

# Package ‘ciphertext’

May 8, 2026

**Type** Package

**Title** Classical Cryptography Methods for Words and Phrases

**Version** 0.1.1

**Description** Classical cryptography methods for words and brief phrases.  
Substitution, transposition and concealment (null) ciphers are available, like  
Caesar, Vigenère, Atbash, affine, simple substitution, Playfair,  
rail fence, Scytale, single column, bifid, trifid, and Polybius ciphers.

**License** GPL-3

**URL** <https://github.com/Luigi-Annic/ciphertext>

**BugReports** <https://github.com/Luigi-Annic/ciphertext/issues>

**Encoding** UTF-8

**Depends** R (>= 4.3.0)

**RoxygenNote** 7.2.3

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Luigi Annicchiarico [cre, aut]

**Maintainer** Luigi Annicchiarico <luigi.annic@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-01-10 15:10:05 UTC

## Contents

affine	2
atbash	3
bifid_delastelle	3
caesar	4
nullcipher	5
playfair	5
polybius	6

railfence . . . . .	7
scytale . . . . .	8
simple_substitution . . . . .	8
singlecolumn . . . . .	9
trifid_delastelle . . . . .	10
vigenere . . . . .	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

affine	<i>affine</i>
--------	---------------

---

### Description

The affine cipher is a monoalphabetic substitution cipher, where each letter is enciphered with the function  $(ax+b) \bmod 26$  (26 is the number of letters in the alphabet)

### Usage

```
affine(word, a, b, encrypt = TRUE)
```

### Arguments

word	Word or phrase to be encrypted
a	First parameter. This value and 26 must be coprime
b	Second parameter. Magnitude of the shift
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

### Value

a string

### References

[https://en.wikipedia.org/wiki/Affine\\_cipher](https://en.wikipedia.org/wiki/Affine_cipher)

### Examples

```
affine("Hello", 1, -1)
```

---

atbash	<i>atbash</i>
--------	---------------

---

**Description**

The Atbash cipher is a type of monoalphabetic cipher which takes the alphabet and maps it to its reverse. It is a particular case of the affine cipher, with  $a=b=(m-1)$ . As  $m$  is the number of letters and is equal to 26, it means that  $a=b=25$ . Encrypting and decrypting are not separate for this cipher.

**Usage**

```
atbash(word)
```

**Arguments**

word	Word or phrase to be encrypted
------	--------------------------------

**Value**

a string

**References**

<https://en.wikipedia.org/wiki/Atbash>

**Examples**

```
atbash("abcxyz")
```

---

bifid_delastelle	<i>bifid_delastelle</i>
------------------	-------------------------

---

**Description**

The bifid cipher is an encryption method that combines a substitution with a Polybius square and a transposition, and uses fractionation to achieve diffusion. It was invented by Felix Delastelle.

**Usage**

```
bifid_delastelle(input, key = "", period = 100, encrypt = TRUE)
```

**Arguments**

input	Word or phrase to be encrypted, or character vector with the sequence of coordinate numbers if we need to decrypt
key	key Word for creating the modified Polybius square
period	period length for splitting the input phrase. If greater or equal to the length of the input then the split is not executed
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

**Value**

a string

**References**

[https://en.wikipedia.org/wiki/Bifid\\_cipher](https://en.wikipedia.org/wiki/Bifid_cipher)

**Examples**

```
bifid_delastelle("dcode", key = "secret", period = 3, encrypt = TRUE)
bifid_delastelle("apiai", key = "secret", period = 3, encrypt = FALSE)
```

---

caesar

*caesar*

---

**Description**

caesar encryption

**Usage**

```
caesar(word, key = 1, encrypt = TRUE)
```

**Arguments**

word	Word or phrase to be encrypted
key	numeric key
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

**Value**

a string

**Examples**

```
caesar("Hello", 1)
```

---

nullcipher	<i>nullcipher</i>
------------	-------------------

---

**Description**

A null cipher is an encryption method where the plaintext is mixed with a large amount of non-cipher material (decoy).

**Usage**

```
nullcipher(phrase, index, encrypt = FALSE)
```

**Arguments**

phrase	Word or phrase to be decrypted
index	letter of interest for each word in the phrase. Also a pattern vector can be entered.
encrypt	Only Decryption is possible for now, but will be updated in the future

**Value**

a string

**References**

[https://en.wikipedia.org/wiki/Null\\_cipher](https://en.wikipedia.org/wiki/Null_cipher)

**Examples**

```
nullcipher("handy set false posts", c(1,2,3))
```

---

playfair	<i>playfair</i>
----------	-----------------

---

**Description**

The Playfair cipher is a symmetric method which encrypts pairs of letters using a modified Polybius square

**Usage**

```
playfair(word, key = "", added_letter = "x", encrypt = TRUE)
```

**Arguments**

word	Word or phrase to be encrypted or decrypted
key	Word for creating the modified Polybius square
added_letter	Letter to be added in case two letters of a pair are identical; usually "x" is used
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

**Value**

a string

**References**

[https://en.wikipedia.org/wiki/Playfair\\_cipher](https://en.wikipedia.org/wiki/Playfair_cipher)

**Examples**

```
playfair("instruments", "monarchy", added_letter = "z")
playfair("gatlmzclrqtx", "monarchy", added_letter = "z", encrypt = FALSE)
```

---

polybius

*polybius*

---

**Description**

The polybius square is a device which associates each letter to a pair of coordinates. The letter J is excluded and replaced with I in order to get 25 letters and create a 5x5 matrix.

**Usage**

```
polybius(input, encrypt = TRUE)
```

**Arguments**

input	Word or phrase to be encrypted, or character vector with the sequence of coordinate numbers if we need to decrypt
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

**Value**

a string

**References**

[https://en.wikipedia.org/wiki/Polybius\\_square](https://en.wikipedia.org/wiki/Polybius_square)

**Examples**

```
polybius("hello world")  
polybius("23 15 31 31 34 52 34 42 31 14", encrypt = TRUE)
```

---

<i>railfence</i>	<i>railfence</i>
------------------	------------------

---

**Description**

The rail fence is a transposition cipher where the text is written upwards and downwards diagonally (zigzag) on the rails of the fence

**Usage**

```
railfence(word, key = 3)
```

**Arguments**

- word            Word or phrase to be encrypted
- key             numeric key (number of rails)

**Value**

a string

**References**

[https://en.wikipedia.org/wiki/Rail\\_fence\\_cipher](https://en.wikipedia.org/wiki/Rail_fence_cipher)

**Examples**

```
railfence('we are discovered flee at once',3)
```

---

scytale                      *scytale*

---

### Description

The Scytale is a transposition cipher. The diameter of the Scytale (the number of turns) can be regarded as the key of the cipher.

### Usage

```
scytale(word, key = 3, encrypt = TRUE)
```

### Arguments

word	Word or phrase to be encrypted or decrypted
key	Number of turns of the band
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

### Value

a string

### References

<https://en.wikipedia.org/wiki/Scytale>

### Examples

```
scytale('we are discovered flee at once', 3)
```

---

simple\_substitution      *simple\_substitution*

---

### Description

simple substitution cipher. Each letter is monoalphabetically associated with a different one used for the encryption.

### Usage

```
simple_substitution(word, key = "", seed = sample(1:1000, 1))
```

**Arguments**

word	Word or phrase to be encrypted
key	Word to be used as key for the encryption. If not provided, a random shuffle is performed
seed	Seed for reproducibility of the encryption if key is not provided

**Value**

a list with custom class "cipher", which modifies the printing defaults. The list contains the initial phrase (initial), the ciphered output (encrypted), and the alphabet order (keyalphabet)

**Examples**

```
simple_substitution("hello world", seed = 1234)
simple_substitution("hello world", key = "zebras")
```

---

singlecolumn	<i>singlecolumn</i>
--------------	---------------------

---

**Description**

In a columnar transposition cipher, the message is written out in rows of a fixed length, and then read out again column by column. The order of the column follows the alphabetical order of the letters present in the key

**Usage**

```
singlecolumn(word, key, rm.blanks = TRUE)
```

**Arguments**

word	Word or phrase to be encrypted
key	word key: for example, the key "bcea" suggests that the column order is "2-3-4-1"
rm.blanks	Should spaces between words be removed? By default set to 'TRUE'

**Value**

a string

**References**

<https://www.geeksforgeeks.org/columnar-transposition-cipher/>

**Examples**

```
singlecolumn("This is wikipedia", "cipher")
```

---

trifid_delastelle	<i>trifid_delastelle</i>
-------------------	--------------------------

---

### Description

The trifid cipher is an encryption method that uses a 3-dimensional grid. It was invented by Felix Delastelle in 1902. As a 3x3x3 grid is used, 27 characters are needed. Thus, we use all the 26 alphabet letters and add the "+" sign at the bottom.

### Usage

```
trifid_delastelle(input, key = "", period = 100, encrypt = TRUE)
```

### Arguments

input	Word or phrase to be encrypted, or character vector with the sequence of coordinate numbers if we need to decrypt
key	key Word for creating the modified Polybius square
period	period length for splitting the input phrase. If greater or equal to the length of the input then the split is not executed
encrypt	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

### Value

a string

### References

[https://en.wikipedia.org/wiki/Trifid\\_cipher](https://en.wikipedia.org/wiki/Trifid_cipher)

### Examples

```
trifid_delastelle("secret", key = "", period = 5, encrypt = TRUE)
trifid_delastelle("sjlkzt", key = "", period = 5, encrypt = FALSE)
```

---

`vigenere``vigenere`

---

**Description**

Vigenère cipher is a method of encrypting alphabetic text where each letter of the plaintext is encoded with a different Caesar cipher, whose increment is determined by the corresponding letter the key

**Usage**

```
vigenere(word, key, encrypt = TRUE)
```

**Arguments**

<code>word</code>	Word or phrase to be encrypted
<code>key</code>	character key
<code>encrypt</code>	If 'TRUE' (default), the program ciphers the input word, If 'FALSE', the program decrypts it.

**Value**

a string

**References**

<https://en.wikipedia.org/wiki/Vigenere>

**Examples**

```
vigenere("hello world", "opla")
```

# Index

affine, [2](#)

atbash, [3](#)

bifid\_delastelle, [3](#)

caesar, [4](#)

nullcipher, [5](#)

playfair, [5](#)

polybius, [6](#)

railfence, [7](#)

scytale, [8](#)

simple\_substitution, [8](#)

singlecolumn, [9](#)

trifid\_delastelle, [10](#)

vigenere, [11](#)