Topics in Discrete Mathematics - Cryptography

Daniel P. Martin

University of Bristol 2018-2019

1 Introduction

> "How long do you want these messages to remain secret?[...]
> I want them to remain secret for as long as men are capable of evil."
> – Neal Stephenson, Cryptonomicon

Cryptography has been around for hundreds of years in various forms. In this part of the course we apply mathematical underpinnings both to historical and modern cryptography. We show that lessons learnt from older schemes can be used to design schemes that are secure.

However, before we can do this, just what is cryptography? and what do we want from it? Historically the word cryptography was synonymous with encryption - it was all about making your messages unreadable to unwanted viewers. Today cryptography has three major uses:

Confidentiality A message should only be readable by the desired people.
Integrity It should not be possible to change a message.
Authenticity Given a message, a user must be able to know who it came from.

However, it also has many other uses, one of which you will see in this course. Of the three points above this part of the course will focus on confidentiality but a similar mathematical treatment can be given to the other two.

1.1 Notational Comments

Cryptography as a subject is studied in both Mathematics and Computer Science departments. For example the Crytography Group at the University of Bristol is in the Computer Science deparment. Due to this cryptography uses a mixture of both mathematics and computer science notation. In this section I will introduce some notation from computer science which will be vital for this course.

$a \leftarrow f(n)$**:** Evaluate the function f on input n and assign the answer to a.

$a \xleftarrow{\$} f(n)$**:** Evaluate the randomised function f on input n and assign the answer to a. The randomness should be chosen as described by the function.

$a \xleftarrow{\$} \mathcal{X}$**:** Sample from the distribution $\mathcal{X}$ and assign the answer to a.

$\mathcal{E}_k(m)$ This is an alternate form for $\mathcal{E}(k, n)$ - evaluating the function $\mathcal{E}$ on inputs k and n. The input moved to the subscript tends to hold particular importance, normally it is a key.

2 Historical Cryptography

> "A people without the knowledge of their past history, origin and culture is like a tree without roots."
> – Marcus Garvey

Cryptography dates back to at least Julius Caesar, who was known for encoding messages to his generals. This way if the message ever fell into enemy hands it could not be read. This was documented by Suetonius in De Vita Caesarum, Divus Iulius.

> "If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others."

From this we can define Caesar's cipher as follows.

Definition 1 (Caesar Cipher). *To encrypt a message each letter of the alphabet is cyclically shifted by 3 places. So a becomes d, b becomes e and so forth until the end of the alphabet where z is replaced by c, y is replaced by b and x by a.*

The problem with Caesar's cipher is that once it is known, it is trivial to break. That is the secret is in the method. Kerckhoff stated that the secret should not be within the method but in a key.

Definition 2 (Kerckhoff's Principle). *A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.*

Caesar's cipher can be adjusted to fit within this framework. For this we require a mapping between the letters of the alphabet to the numbers $0, \ldots, 25$ - the obvious choice being $a = 0, b = 1, \ldots$. We refer to this invertible mapping as $M : \{a, \ldots, z\} \to \mathbb{Z}_{26}$.

Definition 3 (Shift Cipher). *Given a key $k \in \mathbb{Z}_{26}$ the shift cipher encrypts a message by replacing each letter X with $M^{-1}(M(X) + k \mod 26)$.*

Decryption works by replacing each letter Y with $M^{-1}(M(Y) - k \mod 26)$.

Exercise 1. Show that decryption of the shift cipher is the inverse of encryption.

This method does meet Kerckhoff's principle: the method is public knowledge but the secret key is the information of how much each letter is shifted by. However, despite following this principle it is still easy to break because there are only twenty-six possible keys. Thus an adversary could try all possible keys and see which text "makes sense".

One way to increase the size of the key space is by replacing the shift with an affine function.

Definition 4 (Affine Cipher). *Given a key $a, b \in \mathbb{Z}_{26}^2$ such that $gcd(a, 26) = 1$ the affine cipher encrypts a message by replacing each letter X with $M^{-1}(a \cdot M(X) + b \mod 26)$.*

Decryption works by replacing each letter Y with $M^{-1}(a^{-1} \cdot (M(Y) - b) \mod 26)$.

Exercise 2. Show that if $a = 13$ then messages cannot be decrypted.

While the affine function increases the key space from 26 to approximately $26^2 = 676$, it is still easy for an adversary to check all possible keys to learn the message. Thus we require a method that greatly increases the potential key space.

The first observation is that as long as the mapping $f : \mathbb{Z}_{26} \to \mathbb{Z}_{26}$ is a permutation, the message will be decodable.

Definition 5 (Substitution Cipher). *Let π be a permutation on the letters $a, \ldots, z$. Then encryption works by replacing a letter x by $\pi(x)$ while decryption works by replacing y by $\pi^{-1}(y)$*

There are $26! \approx 2^{88}$ permutations on the alphabet and thus it is no longer possible to try all the permutations. However, it has been shown that there exists another method to efficiently attack this cipher. The observation that allows the attack to take place is that the letters in the English language do not occur equally often. Thus the frequency of letters in the English language can be compared to the frequency of letters in the ciphertext to decode the message. This is known as *frequency analysis*.

Exercise 3. Use frequency analysis to decode the following piece of text:

MX. GIA MXP. ANXPELF, OJ INMTLX JONX, QXCWLZ AXCWL, KLXL QXONA ZO PGF ZYGZ ZYLF KLXL QLXJLSZEF IOXMGE, ZYGIU FON WLXF MNSY. ZYLF KLXL ZYL EGPZ QLOQEL FON'A LRQLSZ ZO TL CIWOEWLA CI GIFZYCIB PZXGIBL OX MFPZLXCONP, TLSGNPL ZYLF HNPZ ACAI'Z YOEA KCZY PNSY IOIPLIPL. MX. ANXPELF KGP ZYL ACXLSZOX OJ G JCXM SGEELA BXNIICIBP, KYCSY MGAL AXCEEP. YL KGP G TCB, TLLJF MGI KCZY YGXAEF GIF ILSU, GEZYONBY YL ACA YGWL G WLXF EGXBL MNPZGSYL. MXP. ANXPELF KGP ZYCI GIA TEOIAL GIA YGA ILGXEF ZKCSL ZYL NPNGE GMONIZ OJ ILSU, KYCSY SGML CI WLXF NPLJNE GP PYL PQLIZ PO MNSY OJ YLX ZCML SXGICIB OWLX BGXALI JLISLP, PQFCIB OI ZYL ILCBYTOXP. ZYL ANXPELFP YGA G PMGEE POI SGEELA ANAELF GIA CI ZYLCX OQCICOI ZYLXL KGP IO JCILX TOF GIFKYLXL. ZYL ANXPELFP YGA LWLXFZYCIB ZYLF KGIZLA, TNZ ZYLF GEPO YGA G PLSXLZ, GIA ZYLCX BXLGZLPZ JLGX KGP ZYGZ POMLTOAF KONEA ACPSOWLX CZ. ZYLF ACAI'Z ZYCIU ZYLF SONEA TLGX CZ CJ GIFOIL JONIA ONZ GTONZ ZYL QOZZLXP. MXP. QOZZLX KGP MXP. ANXPELF'P PCPZLX, TNZ ZYLF YGAI'Z MLZ JOX

PLWLXGE FLGXP; CI JGSZ, MXP. ANXPELF QXLZLIALA PYL ACAI'Z YGWL G PCPZLX, TLSGNPL YLX PCPZLX GIA YLX BOOA-JOX-IOZYCIB YNPTGIA KLXL GP NIANXPELFCPY GP CZ KGP QOPPCTEL ZO TL. ZYL ANXPELFP PYNAALXLA ZO ZYCIU KYGZ ZYL ILCBYTOXP KONEA PGF CJ ZYL QOZZLXP GXXCWLA CI ZYL PZXLLZ. ZYL ANXPELFP UILK ZYGZ ZYL QOZZLXP YGA G PMGEE POI, ZOO, TNZ ZYLF YGA ILWLX LWLI PLLI YCM. ZYCP TOF KGP GIOZYLX BOOA XLGPOI JOX ULLQCIB ZYL QOZZLXP GKGF; ZYLF ACAI'Z KGIZ ANAELF MCRCIB KCZY G SYCEA ECUL ZYGZ.

A separate observation, to improve the security of the shift cipher, is that the same shift does not have to be applied to each position of the message.

Definition 6 (Vigenère Cipher). *The Vigenère cipher can be seen as applying a different shift cipher to different positions in the message. The key can be seen as a string S, where S_i represents applying the shift $M[S_i]$ to the message character. If $|S| = n$ and the character is at position i in the text then S_j is used as the key where $j = i \mod 26$.*

This seems to make the scheme a lot more secure than previous attempts. Intuitively it does this by 'smoothing' out the frequency with which each letter appears in the ciphertext. It took hundreds of years since its proposal for an attack strategy to be devised. Informally, the attack works as follows. You look for places where two consecutive characters appear multiple times in the text. The hope is that these represent the same plaintext characters, which modulo n are in the same position and hence they encrypt to the same pair of characters. The distance between all such pairs is calculated. It follows that n must be a multiple of the gcd of all of these distances. Given this information, it is possible to make a guess at n, before solving the n shift ciphers individually.

The weakness of the Vigenère cipher arises from the key being too short. It would be possible to take this idea to the extreme by choosing the key to have the same length as the message. This is referred to as the one time pad.

Definition 7 (One Time Pad). *Given a message P of length n, the One Time Pad has a key K of length n and encrypts P_i by shifting it by $M^{-1}[K_i]$.*

For each possible text of length n there exists a key of length n that maps the plaintext to a given ciphertext of length n. Therefore, no information can be gained about the plaintext from the ciphertext and thus the One Time Pad provides perfect security.

Exercise 4. Show what can be learnt if the key to a one time pad is used twice.

Given that each key can only be used once, the one time pad simply moves the problem from sending a length n message securely, to that of moving a length n key. Thus, even though the one time pad provides perfect security, it is not practical to use en masse.

The pinnacle of the historic cipher, both from a security point of view and an analysis point of view, is ENIGMA. ENIGMA was the cipher suite used by the German forces during the 2^{nd} World War, and was broken by efforts at Bletchley Park, lead by Alan Turing. While the mathematics of this is fascinating we unfortunately do not have time to cover it in this course.

3 Public Key Encryption

In this section we move over to modern cryptography where all of the schemes have a mathematical underpinning. Where historically, schemes were assumed to be secure until they were broken, in the modern setting schemes are designed such that breaking them can be shown to imply that it is possible to solve a math problem that is believed to be hard.

Most of the schemes that have such a mathematical underpinning are public key, or asymmetric, schemes. The historical schemes discussed above were secret key, or symmetric schemes. In a symmetric scheme the same key is used for encryption and decryption. In public key cryptography a different key is used for encryption and decryption.

If N people wish to talk using a symmetric scheme, each pair requires a secret key and thus $N(N-1)/2$ keys are required.[1] By comparision if, an asymmetric scheme where only N keys are required. Another advantage of asymmetric schemes is that you don't have the issue of securely sharing keys because you keep the decryption key secret and can widely publish the encryption key. However, there are also downsides to asymmetric schemes. The first is the challenge of verifying that a public key found online belongs to who it claims it belongs to. The second issue is that asymmetric schemes tend to be a lot slower than symmetric schemes, so that symmetric schemes tend to be preferable for sending large amounts of data.

A public key scheme can be described as three functions key generation $\mathcal{KG}$, encryption $\mathcal{E}$ and decryption $\mathcal{D}$, with the following syntax:

$$\begin{aligned}\mathcal{KG} &: \{1^*\} \to \mathcal{SK} \times \mathcal{PK} \\ \mathcal{E} &: \mathcal{SK} \times \mathcal{M} \to \mathcal{C} \\ \mathcal{D} &: \mathcal{PK} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}\end{aligned}$$

such that:

$$\begin{gathered}\forall \lambda \in \mathbb{Z}, m \in \mathcal{M}, (sk, pk) \xleftarrow{\$} \mathcal{KG}(1^\lambda) : \\ \mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m\end{gathered}$$

Exercise 5. Give the corresponding syntax and correctness properties for a symmetric encryption scheme.

[1] If fewer keys were used there would exist a larger group of people who shared a key and thus there would be someone who could read messages not intended for them.

The above simply defines the syntax that all public key systems must define. In this section we mainly focus on the correctness of the schemes presented. However, when designing a scheme you have to choose which security property it will meet and show that it does. Some common security properties used for encryption are:

- Given a ciphertext, can learn nothing about the underlying plaintext.
- Given a ciphertext of one of two messages, can't determine which it encrypts.
- Can't tell a real ciphertext from a random string.

We are now in a position to give our first scheme.

3.1 RSA

In this section we introduce our first scheme with a mathematical underpinning. When designing schemes they all follow essentially the same format: we begin by stating a problem which we assume to be hard, along with justification as to why it is believed to be hard. We then give the scheme and show that it works as expected. Finally, we have to justify why breaking our scheme should be as hard as solving the problem we assumed hard to begin with. Ideally this should be in the form of a proof that shows how breaking our scheme solves the hard problem. We now follow this format for RSA, and begin by giving our "hardness assumption".

Definition 8 (Integer Factorisation Problem). *Given an integer x, the integer factorisation problem asks to return all primes p such that p divides x.*

Definition 9 (Integer Factorisation Assumption). *The integer factorisation problem cannot be solved "efficiently".*

Now that we have the definition, we want to build an encryption scheme that relies on this assumption. But first we need an additional definition, which will be useful for the RSA algorithms.

Definition 10 (Euler's Totient Function). *Euler's totient function $\varphi(x)$ is defined as:*

$$\varphi(x) = |\{y : 1 \leq y \leq x \ gcd(x, y) = 1\}|$$

That is, the number of integers less than x that are coprime to x.

We can now give some properties about Euler's totient function which will come in use shortly.

Theorem 1. *Given $m, n \in \mathbb{Z}$ with $gcd(m, n) = 1$:*

$$\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$$

Proof. We begin by defining the following three sets:

$$A = \{y : 1 \leq y \leq x \ gcd(m, y) = 1\}$$
$$B = \{y : 1 \leq y \leq x \ gcd(n, y) = 1\}$$
$$C = \{y : 1 \leq y \leq x \ gcd(m \cdot n, y) = 1\}$$

Then we can define the function $f : C \to A \times B$, as follows:

$$f(x) = (x \mod m, x \mod n)$$

If $f(x) = f(y)$ then $x = y \mod m$ and $x = y \mod n$. Thus by the Chinese Remainder Theorem $x = y \mod m \cdot n$.

If $(a, b) \in A \times B$, by the Chinese Remainder Theorem there exists a $c \in C$ such that $f(a) = (c, d)$

Therefore f is a bijection between C and $A \times B$ and the result follows.

Theorem 2. *For a prime p and $k \in \mathbb{Z}^+$ we have that:*

$$\varphi(p^k) = p^{k-1}(p - 1)$$

Proof. Since p is prime the only possible values of $gcd(m, p^k)$ are $1, p, p^2, \ldots, p^k$. The only way for the gcd not to be 1 is for m to be a multiple of p. The multiples of p are $p, 2p, 3p, \ldots, p^k$: there are p^{k-1} of them. The other $p^k - p^{k-1}$ numbers have a gcd of 1 and the result follows.

Using the above we can derive a closed form expression for φ for all inputs.

Theorem 3. *Euler's totient function has the following closed form:*

$$\varphi(x) = x \prod_{p|x} (1 - \frac{1}{p})$$

for unique primes p that divide x.

Proof. This follows directly from Theorems 1 and 2.

We are now in a position to give the RSA algorithm.

$\mathcal{KG}(1^\lambda)$:	$\mathcal{E}_{pk}(m)$:	$\mathcal{D}_{sk}(c)$:
Pick p, q prime of λ bits	$(e, N) \leftarrow pk$	$(d, N) \leftarrow sk$
$N \leftarrow p \cdot q$	$c \leftarrow m^e \mod N$	$m \leftarrow c^d \mod N$
Choose e: $gcd(e, \varphi(N)) = 1$	**return** c	**return** m
$d \leftarrow e^{-1} \mod \varphi(N)$		
$sk \leftarrow (d, N)$		
$pk \leftarrow (e, N)$		
return (sk, pk)		

Before we prove correctness, we will introduce the required theorem.

Theorem 4 (Euler's Theorem). *Let $N \in \mathbb{Z}$ and $a \in \mathbb{Z}_N$ such that a is coprime to N, then:*

$$a^{\varphi(N)} = 1 \mod N$$

In this course we will prove the theorem for the special case where N is prime, and it will be left as an exercise to generalise the proof. In this special case it is referred to as Fermat's Little Theorem.

Theorem 5 (Fermat's Little Theorem). *Let p be a prime and $a \in \mathbb{Z}_p$ then we have that:*

$$a^p = a \mod p$$

To prove Fermat's little Theorem we require the following Lemma.

Lemma 1. *Let p be a prime, $a \in \mathbb{Z}_p$ and consider the list of numbers:*

$$a, 2 \cdot a, \ldots, (p-1)a$$

If this list is reduced modulo p, the resulting list can be rearranged as:

$$1, 2, \ldots, p-1$$

Proof. Since a is coprime to p and, for $1 \leq x \leq p-1$, x and p are coprime we have that no $a \cdot x$ is congruent to 0 mod p. Therefore we know that all $a, 2 \cdot a, \ldots, (p-1)a$ when reduced modulo p must be found in the set $1, 2, \ldots, p-1$.

However, if we had that $a \cdot x = a \cdot y \mod p$ by the cancellation law, we would have that $x = y \mod p$, thus we get that all of $a, 2 \cdot a, \ldots, (p-1)a$ must be distinct modulo p.

We are now in a position to give a proof of Fermat's Little Theorem.

Proof (Fermat's Little Theorem). Using Lemma 1 we have that:

$$\prod_{i=1}^{p-1} i \cdot a = \prod_{i=1}^{p-1} i \mod p$$
$$a^{p-1}(p-1)! = (p-1)! \mod p$$

Since p is prime, p is coprime with all $1 \leq x \leq p-1$ and thus is coprime with $(p-1)!$. Hence, we can cancel $(p-1)!$ on each side giving:

$$a^{p-1} = 1 \mod p$$

and the result follows.

Exercise 6. Show how the proof of Fermat's Little Theorem can be converted into the proof of Euler's Theorem.

Hence to show that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$ we have the following:

$$\begin{aligned} c^d &= (m^e)^d \mod N \\ &= m^{e \cdot d} \mod N \\ &= m^{k \cdot \varphi(N)+1} \mod N \\ &= m \mod N \end{aligned}$$

Hopefully, it is fairly clear that breaking this scheme requires solving the integer factorisation problem. Given (e, N), it appears that the only way to break the scheme is to learn d. The adversary, trying to break the scheme, knows that $e \cdot d = 1 \mod \varphi(N)$ and thus if they can learn $\varphi(N)$ they can recover d. However, by definition, to calculate $\varphi(N)$ from N, they adversary must learn p and q, which requires factoring N.

However, breaking RSA doesn't require the ability to factor arbitrary integers, just integers that are the product of two (equally sized) primes. This leads to the following hardness assumption.

Definition 11 (RSA Problem). *Given an integer $x = p \cdot q$, for p, q equally sized primes, the RSA problem asks to return p and q.*

Definition 12 (RSA Assumption). *The RSA problem cannot be solved "efficiently".*

3.2 ElGamal

One of the downsides of RSA is that if you encrypt the same message twice, you get the same ciphertext. This may be an undesirable property, for example consider a scenario where people are sending their encrypted votes to someone for counting. Ideally, it shoudn't be possible to figure out who voted the same way as you. In this section we introduce a new scheme, built upon a different hardness assumption, which avoids this pitfall.

Again we follow the same methodology as with RSA, beginning with the hardness assumption.

Definition 13 (Discrete Logarithm Problem). *Given a group $\mathbb{G}$, a generator g and an element g^x find x.*

Exercise 7. Explain why the discrete logarithm problem is easy in the group $(\mathbb{Z}_p, +)$ for prime p.

When discussing schemes, they are often described in terms of arbitrary cyclic groups in which the discrete logarithm problem is hard. However, to actually use a cryptographic scheme a suitable group must be chosen. While any group that meets these properties can be utilised, a popular choice is elliptic curve groups. These will be discussed in more detail in Section 4.

In fact, depending on what properties our crypto scheme requires, we may be able to use a weaker assumption.

Definition 14 (Computational Diffie-Hellman Problem). *Given a group $\mathbb{G}$, a generator g and two elements g^x, g^y, find $g^{x \cdot y}$.*

Exercise 8. Show that the Computational Diffie-Hellman assumption implies the Discrete Logarithm assumption.

Definition 15 (Decisional Diffie-Hellman Problem). *Given a group $\mathbb{G}$, a generator g and two elements g^x, g^y and either $g^{x \cdot y}$ or g^r determine which case you are in.*

Exercise 9. Show that the Decisional Diffie-Hellman assumption implies the Computational Diffie-Helman assumption.

The corresponding assumptions for these problems simply state that it is "hard" for the problem to be solved. We are now in a position to give the ElGamal encryption scheme.

$\mathcal{KG}(1^\lambda)$:
Choose $\mathbb{G} = \langle g \rangle$, of order $q \approx 2^\lambda$
$x \xleftarrow{\$} \{1, \ldots, q-1\}$
$h \leftarrow g^x$
$sk \leftarrow (\mathbb{G}, q, g, h, x)$
$pk \leftarrow (\mathbb{G}, q, g, h)$
return (sk, pk)

$\mathcal{E}_{pk}(m)$:
$(\mathbb{G}, q, g, h) \leftarrow pk$
$r \xleftarrow{\$} \{1, \ldots, q-1\}$
$c_1 \leftarrow g^r$
$c_2 \leftarrow m \cdot h^r$
return (c_1, c_2)

$\mathcal{D}_{sk}(c)$:
$(\mathbb{G}, q, g, h, x) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$s \leftarrow c_1^x$
$m \leftarrow c_2 \cdot s^{-1}$
return m

This version of ElGamal chooses the group in the key generation step. In practice the group tends to be chosen in advance. This allows implementations to have efficient support for the most commonly used groups.

As before we need to show that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$:

$$\begin{aligned} c_2 \cdot s^{-1} &= c_2 \cdot c_1^{-x} \\ &= m \cdot h^r \cdot (g^r)^{-x} \\ &= m \cdot g^{x \cdot r} \cdot g^{-x \cdot r} \\ &= m \end{aligned}$$

The advantage of ElGamal is that it can be shown that the existence of an attack distinguishing between the encryption of a message and a random group element implies that the DDH assumption is false. This is the ideal situation to be in because then cryptography only has to rely on a small handful of assumptions which have been rigorously tested.

4 Elliptic Curves

A lot of cryptographic schemes are built using cyclic groups. While the proofs of security and correctness can work over an abstract group, for a scheme to be implemented and used, a particular group must be chosen. A current favourite

choice of group is an elliptic curve group. This is due to its compact representation and the belief that none of the group structure can be exploited to solve the Discrete Logarithm problem.

We define an elliptic curve E, over a finite field K, by the following equation:

$$y^2 = x^3 + a \cdot x + b$$

where $a, b \in K$. The curve also needs to be non-singular (i.e it has no cusps, self intersections or isolated points). A curve is non-singular if, and only if, the discriminant Δ is non-zero, where:

$$\Delta = -16(4 \cdot a^3 + 27 \cdot b^2)$$

Exercise 10. Show that if $y^2 = x^3 + a \cdot x + b$ has a double root α then $a = -3 \cdot \alpha^2$ and $b = 2\alpha^3$ and thus $\Delta = 0$.

This definition also requires that the field has neither characteristic two nor three.[2] If the characteristic is two or three then curves with more terms in the equation need to be considered. From this point forward we will only discuss fields that do not have characteristic two or three.

Figure 1 gives an example of two elliptic curves over the real numbers. Broadly speaking these represent one of each "class" of elliptic curve; one which is a single unbroken line and one formed of two parts. The choice of type to use for cryptography tends to be personal preference, but there tends to be a country based split as to which is used. Figure 2 gives the same elliptic curve over the finite field of integers modulo five.

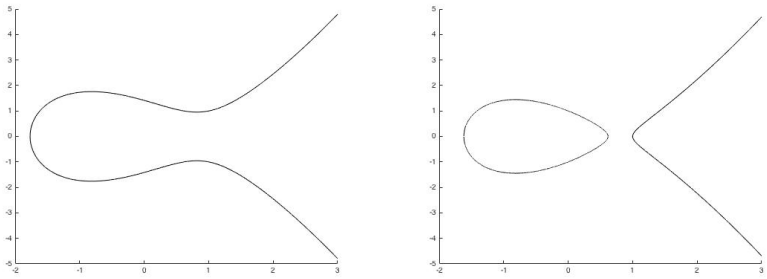

Fig. 1: An example of two elliptic curves.

[2] The characteristic of a field is the number of times the multiplicative identity needs to be added together to give the additive identity.

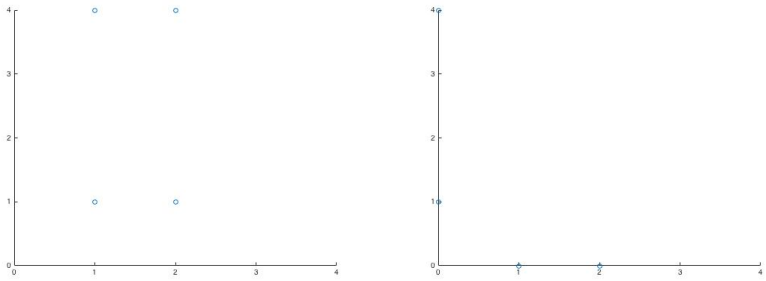

Fig. 2: An example of two elliptic curves over $\mathbb{F}_5$.

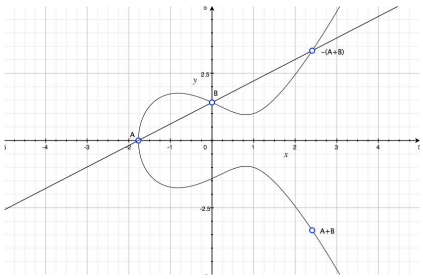

Fig. 3: A graphical representation of addition on an elliptic curve.

Let $E(K)$ be the set of coordinates that satisfy the elliptic curve, together with a point at infinity $\mathcal{O}$. To define an elliptic curve group, a notion of addition is required on the set. Addition can be defined graphically (see Figure 3): to add $A \in E(K)$ to $B \in E(K)$, a straight line is drawn between the two points, and the third point that intersects the line is denoted as $-(A+B)$.[3] To determine $A+B$ the sign of the y coordinate of $-(A+B)$ is negated. This is equivalent to mirroring the point along the line $x = 0$. We will now define addition formally for prime fields[4]: Given two points $A = (x_a, y_a) and B = (x_b, y_b)$, $C = A + B$ is calculated as follows when $x_a \neq x_b$:

$$
\begin{aligned}
s &= \frac{y_a - y_b}{x_a - x_b} \\
x_c &= s^2 - x_a - x_b \\
y_c &= y_a + s \cdot (x_c - x_a)
\end{aligned}
$$

If $x_a = x_b$ and $y_a = -y_b$ then C is the point at infinity. If $A = B$:

[3] If the line only crosses two points, the third point is denoted the point at infinity.
[4] Similar formulas can be given for binary fields.

$$s = \frac{3 \cdot x_a^2 + a}{2 \cdot y_a}$$
$$x_c = s^2 - 2 \cdot x_a$$
$$y_c = y^2 + s \cdot (x_c - x_a)$$

Theorem 6. *Given an elliptic curve E over finite field K, the tuple $(E(K), +)$ forms a group.*

The majority of the group axioms are fairly straightforward to show for an elliptic curve group. The only one that is slightly tricky is associativity. This will be worked through, with suitable hints, in a worksheet.

Since the goal of cryptography is to be used, a choice of curve that makes computation as efficient as possible is desirable. One class of curves which do this are Barreto-Naehrig curves.

Definition 16 (Barreto-Naehrig Curves). *A Barreto-Naehrig curve is an elliptic curve where $a = 0$. That is, it is a curve of the from:*

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

for $b \in K$.

The BN curve is defined over $\mathbb{F}_p$ and the group has order n. The primes p and n are given by:

$$p = p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$$
$$n = n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$$

for some $u \in \mathbb{Z}$.

This means that the entire curve can be parameterised by b and u. The curve is also suitable for efficient computation.

5 Key Exchange

> "Where there is a Key, there is yet hope."
> – Catherynne M. Valente, The Girl Who Circumnavigated Fairyland in a Ship of Her Own Making

Symmetric schemes all assume that both parties magically share a key. Public key schemes get around this by having an encryption key that can be published online. This, however, only moves the problem to that of verifying you are downloading the correct person's keys. Also, in practice symmetric schemes tend to be a lot faster than asymmetric schemes and thus more desirable for sending large amounts of data. Thus it would be desirable to have a method for two

users, who share no secret information, to have a way to establish a shared key without anyone else being able to learn it. This area of cryptography is referred to as key exchange.

The scheme that we are going to discuss is the one by Diffie and Hellman – referred to as Diffie-Hellman Key Exchange. The paper by Diffie and Hellman which proposed the key exchange mechanism kickstarted the area of public key cryptography. As with ElGamal, the scheme's security will depend on the hardness of the discrete logarithm problem.

$$
\begin{array}{rcl}
\mathcal{A}(\mathbb{G}, g, q): & & \mathcal{B}(\mathbb{G}, g, q): \\
a \xleftarrow{\$} \mathbb{Z}_q & & \\
A \leftarrow g^a & & \\
 & & b \xleftarrow{\$} \mathbb{Z}_q \\
 & & B \leftarrow g^b \\
 & \xrightarrow{A} & \\
 & \xleftarrow{B} & \\
K_A \leftarrow B^a & & \\
 & & K_B \leftarrow A^b \\
\textbf{return } K_A & & \textbf{return } K_b
\end{array}
$$

Correctness can be shown as follows:

$$
\begin{aligned}
K_A &= B^a \\
&= (g^b)^a \\
&= g^{a \cdot b} \\
&= (g^a)^b \\
&= A^b \\
&= K_b
\end{aligned}
$$

The scheme described above can be shown to have one of two security properties depending on which assumption is used. If the Computational Diffie-Hellman assumption is used, then the adversary cannot learn the shared secret key. If the Decisional Diffie-Hellman assumption is made then the adversary cannot distinguish the real key from a random group element.

6 Secret Sharing and Multi-Party Computation

"Once exposed, a secret loses all its power."
– Ann Aguirre, Grimspace

In this section we introduce a more advanced area of cryptography that does not fall into the confidentiality, integrity, authenticity framework discussed above. Consider the following:

Eleven scientists want to have a cabinet built where they will keep some top secret work. They want multiple locks installed, with keys distributed in such a way that if any six scientists are present then they can open all the locks, but if only five are present then they cannot open all the locks.

Exercise 11. What is the minimum number of locks required to meet all the criteria.

As can be imagined, this requires each scientist to carry a lot of keys with them. The goal of this section is to design some cryptography such that each scientist would only have to carry a single access card.

6.1 Secret Sharing

In this section we introduce secret sharing, which can be seen as a mathematical generalisation of the above problem. For what follows we will assume that our "secret" information belongs to the finite field $\mathbb{F}$.

Definition 17 (Secret Sharing). *Given a secret value $s \in \mathbb{F}$ and n people $P_1, \ldots, P_n$, give each person some information s_i such that, with all n people, they can learn s but for fewer than n people they learn no more information than if they had none of the s_i.*

The solution we present here was designed by Adi Shamir[5] and relies on the properties of polynomials over finite fields. Before we give the scheme we present some of the required properties of polynomials.

Theorem 7. *A polynomial of degree n can have at most n roots.*

Theorem 8. *Two non-zero polynomials f and g of degree n can agree on at most n points.*

Proof. Assume that f and g agree on more than n points. Then the polynomial $f - g$ is a degree n polynomial with more than n roots. This is a contradiction to Theorem 7 and thus the result follows.

Theorem 9. *Given points (x_i, y_i) for $1 \leq i \leq n$ there exists a unique degree $n-1$ polynomial f such that $f(x_i) = y_i$ for all $1 \leq i \leq n$.*

Proof. Given the pairs (x_i, y_i) for $1 \leq i \leq n$, define the n polynomials δ_i for $1 \leq i \leq n$ as:

$$\delta_i(X) = \frac{\prod_{j=1, j\neq i}^{n}(X - x_j)}{\prod_{j=1, j\neq i}^{n}(x_i - x_j)}$$

[5] The "S" of "RSA".

It is easy to see that $\delta_i(x_j) = 0$ for $j \neq i$ and $\delta_i(x_i) = 1$. Let

$$f(X) = \sum_{i=1}^{n} y_i \cdot \delta_i(X)$$

We have that f is a degree $n-1$ polynomial such that $f(x_i) = y_i$ as required. Uniqueness follows from Theorem 8.

We are now in a position to give the scheme. Given a secret s, a field $\mathbb{F}$, and n people to share the secret between, the scheme works as follows:

Sharing: Given $s \in \mathbb{F}$, choose $a_1, \ldots, a_{n-1}$ uniformly from $\mathbb{F}$ and construct the polynomial $p(X) = s + \sum_{i=1}^{n-1} a_i \cdot X^i$. The party P_i is then given the share $s_i = (i, p(i))$.

Recombination: Given shares $s_1, \ldots, s_n$ the user performs polynomial interpolation to learn a degree $n-1$ polynomial p' and outputs $p'(0)$.

Theorem 10 (Shamir's Secret Sharing Correctness). *Given the two polynomials p, p' as produced from Shamir's Secret Sharing scheme we have that $p' = p$. Thus the scheme is correct because $p(0) = s$.*

Proof. This follows directly from Theorem 9.

Exercise 12. Show that with $n-1$ of the shares, no information can be learnt about the secret s.

Exercise 13. Explain why you don't want to do Shamir's secret sharing over the integers.

Definition 18 (n Choose m Secret Sharing). *Given a secret value $s \in \mathbb{F}$ and n people $P_1, \ldots, P_n$, give each person some information s_i such that with any m people they can learn s but for fewer than m people they learn no more information than if they had none of the s_i.*

Exercise 14. Show that the solution to the original problem trivially extends to the "n choose m" version.

6.2 Multi-Party Computation

In this section we extend the secret sharing scheme defined above to handle addition and multiplication, without revealing any information about the underlying secret. Since any computation can be written out as a binary circuit, this method can perform any computation by defining secret sharing on the binary inputs. More formally, an and and xor form a universal set of gates. This means that any other logic gate can be produced from a combination of these two gates. Since the and gate is multiplication over $\mathbb{Z}_2$ and xor is addition over $\mathbb{Z}_2$, it is possible to use these methods to perform any computation.

Theorem 11. *Given two secret values s, t shared with polynomials f_s, f_t respectively, the n users $1 \leq i \leq n$ with shares $(i, f_s(i)), (i, f_t(i))$ can calculate their share of $s + t$ as $(i, f_s(i) + f_t(i))$.*

Proof. We have that party P_i has shares $(i, f_s(i))$ and $(i, f_t(i))$ of s and t respectively. To show that $(i, f_s(i) + f_t(i))$ is a share of $s + t$ we need to show that polynomial interpolation behaves as expected. Recall we have:

$$\delta_i(X) = \frac{\prod_{j=1, j\neq i}^{n}(X - x_j)}{\prod_{j=1, j\neq i}^{n}(x_i - x_j)}$$

which does not change in our final scheme because the x coordinates given to the parties do not change. The final polynomial will be constructed from the new y coordinates, giving:

$$p(X) = \sum_{i=1}^{n}(f_s(i) + f_t(i)) \cdot \delta_i(X)$$

Evaluating this at 0 we have:

$$\begin{aligned} p(0) &= \sum_{i=1}^{n}(f_s(i) + f_t(i)) \cdot \delta_i(0) \\ &= \sum_{i=1}^{n} f_s(i) \cdot \delta_i(0) + \sum_{i=1}^{n} f_t(i) \cdot \delta_i(0) \\ &= s + t \end{aligned}$$

Hence $p(x) = f_{s+t}(x)$ as required.

We could think about employing a similar process for multiplication. However, while the resulting polynomial does represent the secret, the method introduces two issues. Firstly, the polynomial will no longer be a truly random polynomial as it is now the product of two other polynomials. Secondly, the polynomial will have degree $2 \cdot n$ and the secret cannot be reconstructed by n people. Thus we must produce an algorithm which produces a polynomial of the correct form.

To resolve these issues, each player P_i chooses a random polynomial h_i of degree n such that $h_i(0) = f_s(i) \cdot f_t(i)$. They then send $h_i(j)$ to each other player P_j. Each player P_j computes:

$$H(j) = \sum_{i=1}^{n} \lambda_i \cdot h_i(j) \tag{1}$$

where $\lambda_i = \prod_{j=1, j\neq i}^{n} \frac{j}{j-i}$ as determined by Lagrange interpolation.[6]

[6] Assume the parties are numbered $1, \ldots, n$.

Theorem 12. *Given two secret values s, t shared with polynomials f_s, f_t respectively, the n users $1 \leq i \leq n$ with shares $(i, f_s(i)), (i, f_t(i))$ can calculate their share of $s \cdot t$ as $(i, H(i))$ as defined in Eqn (1).*

Proof. Given the shares $y_i = H(i)$, the recombination algorithm for Shamir's secret sharing is run giving:

$$\begin{aligned} p(X) &= \sum_{j=1}^{n} y_j \delta_j(X) \\ &= \sum_{j=1}^{n} \sum_{i=1}^{n} \lambda_i h_i(j) \delta_j(X) \end{aligned}$$

From this we have;

$$\begin{aligned} p(0) &= \sum_{j=1}^{n} \sum_{i=1}^{n} \lambda_i h_i(j) \delta_j(0) \\ &= \sum_{j=1}^{n} \sum_{i=1}^{n} \lambda_i \lambda_j h_i(j) \\ &= \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{n} \lambda_j h_i(j) \\ &= \sum_{i=1}^{n} \lambda_i h_i(0) \\ &= \sum_{i=1}^{n} \lambda_i f_s(i) f_t(i) \\ &= f_{st}(0) \\ &= s \cdot t \end{aligned}$$