

---

# **Python-BIP38**

*Release v0.1.0*

**Meheret Tesfaye Batu**

**Sep 07, 2023**



# CONTENTS

<b>1</b>	<b>Bitcoin Improvement Proposal - 0038 / BIP38</b>	<b>1</b>
1.1	Installing BIP38 . . . . .	1
1.2	Quick Usage . . . . .	1
1.2.1	no EC multiply: . . . . .	1
1.3	Development . . . . .	10
1.4	Testing . . . . .	10
1.5	Contributing . . . . .	10
1.6	Donations . . . . .	10
1.7	License . . . . .	10
<b>2</b>	<b>BIP38</b>	<b>11</b>
<b>3</b>	<b>Utils</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



## BITCOIN IMPROVEMENT PROPOSAL - 0038 / BIP38

Python library for implementation of Bitcoin Improvement Proposal - 0038 / BIP38 protocol. It supports both [No EC-multiply](#) and [EC-multiply](#) modes.

BIP38 is a cryptographic standard that defines a method for encrypting and securing private keys associated with Bitcoin addresses. It provides a way to create encrypted versions of private keys, which can then be decrypted using a passphrase. This adds an additional layer of security to the process of storing and transmitting private keys.

By encrypting a private key with BIP38, users can protect their funds even if the encrypted private key is exposed. This is because an attacker would need to know the passphrase in order to decrypt the private key and gain access to the associated funds. BIP38 encryption is often used to create “paper wallets” or physical copies of Bitcoin private keys that can be stored offline for enhanced security.

For more info see the [Passphrase-protected private key - BIP38 specs](#).

### 1.1 Installing BIP38

The easiest way to install `bip38` is via `pip`:

```
pip install bip38
```

If you want to run the latest version of the code, you can install from `git`:

```
pip install git+git://github.com/meherett/python-bip38.git
```

For the versions available, see the [tags on this repository](#).

### 1.2 Quick Usage

#### 1.2.1 no EC multiply:

```
#!/usr/bin/env python3

from bip38 import (
    bip38_encrypt, bip38_decrypt
)
from typing import List
```

(continues on next page)

(continued from previous page)

```

import json

# Passphrase / password
PASSPHRASE: str = "meherett" # u"\u03D2\u0301\u0000\u000010400\u00001F4A9"
# To show detail
DETAIL: bool = True
# Wallet important format's
WIFs: List[str] = [
    "5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR", # No compression
    "L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP" # Compression
]

for WIF in WIFs:

    print("WFI:", WIF)
    encrypted_wif: str = bip38_encrypt(
        wif=WIF, passphrase=PASSPHRASE
    )
    print("BIP38 Encrypted WIF:", encrypted_wif)
    print("BIP38 Decrypted:", json.dumps(bip38_decrypt(
        encrypted_wif=encrypted_wif, passphrase=PASSPHRASE, detail=DETAIL
    ), indent=4))

    print("-" * 125)

```

```

WFI: 5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR
BIP38 Encrypted WIF: 6PRVWUbkzNehVoPSCKYviigdnwsc69PLiMPpTVWGENzUAY7spnAZqnxit
BIP38 Decrypted: {
    "wif": "5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR",
    "private_key": "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5",
    "wif_type": "wif",
    "public_key":
↪ "04d2ce831dd06e5c1f5b1121ef34c2af4bc01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b8807
↪ ",
    "public_key_type": "uncompressed",
    "seed": null,
    "address": "1Jq6MksXQVWzrzvZzxcV6oY57oWXd9TXB",
    "lot": null,
    "sequence": null
}

-----
↪ -----
WFI: L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP
BIP38 Encrypted WIF: 6PYNKZ1EASfdDgcUgtxxRi7DkYPTXzwYUzEqzDxv2H8QbeKDV9D9wBWUA7
BIP38 Decrypted: {
    "wif": "L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP",
    "private_key": "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5",
    "wif_type": "wif-compressed",
    "public_key": "02d2ce831dd06e5c1f5b1121ef34c2af4bc01b126e309234adbc3561b60c9360e",
    "public_key_type": "compressed",
    "seed": null,
    "address": "164MQi977u9GUteHr4EPH27VkkdxmfCvGW",

```

(continues on next page)

(continued from previous page)

```

"lot": null,
"sequence": null
}
-----
-----

```

**EC multiply:**

```

#!/usr/bin/env python3

from bip38 import (
    intermediate_code, create_new_encrypted_wif, confirm_code, bip38_decrypt
)
from typing import List

import json
import os

# Passphrase / password
PASSPHRASE: str = "meherett" # u"\u03D2\u0301\u0000\u000010400\u00001F4A9"
# To show detail
DETAIL: bool = True
# List of samples with owner salt, seed, public key type, lot, and sequence
SAMPLES: List[dict] = [
    # Random owner salt & seed, No compression, No lot & sequence
    {"owner_salt": os.urandom(8), "seed": os.urandom(24), "public_key_type":
    ↪ "uncompressed", "lot": None, "sequence": None},
    # Random owner salt & seed, No compression, With lot & sequence
    {"owner_salt": os.urandom(8), "seed": os.urandom(24), "public_key_type":
    ↪ "uncompressed", "lot": 863741, "sequence": 1},
    # Random owner salt & seed, Compression, No lot & sequence
    {"owner_salt": os.urandom(8), "seed": os.urandom(24), "public_key_type": "compressed
    ↪", "lot": None, "sequence": None},
    # Random owner salt & seed, Compression, With lot & sequence
    {"owner_salt": os.urandom(8), "seed": os.urandom(24), "public_key_type": "compressed
    ↪", "lot": 863741, "sequence": 1},
    # With owner salt & seed, No compression, No lot & sequence
    {"owner_salt": "75ed1cdeb254cb38", "seed":
    ↪ "99241d58245c883896f80843d2846672d7312e6195ca1a6c", "public_key_type": "uncompressed",
    ↪ "lot": None, "sequence": None},
    # With owner salt & seed, No compression, With lot & sequence
    {"owner_salt": "75ed1cdeb254cb38", "seed":
    ↪ "99241d58245c883896f80843d2846672d7312e6195ca1a6c", "public_key_type": "uncompressed",
    ↪ "lot": 567885, "sequence": 1},
    # With owner salt & seed, Compression, No lot & sequence
    {"owner_salt": "75ed1cdeb254cb38", "seed":
    ↪ "99241d58245c883896f80843d2846672d7312e6195ca1a6c", "public_key_type": "compressed",
    ↪ "lot": None, "sequence": None},
    # With owner salt & seed, Compression, With lot & sequence
    {"owner_salt": "75ed1cdeb254cb38", "seed":

```

(continues on next page)

(continued from previous page)

```

↪ "99241d58245c883896f80843d2846672d7312e6195ca1a6c", "public_key_type": "compressed",
↪ "lot": 369861, "sequence": 1},
]

for SAMPLE in SAMPLES:

    intermediate_passphrase: str = intermediate_code(
        passphrase=PASSPHRASE, owner_salt=SAMPLE["owner_salt"], lot=SAMPLE["lot"], ↪
↪ sequence=SAMPLE["sequence"]
    )

    print("Intermediate Passphrase:", intermediate_passphrase)

    encrypted_wif: dict = create_new_encrypted_wif(
        intermediate_passphrase=intermediate_passphrase, public_key_type=SAMPLE["public_
↪ key_type"], seed=SAMPLE["seed"]
    )
    print("Encrypted WIF:", json.dumps(encrypted_wif, indent=4))

    print("Confirm Code:", json.dumps(confirm_code(
        passphrase=PASSPHRASE, confirmation_code=encrypted_wif["confirmation_code"], ↪
↪ detail=DETAIL
    ), indent=4))

    print("BIP38 Decrypted:", json.dumps(bip38_decrypt(
        encrypted_wif=encrypted_wif["encrypted_wif"], passphrase=PASSPHRASE, ↪
↪ detail=DETAIL
    ), indent=4))

    print("-" * 125)

```

Intermediate Passphrase: ↪

↪ passphraseqtFiMLZSKYBJo6ZdivCqkPyMX3bnPFnedQRtEHWHmADXqEfSyJHE1CLuTbF6Wf

Encrypted WIF: {

"encrypted\_wif": "6PfPd3hFPNjBMqirrvSSgEtDnErh9BzqK1NUdk6fiQCaN7LwdGFus4PhQV",

"confirmation\_code":

↪ "cfrm38V5QE7EN2eF9SfWsesQCjJZSoSjc5YiqLDCgEJoqEDoV2D9f7NRXSqQHsWb3MKogaN8zAs",

"public\_key":

↪ "0412bb1ec0a2fa1e7c90f4061578d8deaa6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119

↪ ",

"seed": "d010fe7f60a25982f3ee7e056e1bcd027f1c15bd26ddd221",

"public\_key\_type": "uncompressed",

"address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi"

}

Confirm Code: {

"public\_key":

↪ "0412bb1ec0a2fa1e7c90f4061578d8deaa6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119

↪ ",

"public\_key\_type": "uncompressed",

"address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi",

"lot": null,

"sequence": null

(continues on next page)

(continued from previous page)

```

}
BIP38 Decrypted: {
  "wif": "5Jp53JGVEkX2dxXXJyb2UdJw3259yk3YjJCdhcHA3eXpJsr6PBB",
  "private_key": "83348354ac6638ad7ea78505bd85ff96485e17edcffe85572df9a66f997e1324",
  "wif_type": "wif",
  "public_key":
  ↪ "0412bb1ec0a2fa1e7c90f4061578d8deea6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119
  ↪ ",
  "public_key_type": "uncompressed",
  "seed": "d010fe7f60a25982f3ee7e056e1bcd027f1c15bd26ddd221",
  "address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi",
  "lot": null,
  "sequence": null
}
-----
↪ -----
Intermediate Passphrase: ↪
↪ passphraseXyya37d7imwPshCWV77N6SdDCXCGkbUDQ8dgg39Xutzej2UoNTRXCWjcvSk3
Encrypted WIF: {
  "encrypted_wif": "6PgHqxpPU2tA4rqjL5gMMkqeahFRRDDe3g1jJy5mhQdNasT1WtwEkzGcdk",
  "confirmation_code":
  ↪ "cfrm38V8LPy6dJTRpd7Qs74zLAdE26F3ZGqJ1Dmr5HheKY2miBwbJMdk1qY6VhZDjNJkitu5Di5",
  "public_key":
  ↪ "049b3dcf56a38df3a2437055f2ad3aec950a54f7205bbcc9949d5299ee4e0215d0924a756dce3baf3356da8465341ebf1c58
  ↪ ",
  "seed": "8195ac15d84c139531faec482a9d312f86f79242acb728a7",
  "public_key_type": "uncompressed",
  "address": "17YeFTwCoxVhz5P8KiGHv4d8JwUEwPubhj"
}
Confirm Code: {
  "public_key":
  ↪ "049b3dcf56a38df3a2437055f2ad3aec950a54f7205bbcc9949d5299ee4e0215d0924a756dce3baf3356da8465341ebf1c58
  ↪ ",
  "public_key_type": "uncompressed",
  "address": "17YeFTwCoxVhz5P8KiGHv4d8JwUEwPubhj",
  "lot": 863741,
  "sequence": 1
}
BIP38 Decrypted: {
  "wif": "5KGpex1ZJaPoG2L6cHtzAU1nM9un8nw3uD8d6v8xGJs6M6q9qQj",
  "private_key": "bfff2e24adfd0323ecd0b969cb3768adba578a0ea503306fd647e6b11e8739d70",
  "wif_type": "wif",
  "public_key":
  ↪ "049b3dcf56a38df3a2437055f2ad3aec950a54f7205bbcc9949d5299ee4e0215d0924a756dce3baf3356da8465341ebf1c58
  ↪ ",
  "public_key_type": "uncompressed",
  "seed": "8195ac15d84c139531faec482a9d312f86f79242acb728a7",
  "address": "17YeFTwCoxVhz5P8KiGHv4d8JwUEwPubhj",
  "lot": 863741,
  "sequence": 1
}
-----

```

(continues on next page)

(continued from previous page)

```

-----
Intermediate Passphrase:↳
↳passphraseoH4GEqnBR53ipb9gwLfbJM8nKMx4LnZPCzYbvgPyR2zYkF5DqKrW2gf8DZ8s7y
Encrypted WIF: {
  "encrypted_wif": "6PnYW3V9jpb8sKA4aMEWJjBvNTRtVYBCSRWb6Yja6xZqBhVVrDXWsnYz2at",
  "confirmation_code":
↳"cfrm38VUi8UMcgVUDQRSjJn1VxVLfHYQxphSRvAQYSU244oNwHoxT24UByEnUeqSbN6QatRVtaR",
  "public_key": "022604144840ed73bc5055916e2e114efe2a706ee71033b48644e3e322a2c58dab",
  "seed": "e0051112f4903c0bbe52dc698c031467bf4646040b6b12a3",
  "public_key_type": "compressed",
  "address": "1EVSafcuHG8Ce2CF74QwW58wSr7WY4QBaH"
}
Confirm Code: {
  "public_key": "022604144840ed73bc5055916e2e114efe2a706ee71033b48644e3e322a2c58dab",
  "public_key_type": "compressed",
  "address": "1EVSafcuHG8Ce2CF74QwW58wSr7WY4QBaH",
  "lot": null,
  "sequence": null
}
BIP38 Decrypted: {
  "wif": "Kz2v4F99WaPamvCC2LwGTwdr25TnUXUB991wKpVhHGxtJE6iAveq",
  "private_key": "53f56bb7fc1a9e9682aa55be6e501776fc9ac2369654c6c85b00b87d41ab8229",
  "wif_type": "wif-compressed",
  "public_key": "022604144840ed73bc5055916e2e114efe2a706ee71033b48644e3e322a2c58dab",
  "public_key_type": "compressed",
  "seed": "e0051112f4903c0bbe52dc698c031467bf4646040b6b12a3",
  "address": "1EVSafcuHG8Ce2CF74QwW58wSr7WY4QBaH",
  "lot": null,
  "sequence": null
}

```

```

-----
Intermediate Passphrase:↳
↳passphraseawdkWraG6G7W9TCAhCtmoLXbFwDDYjrG8gtv2VPCY7mCvJgbFCoktRkm4ePsQU
Encrypted WIF: {
  "encrypted_wif": "6PoHWWXXJTibxUGKcVmyts86N8rcTHXJpAoj5Verf2FhJqj2oQgCsHheKg",
  "confirmation_code":
↳"cfrm38VX8GoZrei4jxLQKA6Mx2zSWkrQZPhxQW1FcCRjtizmQDoWoomm5SW63ESEAuuLkA8MFmc",
  "public_key": "025f4476d9d8c093a04499fe9d7fbd34533dae14a498a2506a90d6cfdda66e99b3",
  "seed": "1ac2513b9149124a0a0d697ae76cbb4583e85d4a652330a6",
  "public_key_type": "compressed",
  "address": "1ESHxrqxMLrdzwfif9nQbq4PTGhDGi1uq2"
}
Confirm Code: {
  "public_key": "025f4476d9d8c093a04499fe9d7fbd34533dae14a498a2506a90d6cfdda66e99b3",
  "public_key_type": "compressed",
  "address": "1ESHxrqxMLrdzwfif9nQbq4PTGhDGi1uq2",
  "lot": 863741,
  "sequence": 1
}
BIP38 Decrypted: {
  "wif": "L2otjf2N8EpKvh541jw1n3MrXZLpnCfQ2GB4eiGZLFwoSj1UHprw",

```

(continues on next page)

(continued from previous page)

```

"private_key": "a6c57a43bf2a8ecc153b6b1e8807ec2409033616d4fc98a4edae277c02312eb7",
"wif_type": "wif-compressed",
"public_key": "025f4476d9d8c093a04499fe9d7fbd34533dae14a498a2506a90d6cfdda66e99b3",
"public_key_type": "compressed",
"seed": "1ac2513b9149124a0a0d697ae76cbb4583e85d4a652330a6",
"address": "1ESHxrqxMLrdzwfif9nQbq4PTGhDGi1uq2",
"lot": 863741,
"sequence": 1
}

```

-----  
Intermediate Passphrase:↳

↳ passphraseondJvwQGEWFNrnJRPI4G5XAL5SU777GwTNtqmDXqA3CGP7HXfH6AdBxxc5WUKC

Encrypted WIF: {

```

"encrypted_wif": "6Pfp7T3iQ5jLJLsH5DneySLLF5bhd879DHW87Pxzwtwvn2ggcncxsNKN5c",
"confirmation_code":
↳ "cfrm38V5NZfTZKRarDTvFAMkNKqKAXTxDjDdb5RpFfXrVRw7Nov5m2iP3K1Eg5QQRxs52kgapy",
"public_key":
↳ "04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d",
↳ ",
"seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
"public_key_type": "uncompressed",
"address": "18VLTHgu95JPiLiLRtN2WwYroAHvHwE2Ws"
}

```

Confirm Code: {

```

"public_key":
↳ "04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d",
↳ ",
"public_key_type": "uncompressed",
"address": "18VLTHgu95JPiLiLRtN2WwYroAHvHwE2Ws",
"lot": null,
"sequence": null
}

```

BIP38 Decrypted: {

```

"wif": "5Jh21edvnWUXFjRz8mDVN3CSPp1CyTuUSFBKZeWYU726R6MW3ux",
"private_key": "733134eb516f94aa56ab7ef0874a0d71daf38c5c009dec2a1261861a15889631",
"wif_type": "wif",
"public_key":
↳ "04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d",
↳ ",
"public_key_type": "uncompressed",
"seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
"address": "18VLTHgu95JPiLiLRtN2WwYroAHvHwE2Ws",
"lot": null,
"sequence": null
}

```

-----  
Intermediate Passphrase:↳

↳ passphrase7ruSNPsLdQF7t1gh7fs1xvWB4MKDssFQwL11EHkVr4njFX5PtsCUqQqzwh9rS

Encrypted WIF: {

```

"encrypted_wif": "6PgKxJUke6BcDc1XuvPDKCD9krZEebapef98Sj3YAjWQHtR3EVsaeK62ja",

```

(continues on next page)

(continued from previous page)

```

    "confirmation_code":
    ↪ "cfrm38V8TGcdd9WSGpaB56JaiW7cbvv1ZD89BHjBGu7S7yUFGcht8CqFQoexCHCoiCp4JzsH1Pk",
    "public_key":
    ↪ "049afcaa528358eddf54634fee9505e90b9572f8733b94260c94d20b563a65a1c94c338d5c09d20c5895d89bd5a2ba39f96a",
    ↪ ",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "public_key_type": "uncompressed",
    "address": "1DkQJuST62GkJP9kss68fHT8ftLf4SmLVT"
}
Confirm Code: {
    "public_key":
    ↪ "049afcaa528358eddf54634fee9505e90b9572f8733b94260c94d20b563a65a1c94c338d5c09d20c5895d89bd5a2ba39f96a",
    ↪ ",
    "public_key_type": "uncompressed",
    "address": "1DkQJuST62GkJP9kss68fHT8ftLf4SmLVT",
    "lot": 567885,
    "sequence": 1
}
BIP38 Decrypted: {
    "wif": "5JGYLxWwyh9agrM6u63RadubRFjTxbDtvBcQ5EywZrHXBLpPrZW",
    "private_key": "3b9d38cb7d1d97efad80b3934cb1928ae70179317ea4657aaffcdf029f43b90",
    "wif_type": "wif",
    "public_key":
    ↪ "049afcaa528358eddf54634fee9505e90b9572f8733b94260c94d20b563a65a1c94c338d5c09d20c5895d89bd5a2ba39f96a",
    ↪ ",
    "public_key_type": "uncompressed",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "address": "1DkQJuST62GkJP9kss68fHT8ftLf4SmLVT",
    "lot": 567885,
    "sequence": 1
}
-----
↪ -----
Intermediate Passphrase: ↪
↪ passphraseondJvwQGEWFNrnJRPI4G5XAL5SU777GwTNtqmDXqA3CGP7HXfH6AdBxxc5WUKC
Encrypted WIF: {
    "encrypted_wif": "6PnUVPinrvPGwoYJK3GbGBNgFuqEXmfvagE4QiAxj7yrZp4i29p22MrY5r",
    "confirmation_code":
    ↪ "cfrm38VUV4NK45caNN5aomS3dSQLT3FVHq556kehuZX1RNuPs8ArWjw18KCCjyTXktVCDBW65pZ",
    "public_key": "02cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b0",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "public_key_type": "compressed",
    "address": "1BPmkfRYzPAkeErMS6DLDYxPvQEEkoVRz1"
}
Confirm Code: {
    "public_key": "02cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b0",
    "public_key_type": "compressed",
    "address": "1BPmkfRYzPAkeErMS6DLDYxPvQEEkoVRz1",
    "lot": null,
    "sequence": null
}
BIP38 Decrypted: {

```

(continues on next page)

(continued from previous page)

```

"wif": "L15dTs7zPs6UY2HHBGA8BrhV5gTurDkc6RaYw6ZPtdZptsuPR7K3",
"private_key": "733134eb516f94aa56ab7ef0874a0d71daf38c5c009dec2a1261861a15889631",
"wif_type": "wif-compressed",
"public_key": "02cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b0",
"public_key_type": "compressed",
"seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
"address": "1BpMkfRYzPAkeErMS6DLDYxPvQEEkoVRz1",
"lot": null,
"sequence": null
}
-----
↪-----
Intermediate Passphrase:↪
↪passphraseb7ruSNDGP7cmnFHQpmos7TeAy26AFN4GyRTBqq6hiaFbQzQBvirD9oHsafQvzd
Encrypted WIF: {
  "encrypted_wif": "6PoEPBnJjm8UAIsgWQEKKnq9V2GMHqKkTcUqUFzsaX7wgjpQWR2qWPdnpt",
  "confirmation_code":
↪"cfrm38VWx5xH1JFm5EVE3mzQvDPFkz7SqNiaFxyUfp3Fjc2wdYmK7dGEWoW6irDPSrwoaxB5zS",
  "public_key": "024c5175a177a0b6cf0a3d06065345e2e2d0529ea0191ace3d7b003f304353511b",
  "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
  "public_key_type": "compressed",
  "address": "1MQaLNgukYWRkNgtmc1dzJ13yFvJoW34u4"
}
Confirm Code: {
  "public_key": "024c5175a177a0b6cf0a3d06065345e2e2d0529ea0191ace3d7b003f304353511b",
  "public_key_type": "compressed",
  "address": "1MQaLNgukYWRkNgtmc1dzJ13yFvJoW34u4",
  "lot": 369861,
  "sequence": 1
}
BIP38 Decrypted: {
  "wif": "KzFbTBirbEEtEPgWL3xhohUcrg6yUmJupAGrid7vBP9F2Vh8GTUB",
  "private_key": "5a7b39eef5d02551b2d362384e57f9823a1c9bed48a260af920a8bb5d6ad971f",
  "wif_type": "wif-compressed",
  "public_key": "024c5175a177a0b6cf0a3d06065345e2e2d0529ea0191ace3d7b003f304353511b",
  "public_key_type": "compressed",
  "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
  "address": "1MQaLNgukYWRkNgtmc1dzJ13yFvJoW34u4",
  "lot": 369861,
  "sequence": 1
}
-----
↪-----

```

## 1.3 Development

We welcome pull requests. To get started, just fork this [github repository](#), clone it locally, and run:

```
pip install -e .[tests,docs] -r requirements.txt
```

## 1.4 Testing

You can run the tests with:

```
pytest
```

Or use **tox** to run the complete suite against the full set of build targets, or `pytest` to run specific tests against a specific version of Python.

## 1.5 Contributing

Feel free to open an [issue](#) if you find a problem, or a pull request if you've solved an issue. And also any help in testing, development, documentation and other tasks is highly appreciated and useful to the project. There are tasks for contributors of all experience levels.

For more information, see the [CONTRIBUTING.md](#) file.

## 1.6 Donations

Buy me a coffee if You found this tool helpful:

- **BTC** - 12uaGVdX1t86FXLQ4yYPrRQDCK7xGGu82r
- **BTC / ETH / USDT** - [hd.wallet](#)

Thank you very much for your support.

## 1.7 License

Distributed under the [MIT](#) license. See **LICENSE** for more information.

`bip38.bip38.uncompress_public_key(public_key: str | bytes) → str`

Uncompress public key converter

**Parameters**

`public_key` (`Union[str, bytes]`) – Public key

**Returns**

`str` – Uncompressed public key

```
>>> from bip38 import uncompress_public_key
>>> uncompress_public_key(public_key=
↳ "0348ca8b4e7c0c75ecfd4b437535d186a12f3027be0c29d2125e9c0dec48677caa")
↳ '0448ca8b4e7c0c75ecfd4b437535d186a12f3027be0c29d2125e9c0dec48677caacb4cd50b4c5ea3313a69402c8b0a33'
↳ '
```

`bip38.bip38.compress_public_key(public_key: str | bytes) → str`

Compress public key converter

**Parameters**

`public_key` (`Union[str, bytes]`) – Public key

**Returns**

`str` – Compressed public key

```
>>> from bip38 import compress_public_key
>>> compress_public_key(public_key=
↳ "0448ca8b4e7c0c75ecfd4b437535d186a12f3027be0c29d2125e9c0dec48677caacb4cd50b4c5ea3313a69402c8b0a33")
↳ '0348ca8b4e7c0c75ecfd4b437535d186a12f3027be0c29d2125e9c0dec48677caa'
```

`bip38.bip38.private_key_to_public_key(private_key: str | bytes, public_key_type: Literal['uncompressed', 'compressed'] = 'compressed') → str`

Private key to public key converter

**Parameters**

- `private_key` (`Union[str, bytes]`) – Private key
- `public_key_type` (`Literal["uncompressed", "compressed"]`) – Public key type, default to compressed

**Returns**

`str` – Public key

```

>>> from bip38 import private_key_to_public_key
>>> private_key_to_public_key(private_key=
↳ "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5")
'02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e'
>>> private_key_to_public_key(private_key=
↳ "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", public_key_
↳ type="uncompressed")

↳ '04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068'
↳ '

```

`bip38.bip38.private_key_to_wif(private_key: str | bytes, wif_type: Literal['wif', 'wif-compressed'] = 'wif-compressed') → str`

Private key to Wallet Important Fromat (WFI) converter

#### Parameters

- **private\_key** (*Union[str, bytes]*) – Private key
- **wif\_type** (*Literal["wif", "wif-compressed"]*) – Wallet Important Fromat (WFI) type, default to wif-compressed

#### Returns

str – Wallet Important Fromat

```

>>> from bip38 import private_key_to_wif
>>> private_key_to_wif(private_key=
↳ "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5")
'L44B5gGEpQEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP'
>>> private_key_to_wif(private_key=
↳ "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", wif_type="wif
↳ ")
'5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR'

```

`bip38.bip38.wif_to_private_key(wif: str) → str`

Wallet Important Fromat (WFI) to Private key converter

#### Parameters

**wif** (*str*) – Wallet Important Fromat

#### Returns

str – Private key

```

>>> from bip38 import wif_to_private_key
>>> wif_to_private_key(wif="L44B5gGEpQEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'
>>> wif_to_private_key(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'

```

`bip38.bip38.get_wif_type(wif: str) → Literal['wif', 'wif-compressed']`

Get Wallet Important Fromat (WFI) type

#### Parameters

**wif** (*str*) – Wallet Important Fromat

#### Returns

*Literal["wif", "wif-compressed"]* – WFI type

```
>>> from bip38 import get_wif_type
>>> get_wif_type(wif="L44B5gGEpqEDRS9vVPz7QT35jCBG2r3CZwSwQ4fCewXAhAhqGVpP")
'wif-compressed'
>>> get_wif_type(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'wif'
```

`bip38.bip38.get_wif_checksum(wif: str) → str`

Get Wallet Important Fromat (WFI) checksum

**Parameters**

**wif** (*str*) – Wallet Important Fromat

**Returns**

*str* – WFI checksum

```
>>> from bip38 import get_wif_checksum
>>> get_wif_checksum(wif="L44B5gGEpqEDRS9vVPz7QT35jCBG2r3CZwSwQ4fCewXAhAhqGVpP")
'dc37f844'
>>> get_wif_checksum(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'f0a25c0c'
```

`bip38.bip38.public_key_to_addresses(public_key: str | bytes) → str`

Public key to address converter

**Parameters**

**public\_key** (*Union[str, bytes]*) – Public key

**Returns**

*str* – Address

```
>>> from bip38 import public_key_to_addresses
>>> public_key_to_addresses(public_key=
↳ "02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e")
'164MQi977u9GUteHr4EPH27VkkdxmfCvGW'
>>> public_key_to_addresses(public_key=
↳ "04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068
↳ ")
'1Jq6MksXQVWzrzvZzxkV6oY57oWXd9TXB'
```

`bip38.bip38.intermediate_code(passphrase: str, lot: int | None = None, sequence: int | None = None, owner_salt: str | bytes = b'\xb3\x90\xf8L\n\xfe\xc5#') → str`

Intermediate passphrase generator

**Parameters**

- **passphrase** (*str*) – Passphrase or password text
- **lot** (*Optional[int]*) – Lot number between 100000 <= lot <= 999999 range, default to None
- **sequence** (*Optional[int]*) – Sequence number between 0 <= sequence <= 4095 range, default to None
- **owner\_salt** (*Optional[str, bytes]*) – Owner salt, default to `os.urandom(8)`

**Returns**

*str* – Intermediate passphrase

```
>>> from bip38 import intermediate_code
>>> intermediate_code(passphrase="TestingOneTwoThree")
'passphraseqVKbgU4mWMakKGgCtaeVWoETQdzMBy5696bG2w7ckVBeQmoLhMF9vLaxhmzhT3 '
>>> intermediate_code(passphrase="TestingOneTwoThree", lot=199999, sequence=1,
↳ owner_salt="75ed1cdeb254cb38")
'passphraseb7ruSN4At4Rb8hPTNcAVezfsjonvUs4Qo3xSp1fBFsFPvVGSbpP2WTJMhw3mVZ '
```

bip38.bip38.**bip38\_encrypt**(wif: str, passphrase: str) → str

BIP38 Encrypt wallet important format using passphrase/password

#### Parameters

- **wif** (str) – Wallet important format
- **passphrase** (str) – Passphrase or password text

#### Returns

str – Encrypted wallet important format

```
>>> from bip38 import bip38_encrypt
>>> bip38_encrypt(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR",
↳ passphrase="TestingOneTwoThree")
'6PRVWUbkzzsbcVac2qwfssoUJAN1Xhrg6bNk8J7Nzm5H7kxEbn2Nh2ZoGg '
>>> bip38_encrypt(wif="L44B5gGEpQEDRS9vVPz7QT35jCBG2r3CZwSwQ4fCewXAhAhqGVpP",
↳ passphrase="TestingOneTwoThree")
'6PYNKZ1EAgYgmNVamxyXVWHzK5s6DGhwP4J5o44cvXdoY7sRzhtpUeo '
```

bip38.bip38.**create\_new\_encrypted\_wif**(intermediate\_passphrase: str, public\_key\_type:

Literal['uncompressed', 'compressed'] = 'uncompressed', seed: str |

bytes =

b'\xe\x04:\x07\x00\xdcU\x8f\x1b\xbaJ.\xae\x77\x7m\xbe\x7f#\xce8')

→ dict

Create new encrypted wallet important format

#### Parameters

- **intermediate\_passphrase** (str) – Intermediate passphrase text
- **public\_key\_type** (Literal["uncompressed", "compressed"]) – Public key type, default to uncompressed
- **seed** (Optional[str, bytes]) – Seed, default to os.urandom(24)

#### Returns

dict – Encrypted wallet important format

```
>>> from bip38 import create_new_encrypted_wif
>>> create_new_encrypted_wif(intermediate_passphrase=
↳ "passphraseb7ruSN4At4Rb8hPTNcAVezfsjonvUs4Qo3xSp1fBFsFPvVGSbpP2WTJMhw3mVZ")
{'encrypted_wif': '6PgGSHicUccU2dotWbwJd514DqWdmCM4KFNYiP4poXgC57qfUpEkKbTj9U',
↳ 'confirmation_code':
↳ 'cfrm38V8Fog3WpRPMXJD34SF6pGT6ht5ihYMWMMbezczHgPpA1jVkfBTHwQzvuSA4ReF86PHZJY',
↳ 'public_key':
↳ '045cd5348750657daa7ef756582cbd5564fe6f65a4a4370fcc234f642826c8e43ec927612179c2768cf616fd1a2f1cc',
↳ 'seed': 'b6b1122e418c4e188de3f57588119f0f5bdc780f52c3f677', 'public_key_type':
↳ 'uncompressed', 'address': '1JbyXoVN4hXWirGB265q9VE4pQ6qbY6kmr'}
>>> create_new_encrypted_wif(intermediate_passphrase=
```

(continues on next page)

(continued from previous page)

```

↪ "passphraseb7ruSN4At4Rb8hPTNcAVezfsjonvUs4Qo3xSp1fBFsFPvVGSbpP2WTJMhw3mVZ", ↪
↪ public_key_type="compressed", seed=
↪ "99241d58245c883896f80843d2846672d7312e6195ca1a6c")
{'encrypted_wif': '6PoM8coZNg4AGPhQs91RbmmRmfLz6kmmU3XUGGLQxsJ5xN62LsUzMWYcdP',
↪ 'confirmation_code':
↪ 'cfrm38VXL5T6qVke13sHUWtEjibAkK1RquBqMXb2azCv1Zna6JKvBhD1Gf2b15wBj7UPv2BQnf6',
↪ 'public_key': '02100bb0440ff4106a1743750813271e66a7017431e92921e520319f537c7357c1
↪ ', 'seed': '99241d58245c883896f80843d2846672d7312e6195ca1a6c', 'public_key_type':
↪ 'compressed', 'address': '15PuNwFmDqYhRsC9MDPNFvNY4Npzibm67c'}

```

`bip38.bip38.confirm_code(passphrase: str, confirmation_code: str, detail: bool = False) → str | dict`

Confirm passphrase

#### Parameters

- **passphrase** (*str*) – Passphrase or password text
- **confirmation\_code** (*str*) – Confirmation code
- **detail** (*bool*) – To show in detail, default to False

#### Returns

Union[str, dict] – Confirmation of address info's

```

>>> from bip38 import confirm_code
>>> confirm_code(passphrase="TestingOneTwoThree", confirmation_code=
↪ "cfrm38V8F0q3WpRPMXJD34SF6pGT6ht5ihYMWMMbezksHgPpA1jVkfBTHwQzvuSA4ReF86PHZJY")
↪ '1JbyXoVN4hXWirGB265q9VE4pQ6qbY6kmr '
>>> confirm_code(passphrase="TestingOneTwoThree", confirmation_code=
↪ "cfrm38VXL5T6qVke13sHUWtEjibAkK1RquBqMXb2azCv1Zna6JKvBhD1Gf2b15wBj7UPv2BQnf6", ↪
↪ detail=True)
{'public_key': '02100bb0440ff4106a1743750813271e66a7017431e92921e520319f537c7357c1',
↪ 'public_key_type': 'compressed', 'address': '15PuNwFmDqYhRsC9MDPNFvNY4Npzibm67c',
↪ 'lot': 199999, 'sequence': 1}

```

`bip38.bip38.bip38_decrypt(encrypted_wif: str, passphrase: str, detail: bool = False) → str | dict`

BIP38 Decrypt encrypted wallet important format using passphrase/password

#### Parameters

- **encrypted\_wif** (*str*) – Encrypted wallet important format
- **passphrase** (*str*) – Passphrase or password text
- **detail** (*bool*) – To show in detail, default to False

#### Returns

Union[str, dict] – WIF or All private Key info's

```

>>> from bip38 import bip38_decrypt
>>> bip38_decrypt(encrypted_wif=
↪ "6PRVWUbkzzsbcVac2qwfssouJAN1XhrG6bNk8J7Nzm5H7kxEbn2Nh2ZoGg", passphrase=
↪ "TestingOneTwoThree")
↪ '5KN7MzqK5wt2TP1fQCYyHBtDrXDJuXbUzm4A9rKAtGu3Qi5CVR'
>>> bip38_decrypt(encrypted_wif=
↪ "6PRVWUbkzzsbcVac2qwfssouJAN1XhrG6bNk8J7Nzm5H7kxEbn2Nh2ZoGg", passphrase=
↪ "TestingOneTwoThree", detail=True)

```

(continues on next page)

(continued from previous page)

```
{'wif': '5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR', 'private_key':  
→ 'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5', 'wif_type':  
→ 'wif', 'public_key':  
→ '04d2ce831dd06e5c1f5b1121ef34c2af4bc01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b',  
→ 'public_key_type': 'uncompressed', 'address':  
→ '1Jq6MksXQVWzrzvZzXkV6oY57oWXd9TXB', 'lot': None, 'sequence': None}
```

`bip38.utils.get_bytes(data: AnyStr, unhexlify: bool = True) → bytes`

Any string to bytes converter

**Parameters**

- **data** (*AnyStr*) – Data
- **unhexlify** (*bool*) – Unhexlify, default to True

**Returns**

bytes – Data

`bip38.utils.bytes_to_string(data: bytes) → str`

Bytes to string converter

**Parameters**

**data** (*bytes*) – Data

**Returns**

str – Data

`bip38.utils.bytes_to_integer(data: bytes, endianness: Literal['little', 'big'] = 'big', signed: bool = False) → int`

Bytes to integer converter

**Parameters**

- **data** (*bytes*) – Data
- **endianness** (*Literal["little", "big"]*) – Endianness, default to big
- **signed** (*bool*) – Signed, default to False

**Returns**

int – Data

`bip38.utils.integer_to_bytes(data: int, bytes_num: int | None = None, endianness: Literal['little', 'big'] = 'big', signed: bool = False) → bytes`

Integer to bytes converter

**Parameters**

- **data** (*int*) – Data
- **bytes\_num** (*Optional[int]*) – Bytes number, default to None
- **endianness** (*Literal["little", "big"]*) – Endianness, default to big
- **signed** (*bool*) – Signed, default to False

**Returns**

bytes – Data

`bip38.utils.ripemd160(data: str | bytes) → bytes`

Ripemd160 hash

**Parameters**

**data** (*Union[str, bytes]*) – Data

**Returns**

bytes – Data ripemd160 hash

`bip38.utils.sha256(data: str | bytes) → bytes`

SHA256 hash

**Parameters**

**data** (*Union[str, bytes]*) – Data

**Returns**

bytes – Data sha256 hash

`bip38.utils.double_sha256(data: str | bytes) → bytes`

Double SHA256 hash

**Parameters**

**data** (*Union[str, bytes]*) – Data

**Returns**

bytes – Data double sha256 hash

`bip38.utils.hash160(data: str | bytes) → bytes`

Hash160 hash

**Parameters**

**data** (*Union[str, bytes]*) – Data

**Returns**

bytes – Data hash160 hash

## PYTHON MODULE INDEX

### b

`bip38.bip38`, 11

`bip38.utils`, 17



## INDEX

### B

bip38.bip38  
    module, 11  
bip38.utils  
    module, 17

bip38\_decrypt() (in module *bip38.bip38*), 15  
bip38\_encrypt() (in module *bip38.bip38*), 14  
bytes\_to\_integer() (in module *bip38.utils*), 17  
bytes\_to\_string() (in module *bip38.utils*), 17

### C

compress\_public\_key() (in module *bip38.bip38*), 11  
confirm\_code() (in module *bip38.bip38*), 15  
create\_new\_encrypted\_wif() (in module  
    *bip38.bip38*), 14

### D

double\_sha256() (in module *bip38.utils*), 18

### G

get\_bytes() (in module *bip38.utils*), 17  
get\_wif\_checksum() (in module *bip38.bip38*), 13  
get\_wif\_type() (in module *bip38.bip38*), 12

### H

hash160() (in module *bip38.utils*), 18

### I

integer\_to\_bytes() (in module *bip38.utils*), 17  
intermediate\_code() (in module *bip38.bip38*), 13

### M

module  
    bip38.bip38, 11  
    bip38.utils, 17

### P

private\_key\_to\_public\_key() (in module  
    *bip38.bip38*), 11  
private\_key\_to\_wif() (in module *bip38.bip38*), 12

public\_key\_to\_addresses() (in module  
    *bip38.bip38*), 13

### R

ripemd160() (in module *bip38.utils*), 18

### S

sha256() (in module *bip38.utils*), 18

### U

uncompress\_public\_key() (in module *bip38.bip38*),  
    11

### W

wif\_to\_private\_key() (in module *bip38.bip38*), 12