Cryptography and Curves

Cryptography is the art and science of secure communication and information techniques. From the Caesar cipher to the Enigma machine, cryptography has been a staple part of human history for as long as the need for secure communication has existed.

In this essay I hope to impart knowledge and understanding of encryption, problems that we face in cryptography, and how elliptic curves are used to enhance cryptography.

Alice and Bob, back at it again!

Say we have two people, Alice and Bob, and they want to send some messages to each other, without anyone else knowing what the contents of those messages are. To ensure their communication is secure and private, they need to use encryption.

Encryption is the process of scrambling original data, plaintext, into something unreadable, ciphertext, in a way that only authorised parties can convert it back and read it.

One of the earliest methods of encryption is the Caesar cipher, named after Julius Caesar. A Caesar cipher works by replacing each letter of the plaintext with a letter a fixed number of positions down the alphabet.

For example, a shift value of 4 would result in the following encodings.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
E	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*

So, using a Caesar cipher with a shift value of 4, Alice sends the message “I LOVE MATHS” to Bob. Bob would then receive “M PSZI QEXLW”. In order to decrypt this, Bob would need to do the opposite of what Alice did to encrypt it, shift each letter up the alphabet by 4.

This shift value is known as a key. In cryptography, a key is a piece of information, which when used in conjunction with a cryptographic algorithm, can encrypt and decrypt data.

The Caesar cipher shown above is a type of symmetric encryption (albeit very simple). Symmetric-key algorithms use the same key for encryption and decryption. A common example is AES, the Advanced Encryption Standard. Symmetric-key algorithms have one big requirement in order to work, both parties need to know the key. If Bob didn’t know what the key was, he would not be able to decrypt Alice’s message (or at least not quickly).

Key exchange

So, Alice and Bob now have a different problem, how can they securely exchange keys over an insecure channel? Whitfield Diffie and Martin Hellman tackled this exact question in 1976 [1]. This brings us onto the concept of the Diffie-Hellman key exchange: a method for establishing a shared secret key between two parties over an insecure channel.

The trick employed is that some mathematical functions, known as one-way functions, are much easier one direction and much harder in the reverse direction. A common analogy of the protocol is with colour mixing. In colour mixing, we can easily mix two paints together to produce a third paint, but it is very hard to unmix the third into the two original.

We will begin by looking at the original Diffie-Hellman key exchange scheme.

The process is as follows:

1. Alice and Bob need to agree on some initial parameters:
    a. A generator g. For example, 5
    b. A very, very large prime number p. For example, 11 (for ease of calculations).
    c. These parameters are in public space i.e. we assume that all attackers are aware of them.
2. Alice chooses a secret private integer a and computes $A = g^a \; mod \; p$ and sends it to Bob.

    For example,

    $$a = 2$$

    $$A = 5^2 \; mod \; 11 = 3$$

3. Bob chooses a secret private integer b and computes $B = g^b \; mod \; p$ and sends it to Alice.

    For example,

    $$b = 3$$

    $$B = 5^3 \; mod \; 11 = 4$$

4. Alice computes $S = B^a \; mod \; p$

    For example,

    $$S = 4^2 \; mod \; 11 = 5$$

5. Bob computes $S = A^b \; mod \; p$

    For example,

    $$S = 3^3 \; mod \; 11 = 5$$

6. Alice and Bob now have the shared secret key S.

Any outsider trying to eavesdrop, let's say Eve, will not be able to figure out what that shared secret is. The only pieces of information that are in public space are p, g, g^a, g^b.

Below is a table that shows which party knows what values.

Alice	Public	Bob
a $A = g^a \textit{ mod } p$ $S = B^a \textit{ mod } p$	g p $A = g^a \textit{ mod } p$ $B = g^b \textit{ mod } p$	b $B = g^b \textit{ mod } p$ $S = A^b \textit{ mod } p$

How is this secure?

The Diffie-Hellman key exchange relies on a hard problem known as the discrete logarithm problem: given g, p and $g^x \textit{ mod } p$ find x.

As long as p is a very, very large prime, then even the fastest computers would not be able to calculate x in a reasonable time.

As stated earlier, it is easy to mix two paints to produce a third paint, but very hard to unmix that third paint into its original components.

The Diffie-Hellman key exchange is an example of an asymmetric key technique and is a public key protocol. This is because it uses private and public keys to establish a shared secret. It is an asymmetric technique used to establish symmetric keys. It is important to note that Diffie-Hellman key exchange is not an encryption scheme in itself, only a key-exchange scheme. There are however encryption schemes based on the Diffie-Hellman key exchange such as ElGamal which I will not discuss here.

Overall, Diffie-Hellman allows two parties to agree on a common shared secret that is then used for subsequent communication using symmetric key encryption.

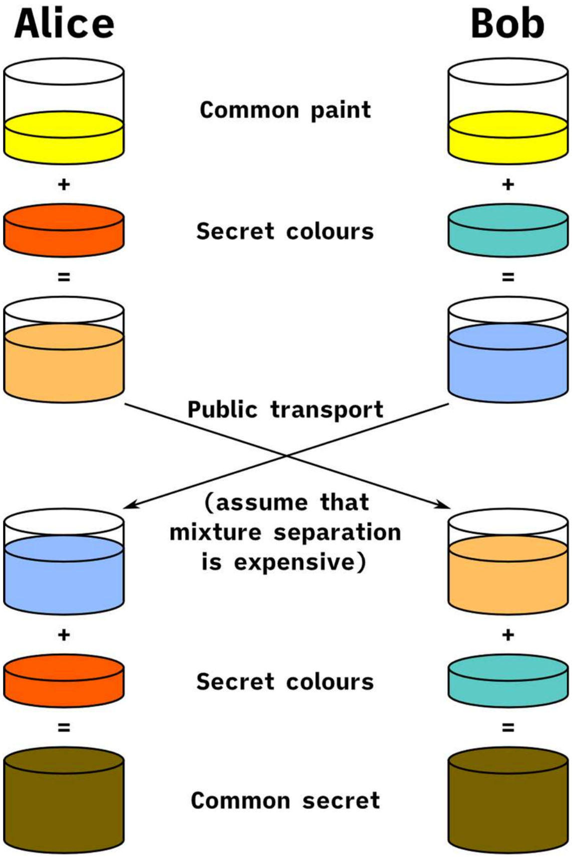


Figure 1 - illustration of the concept of the Diffie-Hellman key exchange using colour mixing [2]

Elliptic Curves

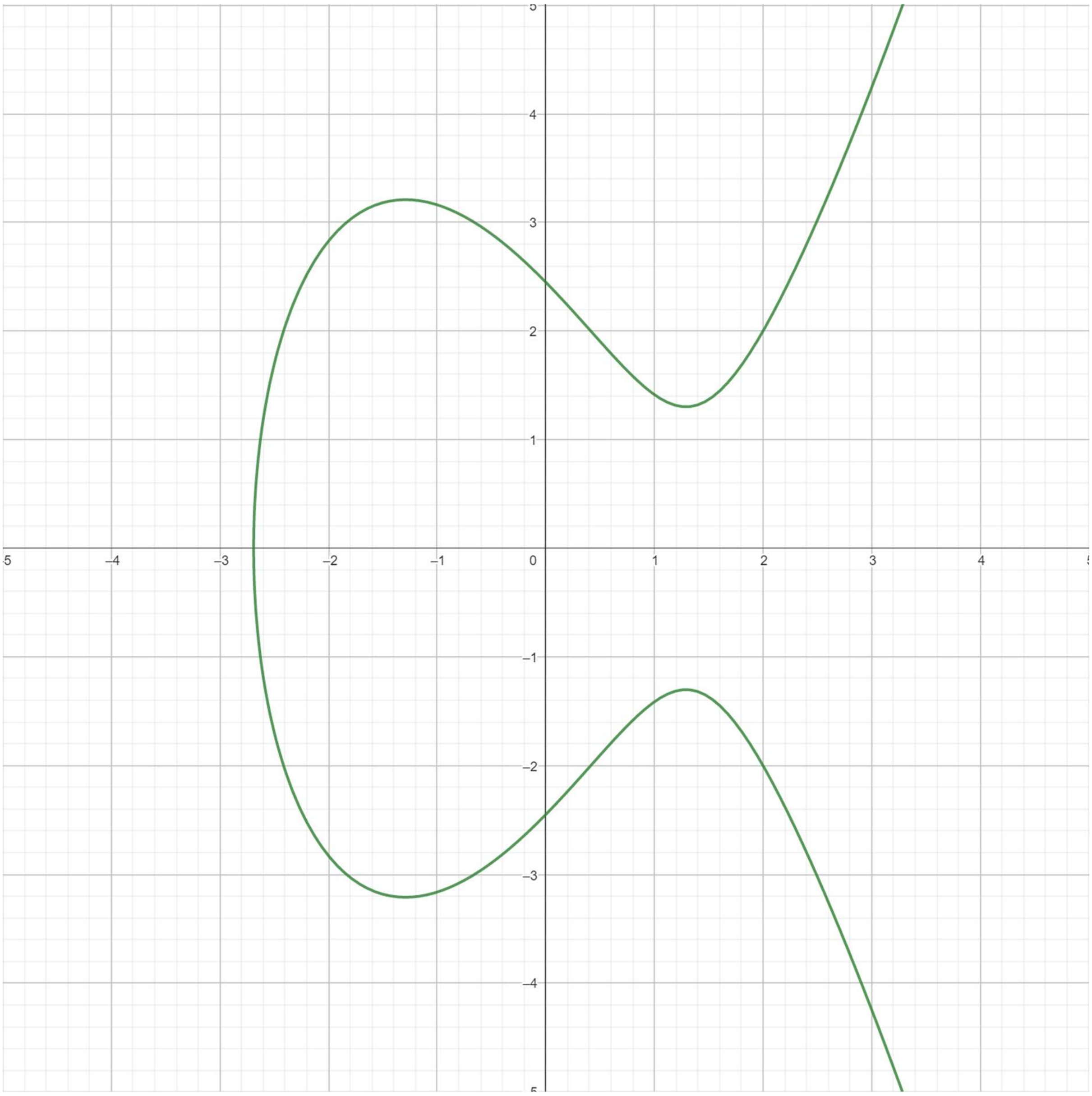
The previously mentioned Diffie-Hellman key exchange is the original implementation of the protocol, however, since 1976 there have been many implementations with one of them being Elliptic-curve Diffie-Hellman.

For now, we are going to step away from cryptography and dive into elliptic curves.

First, we need to define what elliptic curves are. In essence, an elliptic curve is the set of points that satisfy a specific mathematical equation, more specifically

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

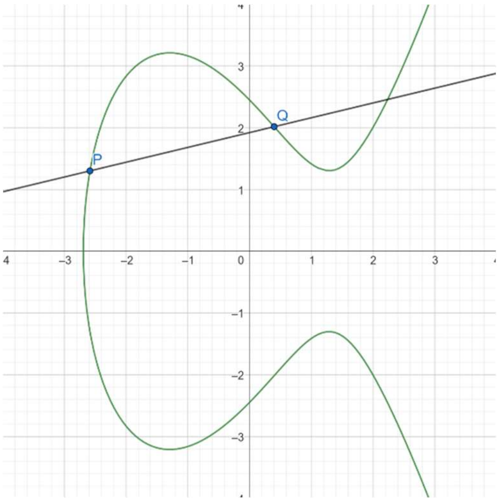
An example, $y^2 = x^3 - 5x + 6$, is shown below.

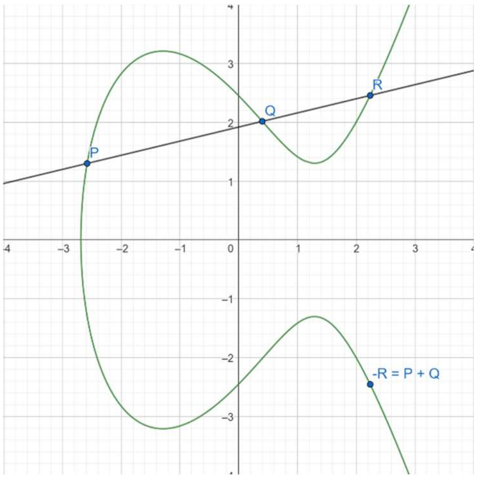


There are also some additional requirements to be satisfied. The curve is required to not have any cusps or self-intersections. This is represented algebraically as $4a^3 + 27b^2 \neq 0$.

One of the most interesting properties of elliptic curves is that if we have two points on the curve, P and Q, and draw the line through them, the maximum number of points that the line will intersect the curve at is 3.

Another interesting property is that the curve is symmetrical about the horizontal axis (This is because of the y^2).





With elliptic curves there is something called the group law that allows us to perform point addition, that is we can add two points on the curve, P and Q, and obtain a third point on the curve $P + Q$.

To do this we start with points P and Q. We draw the line through P and Q and obtain an intersection R. We then reflect this point across the horizontal axis and obtain $-R$. The point $-R$ is then $P + Q$.

This point addition can be calculated algebraically via the following process [3]:

1. Let $P = (x_1, y_1), Q = (x_2, y_2)$ and $P + Q = (x_3, y_3)$ be points on the elliptic curve $y^2 = x^3 + ax + b$.
2. Calculate $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.
3. Then,

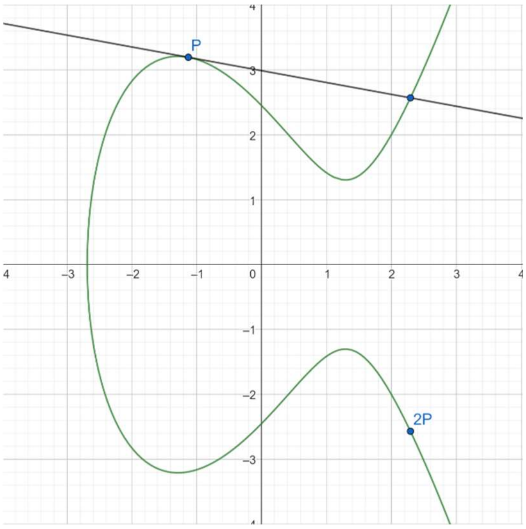
$$x_3 = \lambda^2 - x_1 - x_2$$

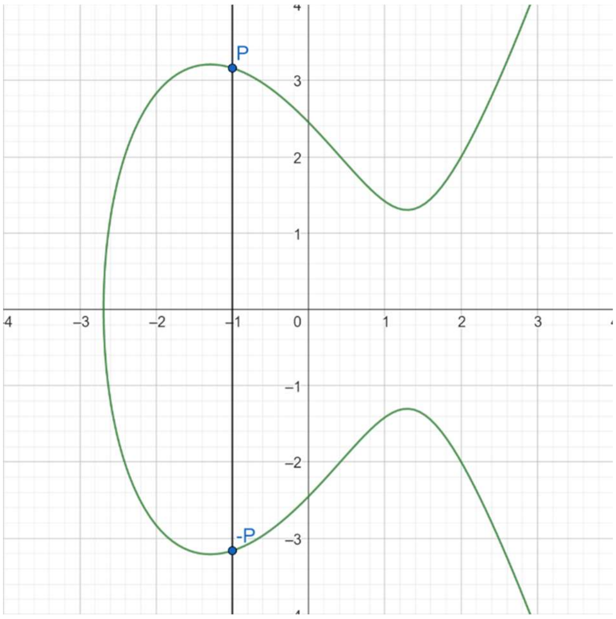
$$y_3 = \lambda(x_1 - x_3) - y_1$$

Furthermore, we can add points to themselves, that is $P + P = 2P$. This is also known as point doubling.

In order to do this, we draw the tangent line to the curve at P. There is then an additional intersection to the curve. We reflect this intersection point and call it $2P$.

Moreover, we can then also calculate $3P$ as $P + 2P$.





Another interesting point to note down is that if we try to add two points that are horizontal reflections of each other, P and $-P$, the line through them does not intersect the curve at a third point. This is known as negation.

We call $P + (-P) = O$ the point at infinity. The point at infinity, O, is then the additive identity i.e. $P + O = P$.

Another two consequences of the group law is that one, if we were to take a point $4P$ and add it to itself 3 times, that would result in the same point as taking a point $3P$ and adding it to itself 4 times, i.e. point addition is commutative; and two, performing $(P + Q) + R$ is no different to $P + (Q + R)$, i.e. point addition is associative.

We now realise that the points on the curve form an abelian group (very exciting!).

The group axioms satisfied are:

- Closure
    - When we perform point addition of two points on the curve, we get another point that is on the curve.
- Associativity
    - $(P + Q) + R = P + (Q + R)$
- Identity
    - For every point on the curve, the point at infinity is the unique identity.
    - Performing point addition on a point and the point at infinity always results in the original point.
    - $P + O = P$
- Inverse
    - For every point on the curve, the point given by the reflection across the horizontal axis is its inverse.
    - If we perform point addition on a point and its inverse we obtain the identity element, the point at infinity.
    - $P + (-P) = O$
- Commutativity
    - $P + Q = Q + P$

Scalar multiplication

Elliptic curves have another operation known as elliptic curve scalar multiplication.

It is defined as the repeated addition of a point on an elliptic curve.

We denote this as nP which is simply the addition of P to itself n times.

Say we wanted to obtain the point $100P$. The straight forward way of performing $P + P + P + \cdots + P$ is incredibly arduous and involves a lot of operations.

There are many better ways to perform elliptic curve scalar multiplication, and one of those methods is Double-and-add.

We can quickly calculate $100P$ as follows:

1. Convert n into its binary representation

   For example,

   $$n = 100$$
   $$n = 2^6 + 2^5 + 2^2$$

2. We shall denote point doubling as $dbl()$
   Then,

   $$P = 2^6P + 2^5P + 2^2P$$
   $$P = dbl\left(dbl\left(dbl\left(dbl\left(dbl(dbl(P))\right)\right)\right)\right) + dbl\left(dbl\left(dbl\left(dbl(dbl(P))\right)\right)\right) + dbl(dbl(P))$$

This calculation of $100P$ involves only 13 point doublings and 2 point additions, a huge saving over calculating $P + P + P + \cdots + P$.

When the scalar values are even larger, the efficiency increases even more.

Elliptic curves over finite fields

One more thing we need to cover is that the elliptic curves used for cryptography do not look like the above. Instead, we restrict ourselves to numbers in a fixed range.

Below is the same curve $y^2 = x^3 - 5x + 6$ over the finite field with 97 elements, $\mathbb{F}_{97}$.

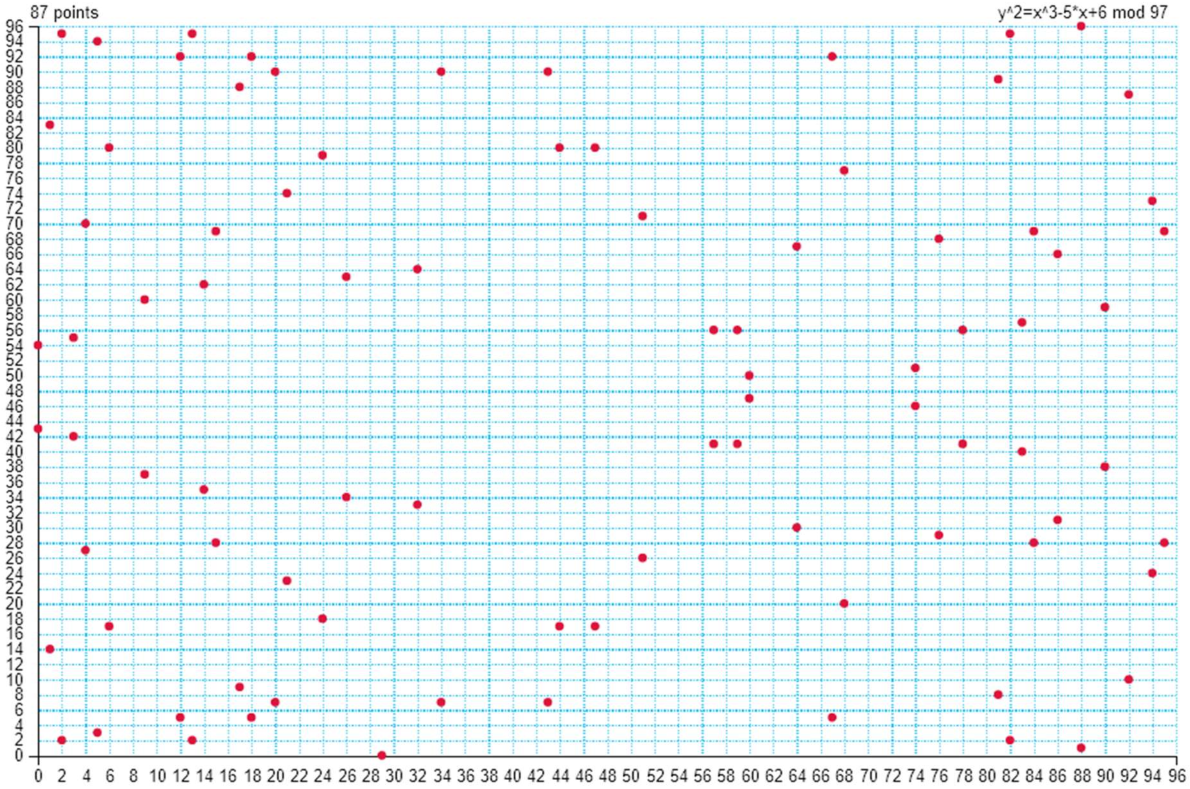


Figure 2 - an elliptic curve over a finite field [4]

This most certainly does not look like a curve in the traditional sense, but it is.

Notice how it still retains horizontal symmetry.

We can still do all of the same operations with an elliptic curve over a finite field such as point addition. When we reach a border, we just wrap around to the other side.

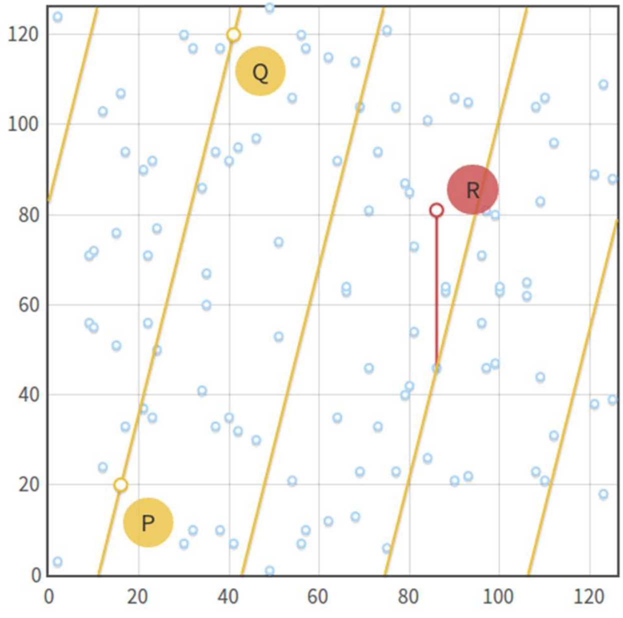


Figure 3 - point addition of elliptic curves over finite fields [5]

Elliptic-curve Diffie-Hellman

For those that were paying close attention earlier, the Diffie-Hellman key exchange protocol can actually be generalised to finite cyclic groups as follows:

1. Alice and Bob will agree on a natural number n and a generator g in the finite cyclic group G of order n. G, n, g are in public space and are assumed to be known by all attackers.
2. Alice chooses a random natural number, a, such that $1 < a < n$. She computes g^a of G and sends it to Bob.
3. Bob chooses a random natural number, b, such that $1 < b < n$. He computes g^b of G and sends it to Alice.
4. Alice then computes $(g^b)^a = g^{ba}$ of G.
5. Bob then computes $(g^a)^b = g^{ab}$ of G.
6. The shared secret key is then the group element $g^{ab} = g^{ba}$.

Now recall that the points on an elliptic curve over a finite field form a finite cyclic abelian group. This means that we can actually use elliptic curve cryptography as a drop-in replacement for the modular exponentiation used in the original implementation.

Back to the original scenario, Alice and Bob need to securely exchange keys over an insecure channel.

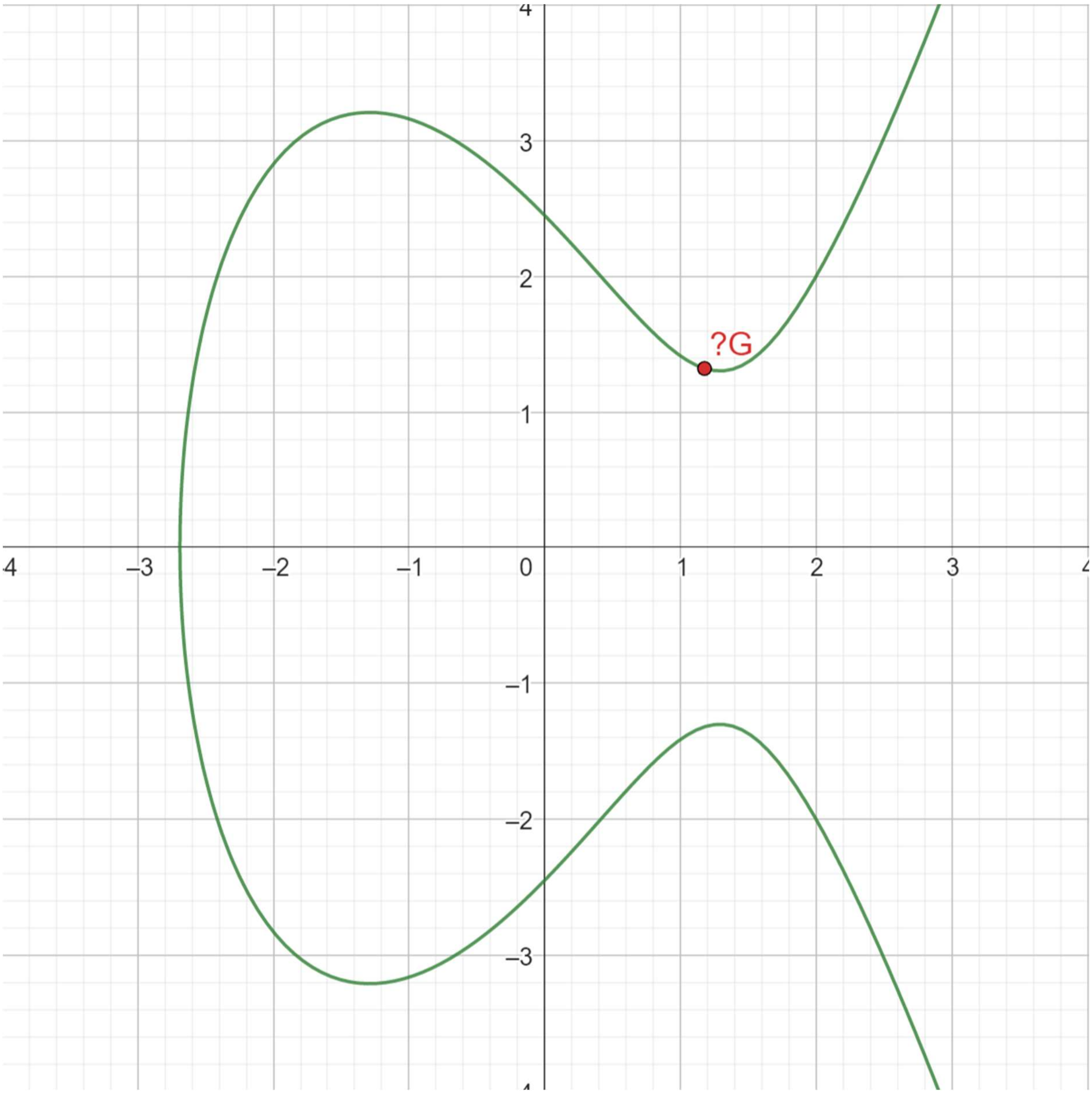
The process for Elliptic-curve Diffie-Hellman is as follows:

1. Alice and Bob need to agree on some parameters:
    a. The curve they are going to use i.e. values of a and b.
    b. A generator point G.
    c. n, the order of G (How many times G needs to be added to itself to obtain the point at infinity O).
    d. A field $\mathbb{F}_p$ that the curve is defined over where p is a prime.
    e. These parameters are in public space i.e. we assume that all attackers are aware of them.
2. Alice chooses a random integer a with $1 \leq a \leq n - 1$ and computes aG and sends it to Bob.
3. Bob chooses a random integer b with $1 \leq b \leq n - 1$ and computes bG and sends it to Alice.
4. Alice then computes abG and Bob computes baG.
5. Their shared secret key is then the point $abG = baG$.

How is this secure?

Elliptic-curve Diffie-Hellman relies on a very similar problem to standard Diffie-Hellman, the elliptic curve discrete logarithm problem: given G and nG, find n.

If I give you this point, $?G$, and ask you how many times have I added G to itself to obtain it, it would be very difficult to calculate the answer. It could be $5G$, it could be $5{,}000{,}000{,}000G$.



On the other hand, if we needed to calculate a point like $5{,}000{,}000{,}000G$, it would be *relatively* quick due to scalar multiplication algorithms like Double-and-add.

This provides our one-way function for elliptic curve cryptography.

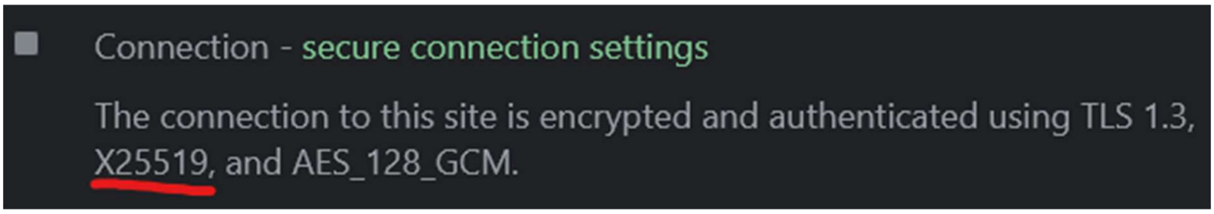
Improvements

So why are we doing all of this? Doesn't this all seem a lot more complicated than the standard procedure of modular exponentiation? Well, yes, it is a lot more complicated, BUT there is a huge benefit in efficiency. Elliptic curves allow us to get away with much smaller keys for the same level of security. Small keys are extremely important, especially when a lot more cryptography nowadays is done on less powerful devices such as mobile phones. Elliptic curve cryptography helps save time, power and computational resources.

Conclusion

So, to sum up everything discussed, Alice and Bob want to use symmetric encryption to securely send each other messages. To do this, they both need a shared secret key. The key is obtained by the Diffie-Hellman key exchange and elliptic curves can be used to improve the efficiency of Diffie-Hellman.

All of this seems awfully complicated but secure communication is more important than ever now. For instance, you can download and read this essay and additionally visit www.tomrocksmaths.com! In fact, you can go to www.tomrocksmaths.com and check out the developer tools security information and see elliptic curve cryptography in use for yourself!



Connection - secure connection settings

The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.

X25519 is a Diffie-Hellman function that uses an elliptic curve known as Curve25519!

Cryptography has evolved a lot throughout the *(thousands of)* years with many talented mathematicians advancing the field. From simple Caesar ciphers to the Enigma machine to now elliptic curves, and in the future (*or now?*), quantum cryptography. For as long as communication exists, cryptography exists, and so mathematicians will keep on trying to find new ways and methods.

Thanks for reading!

References

[1] W. Diffie and M. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory,* pp. 644-654, 1976.

[2] A. Vinck, *Introduction to public key cryptography,* 2011, p. 16.

[3] B. Lynn, “Elliptic Curves - Explicit Addition Formulae,” [Online]. Available: https://crypto.stanford.edu/pbc/notes/elliptic/explicit.html.

[4] S. Grau, “Curves over Finite Fields,” [Online]. Available: https://graui.de/code/ffplot/.

[5] A. Corbellini, “Elliptic Curve Cryptography: finite fields and discrete logarithms,” [Online]. Available: https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/.