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
- Simple Substitution

- Transposition: Columnar, Scytale, Spiral, 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)

'''
Output:
```

(continues on next page)

(continued from previous page)

```
============================================================================================
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 at 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))


key = 3
cipher = Caesar()
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 is in ENGLISH
- **alphabet** (*string*) – Alphabet which will be used(length=25), 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 is in ENGLISH
- **alphabet** (*string*) – Alphabet which will be used(length=25), if there is no a value, ENGLISH_SQUARE_IJ is used

Returns text

Return type string

2.1.1 Examples

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

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


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

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


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


cm0 = CryptMachine(cipher, key)
cm = UpperCase(Block(cm0))
cm.set_alphabet(alphabet)
plaintext = u"Attack at once!"
encdec(cm, plaintext)

cm1 = SaveAll(cm0)
plaintext = u"Attack at once!"
encdec(cm1, plaintext)

alphabet = al.GERMAN_SQUARE
key = "german"
cm.set_key(key)
cm.set_alphabet(alphabet)
plaintext = u"Schweißgequält vom öden Text zürnt Typograf Jakob"
encdec(cm, plaintext)

alphabet = (
    u"uúů", u"aá", u"b", u"cč", u"dd'",
    u"f", u"g", u"h", u"iíj", u"zž",
    u"k", u"l", u"m", u"nň", u"oó",
    u"p", u"q", u"rř", u"sš", u"tt'",
```

(continues on next page)

(continued from previous page)

```
    u"v", u"w", u"x", u"yý", u"eéě",
)
key = "czech"
cm.set_key(key)
cm.set_alphabet(alphabet)
plaintext = u"Necht' již hříšné saxofony d'áblů rozezvučí síň úděsnými tóny waltzu, tanga a quickstepu."
encdec(cm, plaintext)

alphabet = (
    u"zž", u"aáä", u"b", u"cč", u"dd'",
    u"eé", u"f", u"g", u"h", u"iíj",
    u"k", u"lĺl'", u"m", u"oóô", u"p",
    u"q", u"rŕ", u"sš", u"tt'", u"uú",
    u"v", u"w", u"x", u"yý", u"nň",
)
key = "slovak"
cm.set_key(key)
cm.set_alphabet(alphabet)
plaintext = u"Vypätá dcéra grófa Maxwella s IQ nižším ako kôň núti čel'ad' hrýzt' hŕbu jabĺk."
encdec(cm, plaintext)

alphabet = (
    u"α", u"β", u"γ", u"δ", u"ϵ",
    u"ζ", u"η", u"θ", u"ι", u"κ",
    u"λ", u"μ", u"ν", u"ξ", u"ο",
    u"π", u"ρ", u"σ", u"τ", u"υ",
    u"ϕ", u"χ", u"ψ", u"ω", u"",
)
key = "greek"
cm.set_key(key)
cm.set_alphabet(alphabet)
plaintext = u"Ξεσκεπζω την ψυχοφθρα σα βδελυγμα."
encdec(cm, plaintext)

'''
attackatonce
faxdfadddgdgfffafaxafafx
attackatonce
----------------------------------
Attack at once!
FAXDF ADDDG DGFFF AFAXA FAFX
ATTACKATONCE
----------------------------------
Attack at once!
Faxdfa dd dgdg!fffafaxafafx
Attack at once!
----------------------------------
Schweißgequält vom öden Text zürnt Typograf Jakob
DDAAX FFXGG FGDFF DFAAG GGGDG GAADG XGGFF AGGGG FAAAF XDXGD XXXFG DAXFG XAAGF FXGXD GGAAD GGFAA XFXDD D
SCHWEISGEQUALTVOMODENTEXTZURNTTYPOGRAFIAKOB
----------------------------------
Necht' již hříšné saxofony d'áblů rozezvučí síň úděsnými tóny waltzu, tanga a quickstepu.
FGDXD GAXFX FFXAD GAGFX XDDXD DDAXA XGGGG GFFGA ADXAG AXXGF DGAFD AGGAX FDFGX XAXDA XDAGG XGDXX DADAD AGGAX DFFGF XAFGX XGDAG GGGAX GGAAF XAGDG DGXDD GADFX ADFAX FFGDA ADGGG X
```

(continues on next page)

(continued from previous page)

```
NECHTIIZHRISNESAXOFONYDABLUROZEZVUCISINUDESNYMITONYWALTZUTANGAAQUICKSTEPU
----------------------------------
Vypätá dcéra grófa Maxwella s IQ nižším ako kôň núti čel'ad' hrýzt' hŕbu jabĺk.
FADDD ADAGA FFXGD AXDGA XDAFD DADAA FGXGA XGGXF ADXDD DFDFX FDAXX DGADX DXGAA FFXFD␣
↪DDFFG AAGGA AFXAA GGAXF GXGAF XDFDA GDFGG GDGFD DXXXA GXGDD GFDA
VYPATADCERAGROFAMAXWELLASIQNIZSIMAKOKONNUTICELADHRYZTHRBUIABLK
----------------------------------
Ξεσκεπζω την ψυχοφθορα σα βδελυγμα.
AXAGF XXXGF AXDXA AGFXA GFAXA GFFGA DFFFA AAAFA GXDGX DDDAD FFAGD AXDGX FAGGG D
ΞΕΣΚΕΠΑΖΩΤΗΝΨΥΧΟΦΘΟΡΑΣΑΣΒΔΕΛΥΓΜΙΑ
----------------------------------
'''
```

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 with numbers is used

Returns text

Return type string

2.2.1 Examples

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

from secretpy import ADFGVX, CryptMachine
from secretpy.cmdecorators import Block, UpperCase
```

(continues on next page)

(continued from previous page)

```
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 = "privacy"
17  cm = UpperCase(Block(CryptMachine(ADFGVX(), key), 4))
18
19  alphabet = (
20      u"n", u"a", u"1", u"c", u"3", u"h",
21      u"8", u"t", u"b", u"2", u"o", u"m",
22      u"e", u"5", u"w", u"r", u"p", u"d",
23      u"4", u"f", u"6", u"g", u"7", u"i",
24      u"9", u"j", u"0", u"k", u"l", u"q",
25      u"s", u"u", u"v", u"x", u"y", u"z",
26  )
27  cm.set_alphabet(alphabet)
28  plaintext = "Attack at 12.00am!"
29  encdec(cm, plaintext)
30
31  key = "pangram"
32  cm.set_key(key)
33  plaintext = "The quick brown fox jumps over the lazy dog."
34  encdec(cm, plaintext)
35
36  alphabet = (
37      u"5", u"", u"", u"", u"", u"",
38      u"", u"", u"", u"", u"", u"",
39      u"", u"", u"", u"", u"", u"",
40      u"", u"", u"", u"", u"", u"",
41      u"", u"", u"", u"", u"", u"",
42      u"0", u"1", u"2", u"3", u"4", u"",
43  )
44  cm.set_alphabet(alphabet)
45  key = "russian"
46  cm.set_key(key)
47  plaintext = u", !  ?     ."
48  encdec(cm, plaintext)
49
50  '''
51  Attack at 12.00am!
52  DGDD DAGD DGAF ADDF DADV DVFA ADVX
53  ATTACKAT1200AM
54  --------------------------------
55  The quick brown fox jumps over the lazy dog.
56  DXGF VDVD VFAA GGDX AFXG XGFA GFFA DDVG DDXA FAXG ADDF XXXD AXDX VVDD DGVV FXFA VVFX␣
    ↪XV
57  THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG
58  --------------------------------
59  , !  ?     .
60  AAXF FGXG GDDF FVDA FDDG GGVF DGXV VAAV VFXA XDDA DGAV AGDF AXFV VFDX GFVD XFVV FG
61
```

(continues on next page)

(continued from previous page)

```
--------------------------------
'''
```

2.3 Affine

class secretpy.**Affine**
The Affine Cipher

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

Parameters

- **text** (*string*) – Text to decrypt
- **key** (*tuple of 2 integers*) – 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** (*tuple of 2 integers*) – 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, CryptMachine, alphabets as al
from secretpy.cmdecorators import UpperCase, Block, SaveAll


alphabet = al.ENGLISH
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = (7, 8)

cipher = Affine()
print(plaintext)

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

(continues on next page)

(continued from previous page)

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


def encdec(machine, plaintext):
    print("=" * 80)
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    print(machine.decrypt(enc))


cm0 = CryptMachine(cipher, key)

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

cm = Block(cm, length=4, sep="/=/")
cm.set_alphabet(al.SWEDISH + al.DECIMAL)
plaintext = "This text is divided by blocks of length 4!"
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)
plaintext = u"text          !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
cm.set_alphabet(alphabet)
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
thequickbrownfoxjumpsoverthelazydog
lfkqsmwapxcgvrcntsojeczkxlfkhibudcy
thequickbrownfoxjumpsoverthelazydog
================================================================================
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
mdcvlhczkvcvihjfipklwfixiwlkxemgmhhxkoczkihhcrlfkoyxkil
idontlovenonalphabetcharactersiwillremovealloftheмgreat
================================================================================
This text is divided by blocks of length 4!
yszr/=/y7ny/=/zr0z/=/9z07/=/0pup/=/höwa/=/röeh/=/7vly/=/sf
thistextisdividedbyblocksoflength4
```

(continues on next page)

(continued from previous page)

```
================================================================================
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
Z hö97 vöv-ih6sip7y wsikiwy7kr. Ys7r7 ik7 : ^,&@$~(*;?&#. Ysiy'r zy!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
================================================================================
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Pp qmxzlmc Pp qmxircm fm rcm MSADPCH_CORIVM_PP!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
================================================================================
text         !
text         !
text         !
================================================================================
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΛΟΘΙΡ ΗΦΙΝΖ ΓΗΡ ΝΚΘΞΑ Α ΙΘΙΣΛΙΦΧΗ. (ΗΤΔΦΟΗ ΓΜΘΣ)
ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΒΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
'''
```

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*) – is not used
- **alphabet** (*string or tuple of strings*) – 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*) – is not used
- **alphabet** (*string or tuple of strings*) – 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, CryptMachine, alphabets
from secretpy.cmdecorators import UpperCase, SaveAll


def encdec(machine, plaintext):
    print("-" * 80)
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)


cm = UpperCase(CryptMachine(Atbash()))

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

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

cm = SaveAll(cm)
plaintext = u"The quick brown fox jumps over the lazy dog."
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)

'''
--------------------------------------------------------------------------------
attackatdawn
ZGGZXPZGWZDM
ATTACKATDAWN
--------------------------------------------------------------------------------



--------------------------------------------------------------------------------
The quick brown fox jumps over the lazy dog.
KWZ NJVÖT ÜMPHQ YPG UJROL PIZM KWZ SßEF ÄPX.
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
--------------------------------------------------------------------------------
Achtung Minen!!!
ßÖWKJQX RVQZQ!!!
ACHTUNG MINEN!!!
--------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
    :
    :
    :
    '''
```

2.5 Autokey

class secretpy.**Autokey**
The Autokey Cipher

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

Parameters

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string or tuple of strings*) – 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 or tuple of strings*) – 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, CryptMachine, alphabets as al
from secretpy.cmdecorators import UpperCase, Block, SaveAll


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

cipher = Autokey()
print(plaintext)
```

(continues on next page)

(continued from previous page)

```

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


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 + al.DECIMAL)
cm.set_key("keys")
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(u"")
plaintext = u"text          !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
cm.set_alphabet(alphabet)
cm.set_key(u"κλειδ")
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
thequickbrownfoxjumpsoverthelazydog
föiudtäßivamvhyyäeeüxüonhbwwzvß1wvk
thequickbrownfoxjumpsoverthelazydog
--------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
shc51o28xy28ey3uamt0cieacjtvru10z3tdmxzcimz6sf4ssrzyimz
idontlovenonalphabetcharactersiwillremoveallofthemgreat
--------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
This text is divided by blocks of length 5!
3l6ac-l5b1w-0130g-myj1f-op0l3-2hvw1-l4li
thistextisdividedbyblocksoflength5
------------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
S pcdm y28-ey3uamt0 cieacjtvru. Clvax hvw : ^,&@$~(*;?&#. Xhrx'b pt!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Sn zwlwniu Wu fwlivwg wy mwa IGYPNEO_CYMHIU_CI!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------------
text         !
text         !
text         !
------------------------------------------------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΣΠNΜ IΛ TAA ΨIΣXH  ΨHOKE. (ΥΘOAΣ ΨEΓHOΛ)
ΛEI AΣET KAΘ TKMH  EΛΥΘΔP. (ANEPΣ KAΛBΞT)
'''
```

2.6 Bazeries

class secretpy.**Bazeries**

The Bazeries 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

2.6.1 Examples

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

from secretpy import Bazeries, CryptMachine, alphabets as al
from secretpy.cmdecorators import UpperCase, SaveAll


def encdec(machine, plaintext):
    print("=" * 80)
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)


cm = UpperCase(CryptMachine(Bazeries()))
cm.set_alphabet(al.ENGLISH_SQUARE_IJ)

key = (81257, u"eightyonethousandtwohundredfiftyseven")
cm.set_key(key)
plaintext = u"Whoever has made a voyage up the Hudson" \
            u" must remember the Kaatskill mountains"
encdec(cm, plaintext)

cm = SaveAll(cm)
encdec(cm, plaintext)

cm = UpperCase(SaveAll(CryptMachine(Bazeries(), key)))
cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
plaintext = u"The quick brown fox jumps over the lazy dog!"
encdec(cm, plaintext)

cm.set_alphabet(al.GREEK_SQUARE)
cm.set_key((2358, u"κλειδ"))
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
================================================================================
Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
DUMTMCDSENRTEMVEQXMOELCCRVXDMDKWXNNMUKRDKUMYNMBPRKEEPMGNGEKWXCRWB
WHOEVERHASMADEAVOYAGEUPTHEHUDSONMUSTREMEMBERTHEKAATSKILLMOUNTAINS
================================================================================
Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
DUMTMCD SEN RTEM V EQXMOE LC CRV XDMDKW XNNM UKRDKUMY NMB PRKEEPMGN GEKWXCRWB
WHOEVER HAS MADE A VOYAGE UP THE HUDSON MUST REMEMBER THE KAATSKILL MOUNTAINS
================================================================================
The quick brown fox jumps over the lazy dog!
PAB XHMDK YCUFC IWS TCRQN XBZE GMD KUML CVO!
THE QUICK BROWN FOX IUMPS OVER THE LAZY DOG!
================================================================================
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΥMPΥE BΨΥZK BΓE XΨPKΦ Υ BΔΥEKPZΥM. (ΦΣKΥZΠN KEKΣXA)
ΘEΛEI APETH KAI TOΛMH H EΛEΥΘEPIA. (ANΔPEAΞ KAΛBOΞ)
'''
```

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"
```

(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)

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
```

(continues on next page)

(continued from previous page)

```
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 as al
from secretpy.cmdecorators import UpperCase, SaveAll, Block


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"",
    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)

cm1 = Block(cm)

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"
]
cm1.set_alphabet(alphabet)
plaintext = u"Defend the East Wall of the Castle!"
encdec(cm1, plaintext)

cm1 = UpperCase(cm1)
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"
]
cm1.set_alphabet(alphabet)
cm1.set_key(10)
plaintext = "flee at once"
encdec(cm1, plaintext)

```

(continues on next page)

(continued from previous page)

```
cm = SaveAll(cm)

alphabet = al.GREEK_SQUARE
cm.set_alphabet(alphabet)
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
 , !!!
4

----------------------------------
Defend the East Wall of the Castle!
ffyhm khycp liash adtrl hcchl blr
defendtheeastwallofthecastle
----------------------------------
flee at once
UAEOL WRINS
FLEEATONCE
----------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
Ζλζπρ οεπκφ ζιν μζυτυ η κλφδζγεγφ. (Πγδαργγ Ρυτακ)
Θελει αρετη και τολμη η ελευθερια. (Ανδρεα Καλβο)
----------------------------------
'''
```

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()

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)

```

(continues on next page)

(continued from previous page)

```
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)

alphabet = u"abcdeABCDEfghijFGHIJ"
cm.set_alphabet(alphabet)
cm.set_key(3)
plaintext = u"Text aBcdeHijf"
encdec(cm, plaintext)

'''
Output:

==========================================================================================
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
------------------------------------------------------------------------
Text aBcdeHijf
TCxt dEABCaGHi
Text aBcdeHijf
'''
```

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"
```

(continues on next page)

(continued from previous page)

```
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)

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!
```

(continues on next page)

(continued from previous page)

```
----------------------------------------------------------------------
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, SaveAll


def encdec(machine, plaintext):
```

(continues on next page)

(continued from previous page)

```
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    print(machine.decrypt(enc))
    print("--------------------------------")


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

cm = UpperCase(SaveAll(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)

'''
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, alphabets as al
from secretpy.cmdecorators import SaveAll, Block


cipher = ColumnarTransposition()
alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = u"schlüssel"

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


def encdec(machine, plaintext):
    print("=" * 80)
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    print(machine.decrypt(enc))


key = "manykeys"
cm0 = CryptMachine(cipher, key)

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

(continues on next page)

(continued from previous page)

```
35
36  cm = Block(cm, length=5, sep=" ")
37  plaintext = "This text is divided by blocks of length 5!"
38  encdec(cm, plaintext)
39
40  cm = SaveAll(cm0)
41  plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
42  encdec(cm, plaintext)
43
44  cm.set_alphabet(al.JAPANESE_HIRAGANA)
45  cm.set_key(u"")
46  plaintext = u"text  "
47  encdec(cm, plaintext)
48
49  cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
50  cm.set_key(u"keyj")
51  plaintext = "ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition␣
    ↪cipher!"
52  encdec(cm, plaintext)
53
54  '''
55  schweißgequältvomödentextzürnttypografjakob
56  cuenfgmzghäntjwlttaeöürsqdraivxpoßotobeteyk
57  schweißgequältvomödentextzürnttypografjakob
58  ================================================================================
59  I don't love non-alphabet characters and uppercase. I will remove all of them: ^,&@$~
    ↪(*;?&#.
60  dnbcuemfllhsrlamtacreieeieaadseoooetpiotvhrnarlnntepwvhopaacll
61  idontlovenonalphabetcharactersanduppercaseiwillremoveallofthem
62  ================================================================================
63  This text is divided by blocks of length 5!
64  hsboe iontv letid shidy ftekt siblx dcg
65  thistextisdividedbyblocksoflength
66  ================================================================================
67  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
68  L lhse nbc-steaaeai acrrtopatt. Nteai vhr : ^,&@$~(*;?&#. Hhoe't es!
69  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
70  ================================================================================
71  text
72  text
73  text
74  ================================================================================
75  ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition cipher!
76  NSUIECG_TBUTRP_TC elsrd'n aijcea Ns op Ei qeotnij ai't s inhghajsheoes saoiir!
77  ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition cipher!
78  '''
```

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"
```

(continues on next page)

(continued from previous page)

```
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)


'''
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** (*tuple of integers*) – 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** (*tuple of integers*) – 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, CryptMachine, alphabets as al
from secretpy.cmdecorators import UpperCase, Block, SaveAll


alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = (4, 17, 9)

cipher = Gronsfeld()

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

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


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


key = (14, 2, 11)
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=" ")
cm.set_key((1, 12, 7, 2))
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)
plaintext = u"text          !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
cm.set_alphabet(alphabet)
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
schweißgequältvomödentextzürnttypografjakob
wtqävrdxnuhfpgasßghvwxvcxmhvaüxlysxäewseöxf
schweißgequältvomödentextzürnttypografjakob
------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
wfzbvwcxpbqyonavcmsvnvccoeestdwytzncsozjglznztvssorfglh
idontlovenonalphabetcharactersiwillremoveallofthemgreat
------------------------------------------------------------------
This text is divided by blocks of length 5!
utpuu qevje kkwuk genfd majmt amnfz nvi
thistextisdividedbyblocksoflength
------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
J xvxf zvp-bxwjbnlv dthtboagse. Ajfel csq : ^,&@$~(*;?&#. Ajbf'z ku!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Kv igdatgt Vq dfphwtr dg vem GOTSLTU_ZSVNYG_KV!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------
text          !
text          !
text          !
------------------------------------------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
IOPZ XZΥP ΠB ΔNNΠ M ZMΞAB. (XIΣZΩ MBΠT)
ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΠΑΣ ΚΛΒΟΣ)
'''
```

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


alphabet = al.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)


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

cm = UpperCase(Block(cm, length=5, sep="-"))
```

(continues on next page)

(continued from previous page)

```
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(u"")
plaintext = u"text         !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
cm.set_alphabet(alphabet)
cm.set_key(u"κλειδ")
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
thequickbrownfoxjumpsoverthelazydog
rblmscefuojviyjwdshkpjtlorblgqzxnja
thequickbrownfoxjumpsoverthelazydog
------------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
c6jirgju7iji3glb347r5b3p35r7pqcvcggp7hju73ggj8rb7h9p73r
idontlovenonalphabetcharactersiwillremoveallofthemgreat
------------------------------------------------------------------------
This text is divided by blocks of length 5!
RBCQR-7WRCQ-6CUC6-764X4-GJ5FQ-J8G7I-9RBE
THISTEXTISDIVIDEDBYBLOCKSOFLENGTH5
------------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
C gju7 iji-3glb347r 5b3p35r7pq. Rb7q7 3p7 : ^,&@$~(*;?&#. Rb3r'q cr!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Cc aknigkq Cc aknmtqk vk tqk KHSFCQB_QOTMPK_CC!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------------
text         !
text         !
text         !
------------------------------------------------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΑΘΖ ΚΡΤΓ ΚΖ ΤΘΜΒ Β ΘΥΡΗΚ. (ΚΝΔΡΑΚΣ ΛΘΕΟΣ)
ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
'''
```

2.16 MyszkowskiTransposition

class secretpy.**MyszkowskiTransposition**
The Myszkowski Transposition 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

2.16.1 Examples

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

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

alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = u"schlüssel"

cipher = MyszkowskiTransposition()
print(plaintext)

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


def encdec(machine, plaintext):
    print("------------------------------------------------------------------------")
```

(continues on next page)

(continued from previous page)

```
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    print(machine.decrypt(enc))


cm0 = CryptMachine(cipher, key)

cm = cm0
cm.set_alphabet(al.ENGLISH)
cm.set_key("tomatokey")
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.JAPANESE_HIRAGANA)
cm.set_key(u"")
plaintext = u"text         !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
cm.set_alphabet(alphabet)
cm.set_key(u"κλειδ")
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

cm.set_key(u"kryptonyckeln")
cm.set_alphabet(al.SWEDISH)
plaintext = u"FAQ om Schweiz: Klöv du trång pjäxby?"
encdec(cm, plaintext)

'''
schweißgequältvomödentextzürnttypografjakob
cuenfgmzghäntjwelötütrasißqvodxtrpoaobeteyk
schweißgequältvomödentextzürnttypografjakob
------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
nahivevaclleohallroncsohdloptrrimatgitnleaeweefmtebtroa
idontlovenonalphabetcharactersiwillremoveallofthemgreat
------------------------------------------------------------------
This text is divided by blocks of length 5!
svogt-doxes-iilnh-eddbk-ehtts-iyclt-ibf
thistextisdividedbyblocksoflength
------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
V acei ncs-totreoha sslnpereer. Tielb ath : ^,&@$~(*;?&#. Aata'h th!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
text          !
text          !
text          !
------------------------------------------------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
IKMϒA ΣΣΛT Λ BEΛE A AOAAHΘNKΘ. (PIHEΔ ETEPPΛ)
ΘΛEI APET KAI TΛMH H EΛEϒΘEPA. (ANΔPAΣ KΛBOΣ)
------------------------------------------------------------------------
FAQ om Schweiz: Klöv du trång pjäxby?
WNI PF ELGXZJC: KRÄS TO DAÖBM UQHVÅY?
FAQ OM SCHWEIZ: KLÖV DU TRÅNG PJÄXBY?
'''
```

2.17 Nihilist

class secretpy.**Nihilist**

The Nihilist 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*) – 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

2.17.1 Examples

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

```

(continues on next page)

(continued from previous page)

```
from secretpy import Nihilist, CryptMachine, alphabets as al


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


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


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


key = "mykey"
cm = CryptMachine(cipher, 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",
    u"n", u"o", u"p", u"q", u"t",
    u"u", u"v", u"w", u"x", u"y"
)
cm.set_alphabet(alphabet)
plaintext = u"Waltz, bad nymph, for quick jigs vex."
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",
)
cm.set_alphabet(alphabet)
key = "freedom"
cm.set_key(key)
plaintext = u"Meet thursday 2300hr!"
encdec(cm, plaintext)

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

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

'''
Output:

thequickbrownfoxjumpsoverthelazydog
57 54 36 61 66 64 35 40 44 57 59 68 73 38 48 78 45 69 54 75 63 48 76 36 62 65 63 37
↪41 43 73 77 37 74 43
thequickbrownfoxjumpsoverthelazydog
----------------------------------
Waltz, bad nymph, for quick jigs vex.
88 70 67 57 66 48 70 56 53 110 70 98 64 36 97 49 99 84 44 77 68 87 65 37 76 87 67 87
waltzbadnymphforquickiigsvex
----------------------------------
Meet thursday 2300hr!
47 51 30 57 56 55 74 52 77 29 26 65 88 87 69 89 37 51
meetthursday2300hr
----------------------------------

60 41 79 80 60

----------------------------------
ΠΙΝΑΚΑΣ
95 78 87 65 79 65 97
πινακασ
----------------------------------
'''
```

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


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)
```

(continues on next page)

(continued from previous page)

```
'''
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=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

2.19.1 Examples

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

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


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


key = "mykey"
cm = CryptMachine(Polybius(), key)

plaintext = u"Defend the east wall of the castle"
encdec(cm, plaintext)

cm = SaveAll(cm)

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)
encdec(cm, plaintext)

plaintext = "thisisasecretmessage"
encdec(cm, plaintext)

cm.set_alphabet(al.GREEK)
cm.set_key(u"πινακασ")
plaintext = u"Ξεσκεπζω την ψυχοφθρα σα βδελυγμα."
encdec(cm, plaintext)

'''
Defend the east wall of the castle
2214231433224325141415424346153232342343251421154243321 4
defendtheeastwallofthecastle
----------------------------------
Defend the east wall of the castle
341433 143 1345 4211 41 424 44545124252532335421144224445425 14
defend the east wall of the castle
----------------------------------
thisisasecretmessage
5421554455442444144241145411144444242314
thisisasecretmessage
----------------------------------
```

(continues on next page)

(continued from previous page)

```
Ξεσκεπζω την ψυχοφθρα σα βδελυγμα.
422516152 511 213161513 213 565255435.434444514161446222425365223413514
ξεσκεπζω την ψυχοφθρα σα βδελυγμα.
---------------------------------
'''
```

2.20 Porta

class secretpy.**Porta**

The Porta 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

2.20.1 Examples

```
#!/usr/bin/python
# -*- encoding: utf-8 -*-
from secretpy import Porta, CryptMachine, alphabets as al
from secretpy.cmdecorators import UpperCase, Block, SaveAll


alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = u"schlüssel"

cipher = Porta()
print(plaintext)
```

(continues on next page)

(continued from previous page)

```
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
22  def encdec(machine, plaintext):
23      print("--------------------------------------------------------------------------")
24      print(plaintext)
25      enc = machine.encrypt(plaintext)
26      print(enc)
27      print(machine.decrypt(enc))
28
29
30  cm0 = CryptMachine(cipher, key)
31
32  cm = cm0
33  cm.set_alphabet(al.ENGLISH)
34  cm.set_key("keys")
35  plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
    ↪(*;?&#. Great!"
36  encdec(cm, plaintext)
37
38  cm = Block(cm, length=4, sep="::")
39  plaintext = "This text is divided by blocks of length 5!"
40  encdec(cm, plaintext)
41
42  cm = SaveAll(cm0)
43  plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44  encdec(cm, plaintext)
45
46  cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
47  plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
48  encdec(cm, plaintext)
49
50  cm.set_alphabet(al.JAPANESE_HIRAGANA)
51  cm.set_key(u"")
52  plaintext = u"text         !"
53  encdec(cm, plaintext)
54
55  cm = UpperCase(cm)
56  alphabet = al.GREEK
57  cm.set_alphabet(alphabet)
58  cm.set_key(u"κλειδ")
59  plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
60  encdec(cm, plaintext)
61
62  '''
63  '''
```

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*) – not used
- **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*) – not used
- **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, Block


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


cm = CryptMachine(Rot13())

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

```

(continues on next page)

(continued from previous page)

```
cm = SaveAll(cm)

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

plaintext = u"Die heiße Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis zur Küste."
cm.set_alphabet(al.GERMAN)
encdec(cm, plaintext)

plaintext = u"FAQ om Schweiz: Klöv du trång pjäxby?"
cm.set_alphabet(al.SWEDISH)
encdec(cm, plaintext)

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

cm = Block(cm)
encdec(cm, plaintext)

'''
----------------------------------
The quick brown fox jumps over the lazydog.
gurdhvpxoebjasbkwhzcfbiregurynmlqbt
thequickbrownfoxjumpsoverthelazydog
----------------------------------
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
Why did the chicken cross the road? Gb trg gb gur bgure fvqr
----------------------------------
Die heiße Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis zur Küste.
Sxt wtxot Kjatcüdßüüt bfläet Öpi füs Gxreßcxp yp qmdt pfu stö Htv qxd kfc Zndet.
Die heiße Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis zur Küste.
----------------------------------
FAQ om Schweiz: Klöv du trång pjäxby?
UPB oä Drwhtxk: Zång sf eclöv aymiqj?
FAQ om Schweiz: Klöv du trång pjäxby?
----------------------------------




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



'''
```

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 -*-

from secretpy import Rot5, alphabets, CryptMachine


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


cm = CryptMachine(Rot5())

plaintext = "My text " + alphabets.DECIMAL + " your text"
encdec(cm, plaintext)

'''
----------------------------------
My text 0123456789 your text
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*) – is not used
- **alphabet** (*string*) – is not used

Returns text

Return type string

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

Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – is not used
- **alphabet** (*string*) – is not used

Returns text

Return type string

2.23.1 Examples

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

from secretpy import Rot18, CryptMachine
from secretpy.cmdecorators import SaveAll, UpperCase, Block


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


plaintext = u"The man has 536 dogs! How many dogs do you have?"

cm = CryptMachine(Rot18())
encdec(cm, plaintext)

cm1 = Block(cm)
encdec(cm1, plaintext)
```

(continues on next page)

(continued from previous page)

```
cm = SaveAll(cm)
encdec(cm, plaintext)

cm = UpperCase(cm)
encdec(cm, plaintext)

'''
---------------------------------
The man has 536 dogs! How many dogs do you have?
gurznaunf081qbtfubjznalqbtfqblbhunir
themanhas536dogshowmanydogsdoyouhave
---------------------------------
The man has 536 dogs! How many dogs do you have?
gurzn aunf0 81qbt fubjz nalqb tfqbl bhuni r
themanhas536dogshowmanydogsdoyouhave
---------------------------------
The man has 536 dogs! How many dogs do you have?
Gur zna unf 081 qbtf! Ubj znal qbtf qb lbh unir?
The man has 536 dogs! How many dogs do you have?
---------------------------------
The man has 536 dogs! How many dogs do you have?
GUR ZNA UNF 081 QBTF! UBJ ZNAL QBTF QB LBH UNIR?
THE MAN HAS 536 DOGS! HOW MANY DOGS DO YOU HAVE?
'''
```

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, CryptMachine
from secretpy.cmdecorators import SaveAll


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


plaintext = u"The Quick Brown Fox Jumps Over The Lazy Dog."

cm = CryptMachine(Rot47())
encdec(cm, plaintext)

cm = SaveAll(cm)
encdec(cm, plaintext)

'''
----------------------------------
The Quick Brown Fox Jumps Over The Lazy Dog.
%96"F:4<qC@H?u@IyF>AD~G6C%96{2KJs@8]
TheQuickBrownFoxJumpsOverTheLazyDog.
----------------------------------
The Quick Brown Fox Jumps Over The Lazy Dog.
%96 "F:4< qC@H? u@I yF>AD ~G6C %96 {2KJ s@8]
The Quick Brown Fox Jumps Over The Lazy Dog.
'''
```

2.25 Scytale

class secretpy.**Scytale**

The Scytale Cipher

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

Decryption method

Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key - Number of windings
- **alphabet** – Alphabet is not used

Returns decrypted text

Return type string

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

Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key - Number of windings
- **alphabet** – Alphabet is not used

Returns encrypted text

Return type string

2.25.1 Examples

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

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


alphabet = al.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)


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


cm0 = CryptMachine(cipher, key)
cm0.set_alphabet(al.ENGLISH + al.DECIMAL)

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

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

(continues on next page)

(continued from previous page)

```
38
39  cm = SaveAll(cm0)
40  plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
41  encdec(cm, plaintext)
42
43  cm.set_alphabet(al.JAPANESE_HIRAGANA)
44  plaintext = u"text         !"
45  encdec(cm, plaintext)
46
47  cm = UpperCase(cm)
48  alphabet = al.GREEK
49  cm.set_alphabet(alphabet)
50  plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
51  encdec(cm, plaintext)
52
53  '''
54  thequickbrownfoxjumpsoverthelazydog
55  tqcrnxmorezohukofjpvtlygeibwousehad
56  thequickbrownfoxjumpsoverthelazydog
57  ------------------------------------------------------------------------
58  I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
59  inonahehaeilevlfertdtvolatacrwlmeltmeolenpbcrtsiroaohga
60  idontlovenonalphabetcharactersiwillremoveallofthemgreat
61  ------------------------------------------------------------------------
62  This text is divided by blocks of length 5!
63  TSXSV-EYOSL-G5HTT-DIDBC-OETIE-IIDBL-KFNH
64  THISTEXTISDIVIDEDBYBLOCKSOFLENGTH5
65  ------------------------------------------------------------------------
66  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
67  I vola tac-rheeaile npbcrtseat. Ttona heh : ^,&@$~(*;?&#. Aets'r hs!
68  I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
69  ------------------------------------------------------------------------
70  text         !
71  text         !
72  text         !
73  ------------------------------------------------------------------------
74  Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
75  ΘΕΡΙ ΛΗΕΕΑ ΔΑ ΟΙΕΚ Τ ΜΕΥΡΑΡΣΛΣ. (ΛΑΤΑΗΛ ΘΝΚΒ)
76  ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
77  '''
```

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** (*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

2.26.1 Examples

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

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


cipher = SimpleSubstitution()
alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = u"fwxyääöeßüdabcglmnpqrstuvhjoikz"

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

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


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

(continues on next page)

(continued from previous page)

```
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)
key = (
    "n", "g", "a", "b", "l",
    "s", "t", "u", "v", "c",
    "m", "o", "p", "q", "h",
    "ij", "k", "w", "x", "y",
    "r", "d", "e", "f", "z"
)
cm.set_key(key)
plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
encdec(cm, plaintext)

alphabet = u"abcdABCDEfghijFGHIJ"
key = u"dABDFIJEfgCHabchijG"
cm.set_alphabet(alphabet)
cm.set_key(key)
plaintext = u"Text aBcdHijf"
encdec(cm, plaintext)

'''
schweißgequältvomödentextzürnttypografjakob
qxßuäüzeänsobrtlciyägrävrjkpgrrhmlepfödfalw
schweißgequältvomödentextzürnttypografjakob
------------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
kbnmxfnpcmnmyftjyzcxajyvyaxcvwkqkffvclnpcyffnhxjclivcyx
idontlovenonalphabetcharactersiwillremovealloftheмgreat
------------------------------------------------------------------------
This text is divided by blocks of length 5!
xjkwx-crxkw-bkpkb-cbzsz-fnaew-nhfcm-ixj
thistextisdividedbyblocksoflength
------------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
K fnpc mnm-yftjyzcx ajyvyaxcvw. Xjcwc yvc : ^,&@$~(*;?&#. Xjyx'w kx!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Vv glaqolw Vv glanywl dl ywl LPTMVWU_WIYNKL_VV!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------------
Text aBcdHijf
Text dIBDiabg
Text aBcdHijf
'''
```

2.27 Three Square

class secretpy.**ThreeSquare**

The Three Square 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** (*tuple of 3 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*, *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 3 strings*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

Returns text

Return type string

2.27.1 Examples

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

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


def encdec(machine, plaintext):
    print("=" * 80)
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)


key = (u"example", u"keyword", u"third")
cm = UpperCase(CryptMachine(ThreeSquare()))
cm.set_alphabet(al.ENGLISH_SQUARE_OQ)
```

(continues on next page)

(continued from previous page)

```
cm.set_key(key)
plaintext = u"The quick brown fox jumps over the lazy dog!"
encdec(cm, plaintext)

cm = SaveAll(cm)
encdec(cm, plaintext)

cm = SaveAll(CryptMachine(ThreeSquare()))
cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
key = (u"criptog", u"segurt", u"mars")
cm.set_key(key)
plaintext = u"Attack at dawns!!!"
encdec(cm, plaintext)

cm.set_alphabet(al.GREEK_SQUARE)
key = (u"κλειδ", u"κλειδ", u"λειδκ")
cm.set_key(key)
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

cm = UpperCase(Block(CryptMachine(ThreeSquare())))
cm.set_alphabet(al.GREEK_SQUARE)
key = (u"κλειδ", u"κλειδ", u"λειδκ")
cm.set_key(key)
encdec(cm, plaintext)

'''
================================================================================
The quick brown fox jumps over the lazy dog!
TPGEDELYICAWBACHSYNNJOSUJJTYRPSUWHWYINTBJELCBTXYSFONMV
THEOUICKBROWNFOXJUMPSOVERTHELAZYDOGZ
================================================================================
The quick brown fox jumps over the lazy dog!
TPJ LDONY IAAEX ARB SOUNH OSXS JVM RMMU EHW!OWNZBJEGCAKXKSFYUMU
THE OUICK BROWN FOX JUMPS OVER THE LAZY DOG!Z
================================================================================
Attack at dawns!!!
Actuac vs ihctz!!!ddnwosuz
Attack at dawns!!!z
================================================================================
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
Ινιχι ιλβρ σετ τβκλβ θ κσωμεμογο. (Γκνιιπυ Οζεσθλ)ζπγξδεσξληπλαβιβηνετψ
Θελει αρετη και τολμη η ελευθεριa. (Ανδρεα Καλβο)ω
================================================================================
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΠΝΚΛΙ ΕΙΛΒΒ ΣΙΗΤΓ ΜΛΓΑΚ ΣΔΜΕΚ ΟΗΩΓΙ ΣΙΚΠΥ ΜΖΕΠΥ ΛΒΠΓΘ ΤΕΡΞΛ ΖΕΠΕΘ ΒΚΚΗΞ ΞΤΨ
ΘΕΛΕΙΑΡΕΤΗΚΑΙΤΟΛΜΗΗΕΛΕΥΘΕΡΙΑΑΝΔΡΕΑΣΚΑΛΒΟΣΩ
'''
```

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 -*-

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",
```

(continues on next page)

(continued from previous page)

```
    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
----------------------------------
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)
    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
```

(continues on next page)

(continued from previous page)

```
----------------------------------
'''
```

2.30 Vic

class secretpy.**Vic**

The Vic 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** (*number 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=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** (*number string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

Returns text

Return type string

2.30.1 Examples

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

from secretpy import Vic, CryptMachine
from secretpy.cmdecorators import SaveAll


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

(continues on next page)

(continued from previous page)

```
14
15
16  key = "0452"
17  cm = SaveAll(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"Attack at dawn!"
24  cm.set_alphabet(alphabet)
25  encdec(cm, plaintext)
26
27  '''
28  Output:
29
30  ---------------------------------
31  Attack at dawn!
32  Anwhrs an roae!er
33  Attack at dawn!
34  '''
```

2.31 Vigenere

class secretpy.**Vigenere**

The Vigenere 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

2.31.1 Examples

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

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

alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
key = u"schlüssel"

cipher = Vigenere()
print(plaintext)

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


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)
cm.set_key("keys")
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(u"")
plaintext = u"text         !"
encdec(cm, plaintext)

cm = UpperCase(cm)
alphabet = al.GREEK
```

(continues on next page)

(continued from previous page)

```
cm.set_alphabet(alphabet)
cm.set_key(u"κλειδ")
plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
encdec(cm, plaintext)

'''
schweißgequältvomödentextzürnttypografjakob
geodcärkpewdwrjcqivguaclhßjfpäawdcküshqlict
schweißgequältvomödentextzürnttypografjakob
------------------------------------------------------------------
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
shmfdpmnormfkpnzkfclmlyjkgrwbwgospjjoqmnoejdyjrzoqejoer
idontlovenonalphabetcharactersiwillremoveallofthemgreat
------------------------------------------------------------------
This text is divided by blocks of length 5!
dlgkd-ivlsw-bafmb-wnfwt-vsacc-sddor-elr
thistextisdividedbyblocksoflength
------------------------------------------------------------------
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
S pmno rmf-kpnzkfcl mlyjkgrwbw. Rzowc sbi : ^,&@$~(*;?&#. Rzkx'q ad!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
------------------------------------------------------------------
Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Sn zwmskwb Ng togymbi uw dwc WWLIABM_QHDEPW_SN!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
------------------------------------------------------------------
text         !
text         !
text         !
------------------------------------------------------------------
Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
ΣΠΝΜ ΒΨΠ ΞΤ ΤΥΠ Ρ ΝΨΣΟΓΞΙ. (ΔΧΞΓΙΙΦ ΥΛΧΖΨΦ)
ΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
'''
```

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()
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)

'''
```

(continues on next page)

(continued from previous page)

```
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