Python-BIP38

Release v0.3.0

Meheret Tesfaye Batu

Oct 25, 2023

CONTENTS

CHAPTER

ONE

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 (
    private_key_to_wif, bip38_encrypt, bip38_decrypt
)
from typing import (
    List, Literal
```

(continues on next page)

(continued from previous page)

```
)

import json

# Private key
PRIVATE_KEY: str = "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5"
# Passphrase / password
PASSPHRASE: str = "meherett"  # u"\u03D2\u0301\u0000\U00010400\U0001F4A9"
# Network type
NETWORK: Literal["mainnet", "testnet"] = "mainnet"
# To show detail
DETAIL: bool = True
# Wallet important format's
WIFs: List[str] = [
    private_key_to_wif(private_key=PRIVATE_KEY, wif_type="wif", network=NETWORK),  # No␣
↪compression
    private_key_to_wif(private_key=PRIVATE_KEY, wif_type="wif-compressed",␣
↪network=NETWORK)  # Compression
]

for WIF in WIFs:

    print("WIF:", WIF)

    encrypted_wif: str = bip38_encrypt(
        wif=WIF, passphrase=PASSPHRASE, network=NETWORK
    )
    print("BIP38 Encrypted WIF:", encrypted_wif)

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

    print("-" * 125)
```

```
WFI: 5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR
BIP38 Encrypted WIF: 6PRVWUbkzNehVoPSCKYviigdnwsck69PLiMPpTVWGENzUAy7spnAZqnxit
BIP38 Decrypted: {
    "wif": "5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR",
    "private_key": "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5",
    "wif_type": "wif",
    "public_key":
↪"04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b8807
↪",
    "public_key_type": "uncompressed",
    "seed": null,
    "address": "1Jq6MksXQVWzrznvZzxkV6oY57oWXD9TXB",
    "lot": null,
    "sequence": null
}
-----------------------------------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
↪-----------------------------------
WFI: L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP
BIP38 Encrypted WIF: 6PYNKZ1EASfdDgcUgtxxRi7DkYPTXzwYUzEqzDxv2H8QbeKDV9D9wBWUA7
BIP38 Decrypted: {
    "wif": "L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP",
    "private_key": "cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5",
    "wif_type": "wif-compressed",
    "public_key": "02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e",
    "public_key_type": "compressed",
    "seed": null,
    "address": "164MQi977u9GUteHr4EPH27VkkdxmfCvGW",
    "lot": null,
    "sequence": null
}
-------------------------------------------------------------------------------------------
↪-----------------------------------
```

EC multiply:

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

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

import json
import os

# Passphrase / password
PASSPHRASE: str = "meherett"  # u"\u03D2\u0301\u0000\U00010400\U0001F4A9"
# Network type
NETWORK: Literal["mainnet", "testnet"] = "mainnet"
# 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
```

(continues on next page)

(continued from previous page)

```
↪", "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":
↪"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"], network=NETWORK
    )
    print("Encrypted WIF:", json.dumps(encrypted_wif, indent=4))

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

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

    print("-" * 125)
```

```
Intermediate Passphrase:␣
↪passphraseqtFiMLZSKYBJo6ZdivCqkPyMX3bnPFnedQRtEHWHmADXqEfSyJHE1CLuTbF6Wf
Encrypted WIF: {
    "encrypted_wif": "6PfPd3hFPNjBMqirrvSSgEtDnErh9BzqK1NUdk6fiQCaN7LwdGFus4PhQV",
    "confirmation_code":
↪"cfrm38V5QE7EN2eF9SfWsesQCjJZSoSjc5YiqLDCgEJoqEDoV2D9f7NRXSqQHsWb3MKogaN8zAs",
    "public_key":
```

(continues on next page)

(continued from previous page)

```
↪"0412bb1ec0a2fa1e7c90f4061578d8deeaa6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119
↪",
    "seed": "d010fe7f60a25982f3ee7e056e1bcd027f1c15bd26ddd221",
    "public_key_type": "uncompressed",
    "address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi"
}
Confirm Code: {
    "public_key":
↪"0412bb1ec0a2fa1e7c90f4061578d8deeaa6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119
↪",
    "public_key_type": "uncompressed",
    "address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi",
    "lot": null,
    "sequence": null
}
BIP38 Decrypted: {
    "wif": "5Jp53JGVEkX2dxXXJyb2UdJw3259yk3YjJCdhcHA3eXpJsr6PBB",
    "private_key": "83348354ac6638ad7ea78505bd85ff96485e17edcffe85572df9a66f997e1324",
    "wif_type": "wif",
    "public_key":
↪"0412bb1ec0a2fa1e7c90f4061578d8deeaa6984c9ec5c37717546fb0d127573a03f3050a9f7cb24f62e107c4347038853119
↪",
    "public_key_type": "uncompressed",
    "seed": "d010fe7f60a25982f3ee7e056e1bcd027f1c15bd26ddd221",
    "address": "1CHsGDzDbZJPVKiC9hUKe1hnAevwu5RTKi",
    "lot": null,
    "sequence": null
}
------------------------------------------------------------------------------------------
↪-------------------------------------
Intermediate Passphrase:␣
↪passphrasedcXyya37d7imwPshCWV77N6SdDCXCGkbUDQ8dgg39Xutzej2UoNTRXCWjcVSk3
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
}
```

(continues on next page)

(continued from previous page)

```
BIP38 Decrypted: {
    "wif": "5KGpex1ZJaPoG2L6cHtzAU1nM9un8nw3uD8d6v8xGJs6M6q9qQj",
    "private_key": "bff2e24adfd0323ecd0b969cb3768adba578a0ea503306fd647e6b11e8739d70",
    "wif_type": "wif",
    "public_key":
↪"049b3dcf56a38df3a2437055f2ad3aec950a54f7205bbcc9949d5299ee4e0215d0924a756dce3baf3356da8465341ebf1c58
↪",
    "public_key_type": "uncompressed",
    "seed": "8195ac15d84c139531faec482a9d312f86f79242acb728a7",
    "address": "17YeFTwCoxVhz5P8KiGHv4d8JwUEwPUbhj",
    "lot": 863741,
    "sequence": 1
}
------------------------------------------------------------------------------------------
↪-----------------------------------
Intermediate Passphrase:␣
↪passphraseoH4GEqnBR53ipb9gwLfbJM8nKMx4LnZPCzYbvgPyR2zYkF5DqKrW2gf8DZ8s7y
Encrypted WIF: {
    "encrypted_wif": "6PnYW3V9jp8sKA4aMEWJjBvNTRtVYBCSRWb6Yja6xZqBhVVrDXWSnYz2at",
    "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",
```

(continues on next page)

(continued from previous page)

```
    "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",
    "private_key": "a6c57a43bf2a8ecc153b6b1e8807ec2409033616d4fc98a4edae277c02312eb7",
    "wif_type": "wif-compressed",
    "public_key": "025f4476d9d8c093a04499fe9d7fbd34533dae14a498a2506a90d6cfdda66e99b3",
    "public_key_type": "compressed",
    "seed": "1ac2513b9149124a0a0d697ae76cbb4583e85d4a652330a6",
    "address": "1ESHxrqxMLrdzwfif9nQbq4PTGhDGi1uq2",
    "lot": 863741,
    "sequence": 1
}
------------------------------------------------------------------------------------------
↪-----------------------------------
Intermediate Passphrase:␣
↪passphraseondJwvQGEWFNrNJRPi4G5XAL5SU777GwTNtqmDXqA3CGP7HXfH6AdBxxc5WUKC
Encrypted WIF: {
    "encrypted_wif": "6PfP7T3iQ5jLJLsH5DneySLLF5bhd879DHW87Pxzwtwvn2ggcncxsNKN5c",
    "confirmation_code":
↪"cfrm38V5NZfTZKRaRDTvFAMkNKqKAxTxdDjDdb5RpFfXrVRw7Nov5m2iP3K1Eg5QQRxs52kgapy",
    "public_key":
↪"04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d4
↪",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "public_key_type": "uncompressed",
    "address": "18VLTHgu95JPi1iLRtN2WwYroAHvHwE2Ws"
}
Confirm Code: {
    "public_key":
↪"04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d4
↪",
    "public_key_type": "uncompressed",
    "address": "18VLTHgu95JPi1iLRtN2WwYroAHvHwE2Ws",
    "lot": null,
    "sequence": null
}
BIP38 Decrypted: {
    "wif": "5Jh21edvnWUXFjRz8mDVN3CSPp1CyTuUSFBKZeWYU726R6MW3ux",
    "private_key": "733134eb516f94aa56ab7ef0874a0d71daf38c5c009dec2a1261861a15889631",
    "wif_type": "wif",
    "public_key":
```

(continues on next page)

(continued from previous page)

```
↪"04cdcd8f846a73e75c8a845d1df19dc23031648c219d1efc6fe945cd089f3052b09e25cb1d8628cd559c6c57c627fa486b8d
↪",
    "public_key_type": "uncompressed",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "address": "18VLTHgu95JPi1iLRtN2WwYroAHvHwE2Ws",
    "lot": null,
    "sequence": null
}
------------------------------------------------------------------------------------------
↪-----------------------------------
Intermediate Passphrase:␣
↪passphraseb7ruSNPsLdQF7t1gh7fs1xvWB4MKDssFQwL11EHkVr4njFX5PtsCUqQqwzh9rS
Encrypted WIF: {
    "encrypted_wif": "6PgKxJUke6BcDc1XuvPDKCD9krZEebapef98SJ3YAjWQHtR3EVsaeK62ja",
    "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": "3b9d38cb7d1d97efad80b3934cb1928ae70179317ea4657aaffcdff029f43b90",
    "wif_type": "wif",
    "public_key":
↪"049afcaa528358eddf54634fee9505e90b9572f8733b94260c94d20b563a65a1c94c338d5c09d20c5895d89bd5a2ba39f96a
↪",
    "public_key_type": "uncompressed",
    "seed": "99241d58245c883896f80843d2846672d7312e6195ca1a6c",
    "address": "1DkQJuST62GkJP9kss68fHT8ftLf4SmLVT",
    "lot": 567885,
    "sequence": 1
}
------------------------------------------------------------------------------------------
↪-----------------------------------
Intermediate Passphrase:␣
↪passphraseondJwvQGEWFNrNJRPi4G5XAL5SU777GwTNtqmDXqA3CGP7HXfH6AdBxxc5WUKC
Encrypted WIF: {
    "encrypted_wif": "6PnUVPinrvPGwoYJK3GbGBNgFuqEXmfvagE4QiAxj7yrZp4i29p22MrY5r",
    "confirmation_code":
```

(continues on next page)

(continued from previous page)

```
↪"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: {
    "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":
↪"cfrm38VWx5xH1JFm5EVE3mzQvDPFkz7SqNiaFxhyUfp3Fjc2wdYmK7dGEWoW6irDPSrwoaxB5zS",
    "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",
```

(continues on next page)

(continued from previous page)

```
    "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.

CHAPTER

TWO

BIP38

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", public_key_
↪type="compressed")
'02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e'
>>> private_key_to_public_key(private_key=
↪"cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", public_key_
↪type="uncompressed")

↪'04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b
↪'
```

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

Private key to WIF (Wallet Important Format) converter

Parameters

- **private_key** (*Union[str, bytes]*) – Private key
- **wif_type** (*Literal["wif", "wif-compressed"]*) – WIF (Wallet Important Format) type, default to `wif-compressed`
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type

Returns

str – Wallet Important Format

```
>>> from bip38 import private_key_to_wif
>>> private_key_to_wif(private_key=
↪"cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", network=
↪"mainnet", wif_type="wif")
'5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR'
>>> private_key_to_wif(private_key=
↪"cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", network=
↪"mainnet", wif_type="wif-compressed")
'L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP'
>>> private_key_to_wif(private_key=
↪"cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", network=
↪"testnet", wif_type="wif")
'938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX'
>>> private_key_to_wif(private_key=
↪"cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5", network=
↪"testnet", wif_type="wif-compressed")
'cURAYbG6FtvUasdBsooEmmY9MqUfhJ8tdybQWV7iA4BAwunCT2Fu'
```

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

WIF (Wallet Important Format) to Private key converter

Parameters

wif (*str*) – Wallet Important Format

Returns

str – Private key

```
>>> from bip38 import wif_to_private_key
>>> wif_to_private_key(wif="L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP",␣
↪network="mainnet")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'
>>> wif_to_private_key(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR",␣
↪network="mainnet")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'
>>> wif_to_private_key(wif="938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX",␣
↪network="testnet")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'
>>> wif_to_private_key(wif="cURAYbG6FtvUasdBsooEmmY9MqUfhJ8tdybQWV7iA4BAwunCT2Fu",␣
↪network="testnet")
'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5'
```

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

Get WIF (Wallet Important Format) type

Parameters
wif (*str*) – Wallet Important Format

Returns
Literal[“wif”, “wif-compressed”] – WIF type

```
>>> from bip38 import get_wif_type
>>> get_wif_type(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'wif'
>>> get_wif_type(wif="L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP")
'wif-compressed'
>>> get_wif_type(wif="938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX")
'wif'
>>> get_wif_type(wif="cURAYbG6FtvUasdBsooEmmY9MqUfhJ8tdybQWV7iA4BAwunCT2Fu")
'wif-compressed'
```

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

Get WIF (Wallet Important Format) checksum

Parameters
wif (*str*) – Wallet Important Format

Returns
str – WIF checksum

```
>>> from bip38 import get_wif_checksum
>>> get_wif_checksum(wif="L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP")
'dc37f844'
>>> get_wif_checksum(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'f0a25c0c'
```

bip38.bip38.**get_wif_network**(*wif: str*) → Literal['mainnet', 'testnet']

Get WIF (Wallet Important Format) network type

Parameters
wif (*str*) – Wallet Important Format

Returns
Literal[“mainnet”, “testnet”] – Network type

```
>>> from bip38 import get_wif_network
>>> get_wif_network(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR")
'mainnet'
>>> get_wif_network(wif="L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP")
'mainnet'
>>> get_wif_network(wif="938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX")
'testnet'
>>> get_wif_network(wif="cURAYbG6FtvUasdBsooEmmY9MqUfhJ8tdybQWV7iA4BAwunCT2Fu")
'testnet'
```

bip38.bip38.**public_key_to_addresses**(*public_key: str | bytes*, *network: Literal['mainnet', 'testnet'] = 'mainnet'*) → str

Public key to address converter

Parameters

- **public_key** (*Union[str, bytes]*) – Public key
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type

Returns

str – Address

```
>>> from bip38 import public_key_to_addresses
>>> public_key_to_addresses(public_key=
↪"02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e", network=
↪"mainnet")
'164MQi977u9GUteHr4EPH27VkkdxmfCvGW'
>>> public_key_to_addresses(public_key=
↪"04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b
↪", network="mainnet")
'1Jq6MksXQVWzrznvZzxkV6oY57oWXD9TXB'
>>> public_key_to_addresses(public_key=
↪"02d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360e", network=
↪"testnet")
'mkaJhmE5vvaXG17uZdCm6wKpckEfnG4yt9'
>>> public_key_to_addresses(public_key=
↪"04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b
↪", network="testnet")
'myM3eoxWDWxFe7GYHZw8K21rw7QDNZeDYM'
```

bip38.bip38.**intermediate_code**(*passphrase: str*, *lot: int | None = None*, *sequence: int | None = None*, *owner_salt: str | bytes = b'\xd0\x8f(D$\xbd\xbe\xdd'*) → 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*, *network: Literal['mainnet', 'testnet'] = 'mainnet'*) → str
BIP38 Encrypt WIF (Wallet Important Format) using passphrase/password

Parameters

- **wif** (*str*) – Wallet important format
- **passphrase** (*str*) – Passphrase or password text
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type

Returns
str – Encrypted WIF (Wallet Important Format)

```
>>> from bip38 import bip38_encrypt
>>> bip38_encrypt(wif="5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR",␣
↪passphrase="TestingOneTwoThree")
'6PRVWUbkzzsbcVac2qwfssoUJAN1Xhrg6bNk8J7Nzm5H7kxEbn2Nh2ZoGg'
>>> bip38_encrypt(wif="L44B5gGEpqEDRS9vVPz7QT35jcBG2r3CZwSwQ4fCewXAhAhqGVpP",␣
↪passphrase="TestingOneTwoThree")
'6PYNKZ1EAgYgmQfmNVamxyXVWHzK5s6DGhwP4J5o44cvXdoY7sRzhtpUeo'
>>> bip38_encrypt(wif="938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX",␣
↪passphrase="TestingOneTwoThree")
'6PRVWUbkzzsbcVac2qwfssoUJAN1Xhrg6bNk8J7Nzm5H7kxEbn2Nh2ZoGg'
>>> bip38_encrypt(wif="cURAYbG6FtvUasdBsooEmmY9MqUfhJ8tdybQWV7iA4BAwunCT2Fu",␣
↪passphrase="TestingOneTwoThree")
'6PRVWUbkzzsbcVac2qwfssoUJAN1Xhrg6bNk8J7Nzm5H7kxEbn2Nh2ZoGg'
```

bip38.bip38.**create_new_encrypted_wif**(*intermediate_passphrase: str*, *public_key_type: Literal['uncompressed', 'compressed'] = 'uncompressed'*, *seed: str | bytes = b'\xf3\x8d\xfd\xb1\xcf\x8e1\xb9 X\xef+\xec\x96%\x82-\x9fG\xcf\xd79\xa9\xe4'*, *network: Literal['mainnet', 'testnet'] = 'mainnet'*) → dict
Create new encrypted WIF (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)`
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type

Returns
dict – Encrypted WIF (Wallet Important Format)

```
>>> from bip38 import create_new_encrypted_wif
>>> create_new_encrypted_wif(intermediate_passphrase=
↪"passphraseb7ruSN4At4Rb8hPTNcAVezfsjonvUs4Qo3xSp1fBFsFPvVGSbpP2WTJMhw3mVZ",␣
↪network="testnet", public_key_type="uncompressed")
{'encrypted_wif': '6PgMqVygYm6reoPXjsUPxtPDhExjFtUFfHAPYM5svyzVKQmDR1hRy2kgu8',
↪'confirmation_code':
↪'cfrm38V8ZQSdCuzcrYYKGNXVwbHgdjsUEfAbFGoEUouB4YEKaXVdFiMcBWai1Exdu8jN7DcoKtM',
↪'public_key':
↪'04cb64d6e93174b827d3c54965cff210348888bf959f8e198f2483ff9d61d7de3f938e11b7198455fde1f1dea7d3823c
↪', 'seed': 'b8e9e1178307d5863d011a25f4d887f61d3b2531a990fe37', 'public_key_type':
↪'uncompressed', 'address': 'mwW38M23zvDmhbsTdnVFzw3bVnueDhrKec'}
>>> create_new_encrypted_wif(intermediate_passphrase=
↪"passphraseb7ruSN4At4Rb8hPTNcAVezfsjonvUs4Qo3xSp1fBFsFPvVGSbpP2WTJMhw3mVZ",␣
↪network="mainnet", public_key_type="compressed", seed=
↪"99241d58245c883896f80843d2846672d7312e6195ca1a6c")
{'encrypted_wif': '6PoM8coZNg4AGPhQs91RbmmRmfLz6kmnU3XUGGLQxsJ5xN62LsUzMWYcdP',
↪'confirmation_code':
↪'cfrm38VXL5T6qVke13sHUWtEjibAkK1RquBqMXb2azCv1Zna6JKvBhD1Gf2b15wBj7UPv2BQnf6',
↪'public_key': '02100bb0440ff4106a1743750813271e66a7017431e92921e520319f537c7357c1
↪', 'seed': '99241d58245c883896f80843d2846672d7312e6195ca1a6c', 'public_key_type':
↪'compressed', 'address': '15PuNwFmDqYhRsC9MDPNFvNY4Npzibm67c'}
```

bip38.bip38.**confirm_code**(*passphrase: str*, *confirmation_code: str*, *network: Literal['mainnet', 'testnet'] = 'mainnet'*, *detail: bool = False*) → str | dict

Confirm passphrase

Parameters

- **passphrase** (*str*) – Passphrase or password text
- **confirmation_code** (*str*) – Confirmation code
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type
- **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=
↪"cfrm38V8ZQSdCuzcrYYKGNXVwbHgdjsUEfAbFGoEUouB4YEKaXVdFiMcBWai1Exdu8jN7DcoKtM",␣
↪network="testnet")
'mwW38M23zvDmhbsTdnVFzw3bVnueDhrKec'
>>> confirm_code(passphrase="TestingOneTwoThree", confirmation_code=
↪"cfrm38V8Foq3WpRPMXJD34SF6pGT6ht5ihYMWWMbezkzHgPpA1jVkfbTHwQzvuSA4ReF86PHZJY",␣
↪network="mainnet")
'1JbyXoVN4hXWirGB265q9VE4pQ6qbY6kmr'
>>> confirm_code(passphrase="TestingOneTwoThree", confirmation_code=
↪"cfrm38VXL5T6qVke13sHUWtEjibAkK1RquBqMXb2azCv1Zna6JKvBhD1Gf2b15wBj7UPv2BQnf6",␣
↪network="mainnet", detail=True)
{'public_key': '02100bb0440ff4106a1743750813271e66a7017431e92921e520319f537c7357c1',
↪ 'public_key_type': 'compressed', 'address': '15PuNwFmDqYhRsC9MDPNFvNY4Npzibm67c',
↪ 'lot': 199999, 'sequence': 1}
```

bip38.bip38.**bip38_decrypt**(*encrypted_wif: str*, *passphrase: str*, *network: Literal['mainnet', 'testnet'] = 'mainnet'*, *detail: bool = False*) → str | dict

BIP38 Decrypt encrypted WIF (Wallet Important Format) using passphrase/password

Parameters

- **encrypted_wif** (*str*) – Encrypted WIF (Wallet Important Format)
- **passphrase** (*str*) – Passphrase or password text
- **network** (Literal[“mainnet”, “testnet”], default to `mainnet`) – Network type
- **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", network="mainnet")
'5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR'
>>> bip38_decrypt(encrypted_wif=
↪"6PRL8jj6dLQjBBJjHMdUKLSNLEpjTyAfmt8GnCnfT87NeQ2BU5eAW1tcsS", passphrase=
↪"TestingOneTwoThree", network="testnet")
'938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX'
>>> bip38_decrypt(encrypted_wif=
↪"6PRVWUbkzzsbcVac2qwfssoUJAN1Xhrg6bNk8J7Nzm5H7kxEbn2Nh2ZoGg", passphrase=
↪"TestingOneTwoThree", network="mainnet", detail=True)
{'wif': '5KN7MzqK5wt2TP1fQCYyHBtDrXdJuXbUzm4A9rKAteGu3Qi5CVR', 'private_key':
↪'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5', 'wif_type':
↪'wif', 'public_key':
↪'04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b
↪', 'public_key_type': 'uncompressed', 'seed': None, 'address':
↪'1Jq6MksXQVWzrznvZzxkV6oY57oWXD9TXB', 'lot': None, 'sequence': None}
>>> bip38_decrypt(encrypted_wif=
↪"6PRL8jj6dLQjBBJjHMdUKLSNLEpjTyAfmt8GnCnfT87NeQ2BU5eAW1tcsS", passphrase=
↪"TestingOneTwoThree", network="testnet", detail=True)
{'wif': '938jwjergAxARSWx2YSt9nSBWBz24h8gLhv7EUfgEP1wpMLg6iX', 'private_key':
↪'cbf4b9f70470856bb4f40f80b87edb90865997ffee6df315ab166d713af433a5', 'wif_type':
↪'wif', 'public_key':
↪'04d2ce831dd06e5c1f5b1121ef34c2af4bcb01b126e309234adbc3561b60c9360ea7f23327b49ba7f10d17fad15f068b
↪', 'public_key_type': 'uncompressed', 'seed': None, 'address':
↪'myM3eoxWDWxFe7GYHZw8K21rw7QDNZeDYM', 'lot': None, 'sequence': None}
```

CHAPTER

THREE

UTILS

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

INDEX