iut de BORDEAUX

S4.A.01 2023-24

Computer Science Bachelor, A

Information theory

Authors:

GODINAT Caetano, PICAURON Adam, DAUGE Alexis
(Group 21)

Table of Contents

1 Introduction

In today's interconnected world, the abundance of information presents a significant challenge in measuring its transmission capacity. Enter Claude Elwood Shannon, a mathematician whose groundbreaking article revolutionized the field of communication and information. Shannon's Mathematical Theory of Communication introduced a systematic framework for quantifying the flow of information through a given system. In this article, we try to apply Shannon's theory to two different fields information compression and cryptography to see whether the outcomes align with our expectations.

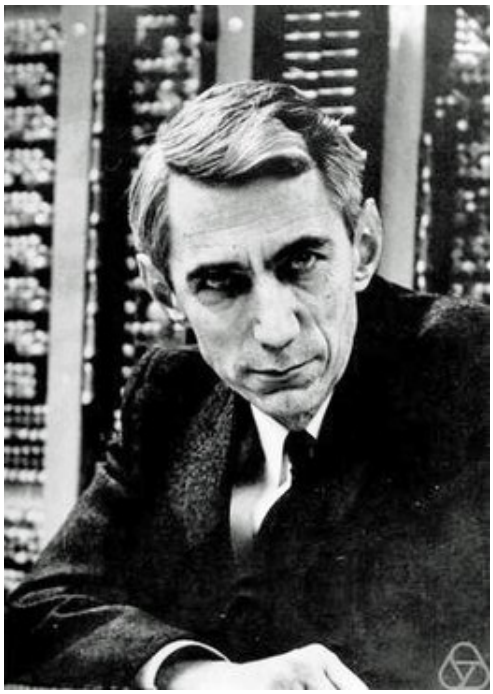


Figure 1: Claude Shannon around 1963 [Wik24]

2 Foundations

Mr Shannon's information theory is based on a number of mathematical principles. Some choices have been made, and we'll explain them below. The founding principles can be summed up roughly as the choice of the logarithm and the use of entropy. To make these principles easier to understand, we have developed solutions in Python, notably by implementing a bot that plays the game Wordle.

2.1 The logarithmic function

In his paper [Sha48], Claude Shannon addresses the fundamental problem of communication, which technically consists of reproducing a message selected at one point from another point. He stresses that the semantic aspects of communication, such as the meaning of messages, are unimportant. What matters is that the actual message comes from a set of possible messages, and the system must be designed to work for every possible selection.

To quantify the information produced when a message is selected from a set, Shannon proposed a measure based on the logarithm. This measure is practical for several reasons. Firstly, it is more practically useful, as it allows important engineering parameters such as time, bandwidth and number of relays to be expressed linearly. Secondly, it better matches our intuition as to the appropriate measurement, as it allows linear comparisons with common standards. Finally, it is mathematically more suitable, as many limit operations are simple in logarithmic terms but would require awkward reformulation in terms of the number of possibilities.

The choice of logarithmic base corresponds to the choice of a unit for measuring information. The use of base 2 leads to units called "bits", while base 10 leads to units called "decimal digits". In certain analytical contexts, base "e" may also be used, giving rise to units called "natural units". Switching from one base to another simply requires multiplication by the logarithm of the ratio between the bases.In the context of Wordle, the words are a sequence of letters that can be converted to bits so we will use the base 2.

An alternative to the logarithm would be the square root. In some cases, this function could be used to give less weight to extreme values.

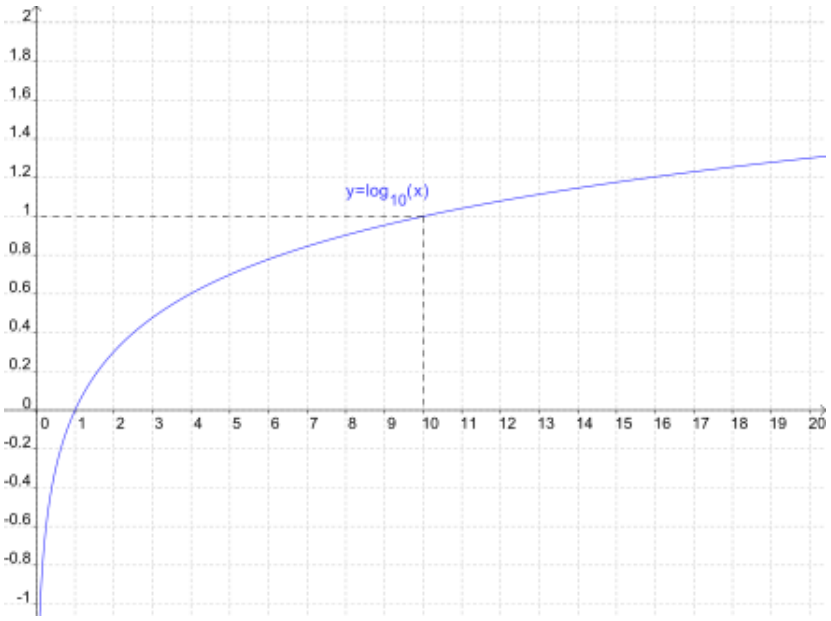


Figure 2: logarithm curve [Wik23]

2.2 Entropy and a "word game" implementation

To explain exactly what entropy is, what better way than to create a bot that has to make choices. Because that's exactly what entropy is, a measure of uncertainty.

Word game or Wordle is a game in which you have to find a random word, and each time you make a move, you receive feedback that lets you know whether for each letter 0: it's in the right place, 1: it's not in the right place, 2: it's not present or no longer present in the word. The words we play are defined in a text document (5-letter English words). We implemented this game in python and created a bot capable of making choices thanks to entropy. In our case, each letter of the word is a symbol. In the context where each symbol is independent, the entropy formula is as follows.

$$-\sum p_i \log(p_i)$$

where pi is the probability of symbol i.

For a word, we'll calculate symbol by symbol, the probability that it will appear (i.e. its frequency at that location) and then apply the formula. Finally, we add up each result, giving the word's entropy.

Entropy varies between 0 and 1, when using base 2. When it comes to comparing them and making a choice, the one that tends most towards 0 is the most likely. For our context: in the list of available words, we calculate and compare the entropy of each word. Then we use the word with the highest entropy to play and get the next feedback (using the maximum entropy allow us to test the widest sample).

This process makes it possible to make the best possible choice when feedback is already available. But what about the first choice? We have several options for several results.

1. Calculate the entropy of each word in the complete list, then select the word with the highest entropy.

2. Put each letter where its frequency is highest.

3. Calculate the "pairs" of symbols most repeated at each location.

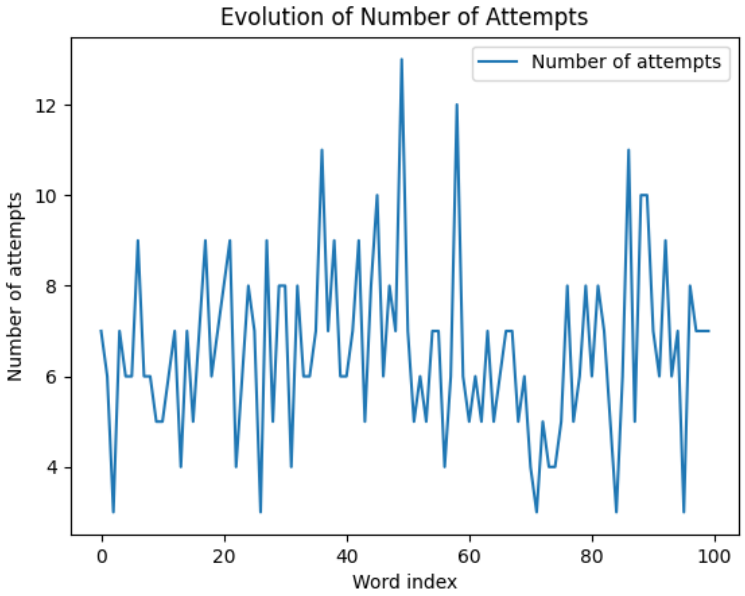


Figure 3: 100 first words, for start word 1 (50% win rate)

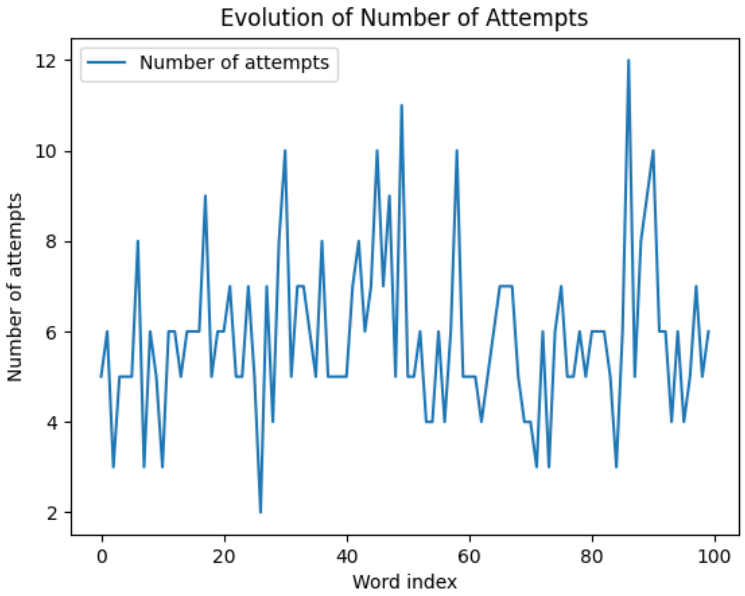


Figure 4: 100 first words, for start word 2 (70% win rate)

It's obvious that the second "first word technique" is the best, giving approximately 70% of win, against 50% for the first. (we didn't implement the third)

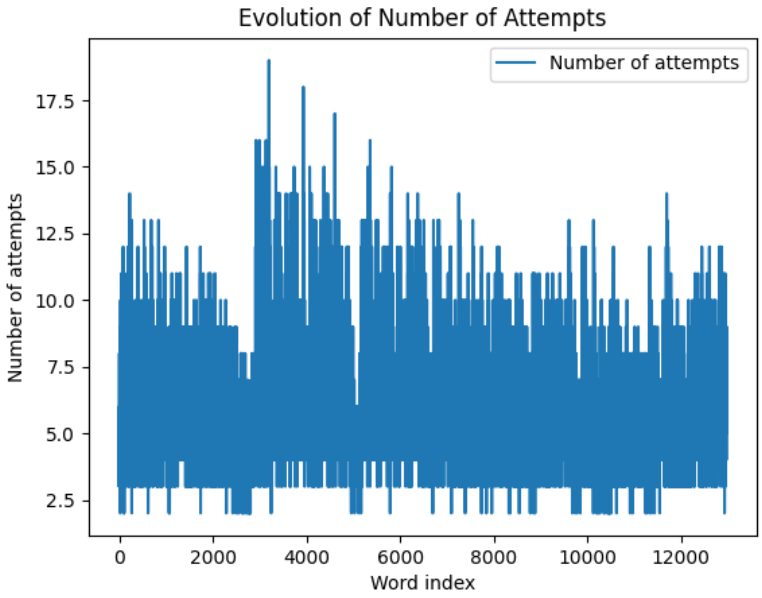


Figure 5: all words in list, for start word 2

Note that these comparisons and calculations are based on the given list of 5-letter words. An interesting approach would have been to have the same statistics but on the whole English language. In this way, the frequencies of symbols and "blocks" of symbols would not be the same.

Once it has the best first word, the bot will sort the list of possible words according to feedback, then play the word with the highest entropy from this list.

2.3 Shannon's theorem

[Sha48]

So what is Shannon's objective? Through logarithmic entropy, Claude Shannon showed us that a certain amount of redundancy was necessary to be sure of getting the right message from the available list. However, thanks to data compression, we can greatly reduce redundancy without any loss of efficiency or accuracy. But the aim behind all this is to maximize the variable C, which represents the maximum transmission capacity for a given time.

2.4 Capacity

The capacity C is defined in the Theorem 1 by $C = logW$, W being the Determinant of the matrix of a communication system defined by:

Theorem 1: Let $b_{ij}^{(s)}$ be the duration of the s^{th} symbol which is allowable in state i and leads to state j. Then the channel capacity C is equal to $logW$ where W is the largest real root of the determinant equation equation: $|\sum_s W - b_{ij}^{(s)} \delta_{ij}| = 0$

Where $\delta_{ij} = 1$ if $i = j$ 0 otherwise.

3 Applications

3.1 Data compression

A "lossless" compression algorithm is defined as any coding procedure whose objective is to represent a certain quantity of information using or occupying a smaller space allowing the exact reconstruction of the original data.

3.1.1 Introduction to Huffman's compression

Invented by David Albert Huffman and published in 1952, Huffman coding is a lossless data compression algorithm. It uses a variable-length code to represent a symbol in the source message. The compressed binary code is determined by estimating the probability of occurrence of the symbols linked to the branches of a binary tree.

There are several implementations of this binary tree: [con]

- Static : Each byte has a predefined code, the tree does not need to be transmitted ;
- Semi-Adaptive : Calculates the occurrence of each byte, the tree needs to be transmitted ;
- Adaptive : Uses a known tree modified dynamically as the stream is compressed according to the symbols previously encountered ;

For the purposes of this report, we will be implementing the semi-adaptive

3.1.2 Principles

age

The string we are going to compress is **"Information theory"**, and its binary code is as follows: **01001001 01101110 01100110 01101111 01110010 01101101 01100001 01110100 01101001 01101111 01101110 00100000 01110100 01101000 01100101 01101111 01110010 01111001**

Needless to say, it's a lot and that's why we're trying to compress it using our algorithm. The principle is as follows: we calculate the occurrence of each letter in the source (the character string), then we sort this list in ascending order of occurrence. This list can be used to build a binary tree in the following "Bottom-Up" way:

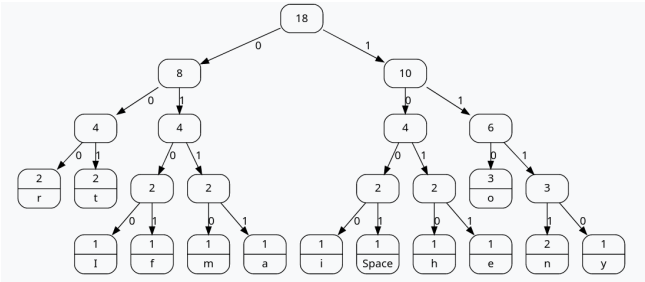


Figure 6: Enter Caption

By traversing this tree from node to node to reach the nodes corresponding to the letters we are looking for, we can find the compressed binary code assigned to them. The nodes on the left have a metric of 0 on their branch and a metric of 1 for the branches on the right (refer to the diagram of the previous tree). This gives the following binary code for the chosen source: **1101100100101110111101100100111011000011000000010111100** ... Much better, isn't it?

In order to decompress our compressed source and retrieve the exact starting data, we need to transmit the tree when communicating the source. (In Python we pass the object created for the tree).

As each node corresponds to a letter and has a weight on the branch linking it to other nodes, we can recover the exact source by simply traversing the tree and obtaining the **"Information theory"** string again.

3.1.3 Limits and Performances

For any source **X** with a Shannon entropy noted **H(X)** has an average length **L** of a codeword obtained by Huffman coding verified the following property:

$H(X) <= L < H(X) + 1.$

The previous relationship shows that Huffman's algorithm approximates the entropy of our source and this may not be very interesting in the case of high entropy, where an overhead of 1 bit becomes significant.

One solution to this problem is to work on blocks of **n** symbols. We then show that we can get closer to the entropy:

$H(X) \leq L < H(X) + \frac{1}{n}$

but the process of estimating probabilities becomes more complex and costly.

3.2 Forward error correcting

In telecommunications and computing, forward error correction (FEC) is a digital signal processing technique used to control errors in data transmission over unreliable or noisy communication channels, as well as to independently increase data reliability at the receiver to ensure high quality.

FEC improves bit error rate efficiency in communication systems, but involves additional digital processing, making it more expensive in terms of cost and spectral efficiency. The balance between these two elements needs to be taken into account in systems.

This is achieved by introducing redundant data using error-correcting codes (ECC) before the message is transmitted. Without redundancy, every piece of data in the message is essential to its understanding. Any error in the message can therefore change its meaning. The aim is to ensure that errors do not compromise the overall understanding of the message, and do not trigger a request for a re-transmission of the message by the receiver, thereby reducing channel traffic by more than a factor of two.

3.2.1 Introduction to error correction code

The problems encountered by modern industry are very diverse, all offering multiple techniques used for error-correcting codes. In some cases, such as data transmission over the Internet, the ECC's role is limited to error detection. In other cases, such as the TCP protocol on the transport layer of the OSI model, when an error is detected during message transmission, it is corrected by retransmitting the message. The techniques listed here are just a few of a sea of others, and today we're going to take a look at a very special technique known for its effectiveness : **Convolutional correctors**

The special feature of a convolutional ECC is that its output depends on the current symbol to be coded, as well as the previous symbol and the result of coding the previous symbol, unlike block codes, which process each block of information independently of the others.

For the purposes of this report, we will confine ourselves to the case of discrete communication channels, which are nothing other than binary channels transmitting only 0 or 1, i.e. bits of information.

3.2.2 Convolutional encoding

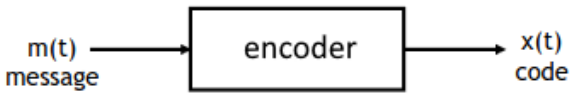


Figure 7: Communication encoder diagram

A convolutional encoder is a finite state machine, and its state diagram can be represented as a lattice (the temporal equivalent of a state diagram). All states are arranged in columns to represent a state at time **t**, where **m(t)** and **x(t)** are discrete signals whose samples are binary. These states are linked to the same sets of states representing time t+1, in accordance with the transition table.

Initial State $s_0[i]s_1[i]$	*Information* $u[i]$	Final State $s_0[i+1]s_1[i+1]$	Exit $v^{(0)}[i]v^{(1)}[i]$
00	0	00	00
00	1	10	11
01	0	00	11
01	1	10	00
10	0	01	10
10	1	11	01
11	0	01	01
11	1	11	10

Figure 8: Transition table with i for the clock time

If the encoder is a linear system, **x(t)** can be generated by a convolution product:

$$x_j = \sum_{i=0}^{M} g_i m_{j-i}$$

In order to carry out a binary convolution coding we will set some parameters:

- $g_i = 0 \rightarrow$ connection absent

- $g_i = 1 \rightarrow$ connection present
- $K \rightarrow$ Constraint length (number of sampling instants during which a message bit participates in the elaboration of the output)

This is a so-called sliding window coding where there are two integers m (for memory) and a (for anticipation) such that the index symbol n of the coded sequence depends only on the finite source block between the indices $n - m$ and $n + a$.

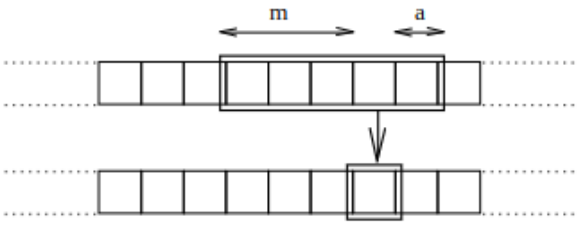


Figure 9: Shift register

For 1/2-rate 2-memory convolutional encoder we take M = 2, then we have K = M + 1 = 3 with an output interleaving n = 2, so using the above formula we obtain the following generator polynomials: $g_1 = [1, 1, 1]$ and $g_2 = [1, 0, 1]$.

All these parameters allow us to obtain the following logic diagram, enabling convolutional coding of our ECC:

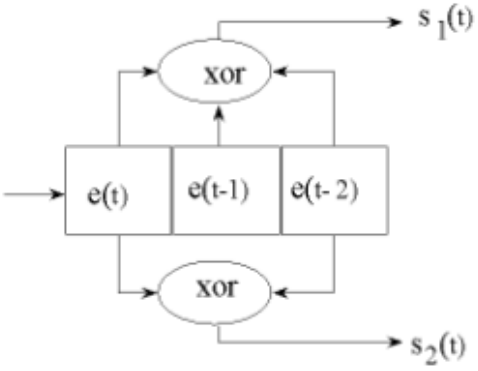


Figure 10: Data transmission in a convolutional encoder with a shift register and exclusive ORs

Using this schema and XORing according to the connections said to be absent or present on the register undergoing a shift along the chain to be encoded we obtain:

- Initial string : **1010**
- Encoded string: **11100010**

We therefore observe the presence of redundancy, as explained in the introduction, with a redundancy bit for each bit of the initial character string, which we'll use for decoding.

Before we start decoding, as we are working on noisy discrete channels, we will apply noise to the character string to check that our appended code is working correctly. Our message therefore becomes → **11100011**

3.2.3 Decoding with Viterbi algorithm

The Viterbi algorithm is based on the dynamic programming paradigm. We start from a starting state (default 00) and seek to find the path from the starting state whose output has the same length and is as close as possible to the message to be decoded. To apply this notion of proximity, we use the Hamming distance, which counts the number of different bits between two messages of the same finite or infinite length.

This Hamming distance can be translated into the following pseudo-code:

---

function HAMMING DISTANCE(s1, s2)
  if len(s1) ≠ len(s2) **then**
    return "Strings must be of equal length"
  end if
  dist ← 0
  for i ← 0 **to** len(s1) − 1 **do**
    if s1[i] ≠ s2[i] **then**
      dist ← dist + 1
    end if
  end for
  return dist
end function

In response to the dynamic programming paradigm, the Viterbi algorithm can be represented in ways other than as a lattice. For the sake of simplicity, it will be presented in the form of a binary tree, so that we can create a history of all possible paths and perform a recursion to calculate the metric of each path from that of each branch, and reconstruct the message to be decoded.

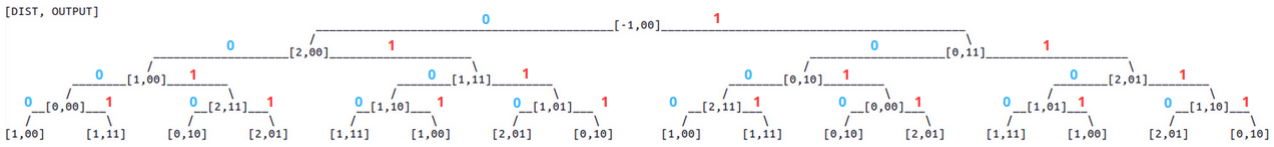


Figure 11: Representation of the binary tree of the Viterbi lattice

Using the transition table in figure n°8, we can move from one state to another and retrieve the new state, as well as the output bits, all thanks to the input bits which are 0 for the nodes on the left and 1 for the nodes on the right. A state is in fact a pair of bits resulting from the concatenation of the initial bit and the redundancy bit. If we go back up our tree, all we have to do is concatenate the input bits to reconstitute the message, even if it has been noisy. We know which is the safest path to take thanks to the distances noted above.

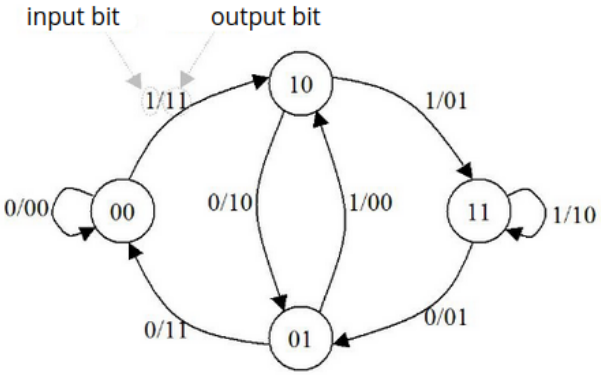


Figure 12: State diagram of a convolutional encoder with 2 memories and 1/2 rate

Now, we can retrieve the correct message that is **1010**.

3.3 Cryptography

3.3.1 Introduction

In a traditional definition of security of a cryptography scheme, one usually defines a game that characterizes the capabilities of a (hypothetical) adversary. A cryptography scheme is defined to be secure if no computationally feasible strategy allows the adversary to win the game with non-negligible probability (or advantage), for reasonable notions of feasible and negligible. [Mau11]

Important Elements of Cryptography

Plain-text Original text that is immediately intelligible and can therefore be directly, without the need for deciphering. [Ram23]

Encrypting An operation that substitutes a plain text with an unintelligible text that cannot be understood by anyone who does not possess the key. Inoperable by anyone who doesn't have the key.

Ciphertext Message made unintelligible through encryption, which can only be and used only by those in possession of the encryption key.

Decryption Reverse operation of a reversible encryption, allowing an authorized in possession of the key, to restore a ciphertext in clear text.

Types of Cryptography There are two types of Cryptography: symmetric key Cryptography and asymmetric key Cryptography, both still in use today, but they have different applications due to their different qualities.

Symmetric Cryptography Symmetric Cryptography is the first one to appear, as it is a basic concept, it is as if we had a key to open a chest. In Cesar cipher is the number of shifts we will do on the alphabet to write and read the plain-text and the ciphertext respectively. The key being the most essential element in computational Cryptography, it can be almost anything that is information, a string of text, numbers, an image, etc... That information being processed by a Cryptography algorithm it can then encode or decode Cryptographic data. Today symmetric Cryptography is used in cases where we need fast service using not much resources and not needing to share the secret key often. Such qualities interest banks for example. There are multiple symmetric cryptography algorithms such as AES, DES, 3DES, and Blowfish. All of them are based on the Feistel Network except for AES which uses a Substitution and Permutation Network.

Asymmetric Cryptography ECC (Elliptic-curve-Cryptography) Elliptic Curve Cryptography Overview Elliptic Curve Diffie Hellman A 384 bits ECC key has the same level of security as a 7680 bit RSA key, it is considered to be a Top Secret level of security by many governments such as the USA Government Shannon Information Theory and Cryptography.

3.3.2 Advanced Encryption Standard (AES)

AES was invented by two Belgians, Joan Daemen and Vincent Rijmen in 1998 and submitted to the National Institute of Standards and Technology (NIST) in 2001 in order to replace the DES (Data Encryption Standard) which was starting to get outdated. It uses a key of 3 different lengths 128, 192, and 256 bits. [Wik22]

Variables [Nat01]

- **state** Intermediate result of the AES block cipher that is represented as a two-dimensional array of bytes with four rows and Nb columns.
- **keys** the key schedule generated by the key expansion algorithm

Constants [Nat01]

- **s_box** A non-linear substitution table used in SUBBYTES and KEYEXPANSION to perform a one-to-one substitution of a byte value.
- **inv_s_box** A non-linear substitution table with the inverse values of the S-box used for the decryption.
- **mix_columns** A fixed matrix used on the MIXCOLUMNS
- **inv_mix_columns** The matrix with the hexadecimal values of mix_columns used in INVMIX-COLUMNS
- **rcon** Is the key expansion round constants used on the KEYEXPANSION

Functions [Nat01]

- **SUBBYTES** Takes the state as the parameter and applies the s_box to each byte independently.
- **SHIFTROWS** Takes the state as the parameter and transforms the state shifting the last three rows with different offsets (2nd row has 1 offset, 3rd row has 2 offsets, and 4th row has 3 offsets)
- **MIXCOLUMNS** Takes the state as the parameter and each column is multiplied by the Mix Columns matrix. But the multiplication is done in the Galois Field. Which means that the sum is done with the XOR operation and the multiplication is done with the AND operation.
- **KEYEXPANSION** Takes the key as the parameter, the routine that generates the round keys from the key.
- **ADDROUNDKEY** Takes the state and round key as the parameters and transforms the state by performing an XOR operation on the state with the round key.
- **CIPHER** Takes the plain text and the key as arguments and perform the AES algorithm using the functions SUBBYTES, SHIFTROWS, MIXCOLUMNS, KEYEXPANSION and AD-DROUNDKEY.
- **INVCIPHER** Takes the cipher text and makes the reverse operation as CIPHER.

AES Algorithm [Nat01]

Algorithm 1 CIPHER Algorithm

```
procedure CIPHER(plain_text : list[int], key : list[int])
    state ← 4 × 4 matrix of zeros
    for i ∈ [0, 4) do
        for j ∈ [0, 4) do
            if i * 4 + j < len(plain_text) then
                state[j][i] ← plain_text[i * 4 + j]
            end if
        end for
    end for
    keys ← KEYEXPANSION(key)
    state ← ADDROUNDKEY(state, keys[0])
    for i ∈ [1, 10) do
        state ← SUBBYTES(state)
        state ← SHIFTROWS(state)
        state ← MIXCOLUMNS(state)
        state ← ADDROUNDKEY(state, keys[i])
    end for
    state ← SUBBYTES(state)
    state ← SHIFTROWS(state)
    state ← ADDROUNDKEY(state, keys[10])
    return [state[i][j] for i ∈ [0, 4) for j ∈ [0, 4)]
end procedure
```

Why is AES so hard to break For symmetric Cryptography we have three properties that measures if a cryptography system is secure or not.

Confusion and Diffusion [Ram23] Confusion is when there is no statistical property that can be deduced from the ciphertext and Diffusion is when a change on the plain text changes the majority of the ciphertext.

Example of Diffusion and Confusion in AES **Plain-text 1:** Hello World!

Ciphertext 1:


```
01011111110011111111111000010100001101100110001011011001010001000
11101010000000110000111010001000100110001101100011000110010000010
```


Plain-text 2: Hallo World!

Ciphertext 2:


```
01000110011110010111111000100101000100100010011110101001100010100
10110010101100101101100010100101011111111011010111100111101001111
```


Analysis As observed, changing one letter in the plain-text resulted in a completely different ciphertext. This demonstrates the **Diffusion** property of AES. Moreover, due to the significant changes in the ciphertext, it becomes challenging to deduce statistical properties from these two messages, showcasing the **Confusion** property.

Key robustness When is hard to find or enumerate all possibilities of key.

If the algorithm has the Diffusion and the Confusion properties, the only way to find the key would be by trying every combination of key. Which means that to find the key it would be needed to try 2 to the power 128 possibilities.

Why is AES better than DES

- AES provides stronger security compared to DES.
- AES is faster and more efficient in terms of computation compared to DES.
- AES has been subjected to more scrutiny and analysis than DES, making it more reliable.

3.3.3 Elliptic-curve Cryptography (ECC)

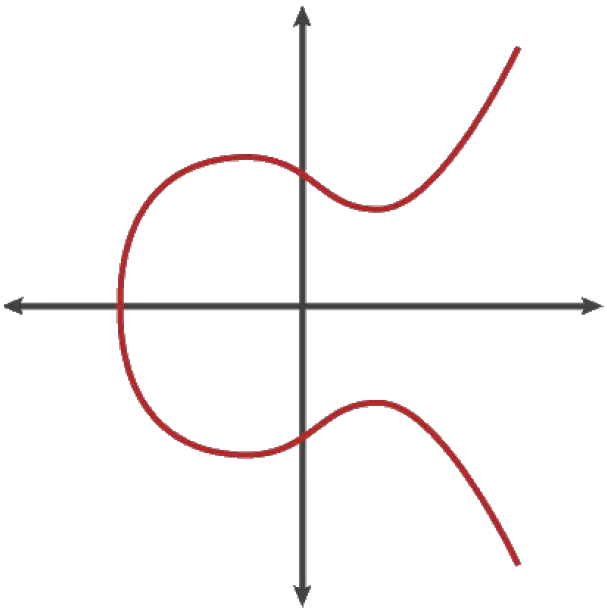
ECC (Elliptic-curve Cryptography) is a public-key cryptography algorithm based on the algebraic structure of elliptic curves over finite fields. It offers a high level of security with relatively short key sizes compared to other public-key cryptography algorithms like RSA. Elliptic Curve Diffie-Hellman (ECDH) is a key exchange protocol based on ECC, and it provides a secure way for two parties to establish a shared secret over an insecure channel. Although ECC is more secure than RSA, a 384 bit ECC key is equivalent to a 7680 RSA key for example.

How does Elliptic-curve Cryptography works [Pie14]

Curve An Elliptic-curve is defined by

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

where a and b are constants. And the curve looks like this.



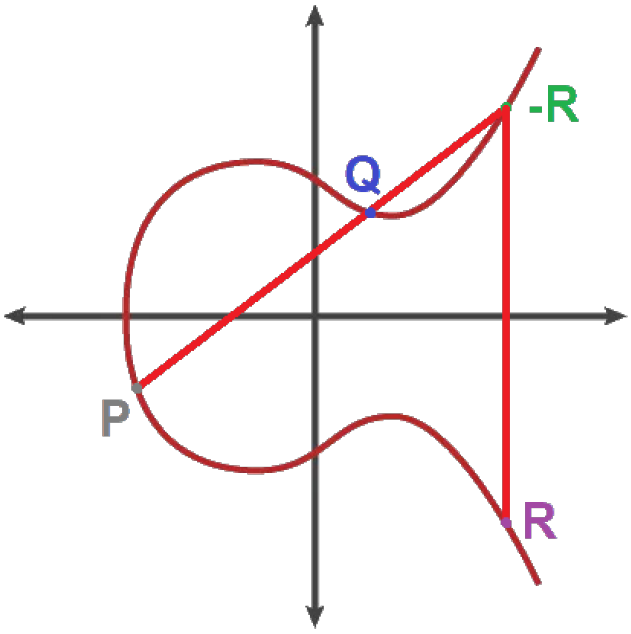
An Elliptic-curve is defined by $y^2 = x^3 + ax + b$ where a and b are constants. This figure illustrates the general shape of an elliptic curve.

Figure 13: Elliptic-curve.

Addition The addition operation consists of finding a 3rd point on the curve. In the example we see the following operation:

$$P + Q = R \tag{1}$$

When we draw a line that passes through P and Q we get $-R$ and through the property of symmetry of elliptic curves, we can find R.



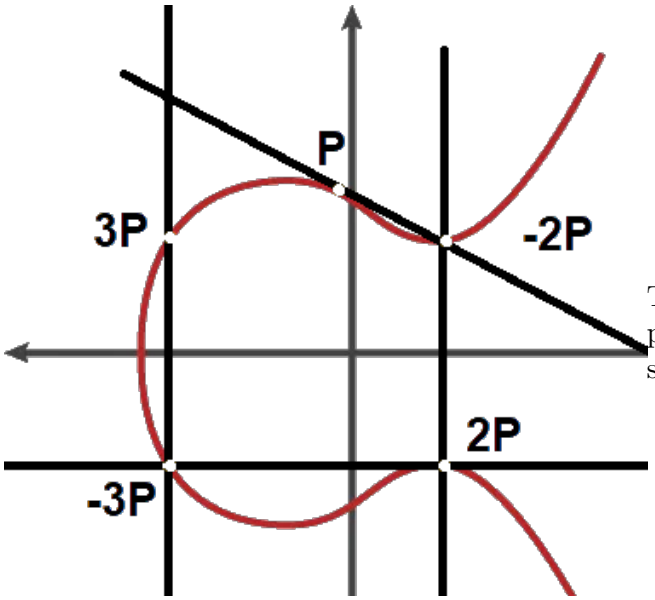
The addition operation consists of finding a third point on the curve. This figure illustrates the geometric interpretation of point addition in ECC.

Figure 14: Point addition in ECC.

Multiplication The multiplication operation consists of adding a point k times.

$$kP = P + P + \ldots + P \tag{2}$$

When adding a point to itself on a curve, the line drawn is the tangent to the curve at that point, P.



The multiplication operation consists of adding a point k times. This figure illustrates the concept of scalar multiplication in ECC.

Figure 15: Scalar Multiplication.

Implementation We will need to Implement two functions, add (make the addition of points), multiply(multiply a point).

The multiplication function will be a simple function of repeating an addition n times. For the addition it wont be as simple. So the algorithm will implement this logic.

1. If one of the points is the point at infinity (represented as (0, 0)), then adding it to another point returns the other point.
2. If the points are additive inverses (negatives) of each other, their sum is the point at infinity.
3. If the points are distinct, calculate the slope of the line passing through them.

$$s = \frac{Y_P - Y_Q}{X_P - X_Q} \tag{3}$$

$$X_R = s^2 - (X_P - X_Q) \tag{4}$$

$$Y_R = s(X_P - X_R) - Y_P \tag{5}$$

4. If the points are the same, calculate the slope of the tangent line to the curve at that point.

$$s = \frac{3X_P^2 + a}{2Y_P} \tag{6}$$

$$X_R = s^2 - 2X_P \tag{7}$$

$$Y_R = s(X_P - X_R) - Y_P \tag{8}$$

Algorithm 2 Add two points on the curve

```
1:  procedure ADD(p, q)
2:      if p = (0, 0) then
3:          return q
4:      end if
5:      if q = (0, 0) then
6:          return p
7:      end if
8:      if p[0] = q[0] and p[1] = −q[1] then
9:          return (0, 0)
10:     end if
11:     if p ≠ q then
12:         l ← (q[1] − p[1]) · (q[0] − p[0])^{−1}  mod p % p
13:     else
14:         l ← (3 · p[0]^2 + a) · (2 · p[1])^{−1}  mod p % p
15:     end if
16:     x ← (l^2 − p[0] − q[0])  mod p % p
17:     y ← (l · (p[0] − x) − p[1])  mod p % p
18:     return (x, y)
19: end procedure
```

What makes ECC secure For a asymmetric Cryptography system the most important thing is to create a common key between two parts while not revealing their own private key, so it must be easy to calculate a public key from the private key but hard to make the reverse operation from the public key, that is called a trapdoor function.

To show how ECC works let's try a practical example:

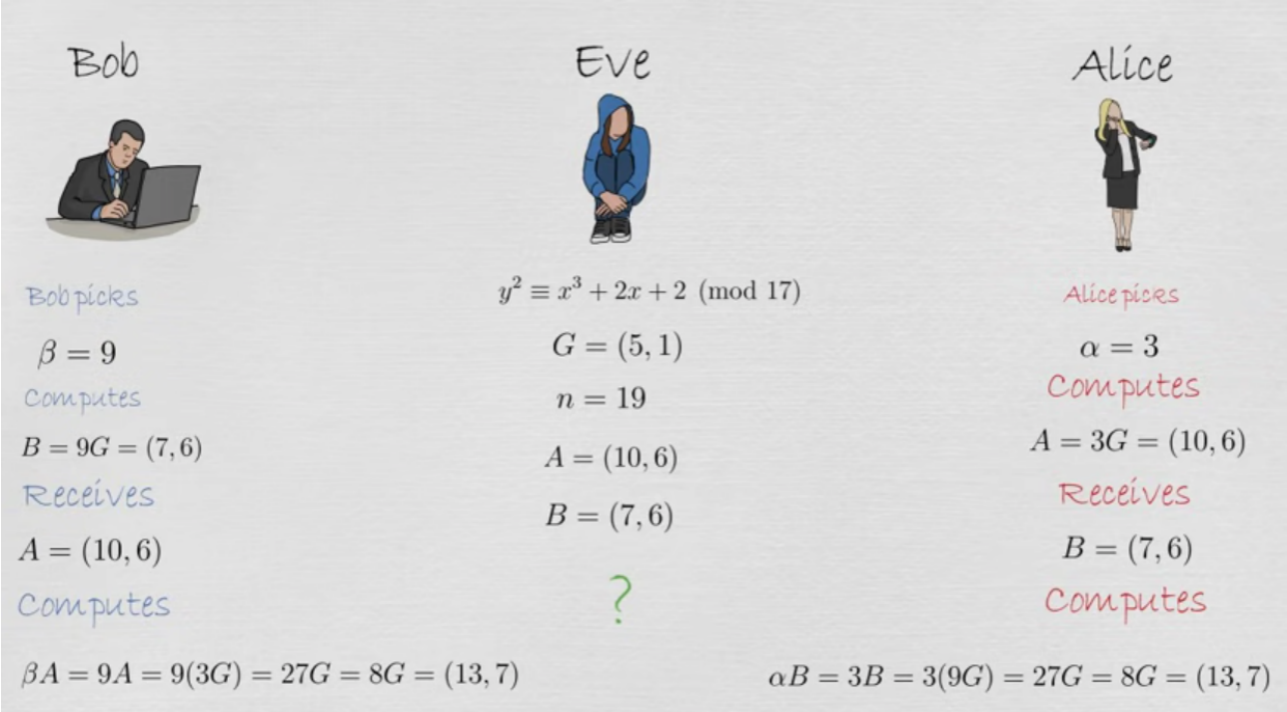


Figure 16: Practical example of ECC.

[Pie14]

In this scenario, Eve, acting as the intermediary, managed to obtain some information, albeit incomplete. However, she lacks the critical data necessary to compute the private keys. The inherent complexity of elliptic curves ensures that deriving either the private key β (Bob's private key) or the private key α (Alice's private key) in ECC solely from a point on the curve is highly challenging. This is a simple example, as there are standard parameters for the curb such as sect571k1. [CB10]

3.3.4 Cryptography and Shannon Information Theory

[Sha48]

In The Theory of Communication the encoding and decoding are operations made by the transmitter and the receiver both will be called transducers.

And according to the Theorem 9 it is not possible for the channel to transmit at an average rate greater than: $\frac{C}{H} - \epsilon$

For the AES cryptography system the limit is calculated like this:

First let's calculate the capacity C of the system. As in ASCII we have a set of characters of length 128, let's name it A* and every character (or symbol) is coded in 7 bits they have the same time of encoding or decoding. Let's define **z** which is theoretical time that a transducer takes to encode or decode a character

1. Build the system for the calculations

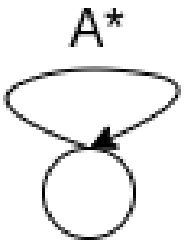


Figure 17: cryptosystem schema.

The matrix defining the system above will look like: $|(128 * W^z) - 1| = 0$

2. Find the determinant of the equation, in this case as the matrix is a singleton matrix, the determinant of the matrix is itself.
3. Solve $128 * W^z - 1 = 0$ to find W
4. Calculate the capacity $C = logW$

Secondly, H is defined as the bits per symbol as said before: $H = 7$

Finally calculate the maximum average rate of the channel $\frac{C}{H}$

$$\frac{logW}{7} \tag{9}$$

4 Conclusion

The Theory of Communication of Shannon showed to be a useful tool of prediction, on the WORD GAME where it is used to predict the path with the most possibilities to find a specific word between a set of words or measuring the maximum of information that can flow in a communication channel.

5 Appendices and Bibliography

References

[Sha48] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[Nat01] National Institute of Standards and Technology (NIST). *FIPS Publication 197: Advanced Encryption Standard (AES)*. Tech. rep. 2001. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf.

[Nic06] Nicolas Sendrier. *Sujet de projet*. 2006. URL: https://www.enseignement.polytechnique.fr/informatique/INF431/ARCHIVES/X04-2006/projets/sendrier/sujet.html.

[Cha08] Marc Chaumont. *Code Correcteurs d'Erreurs / Les codes convolutifs binaires*. 2008. URL: https://www.lirmm.fr/~chaumont/download/cours/codescorrecteur/05_codes_correcteurs_d%27erreurs_4_transparents_par_page.pdf.pdf.

[CB10] Certicom Research and Brown, D. R. L. *Standards for Efficient Cryptography SEC 2: Recommended Elliptic Curve domain Parameters*. Tech. rep. Standards for Efficient Cryptography Group (SECG), 2010. URL: https://www.secg.org/sec2-v2.pdf.

[Mau11] Ueli Maurer. "Constructive Cryptography–a new paradigm for security definitions and proofs". In: *Joint Workshop on Theory of Security and Applications*. Springer Berlin Heidelberg. Berlin, Heidelberg, Mar. 2011, pp. 33–56.

[MN12] Marie-Pierre Béal and Nicolas Sendrier. "Théorie de l'information et codage". In: (2012).

[Pie14] Robert Pierce. *Elliptic Curve Diffie Hellman*. Dec. 2014. URL: https://www.youtube.com/watch?v=F3zzNa42-tQ.

[Wag15] John Wagnon. *Elliptic Curve Cryptography Overview*. Oct. 2015. URL: https://www.youtube.com/watch?v=dCvB-mhkT0w&t.

[OPo17] O.Pothier. *Les codes convolutifs*. 2017. URL: http://wcours.gel.ulaval.ca/2017/h/GEL4200/default/7references/codage_convolutif.pdf.

[M M20] M. Moussaoui. *Code convolutif*. 2020. URL: http://ensat.ac.ma/Portail/wp-content/uploads/2020/03/Modul25_TI_cours3.pdf.

[Wik22] Wikipedia contributors. *Advanced Encryption Standard*. Wikipedia, The Free Encyclopedia. 2022. URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[Ram23] P. Ramet. *R3.09*. 2023. URL: https://ramet.gitlab.io/r3.09-crypto/.

[Wik23] Wikipedia contributors. *Logarithme décimal*. Wikipedia, The Free Encyclopedia. 2023. URL: https://fr.wikipedia.org/wiki/Logarithme_d%C3%A9cimal.

[Wik24] Wikipedia contributors. *Claude Shannon*. Wikipedia, The Free Encyclopedia. 2024. URL: https://fr.wikipedia.org/wiki/Claude_Shannon.

[con] Wikipedia contributors. *Codage de Huffman*. URL: https://fr.wikipedia.org/wiki/Codage_de_Huffman.

[Jul] Julio Hernandez-Castro. *Wheedham: An Automatically Designed Block Cipher by means of Genetic Programming*. Scientific Figure on ResearchGate. Accessed 22 Mar, 2024. URL: https://www.researchgate.net/figure/Illustration-of-a-round-in-a-Feistel-network_fig1_224645711.

[Unp] CS Unplugged. *Computer Science Field Guide*. URL: https://www.csfieldguide.org.nz/.