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: 3958 (Unofficial count!)

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 the information transmitted.

Diffie-Hellman is an important element in 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].

Much of cryptography is developed on the need to communicate messages securely in that other people cannot know what messages are being sent. Password-based, or *symmetric* cryptography uses a secret key known between two parties to communicate messages safely [2].

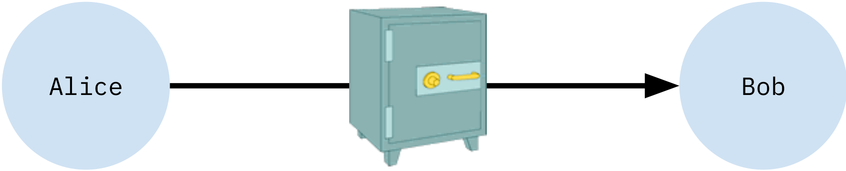

Figure 1: Diagram by author, derived from [3].

Under an analogy using safes, if Alice and Bob both know a secret combination, they can send each other secret messages within safes configured with that combination which protects their messages from being inspected by anyone else. Symmetric cryptography is then concerned with the mathematical processes that can be used to construct such a safe.

Diffie-Hellman arises from a need for Alice and Bob to quickly agree on a combination to use even if their own form of communication is one that can be eavesdropped by third parties like Eve [4]. This is at the core of applications such as securing Internet connections, since visitors of a website should not be forced to physically visit a company in order to establish some secret key for communication, but communication through the Internet without encryption provided by cryptography would be insecure.

Diffie-Hellman Key Exchange is designed specifically so that people can establish a shared secret (the combination 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 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?) to show the effectiveness of elliptic curves.

Group Theory

As much of the math required to understand elliptic curves and how they help in establishing shared secrets builds upon group theory, we discuss some notation and concepts used in the cryptographic methods we will be using.

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

Fermat's Little Theorem suggests the following to be true for any non-zero 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 [5, pp. 73-76].

A group is, at its core, a set equipped with an operation. Therefore, the $\in$ symbol and the word *element* applies to groups as well. We will refer to the specific group we discussed above as $\mathbb{Z}_p^\times$. In a similar vein, $\mathbb{Z}_p^+$ specifies addition modulo p 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 $|\mathbb{Z}_p^\times| = p - 1$. The additive group includes zero as the identity, 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 $a = 1008$ is the smallest a such that $17^a \equiv 1 \pmod{1009}$, whereas the order of 2 in the same group is 504, since $b = 504$ is the smallest b such that $2^b \equiv 1 \pmod{1009}$. Therefore, we have $|17| = 1008$ and $|2| = 504$.

The order of groups and specific elements are used in analyzing and comparing the difficulty of any third-party breaking the cryptography and obtaining a shared secret secured with group operations.

The Discrete Log Problem

Under $\mathbb{Z}_{1009}^\times$, we are asked to find the smallest integer n for which $17^n \equiv 24$. In this case,

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

And 456 is the first exponent for which the equivalence holds. Therefore $n = 456$ is the solution to this question. More generally, the discrete log problem (DLP) asks for a smallest exponent n in a group g and $a, b \in g$ such that

$$a^n = b$$

The brute-force approach to this problem would be repeatedly performing the group multiplication, calculating a^2, a^3, a^4 and checking if any of them matches b. In the example problem, it would take 455 multiplications before finally arriving at the answer. If we tried to solve questions of this kind repeatedly with the exponent n taken at random, brute-forcing would take on average $\frac{1}{2}|a|$ operations since answers are in the range of 0 to $|a| - 1$. As the order $|a|$ gets big (towards numbers as big as 2^{200}), this approach quickly becomes infeasible.

Cryptographic techniques are, then, built upon the assumption that the Discrete Log Problem is non-trivial to solve.

Finite Field Cryptography and Attacks

A field is a set that forms a group under addition, forms a group under multiplication with the zero removed, and where multiplication is distributive: $a(b + c) = ab + ac$ [6], [7].

As it turns out, all fields with the same finite order q can be mapped to each other while preserving their structure (i.e. *finite fields* with equal order are all *isomorphic*). Therefore any finite field with prime order p is isomorphic to the field of integers modulo p. We write it as $\mathbb{F}_p$, and use $\mathbb{F}_p^\times$ to explicitly refer to the multiplicative subgroup (equivalent to $\mathbb{Z}_p^\times$ used above) [8].

Diffie-Hellman Key Exchange

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

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

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

$$17^b \equiv 24$$
$$a = 123$$
$$17^{ab} \equiv \left(17^b\right)^a \equiv 24^{123} \equiv 578$$

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}. However, given just 17^a and 17^b, there is no trivial way to find the answer.

Finding x^{ab} when just given x, x^a, and x^b is named the Diffie-Hellman problem. 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.

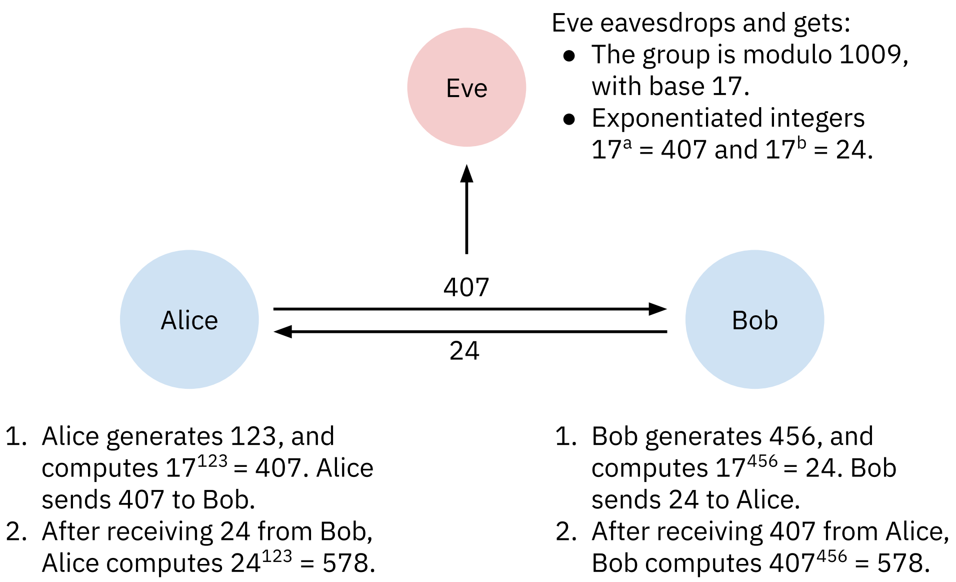

Figure 2: Diagram by author, a description of the Diffie-Hellman Key Exchange process.

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 was 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 [9], [10]. The more general technique of performing Diffie-Hellman on the multiplicative subgroup of a finite field is called Finite Field Diffie-Hellman. We first consider one way to break this, and afterwards show that elliptic curves would take more effort to break.

Index Calculus

Diffie-Hellman Key Exchange on Finite Fields normally uses groups $\mathbb{F}_p$ where $2^{2048} < p < 2^{8192}$ [9], [10]. 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)

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

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. (Note that $x = 0$ works here since 24 can be easily factorized, but in most cases it won't, so we show the more frequent case) Indeed, we have $17^2 \cdot 24 \equiv 2 \cdot 3^2 \cdot 7^2 \pmod{1009}$, so then we have

$$L(17^2 \cdot 24) \equiv L(2 \cdot 3^2 \cdot 7^2) \pmod{1008}$$
$$2 + L(24) \equiv L(2) + 2L(3) + 2L(7) \pmod{1008}$$
$$L(24) = 456$$

We arrive at the answer $n = 456$.

Index Calculus follows the following procedure for finding n in $a^n = b$ [11, pp. 144-146]:

- Find prime factorizations of a^i for various i, limiting the largest prime in the factor base (in our example, we only factored into primes up to 13).
- Solve $L(p)$ for these small primes with the system of equations.
- Find $L(b)$ through varying j in $a^j b$ until it can be factored with our factor base

This method relies on prime factorizations always existing for integers, which in general does not apply to ellpitic curve points [11, pp. 154-157].

For the example above, we examined exponents of 17 up to 17^{36}, and also computed $17 \cdot 24$ and $17^2 \cdot 24$. This took significantly less time than enumerating 17^n for all n until we reach 456. Therefore, Index Calculus is much more efficient than brute-forcing.

An improved method called General Number Field Sieve, based on the idea of reducing discrete logarithm problems to systems of linear equations in Index Calculus, is more efficient than simple index calculus for large primes [12]. The number of operations expected for the GNFS algorithm can be written as [13]:

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

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

The equation $y^2 = x^3 + Ax + B$ (named the short Weierstrass form) where A and B are constants define an elliptic curve. 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.

The idea of a group operation for two points on the curve comes from the idea of “finding a third point” on the curve, usually through drawing a line and finding another intersection with the curve. We now summarize the steps for defining a group law as shown in [11, pp. 12-15].

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 and line equation

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$
$$y = m(x - x_1) + y_1$$

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, but flip the resulting y-coordinate. This is important as otherwise the operation does not define a group [14, p. 12].

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

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. For special cases, such as adding two points on a vertical line, 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. This is studied more rigorously in projective geometry, though we incorporate this concept for simply defining the group law on elliptic curves [11, p. 11].

For an elliptic curve $C : y^2 = x^3 + Ax + B$ we define the following set:

$$E(C) = \{\mathbf{0}\} \cup \{(x, y) \mid y^2 = x^3 + Ax + B\}$$

Where $\mathbf{0}$ is the "point at infinity". We now show that $E(C)$ forms a group.

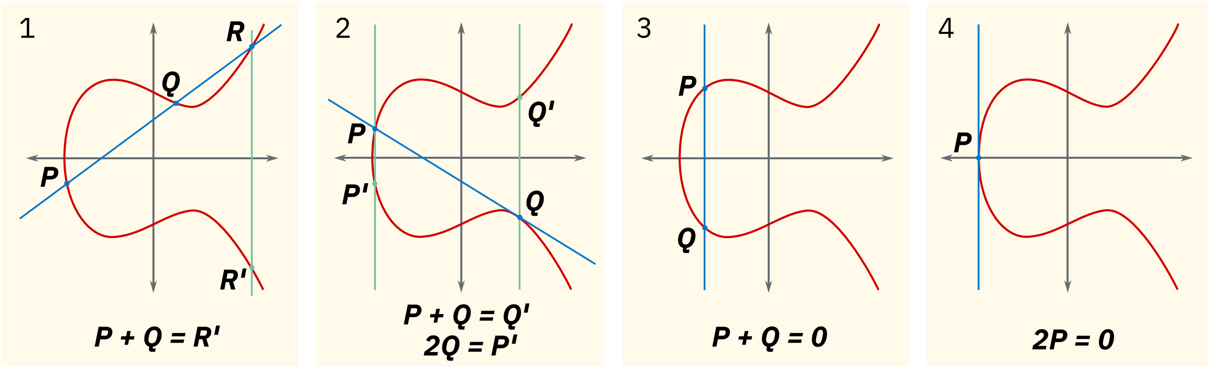

Figure 3: Diagram from [15], an illustration of the group operation defined on elliptic curves

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 $P_3 = \mathbf{0}$.
- 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 $y_1 \neq y_2$ (if and only if $y_1 = -y_2$): let $P_3 = \mathbf{0}$.
- 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 $P_1 + \mathbf{0} = \mathbf{0} + P_1 = P_1$, as well as $\mathbf{0} + \mathbf{0} = \mathbf{0}$.

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 $P_1, P_2, P_3 \in E(C)$. Proving this algebraicly becomes tedious, but there is a geometric argument using cubic space curves and Bézout's theorem for a specific case where the points have distinct x coordinates. A proof from [14, pp. 8-15] is outlined below.

A cubic space curve is given with the following formula:

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

An elliptic curve $x^3 + Ax + B - y^2 = 0$ is a cubic space curve. The union of three lines

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

is also a cubic space curve.

A consequence of Bézout's theorem is that two cubic space curves intersect at 9 points. The nine points could include the point at infinity in projective geometry, counts multiplicities as more than one point of intersection such as when at a tangent, and allows complex numbers as coordinates. For simplicity, the proof ignores these technicalities.

We first prove the following:

> 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.

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 cubic 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 (a, b) will satisfy $F_1(a, b) = F_2(a, b) = 0$. 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 be able to take the form $F_3(x, y) = 0$ 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. $\square$

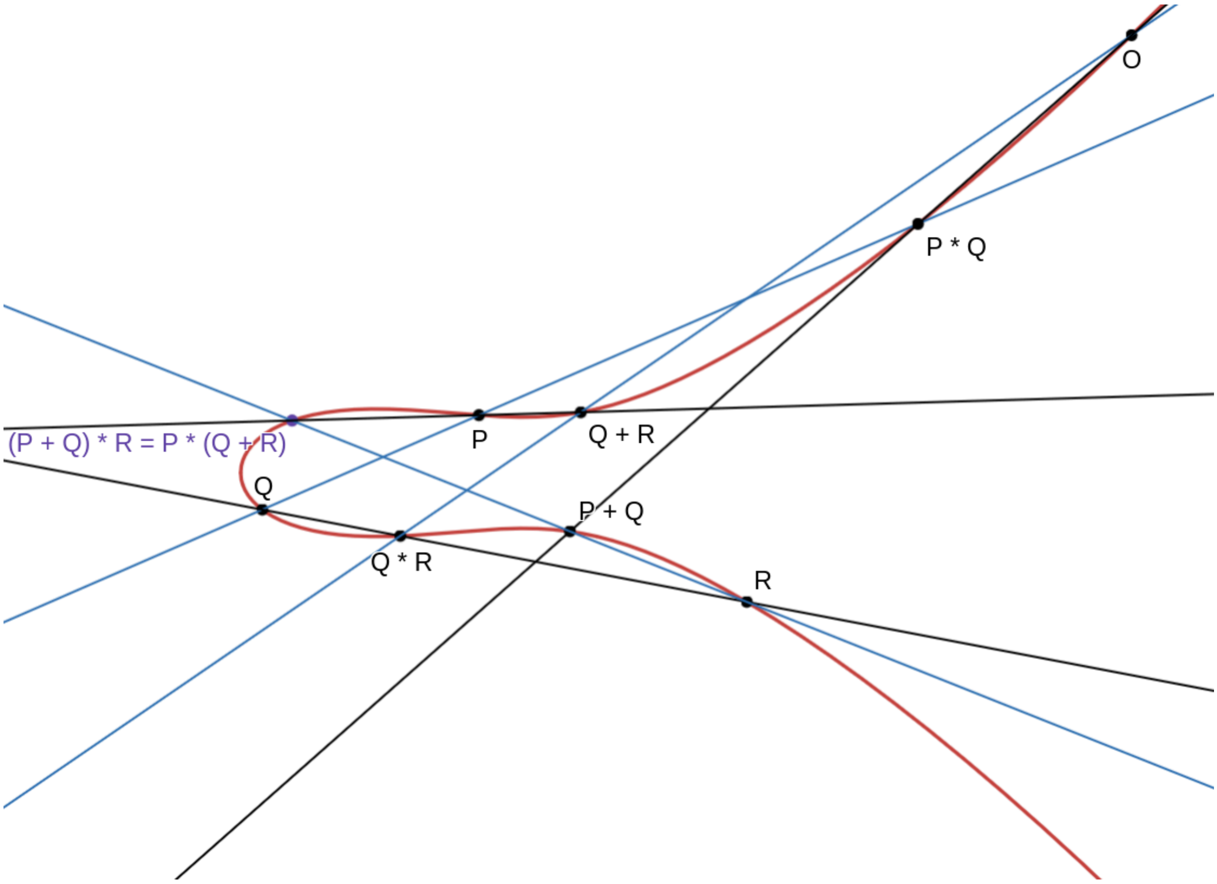

Figure 4: By author, graphical proof of associativity

Now, we use the theorem to prove that the elliptic curve point addition operation is associative for the specific case where points are at distinct x coordinates. Normally, the point at infinity is not shown in diagrams, but since it follows Bézout's theorem and for the graphical proof, the point will be denoted as O in the diagram.

As shown in Figure 4, we start with three arbitrary points on the elliptic curve, P, Q, and R, and our goal is to show that $(P+Q)+R = P+(Q+R)$. 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 (for example $Q * R$), draw a line from it to O, then take the third intersection point, equivalent to "flipping" the intersection point used in the definition of elliptic curve addition.

We then create $(P+Q) * R$ and $P * (Q+R)$, and 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.

Let C_1 be the elliptic curve, and let C_2 be the cubic space curve formed through the three black lines, C is the curve formed through three blue lines. As C goes 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, it must also go through $P * (Q+R)$, the last intersection point between the black lines and the curve. However, we also know that point as $(P+Q) * R$ from the blue lines' intersection, therefore they must be the same point. $P * (Q+R) = (P+Q) * R$ implies $P+(Q+R) = (P+Q)+R$, proving the associativity in this case.

This is deliberately an incomplete proof, since we did not cover corner cases such as when two of the three points have the same x-coordinate, a more complete proof can be seen in [11, pp. 20-32], we will assume associativity is true in the general sense here.

Group of elliptic curve points

Therefore, $E(C)$ forms a group:

1. The operation $P_a + P_b$ is well-defined for any points $P_a, P_b \in E(C)$.
2. $\mathbf{0}$ is the identity, where $P_a + \mathbf{0} = P_a$ for all $P_a \in E(C)$.
3. Existence of inverse: let $P_a = (x, y)$, its inverse is $-P_a = (x, -y)$ since $(x, y) + (x, -y) = \mathbf{0}$.
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

Elliptic curve operations and modular multiplicative group operations differ 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, an operation in previous sections 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 P. Alice generates a secret integer a and sends Bob aP. Bob generates a secret integer b and sends Alice bP. They can now both calculate abP, which cannot be known by third parties unless they can solve the discrete log problem in elliptic curves.

A short example is as follows: Alice and Bob agrees to use the curve $y^2 = x^3 + 6692x + 9667$ in $\mathbb{F}_{10037}$, with the base point $P = (3354, 7358)$ (from [14, p. 164]). Alice generates $a = 1277$ and sends $Q = aP = (5403, 5437)$ to Bob. Bob generates $b = 1337$ and sends $R = bP = (7751, 1049)$ to Alice.
Alice calculates $aR = abP = (8156, 1546)$, and bob calculates $bQ = baP = (8156, 1546)$ as their shared secret.

To figure out this shared secret, Eve could try to break the discrete log for $Q = aP$.

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 $O(\sqrt{N})$ time on average in a group G where $|G| = N$ [11, pp. 147-150].

Pollard's ρ can be used to solve the problem above: for the curve $y^2 = x^3 + 6692x + 9667$ in $\mathbb{F}_{10037}$, 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:

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

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 $A_{95} = 3126P + 2682Q$ and $A_{100} = 3298P + 2817Q$. With $3126P + 2682Q = 3298P + 2817Q$, and knowing that $|P| = 10151$, we have:

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

With verification, we indeed have $1277P = Q$, and we can then calculate $1277R = (8156, 1546)$ in order to find the shared secret on the example above.

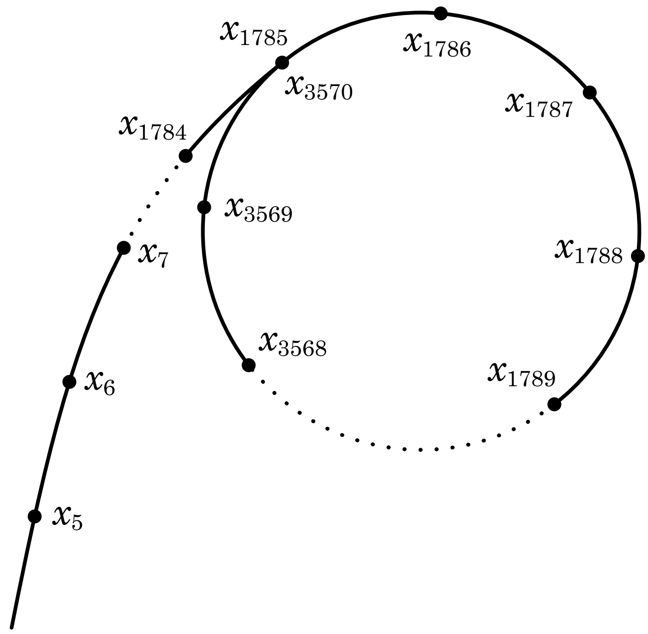

Figure 5: Diagram from [16], a visual explanation for the name (ρ) of the algorithm

Therefore, Pollard's ρ operates with the following steps:

- For finding k where $kP = Q$, first generate a random set of elements consisting of $M_i = x_i P + y_i Q$, then generate the initial point $A_0 = x_a P + y_a Q$
- With a specific criteria, do a random walk based on some property of A_n to make A_{n+1} (In our case, we used the ones digit)
- When a cycle is reached, we can tabulate all the coefficients in making the random walk to have a form like $aP + bQ = (a + c)P + (b + d)Q$, then

$$cP + dQ = \mathbf{0}$$
$$(c + kd)P = \mathbf{0}$$

And then k can be solved through c and d based on $|P|$.

Evaluation

Pollard's ρ algorithm on elliptic curve groups works on average with $\sqrt{\frac{\pi}{4}N}$ elliptic curve additions with $N = |P|$ [17]. On the other hand, the general number field sieve takes about $\exp\left((64/9)^{1/3}(\ln p)^{1/3}(\ln \ln p)^{2/3}\right)$ operations 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 secured through cryptography [18], with the latest version being TLS 1.3. One important part of this protocol is Diffie-Hellman Key Exchange. 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 [19], with the prime modulus defined as

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

The base element is chosen as 2, with $|2| = (p-1)/2$. 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 it provides 103 bits of security [20].

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

Elliptic Curve Diffie-Hellman
The elliptic curve with the smallest element size 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 base point is $x = 9$, and the order of that point is $2^{252} + 27742317777372353535851937790883648493$. As per [21] the fastest known method to break the discrete logarithm (an improved Pollard's ρ algorithm) takes $\sqrt{\frac{\pi}{4}N}$ group operations, this specific curve requires approximately 2^{126} operations to break, or providing approximately 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 of the eight digits from the first number needs to multiply with the eight from the second number [22, p. 398].

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 essay but not the other way around. Diffie-Hellman for curves in that form 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 [23].

Under Montgomery arithmetic where only the x coordinates of curve points are involved, 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 equal cost or less 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 $M < \boldsymbol{C}$.

Adding two curve points with curve25519 only costs $5M$, while multiplying in ffdhe2048 costs $64C$. (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.

Curve25519 only requires about 512 bits of storage for a full point, about 256 bits if only storing the x-coordinate, while in ffdhe2048, each element requires 2048 bits of storage, taking 8x as much storage than curve25519.

Adding two curve points in those groups compared to multiplying two finite field elements provide similar benefits in performance as well, with an approximate 13x difference in the number of operations required.

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, larger Finite Fields are required to provide the same level of security, which makes elliptic curves more efficient in comparison.

Conclusion

Elliptic Curve Cryptography offers a much better alternative to other existing cryptographic methods for establishing secrets through an insecure channel. This is partly because of the difficulty of the Discrete Log Problem for elliptic curves compared to other groups, which allows it to provide the same level of security while being more efficient. Compared to Finite Field Diffie-Hellman, using Elliptic Curves takes us as 8 times less memory, and performs about 13 times faster.

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] PDClipart.org, "Safe Clip Art." Accessed: Oct. 11, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:Safe_clip_art.png

[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] T. Shemanske, *Modern Cryptography and Elliptic Curves*, vol. 83. in The Student Mathematical Library, vol. 83. Providence, Rhode Island: American Mathematical Society, 2017. doi: 10.1090/stml/083.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] SuperManu, “ECClines-2.” Accessed: Sep. 15, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:ECClines-2.svg

[16] □□□, “Pollard rho cycle.” Accessed: Sep. 16, 2024. [Online]. Available: https://commons.wikimedia.org/wiki/File:Pollard_rho_cycle.svg

[17] 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.

[18] 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.

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

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

[21] D. Hankerson and A. Menezes, “Elliptic Curve Discrete Logarithm Problem,” *Encyclopedia of Cryptography and Security*. Springer US, Boston, MA, pp. 397–400, 2011. doi: 10.1007/978-1-4419-5906-5_246.

[22] P. Pudlák, “The Complexity of Computations,” *Logical Foundations of Mathematics and Computational Complexity: A Gentle Introduction*. Springer International Publishing, Heidelberg, pp. 365–493, 2013. doi: 10.1007/978-3-319-00119-7_5.

[23] 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.