SecretPy Documentation

Release 1.0

Read the Docs

Jul 27, 2021

Contents:

CHAPTER 1

SecretPy

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
idtrpvfkzhlönocygxwuaqjsmbåeä
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)

CHAPTER 2

API

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import ADFGX

alphabet = [
    u"b", u"t", u"a", u"l", u"p", u"d", u"h", u"o", u"z", u"k", u"q",
    u"f", u"v", u"s", u"n", u"g", u"ij", u"c", u"u", u"x", u"m", u"r",
    u"e", u"w", u"y"
]

plaintext = u"attackatonce"
key = "cargo"
cipher = ADFGX()

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

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

################################################################################
print("--------------------------------")

alphabet = [
    u"f", u"n", u"h", u"e", u"q",
    u"r", u"d", u"z", u"o", u"c",
    u"ij", u"s", u"a", u"g", u"u",
    u"b", u"v", u"k", u"p", u"w",
    u"x", u"m", u"y", u"t", u"l"
]
key = "battle"
plaintext = "attackatdawn"

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

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

################################################################################
print("--------------------------------")

key = "deutsch"
plaintext = "howstuffworks"

# use default english alphabet 5x5
print(plaintext)
enc = cipher.encrypt(plaintext, key)
print(enc)

dec = cipher.decrypt(enc, key)
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import ADFGVX, CryptMachine


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


key = "cargo"
cm = CryptMachine(ADFGVX(), key)

alphabet = [
    u"f", u"n", u"h", u"e", u"q", u"0",
    u"r", u"d", u"z", u"o", u"c", u"9",
    u"ij", u"s", u"a", u"g", u"u", u"8",
```

(continues on next page)

(continued from previous page)

```
    u"b", u"v", u"k", u"p", u"w", u"7",
    u"x", u"m", u"y", u"t", u"1", u"6",
    u"1", u"2", u"3", u"4", u"5", u".",
]
cm.set_alphabet(alphabet)
key = "battle"
plaintext = "attackatdawn11.25"
encdec(cm, plaintext)

key = "deutsch"
cm.set_key(key)
plaintext = "howstuffworks"
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

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 vögaüewsphqmjnqtlücxoqfghvögzidäßqu
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Atbash
from secretpy import CryptMachine
from secretpy import alphabets
import secretpy.cmdecorators as md


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


cm = CryptMachine(Atbash())
cm = md.NoSpaces(md.UpperCase(cm))

plaintext = u"attackatdawn"
encdec(cm, plaintext)

plaintext = u""
cm.set_alphabet(alphabets.HEBREW)
encdec(cm, plaintext)

plaintext = u"The Fox jumps in Zoo too Achtung minen"
cm.set_alphabet(alphabets.GERMAN)
encdec(cm, plaintext)

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

cm.set_alphabet(alphabets.ARABIC)
plaintext = u""
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Autokey, alphabets


alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = "queenly"

cipher = Autokey()
print(plaintext)

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

##########################################################

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 föiudtäßivamvhyyäeeüxüonhbwwzvßlwvk
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

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


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


alphabet = alphabets.ENGLISH_SQUARE_IJ

key = (81257, u"eightyonethousandtwohundredfiftyseven")

cm = NoSpaces(UpperCase(CryptMachine(Bazeries())))

cm.set_alphabet(alphabet)
cm.set_key(key)
plaintext = u"Whoever has made a voyage up the Hudson" \
            u" must remember the Kaatskill mountains"
encdec(cm, plaintext)

'''
Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
DUMTMCDSENRTEMVEQXMOELCCRVXDMDKWXNNMUKRDKUMYNMBPRKEEPMGNGEKWXCRWB
WHOEVERHASMADEAVOYAGEUPTHEHUDSONMUSTREMEMBERTHEKAATSKILLMOUNTAINS
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Beaufort, CryptMachine, alphabets as al
from secretpy.cmdecorators import SaveAll


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 = "key"
cipher = Beaufort()

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)

# using cryptmachine


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

(continues on next page)

(continued from previous page)

```
    print(machine.decrypt(enc))


cm0 = CryptMachine(cipher, key)

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

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

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

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

'''
Output:

==========================================================================================
thequickbrownfoxjumpsoverthelazydog
rxuukqiuxtqcxzknveypgwjutlrgtylgvwy
thequickbrownfoxjumpsoverthelazydog
==========================================================================================
schweißgequältvomödentextzürnttypografjakob
wcrsaqlüuyoüßpdäwöhalvabvjäxvfvkjäühkßpkykj
schweißgequältvomödentextzürnttypografjakob
==========================================================================================
faqomschweizklövdutrångpjäxby
feizvgiåcgzöawzsbeuqäääjbgbjj
faqomschweizklövdutrångpjäxby
--------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
C tkpa lwr-yzprkdur crknyilutm. Fdagg ehg : ^,&@$~(*;?&#. Lrkl'g cl!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
--------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Bw xfclyag Bw xfcyqnu oa esa UXYOBNR_SPEKOU_BW!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
--------------------------------------------------------------------
text
text
text
--------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
cbkxlnwjuxqlktjdexglwdehkcfgngciqzthgskpayztkflrgsstayr
idontlovenonalphabetcharactersiwillremoveallofthemgreat
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Bifid, CryptMachine, alphabets


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


key = 5
cm = CryptMachine(Bifid(), key)
alphabet = [
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
```

(continues on next page)

(continued from previous page)

```
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"1", u"2", u"3", u"4", u"5", u"6"
]
cm.set_alphabet(alphabet)
plaintext = u""
encdec(cm, plaintext)

alphabet = [
    u"p", u"h", u"q", u"g", u"m",
    u"e", u"a", u"y", u"l", u"n",
    u"o", u"f", u"d", u"x", u"k",
    u"r", u"c", u"v", u"s", u"z",
    u"w", u"b", u"u", u"t", u"ij"
]
cm.set_alphabet(alphabet)
plaintext = u"defendtheeastwallofthecastle"
encdec(cm, plaintext)

alphabet = [
    u"b", u"g", u"w", u"k", u"z",
    u"q", u"p", u"n", u"d", u"s",
    u"ij", u"o", u"a", u"x", u"e",
    u"f", u"c", u"l", u"u", u"m",
    u"t", u"h", u"y", u"v", u"r"
]
cm.set_alphabet(alphabet)
cm.set_key(10)
plaintext = "fleeatonce"
encdec(cm, plaintext)

alphabet = alphabets.GREEK.upper()
plaintext = u"ΠΙΝΑΚΑΣ"
cm.set_alphabet(alphabet)
encdec(cm, plaintext)

'''

4

----------------------------------
defendtheeastwallofthecastle
ffyhmkhycpliashadtrlhcchlblr
defendtheeastwallofthecastle
----------------------------------
fleeatonce
uaeolwrins
fleeatonce
----------------------------------
ΠΙΝΑΚΑΣ
ΡΛΖΠΣΕΓ
ΠΙΝΑΚΑΣ
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

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


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"
```

(continues on next page)

(continued from previous page)

```
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)

# using cryptmachine


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)

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


'''
Output:

==========================================================================================
thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
==========================================================================================
schweißgequältvomödentextzürnttypografjakob
vfkzhlcjhtxßowyrpaghqwhäwübuqwwösrjudimdnre
```

(continues on next page)

(continued from previous page)

```
schweißgequältvomödentextzürnttypografjakob
==========================================================================================
faqomschweizklövdutrångpjäxby
idtrpvfkzhlönocygxwuaqjsmbåeä
faqomschweizklövdutrångpjäxby
------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
lgrqworyhqrqdoskdehwfkdudfwhuvlzloouhpryhdooriwkhpjuhdw
idontlovenonalphabetcharactersiwillremoveallofthemgreat
------------------------------------------------------------------
This text is divided by blocks of length 5!
wklvw-hawlv-glylg-hgebe-orfnv-riohq-jwk
thistextisdividedbyblocksoflength
------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. 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
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import CaesarProgressive, CryptMachine, alphabets as al
from secretpy.cmdecorators import SaveAll


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))


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

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)

# using cryptmachine


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


cm0 = CryptMachine(cipher, key)

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

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

'''
Output:

==========================================================================================
thequickbrownfoxjumpsoverthelazydog
wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
thequickbrownfoxjumpsoverthelazydog
==========================================================================================
schweißgequältvomödentextzürnttypografjakob
vgmülqiqpüdkäficbryägnßtqxörovwüuunzjpumxüq
schweißgequältvomödentextzürnttypografjakob
==========================================================================================
faqomschweizklövdutrångpjäxby
ievutålreqvkzäqkwllkuicmhåxcå
faqomschweizklövdutrångpjäxby
------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
L ptbl vxx-lxcvprvl vbvnxasesu. Wljyl iao : ^,&@$~(*;?&#. Etnh'h yk!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Mn glkwvpd Vw qutsnmz sb sre FPKPOYP_AZEMDS_XY!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------
text
text
text
------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
jfrryrvdnxzznzexrtxnxdxpzcuguwncptubpybjtqcdhzodbkfrfcw
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

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


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


alphabet = "ptlnbqdeoysfavzkgjrihwxumc"  # RIGHT WHEEL PT
key = "hxuczvamdslkpefjrigtwobnyq"       # LEFT WHEEL CT

cm = SaveSpaces(UpperCase(CryptMachine(Chao(), key)))
cm.set_alphabet(alphabet)

plaintext = "well done is better than well said"
encdec(cm, plaintext)

plaintext = "plaintext"
encdec(cm, plaintext)

cm.set_alphabet(alphabets.ENGLISH)
cm.set_key(alphabets.ENGLISH)
plaintext = "do not use pc"
encdec(cm, plaintext)

'''
```

(continues on next page)

(continued from previous page)

```
Output:

well done is better than well said
OAHQ HCNY NX TSZJRR HJBY HQKS OUJY
WELL DONE IS BETTER THAN WELL SAID
------------------------------
plaintext
HULROKQUA
PLAINTEXT
------------------------------
do not use pc
DN LLQ QYM MW
DO NOT USE PC
------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import ColumnarTransposition, CryptMachine


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


key = "cargo"
cm = CryptMachine(ColumnarTransposition(), key)

plaintext = "attackatdawn"
encdec(cm, plaintext)

key = "deutsch"
cm.set_key(key)
plaintext = "howstuffworks"
encdec(cm, plaintext)

'''
attackatdawn
tanakwadcatt
attackatdawn
-------------------------------
howstuffworks
ushfowftksrwo
howstuffworks
-------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import FourSquare, CryptMachine, alphabets
from secretpy.cmdecorators import UpperCase


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


alphabet = alphabets.ENGLISH_SQUARE_OQ

key = (u"example", u"keyword")

cm = UpperCase(CryptMachine(FourSquare()))

cm.set_alphabet(alphabet)
cm.set_key(key)
plaintext = u"Help me Obi wan Kenobi"
encdec(cm, plaintext)

plaintext = u"Help me Obi wan Kenobi a"
encdec(cm, plaintext)

alphabet = alphabets.ENGLISH_SQUARE_IJ
cm.set_alphabet(alphabet)
key = (u"criptog", u"segurt")
cm.set_key(key)
plaintext = u"Attack at dawn!"
encdec(cm, plaintext)


'''
```

(continues on next page)

(continued from previous page)

```
Help me Obi wan Kenobi
FYGMKYHOBXMFKKKIMD
HELPMEOBIWANKENOBI
---------------------------------
Help me Obi wan Kenobi a
FYGMKYHOBXMFKKKIMDPT
HELPMEOBIWANKENOBIAZ
---------------------------------
Attack at dawn!
PMMUTBPMCUXH
ATTACKATDAWN
---------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Gronsfeld
```

(continues on next page)

(continued from previous page)

```
from secretpy import alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = (4, 7, 9)

cipher = Gronsfeld()
print(plaintext)

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

##########################################################

print("----------------------------------")

plaintext = u"attackatdawn"
key = (14, 2, 11)

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

'''
thequickbrownfoxjumpsoverthelazydog
xonuörgrkvvbrmxöqßqwösünväqisjßbmsn
thequickbrownfoxjumpsoverthelazydog
----------------------------------
attackatdawn
oveoevovooyy
attackatdawn
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Keyword, alphabets


alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = "queenly"

cipher = Keyword()
print(plaintext)

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

print("----------------------------------")

plaintext = u"thisisasecretmessage"
key = "keyword"

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

2.16 MyszkowskiTransposition

class secretpy.**MyszkowskiTransposition**
The Myszkowski 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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import MyszkowskiTransposition, CryptMachine
from secretpy import alphabets


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


key = "tomato"
cm = CryptMachine(MyszkowskiTransposition(), key)

alphabet = alphabets.ENGLISH

cm.set_alphabet(alphabet)
plaintext = "wearediscoveredfleeatonce"
encdec(cm, plaintext)

'''
wearediscoveredfleeatonce
rofoacdtedseeeacweivrlene
wearediscoveredfleeatonce
--------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Nihilist
from secretpy import CryptMachine


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


key = "russian"
cm = CryptMachine(Nihilist(), key)
alphabet = [
    u"z", u"e", u"b", u"r", u"a",
    u"s", u"c", u"d", u"f", u"g",
    u"h", u"ij", u"k", u"l", u"m",
```

(continues on next page)

(continued from previous page)

```
    u"n", u"o", u"p", u"q", u"t",
    u"u", u"v", u"w", u"x", u"y"
]
plaintext = u"dynamitewinterpalace"
cm.set_alphabet(alphabet)
encdec(cm, plaintext)

alphabet = [
    u"a", u"b", u"c", u"d", u"e", u"f",
    u"g", u"h", u"i", u"j", u"k", u"l",
    u"m", u"n", u"o", u"p", u"q", u"r",
    u"s", u"t", u"u", u"v", u"w", u"x",
    u"y", u"z", u"0", u"1", u"2", u"3",
    u"4", u"5", u"6", u"7", u"8", u"9",
]
key = "freedom"
plaintext = u"meetthursday2300hr"
cm.set_alphabet(alphabet)
cm.set_key(key)
encdec(cm, plaintext)

alphabet = [
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"", u"", u"", u"", u"", u"",
    u"1", u"2", u"3", u"4", u"5", u"6"
]

cm.set_alphabet(alphabet)
key = u""
plaintext = u""
encdec(cm, plaintext)

alphabet = [
    u"A", u"B", u"Γ", u"Δ", u"E",
    u"Z", u"H", u"Θ", u"I", u"K",
    u"Λ", u"M", u"N", u"Ξ", u"O",
    u"Π", u"P", u"Σ", u"T", u"Υ",
    u"Φ", u"X", u"Ψ", u"Ω"
]
plaintext = u"ΠINAKAΣ"
cm.set_alphabet(alphabet)
encdec(cm, plaintext)

'''
Output:

dynamitewinterpalace
37 106 62 36 67 47 86 26 104 53 62 77 27 55 57 66 55 36 54 27
dynamitewinterpalace
----------------------------------
meetthursday2300hr
47 51 30 57 56 55 74 52 77 29 26 65 88 87 69 89 37 51
meetthursday2300hr
----------------------------------
```

(continues on next page)

(continued from previous page)

```
102 82 90 101 102

----------------------------------
ΠINAKAΣ
95 78 87 65 79 65 97
ΠINAKAΣ
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Playfair, CryptMachine
from secretpy.cmdecorators import UpperCase

```

(continues on next page)

(continued from previous page)

```

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


cm = UpperCase(CryptMachine(Playfair()))
alphabet = [
    u"p", u"l", u"a", u"y", u"f",
    u"i", u"r", u"e", u"x", u"m",
    u"b", u"c", u"d", u"g", u"h",
    u"k", u"n", u"o", u"q", u"s",
    u"t", u"u", u"v", u"w", u"z",
]
cm.set_alphabet(alphabet)
plaintext = u"Hide the gold in the tree stump"
encdec(cm, plaintext)

plaintext = "sometext"
encdec(cm, plaintext)

plaintext = "this is a secret message"
encdec(cm, plaintext)

'''
Hide the gold in the tree stump
BMODZBXDNABEKUDMUIXMMOUVIF
HIDETHEGOLDINTHETREESTUMP
----------------------------------
sometext
KQIXVIIW
SOMETEXT
----------------------------------
this is a secret message
ZBMKMKFORDEXZIMOOFDX
THISISASECRETMESSAGE
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Polybius, CryptMachine
from secretpy.cmdecorators import LowerCase
import secretpy.alphabets as alph


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


cm = CryptMachine(Polybius())

plaintext = u"defendtheeastwallofthecastle"
encdec(cm, plaintext)

alphabet = [
    u"p", u"h", u"q", u"g", u"m",
    u"e", u"a", u"y", u"l", u"n",
    u"o", u"f", u"d", u"x", u"k",
    u"r", u"c", u"v", u"s", u"z",
    u"w", u"b", u"u", u"t", u"ij"
]
cm.set_alphabet(alphabet)
plaintext = "sometext"
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"ΠΙΝΑΚΑΣ"
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  dßwheputrkrnßöroznpgcvdübmzüöwhatvy
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Rot13, CryptMachine, alphabets as al
from secretpy.cmdecorators import SaveAll


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


cm = SaveAll(CryptMachine(Rot13()))

plaintext = u"This is a secret message"
encdec(cm, plaintext)

plaintext = u"Why did the chicken cross the road Gb trg gb gur bgure fvqr"
encdec(cm, plaintext)

plaintext = u"The quick brown fox jumps over the lazydog"
cm.set_alphabet(al.GERMAN)
encdec(cm, plaintext)

plaintext = u""
cm.set_alphabet(al.RUSSIAN)
encdec(cm, plaintext)

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

'''
-----------------------------------
This is a secret message
Guvf vf n frperg zrffntr
This is a secret message
-----------------------------------
Why did the chicken cross the road Gb trg gb gur bgure fvqr
Jul qvq gur puvpxra pebff gur ebnq To get to the other side
```

(continues on next page)

(continued from previous page)

```
Why did the chicken cross the road Gb trg gb gur bgure fvqr
--------------------------------
The quick brown fox jumps over the lazydog
Ewt bfxrz qcßhü ußi yföad ßgtc ewt äpkjsßv
The quick brown fox jumps over the lazydog
--------------------------------



--------------------------------



'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

```

(continues on next page)

(continued from previous page)

```
from secretpy import Rot5
from secretpy import alphabets
from secretpy import CryptMachine


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


cm = CryptMachine(Rot5())

plaintext = alphabets.DECIMAL
encdec(cm, plaintext)
'''
---------------------------------
0123456789
5678901234
0123456789
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Rot18
from secretpy import CryptMachine
from secretpy.cmdecorators import SaveCase, SaveSpaces, UpperCase
from secretpy import alphabets


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


cm = SaveCase(SaveSpaces(CryptMachine(Rot18())))

plaintext = u"The man has 536 dogs"
encdec(cm, plaintext)

plaintext = alphabets.RUSSIAN + alphabets.DECIMAL
cm.set_alphabet(alphabets.RUSSIAN)
encdec(cm, plaintext)

plaintext = u"  536 "
encdec(cm, plaintext)

plaintext = alphabets.GREEK + " " + alphabets.DECIMAL
cm = UpperCase(cm)
cm.set_alphabet(alphabets.GREEK)
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Rot47
from secretpy import CryptMachine
from secretpy.cmdecorators import SaveSpaces


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


cm = SaveSpaces(CryptMachine(Rot47()))

plaintext = u"The man has 536 dogs"
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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Scytale, CryptMachine, alphabets
from secretpy.cmdecorators import SaveAll


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

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)
print(dec)

# using cryptmachine


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


cm0 = CryptMachine(cipher, key)
cm0.set_alphabet(alphabet)

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

```

(continues on next page)

(continued from previous page)

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

'''
Output:

thequickbrownfoxjumpsoverthelazydog
tqcrnxmorezohukofjpvtlygeibwousehad
thequickbrownfoxjumpsoverthelazydog
====================================
thequickbrownfoxjumpsoverthelazydog
tqcrnxmorezohukofjpvtlygeibwousehad
thequickbrownfoxjumpsoverthelazydog
------------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
I vola tac-rheeaile npbcrtseat. Ttona heh : ^,&@$~(*;?&#. Aets'r hs!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
inonahehaeilevlfertdtvolatacrwlmeltmeolenpbcrtsiroaohga
idontlovenonalphabetcharactersiwillremoveallofthemgreat

'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import SimpleSubstitution
from secretpy import alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = u"dabcghijokzlmnpqrstuvfwxyäöeüß"

cipher = SimpleSubstitution()
print(plaintext)

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

##########################################################

print("-----------------------------------")

plaintext = u"thisisasecretmessage"
alphabet = alphabets.ENGLISH
key = u"dabcghijokzlmnpqrstuvfwxye"

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

'''
thequickbrownfoxjumpsoverthelazydog
ujgrvobzaspwnhpxkvmqtpfgsujgldäycpi
thequickbrownfoxjumpsoverthelazydog
-----------------------------------
thisisasecretmessage
ujototdtgbsgumgttdig
thisisasecretmessage
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

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


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


alphabet = alphabets.ENGLISH_SQUARE_OQ
key = (u"example", u"keyword", u"third")
cm = NoSpaces(UpperCase(CryptMachine(ThreeSquare())))
cm.set_alphabet(alphabet)
cm.set_key(key)
plaintext = u"Help me Obi wan Kenobi"
encdec(cm, plaintext)

alphabet = alphabets.ENGLISH_SQUARE_IJ
cm.set_alphabet(alphabet)
key = (u"criptog", u"segurt", u"mars")
```

(continues on next page)

(continued from previous page)

```
cm.set_key(key)
plaintext = u"attack at dawn"
encdec(cm, plaintext)

'''
Help me Obi wan Kenobi
HJKNEMDHOHSACLYRISFJKUUKBEF
HELPMEOBIWANKENOBI
-----------------------------------
attack at dawn
QCTZABCSKXCATDAFWN
ATTACKATDAWN
-----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

```

(continues on next page)

(continued from previous page)

```
from secretpy import Trifid, CryptMachine
from secretpy.cmdecorators import SaveAll


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


key = 5
cm = CryptMachine(Trifid(), key)

alphabet = u"epsducvwym.zlkxnbtfgorijhaq"  # 27 characters
cm.set_alphabet(alphabet)

plaintext = u"defendtheeastwallofthecastle"
encdec(cm, plaintext)

cm1 = cm
alphabet = (
    u"aåä", u"b", u"c",
    u"d", u"e",  u"f",
    u"g", u"h", u"i",

    u"j", u"k", u"l",
    u"m", u"n", u"oö",
    u"p", u"q", u"r",

    u"s", u"t", u"u",
    u"v", u"w", u"x",
    u"y", u"z", u"+",
)
cm1.set_alphabet(alphabet)

plaintext = u"Flygande bäckasiner söka hwila på mjuka tuvor!"
encdec(cm1, plaintext)

cm2 = SaveAll(cm)
alphabet = "felixmardstbcghjknopquvwyz+"
cm2.set_alphabet(alphabet)

plaintext = u"Aide-toi, le ciel t'aidera"
encdec(cm2, plaintext)


'''
defendtheeastwallofthecastle
suefecphsegyyjiximfofocejlrf
defendtheeastwallofthecastle
-----------------------------------
Flygande bäckasiner söka hwila på mjuka tuvor!
fbiiajbmdmdsazckwpnujshvokdgpaqgackzkri
flygandebackasinersokahwilapamjukatuvor
-----------------------------------
```

(continues on next page)

(continued from previous page)

```
Aide-toi, le ciel t'aidera
Fmjf-voi, ss uftf p'ufeqqc
Aide-toi, le ciel t'aidera
-----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import TwoSquare, CryptMachine, alphabets
from secretpy.cmdecorators import UpperCase


def encdec(machine, plaintext):
    print(plaintext)
    enc = machine.encrypt(plaintext)
```

(continues on next page)

(continued from previous page)

```
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)
    print("----------------------------------")


alphabet = alphabets.ENGLISH_SQUARE_OQ

key = (u"example", u"keyword")

cm = UpperCase(CryptMachine(TwoSquare()))

cm.set_alphabet(alphabet)
cm.set_key(key)
plaintext = u"Help me Obi wan Kenobi"
encdec(cm, plaintext)

plaintext = u"Help me Obi wan Kenobi y"
encdec(cm, plaintext)

'''
Help me Obi wan Kenobi
XGDLXWSDJYRYHOTKDG
HELPMEOBIWANKENOBI
----------------------------------
Help me Obi wan Kenobi y
XGDLXWSDJYRYHOTKDGZX
HELPMEOBIWANKENOBIYZ
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Vic
from secretpy import CryptMachine


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


key = "0452"
cm = CryptMachine(Vic(), key)
alphabet = [
    u"e", u"t", u"", u"a", u"o", u"n", u"", u"r", u"i", u"s",
    u"b", u"c", u"d", u"f", u"g", u"h", u"j", u"k", u"l", u"m",
    u"p", u"q", u"/", u"u", u"v", u"w", u"x", u"y", u"z", u".",
]
plaintext = u"attackatdawn"
cm.set_alphabet(alphabet)
encdec(cm, plaintext)

'''
Output:

attackatdawn
anwhrsanroaeer
attackatdawn
----------------------------------
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Vigenere, alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = u"kss"

cipher = Vigenere()
print(plaintext)

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

#########################################################

print("-----------------------------------")

plaintext = u"attackatdawn"
key = u"lemon"

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

'''
```

(continues on next page)

(continued from previous page)

```
thequickbrownfoxjumpsoverthelazydog
ßzwäiämütöckxxcdöiwdgyjwöhzoßsfmvyy
thequickbrownfoxjumpsoverthelazydog
----------------------------------
attackatdawn
lxfopvefrnhr
attackatdawn
'''
```

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

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-

from secretpy import Zigzag


plaintext = u"thequickbrownfoxjumpsoverthelazydog"
plaintext = u"thequick"
key = 3

chipher = Zigzag()
```

(continues on next page)

(continued from previous page)

```
print(plaintext)

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

##########################################################

print("-----------------------------------")

plaintext = u"wearediscoveredfleeatonce"

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

##########################################################

print("-----------------------------------")

plaintext = u"defendtheeastwallofthecastle"
key = 4

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

'''
thequickbrownfoxjumpsoverthelazydog
tubnjsrldhqikrwfxupoeteayoecoomvhzg
thequickbrownfoxjumpsoverthelazydog
---------------------------------
wearediscoveredfleeatonce
wecrlteerdsoeefeaocaivden
wearediscoveredfleeatonce
---------------------------------
defendtheeastwallofthecastle
dttfsedhswotatfneaalhcleelee
defendtheeastwallofthecastle
'''
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

F

G

K

M

N

P

R

S

T

V

Z