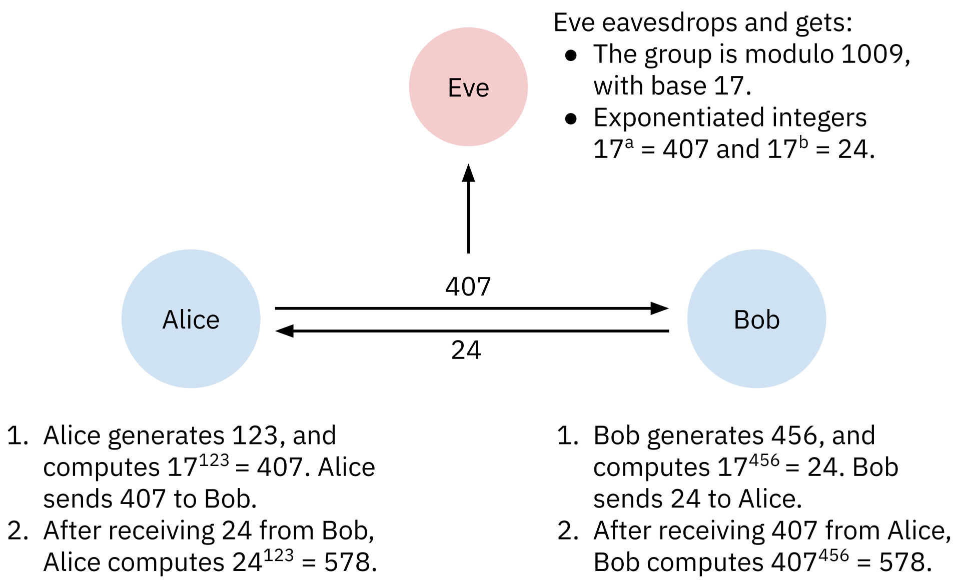
Eve eavesdrops and gets:
The group is modulo 1009, with base 17.
Exponentiated integers 17^a = 407 and 17^b = 24.
Eve
407
24
Alice
Bob
1. Alice generates 123, and computes 17^123 = 407. Alice sends 407 to Bob.
2. After receiving 24 from Bob, Alice computes 24^123 = 578.
1. Bob generates 456, and computes 17^456 = 24. Bob sends 24 to Alice.
2. After receiving 407 from Alice, Bob computes 407^456 = 578.

Figure 1: Diagram by author, a description of the Diffie-Hellman Key Exchange process.

To generalize, the Diffie-Hellman Key Exchange utilizes the difficulty of the Diffie-Hellman problem. Under an agreed upon group G and base $x \in G$, two parties Alice and Bob can establish a shared secret. Alice can generate an exponent a and send Bob x^a, while Bob can generate a secret exponent b and send Alice x^b. Together, they can both compute x^{ab} as their shared secret securely, even if Eve is able to intercept this communication.

The specific example we did is done in $\mathbb{F}_{1009}^{\times}$. In reality, the size of the field will be much larger to prevent anyone from trivially breaking the exchange and finding the secret [8], [9]. The more general technique of performing Diffie-Hellman on the multiplicative subgroup of a finite field is called Finite Field Diffie-Hellman. We will describe Elliptic Curve Diffie-Hellman later, but first, we can consider ways to break the current example.

Index Calculus

Diffie-Hellman Key Exchange on Finite Fields normally uses groups $\mathbb{F}_p$ where $2^{2048} \leq p \leq 2^{8192}$ [8], [9]. The large size of the prime ensures that solving DLP is inefficient. Below we will describe Index Calculus, which efficiently solves DLP for smaller finite fields.

It is best to illustrate with an example. We'll reuse the one presented earlier:

$$17^n \equiv 24 \pmod{1009}$$

To find n, we first define a logarithm function L. $L(x)$ is defined such that

$$17^{L(x)} \equiv x \pmod{1009}$$

Therefore, our goal is to find $L(24)$.

Note that when we have

$$17^{L(x)+L(y)} \equiv 17^{L(x)} \times 17^{L(y)} \equiv xy \equiv 17^{L(xy)} \pmod{1009}$$

So then

$$17^{L(x)+L(y)-L(xy)} \equiv 1 \pmod{1009}$$

Because $|17| = 1008$, we have (notice the change in modulus from 1009 to 1008)

$$\begin{aligned} L(x)+L(y)-L(xy) &\equiv 0 \pmod{1008} \\ L(x)+L(y) &\equiv L(xy) \pmod{1008} \end{aligned}$$

As such, we have a relation analogous to the laws of logarithm on real numbers. Since every number can be factorized into primes, the idea is to obtain $L(p)$ for small primes p, then figure out $L(24)$ afterwards. We first try to factorize exponents of 17 into relatively small primes:

$$\begin{aligned} 17^{15} &\equiv 2^2 \cdot 5 \cdot 13 \pmod{1009} \\ 17^{16} &\equiv 2^7 \cdot 3 \pmod{1009} \\ 17^{24} &\equiv 2 \cdot 11^2 \pmod{1009} \\ 17^{25} &\equiv 2 \cdot 3 \cdot 13 \pmod{1009} \\ 17^{33} &\equiv 2^2 \cdot 3^2 \cdot 11 \pmod{1009} \\ 17^{36} &\equiv 2^2 \cdot 7^2 \pmod{1009} \end{aligned}$$

Applying L to both sides of the equations, we obtain

$$\begin{aligned} 15 &\equiv 2L(2)+L(5)+L(13) \pmod{1008} \\ 16 &\equiv 2L(7)+L(3) \pmod{1008} \\ 24 &\equiv L(2)+2L(11) \pmod{1008} \\ 25 &\equiv L(2)+L(3)+L(13) \pmod{1008} \\ 33 &\equiv 2L(2)+2L(3)+L(11) \pmod{1008} \\ 36 &\equiv 2L(2)+2L(7) \pmod{1008} \end{aligned}$$

There are six unknowns $L(2), L(3), L(5), L(7), L(11), L(13)$ and six equations, using linear algebra methods, we can arrive at the solution

$$\begin{gathered} L(2) = 646, L(3) = 534, L(5) = 886, \\ L(7) = 380, L(11) = 697, L(13) = 861 \end{gathered}$$

Next up, the idea is to find $17^x \cdot 24$ and find one that can factorize over primes not greater than 13. Indeed, we have $17^2 \cdot 24 \equiv 2 \cdot 3^2 \cdot 7^2 \pmod{1009}$, so then we have

$$\begin{gathered} 2 + L(24) \equiv L(2) + 2L(2) + 2L(7) \pmod{1008} \\ L(24) = 456 = n \end{gathered}$$

Therefore, we indeed arrive at the answer $n = 456$. As seen above, this method relies on the property that prime factorizations always exist, which does not apply to most ellpitic curves. [10, pp. 154-157]

For the example above, we examined exponents of 17 up to 17^{36}, and also computed $17 \cdot 24$ and 17^{24}. This takes significantly less time than enumerating 17^n for all n until we reach 456. Therefore, Index Calculus is much more efficient than brute-forcing.

We've just descibed the basic algorithm for Index Calculus, there is a more sophisticated method called General Number Field Sieve which builds upon index calculus. GNFS is in general more efficient than simple index calculus for large primes [11]. The number of operations expected for the GNFS algorithm can be written as [12]:

$$\exp\left((64/9)^{1/3}(\ln p)^{1/3}(\ln \ln p)^{2/3}\right)$$

where p is the prime that defines the finite field in $\mathbb{F}_p$. For a field with $p = 2^{2048}$, the expected running time for GFNS to solve the Discrete Log would take about $1.5 \cdot 10^{35} \approx 2^{117}$ operations. In later sections, we will take a look at Diffie-Hellman done on elliptic curves and how the number of operations needed to solve the discrete log problem on elliptic curves compares with Diffie-Hellman on finite fields.

Elliptic Curve Cryptography

Let an elliptic curve be denoted by the equation $y^2 = x^3 + Ax + B$ (named the short Weierstrass form) where A and B are constants. The curve is symmetric about the x-axis, since if (x, y) is a point on the curve, $(x, -y)$ is also on the curve.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be distinct points on the curve, where $x_1 \neq x_2$. We can find a new point on the curve by defining a line that goes across the two points, with slope

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

And line equation

$$y = m(x - x_1) + y_1$$

We substitute this into the equation of the curve and try to solve for x:

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B$$

Expanding and rearranging gives:

$$x^3 - m^2x^2 + (2m^2x_1 - 2y_1m + A)x + 2y_1mx_1 - m^2x_1^2 - y_1 = 0$$

With Vieta's formulas, the sum of roots for the cubic is m^2. We already know two roots of this polynomial as P_1 and P_2 are common points on the curve and the line, so we can find the x coordinate of the third point:

$$x_3 = m^2 - x_1 - x_2$$

To find the y-coordinate, we use the original line, equation, while flipping the resulting y-coordinate. This will be important to us later since flipping the y coordinate results in the operation being associative.

$$y_3 = -(m(x_3 - x_1) + y_1) = m(x_1 - x_3) - y_1$$

Therefore, we have arrived at $P_3 = (x_3, y_3)$, a third point distinct from P_1 and P_2.

If only one point $P_1 = (x_1, y_1)$ is known, we can use implicit differentiation to find the tangent line:

$$y^2 = x^3 + Ax + B$$
$$2y\frac{\mathrm{d}y}{\mathrm{d}x} = 3x^2 + A$$
$$m = \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{3x^2 + A}{2y} = \frac{3x_1^2 + A}{2y_1}$$

With the same line equation $y = m(x - x_1) + y_1$, with the same expanded formula:

$$x^3 - m^2x^2 + ... = 0$$

But this time, x_1 is a repeated root, as a tangent line either touches no other points at all (the case when $y = 0$) or touch one other point.

Therefore, we can find the third point with

$$x_3 = m^2 - 2x_1$$

And

$$y_3 = -(m(x_3 - x_1) + y_1) = m(x_1 - x_3) - y_1$$

Therefore, we can begin to define a group law for points on elliptic curves. A "point at infinity" is added to the normal set of points on the curve, so that the group is well-defined for operations on all points. In projective geometry, a point at infinity is derived from the geometric assumption that parallel lines "meet at infinity".

Let $C : y^2 = x^3 + Ax + B$ be the elliptic curve with the set of points that satisfy the given equation. We now show that $C \cup \{\infty\}$ forms a group.

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points that are on the curve. Define $P_3 = P_1 + P_2$ to be as follows:

- If $P_1 = P_2 = (x_1, 0)$, let $P_3 = \infty$.
- If $P_1 = P_2 = (x_1, y_1)$ where $y_1 \neq 0$, let

$$P_3 = (m^2 - 2x_1, m(x_1 - x_3) - y_1), \text{where } m = \frac{3x_1^2 + A}{2y_1}$$

- If $x_1 = x_2$ but $y_1 \neq y_2$ (N.B. the only case where this happens is $y_1 = -y_2$): let $P_3 = \infty$.
- Otherwise, let

$$P_3 = (m^2 - x_1 - x_2, m(x_1 - x_3) - y_1), \text{where } m = \frac{y_2 - y_1}{x_2 - x_1}$$

Additionally, define $P_1 + \infty = \infty + P_1 = P_1$, as well as $\infty + \infty = \infty$.

Proof of Associativity

Perhaps the most surprising result of defining this operation is that the operation is associative, that is, $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ for any three points P_1, P_2, P_3 that belong to the set $C \cup \{\infty\}$. Proving this algebraicly becomes very tedious, but there is a geometric argument using cubic space curves and Bézout's theorem for the case where the points are distinct and none of them have the same x value. A proof is outlined below.

A cubic space curve is given with the following formula [13]:

$$ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j = 0$$

Since an elliptic curve is given by $x^3 + Ax + B - y^2 = 0$, an elliptic curve is also a cubic space curve.

A consequence of Bézout's theorem states that two cubic space curves intersect at 9 points, counting multiplicities such as self-intersections. In our case, we ignore multiplicities so the curves will intersect at 9 distinct points. We now summarize the proof of the follow theorem in [13], used for proving associativity:

> Let C, C_1, C_2 be cubic space curves. Suppose C goes through eight of the nine intersection points between C_1 and C_2. Then C also goes through the nineth intersection point.

Firstly, note that a total of 10 coefficients were used in the formula for a cubic space curve: $a, b, c, d, e, f, g, h, i, j$. If the equation is scaled by a linear factor, it results in the same curve, so we can say that this curve is nine-dimensional (constrained by nine linear factors). Constraining a curve to go through a single point reduces its dimension by one, therefore the set of all curves that go through eight specific points is one-dimensional.

Suppose C_1 is specified by the equation $F_1(x, y) = 0$ and C_2 by $F_2(x, y) = 0$. Then any intersection point (x_1, y_1) will satisfy $F_1(x_1, y_1) = F_2(x_1, y_1) = 0$. Therefore, a linear combination of the two functions will result in a cubic space curve that goes through the eight intersection points: $F_3 = \lambda_1 F_1 + \lambda_2 F_2$. Since this linear combination also represents a one-dimentional family of cubic space curves, it follows that C must also be specified by $F_3(x, y) = 0$ for specific factors λ_1 and λ_2.

Since we know that the nineth intersection point also satisfies $F_3(x, y) = 0$, we know that C goes through the nineth intersection point. $\square$

Now, we use this theorem to prove that the elliptic curve point addition operation is associative. Normally, the point at infinity is not shown in diagrams, but since it follows Bézout's theorem and for ease of presentation, the point will be denoted as O in the diagram.

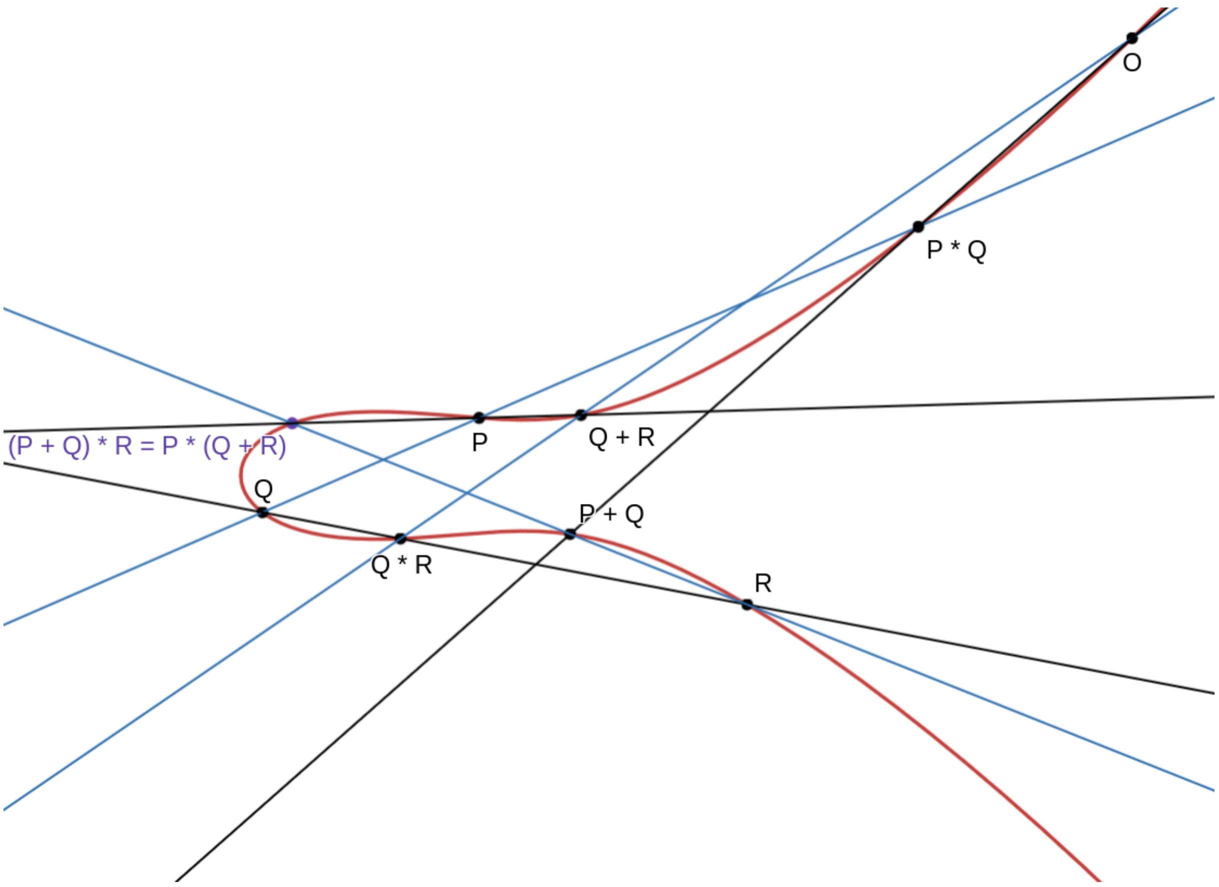

Figure 2: By author, graphical proof of associativity

On the elliptic curve, we start with three arbitrary points, P, Q, and R. The $*$ operation takes two points, draws a line through them, then use the third intersection point on the elliptic curve as a result. Through this, we create $Q * R$ and $P * Q$. To find $Q + R$ and $P + Q$, we take the starred point ($Q * R$ for example), draw a line from it to O, then take the third intersection point. Note that this is equivalent to "flipping" the intersection point used in the definition of elliptic curve addition.

With the same $*$ operation, we create $(P + Q) * R$ and $P * (Q + R)$. Proving that these two points are equal proves associativity.

Next, we draw six lines. The three blue lines go through $(P + Q) * R$, P, Q, R, $Q * R$, $Q + R$, $P * R$, $P + Q$, and O. The three black lines $P * (Q + R)$, P, Q, R, $Q * R$, $Q + R$, $P * R$, $P + Q$, and O. Three lines form a cubic space curve:

$$(y - (m_1x + b_1))(y - (m_2x + b_2))(y - (m_3x + b_3)) = 0$$

Let C_1 be the elliptic curve (itself a cubic space curve), and let C_2 be the cubic space curve formed through the three black lines. As C, the cubic space curve formed through the three blue lines, go through eight of the nine points that are intersections between C_1 and C_2: P, Q, R, $Q * R$, $Q + R$, $P * R$, $P + Q$, and O, the last point of intersection must be same point. The last point on C_2 is denoted $P * (Q + R)$, while the last point on C is denoted $(P + Q) * R$, therefore, we have proven $P * (Q + R) = (P + Q) * R$, which implies $P + (Q + R) = (P + Q) + R$, proving the associativity. $\square$

Group of elliptic curve points

Therefore, $C \cup \{\infty\}$ forms a group since:

1. The operation $+$ is well-defined for any points $P_a + P_b$ where $P_a, P_b \in C \cup \{\infty\}$ as above.
2. ∞ is the identity element, where $P_a + \infty = P_a$ for all $P_a \in C \cup \{\infty\}$.
3. Every element has an inverse: let $P_a = (x, y)$, its inverse is $-P_a = (x, -y)$. We know that $-P_a \in C$ since the curve is given as $y^2 = x^3 + Ax + B$ and swapping y with $-y$ will still hold.
4. The operation is associative.

Since elliptic curve points form a group, cryptographic techniques such as Diffie-Hellman Key Exchange which relies on group operations can also be applied to elliptic curves.

Elliptic Curve Diffie-Hellman

One important difference between elliptic curve operations and modular multiplicative group operations is in notation. In elliptic curves, the operation is commonly represented as addition of two points. Therefore $A + B$ is the normal operation on two points A and B while kA is the operation repeated (e.g. $2A = A + A$). In the multiplicative group modulo p, it corresponds to AB and A^k. Thus, in previous sections, an operation such as A^k will now be written as kA in the context of elliptic curves.

With that note, Diffie-Hellman in elliptic curves follows the exact same procedure: two parties agree on a curve group to use, then decide on a base point G. Alice generates a secret integer a and sends Bob aG. Bob generates a secret integer b and sends Alice bG. They can now both calculate abG, which cannot be known by third parties unless they can solve the discrete log problem in elliptic curves.

Finding the Discrete Log with Pollard's ρ algorithm

Pollard's ρ algorithm is a general algorithm for solving the discrete log problem for any Abelian (commutative) group. It is less efficient than the general number field sieve on discrete log in finite fields, taking $O\left(\sqrt{N}\right)$ time on average in group G where $|G| = N$.

We first take an example adapted from page 164 of Silverman and Tate's book: $y^2 = x^3 + 6692x + 9667$, in F_{10037}, with $P = (3354, 7358)$, $Q = (5403, 5437)$. Find k such that $kP = Q$.

Generate 10 random points on the curve based on multiples of P and Q:

$$\begin{gathered}
M_0 = 42P + 37Q \qquad M_1 = 21P + 12Q \qquad M_2 = 25P + 20Q \\
M_3 = 39P + 15Q \qquad M_4 = 23P + 29Q \qquad M_5 = 45P + 25Q \\
M_6 = 14P + 37Q \qquad M_7 = 30P + 12Q \qquad M_8 = 45P + 49Q \qquad M_9 = 40P + 45Q
\end{gathered}$$

Then pick, in the same way, a random initial point:

$$A_0 = 15P + 36Q = (7895, 3157)$$

Then, choose an M_i point to add to based on the ones digit of the x coordinate of the point. As A_0 has $x = 7895$, $A_1 = A_0 + M_5 = (7895, 3157) + (5361, 3335) = (6201, 273)$.

Formally, define

$$A_{n+1} = A_n + M_i \text{ where } i \equiv x_n \pmod{10}$$

for $A_n = (x_n, y_n)$. The choice of random M_i points creates a kind of “random walk” of the points in the elliptic curve. As we keep calculating, we get:

$$\begin{aligned} A_0 &= (7895, 3157), A_1 &&= (6201, 273), ..., \\ A_{95} &= (170, 7172), A_{96} &&= (7004, 514), ..., \\ A_{100} &= (170, 7172), A_{101} &&= (7004, 514) \end{aligned}$$

We reach a cycle with $A_{95} = A_{100}$. Since we know the multiples of P and Q for all of the M_i points and thus all A_n points, keeping track of them gives us $A_{95} = 3126P + 2682Q$, we also have $A_{100} = 3298P + 2817Q$. With $3126P + 2682Q = 3298P + 2817Q$, we have:

$$\infty = 172P + 135Q = (172 + 135n)P$$
$$172 + 135n \equiv 0 \pmod{10151}$$
$$n \equiv 1277 \pmod{10151}$$

With verification, we indeed have $1277P = Q$.

Evaluation

Pollard’s ρ algorithm on elliptic curve groups works on average with $\sqrt{\frac{\pi}{4}N}$ elliptic curve additions with N being the order for the base point P [14]. On the other hand, the general number field sieve takes about $\exp\left((64/9)^{1/3}(\ln p)^{1/3}(\ln \ln p)^{2/3}\right)$ in a prime field with order p. Assigning real numbers to these expressions, we can evaluate the current industry standards for cryptography.

Diffie-Hellman in TLS 1.3

The Transport Layer Security (TLS) protocol is the protocol used in virtually all internet connections that are protected through cryptography [15]. One important part of this protocol is Diffie-Hellman Key Exchange. As of writing, the latest version of TLS is 1.3. We shall now examine the Diffie-Hellman methods it supports.

Finite Field Diffie-Hellman

The smallest finite field used by TLS for Diffie-Hellman is named ffdhe2048 [16], with the prime modulus defined as

$$p = 2^{2048} - 2^{1984} + (\lfloor 2^{1918} \cdot e \rfloor + 560316) \cdot 2^{64} - 1$$

With the group size being $(p-1)/2$. If we computed the number of operations needed for the general number field sieve to run, we get approximately 2^{117} operations or equivalently, this provides 117 bits of security. The original definition of ffdhe2048 takes a more conservative estimate, and claims that this provides 103 bits of security [17].

As this field uses a prime 2048 bits of size, each group element requires 2048 bits of storage.

Elliptic Curve Diffie-Hellman

The smallest elliptic curve supported by TLS appears to be curve25519, using the prime $p = 2^{255} - 19$ as the field $\mathbb{F}_p$ the elliptic curve is over, and the curve $y^2 = x^3 + 486662x^2 + x$. The order of the group is $2^{252} + 27742317777372353535851937790883648493$. As the fastest method to break the discrete logarithm takes $\sqrt{\frac{\pi}{4}N}$ operations, this specific curve requires approximately 2^{126} operations to break, or providing 126 bits of security.

As elliptic curve points have coordinates under the prime field $2^{255} - 19$, each coordinate value requires 255 bits of storage, therefore an entire point (both x and y coordinates) would take about 510 bits of storage.

Performance of group operations

Assume that multiplying two 256-bit integers has cost $\boldsymbol{C}$. Multiplication of two 2048-bit integers thus will cost $64\boldsymbol{C}$ as each 2048-bit integer has 8 256-bit digits and each digit from the first operand needs to multiply with the next operand.

The story in elliptic curves is much more complicated. Curve25519 follows the form $By^2 = x^3 + Ax^2 + x$ called a Montgomery curve. All curves of that form can be transformed into the short Weierstrass form we used in this paper but not the other way around. Detailed in [18], the Diffie-Hellman Key Exchange protocol could be designed so that only the x-coordinate of each point in the process is needed, which simplifies the process by removing the need to compute y coordinates. Under the arithmetic of only the x coordinates of curve points, adding two curve points costs $3M + 2S + 3a + 3s$, where M, S, a, s are costs for multiplying two numbers, squaring a number, adding two numbers, subtracting two numbers in the field the curve is defined on respectively. Assuming that the cost for addition and subtraction is negligible compared to multiplication, and assuming that squaring has approximately the same cost as multiplying two numbers, the cost for adding two curve points is approximately $5M$. Note that the field is defined over $2^{255} - 19$, so the cost of a multiplication M (for two 255-bit integers) can be considered as less than the cost of multiplying two 256-bit integers. So we have $M < \boldsymbol{C}$.

Note how adding two curve points only costs $5M$, while multiplying in finite fields costs $64\boldsymbol{C}$. (approximately 13x difference) As performing the group operation is the primary backbone behind Diffie-Hellman key exchange, this performance difference can have huge implications.

Comparison

The specific methods we have chosen to evaluate provide a general insight into the efficiencies of different methods of diffie-hellman key exchange.

Elliptic curves only require about 512 bits of storage for a full point, about 256 bits if only storing the x-coordinate, while in finite fields, each element requires 2048 bits of storage, taking 8x as much storage than elliptic curves.

Adding two curve points compared to multiplying two finite field elements provide similar benefits in performance as well, with an approximate 13x speedup.

Both of these advantages can be seen from the fact that the Discrete Log Problem is much harder on elliptic curves than in finite fields in general, which we have shown above through comparing the General Number Field Sieve and Pollard's ρ algotithm. As a consequence the latter requires much more space and time for their computations in order to provide the same level of security in the trade-off between efficiency and security.

Conclusion

Elliptic curves offer a much better alternative to existing cryptographic methods and is representative of the progress mathematicians have made towards helping build a large system (i.e. the Internet) that scales. To answer the question of "To what extent can elliptic curves be used to establish a shared secret over an insecure channel", the answer is "Yes, and it is fast and efficient!"

Bibliography

[1] D. Warburton and S. Vinberg, "The 2021 TLS Telemetry Report." Accessed: Aug. 13, 2024. [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report

[2] K. Sako, "Public Key Cryptography," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 487–488, 2005. doi: 10.1007/0-387-23483-7_331.

[3] H. Krawczyk, "Perfect Forward Secrecy," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 457–458, 2005. doi: 10.1007/0-387-23483-7_298.

[4] M. Just, "Diffie–Hellman Key Agreement," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, p. 154–155, 2005. doi: 10.1007/0-387-23483-7_111.

[5] B. Kaliski, "Field," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, p. 458–459, 2011. doi: 10.1007/978-1-4419-5906-5_407.

[6] B. Kaliski, "Ring," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 1049–1050, 2011. doi: 10.1007/978-1-4419-5906-5_431.

[7] B. Kaliski, "Finite Field," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, p. 468–469, 2011. doi: 10.1007/978-1-4419-5906-5_409.

[8] M. Friedl, N. Provos, and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol," Mar. 2006, doi: 10.17487/RFC4419.

[9] L. Velvindron and M. D. Baushke, "Increase the Secure Shell Minimum Recommended Diffie-Hellman Modulus Size to 2048 Bits," Dec. 2017. doi: 10.17487/RFC8270.

[10] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, 2nd ed. in Discrete mathematics and its applications. Boca Raton, FL: Chapman & Hall/CRC, 2008.

[11] K. Nguyen, "Index Calculus," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 287–289, 2005. doi: 10.1007/0-387-23483-7_198.

[12] A. K. Lenstra, "L-Notation," *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, p. 358–359, 2005. doi: 10.1007/0-387-23483-7_237.

[13] J. H. Silverman and J. T. Tate, *Rational Points on Elliptic Curves*. in Undergraduate Texts in Mathematics. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-18588-0.

[14] D. J. Bernstein, T. Lange, and P. Schwabe, “On the Correct Use of the Negation Map in the Pollard rho Method,” in *Public Key Cryptography – PKC 2011*, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., Berlin, Heidelberg: Springer, 2011, pp. 128–146. doi: 10.1007/978-3-642-19379-8_8.

[15] C. Heinrich, “Transport Layer Security (TLS),” *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 1316–1317, 2011. doi: 10.1007/978-1-4419-5906-5_234.

[16] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” Aug. 2018. doi: 10.17487/RFC8446.

[17] D. K. Gillmor, “Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS),” Aug. 2016. doi: 10.17487/RFC7919.

[18] C. Costello and B. Smith, “Montgomery curves and their arithmetic,” *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 227–240, Sep. 2018, doi: 10.1007/s13389-017-0157-6.