
PyOTP Documentation

Release 0.0.1

PyOTP contributors

Jun 10, 2017

Contents

1	Quick overview of using One Time Passwords on your phone	3
2	Installation	5
3	Usage	7
3.1	Time-based OTPs	7
3.2	Counter-based OTPs	7
3.3	Generating a base32 Secret Key	7
3.4	Google Authenticator Compatible	8
3.5	Working example	8
3.6	Links	8
4	API documentation	9
5	Table of Contents	13
	Python Module Index	15

PyOTP is a Python library for generating and verifying one-time passwords. It can be used to implement two-factor (2FA) or multi-factor (MFA) authentication methods in web applications and in other systems that require users to log in.

Open MFA standards are defined in [RFC 4226](#) (HOTP: An HMAC-Based One-Time Password Algorithm) and in [RFC 6238](#) (TOTP: Time-Based One-Time Password Algorithm). PyOTP implements server-side support for both of these standards. Client-side support can be enabled by sending authentication codes to users over SMS or email (HOTP) or, for TOTP, by instructing users to use [Google Authenticator](#), [Authy](#), or another compatible app. Users can set up auth tokens in their apps easily by using their phone camera to scan `otpauth://` QR codes provided by PyOTP.

We recommend that implementers read the [OWASP Authentication Cheat Sheet](#) and [NIST SP 800-63-3: Digital Authentication Guideline](#) for a high level overview of authentication best practices.

Quick overview of using One Time Passwords on your phone

- OTPs involve a shared secret, stored both on the phone and the server
- OTPs can be generated on a phone without internet connectivity
- OTPs should always be used as a second factor of authentication (if your phone is lost, your account is still secured with a password)
- Google Authenticator and other OTP client apps allow you to store multiple OTP secrets and provision those using a QR Code

CHAPTER 2

Installation

```
pip install pyotp
```


Time-based OTPs

```
totp = pyotp.TOTP('base32secret3232')
totp.now() # => 492039

# OTP verified for current time
totp.verify(492039) # => True
time.sleep(30)
totp.verify(492039) # => False
```

Counter-based OTPs

```
hotp = pyotp.HOTP('base32secret3232')
hotp.at(0) # => 260182
hotp.at(1) # => 55283
hotp.at(1401) # => 316439

# OTP verified with a counter
hotp.verify(316439, 1401) # => True
hotp.verify(316439, 1402) # => False
```

Generating a base32 Secret Key

```
pyotp.random_base32() # returns a 16 character base32 secret. Compatible with Google_
↳ Authenticator and other OTP apps
```

Google Authenticator Compatible

PyOTP works with the Google Authenticator iPhone and Android app, as well as other OTP apps like Authy. PyOTP includes the ability to generate provisioning URIs for use with the QR Code scanner built into these MFA client apps:

```
pyotp.totp.TOTP('JBSWY3DPEHPK3PXP').provisioning_uri("alice@google.com", issuer_name=
↳ "Secure App")

>>> 'otpauth://totp/Secure%20App:alice%40google.com?secret=JBSWY3DPEHPK3PXP&
↳ issuer=Secure%20App'

pyotp.hotp.HOTP('JBSWY3DPEHPK3PXP').provisioning_uri("alice@google.com", initial_
↳ count=0, issuer_name="Secure App")

>>> 'otpauth://hotp/Secure%20App:alice%40google.com?secret=JBSWY3DPEHPK3PXP&
↳ issuer=Secure%20App&counter=0'
```

This URL can then be rendered as a QR Code (for example, using <https://github.com/neocotic/qrious>) which can then be scanned and added to the users list of OTP credentials.

Working example

Scan the following barcode with your phone's OTP app (e.g. Google Authenticator):

Now run the following and compare the output:

```
import pyotp
totp = pyotp.TOTP("JBSWY3DPEHPK3PXP")
print("Current OTP:", totp.now())
```

Links

- [Project home page \(GitHub\)](#)
- [Documentation \(Read the Docs\)](#)
- [Package distribution \(PyPI\)](#)
- [Change log](#)
- [RFC 4226: HOTP: An HMAC-Based One-Time Password](#)
- [RFC 6238: TOTP: Time-Based One-Time Password Algorithm](#)
- [ROTP - Original Ruby OTP library by Mark Percival](#)
- [OTPHP - PHP port of ROTP by Le Lag](#)
- [OWASP Authentication Cheat Sheet](#)
- [NIST SP 800-63-3: Digital Authentication Guideline](#)

class `pyotp.totp.TOTP` (**args, **kwargs*)

Handler for time-based OTP counters.

at (*for_time, counter_offset=0*)

Accepts either a Unix timestamp integer or a datetime object.

Parameters

- **for_time** (*int or datetime*) – the time to generate an OTP for
- **counter_offset** – the amount of ticks to add to the time counter

Returns OTP value

Return type str

now ()

Generate the current time OTP

Returns OTP value

Return type str

provisioning_uri (*name, issuer_name=None*)

Returns the provisioning URI for the OTP. This can then be encoded in a QR Code and used to provision an OTP app like Google Authenticator.

See also: <https://github.com/google/google-authenticator/wiki/Key-Uri-Format>

Parameters

- **name** (*str*) – name of the user account
- **issuer_name** – the name of the OTP issuer; this will be the organization title of the OTP entry in Authenticator

Returns provisioning URI

Return type str

verify (*otp*, *for_time=None*, *valid_window=0*)

Verifies the OTP passed in against the current time OTP.

Parameters

- **otp** (*str*) – the OTP to check against
- **for_time** (*int or datetime*) – Time to check OTP at (defaults to now)
- **valid_window** (*int*) – extends the validity to this many counter ticks before and after the current one

Returns True if verification succeeded, False otherwise

Return type bool

class `pyotp.hotp.HOTP` (*s*, *digits=6*, *digest=<built-in function openssl_shal>*)

Handler for HMAC-based OTP counters.

at (*count*)

Generates the OTP for the given count.

Parameters **count** (*int*) – the OTP HMAC counter

Returns OTP

Return type str

provisioning_uri (*name*, *initial_count=0*, *issuer_name=None*)

Returns the provisioning URI for the OTP. This can then be encoded in a QR Code and used to provision an OTP app like Google Authenticator.

See also: <https://github.com/google/google-authenticator/wiki/Key-Uri-Format>

Parameters

- **name** (*str*) – name of the user account
- **initial_count** (*int*) – starting HMAC counter value, defaults to 0
- **issuer_name** – the name of the OTP issuer; this will be the organization title of the OTP entry in Authenticator

Returns provisioning URI

Return type str

verify (*otp*, *counter*)

Verifies the OTP passed in against the current time OTP.

Parameters

- **otp** (*str*) – the OTP to check against
- **count** (*int*) – the OTP HMAC counter

`pyotp.utils.build_uri` (*secret*, *name*, *initial_count=None*, *issuer_name=None*, *algorithm=None*, *digits=None*, *period=None*)

Returns the provisioning URI for the OTP; works for either TOTP or HOTP.

This can then be encoded in a QR Code and used to provision the Google Authenticator app.

For module-internal use.

See also: <https://github.com/google/google-authenticator/wiki/Key-Uri-Format>

Parameters

- **secret** (*str*) – the hotp/totp secret used to generate the URI
- **name** (*str*) – name of the account
- **initial_count** (*int*) – starting counter value, defaults to None. If none, the OTP type will be assumed as TOTP.
- **issuer_name** (*str*) – the name of the OTP issuer; this will be the organization title of the OTP entry in Authenticator
- **algorithm** (*str*) – the algorithm used in the OTP generation.
- **digits** (*int*) – the length of the OTP generated code.
- **period** (*int*) – the number of seconds the OTP generator is set to expire every code.

Returns provisioning uri

Return type str

`pyotp.utils.strings_equal` (*s1*, *s2*)

Timing-attack resistant string comparison.

Normal comparison using `==` will short-circuit on the first mismatching character. This avoids that by scanning the whole string, though we still reveal to a timing attack whether the strings are the same length.

CHAPTER 5

Table of Contents

- genindex
- modindex
- search

p

pyotp, 9
pyotp.hotp, 10
pyotp.totp, 9
pyotp.utils, 10

A

at() (pyotp.hotp.HOTP method), 10
at() (pyotp.totp.TOTP method), 9

B

build_uri() (in module pyotp.utils), 10

H

HOTP (class in pyotp.hotp), 10

N

now() (pyotp.totp.TOTP method), 9

P

provisioning_uri() (pyotp.hotp.HOTP method), 10
provisioning_uri() (pyotp.totp.TOTP method), 9
pyotp (module), 9
pyotp.hotp (module), 10
pyotp.totp (module), 9
pyotp.utils (module), 10

S

strings_equal() (in module pyotp.utils), 11

T

TOTP (class in pyotp.totp), 9

V

verify() (pyotp.hotp.HOTP method), 10
verify() (pyotp.totp.TOTP method), 9