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

Words: 3306

Contents

Introduction

Our society is built on cryptography. Cryptography is built on math. This essay will discuss the math behind the cryptography that is behind our society.

Well, more specifically, *one* specific method in cryptography, which provides *one* functionality that is in such a broad use today. 99.3% of the top 1 million websites prefer this method over others to encrypt their users' internet connections with them.[1]

To take a look at why we want cryptography, let's situate ourselves in a time and place where Bob wants to send something to Alice, with an extra adversary named Eve. Eve hates Bob and Alice, and will try to get information from the communication in any way she can. One conventional conventional cryptographic technique called *public key cryptography* can help Bob in sending messages securely.

Under public key cryptography, Alice and Bob both have a pair of keys, one key that is known by the public called their public key, and one key that is kept as a secret called the private key. Under this system, Bob can use cryptographic methods based on Alice's public key to encrypt a message that only Alice can decrypt using her private key.

There are some disadvantages to this method. One is that encryption using public keys is often slower than *symmetric cryptography*, where a password only known to both sides is used to encrypt and decrypt messages instead. A simple solution for this scenario would be Bob could create a password, use Alice's public key to send this password securely to her, and then use the password for future communications.[2]

But that can also be insecure. If Eve keeps a record of all encrypted messages sent between Bob and Alice, and she obtains Alice's private key, she will be able to decrypt the passwords, and therefore the messages. On cryptographic terms, we say that this method does not have *forward secrecy*.

But then there came Diffie and Hellman. With the cryptographic technique called Diffie-Hellman Key Exchange in their papers, Alice and Bob can quickly establish a password, while Eve is unable to obtain the password just from inspecting their communication. This is great for Alice and Bob, as they can generate a password each time they communicate. 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). Note that additional cryptographic techniques are used to prevent *tampering*, in a situation where Eve can modify any messages sent between Alice and Bob. Tampering, and methods that are resistant to it, are out of scope for this paper.

Diffie-Hellman takes on different forms. There is Finite Field Diffie-Hellman and Elliptic Curve Diffie-Hellman. 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 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 has a multiplicative inverse 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. These properties, existence of an identity element and inverses, associativity and 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.[1] 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 *order* of a specific element x, refers to the smallest integer k such that $x^k = 1$, where 1 is the identity element.[2] For example, the order of 17 in $\mathbb{Z}_{1009}^\times$ is 1008, because 1008 is the smallest integer such that $17^{1008} = 1$, whereas the order of 2 in the same group is 504, since $2^{504} = 1$. Therefore, we have $|17| = 1008$ and $|2| = 504$.

The Discrete Log Problem

Under a specific group $\mathbb{Z}_{1009}^\times$, we ask for 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 or complete search 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

[1] In a similar vein, $\mathbb{Z}_p^+$ refers to the same set of numbers, but specifies addition as its group operation.

[2] On a first glance, the definitions of order for a group and its elements seem to be unrelated. While in fact, the order of an element is also the order of a *subgroup* generated by that element. The meanings of "subgroup" and "generated" are outside the scope of this essay.

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. We shall discuss those methods in a later section, though cryptography is done on well-chosen groups such that even those more advanced attacks become infeasible.

Finite Field Cryptography and Attacks

The term *finite field* refers to the fact that the set of numbers from 1 to $p-1$, alongside with zero, form another group under addition. Moreover, multiplication is distributive over addition: $a(b+c) = ab + ac$ mod p. A field is a set of elements that forms a group under addition and its non-zero elements forms a group under multiplication, where multiplication distributes over addition. The field of integers modulo p is written as $\mathbb{F}_p$. Note that $|\mathbb{F}_p| = p$ due to the inclusion of zero. We use $\mathbb{F}_p^\times$ to explicitly refer to the multiplicative subgroup where $|\mathbb{F}_p^\times| = p - 1$.

Diffie-Hellman Key Exchange

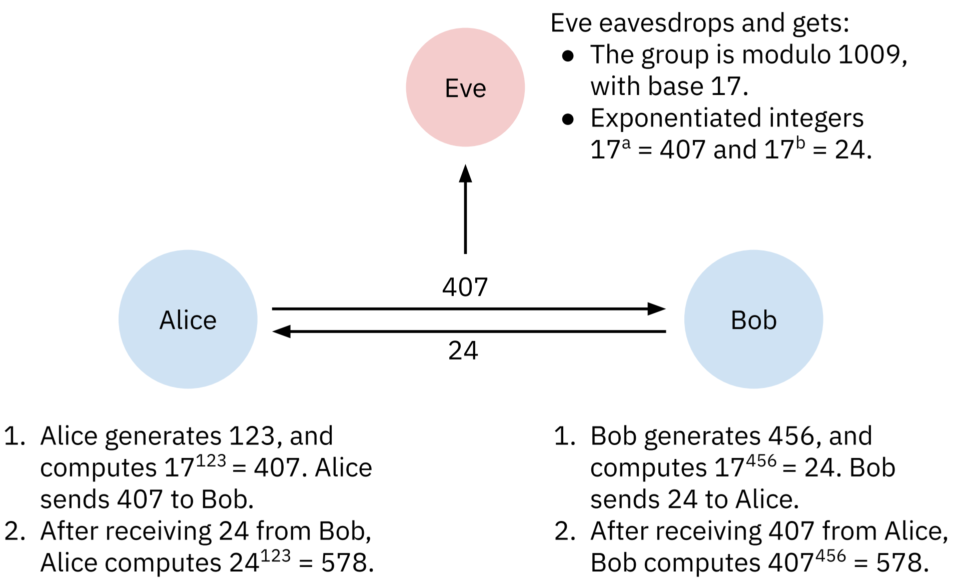
Building off the previous example, suppose we're given the numbers 407 and 24, which are both exponents of a known base 17 in the group modulo 1009. Let's let

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

Is it possible for us to find 17^{ab}? If we know the value of $a = 123$, we can raise 24 to 123, since $\left(17^b\right)^a = 17^{ab}$, which helps us obtain 578, the secret answer. 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. Assuming this 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 $17^b = 24$.

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: A diagram describing 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.

Index Calculus

Diffie-Hellman Key Exchange on Finite Fields normally uses groups $\mathbb{F}_p$ where $2^{2048} \le p \le 2^{8192}$ [5], [6]. The 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

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

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

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

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 the obtain $L(p)$ for small primes p, then figuring out $L(24)$ afterwards. We first try to factorize exponents of the base, 17, looking for ones that can be factorized 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 (notice the change in modulus from 1009 to 1008)

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

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 may not apply to ellpitic curves.

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 [7]. The number of operations expected for the GNFS algorithm can be written as [8]:

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

where p is the prime that defines the finite field in $\mathbb{F}_p$. For a field with $p = 2^{2048}$, it will 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 $y^2 = x^3 + Ax + B$ where A and B are constants. Note that 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:

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

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:

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

In the group law, the y coordinate of the resulting point is flipped: (TODO: explain why)

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

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.

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 $P_1 = P_2 = (x_1, y_1)$, 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 $\infty + \infty = \infty$.

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 cubics and Bézout's theorem for the case where the points are distinct and none of them have the same x value.

Proof of Associativity

Bézout's theorem states that in general two plane curves given in the equations $a(x, y) = 0$ and $b(x, y) = 0$ with degrees d_a and d_b will have $d_a d_b$ intersections.

Formally, the group of elliptic curves can be defined in projective space where the point at infinity can be treated like any other point. For ease of presentation, the point at infinity will be denote as O.

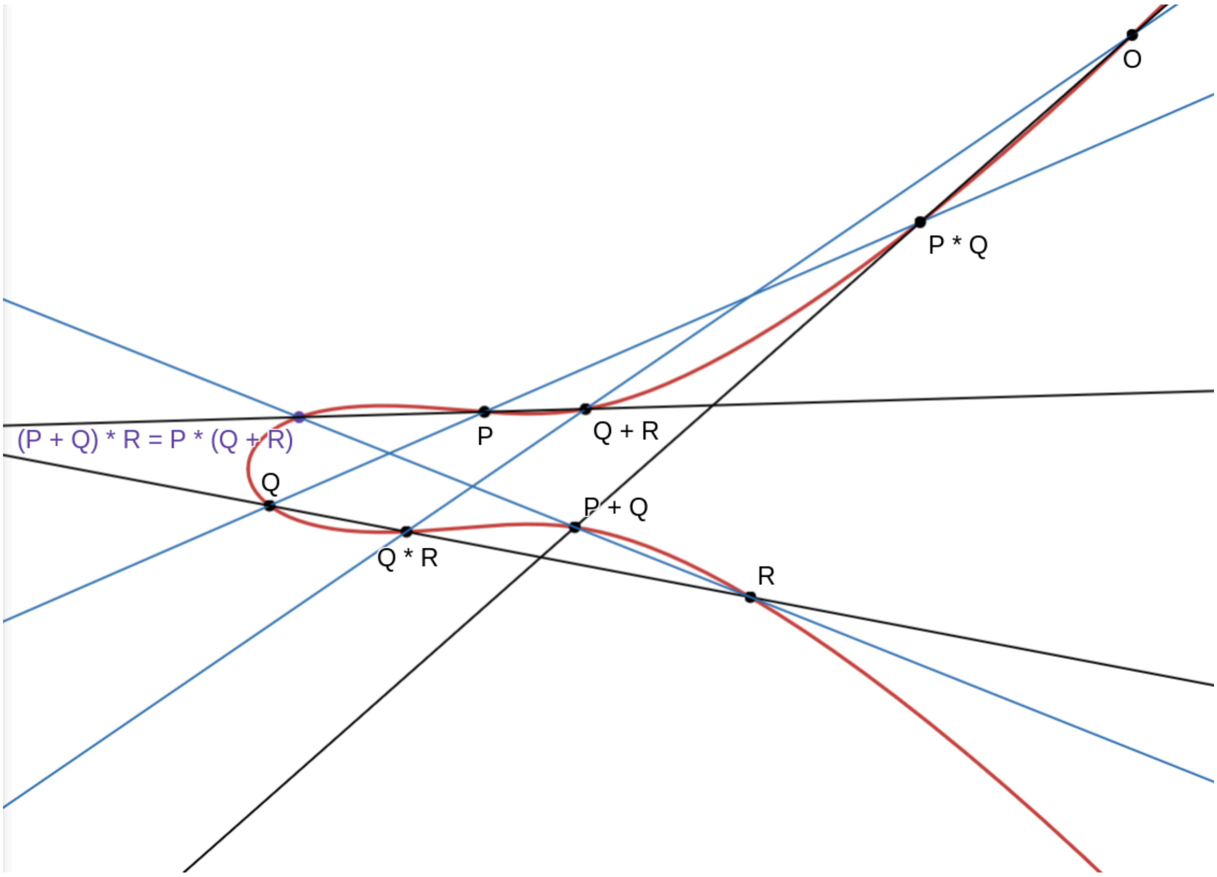


Figure 2: Graphical proof of associativity

Note that the intersections between the blue lines and the curve are $(P+Q)*R$, P, Q, R, $Q*R$, $Q+R$, $P*R$, $P+Q$, and O.

The intersections between the black lines and the elliptic curve are $P*(Q+R)$, P, Q, R, $Q*R$, $Q+R$, $P*R$, $P+Q$, and O.

As three lines form a cubic, and the two groups of three lines both intersect the same eight out of the nine points with the elliptic curve, it can be shown that the ninth point is the same for both cubics that intersect with the elliptic curve, thus proving that $(P+Q)*R = P*(Q+R)$.

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 curve, 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, the correspondence goes to AB and A^k. Thus, in previous sections about the multiplicative groups, 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 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 with N being the order of the group.

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 $kP = Q$.[3]

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:

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

[3] Originally a Montgomery equation, used substitution to turn it into the short Weierstrass form for consistency.

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 [9]. 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.

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 [10]. 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 [11], with the prime modulus defined as

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

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 that the field provides 117 bits of security. The original definition indeed has taken a conservative estimate that this provides 103 bits of security [12].

As this field uses a prime 2048 bits of size, each group element requires 2048 bits of storage or 2KiB of storage.

Elliptic Curve Diffie-Hellman

The smallest elliptic curve supported by TLS appears to be curve25519, using the prime $2^{255} - 19$ as the field 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 requires 255 bits of storage, making the point requiring 510 bits of storage. Therefore, each group element requires approximately 0.5KiB of storage.

Performance of group operations

Assume that multiplying two 256-bit integers has cost C. Multiplication of two 2048-bit integers thus will cost $64C$ 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 this form can be transformed into the short Weierstrass form but not the other way around. Detailed in [13], 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,[4] 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 $M < C$.

Note how adding two curve points only costs $5M$, while multiplying in finite fields 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. In general, elliptic curves take much less space to store, providing similar bits of security while using approximately four times less storage for each group element, and is able to perform group operations at much faster speeds, providing an approximate 13x speedup compared to older methods.

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 its 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.

[4] In reality, squaring has slightly less costs than multiplying as the former can be optimized a bit more for efficiency.

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.