SecretPy Documentation

Release 1.0

Read the Docs

May 24, 2019

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:

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

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

1.2 Installation

To install this library, you can use pip:

```
pip install secretpy
```

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

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

1.3 Usage

1.3.1 Direct way

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

```
from secretpy import Caesar
from secretpy import alphabets

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

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

print('=====================================')

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

(continues on next page)

(continued from previous page)

```
print(dec)

'''
Output:

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

1.3.2 CryptMachine

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

```
from secretpy import Atbash
from secretpy import Caesar

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


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


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

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

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

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

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

(continues on next page)

(continued from previous page)

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

'''
Output:

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

1.3.3 CompositeMachine

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

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


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


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

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

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

(continues on next page)

(continued from previous page)

```
print("========================================")

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

encdec(cm, plaintext)

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

'''
Output:

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

1.4 Maintainers

- @tigertv (Max Vetrov)

CHAPTER 2

API

2.1 ADFGX

class secretpy.**ADFGX**

The ADFGX Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.1.1 Examples

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

from secretpy import ADFGX

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

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

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

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

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

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

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

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

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

key = "deutsch"
plaintext = "howstuffworks"

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

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

2.2 ADFGVX

class secretpy.**ADFGVX**

The ADFGVX Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.2.1 Examples

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

from secretpy import ADFGVX, CryptMachine


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


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

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

(continues on next page)

(continued from previous page)

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

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

2.3 Affine

class secretpy.**Affine**

The Affine Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.3.1 Examples

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

from secretpy import Affine
```

(continues on next page)

(continued from previous page)

```python
from secretpy import alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = [7, 8]

cipher = Affine()
print(plaintext)

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

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

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

key = [3, 4]
plaintext = u"attackatdawn"

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

"""
thequickbrownfoxjumpsoverthelazydog
vögaüewsphqmjnqtlücxoqfghvögzidäßqu
thequickbrownfoxjumpsoverthelazydog
----------------------------------
attackatdawn
ejjekiejnesr
attackatdawn
"""
```

2.4 Atbash

class secretpy.**Atbash**

The Atbash Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.4.1 Examples

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

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


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


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

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

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

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

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

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

2.5 Autokey

class secretpy.**Autokey**

The Autokey Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.5.1 Examples

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

from secretpy import Autokey
from secretpy import alphabets

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

cipher = Autokey()
print(plaintext)

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

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

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

(continues on next page)

(continued from previous page)

```

plaintext = u"attackatdawn"

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

'''
thequickbrownfoxjumpsoverthelazydog
föiudtäßivamvhyyäeeüxüonhbwwzvßlwvk
thequickbrownfoxjumpsoverthelazydog
----------------------------------
attackatdawn
qnxepvytwtwp
attackatdawn
'''
```

2.6 Bazeries

class secretpy.**Bazeries**

The Bazeries Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.6.1 Examples

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

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


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


alphabet = alphabets.ENGLISH_SQUARE_IJ

key = (81257, u"eightyonethousandtwohundredfiftyseven")

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

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

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

2.7 Beaufort

class secretpy.**Beaufort**

The Beaufort Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.7.1 Examples

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

from secretpy import CryptMachine
from secretpy import Beaufort
from secretpy import alphabets

plaintext = u"helloworld"
key = "key"

cm = CryptMachine(Beaufort(), key)

print(plaintext)
enc = cm.encrypt(plaintext)
print(enc)
dec = cm.decrypt(enc)
print(dec)

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

alphabet = alphabets.GERMAN
cm.set_alphabet(alphabet)

print(plaintext)
enc = cm.encrypt(plaintext)
print(enc)
dec = cm.decrypt(enc)
print(dec)

'''
helloworld
danzqcwnnh
helloworld
-----------------------------------
helloworld
danßucärnh
helloworld
'''
```

2.8 Bifid

class secretpy.**Bifid**

The Bifid Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.8.1 Examples

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

from secretpy import Bifid
from secretpy import CryptMachine


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"",
```

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

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

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

2.9 Caesar

class secretpy.**Caesar**

The Caesar Cipher

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

Decryption method

Parameters

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

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

Returns decrypted text

Return type string

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

Parameters

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

Returns encrypted text

Return type string

2.9.1 Examples

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

from secretpy import Caesar
from secretpy import alphabets

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

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

print('=====================================')

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

'''
Output:

thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqirämxpsvryhuwkhodüögrj
thequickbrownfoxjumpsoverthelazydog
=====================================
thequickbrownfoxjumpsoverthelazydog
```

(continues on next page)

(continued from previous page)

```
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
'''
```

2.10 Caesar Progressive

class secretpy.**CaesarProgressive**

The Caesar Progressive Cipher

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

Decryption method

Parameters

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

Returns decrypted text

Return type string

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

Encryption method

Parameters

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

Returns encrypted text

Return type string

2.10.1 Examples

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

from secretpy import CaesarProgressive
from secretpy import alphabets

alphabet = alphabets.ENGLISH
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = 3
cipher = CaesarProgressive()

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

(continues on next page)

(continued from previous page)

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

print('=====================================')

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

'''
thequickbrownfoxjumpsoverthelazydog
wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
thequickbrownfoxjumpsoverthelazydog
=====================================
thequickbrownfoxjumpsoverthelazydog
wljwbqlumdbkcvfpcohlpmuesvkiqgggmyr
thequickbrownfoxjumpsoverthelazydog
'''
```

2.11 Columnar Transposition

class `secretpy.`**`ColumnarTransposition`**

The Columnar Transposition Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.11.1 Examples

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

from secretpy import ColumnarTransposition, CryptMachine
from secretpy import alphabets


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


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

alphabet = alphabets.ENGLISH

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

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

'''
attackatdawn
tanakwadzcazttz
attackatdawnzzz
--------------------------------
howstuffworks
ushfowfztksrwo
howstuffworksz
--------------------------------
'''
```

2.12 Four Square

class secretpy.**FourSquare**

The Four-Square Cipher

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

Decryption method

Parameters

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

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.12.1 Examples

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

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


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


alphabet = alphabets.ENGLISH_SQUARE_OQ

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

cm = NoSpaces(UpperCase(CryptMachine(FourSquare())))

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

alphabet = alphabets.ENGLISH_SQUARE_IJ
cm.set_alphabet(alphabet)
key = (u"criptog", u"segurt")
cm.set_key(key)
plaintext = u"attack at dawn"
```

(continues on next page)

(continued from previous page)

```
encdec(cm, plaintext)

'''
Help me Obi wan Kenobi
FYGMKYHOBXMFKKKIMD
HELPMEOBIWANKENOBI
----------------------------------
attack at dawn
PMMUTBPMCUXH
ATTACKATDAWN
----------------------------------
'''
```

2.13 Gronsfeld

class secretpy.**Gronsfeld**

The Gronsfeld Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.13.1 Examples

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

from secretpy import Gronsfeld
from secretpy import alphabets
```

(continues on next page)

(continued from previous page)

```

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

cipher = Gronsfeld()
print(plaintext)

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

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

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

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

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

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

2.14 Keyword

class secretpy.**Keyword**

The Keyword Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.14.1 Examples

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

from secretpy import Keyword
from secretpy import alphabets

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

cipher = Keyword();
print(plaintext)

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

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

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

plaintext  = u"thisisasecretmessage"
key = "keyword"

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

2.15 Nihilist

class secretpy.**Nihilist**
The Nihilist Cipher

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

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.15.1 Examples

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

from secretpy import Nihilist
from secretpy import CryptMachine


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


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

alphabet = [
```

(continues on next page)

(continued from previous page)

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

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

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

alphabet = [
    u"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"
]
plaintext = u"ΠΙΝΑΚΑΣ"
cm.set_alphabet(alphabet)
encdec(cm, plaintext)

'''
Output:

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

102 82 90 101 102

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

(continues on next page)

(continued from previous page)

```
'''
```

2.16 Playfair

class secretpy.**Playfair**

The Playfair Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.16.1 Examples

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

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


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

```

(continues on next page)

(continued from previous page)

```

cm = NoSpaces(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)
```

2.17 Polybius

class secretpy.**Polybius**

The Polybius Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.17.1 Examples

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

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


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


cm = CryptMachine(Polybius())

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

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

plaintext = "thisisasecretmessage"
encdec(cm, plaintext)

cm.set_alphabet(alph.GREEK)
plaintext = u"ΠΙΝΑΚΑΣ"
cm = LowerCase(cm)
encdec(cm, plaintext)
```

2.18 Porta

class secretpy.**Porta**

The Porta Cipher

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

Decryption method

Parameters

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

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.18.1 Examples

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

from secretpy import Porta
from secretpy import alphabets

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

cipher = Porta()
print(plaintext)

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

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

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

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

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

'''
thequickbrownfoxjumpsoverthelazydog
dßwheputrkrnßöroznpgcvdübmzüöwhatvy
```

(continues on next page)

(continued from previous page)

```
thequickbrownfoxjumpsoverthelazydog
----------------------------------
attackatdawn
seauvppaxtel
attackatdawn
'''
```

2.19 Rot13

class secretpy.**Rot13**

The Rot13 Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.19.1 Examples

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

from secretpy import Rot13
from secretpy import CryptMachine
from secretpy.cmdecorators import SaveCase, SaveSpaces
from secretpy import alphabets


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

(continues on next page)

(continued from previous page)

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


cm = SaveCase(CryptMachine(Rot13()))

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

cm = SaveSpaces(cm)

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

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

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

2.20 Rot5

class secretpy.**Rot5**
The Rot5 Cipher

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

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.20.1 Examples

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

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


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


cm = CryptMachine(Rot5())

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

2.21 Rot18

class secretpy.**Rot18**

The Rot18 Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.21.1 Examples

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

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


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


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

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

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

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

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

2.22 Rot47

class secretpy.**Rot47**

The Rot47 Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.22.1 Examples

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

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


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


cm = SaveSpaces(CryptMachine(Rot47()))

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

2.23 Simple Substitution

class secretpy.**SimpleSubstitution**
The Simple Substitution Cipher

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

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.23.1 Examples

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

from secretpy import SimpleSubstitution
from secretpy import alphabets

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

cipher = SimpleSubstitution()
print(plaintext)

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

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

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

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

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

(continues on next page)

(continued from previous page)

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

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

2.24 Three Square

class secretpy.**ThreeSquare**

The Three Square Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.24.1 Examples

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

from secretpy import ThreeSquare
from secretpy import CryptMachine
```

(continues on next page)

(continued from previous page)

```python
from secretpy import alphabets
from secretpy.cmdecorators import NoSpaces, UpperCase


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


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

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

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

2.25 Trifid

class secretpy.**Trifid**

The Trifid Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.25.1 Examples

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

from secretpy import Trifid
from secretpy import CryptMachine


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"e", u"p", u"s",
    u"d", u"u", u"c",
    u"v", u"w", u"y",

    u"m", u".", u"z",
    u"l", u"k", u"x",
    u"n", u"b", u"t",

    u"f", u"g", u"o",
    u"r", u"i", u"j",
    u"h", u"a", u"q",
]

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

'''
defendtheeastwallofthecastle
suefecphsegyyjiximfofocejlrf
```

(continues on next page)

(continued from previous page)

```
defendtheeastwallofthecastle
----------------------------------
'''
```

2.26 Two Square

class secretpy.**TwoSquare**
The Two-Square Cipher

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

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.26.1 Examples

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

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


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

(continues on next page)

(continued from previous page)

```python
    print(dec)
    print("----------------------------------")


alphabet = alphabets.ENGLISH_SQUARE_OQ

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

cm = NoSpaces(UpperCase(CryptMachine(TwoSquare())))

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

'''
Help me Obi wan Kenobi
XGDLXWSDJYRYHOTKDG
HELPMEOBIWANKENOBI
----------------------------------
'''
```

2.27 Vic

class secretpy.**Vic**

The Vic Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.27.1 Examples

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

from secretpy import Vic
from secretpy import CryptMachine


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


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

'''
Output:

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

2.28 Vigenere

class secretpy.**Vigenere**

The Vigenere Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Parameters

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

Returns text

Return type string

2.28.1 Examples

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

from secretpy import Vigenere
from secretpy import alphabets

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

cipher = Vigenere()
print(plaintext)

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

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

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

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

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

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

2.29 Zigzag

class secretpy.**Zigzag**

The Zigzag Cipher

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

Decryption method

Parameters

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

Returns text

Return type string

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

Encryption method

Parameters

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

Returns text

Return type string

2.29.1 Examples

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

from secretpy import Zigzag
from secretpy import alphabets

alphabet = alphabets.GERMAN
plaintext = u"thequickbrownfoxjumpsoverthelazydog"
key = 3

chipher = Zigzag()
print(plaintext)

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

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

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

(continues on next page)

(continued from previous page)

```

plaintext = u"wearediscoveredfleeatonce"

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

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

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

plaintext = u"defendtheeastwallofthecastle"
key = 4

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

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

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index