
SecretPy Documentation

Release 1.0

[Read the Docs](#)

Jul 27, 2021

Contents:

1	SecretPy	1
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
2	API	7
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	18
2.8.1	Examples	18
2.9	Caesar	20
2.9.1	Examples	20
2.10	Caesar Progressive	22
2.10.1	Examples	23
2.11	Chao	24
2.11.1	Examples	25
2.12	Columnar Transposition	26
2.12.1	Examples	26
2.13	Four Square	27
2.13.1	Examples	28

2.14	Gronsfeld	29
2.14.1	Examples	29
2.15	Keyword	30
2.15.1	Examples	31
2.16	MyszkowskiTransposition	31
2.16.1	Examples	32
2.17	Nihilist	33
2.17.1	Examples	33
2.18	Playfair	35
2.18.1	Examples	35
2.19	Polybius	36
2.19.1	Examples	37
2.20	Porta	38
2.20.1	Examples	38
2.21	Rot13	39
2.21.1	Examples	40
2.22	Rot5	41
2.22.1	Examples	41
2.23	Rot18	42
2.23.1	Examples	43
2.24	Rot47	43
2.24.1	Examples	44
2.25	Simple Substitution	44
2.25.1	Examples	45
2.26	Simple Substitution	46
2.26.1	Examples	47
2.27	Three Square	47
2.27.1	Examples	48
2.28	Trifid	49
2.28.1	Examples	49
2.29	Two Square	51
2.29.1	Examples	51
2.30	Vic	52
2.30.1	Examples	53
2.31	Vigenere	53
2.31.1	Examples	54
2.32	Zigzag	55
2.32.1	Examples	55
3	Indices and tables	57
	Index	59

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:

- Affine
- Atbash
- Bazeries
- Beaufort
- Caesar, Caesar Progressive
- Chaocipher
- Keyword
- Playfair, Two Square(Double Playfair), Three Square, Four Square
- Polybius, ADFGX, ADFGVX, Bifid, Trifid, Nihilist
- Rot13, Rot5, Rot18, Rot47
- Scytale

- Simple Substitution
- Transposition - Columnar, Myszkowski, Zigzag(Railfence)
- Vic
- Vigenere, Autokey, Gronsfeld, Porta

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 characters which exist in the alphabet, and they don't have a state.

```
from secretpy import Caesar, alphabets as al

def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
    print(
        '=====')
    print(
        plaintext)
    enc = cipher.encrypt(plaintext, key, alphabet)
    print(enc)
    print(cipher.decrypt(enc, key, alphabet))

key = 3
cipher = Caesar()

plaintext = u"thequickbrownfoxjumpsoverthelazydog"
encdec(cipher, plaintext, key)

alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
encdec(cipher, plaintext, key, alphabet)

alphabet = al.SWEDISH
plaintext = u"faqomschweizklövdutrångpjäxby"
encdec(cipher, plaintext, key, alphabet)

'''
```

(continues on next page)

(continued from previous page)

Output:

```

=====
thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
=====
schweißgequältvomödentextzürnttypografjakob
vfkzhlcjhtxßowyrpaghqwhäwübuqwwösrjudimdnre
schweißgequältvomödentextzürnttypografjakob
=====
faqomschweizklövdutrångpjäxby
idtrpvfkzhlöncygxwuaqjsmbåeä
faqomschweizklövdutrångpjäxby
'''

```

1.3.2 CryptMachine

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

```

from secretpy import Caesar, CryptMachine, alphabets as al
from secretpy.cmdecorators import SaveAll, Block

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

cm0 = CryptMachine(cipher, key)

cm = cm0
cm.set_alphabet(al.ENGLISH)
plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
↳(*;?&#. Great!"
encdec(cm, plaintext)

cm = Block(cm, length=5, sep="-")
plaintext = "This text is divided by blocks of length 5!"
encdec(cm, plaintext)

cm = SaveAll(cm0)
plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
encdec(cm, plaintext)

cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

cm.set_alphabet(al.JAPANESE_HIRAGANA)
cm.set_key(1)
plaintext = u"text  "
encdec(cm, plaintext)

'''
Output:

-----
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
lgrqworyhqrqdoskdehwfkdudfwhuvlzlouhpryhdooriwkhpjuhdw
idontlovenonalphabetcharactersiwillremoveallofthemgreat
-----

This text is divided by blocks of length 5!
wklvw-hawlv-glylg-hgebe-orfnv-riohq-jwk
thistextisdividedbyblocksoflength

-----

I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
L oryh qrq-doskdehw fkdudfwhuv. Wkhvh duh : ^,&@$~(*;?&#. Wkdw'v lw!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
-----

Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Mm ehfrphv Mm ehfdxvh zh xvh HQKOMVL_VTXDUH_MM!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
-----

text
text
text
'''

```

1.3.3 CompositeMachine

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

```

from secretpy import Rot13, Caesar, CryptMachine, CompositeMachine
from secretpy.cmdecorators import SaveAll

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

```

(continues on next page)

(continued from previous page)

```

cm2 = SaveAll(CryptMachine(Rot13()))
enc = cm2.encrypt(enc)
print(enc)

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*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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, alphabets

```

(continues on next page)

(continued from previous page)

```

5
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, alphabets
5
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 föiudtäßivamvhyyäeeüxüonhbwzvbßlwvk
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** (*string*) – 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** (*string*) – 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 Beaufort, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====')
11         ↪ ')
12         print(plaintext)
13         enc = cipher.encrypt(plaintext, key, alphabet)
14         print(enc)
15         print(cipher.decrypt(enc, key, alphabet))
16
17 key = "key"
18 cipher = Beaufort()
19
20 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
21 encdec(cipher, plaintext, key)
22
23 alphabet = al.GERMAN
24 plaintext = u"schweißgequältvomödentextzürnttypografjakob"
25 encdec(cipher, plaintext, key, alphabet)
26
27 alphabet = al.SWEDISH
28 plaintext = u"faqomschweizklövdutrångpjäxby"
29 encdec(cipher, plaintext, key, alphabet)
30
31 # using cryptmachine
32
33 def encdec(machine, plaintext):
34     print("-----")
35     print(plaintext)
36     enc = machine.encrypt(plaintext)
37     print(enc)
```

(continues on next page)

(continued from previous page)

```

38     print(machine.decrypt(enc))
39
40
41 cm0 = CryptMachine(cipher, key)
42
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 cm = SaveAll(cm0)
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)
50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 cm.set_key(u"")
53 plaintext = u"text "
54 encdec(cm, plaintext)
55
56 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
57 ↳(*;?&#. Great!"
58 cm = cm0
59 cm.set_alphabet(al.ENGLISH)
60 cm.set_key(key)
61 encdec(cm, plaintext)
62
63 '''
64 Output:
65
66 =====
67 thequickbrownfoxjumpsoverthelazydog
68 rxuukqiuxtqcxzknveypgwjutlrgtylgvwy
69 thequickbrownfoxjumpsoverthelazydog
70 =====
71 schweißgequältvomödentextzürnttypografjakob
72 wcrsaqlüuyöüßpdäwöhalvabvjäxvfvkjäühkßpkykj
73 schweißgequältvomödentextzürnttypografjakob
74 =====
75 faqomschweizklövdutrångpjäxby
76 feizvgiåcgzöawzsbeuqäääjbgbjj
77 faqomschweizklövdutrångpjäxby
78 -----
79 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
80 C tkpa lwr-yzprkdur crknyilutm. Fdagg ehg : ^,&@$~(*;?&#. Lrkl'g cl!
81 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
82 -----
83 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
84 Bw xfclyag Bw xfcyqnu oa esa UXYOBNR_SPEKOU_BW!
85 Ii becomes Ii because we use ENGLISH_SQUARE_II!
86 -----
87 text
88 text
89 text
90 -----
91 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
92 cbkxlnwjuxqlktjdexglwdehkcfcgngciqzthgskpayztkflrgsstayr
93 idontlovenonalphabetcharactersiwillremoveallofthemgreat
94 '''

```

2.8 Bifid

class secretpy.Bifid

The Bifid Cipher

decrypt (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)
Decryption method

Parameters

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

Returns text

Return type string

encrypt (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)
Encryption method

Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string or tuple or list*) – 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, CryptMachine, alphabets
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 = 5
16  cm = CryptMachine(Bifid(), key)
17  alphabet = [
18      u"", u"", u"", u"", u"", u"",
19      u"", u"", u"", u"", u"", u""]

```

(continues on next page)

(continued from previous page)

```

20     u"", u"", u"", u"", u"", u"",
21     u"", u"", u"", u"", u"", u"",
22     u"", u"", u"", u"", u"", u"",
23     u"1", u"2", u"3", u"4", u"5", u"6"
24 ]
25 cm.set_alphabet(alphabet)
26 plaintext = u""
27 encdec(cm, plaintext)
28
29 alphabet = [
30     u"p", u"h", u"q", u"g", u"m",
31     u"e", u"a", u"y", u"l", u"n",
32     u"o", u"f", u"d", u"x", u"k",
33     u"r", u"c", u"v", u"s", u"z",
34     u"w", u"b", u"u", u"t", u"ij"
35 ]
36 cm.set_alphabet(alphabet)
37 plaintext = u"defendtheeastwallofthecastle"
38 encdec(cm, plaintext)
39
40 alphabet = [
41     u"b", u"g", u"w", u"k", u"z",
42     u"q", u"p", u"n", u"d", u"s",
43     u"ij", u"o", u"a", u"x", u"e",
44     u"f", u"c", u"l", u"u", u"m",
45     u"t", u"h", u"y", u"v", u"r"
46 ]
47 cm.set_alphabet(alphabet)
48 cm.set_key(10)
49 plaintext = "fleeatonce"
50 encdec(cm, plaintext)
51
52 alphabet = alphabets.GREEK.upper()
53 plaintext = u"ΠΙΝΑΚΑΣ"
54 cm.set_alphabet(alphabet)
55 encdec(cm, plaintext)
56
57 '''
58
59 4
60
61 -----
62 defendtheeastwallofthecastle
63 ffyhmkhycliashadtrlhcchlblr
64 defendtheeastwallofthecastle
65 -----
66 fleeatonce
67 uaeolwrins
68 fleeatonce
69 -----
70 ΠΙΝΑΚΑΣ
71 ΡΑΖΠΣΕΓ
72 ΠΙΝΑΚΑΣ
73 -----
74 '''

```

2.9 Caesar

class secretpy.Caesar

The Caesar Cipher

decrypt (*text*, *key*=3, *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 decrypted text

Return type string

encrypt (*text*, *key*=3, *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 encrypted text

Return type string

2.9.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Caesar, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll, Block
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====
11         ↪ ')
12      print(plaintext)
13      enc = cipher.encrypt(plaintext, key, alphabet)
14      print(enc)
15      print(cipher.decrypt(enc, key, alphabet))
16
17  key = 3
18  cipher = Caesar()
19  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
```

(continues on next page)

(continued from previous page)

```

20 encdec(cipher, plaintext, key)
21
22 alphabet = al.GERMAN
23 plaintext = u"schweißgequältvomödentextzürnttypografjakob"
24 encdec(cipher, plaintext, key, alphabet)
25
26 alphabet = al.SWEDISH
27 plaintext = u"faqomschweizklövdutrångpjäxby"
28 encdec(cipher, plaintext, key, alphabet)
29
30 # using cryptmachine
31
32
33 def encdec(machine, plaintext):
34     print("-----")
35     print(plaintext)
36     enc = machine.encrypt(plaintext)
37     print(enc)
38     print(machine.decrypt(enc))
39
40
41 cm0 = CryptMachine(cipher, key)
42
43 cm = cm0
44 cm.set_alphabet(al.ENGLISH)
45 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
46 ↳(*;?&#. Great!"
47 encdec(cm, plaintext)
48
49 cm = Block(cm, length=5, sep="-")
50 plaintext = "This text is divided by blocks of length 5!"
51 encdec(cm, plaintext)
52
53 cm = SaveAll(cm0)
54 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
55 encdec(cm, plaintext)
56
57 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
58 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
59 encdec(cm, plaintext)
60
61 cm.set_alphabet(al.JAPANESE_HIRAGANA)
62 cm.set_key(1)
63 plaintext = u"text "
64 encdec(cm, plaintext)
65
66 '''
67 Output:
68
69 =====
70 thequickbrownfoxjumpsoverthelazydog
71 wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
72 thequickbrownfoxjumpsoverthelazydog
73 =====
74 schweißgequältvomödentextzürnttypografjakob
75 vfkzhlcjhtxßowyrpaghqwhäwübuqwwösrjudimdnre

```

(continues on next page)

(continued from previous page)

```

76  schweißgequält vom ödentext zur nttypograf jakob
77  =====
78  faqomschweizklövdutrångpjäxby
79  idtrpvfkzhlöncygxwuaqjsmbåeä
80  faqomschweizklövdutrångpjäxby
81  -----
82  I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
83  lgrqworyhqrqdoskdehwfkdudfwhuvlzloouhpryhdooriwkhpjuhdw
84  idontlovenonalphabetcharactersiwillremoveallofthemgreat
85  -----
86  This text is divided by blocks of length 5!
87  wkllvw-hawlv-glylg-hgebe-orfnv-riohq-jwk
88  thistextisdividedbyblocksoflength
89  -----
90  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
91  L oryh qrq-doskdehw fkdudfwhuv. Wkhvh duh : ^,&@$~(*;?&#. Wkdw'v lw!
92  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
93  -----
94  Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
95  Mm ehfrphv Mm ehfdxvh zh xvh HQKOMVL_VTXDUH_MM!
96  Ii becomes Ii because we use ENGLISH_SQUARE_II!
97  -----
98  text
99  text
100 text
101 '''

```

2.10 Caesar Progressive

class secretpy.CaesarProgressive

The Caesar Progressive Cipher

decrypt (text, key=3, 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 decrypted text

Return type string

encrypt (text, key=3, 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 encrypted text

Return type string

2.10.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import CaesarProgressive, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====
11         ↪ ')
12         print(plaintext)
13         enc = cipher.encrypt(plaintext, key, alphabet)
14         print(enc)
15         print(cipher.decrypt(enc, key, alphabet))
16
17 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
18 key = 3
19 cipher = CaesarProgressive()
20
21 encdec(cipher, plaintext, key)
22
23 alphabet = al.GERMAN
24 plaintext = u"schweißgequältvomödentextzürnttypografjakob"
25 encdec(cipher, plaintext, key, alphabet)
26
27 alphabet = al.SWEDISH
28 plaintext = u"faqomschweizklövdutrångpjäxby"
29 encdec(cipher, plaintext, key, alphabet)
30
31 # using cryptmachine
32
33 def encdec(machine, plaintext):
34     print("-----")
35     print(plaintext)
36     enc = machine.encrypt(plaintext)
37     print(enc)
38     print(machine.decrypt(enc))
39
40
41 cm0 = CryptMachine(cipher, key)
42
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 cm = SaveAll(cm0)
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 cm.set_key(1)
53 plaintext = u"text "
54 encdec(cm, plaintext)
55
56 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
57 ↳(*;?&#. Great!"
58 cm = cm0
59 cm.set_alphabet(al.ENGLISH)
60 encdec(cm, plaintext)
61
62 '''
63 Output:
64
65 =====
66 thequickbrownfoxjumpsoverthelazydog
67 wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
68 thequickbrownfoxjumpsoverthelazydog
69 =====
70 schweißgequältvomödentextzürnttypografjakob
71 vgmülqi qpüdkäficbryägnßtqxörovwüuunzjpumxüq
72 schweißgequältvomödentextzürnttypografjakob
73 =====
74 faqomschweizklövdutrångpjäxby
75 ievutålrqgvkzäqkwlkui cmhåxcå
76 faqomschweizklövdutrångpjäxby
77 -----
78 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
79 L ptbl vxx-lxcvprvl vbvnxasesu. Wljyl iao : ^,&@$~(*;?&#. Etnh'h yk!
80 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
81 -----
82 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
83 Mn glkwvpd Vw qutsnmz sb sre FPKPOYP_AZEMDS_XY!
84 Ii becomes Ii because we use ENGLISH_SQUARE_II!
85 -----
86 text
87 text
88 text
89 -----
90 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
91 jfriryrdnxzznzexrtxnxdxpcuguwncptubpybjtqcdhzodbkfrfcw
92 idontlovenonalphabetcharactersiwillremoveallofthemgreat
'''

```

2.11 Chao

class secretpy.Chao

The Chaocipher

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

2.11.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Chao, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase, SaveSpaces
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  alphabet = "ptlnbqdeoysfavzkgjrihwumc" # RIGHT WHEEL PT
17  key = "hxuczvamdslkpefjrigtwobnyq"      # LEFT WHEEL CT
18
19  cm = SaveSpaces(UpperCase(CryptMachine(Chao(), key)))
20  cm.set_alphabet(alphabet)
21
22  plaintext = "well done is better than well said"
23  encdec(cm, plaintext)
24
25  plaintext = "plaintext"
26  encdec(cm, plaintext)
27
28  cm.set_alphabet(alphabets.ENGLISH)
29  cm.set_key(alphabets.ENGLISH)
30  plaintext = "do not use pc"
31  encdec(cm, plaintext)
32
33  '''

```

(continues on next page)

(continued from previous page)

```

34 Output:
35
36 well done is better than well said
37 OAHQ HCNY NX TSZJRR HJBY HQKS OUJY
38 WELL DONE IS BETTER THAN WELL SAID
39 -----
40 plaintext
41 HULROKQUA
42 PLAINTEXT
43 -----
44 do not use pc
45 DN LLQ QYM MW
46 DO NOT USE PC
47 -----
48 '''

```

2.12 Columnar Transposition

class secretpy.ColumnarTransposition

The Columnar Transposition Cipher

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

Decryption method

Parameters

- **text** (*string*) – Text to decrypt
- **key** (*string*) – 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** (*string*) – 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 ColumnarTransposition, 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(ColumnarTransposition(), key)
17
18  plaintext = "attackatdawn"
19  encdec(cm, plaintext)
20
21  key = "deutsch"
22  cm.set_key(key)
23  plaintext = "howstuffworks"
24  encdec(cm, plaintext)
25
26  '''
27  attackatdawn
28  tanakwadcatt
29  attackatdawn
30  -----
31  howstuffworks
32  ushfowftksrwo
33  howstuffworks
34  -----
35  '''

```

2.13 Four Square

class secretpy.FourSquare

The Four-Square Cipher

decrypt (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Decryption method

Parameters

- **text** (*string*) – Text to decrypt
- **key** (*tuple of two strings*) – 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=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Encryption method

Parameters

- **text** (*string*) – Text to encrypt
- **key** (*tuple of two strings*) – 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 FourSquare, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17 alphabet = alphabets.ENGLISH_SQUARE_OO
18
19 key = (u"example", u"keyword")
20
21 cm = UpperCase(CryptMachine(FourSquare()))
22
23 cm.set_alphabet(alphabet)
24 cm.set_key(key)
25 plaintext = u"Help me Obi wan Kenobi"
26 encdec(cm, plaintext)
27
28 plaintext = u"Help me Obi wan Kenobi a"
29 encdec(cm, plaintext)
30
31 alphabet = alphabets.ENGLISH_SQUARE_IJ
32 cm.set_alphabet(alphabet)
33 key = (u"criptog", u"segurt")
34 cm.set_key(key)
35 plaintext = u"Attack at dawn!"
36 encdec(cm, plaintext)
37
38
39 '''
```

(continues on next page)

(continued from previous page)

```

40 Help me Obi wan Kenobi
41 FYGMKYHOBXMFKKKIMD
42 HELPMEOWIWANKENOBI
43 -----
44 Help me Obi wan Kenobi a
45 FYGMKYHOBXMFKKKIMDPT
46 HELPMEOWIWANKENOBIAZ
47 -----
48 Attack at dawn!
49 PMMUTBPMCUXH
50 ATTACKATDAWN
51 -----
52 '''

```

2.14 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`

2.14.1 Examples

```

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

```

(continues on next page)

(continued from previous page)

```

5  from secretpy import alphabets
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 '''

```

2.15 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

2.15.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Keyword, alphabets
5
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 print("-----")
20
21 plaintext = u"thisisasecretmessage"
22 key = "keyword"
23
24 # use default english alphabet
25 print(plaintext)
26 enc = cipher.encrypt(plaintext, key)
27 print(enc)
28 dec = cipher.decrypt(enc, key)
29 print(dec)

```

2.16 MyszowskiTransposition

class secretpy.**MyszowskiTransposition**

The Myszowski 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

2.16.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import MyszkowskiTransposition, 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 = "tomato"
17  cm = CryptMachine(MyszkowskiTransposition(), key)
18
19  alphabet = alphabets.ENGLISH
20
21  cm.set_alphabet(alphabet)
22  plaintext = "wearediscoveredfleeatonce"
23  encdec(cm, plaintext)
24
25  '''
26  wearediscoveredfleeatonce
27  rofoacdtedseeeacweivrlene
28  wearediscoveredfleeatonce
29  -----
30  '''
```

2.17 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`

2.17.1 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",

```

(continues on next page)

(continued from previous page)

```

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 = [
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"Z", 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"X", 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 -----

```

(continues on next page)

(continued from previous page)

```

79
80 102 82 90 101 102
81
82 -----
83 ΠΙΝΑΚΑΣ
84 95 78 87 65 79 65 97
85 ΠΙΝΑΚΑΣ
86 -----
87 '''

```

2.18 Playfair

class secretpy.**Playfair**

The Playfair Cipher

decrypt (*text*, *key*=", *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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_SQUARE_IJ is used

Returns text

Return type string

encrypt (*text*, *key*=", *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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_SQUARE_IJ is used

Returns text

Return type string

2.18.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3
4 from secretpy import Playfair, CryptMachine
5 from secretpy.cmdecorators import UpperCase
6

```

(continues on next page)

(continued from previous page)

```

7
8 def encdec(machine, plaintext):
9     print(plaintext)
10    enc = machine.encrypt(plaintext)
11    print(enc)
12    dec = machine.decrypt(enc)
13    print(dec)
14    print("-----")
15
16
17 cm = 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
35 '''
36 Hide the gold in the tree stump
37 BMODZBXDNABEKUDMUIXMMOUVIF
38 HIDETHEGOLDINTHETREESTUMP
39 -----
40 sometext
41 KQIXVIIW
42 SOMETEXT
43 -----
44 this is a secret message
45 ZBMKMKFORDEXZIMOOFDX
46 THISISASECRETMESSAGE
47 -----
48 '''

```

2.19 Polybius

class secretpy.Polybius

The Polybius Cipher

decrypt (text, key="", alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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*=(*'a'*, *'b'*, *'c'*, *'d'*, *'e'*, *'f'*, *'g'*, *'h'*, *'ij'*, *'k'*, *'l'*, *'m'*, *'n'*, *'o'*, *'p'*, *'q'*, *'r'*, *'s'*, *'t'*, *'u'*, *'v'*, *'w'*, *'x'*, *'y'*, *'z'*))

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 Polybius, CryptMachine
5  from secretpy.cmdecorators import LowerCase
6  import secretpy.alphabets as alph
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
17
18  cm = CryptMachine(Polybius())
19
20  plaintext = u"defendtheeastwallofthecastle"
21  encdec(cm, plaintext)
22
23  alphabet = [
24      u"p", u"h", u"q", u"g", u"m",
25      u"e", u"a", u"y", u"l", u"n",
26      u"o", u"f", u"d", u"x", u"k",
27      u"r", u"c", u"v", u"s", u"z",
28      u"w", u"b", u"u", u"t", u"ij"
29  ]
30  cm.set_alphabet(alphabet)
31  plaintext = "sometext"
32  encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

33
34 plaintext = "thisisasecretmessage"
35 encdec(cm, plaintext)
36
37 cm.set_alphabet(alph.GREEK)
38 plaintext = u"ΙΙΙΝΑΚΑΣ"
39 cm = LowerCase(cm)
40 encdec(cm, plaintext)

```

2.20 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

2.20.1 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"

```

(continues on next page)

(continued from previous page)

```

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

```

2.21 Rot13

class secretpy.Rot13

The Rot13 Cipher (Half)

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

2.21.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot13, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(machine, plaintext):
9      print("-----")
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17  cm = SaveAll(CryptMachine(Rot13()))
18
19  plaintext = u"This is a secret message"
20  encdec(cm, plaintext)
21
22  plaintext = u"Why did the chicken cross the road Gb trg gb gur bgure fvqr"
23  encdec(cm, plaintext)
24
25  plaintext = u"The quick brown fox jumps over the lazydog"
26  cm.set_alphabet(al.GERMAN)
27  encdec(cm, plaintext)
28
29  plaintext = u""
30  cm.set_alphabet(al.RUSSIAN)
31  encdec(cm, plaintext)
32
33  cm.set_alphabet(al.JAPANESE_HIRAGANA)
34  cm.set_key(1)
35  plaintext = u""
36  encdec(cm, plaintext)
37
38  '''
39  -----
40  This is a secret message
41  Guvf vf n frperg zrffntr
42  This is a secret message
43  -----
44  Why did the chicken cross the road Gb trg gb gur bgure fvqr
45  Jul qvq gur puvpxra pebff gur ebnq To get to the other side
```

(continues on next page)

(continued from previous page)

```

46 Why did the chicken cross the road Gb trg gb gur bgure fvqr
47 -----
48 The quick brown fox jumps over the lazydog
49 Ewt bfxrz qcbhü ußi yföad ßgtc ewt äpkjsßv
50 The quick brown fox jumps over the lazydog
51 -----
52
53
54
55 -----
56
57
58
59 '''

```

2.22 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

2.22.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3

```

(continues on next page)

(continued from previous page)

```

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  '''

```

2.23 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

2.23.1 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)

```

2.24 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

2.24.1 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)
```

2.25 Simple Substitution

class secretpy.**Scytale**

The Scytale Cipher

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

Decryption method :param text: Text to decrypt :param key: Decryption key - Number of windings :param alphabet: 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 :param text: Text to encrypt :param key: Encryption key - Number of windings :param alphabet: Alphabet which will be used,

if there is no a value, English is used

Returns encrypted text

Return type string

2.25.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Scytale, CryptMachine, alphabets
5  from secretpy.cmdecorators import SaveAll
6
7
8  alphabet = alphabets.GERMAN
9  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
10 key = 3
11 cipher = Scytale()
12
13 print(plaintext)
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 print('=====')
20
21 print(plaintext)
22 # use default english alphabet
23 enc = cipher.encrypt(plaintext, key)
24 print(enc)
25 dec = cipher.decrypt(enc, key)
26 print(dec)
27
28 # using cryptmachine
29
30
31 def encdec(machine, plaintext):
32     print("-----")
33     print(plaintext)
34     enc = machine.encrypt(plaintext)
35     print(enc)
36     print(machine.decrypt(enc))
37
38
39 cm0 = CryptMachine(cipher, key)
40 cm0.set_alphabet(alphabet)
41
42 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
43 cm = SaveAll(cm0)
44 encdec(cm, plaintext)
45

```

(continues on next page)

(continued from previous page)

```

46 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
   ↳ (*;?&#. Great!"
47 cm = cm0
48 encdec(cm, plaintext)
49
50 '''
51 Output:
52
53 thequickbrownfoxjumpsoverthelazydog
54 tqcrnxmorezohukofjpvtlygeibwousehad
55 thequickbrownfoxjumpsoverthelazydog
56 =====
57 thequickbrownfoxjumpsoverthelazydog
58 tqcrnxmorezohukofjpvtlygeibwousehad
59 thequickbrownfoxjumpsoverthelazydog
60 -----
61 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
62 I vola tac-rheeaile npbcrtseat. Ttona heh : ^,&@$~(*;?&#. Aets'r hs!
63 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
64 -----
65 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
66 inonahehaeilevlfertdtvolatacrwlmeltmeolenpbcrtsiroaohga
67 idontlovenonalphabetcharactersiwillremoveallofthemgreat
68
69 '''

```

2.26 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

2.26.1 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)
30 dec = cipher.decrypt(enc, key, alphabet)
31 print(dec)
32
33 '''
34 thequickbrownfoxjumpsoverthelazydog
35 ujgrvobzaspwnhpxkvmqtpfgsujgldäycpi
36 thequickbrownfoxjumpsoverthelazydog
37 -----
38 thisisasecretmessage
39 u jototdtgbsgumgttdig
40 thisisasecretmessage
41 '''

```

2.27 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

2.27.1 Examples

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

(continues on next page)

(continued from previous page)

```

30 cm.set_key(key)
31 plaintext = u"attack at dawn"
32 encdec(cm, plaintext)
33
34 '''
35 Help me Obi wan Kenobi
36 HJKNEMDHOHSACLYRISFJKUUKBEF
37 HELPMEOBIWANKENOB I
38 -----
39 attack at dawn
40 QCTZABCSKXCATDAFWN
41 ATTACKATDAWN
42 -----
43 '''

```

2.28 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 with ‘.’ 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 with ‘.’ is used

Returns text

Return type string

2.28.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3

```

(continues on next page)

(continued from previous page)

```

4  from secretpy import Trifid, CryptMachine
5  from secretpy.cmdecorators import SaveAll
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 = u"epsducvwym.zlkxnbtfgorijhaq" # 27 characters
20  cm.set_alphabet(alphabet)
21
22  plaintext = u"defendtheeastwallofthecastle"
23  encdec(cm, plaintext)
24
25  cm1 = cm
26  alphabet = (
27      u"aää", u"b", u"c",
28      u"d", u"e", u"f",
29      u"g", u"h", u"i",
30
31      u"j", u"k", u"l",
32      u"m", u"n", u"oö",
33      u"p", u"q", u"r",
34
35      u"s", u"t", u"u",
36      u"v", u"w", u"x",
37      u"y", u"z", u"+",
38  )
39  cm1.set_alphabet(alphabet)
40
41  plaintext = u"Flygande bäckasiner söka hwila på mjuka tuvor!"
42  encdec(cm1, plaintext)
43
44  cm2 = SaveAll(cm)
45  alphabet = "felimardstbcghjknopquvwyz+"
46  cm2.set_alphabet(alphabet)
47
48  plaintext = u"Aide-toi, le ciel t'aidera"
49  encdec(cm2, plaintext)
50
51
52  '''
53  defendtheeastwallofthecastle
54  suefecphsegyyjiximfofocejlr
55  defendtheeastwallofthecastle
56  -----
57  Flygande bäckasiner söka hwila på mjuka tuvor!
58  fbiiabmdmdsazckwpnujshvokdgpaggackzkri
59  flygandebäckasinersokahwilapamjukatur
60  -----

```

(continues on next page)

(continued from previous page)

```

61 Aide-toi, le ciel t'aidera
62 Fmjf-voi, ss uftf p'ufeqqc
63 Aide-toi, le ciel t'aidera
64 -----
65 '''

```

2.29 Two Square

class secretpy.**TwoSquare**

The Two-Square Cipher, also called Double Playfair

decrypt (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

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 TwoSquare, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)

```

(continues on next page)

(continued from previous page)

```

11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17 alphabet = alphabets.ENGLISH_SQUARE_OO
18
19 key = (u"example", u"keyword")
20
21 cm = UpperCase(CryptMachine(TwoSquare()))
22
23 cm.set_alphabet(alphabet)
24 cm.set_key(key)
25 plaintext = u"Help me Obi wan Kenobi"
26 encdec(cm, plaintext)
27
28 plaintext = u"Help me Obi wan Kenobi y"
29 encdec(cm, plaintext)
30
31 '''
32 Help me Obi wan Kenobi
33 XGDLXWSDJYRYHOTKDG
34 HELPMEOWIWANKENOBI
35 -----
36 Help me Obi wan Kenobi y
37 XGDLXWSDJYRYHOTKDGZX
38 HELPMEOWIWANKENOBIYZ
39 -----
40 '''

```

2.30 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

2.30.1 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  '''

```

2.31 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

2.31.1 Examples

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

(continues on next page)

(continued from previous page)

```

32 thequickbrownfoxjumpsoverthelazydog
33 ßzwäiämütöckxxcdöiwdgyjwöhzoßsfmvy
34 thequickbrownfoxjumpsoverthelazydog
35 -----
36 attackatdawn
37 lxfopefrnhr
38 attackatdawn
39 '''

```

2.32 Zigzag

class `secretpy.Zigzag`

The Zigzag Cipher (Rail-Fence)

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

Decryption method

Parameters

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

Returns `text`

Return type `string`

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

Encryption method

Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – unused

Returns `text`

Return type `string`

2.32.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Zigzag
5
6
7  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
8  plaintext = u"thequick"
9  key = 3
10
11 chipher = Zigzag()

```

(continues on next page)

(continued from previous page)

```

12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key)
15 print(enc)
16 dec = cipher.decrypt(enc, key)
17 print(dec)
18
19 #####
20
21 print("-----")
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 dttfsehdswotatfneaalhcleeele
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*), 18

C

Caesar (*class in secretpy*), 20
 CaesarProgressive (*class in secretpy*), 22
 Chao (*class in secretpy*), 24
 ColumnarTransposition (*class in secretpy*), 26

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*), 18
 decrypt () (*secretpy.Caesar method*), 20
 decrypt () (*secretpy.CaesarProgressive method*), 22
 decrypt () (*secretpy.Chao method*), 24
 decrypt () (*secretpy.ColumnarTransposition method*), 26
 decrypt () (*secretpy.FourSquare method*), 27
 decrypt () (*secretpy.Gronsfeld method*), 29
 decrypt () (*secretpy.Keyword method*), 30
 decrypt () (*secretpy.MyszkowskiTransposition method*), 31
 decrypt () (*secretpy.Nihilist method*), 33
 decrypt () (*secretpy.Playfair method*), 35

decrypt () (*secretpy.Polybius method*), 36
 decrypt () (*secretpy.Porta method*), 38
 decrypt () (*secretpy.Rot13 method*), 39
 decrypt () (*secretpy.Rot18 method*), 42
 decrypt () (*secretpy.Rot47 method*), 43
 decrypt () (*secretpy.Rot5 method*), 41
 decrypt () (*secretpy.Scytale method*), 44
 decrypt () (*secretpy.SimpleSubstitution method*), 46
 decrypt () (*secretpy.ThreeSquare method*), 47
 decrypt () (*secretpy.Trifid method*), 49
 decrypt () (*secretpy.TwoSquare method*), 51
 decrypt () (*secretpy.Vic method*), 52
 decrypt () (*secretpy.Vigenere method*), 53
 decrypt () (*secretpy.Zigzag method*), 55

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*), 18
 encrypt () (*secretpy.Caesar method*), 20
 encrypt () (*secretpy.CaesarProgressive method*), 22
 encrypt () (*secretpy.Chao method*), 25
 encrypt () (*secretpy.ColumnarTransposition method*), 26
 encrypt () (*secretpy.FourSquare method*), 27
 encrypt () (*secretpy.Gronsfeld method*), 29
 encrypt () (*secretpy.Keyword method*), 30
 encrypt () (*secretpy.MyszkowskiTransposition method*), 32
 encrypt () (*secretpy.Nihilist method*), 33
 encrypt () (*secretpy.Playfair method*), 35
 encrypt () (*secretpy.Polybius method*), 37
 encrypt () (*secretpy.Porta method*), 38
 encrypt () (*secretpy.Rot13 method*), 39
 encrypt () (*secretpy.Rot18 method*), 42

`encrypt()` (*secretpy.Rot47 method*), 44
`encrypt()` (*secretpy.Rot5 method*), 41
`encrypt()` (*secretpy.Scytale method*), 44
`encrypt()` (*secretpy.SimpleSubstitution method*), 46
`encrypt()` (*secretpy.ThreeSquare method*), 48
`encrypt()` (*secretpy.Trifid method*), 49
`encrypt()` (*secretpy.TwoSquare method*), 51
`encrypt()` (*secretpy.Vic method*), 52
`encrypt()` (*secretpy.Vigenere method*), 54
`encrypt()` (*secretpy.Zigzag method*), 55

F

`FourSquare` (*class in secretpy*), 27

G

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

K

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

M

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

N

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

P

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

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

`Porta` (*class in secretpy*), 38

R

`Rot13` (*class in secretpy*), 39

`Rot18` (*class in secretpy*), 42

`Rot47` (*class in secretpy*), 43

`Rot5` (*class in secretpy*), 41

S

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

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

T

`ThreeSquare` (*class in secretpy*), 47

`Trifid` (*class in secretpy*), 49

`TwoSquare` (*class in secretpy*), 51

V

`Vic` (*class in secretpy*), 52

`Vigenere` (*class in secretpy*), 53

Z

`Zigzag` (*class in secretpy*), 55