

---

# SecretPy Documentation

*Release 1.0*

**[Read the Docs](#)**

**May 24, 2019**



---

## Contents:

---

<b>1</b>	<b>SecretPy</b>	<b>1</b>
1.1	Description . . . . .	1
1.2	Installation . . . . .	2
1.3	Usage . . . . .	2
1.3.1	Direct way . . . . .	2
1.3.2	CryptMachine . . . . .	3
1.3.3	CompositeMachine . . . . .	4
1.4	Maintainers . . . . .	5
<b>2</b>	<b>API</b>	<b>7</b>
2.1	ADFGX . . . . .	7
2.1.1	Examples . . . . .	8
2.2	ADFGVX . . . . .	9
2.2.1	Examples . . . . .	9
2.3	Affine . . . . .	10
2.3.1	Examples . . . . .	10
2.4	Atbash . . . . .	11
2.4.1	Examples . . . . .	12
2.5	Autokey . . . . .	13
2.5.1	Examples . . . . .	13
2.6	Bazeries . . . . .	14
2.6.1	Examples . . . . .	15
2.7	Beaufort . . . . .	15
2.7.1	Examples . . . . .	16
2.8	Bifid . . . . .	17
2.8.1	Examples . . . . .	17
2.9	Caesar . . . . .	18
2.9.1	Examples . . . . .	19
2.10	Caesar Progressive . . . . .	20
2.10.1	Examples . . . . .	20
2.11	Columnar Transposition . . . . .	21
2.11.1	Examples . . . . .	22
2.12	Four Square . . . . .	22
2.12.1	Examples . . . . .	23
2.13	Gronsfeld . . . . .	24
2.13.1	Examples . . . . .	24

2.14	Keyword . . . . .	25
2.14.1	Examples . . . . .	26
2.15	Nihilist . . . . .	26
2.15.1	Examples . . . . .	27
2.16	Playfair . . . . .	29
2.16.1	Examples . . . . .	29
2.17	Polybius . . . . .	30
2.17.1	Examples . . . . .	31
2.18	Porta . . . . .	31
2.18.1	Examples . . . . .	32
2.19	Rot13 . . . . .	33
2.19.1	Examples . . . . .	33
2.20	Rot5 . . . . .	34
2.20.1	Examples . . . . .	35
2.21	Rot18 . . . . .	35
2.21.1	Examples . . . . .	36
2.22	Rot47 . . . . .	36
2.22.1	Examples . . . . .	37
2.23	Simple Substitution . . . . .	37
2.23.1	Examples . . . . .	38
2.24	Three Square . . . . .	39
2.24.1	Examples . . . . .	39
2.25	Trifid . . . . .	40
2.25.1	Examples . . . . .	41
2.26	Two Square . . . . .	42
2.26.1	Examples . . . . .	42
2.27	Vic . . . . .	43
2.27.1	Examples . . . . .	44
2.28	Vigenere . . . . .	44
2.28.1	Examples . . . . .	45
2.29	Zigzag . . . . .	46
2.29.1	Examples . . . . .	46

### 3 Indices and tables 49

**Download:**

<https://pypi.org/project/secretpy>

**Documentation:**

<https://secretpy.readthedocs.io>

**Source code & Development:**

<https://github.com/tigertv/secretpy>

## 1.1 Description

SecretPy is a cryptographic Python package. It uses the following classical cipher algorithms:

- ADFGX, ADFGVX
- Affine
- Atbash
- Autokey
- Bazeries
- Beaufort
- Bifid
- Caesar, Caesar Progressive
- Columnar Transposition
- Keyword
- Nihilist

- Simple Substitution
- Playfair, Two Square(Double Playfair), Three Square, Four Square
- Polybius
- Rot13, Rot5, Rot18, Rot47
- Trifid
- Vic
- Vigenere, Gronsfeld, Porta
- Zigzag(Railfence)

## 1.2 Installation

To install this library, you can use pip:

```
pip install secretpy
```

Alternatively, you can install the package using the repo's cloning and the make:

```
git clone https://github.com/tigertv/secretpy
cd secretpy
make install
```

## 1.3 Usage

### 1.3.1 Direct way

The cipher classes can encrypt only letters which exist in the alphabet, and they don't have a state.

```
from secretpy import Caesar
from secretpy import alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = 3
cipher = Caesar()

print(plaintext)
enc = cipher.encrypt(plaintext, key, alphabet)
print(enc)
dec = cipher.decrypt(enc, key, alphabet)
print(dec)

print('=====')

print(plaintext)
# use default english alphabet
enc = cipher.encrypt(plaintext, key)
print(enc)
dec = cipher.decrypt(enc, key)
```

(continues on next page)

(continued from previous page)

```

print(dec)

'''
Output:

thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqirämxpsvryhuwkhodüögrj
thequickbrownfoxjumpsoverthelazydog
=====
thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
'''

```

### 1.3.2 CryptMachine

CryptMachine saves state. There are alphabet, key and cipher, they can be changed in anytime. In the previous example, plaintext contains only letters existing in the alphabet and in the lower case without spaces. To change the behaviour, you can use CryptMachine and decorators(UpperCase, NoSpace, SaveCase and etc.), so it's a preferred way to do encryption/decryption:

```

from secretpy import Atbash
from secretpy import Caesar

from secretpy import CryptMachine
from secretpy.cmdecorators import UpperCase, SaveSpaces, NoSpaces
from secretpy import alphabets

def encdec(machine, plaintext):
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)
    print("-----")

plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = 3
cipher = Caesar()

cm = CryptMachine(cipher, key)
encdec(cm, plaintext)

cm.set_alphabet(alphabets.GERMAN)
encdec(cm, plaintext)

cm = SaveSpaces(cm)
cm.set_key(9)
plaintext = u"the quick brown fox jumps over the lazy dog"
encdec(cm, plaintext)

cm = NoSpaces(UpperCase(cm))
cm.set_cipher(Atbash())

```

(continues on next page)

(continued from previous page)

```

plaintext = u"Achtung Minen"
encdec(cm, plaintext)

'''
Output:

thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
-----
thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodüögrj
thequickbrownfoxjumpsoverthelazydog
-----
the quick brown fox jumps over the lazy dog
üqn zßrlt käxbw oxc sßvyö xanä üqn ujed mxp
the quick brown fox jumps over the lazy dog
-----
Achtung Minen
ßÖWKJQXRVQZQ
ACHTUNGMINEN
-----
'''

```

### 1.3.3 CompositeMachine

Combining several ciphers to get more complex cipher, you can use CompositeMachine:

```

from secretpy import Rot13
from secretpy import Caesar
from secretpy import CryptMachine
from secretpy import CompositeMachine
from secretpy.cmdecorators import SaveCase, SaveSpaces

def encdec(machine, plaintext):
    print("=====")
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)

key = 5
plaintext = u"Dog jumps four times and cat six times"
print(plaintext)

cm1 = SaveSpaces(SaveCase(CryptMachine(Caesar(), key)))
enc = cm1.encrypt(plaintext)
print(enc)

cm2 = SaveSpaces(SaveCase(CryptMachine(Rot13())))
enc = cm2.encrypt(enc)
print(enc)

```

(continues on next page)



(continued from previous page)

```

print("=====")

cm = CompositeMachine(cm1)
cm.add_machines(cm2)
enc = cm.encrypt(plaintext)
print(enc)

encdec(cm, plaintext)

cm.add_machines(cm1, cm2)
encdec(cm, plaintext)

'''
Output:

Dog jumps four times and cat six times
Itl ozrux ktzw ynrjx fsi hfy xnc ynrjx
Vgy bmehk xgmj laewk sfv usl kap laewk
=====
Vgy bmehk xgmj laewk sfv usl kap laewk
=====
Dog jumps four times and cat six times
Vgy bmehk xgmj laewk sfv usl kap laewk
Dog jumps four times and cat six times
=====
Dog jumps four times and cat six times
Nyq tewzc pyeb dswoc kxn mkd csh dswoc
Dog jumps four times and cat six times
'''

```

## 1.4 Maintainers

- @tigertv (Max Vetrov)



## 2.1 ADFGX

**class** `secretpy.ADFGX`

The ADFGX Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

## 2.1.1 Examples

```
1 #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ADFGX
5
6  alphabet = [
7      u"b", u"t", u"a", u"l", u"p", u"d", u"h", u"o", u"z", u"k", u"q",
8      u"f", u"v", u"s", u"n", u"g", u"ij", u"c", u"u", u"x", u"m", u"r",
9      u"e", u"w", u"y"
10 ]
11
12 plaintext = u"attackatonce"
13 key = "cargo"
14 cipher = ADFGX()
15
16 print(plaintext)
17 enc = cipher.encrypt(plaintext, key, alphabet)
18 print(enc)
19
20 dec = cipher.decrypt(enc, key, alphabet)
21 print(dec)
22
23 #####
24 print("-----")
25
26 alphabet = [
27     u"f", u"n", u"h", u"e", u"q",
28     u"r", u"d", u"z", u"o", u"c",
29     u"ij", u"s", u"a", u"g", u"u",
30     u"b", u"v", u"k", u"p", u"w",
31     u"x", u"m", u"y", u"t", u"l"
32 ]
33 key = "battle"
34 plaintext = "attackatdawn"
35
36 print(plaintext)
37 enc = cipher.encrypt(plaintext, key, alphabet)
38 print(enc)
39
40 dec = cipher.decrypt(enc, key, alphabet)
41 print(dec)
42
43 #####
44 print("-----")
45
46 key = "deutsch"
47 plaintext = "howstuffworks"
48
49 # use default english alphabet 5x5
50 print(plaintext)
51 enc = cipher.encrypt(plaintext, key)
52 print(enc)
53
54 dec = cipher.decrypt(enc, key)
55 print(dec)
```

## 2.2 ADFGVX

**class** secretpy.ADFGVX

The ADFGVX Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.2.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ADFGVX, CryptMachine
5
6
7  def encdec(machine, plaintext):
8      print(plaintext)
9      enc = machine.encrypt(plaintext)
10     print(enc)
11     print(machine.decrypt(enc))
12     print("-----")
13
14
15  key = "cargo"
16  cm = CryptMachine(ADFGVX(), key)
17
18  alphabet = [
19      u"f", u"n", u"h", u"e", u"q", u"0",
20      u"r", u"d", u"z", u"o", u"c", u"9",
21      u"ij", u"s", u"a", u"g", u"u", u"8",

```

(continues on next page)

(continued from previous page)

```

22     u"b", u"v", u"k", u"p", u"w", u"7",
23     u"x", u"m", u"y", u"t", u"1", u"6",
24     u"1", u"2", u"3", u"4", u"5", u".",
25 ]
26 cm.set_alphabet(alphabet)
27 key = "battle"
28 plaintext = "attackatdawn11.25"
29 encdec(cm, plaintext)
30
31 key = "deutsch"
32 cm.set_key(key)
33 plaintext = "howstuffworks"
34 encdec(cm, plaintext)

```

## 2.3 Affine

**class** secretpy.**Affine**

The Affine Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.3.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Affine

```

(continues on next page)

(continued from previous page)

```

5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = [7, 8]
10
11  cipher = Affine()
12  print(plaintext)
13
14  enc = cipher.encrypt(plaintext, key, alphabet)
15  print(enc)
16  dec = cipher.decrypt(enc, key, alphabet)
17  print(dec)
18
19  #####
20
21  print("-----")
22
23  key = [3, 4]
24  plaintext = u"attackatdawn"
25
26  # use default english alphabet
27  print(plaintext)
28  enc = cipher.encrypt(plaintext, key)
29  print(enc)
30  dec = cipher.decrypt(enc, key)
31  print(dec)
32
33  """
34  thequickbrownfoxjumpsoverthelazydog
35  vögaüewsphqmjntlücxoqfghvögzidäßqu
36  thequickbrownfoxjumpsoverthelazydog
37  -----
38  attackatdawn
39  ejjekiejnesr
40  attackatdawn
41  """

```

## 2.4 Atbash

**class** secretpy.**Atbash**

The Atbash Cipher

**decrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)  
Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.4.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Atbash
5  from secretpy import CryptMachine
6  from secretpy import alphabets
7  import secretpy.cmdecorators as md
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16     print("-----")
17
18
19 cm = CryptMachine(Atbash())
20 cm = md.NoSpaces(md.UpperCase(cm))
21
22 plaintext = u"attackatdawn"
23 encdec(cm, plaintext)
24
25 plaintext = u"
26 cm.set_alphabet(alphabets.HEBREW)
27 encdec(cm, plaintext)
28
29 plaintext = u"The Fox jumps in Zoo too Achtung minen"
30 cm.set_alphabet(alphabets.GERMAN)
31 encdec(cm, plaintext)
32
33 plaintext = u"Achtung Minen"
34 encdec(cm, plaintext)
35
36 cm.set_alphabet(alphabets.ARABIC)
37 plaintext = u"
38 encdec(cm, plaintext)
```



## 2.5 Autokey

**class** `secretpy.Autokey`

The Autokey Cipher

**decrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

### 2.5.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Autokey
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = "queenly"
10
11 cipher = Autokey()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")

```

(continues on next page)

(continued from previous page)

```
22
23 plaintext = u"attackatdawn"
24
25 # use default english alphabet
26 print(plaintext)
27 enc = cipher.encrypt(plaintext, key)
28 print(enc)
29 dec = cipher.decrypt(enc, key)
30 print(dec)
31
32 '''
33 thequickbrownfoxjumpsoverthelazydog
34 föiudtäßivamvhyyäeeüxüonhbwzvbßlwk
35 thequickbrownfoxjumpsoverthelazydog
36 -----
37 attackatdawn
38 qnxepvytwtwp
39 attackatdawn
40 '''
```

## 2.6 Bazeries

**class** secretpy.**Bazeries**

The Bazeries Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.6.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Bazeries
5  from secretpy import CryptMachine
6  from secretpy import alphabets
7  from secretpy.cmdecorators import NoSpaces, UpperCase
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16     print("-----")
17
18
19 alphabet = alphabets.ENGLISH_SQUARE_IJ
20
21 key = (81257, u"eightyonethousandtwohundredfiftyseven")
22
23 cm = NoSpaces(UpperCase(CryptMachine(Bazeries())) )
24
25 cm.set_alphabet(alphabet)
26 cm.set_key(key)
27 plaintext = u"Whoever has made a voyage up the Hudson" \
28             u" must remember the Kaatskill mountains"
29 encdec(cm, plaintext)
30
31 '''
32 Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
33 DUMTMCDSENRTMVEQXMOELCCRVXDMDKWNNMUKRDKUMYNMBPRKEEPMGNGEKWXCROWB
34 WHOEVERHASMADEAVOYAGEUPTHEHUDSONMUSTREMEMBERTHEKAATSKILLMOUNTAINS
35 -----
36 '''

```

## 2.7 Beaufort

**class** secretpy.Beaufort

The Beaufort Cipher

**decrypt** (text, key, alphabet='abcdefghijklmnopqrstuvwxyz')

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.7.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import CryptMachine
5  from secretpy import Beaufort
6  from secretpy import alphabets
7
8  plaintext = u"helloworld"
9  key = "key"
10
11 cm = CryptMachine(Beaufort(), key)
12
13 print(plaintext)
14 enc = cm.encrypt(plaintext)
15 print(enc)
16 dec = cm.decrypt(enc)
17 print(dec)
18
19 print("-----")
20
21 alphabet = alphabets.GERMAN
22 cm.set_alphabet(alphabet)
23
24 print(plaintext)
25 enc = cm.encrypt(plaintext)
26 print(enc)
27 dec = cm.decrypt(enc)
28 print(dec)
29
30 '''
31 helloworld
32 danzqcwnnh
33 helloworld
34 -----
35 helloworld
36 danßucärnh
37 helloworld
38 '''
```

## 2.8 Bifid

**class** secretpy.Bifid

The Bifid Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.8.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Bifid
5  from secretpy import CryptMachine
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16 key = 5
17 cm = CryptMachine(Bifid(), key)
18 alphabet = [
19     u" ", u" ", u" ", u" ", u" ", u" ",
20     u" ", u" ", u" ", u" ", u" ", u" ",
21     u" ", u" ", u" ", u" ", u" ", u" ",

```

(continues on next page)

(continued from previous page)

```

22     u"", u"", u"", u"", u"", u"",
23     u"", u"", u"", u"", u"", u"",
24     u"1", u"2", u"3", u"4", u"5", u"6"
25 ]
26
27 cm.set_alphabet(alphabet)
28 plaintext = u""
29 encdec(cm, plaintext)
30
31 alphabet = [
32     u"p", u"h", u"q", u"g", u"m",
33     u"e", u"a", u"y", u"l", u"n",
34     u"o", u"f", u"d", u"x", u"k",
35     u"r", u"c", u"v", u"s", u"z",
36     u"w", u"b", u"u", u"t", u"ij"
37 ]
38 plaintext = u"defendtheeastwallofthecastle"
39 cm.set_alphabet(alphabet)
40 encdec(cm, plaintext)
41
42 alphabet = [
43     u"b", u"g", u"w", u"k", u"z",
44     u"q", u"p", u"n", u"d", u"s",
45     u"ij", u"o", u"a", u"x", u"e",
46     u"f", u"c", u"l", u"u", u"m",
47     u"t", u"h", u"y", u"v", u"r"
48 ]
49 plaintext = "fleeatonce"
50 cm.set_alphabet(alphabet)
51 cm.set_key(10)
52 encdec(cm, plaintext)
53
54 alphabet = [
55     u"A", u"B", u"Γ", u"Δ", u"E",
56     u"Z", u"H", u"Θ", u"I", u"K",
57     u"Λ", u"M", u"N", u"Ξ", u"O",
58     u"Π", u"P", u"Σ", u"Τ", u"Υ",
59     u"Φ", u"Χ", u"Ψ", u"Ω"
60 ]
61 plaintext = u"ΠΙΝΑΚΑΣ"
62 cm.set_alphabet(alphabet)
63 encdec(cm, plaintext)

```

## 2.9 Caesar

**class** secretpy.**Caesar**

The Caesar Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key

- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** decrypted text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** encrypted text

**Return type** string

## 2.9.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Caesar
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = 3
10 cipher = Caesar()
11
12 print(plaintext)
13 enc = cipher.encrypt(plaintext, key, alphabet)
14 print(enc)
15 dec = cipher.decrypt(enc, key, alphabet)
16 print(dec)
17
18 print('=====')
19
20 print(plaintext)
21 # use default english alphabet
22 enc = cipher.encrypt(plaintext, key)
23 print(enc)
24 dec = cipher.decrypt(enc, key)
25 print(dec)
26
27 '''
28 Output:
29
30 thequickbrownfoxjumpsoverthelazydog
31 wkhtxlfneurzqirämxpsvryhuwkhodüögrj
32 thequickbrownfoxjumpsoverthelazydog
33 =====
34 thequickbrownfoxjumpsoverthelazydog

```

(continues on next page)

(continued from previous page)

```

35 wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
36 thequickbrownfoxjumpsoverthelazydog
37 '''

```

## 2.10 Caesar Progressive

**class** `secretpy.CaesarProgressive`

The Caesar Progressive Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** decrypted text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** encrypted text

**Return type** string

### 2.10.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import CaesarProgressive
5  from secretpy import alphabets
6
7  alphabet = alphabets.ENGLISH
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = 3
10 cipher = CaesarProgressive()
11
12 print(plaintext)
13 enc = cipher.encrypt(plaintext, key, alphabet)
14 print(enc)

```

(continues on next page)



(continued from previous page)

```

15 dec = cipher.decrypt(enc, key, alphabet)
16 print(dec)
17
18 print('=====')
19
20 print(plaintext)
21 # use default english alphabet
22 enc = cipher.encrypt(plaintext, key)
23 print(enc)
24 dec = cipher.decrypt(enc, key)
25 print(dec)
26
27 '''
28 thequickbrownfoxjumpsoverthelazydog
29 wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
30 thequickbrownfoxjumpsoverthelazydog
31 =====
32 thequickbrownfoxjumpsoverthelazydog
33 wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
34 thequickbrownfoxjumpsoverthelazydog
35 '''

```

## 2.11 Columnar Transposition

**class** secretpy.ColumnarTransposition

The Columnar Transposition Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.11.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ColumnarTransposition, CryptMachine
5  from secretpy import alphabets
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = "cargo"
17  cm = CryptMachine(ColumnarTransposition(), key)
18
19  alphabet = alphabets.ENGLISH
20
21  cm.set_alphabet(alphabet)
22  plaintext = "attackatdawn"
23  encdec(cm, plaintext)
24
25  key = "deutsch"
26  cm.set_key(key)
27  plaintext = "howstuffworks"
28  encdec(cm, plaintext)
29
30  '''
31  attackatdawn
32  tanakwadzcaztz
33  attackatdawnzzz
34  -----
35  howstuffworks
36  ushfowfztksrwo
37  howstuffworksz
38  -----
39  '''
```

## 2.12 Four Square

**class** secretpy.FourSquare

The Four-Square Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key

- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.12.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import FourSquare
5  from secretpy import CryptMachine
6  from secretpy import alphabets
7  from secretpy.cmdecorators import NoSpaces, UpperCase
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16     print("-----")
17
18
19 alphabet = alphabets.ENGLISH_SQUARE_OQ
20
21 key = (u"exampl", u"keyword")
22
23 cm = NoSpaces(UpperCase(CryptMachine(FourSquare())))
24
25 cm.set_alphabet(alphabet)
26 cm.set_key(key)
27 plaintext = u"Help me Obi wan Kenobi"
28 encdec(cm, plaintext)
29
30 alphabet = alphabets.ENGLISH_SQUARE_IJ
31 cm.set_alphabet(alphabet)
32 key = (u"criptog", u"segurt")
33 cm.set_key(key)
34 plaintext = u"attack at dawn"

```

(continues on next page)

(continued from previous page)

```

35 encdec(cm, plaintext)
36
37 '''
38 Help me Obi wan Kenobi
39 FYGMKYHOBXMFKKKIMD
40 HELPMEOBIWANKENOBI
41 -----
42 attack at dawn
43 PMMUTBPMCUXH
44 ATTACKATDAWN
45 -----
46 '''

```

## 2.13 Gronsfeld

**class** secretpy.Gronsfeld

The Gronsfeld Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.13.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3
4 from secretpy import Gronsfeld
5 from secretpy import alphabets

```

(continues on next page)

(continued from previous page)

```

6
7 alphabet = alphabets.GERMAN
8 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9 key = (4, 7, 9)
10
11 cipher = Gronsfeld()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
22
23 plaintext = u"attackatdawn"
24 key = (14, 2, 11)
25
26 print(plaintext)
27 enc = cipher.encrypt(plaintext, key)
28 print(enc)
29 dec = cipher.decrypt(enc, key)
30 print(dec)
31
32 '''
33 thequickbrownfoxjumpsoverthelazydog
34 xonuögrkvvbrmxöqßqwösünväqisjßbmsn
35 thequickbrownfoxjumpsoverthelazydog
36 -----
37 attackatdawn
38 oveoevovooyy
39 attackatdawn
40 '''

```

## 2.14 Keyword

**class** secretpy.**Keyword**

The Keyword Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.14.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Keyword
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = "queenly"
10
11 cipher = Keyword();
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
22
23 plaintext = u"thisisasecretmessage"
24 key = "keyword"
25
26 # use default english alphabet
27 print(plaintext)
28 enc = cipher.encrypt(plaintext, key)
29 print(enc)
30 dec = cipher.decrypt(enc, key)
31 print(dec)
```

## 2.15 Nihilist

**class** secretpy.**Nihilist**

The Nihilist Cipher

**decrypt** (*text*, *key*=*None*, *alphabet*=*None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string

**encrypt** (*text, key=None, alphabet=None*)  
Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string

## 2.15.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Nihilist
5  from secretpy import CryptMachine
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = "russian"
17  cm = CryptMachine(Nihilist(), key)
18  alphabet = [
19      u"z", u"e", u"b", u"r", u"a",
20      u"s", u"c", u"d", u"f", u"g",
21      u"h", u"ij", u"k", u"l", u"m",
22      u"n", u"o", u"p", u"q", u"t",
23      u"u", u"v", u"w", u"x", u"y"
24  ]
25  plaintext = u"dynamitewinterpalace"
26  cm.set_alphabet(alphabet)
27  encdec(cm, plaintext)
28
29  alphabet = [

```

(continues on next page)

(continued from previous page)

```

30     u"a", u"b", u"c", u"d", u"e", u"f",
31     u"g", u"h", u"i", u"j", u"k", u"l",
32     u"m", u"n", u"o", u"p", u"q", u"r",
33     u"s", u"t", u"u", u"v", u"w", u"x",
34     u"y", u"z", u"0", u"1", u"2", u"3",
35     u"4", u"5", u"6", u"7", u"8", u"9",
36 ]
37 key = "freedom"
38 plaintext = u"meetthursday2300hr"
39 cm.set_alphabet(alphabet)
40 cm.set_key(key)
41 encdec(cm, plaintext)
42
43 alphabet = [
44     u"", u"", u"", u"", u"", u"",
45     u"", u"", u"", u"", u"", u"",
46     u"", u"", u"", u"", u"", u"",
47     u"", u"", u"", u"", u"", u"",
48     u"", u"", u"", u"", u"", u"",
49     u"1", u"2", u"3", u"4", u"5", u"6"
50 ]
51
52 cm.set_alphabet(alphabet)
53 key = u""
54 plaintext = u""
55 encdec(cm, plaintext)
56
57 alphabet = [
58     u"A", u"B", u"Γ", u"Δ", u"E",
59     u"Ζ", u"H", u"Θ", u"I", u"K",
60     u"Λ", u"M", u"N", u"Ξ", u"O",
61     u"Π", u"P", u"Σ", u"Τ", u"Υ",
62     u"Φ", u"Χ", u"Ψ", u"Ω"
63 ]
64 plaintext = u"ΠΙΝΑΚΑΣ"
65 cm.set_alphabet(alphabet)
66 encdec(cm, plaintext)
67
68 '''
69 Output:
70
71 dynamitewinterpalace
72 37 106 62 36 67 47 86 26 104 53 62 77 27 55 57 66 55 36 54 27
73 dynamitewinterpalace
74 -----
75 meetthursday2300hr
76 47 51 30 57 56 55 74 52 77 29 26 65 88 87 69 89 37 51
77 meetthursday2300hr
78 -----
79
80 102 82 90 101 102
81
82 -----
83 ΠΙΝΑΚΑΣ
84 95 78 87 65 79 65 97
85 ΠΙΝΑΚΑΣ
86 -----

```

(continues on next page)



(continued from previous page)

87 ' ' '

## 2.16 Playfair

**class** secretpy.**Playfair**

The Playfair Cipher

**decrypt** (*text*, *key*=", *alphabet*=None)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=", *alphabet*=None)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.16.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Playfair
5  from secretpy import CryptMachine
6  from secretpy.cmdecorators import NoSpaces, UpperCase
7
8
9  def encdec(machine, plaintext):
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15     print("-----")
16

```

(continues on next page)

(continued from previous page)

```

17 cm = NoSpaces(UpperCase(CryptMachine(Playfair()))))
18 alphabet = [
19     u"p", u"l", u"a", u"y", u"f",
20     u"i", u"r", u"e", u"x", u"m",
21     u"b", u"c", u"d", u"g", u"h",
22     u"k", u"n", u"o", u"q", u"s",
23     u"t", u"u", u"v", u"w", u"z",
24 ]
25 cm.set_alphabet(alphabet)
26 plaintext = u"Hide the gold in the tree stump"
27 encdec(cm, plaintext)
28
29 plaintext = "sometext"
30 encdec(cm, plaintext)
31
32 plaintext = "this is a secret message"
33 encdec(cm, plaintext)
34

```

## 2.17 Polybius

**class** secretpy.Polybius

The Polybius Cipher

**decrypt** (*text*, *key*=", *alphabet*=None)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=", *alphabet*=None)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.17.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Polybius
5  from secretpy import CryptMachine
6  from secretpy.cmdecorators import LowerCase
7  import secretpy.alphabets as alph
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16     print("-----")
17
18
19 cm = CryptMachine(Polybius())
20
21 plaintext = u"defendtheeastwallofthecastle"
22 encdec(cm, plaintext)
23
24 alphabet = [
25     u"p", u"h", u"q", u"g", u"m",
26     u"e", u"a", u"y", u"l", u"n",
27     u"o", u"f", u"d", u"x", u"k",
28     u"r", u"c", u"v", u"s", u"z",
29     u"w", u"b", u"u", u"t", u"ij"
30 ]
31 cm.set_alphabet(alphabet)
32 plaintext = "sometext"
33 encdec(cm, plaintext)
34
35 plaintext = "thisisasecretmessage"
36 encdec(cm, plaintext)
37
38 cm.set_alphabet(alph.GREEK)
39 plaintext = u"ΠΙΝΑΚΑΣ"
40 cm = LowerCase(cm)
41 encdec(cm, plaintext)

```

## 2.18 Porta

**class** secretpy.Porta

The Porta Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key

- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.18.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Porta
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = u"dogs"
10
11 cipher = Porta()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
22
23 plaintext = u"attackatdawn"
24 key = u"lemon"
25
26 print(plaintext)
27 enc = cipher.encrypt(plaintext, key)
28 print(enc)
29 dec = cipher.decrypt(enc, key)
30 print(dec)
31
32 '''
33 thequickbrownfoxjumpsoverthelazydog
34 dßwheputrkrnßöroznpgcvdübmzüöwhatvy
```

(continues on next page)

(continued from previous page)

```

35 thequickbrownfoxjumpsoverthelazydog
36 -----
37 attackatdawn
38 seauvppaxtel
39 attackatdawn
40 '''

```

## 2.19 Rot13

**class** secretpy.**Rot13**

The Rot13 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

**Return type** *string*

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

**Return type** *string*

### 2.19.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot13
5  from secretpy import CryptMachine
6  from secretpy.cmdecorators import SaveCase, SaveSpaces
7  from secretpy import alphabets
8
9
10 def encdec(machine, plaintext):
11     print("-----")

```

(continues on next page)

(continued from previous page)

```
12     print(plaintext)
13     enc = machine.encrypt(plaintext)
14     print(enc)
15     dec = machine.decrypt(enc)
16     print(dec)
17
18
19 cm = SaveCase(CryptMachine(Rot13()))
20
21 plaintext = u"thisisasecretmessage"
22 encdec(cm, plaintext)
23
24 cm = SaveSpaces(cm)
25
26 plaintext = u"Why did the chicken cross the road Gb trg gb gur bgure fvqr"
27 encdec(cm, plaintext)
28
29 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
30 cm.set_alphabet(alphabets.GERMAN)
31 encdec(cm, plaintext)
32
33 plaintext = u""
34 cm.set_alphabet(alphabets.RUSSIAN)
35 encdec(cm, plaintext)
```

## 2.20 Rot5

**class** secretpy.Rot5

The Rot5 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.20.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot5
5  from secretpy import alphabets
6  from secretpy import CryptMachine
7
8
9  def encdec(machine, plaintext):
10     print("-----")
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16
17
18  cm = CryptMachine(Rot5())
19
20  plaintext = alphabets.DECIMAL
21  encdec(cm, plaintext)
22  '''
23  -----
24  0123456789
25  5678901234
26  0123456789
27  '''

```

## 2.21 Rot18

**class** secretpy.Rot18

The Rot18 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.21.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot18
5  from secretpy import CryptMachine
6  from secretpy.cmdecorators import SaveCase, SaveSpaces, UpperCase
7  from secretpy import alphabets
8
9
10 def encdec(machine, plaintext):
11     print("-----")
12     print(plaintext)
13     enc = machine.encrypt(plaintext)
14     print(enc)
15     dec = machine.decrypt(enc)
16     print(dec)
17
18
19 cm = SaveCase(SaveSpaces(CryptMachine(Rot18()))))
20
21 plaintext = u"The man has 536 dogs"
22 encdec(cm, plaintext)
23
24 plaintext = alphabets.RUSSIAN + alphabets.DECIMAL
25 cm.set_alphabet(alphabets.RUSSIAN)
26 encdec(cm, plaintext)
27
28 plaintext = u" 536 "
29 encdec(cm, plaintext)
30
31 plaintext = alphabets.GREEK + " " + alphabets.DECIMAL
32 cm = UpperCase(cm)
33 cm.set_alphabet(alphabets.GREEK)
34 encdec(cm, plaintext)
```

## 2.22 Rot47

**class** secretpy.**Rot47**

The Rot47 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**



- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.22.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot47
5  from secretpy import CryptMachine
6  from secretpy.cmdecorators import SaveSpaces
7
8
9  def encdec(machine, plaintext):
10     print("-----")
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16
17
18  cm = SaveSpaces(CryptMachine(Rot47()))
19
20  plaintext = u"The man has 536 dogs"
21  encdec(cm, plaintext)

```

## 2.23 Simple Substitution

**class** secretpy.SimpleSubstitution

The Simple Substitution Cipher

**decrypt** (*text*, *key*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string**encrypt** (*text, key, alphabet='abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string

## 2.23.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import SimpleSubstitution
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = u"dabcghijokzlmnpqrstuvfwxyäöeüß"
10
11 cipher = SimpleSubstitution()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
22
23 plaintext = u"thisisasecretmessage"
24 alphabet = alphabets.ENGLISH
25 key = u"dabcghijokzlmnpqrstuvfwxye"
26
27 print(plaintext)
28 enc = cipher.encrypt(plaintext, key, alphabet)
29 print(enc)
```

(continues on next page)

(continued from previous page)

```

30 dec = cipher.decrypt(enc, key, alphabet)
31 print(dec)
32
33 '''
34 thequickbrownfoxjumpsoverthelazydog
35 ujgrvobzaspwnhpxkvmqtpfgsu jgldäycpi
36 thequickbrownfoxjumpsoverthelazydog
37 -----
38 thisisasecretmessage
39 u jototdtgbsgumgttdig
40 thisisasecretmessage
41 '''

```

## 2.24 Three Square

**class** secretpy.**ThreeSquare**

The Three Square Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.24.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3
4 from secretpy import ThreeSquare
5 from secretpy import CryptMachine

```

(continues on next page)

(continued from previous page)

```

6  from secretpy import alphabets
7  from secretpy.cmdecorators import NoSpaces, UpperCase
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)
15     print(dec)
16     print("-----")
17
18
19 alphabet = alphabets.ENGLISH_SQUARE_OQ
20 key = (u"example", u"keyword", u"third")
21 cm = NoSpaces(UpperCase(CryptMachine(ThreeSquare())) )
22 cm.set_alphabet(alphabet)
23 cm.set_key(key)
24 plaintext = u"Help me Obi wan Kenobi"
25 encdec(cm, plaintext)
26
27 alphabet = alphabets.ENGLISH_SQUARE_IJ
28 cm.set_alphabet(alphabet)
29 key = (u"criptog", u"segurt", u"mars")
30 cm.set_key(key)
31 plaintext = u"attack at dawn"
32 encdec(cm, plaintext)
33
34 '''
35 Help me Obi wan Kenobi
36 HJKNEMDHOHSACL YRISFJKUUKBEF
37 HELPMEOBIWANKENOBI
38 -----
39 attack at dawn
40 QCTZABCSKXCATDAFWN
41 ATTACKATDAWN
42 -----
43 '''

```

## 2.25 Trifid

**class** secretpy.Trifid

The Trifid Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.25.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Trifid
5  from secretpy import CryptMachine
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = 5
17  cm = CryptMachine(Trifid(), key)
18
19  alphabet = [
20      u"e", u"p", u"s",
21      u"d", u"u", u"c",
22      u"v", u"w", u"y",
23
24      u"m", u".", u"z",
25      u"l", u"k", u"x",
26      u"n", u"b", u"t",
27
28      u"f", u"g", u"o",
29      u"r", u"i", u"j",
30      u"h", u"a", u"q",
31  ]
32
33  plaintext = u"defendtheeastwallofthecastle"
34  cm.set_alphabet(alphabet)
35  encdec(cm, plaintext)
36
37  '''
38  defendtheeastwallofthecastle
39  suefecphsegyyjiximfofocejlr

```

(continues on next page)

(continued from previous page)

```

40 defendtheeastwallofthecastle
41 -----
42 '''

```

## 2.26 Two Square

**class** `secretpy.TwoSquare`

The Two-Square Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

### 2.26.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import TwoSquare
5  from secretpy import CryptMachine
6  from secretpy import alphabets
7  from secretpy.cmdecorators import NoSpaces, UpperCase
8
9
10 def encdec(machine, plaintext):
11     print(plaintext)
12     enc = machine.encrypt(plaintext)
13     print(enc)
14     dec = machine.decrypt(enc)

```

(continues on next page)

(continued from previous page)

```

15     print(dec)
16     print("-----")
17
18
19 alphabet = alphabets.ENGLISH_SQUARE_OO
20
21 key = (u"example", u"keyword")
22
23 cm = NoSpaces(UpperCase(CryptMachine(TwoSquare()))))
24
25 cm.set_alphabet(alphabet)
26 cm.set_key(key)
27 plaintext = u"Help me Obi wan Kenobi"
28 encdec(cm, plaintext)
29
30 '''
31 Help me Obi wan Kenobi
32 XGDLXWSDJYRYHOTKDG
33 HELPMEOWIWANKENOBI
34 -----
35 '''

```

## 2.27 Vic

**class** secretpy.Vic

The Vic Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.27.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Vic
5  from secretpy import CryptMachine
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = "0452"
17  cm = CryptMachine(Vic(), key)
18  alphabet = [
19      u"e", u"t", u" ", u"a", u"o", u"n", u" ", u"r", u"i", u"s",
20      u"b", u"c", u"d", u"f", u"g", u"h", u"j", u"k", u"l", u"m",
21      u"p", u"q", u"/", u"u", u"v", u"w", u"x", u"y", u"z", u".",
22  ]
23  plaintext = u"attackatdawn"
24  cm.set_alphabet(alphabet)
25  encdec(cm, plaintext)
26
27  '''
28  Output:
29
30  attackatdawn
31  anwhrsanroaeer
32  attackatdawn
33  -----
34  '''
```

## 2.28 Vigenere

**class** secretpy.Vigenere

The Vigenere Cipher

**decrypt** (text, key, alphabet='abcdefghijklmnopqrstuvwxyz')

Decryption method

#### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string



**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

#### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.28.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Vigenere
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = u"kss"
10
11 cipher = Vigenere()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
22
23 plaintext = u"attackatdawn"
24 key = u"lemon"
25
26 print(plaintext)
27 enc = cipher.encrypt(plaintext, key)
28 print(enc)
29 dec = cipher.decrypt(enc, key)
30 print(dec)
31
32 '''
33 thequickbrownfoxjumpsoverthelazydog
34 ßzwäïämütöckxxcdöiwdgyjwöhzoßsfmvy
35 thequickbrownfoxjumpsoverthelazydog
36 -----
37 attackatdawn
38 lxfopefrnhr
39 attackatdawn
40 '''

```

## 2.29 Zigzag

**class** `secretpy.Zigzag`

The Zigzag Cipher

**decrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

### 2.29.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Zigzag
5  from secretpy import alphabets
6
7  alphabet = alphabets.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = 3
10
11 chipher = Zigzag()
12 print(plaintext)
13
14 enc = chipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = chipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21 print("-----")
```

(continues on next page)

(continued from previous page)

```

22
23 plaintext = u"wearediscoveredfleeatonce"
24
25 print(plaintext)
26 enc = cipher.encrypt(plaintext, key)
27 print(enc)
28 dec = cipher.decrypt(enc, key)
29 print(dec)
30
31 #####
32
33 print("-----")
34
35 plaintext = u"defendtheeastwallofthecastle"
36 key = 4
37
38 print(plaintext)
39 enc = cipher.encrypt(plaintext, key)
40 print(enc)
41 dec = cipher.decrypt(enc, key)
42 print(dec)
43
44 '''
45 thequickbrownfoxjumpsoverthelazydog
46 tubnjsrldhqikrwxupoeteayoecoomvhzg
47 thequickbrownfoxjumpsoverthelazydog
48 -----
49 wearediscoveredfleeatonce
50 wecrlteerdsoeefaocaivden
51 wearediscoveredfleeatonce
52 -----
53 defendtheeastwallofthecastle
54 dttfsedhswotatfneaalhcleeele
55 defendtheeastwallofthecastle
56 '''

```



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**A**

ADFGVX (class in *secretpy*), 9  
ADFGX (class in *secretpy*), 7  
Affine (class in *secretpy*), 10  
Atbash (class in *secretpy*), 11  
Autokey (class in *secretpy*), 13

**B**

Bazeries (class in *secretpy*), 14  
Beaufort (class in *secretpy*), 15  
Bifid (class in *secretpy*), 17

**C**

Caesar (class in *secretpy*), 18  
CaesarProgressive (class in *secretpy*), 20  
ColumnarTransposition (class in *secretpy*), 21

**D**

decrypt () (*secretpy.ADFGVX* method), 9  
decrypt () (*secretpy.ADFGX* method), 7  
decrypt () (*secretpy.Affine* method), 10  
decrypt () (*secretpy.Atbash* method), 11  
decrypt () (*secretpy.Autokey* method), 13  
decrypt () (*secretpy.Bazeries* method), 14  
decrypt () (*secretpy.Beaufort* method), 15  
decrypt () (*secretpy.Bifid* method), 17  
decrypt () (*secretpy.Caesar* method), 18  
decrypt () (*secretpy.CaesarProgressive* method), 20  
decrypt () (*secretpy.ColumnarTransposition* method), 21  
decrypt () (*secretpy.FourSquare* method), 22  
decrypt () (*secretpy.Gronsfeld* method), 24  
decrypt () (*secretpy.Keyword* method), 25  
decrypt () (*secretpy.Nihilist* method), 26  
decrypt () (*secretpy.Playfair* method), 29  
decrypt () (*secretpy.Polybius* method), 30  
decrypt () (*secretpy.Porta* method), 31  
decrypt () (*secretpy.Rot13* method), 33  
decrypt () (*secretpy.Rot18* method), 35

decrypt () (*secretpy.Rot47* method), 36  
decrypt () (*secretpy.Rot5* method), 34  
decrypt () (*secretpy.SimpleSubstitution* method), 37  
decrypt () (*secretpy.ThreeSquare* method), 39  
decrypt () (*secretpy.Trifid* method), 40  
decrypt () (*secretpy.TwoSquare* method), 42  
decrypt () (*secretpy.Vic* method), 43  
decrypt () (*secretpy.Vigenere* method), 44  
decrypt () (*secretpy.Zigzag* method), 46

**E**

encrypt () (*secretpy.ADFGVX* method), 9  
encrypt () (*secretpy.ADFGX* method), 7  
encrypt () (*secretpy.Affine* method), 10  
encrypt () (*secretpy.Atbash* method), 12  
encrypt () (*secretpy.Autokey* method), 13  
encrypt () (*secretpy.Bazeries* method), 14  
encrypt () (*secretpy.Beaufort* method), 16  
encrypt () (*secretpy.Bifid* method), 17  
encrypt () (*secretpy.Caesar* method), 19  
encrypt () (*secretpy.CaesarProgressive* method), 20  
encrypt () (*secretpy.ColumnarTransposition* method), 21  
encrypt () (*secretpy.FourSquare* method), 23  
encrypt () (*secretpy.Gronsfeld* method), 24  
encrypt () (*secretpy.Keyword* method), 25  
encrypt () (*secretpy.Nihilist* method), 27  
encrypt () (*secretpy.Playfair* method), 29  
encrypt () (*secretpy.Polybius* method), 30  
encrypt () (*secretpy.Porta* method), 32  
encrypt () (*secretpy.Rot13* method), 33  
encrypt () (*secretpy.Rot18* method), 35  
encrypt () (*secretpy.Rot47* method), 37  
encrypt () (*secretpy.Rot5* method), 34  
encrypt () (*secretpy.SimpleSubstitution* method), 38  
encrypt () (*secretpy.ThreeSquare* method), 39  
encrypt () (*secretpy.Trifid* method), 41  
encrypt () (*secretpy.TwoSquare* method), 42  
encrypt () (*secretpy.Vic* method), 43  
encrypt () (*secretpy.Vigenere* method), 44

`encrypt()` (*secretpy.Zigzag method*), 46

## F

`FourSquare` (*class in secretpy*), 22

## G

`Gronsfeld` (*class in secretpy*), 24

## K

`Keyword` (*class in secretpy*), 25

## N

`Nihilist` (*class in secretpy*), 26

## P

`Playfair` (*class in secretpy*), 29

`Polybius` (*class in secretpy*), 30

`Porta` (*class in secretpy*), 31

## R

`Rot13` (*class in secretpy*), 33

`Rot18` (*class in secretpy*), 35

`Rot47` (*class in secretpy*), 36

`Rot5` (*class in secretpy*), 34

## S

`SimpleSubstitution` (*class in secretpy*), 37

## T

`ThreeSquare` (*class in secretpy*), 39

`Trifid` (*class in secretpy*), 40

`TwoSquare` (*class in secretpy*), 42

## V

`Vic` (*class in secretpy*), 43

`Vigenere` (*class in secretpy*), 44

## Z

`Zigzag` (*class in secretpy*), 46