Cryptography Handout

Maths and Physics Club

December 2020

1 Introduction

While cryptography originated from protecting the secrecy or the confidentiality of messages, it has a multitude of applications today from integrity protection and digital transactions to bitcoins, or providing a seed or a key for random access to denote if a system is available. In addition to learning the cryptographic techniques and their purposes, I also want to digress to talk about the history and the progression of cryptography as a field. We could also discuss about the relevant questions raised during the cryptographic developments (which were significant enough to affect the development of the algorithm and the cryptographic landscape). This facilitates to understand the logic behind the cryptographic scheme design. In other words, while applying or using a cryptographic algorithm may be possible while treating the algorithm as a black box and only knowing the input-output relationships, to round things off, you will have to practice the logic of the cryptographic design through the notebooks that were given and get a better sense of what is happening within the black box and understanding how and why it was designed that way.

Maybe it is nice to start of by saying that using an algorithm which is overly complex may not always thwart it from being hacked. Research and development in cryptography is often driven by simplicity. Advanced mathematics is used for simpler descriptions of the cryptographic algorithms. Yet, the designs of algorithms themselves are often public and the implementations are efficient for computer processing and is downright easy, if the key is known.

2 What Is Cryptography?

It is the study of digital coding to ensure legitimate access of data. And this coding is quite synonymous with computer programming, but that wasn't how it all began. The topic of focus here will be to learn how to ensure the confidentiality of a message against unknown adversaries.

To start things off, here is a description of the very popular framework used in cryptography. Introducing Alice, Bob and Eve. Continuing with the cryptography jargon - Alice has the data or the message and wants to share it with Bob, who is authorised to access the data. Alice transmits the message to Bob via a medium or a communication channel - something like wireless transmission or a cable connection. Security only becomes relevant when there's an external entity, or an adversary, or an attacker. Eve represents this attacker and somehow Eve has access to this communication channel. That can be a non-trivial task in itself, but we won't get into those nitty-gritty details. We'll just assume Eve already has access to the channel, kind of the worst-case scenario of part 1.

Your internet service provider when you are trying to communicate with some remote server in Iceland is a potential attacker.

Some more terminology before we start:

- **Plaintext** or p is the original message
- **Ciphertext** or c is the coded message that gets sent through the communication channel
- A **cipher** is the algorithm that is used to convert from plaintext to ciphertext
- A **key** is a piece of information that is only known to Alice and Bob and not to Eve and this is typically used in the cipher transformation
- The whole process of converting plaintext to ciphertext is called **encryption** and the reverse process is called **decryption**

Another point of note:

- **Cryptography** is the study of encryption and decryption techniques, or to secure information

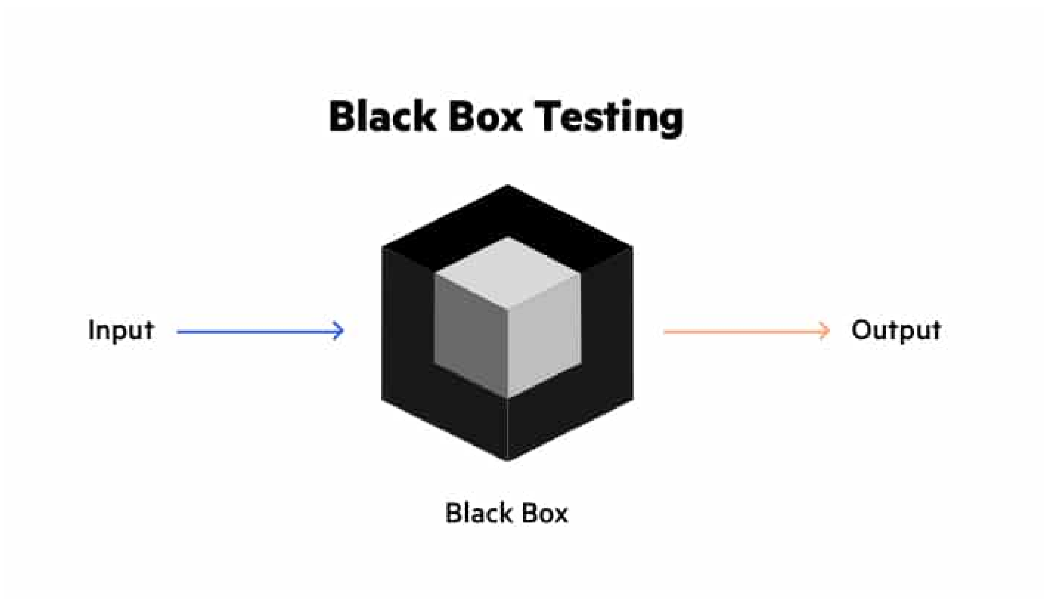

Figure 1: How a Black Box works

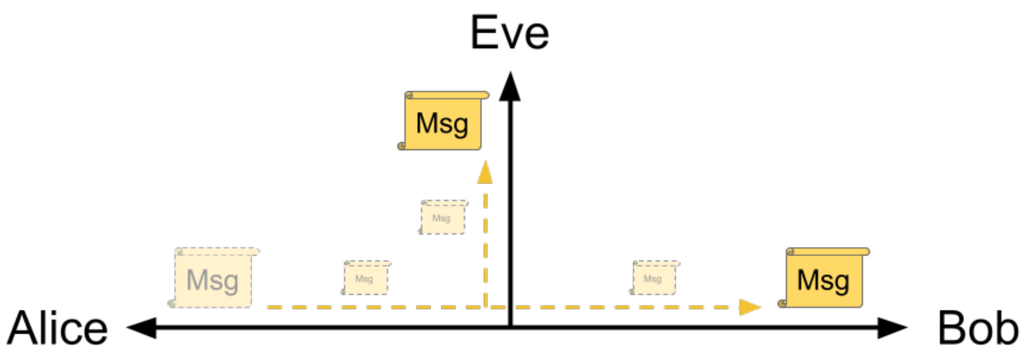

Figure 2: The Alice-Bob-Eve Framework

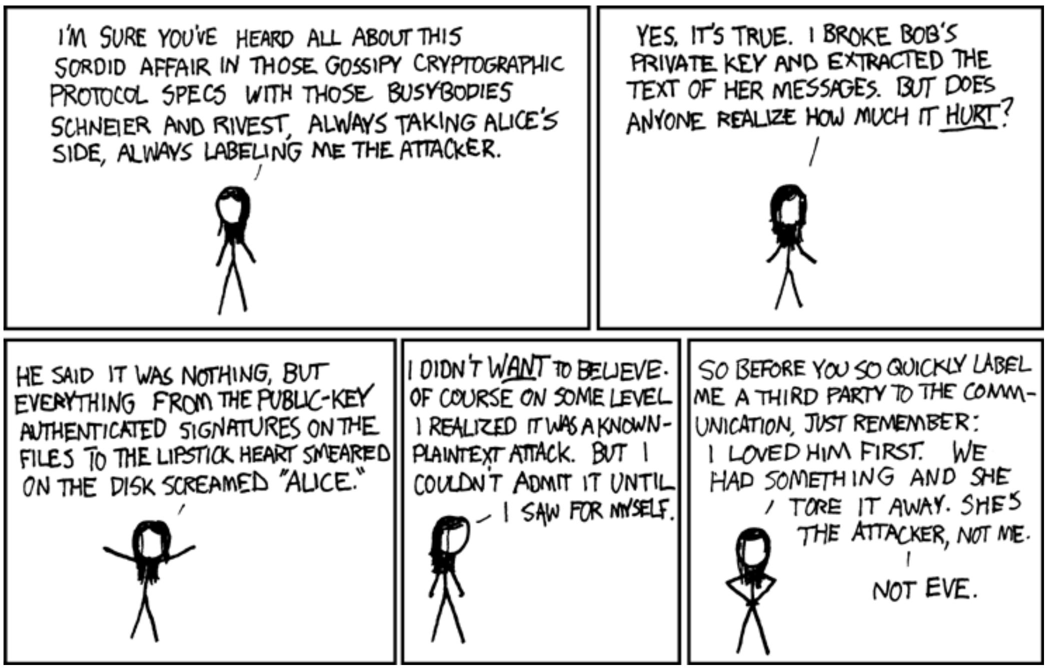
I'M SURE YOU'VE HEARD ALL ABOUT THIS SORDID AFFAIR IN THOSE GOSSIPY CRYPTOGRAPHIC PROTOCOL SPECS WITH THOSE BUSYBODIES SCHNEIER AND RIVEST, ALWAYS TAKING ALICE'S SIDE, ALWAYS LABELING ME THE ATTACKER.
YES, IT'S TRUE. I BROKE BOB'S PRIVATE KEY AND EXTRACTED THE TEXT OF HER MESSAGES. BUT DOES ANYONE REALIZE HOW MUCH IT HURT?
HE SAID IT WAS NOTHING, BUT EVERYTHING FROM THE PUBLIC-KEY AUTHENTICATED SIGNATURES ON THE FILES TO THE LIPSTICK HEART SMEARED ON THE DISK SCREAMED "ALICE."
I DIDN'T WANT TO BELIEVE. OF COURSE ON SOME LEVEL I REALIZED IT WAS A KNOWN-PLAINTEXT ATTACK. BUT I COULDN'T ADMIT IT UNTIL I SAW FOR MYSELF.
SO BEFORE YOU SO QUICKLY LABEL ME A THIRD PARTY TO THE COMMUNICATION, JUST REMEMBER: I LOVED HIM FIRST. WE HAD SOMETHING AND SHE TORE IT AWAY. SHE'S THE ATTACKER, NOT ME.
NOT EVE.

Figure 3: How can you not like xkcd?

- **Cryptanalysis** is the study of codebreaking and deciphering text without the key, or to break the security
- **Cryptology** is the field that embodies both cryptography and cryptology
  It goes without saying, that advancements in either field go hand-in-hand. In general use though, cryptography is used synonymously with cryptology

3 Attacker's Perspective

What do you see from the attacker's perspective? Do they know the same amount of information or is there some kind of an information asymmetry, if I may?

Generally there is an unbalance as to how much Alice and Bob knows versus what Eve knows. If the attacker knows as much as Alice and Bob, then the decryption becomes equivalent to Bob and Eve, and the attacker has all the capabilities for correctly decrypting and breaking the cryptographic protection. Knowing an equivalent amount of information about the system, about the algorithms and how they work, except for one specific aspect, is called Kerckhoff's Principle or Open Design or Shannon's Maxim.

Cryptography requires information asymmetry - there's a part of what Alice and Bob knows that is designed to be secret against the attacker. In cryptography, this is taken care by the key - which is used to map the plaintext into different letters or characters so that the ciphertext looks like gibberish.

In layman's terms, Kerckhoff's principle states that the attacker knows the system except for the key. So, the only disadvantage of the attacker is not knowing the key, which is intended to be a secret in the first place.

In contrast to Kirckhoff's principle, an approach called *security by obscurity* relies on the attacker not knowing the algorithm protocol or the system execution. Some proprietary protocols do not make the descriptions publicly available or explicitly state its details as the security of the system relies on such obscurity and the system is no longer secure once the attacker learns about the how-to of the system. There's a fancy term for this kind of cryptography called *steganography*.

Ever used `invisible ink` *which reveals itself under UV light, that's an example of steganography*

Give this a shot. "Since Everyone Can Read, Encoding Text In Neutral Sentences Is Doubtfully Effective". There is a hidden message. So take a moment and think about it. You can find the answer to this and subsequent brainteasers here

An even harder example or problem or whatever you choose to call it would be the following:
Dear George,
Greetings to all at Oxford. Many thanks for your
letter and for the summer examination package.
All entry forms and fees forms should be ready
for final despatch to the Syndicate by Friday
20th or at the very latest, I'm told, by the 21st.

Admin has improved here, though there's room
for improvement still; just give us all two or three
more years and we'll really show you! Please
don't let these wretched 16+ proposals destroy
your basis O and A pattern. Certainly this
sort of change, if implemented immediately,
would bring chaos.
Sincerely yours.

Security by obscurity is useful to deter casual attackers, but is insufficient to address more persistent attackers and history has shown that to be the case. Moving forward (and during the live session later), the Kerckhoff's principle is assumed to hold effect - The scope of secrecy is clearly defined and everything else shall be known to the attacker, which in the worst-case, everything that can be known, shall be known

4 Information Entropy

These slides on the topic of information theory will be an interesting digression and fields such as these contribute to cryptology.

For our purposes, information entropy is used to quantify the strength of the key or the secret information that access the inputs to drive the cryptosystem. The information asymmetry induces a difference in *computation time*. With the key, the decryption becomes ***deterministic*** which corresponds to zero entropy. But without the key, the decryption is ***random***. In such systems, information entropy can be used to quantify the difficulty for the attacker to derive the key.

A deterministic process is a process where the outcome is known with certainty, in other words, with a probability of 1. A random process is basically a non-deterministic process and the outcome is not known with certainty, two or more outcomes with a non-zero probability. A more involved statement to make will be, to define a deterministic process, as an outcome x that can be defined by the scalar function $f(x)$. But to determine the image of a random function at x, x scalar values are required. The two processes carry different amounts of information as x grows, while the output of a deterministic process stays constant, the random output grows linearly with x (this will be a bit more clear in the attached slides).

Another point of note is that the attacker doesn't know the value of the key and the key looks random to the attacker. The more random it looks, the higher the security strength (this is pretty intuitive, isn't it? The unpredictability is the greatest when there's no bias in the probabilities).

Back with a brainteaser - *there is a typical dice and in each event, you throw the dice and observe the number. What is the information entropy for five events in bits?*

What is the information entropy for the weather in Mumbai, if the probability that it rains is 1?

Here is an extract about randomness and the idea of leaving an algorithm to chance from the book *Algorithms To Live By* by *Brian Christian*

> *"I must admit that after many years of work in this area, the efficacy of randomness for so many algorithmic problems is absolutely mysterious to me. It is efficient, it works; but why and how is absolutely mysterious"*
> — Michael Rabin

Randomness seems like the opposite of reason — a form of giving up on a problem, a last resort. Far from it. The surprising and increasingly important role of randomness in computer science shows us that making use of chance can be a deliberate and effective part of approaching the hardest sets of problems. In fact, there are times when nothing else will do.

In contrast to the standard "deterministic" algorithms we typically imagine computers using, where one step follows from another in exactly the same way every time, a randomized algorithm uses randomly generated numbers to solve a problem. Recent work in computer science has shown that there are cases where randomized algorithms can produce good approximate answers to diffuclt question daster than all known deterministic algorithms. And while they do not always guarantee the optimal solutions, randomized algorithms can get surprisingly close to them in a fraction of the time, just be strategically flipping a few coins while their deterministic cousing sweat it out.

There is a deep message in the fact that on certain problems, randomized approaches can outperform even the best deterministic ones, Sometimes the best solution to a problem is to turn to chance rather than trying to fully reason out an answer.

5 The Attacker's Perspective Revisited

What are the ways the attacker can plan their attack? A simple way for an attacker to attack a crypto system and learn the key or the plain text is to try **all** possible keys. Because the key has finitely many options, the attacker would eventually exhaust the key options and will try the correct key that is being used by the legitimate parties. It has a simple basis - to try all the key options in a random order until they find the correct key. This method has crept its way into day-to-day use and is known as the **Brute Force Attack**.

The one requirement for this attack to work, is that the attacker should be capable of knowing the right key when they tried that key. In other words, they are capable of distinguishing between the correct key and an incorrect key.

This is generally the last resort for the attacker. But, if the attacker knows which key options are more likely than other options and uses such information for learning the key, then it is no longer considered brute forcing.

Information entropy quantifies the difficulty of the brute force attack in the attacker's perspective. As the attacker progresses through the brute force attack and learns more about the system, the entropy decreases - the entropy is *dynamically* updated.

When a key is n bits long, how many keys would the attacker need to try on average?

Moving on. A non-uniform distribution or a bias in choosing the key yields a reduced entropy in the cryptosystem. And the known information is exploited to crack the ciphertext in a process known as **cryptanalysis**

Many instances of entropy reduction can be observed in day-to-day life, and one such instance in a non-cryptographic context comes in gambling. In a game of *blackjack*, *card counting*, i.e., using the fact that the card distribution of the remaining deck is uneven (after a few initial rounds), can be used to one's advantage. The slight advantage in information is used to increase the bet size when the remaining deck is favourable to the player and decreasing the bet when it's unfavourable. While a casino has the edge without card counting, much like the other casino games a house offers, card counting shifts the edge to the player. When Edward Thorpe, a mathematician and a basically a *blackjack research*, first exploited this simple technique, the only thing a casino could do to thwart him was to bar him from gambling houses.

When a ciphertext provides no information about the plaintext without the key, and an attacker cannot learn the plain text regardless of how much time or effort they put in, then the cryptosystem is said to achieve **perfect secrecy** and is cryptanalytically unbreakable. In reality, this is achievable by using a key that is as long as the message, or in other words, the key entropy is as great as the message entropy. In addition, the key should never be reused in the encryption, i.e., it should be a one-time pad (and this is not to be confused with a one-time password). Although perfect secrecy is the strongest notion for encryption, or *information-theoretic security*, it is hard to implement at a large scale due to its core principle that the key should be as long as the message (and it is obvious that as the message grows, the key grows as long).

Here is a repurposed version of Claude Shannon's originally classified paper on Perfect Secrecy

6 Computational Security

Due to the inherent complexity of a perfectly secret system, the more common practice employed is to build a cryptosystem that is *computationally secure* - one that can be brute forced through, just that it would take *a few* multiples of the age of the Universe to crack using classical computers.

Computational security lies on two intuitive assumptions:

1. The attackers are actually computationally limited (and this is applicable to all real world systems)
2. The system relies on mathematical problems that are assumed to be difficult to solve, they needn't be proven to be difficult, just **cough** widely believed **cough** to be practically infeasible

The goal of this module in the workshop is to introduce you to the design of a cryptosystem, hence, you will find limited information about the implementation of such a system. The avenues are endless and I highly encourage you to read them.

A brute force attack at a speed of $x = 1$ *quintillion* $= 10^{18}$ (1 quintillion decryptions per second)

Key Size	**# of Decryptions**	**Expected Time (sec)**	**Expected Time (years)**
n	2^{n-1}	$\frac{2^{n-1}}{x}$	$\frac{2^{n-1}}{x*60*60*24*365}$
56 bit DES	3.6×10^{16}	3.6×10^{-2}	1.14×10^{-9}
128 bit AES	1.7×10^{38}	1.7×10^{20}	5.4×10^{12}
168 bit Triple DES	1.9×10^{50}	1.9×10^{32}	5.9×10^{24}

Table 1: Massively Underestimated Computation Times for a Brute Force Attack

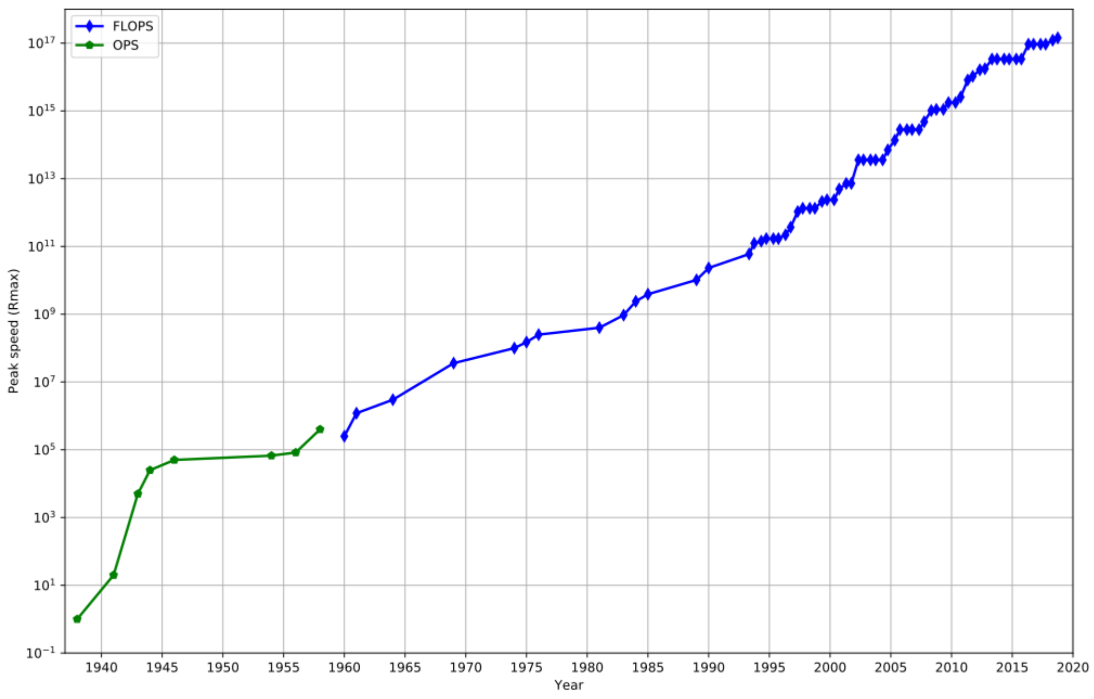

Figure 4: Growth of Computation Speeds of Supercomputers over the years (in a logarithmic scale)

Side-Channel Attacks

For the sake of roundedness, I will introduce one form of **implementation-based defect** in cryptosystems - side-channel attacks

Different inputs of keys and plaintext produce different hardware loads in system computations (if you are familiar with the hardware behind Arithmetic Logical Units, you know the drill). To capture the computational asymmetric between different inputs, a well-known present age cryptographer named Paul Kocher, specifically used the *timing information*, how long it takes to compute and the *power signals*, how much power it consumes. Having more bits of 1's in the input of the cipher can increase both the computation time and the power consumption.

If these things fascinate, check out the following article on Physical Key Extraction Attacks on PCs.

On a different note, here is a video on how to hack your own car by Steve Mould

7 XOR Cipher

Any workshop on cryptography will be incomplete without a mention about this - the XOR Cipher. The XOR operation is defined as

$$p \oplus q = (\neg p \wedge q) \vee (p \wedge \neg q)$$

For our purposes, we just need to know the properties of an XOR function

$$A \oplus 0 = A$$

$$A \oplus A = 0$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

$$(B \oplus A) \oplus A = B \oplus 0 = B$$

The basis is simple and can be obtained from the last property listed. For a message m and key k, the ciphertext c can be

$$c = m \oplus k$$

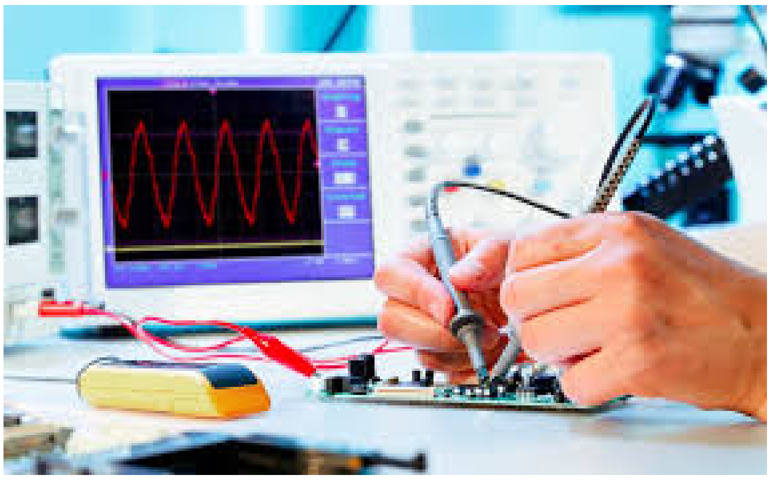

Figure 5: Attempting a side-channel attack

And the message or plaintext can be subsequently obtained by

$$c \oplus k = m \oplus k \oplus k = m \oplus 0 \equiv m$$

Decrypt the message from the ciphertext `011011` *using the key* `010101`

8 Caesar Cipher

This was one of the first well-known ciphers. It is based on a simple substitution and was used by Julius Caesar in around 58 BC. During the war, Caesar shifted each letter in his military commands in order to make them appear meaningless, should the enemy intercept it. Imagine, Alice and Bob decided in advance to communicate using the Caesar cipher with a shift of 3. First, Alice writes her message in plain English and then she applies a shift of three to each letter. So A becomes D, B become E,...

```
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
D 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
```

This unreadable or encrypted message is then sent to Bob openly. And as they have shared the key in advance, Bob simply subtracts the shift of three from each letter in order to read the original message. This basic cipher was in fact used by military leaders for hundreds of years after Caesar.

How do you break this if you didn't know the key?

It was published 800 years later by an Arab mathematician named Al-Kindi. He broke Caesar's cipher by using a clue based on an important property of the language, a message is written in. If you scan text from any book and count the frequency of each letter, you will find a fairly consistent pattern.

These are the letter frequencies of English. This can be thought of as the fingerprint of English. We leave this fingerprint when we communicate without realising it and it is one of the most valuable clues to a codebreaker. To break this Caesar cipher, count up the frequencies of each letter in the encrypted text and check how far the fingerprint has shifted. In the above shift of 3 (or key value = +3), the the letter H will become the most popular letter instead of E and reversing it will reveal the message.

This video by VSauce on *The Zipf Mystery* might be of interest if you are into linguistics.

To make a lighter fingerprint, is to flatten the distribution of letter frequencies. And by the mid 15th century, this is exactly what was done through *polyalphabetic ciphers*. Imagine Alice and Bob share a secret *shift word*, say ABCD. To make things simple, you can convert the word into numbers according to the letter position in the alphabet (which. in this case, will be 1234). Next, this sequence of numbers (or the shift word is repeated along the message). And finally, each letter in the message is encrypted by shifting according to the number below it.

Plaintext:	`T H I S  T H E  M E S S A G E`
Key:	`A B C D  A B C  D A B C D A B`
Ciphertext:	`U J L W  U J H  Q F U V E H G`

And the encrypted message is sent openly to Bob. Bob decrypts the message by subtracting the shift according to the key. By the way, this method of encrypting is called the ***Vigener Cipher*** and was used in the 16th century.

The method of frequency analysis shown above won't produce a distinct distribution, i.e., the fingerprint will appear lighter. But... there's a partial fingerprint.

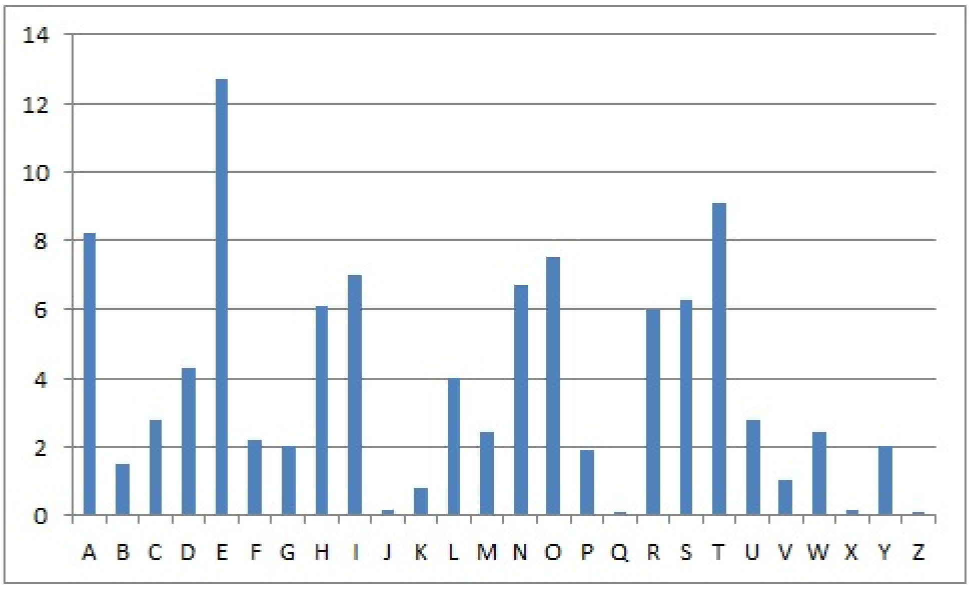

Figure 6: *Frequency Analysis* of the English Alphabets

There is a leak of information due to the repetition in the encrypted message - the repetition of the key. A first step towards breaking this encryption is for Eve to determine the length of the shift word used, not necessarily the word itself. When she checks the distribution of every 4^{th} letter, the fingerprint will reveal itself and she already knows how to break it when there's such a distinct fingerprint - 4 Caesar ciphers in a repeating sequence. Individually, it is a trivial task. Then, how can Alice design a cipher that hides her fingerprint and thus stop the leak of information? By using randomness. And by using a list of shifts as long as the message itself (to avoid any repetition), there will be no leak of information and the message is said to have achieved Perfect Secrecy - the theoretical limit. And the encrypted message, will have a uniform frequency distribution.

Perfect secrecy is the strongest possible notion of encryption and the idea emerged towards the end of the 19th century

But in this case where, one key is required to both encrypt and decrypt the same lock, the key must be shared before hand and thus can become impractical on larger scales

But how do you generate randomness? You could use a coin flip, or a dice, or a random number generator. Check out our post on random number generators on our social media pages.

9 One-Way Functions

A one-way function is a function which is *easy* to compute along one direction but whose inverse is *hard* to compute. Here, *easy* and *hard* are to be understood in the sense of computational complexity theory, specifically the theory of polynomial time problems and it might prove to be too much of a digression at this point. In remains to be proven if one-way functions actually exist. These functions are essential for the existence of computationally secure methods of encryption and form the basis of cryptography in the era of *classical computers*

Although, I just mentioned that the inverse is *hard* to compute, creating a backdoor or, in this context, commonly known as the **trapdoor**, can make it drastically easier to calculate the inverse as well. You might have heard that the advent of Quantum Computers might prove fatal for current encryption methods (this will be further discussed in the Notebooks), especially the RSA Algorithm. This is because the RSA Algorithm depends on two key principles:

1. Multiplying two large primes is easy

2. Prime factorization is difficult

But knowing the prime factors before hand, creates the trapdoor for the rest of the transmission to work effortlessly and securely without the eavesdropper interpreting the information.

Bitcoins, email encryptions and so on, there are papers written about these things so it's not like these are algorithms are classified secrets. But the point is that, even if someone knows exactly how these work, they should have a really difficult time trying to go backwards without some extra information like a secret number or a secret key.

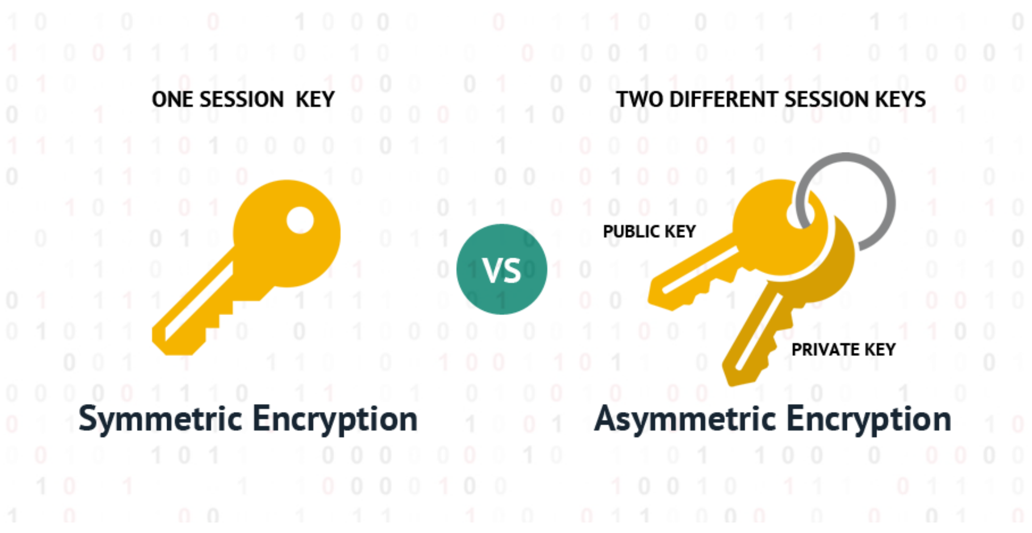

Figure 7: The different types of keys

10 A Word on Keys

10.1 The two families of Cryptographic Systems

Symmetric keys

Until the 1970s, cryptography had been based on symmetric keys, i.e., the sender encrypts their message using a specific key and the receiver decrypts using an identical key. Remember that encryption is a mapping from some message using a specific key to a ciphertext message. Here, to decrypt a ciphertext, you use the same key to reverse the *mapping.* So for Alice and Bob to communicate securely, they must first share identical keys. However, establishing a shared key is often impossible if Alice and Bob can't physically meet. Plus, if Alice needs to communciate with multiple people, perhaps she's a bank, then she is going to have to exchange distinct keys with each person. Now, she will have to manage all of these keys, and send thousands of messages just to establish them. Is there a simpler way?

Asymmetric keys

In 1970, James Ellis, a British engineer and cryptographer was working on an idea for non-secret encryption. It is based on a simple yet clever concept. We can all agree that lock and unlock are inverse operations. Alice could buy a lock, keep the key and send the open lock to Bob. Bob then locks his message and sends it back to Alice. No keys are exchanged. This means, she could publish the lock widely and let anyone in the world use it to send her a message. And, she now only needs to keep track of a single key. Ellis never arrived at a mathematical solution, though he had this intuitive sense of how a no-key-exchange-system should work.

This blog comprehensively covers the differences between the two

10.2 Public and Private Keys

This website extensively covers the method of *end-to-end* encryption so I'll just direct you there

11 Summary

That kind of brings us to the end of this handout. Once again, you can check your answer to those brainteasers at this link. To further test your understanding and expand your horizons on these topics, check out this excellent online interactive website. See you across a LCD display at the live session.

Figure 8: The yin-yang of public and private keys

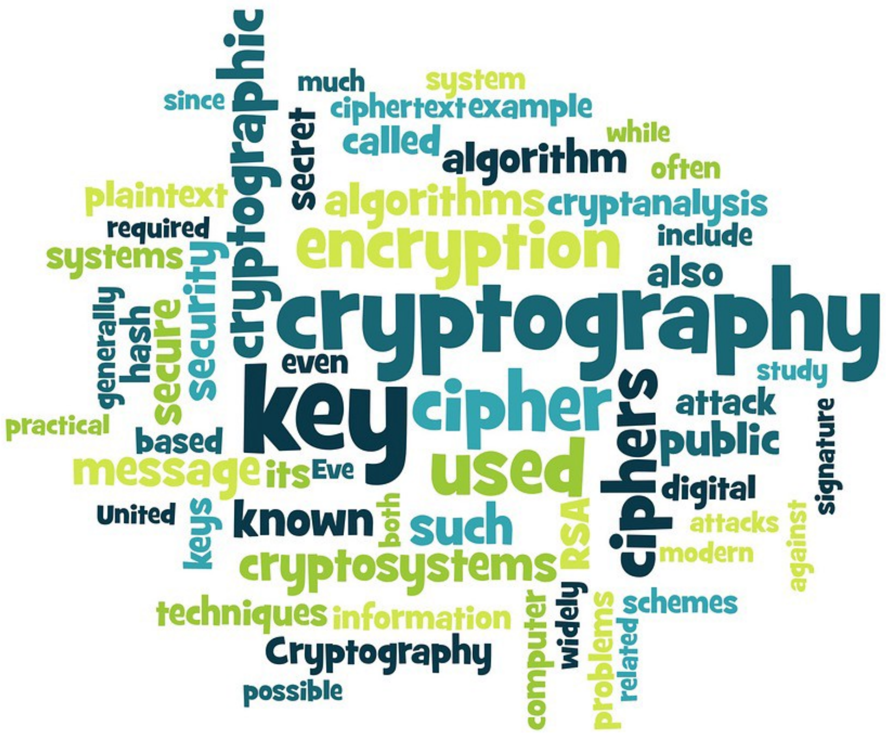

Figure 9: Unwrapping the Words