Monoalphabetic, mono-literal simple substitution systems.

This term refers to systems in which a symbol or letter is replaced by a different one (mono-literal)—specifically, the same replacement is always used, meaning a single substitution alphabet is employed (monoalphabetic). Essentially, we use two alphabets for encryption and decryption. The encryption (and decryption) process is straightforward: we take each letter of the message to be encrypted, identify its position, and replace it with the corresponding letter from the other alphabet. We refer to the base alphabet as the "plaintext alphabet" and the other as the "ciphertext alphabet." For example, if we have the following alphabets:

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

The plaintext message to be encrypted is: ATAQUE AL ALBA. To encrypt it, we locate the first letter and replace it with the one appearing below it. In this case, the A becomes an R. The second letter is T, which we replace with the letter below it—E—and we continue this process until we have processed every letter of the plaintext message. To make the task more difficult for a potential cryptanalyst, we remove the spaces, resulting in the following ciphertext:

Plaintext	A	T	A	Q	U	E	A	L	A	L	B	A
Ciphertext	**R**	**E**	**R**	**Y**	**G**	**Z**	**R**	**N**	**R**	**N**	**T**	**R**

As we shall see, despite the large number of theoretical possibilities (n!, where n is the number of elements in the alphabet), the reality is that monoalphabetic substitution ciphers are very easy to break simply by relying on the statistical properties of the language.
One of the problems associated with using a randomized alphabet is the difficulty of memorizing it. However, there is a simple way to generate a substitution alphabet without needing to memorize the specific letter mappings. To do this, one simply uses a keyword. The alphabet is created by placing the keyword at the beginning—removing any duplicate letters—and then appending the remaining letters of the standard alphabet in a predetermined order. For example, if we use the word SEGURIDAD as the keyword, the resulting alphabet would be:

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

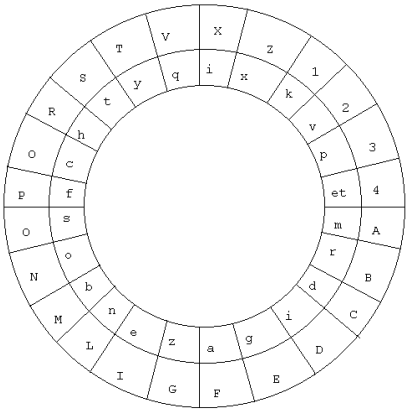
A safer variant of this method is to arrange the characters of the alphabet in a table and select the data from the columns, either in positional order or in alphabetical order.

S	E	G	U	R	I
D	A	B	C	F	H
J	K	L	M	N	Ñ

O	P	Q	T	V	W
X	Y	Z			

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

A method widely used throughout history involves replacing letters with unusual symbols or shapes based on geometric forms. The latter approach is exemplified by the "pigpen" cipher—also known as the Masonic or Templar cipher—while the former is typical of cryptography from the 15th and 16th centuries. It is worth noting that these methods offer no greater security than using a standard, randomized alphabet.
Finally, we can use a circular layout to represent the key, just like in the Alberti cipher. Note, however, that Alberti used it as a polyalphabetic cipher—a method we will cover in another section.



Shift method.

This term refers to all methods that involve replacing a character of the alphabet with another shifted by *x* positions. Generally speaking, if *n* is the number of elements in the alphabet and *k* is the shift—with a value between 1 and *n*–1 (using *n* for *k* would yield the original value, meaning no encryption would take place)—then:

$$C_k = m + k (\mathrm{mod}\, n)$$

The mod (or modulo) operation can be simply defined as the remainder of division by n. For example, 35 mod 27 yields 8.

This method is very easy to crack through the simple application of statistical techniques, a table of possible permutations, or an exhaustive study of the keys until the correct one is found, given that there are only 26 possibilities.

Fun fact: If we use *k* = 3, we get Julius Caesar's encryption method. Octavian Augustus was a bit lazier and used *k* = 1. For movie buffs: the name of the computer in Kubrick's film “2001: A Space Odyssey” was encrypted using *k* = -1; shifting each letter forward by one position to decrypt it transforms HAL into IBM.

Affine Cipher.

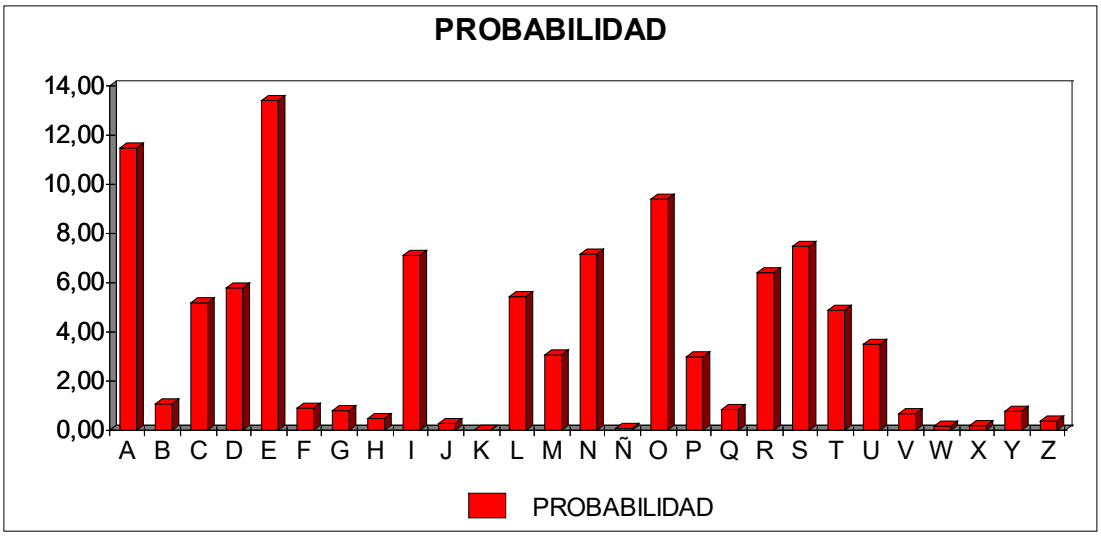
This is a generalization of the previous method, employing a linear transformation of the form $C_k = am_k + b(\mathrm{mod}\, n)$. By setting $a = 1$, we obtain the shift cipher.
In this case, *a* must be coprime to *n*; otherwise, decryption could lead to ambiguities. Let $\left.\begin{array}{l} C_1 = am_1 + b(\mathrm{mod}\, n) \\ C_2 = am_2 + b(\mathrm{mod}\, n) \end{array}\right\}$ and suppose $C_1 = C_2$ In this case, we have $a(m_1 - m_2) = 0(\mathrm{mod}\, n)$, or, in other words, $a(m_1 - m_2) = kn$. For a unique solution to exist and $m_1 = m_2$, *a* and *n* must not share any common factors—that is, they must be relatively prime. Two numbers are said to be relatively prime if they share no common factors.
To decrypt, we must calculate the modular inverse of *a* and apply the following formula:

$$m = (c_k - b).a^{-1} mod\ n$$

Cryptanalysis of simple substitution methods.

Several methods can be used to attack them. The most rudimentary is the brute-force attack; this can work well with the shift method—since one only needs to test 26 possibilities with a 27-letter alphabet—but it is ineffective against random-alphabet ciphers.

One of the most powerful methods involves using basic language characteristics, such as the frequency of specific characters and their most common combinations. This approach narrows the search space for the solution. Every language contains a set of frequently repeated letters and others that are used much less often. Pairs of letters (digraphs) or triplets (trigrams) also appear frequently; cryptanalysts exploit all these characteristics to deduce the original message from the ciphertext. One of the most famous examples of this type of attack is the one employed by the character Legrand in Edgar Allan Poe's “The Gold-Bug”. Below is a chart showing letter frequencies in Spanish. If a letter appears frequently, it is highly likely to be an E or an A. We continue testing until we determine the alphabet used.



Another very practical attack is the “probable word” method. If we know or suspect that a specific word appears in the message, we can search for that word’s pattern within the ciphertext and substitute it wherever it might occur. For example, if we know that one of the words in the message is “atacad,” we would look for a segment of the ciphertext where the first, third, and fifth letters are identical, and assume that this is the word's position. This method, combined with the use of language frequency analysis, yields excellent results.

Tricks to make cryptanalysis more difficult.

Unfortunately, if the system is inherently weak, we cannot turn it into a strong one. However, we can bolster it slightly by removing spaces—so the cryptanalyst cannot tell where each word begins and ends—and by inserting null characters into the message to be encrypted. Ideally, these nulls should consist of letters with a low probability of occurrence, so as to mask the resulting letter frequency distribution. A downside is that this increases the length of the message to be encrypted. For example, if we wanted to encrypt the phrase “la inteligencia es como las pulgas, van saltando de una cabeza a otra, pero no todas se la quedan” (in English “intelligence is like fleas: they jump from one head to another, but not everyone keeps them”), we would start by removing the spaces and punctuation marks, resulting in:

Lainteligenciaescomolaspulgasvansaltandodeunacabezaaotraperonotodasselaquedan

In the second step, we will insert a null letter after every five plaintext letters. Ideally, this would be done randomly, but this will serve as an example. We have marked the null letters in red.

Laintzeligeñnciaefscomoxlaspuhlgasvpansalqtandowdeunaqcabezzaaotrvaperoznotod yasseljaqueddan

The final step would be to encrypt the message and send it.