SIT281 :: Classic Cryptosystems

Trillion White

July 8, 2025

Abstract

This is a fun one - here I felt the contribution of my Sage-maths skills really picked up because we were using them for more impressive output other than the det() and adj() of matrices. I have so far loved this module!

1 So, hey - what are ciphers?

Ciphers are an incredibly interesting way of working on problems related to the idea of hiding information. We can create a list of numbers, letters and sometimes other symbols depending what base we are working in to hide what our message really is.

In this lesson, we are going to cover six different types of ciphers, go through some problems, and extend our knowledge in the space of cryptography by teaching ourselves how these work in the field of matrices and how we can use SageMaths to make our calculations a bit quicker when we go to working on these problems.

To explain ciphers a bit more, let us first think of a scenario.

Imagine two frogs Adam and Even are sheltering under a daisy during a storm:

Figure 1: Ciphers in detail, see [4]

See, didn't that feel nice?

Ok, back to the maths!

By the way - A mono-alphabetic cipher is called that because it only uses a single fixed letter to replace each work in the plain-text. So, there is only one alphabet and a true function conversion for each each letter of the plain-text. Because it only uses a **single** alphabet for the entire encryption process, it makes it a lot simpler to break and there are tonnes of cool resources in SageMaths that allow us to do this.

2 Vigenère cipher

The Vigenère cipher is a method of encrypting alphabetic text using a series of different Caesar ciphers, determined by a keyword. It's a poly-alphabetic substitution cipher, meaning that the same plain-text letter can be encrypted to different cipher-text letters depending on its position in the message, which makes it more secure than simple substitution ciphers.

Example step by step

1. Chose Keyword

A keyword is chosen (e.g., "LEMON").

Keyword Repetition: The keyword is repeated to match the length of the plain-text message.

Encryption: To encrypt a letter, you locate the plain-text letter in the top row of the Vigenère square and the corresponding keyword letter in the leftmost column. The cipher-text letter is found at the intersection of that row and column.

Alternatively, algebraically, if P_i is the numerical value of the i-th plain-text letter and Ki is the numerical value of the i-th keyword letter, the cipher-text letter C_i is calculated as:

$Ci = (Pi + Ki) \pmod{26}$

Decryption: The process is reversed. You find the keyword letter in the leftmost column and the cipher-text letter in that row. The plain-text letter is found at the top of the column.

We would write this as:

$Pi = (C_iK_i) \pmod{26}$

The Vigenère cipher was considered unbreakable for centuries.

3 Extension – Kasiski Examination!!!

The Vigenère cipher **was** considered unbreakable for centuries. But the Kasiski examination was one of the things that was able to prove that it could be broken.

This section goes into it below:

A very beautiful analysis is performed by Crypto Corner [3], who show an example of stepping through the key values and journey to using frequency analysis to find which letter was used in the key, but the other of the letters in the key also.

The steps that they suggest for the Kasiski method include the use of key repetition in text.

For instance, if the key is much shorter than the plain-text, parts of the key will repeat:

For example, if I use the work KEY to encrypt TRILLIANS I will end up with 3 repetitions of the KEY throughout the text. And, because my name has a lot of repeated letters, I will see repeats in the cipher-text also.

I can then use the following steps as part of Kasiskis' method:

1. Find Repeated Sequences: Identify repeating sequences of two or more letters (n-grams) within the cipher-text. When I do this, I know this is areas where the key must have been repeated.

2. Calculate Distances: By determine the distance (number of letters) between the starting positions of each repeated sequence, I get a good sense of what the distance between each repeated sequence is. These might vary and through the understanding of us not knowing what our plain-text is, these have to be taken into account.

Because we are not sure, we are dealing with probability densities and we should note these all down.

For example, if this is part of my encrypted-text:

"VHVSSP**QUCE**MRVBVBBBVHVSURQGIBDUGRNICJ**QUCE**RVUAXSSR"

If I find 5 distances, and they are 3, 1, 21 and 18, its highly unlikely that the distance is actually 1, because that would be a simple Ceasar cipher. And it probably isn't 3 either because otherwise I would see much much more repetition that just one or twice. So I might believe it is 21 or 18 instance, but I need more evidence.

3. Find Common Factors:

Calculate the factors of each distance. The most common factor among these distances is likely the length of the Vigenère key.

In the example I just gave, if we have the distances 3, 1, 21 and 18, we have the factors 1, 3, 6 and 7, and of those 1, 3, 6 are most common. If I found another distance and it was 24, that would give me 1, 3, 6, 4, and 7. And 6 stands out as being common to more than 2 in 3.

So I would try for a key length of 6.

Then, I would perform frequency analysis on the encrypted text when I brute force different values of "letter 1" for the first key.

Each time, I chose a letter than results in the best matched frequency of text to **normal human english**. If I do this each time, I end up with 6 letters that form a key.

If I use this key, I can probably decrypt my text!

Please see Crypto Corner [3] for a more beautiful in depth walk-through.

4 One Time Pad Cipher

The One-Time Pad (OTP) cipher is truly unbreakable because of the way is has been designed. This isn't just "very secure"; it means it's mathematically impossible for an attacker to break it, even with infinite computational power.

The OTP typically uses the exclusive OR (XOR) logical operation (symbolised by $\oplus$) at the bit level, or modular addition (usually modulo 26 for alphabetic characters).

Worked example below:

1. Conversion to Numbers/Bits:

First, both the plain-text message and the key are converted into a numerical representation (e.g., ASCII values, or simply 0-25 for letters).

Then, these numbers are often converted into binary (bits).

2. Encryption

Each bit of the plain-text is XORed with the corresponding bit of the key.

$cipher - text_{bit} = plain - text_{bit} \oplus Key_{bit}$

Recall the XOR truth table:

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

From this we can gain the insight that $(P \oplus K) \oplus K = P \oplus (K \oplus K) = P \oplus 0 = P$

Decryption: The cipher-text is XOR-ed with the exact same key to retrieve the plain-text.

$plain - text_{bit} = cipher - text_{bit} \oplus Key_{bit}$

Example (Simplified, using modulo 26 for letters):

Let's use a simplified example with letters, where A=0, B=1, ..., Z=25.

plain-text $(P) : HELLO$ which is represented as $(7, 4, 11, 11, 14)$

$Key(K) : XMCKL$ which is represented as $(23, 12, 2, 10, 11)$ - Crucially, this key has to be truly random and known only to sender/receiver, and same length as message.

If it is not the same length, then the key must be repeated, and then because it must be repeated then the randomness of it is reduced because now we have repeatability, and can therefore do frequency analysis.

Encryption process, using $(cipher{-}textC = (P{+}K) \pmod{26})$:. HELLO becomes ....:

$H(7) + X(23) = 30 \pmod{26} mod 26 = 4(E)$

$E(4) + M(12) = 16 \pmod{26} = 16(Q)$

$L(11) + C(2) = 13 \pmod{26} = 13(N)$

$L(11) + K(10) = 21 \pmod{26} = 21(V)$

$O(14) + L(11) = 25 \pmod{26} = 25(Z)$

.... the cipher-text: $EQNVZ$

To decrypt, we have the formula $(plain - textP = (C - K) \pmod{26})$. We apply it to the encrypted text and we obtain ...

$E(4) - X(23) = -19 \pmod{26} = 7(H)$

$Q(16) - M(12) = 4 \pmod{26} = 4(E)$

$N(13) - C(2) = 11 \pmod{26} = 11(L)$

$V(21) - K(10) = 11 \pmod{26} = 11(L)$

$Z(25) - L(11) = 14 \pmod{26} = 14(O)$

... our plain-text: $HELLO$

4.1 How come it is unbreakable?

The mathematical proof relies on the perfect randomness and one-time use of the key. If an attacker intercepts the cipher-text, say "$EQNVZ$", and they don't have the key, they cannot use any information to help them find the key.

Any plain-text message of the same length could have produced that cipher-text, given some key. For example, if the attacker wants "HELLO" to

decrypt to "WORLD", there exists a key that would produce "WORLD". If they want it to decrypt to "$ABCDE$", there's another key.

There is nothing to suggest one key should be chosen over the other, which means all are equally probable and therefore any solution is possible, which means finding the one solution tends to infinitely improbable.

Since the key is truly random, every possible plain-text of the same length is equally probable to the attacker. There's no statistical bias, no pattern, nothing for them to exploit. The cipher-text provides absolutely no information about the plain-text.

5 Shift, Affine Cipher

The Affine cipher is a type of mono-alphabetic substitution cipher, but usually has a linear rule regarding the substitution. Each letter in the plain-text is mapped to its numeric equivalent, transformed using a simple linear function $(ax + b) \pmod{m}$, and then converted back to a letter.

How it works:

Letter to Number Conversion: Similar to the Hill cipher, letters are mapped to numbers (A=0, B=1, ..., Z=25).

Key Selection: The key consists of two parts:

'a': A multiplicative key. This value must be co-prime to the size of the alphabet (m). For the English alphabet (m=26), 'a' must be co-prime to 26 (e.g., 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25).

Otherwise, we would end up with a non-unique solution that would defeat the ... say it with me .... porpoise ! :D

'b': is the addition key, which provides the shift. If you think about it, Affine is really Caesar Cipher but with some multiplication.

Then, Encryption: For each plain-text letter 'x' (its numeric equivalent), the encryption function is:

$E(x) = (ax + b) \pmod{m}$

Decryption: To decrypt a cipher-text letter 'y', the decryption function is:

$D(y) = a^{-1}(y - b) \pmod{m}$

Where a^{-1} is the modular multiplicative inverse of a $\pmod{m}$.

The Affine cipher is relatively simple and thus easily broken using frequency analysis or by knowing the plain-text of just two characters. With two characters we can craft two equations like:

$D_1(y_1) = a^{-1}(y_1 - b) \pmod{m}$
$D_2(y_2) = a^{-1}(y_2 - b) \pmod{m}$
And solve for y_1, y_2, and therefore use Cramer's rule to find the D matrix.
We can also use SageMaths or matrix inversion to find the decryption.

6 Permutation Cipher

Permutation ciphers, are different from substitution ciphers like the Caesar or Vigenère ciphers, because they do not change the identity of the letters (e.g., 'A' becomes 'D'), but rather the position of the letters within the plain-text.

This can be risky for small texts because it it very REALC that I meant CLEAR, but can be useful in large texts. However, because no substitution was performed frequency analysis can still play a large role in the way we can crack the code.

Worked Example:

plain-text: "$CRYPTOGRAPHYISFUN$", and we have the Keyword: $SECRET$.

We now work out the Order: 521436 ($C = 1, E = 2, E = 3, R = 4, S = 5, T = 6$)

We put the plain text, next to the order, and then use the message to be encrypted by this key:

S	**E**	**C**	**R**	**E**	**T**
5	2	1	4	3	6
C	R	Y	P	T	O
G	R	A	P	H	Y
I	S	F	U	N	X

Table 1: Columnar Transposition Table

The cipher-text is found by then reading the columns by order 1, 2, 3, 4, 5, 6): Read down the length of the column to find the whole length of the text.

cipher-text: $YAFRSHOYNPPGCGITUX$

The decryption process is just the reverse of encryption. We write the cipher-text into columns based on the keyword length and order, then read off the rows.

7 Block Cipher

Earlier we saw how modular matrices could be used for addition and to create essentially blocks. Today we develop those ideas further and say that these 'blocks' can be used to represent letters, and that they can work in what are called 'block ciphers'.

They allow us to set a sentence according to a square encryption matrix, that, depending on its side length, allow you to encrypt/decrypt messages in "blocks" according to that dimension. A textbook might say you work on 'blocks' of values and distributed them over several rotors using a key.

Block ciphers have become very popular in implementation on computers as 32, 64, 128 are natural block lengths to use when bits are being manipulated. So, we could have a plain text divided into blocks of length 8, and then each block would need to be treated as a vector and multiplied by an 8 x 8 encryption matrix.

They are different from stream ciphers, as these will encrypt data bit by bit. Block ciphers operate on fixed-size blocks of plain-text data, and then allow us to chunk out a block. Its a very efficient process and allows us to work relatively quickly in both directions.

Both encryption and decryption use the same secret key. So, I could then take the inverse of the encryption key to the "blocks" of text if I wanted to decrypt the message.

Popular and widely used block ciphers include the Advanced Encryption Standard (AES), Triple DES (3DES), and Blowfish, but we haven't covered these just yet.

The Hill Cipher is a special kind of block cipher.

8 Hill Cipher

The Hill cipher, invented by Lester S. Hill in 1929, is a polygraphic substitution cipher that uses linear algebra for encryption and decryption. This means it encrypts blocks of letters rather than individual letters, making it more robust against simple frequency analysis than mono-alphabetic ciphers.

Example and instruction

1. Conversion Each letter of the alphabet is assigned a numerical value (e.g., A=0, B=1, ..., Z=25).

2. Key matrix is chosen. The square matrix of integers is chosen as

the encryption key, similar to block ciphers. This matrix must be invertible modulo 26 (for the English alphabet) to allow for decryption. The size of the matrix determines the block size (e.g., a 2x2 matrix encrypts blocks of two letters).

3. Conversion of the letters into vectors The plain-text message is broken into blocks of letters, and each block is converted into a column vector of numbers.

4. Encryption Encryption: To encrypt, the plain-text vector is multiplied by the key matrix, and the result is taken modulo 26. This produces a cipher-text vector, which is then converted back to letters.

$C = (KP) \pmod{26}$

Where C is the cipher-text vector, K is the key matrix, and P is the plain-text vector.

5. Decryption

Decryption simply involves multiplying the cipher-text vector by the modular multiplicative inverse of the key matrix.

$P = (K^1C) \pmod{26}$

Where K1 is the inverse of the key matrix modulo 26.

9 What I learned most

I would say the bit that I found the easiest to learn from and about was the way in which I could use Sage-Maths to answer questions and help me learn as I went along.

It was fun seeing how quick and effective it is at making problems solvable! I am usually someone who likes to do things by hand, but it was enjoyable seeing how much time it took off the hand-written stuff for the matrices.

Pretty cool stuff.

References

[1] "Rule of Sarrus," *Wikipedia*, [Online]. Available: https://en.wikipedia.org/wiki/Rule_of_Sarrus

[2] M. Hikari, *Journal of Advanced Mathematics (IJA)*, vol. 1, no. 4, pp. 2–X, 2021. [Online]. Available: https://www.m-hikari.com/ija/ija-2021/ija-1-4-2021/p/salinasIJA1-4-2021-2.pdf

[3] Daniel Rodriguez-Clark, *Kasiski Analysis: Breaking the Code* [Online] Available: https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.htmlencrypt

[4] Ajar Setaidi, "Worlds Cutest Froggos", [Online]. Avilable at: https://mymodernmet.com/frog-photos-ajar-setiadi/