```
BIP: 129
Layer: Applications
Title: Bitcoin Secure Multisig Setup (BSMS)
Author: Hugo Nguyen <hugo@nunchuk.io>
        Peter Gray <peter@coinkite.com>
        Marko Bencun <marko@shiftcrypto.ch>
        Aaron Chen <aarondongchen@gmail.com>
        Rodolfo Novak <rodolfo@coinkite.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0129
Status: Proposed
Type: Standards Track
Created: 2020-11-10
License: BSD-2-Clause
```

Introduction

Abstract

This document proposes a mechanism to set up multisig wallets securely.

Copyright

Motivation

The Bitcoin multisig experience has been greatly streamlined under BIP-0174 (Partially Signed Bitcoin Transaction). However, what is still missing is a standardized process for setting up multisig wallets securely across different vendors.

There are a number of concerns when it comes to setting up a multisig wallet:

1. Whether the multisig configuration, such as Signer membership, script type, derivation paths and number of signatures required, is correct and not tampered with.
2. Whether the keys or the multisig configuration are leaked during the setup.
3. Whether the Signer persists the multisig configuration in their respective storage, and under what format.
4. Whether the Signer's storage is tamper-proof.
5. Whether the Signer subsequently uses the multisig configuration to generate and verify receive and change addresses.

An attacker who can modify the multisig configuration can steal or hold funds for ransom by duping the user into sending funds to the wrong address. An attacker who cannot modify the configuration but can learn about the keys and/or the configuration can monitor transactions in the wallet, resulting in loss of privacy.

This proposal seeks to address concerns #1, #2 and #3: to mitigate the risk of tampering during the initial setup phase, and to define an interoperable multisig configuration format.

Concerns #4 and #5 should be handled by Signers and are out of scope of this proposal.

Specification

Prerequisites

This proposal assumes the parties in the multisig support BIP-0032, BIP-0322, the descriptor language and AES encryption.

File Extensions

All descriptor and key records should have a `.bsms` file extension. Encrypted data should have a `.dat` extension.

Roles

Coordinator The Coordinator initiates the multisig setup. The Coordinator determines what type of multisig is used and the exact policy script. If encryption is enabled, the Coordinator also distributes a shared secret or shared secrets to the parties involved for secure communication. The Coordinator gathers information from the Signers to generate a descriptor record. The Coordinator distributes the descriptor record back to the Signers.

Signer The Signer is any software or hardware that controls the private keys and can sign using those keys. The Signer is a participating member in the multisig. Its responsibilities include providing its key record -- which contains a public key or an Extended Public Key (XPUB) -- to the Coordinator, verifying that its `KEY` is included in the descriptor record and persisting the descriptor record in its storage.

Setup Process

Round 1

Coordinator

- The Coordinator creates a new multisig wallet creation session. The Coordinator constructs the multisig script and its policy parameters, such as the required number of signatures and the total number of Signers (`M` and `N`).
- The session should expire after some time period determined by the Coordinator, e.g., 24 hours. The timeout allows the encryption key to have lower entropy.

- If encryption is enabled, the Coordinator distributes a secret `TOKEN` to each Signer over a secure channel. The Signer can use the `TOKEN` to derive an `ENCRYPTION_KEY`. Refer to the #Encryption section below for details on the `TOKEN`, the key derivation function and the encryption scheme. Depending on the use case, the Coordinator can decide whether to share one common `TOKEN` for all Signers, or to have one per Signer.
- If encryption is disabled, the `TOKEN` is set to `0x00`, and all the encryption/decryption steps below can be skipped.

Signer

- The Signer initiates the multisig wallet creation session by setting the `TOKEN`. The Signer derives an `ENCRYPTION_KEY` from the `TOKEN`. The Signer can keep the session open until a different value for the `TOKEN` is set.
- The Signer generates a key record by prompting the user for a multisig derivation path and retrieves the `KEY` at that derivation path. Alternatively, the Signer can choose a path on behalf of the user. If the Signer chooses the path, it should try to avoid reusing `KEY`s for different wallets.
- The first line in the record must be the specification version (`BSMS 1.0` as of this writing). The second line must be the hex-encoded `TOKEN`. The third line must be the `KEY`. The `KEY` is a public key or an XPUB plus the key origin information, written in the descriptor-defined format, i.e.: `[{master key fingerprint}/{derivation path}]{KEY}`. The fourth line is a text description of the key, 80 characters maximum. The fifth line must be a `SIG`, whereas `SIG` is the signature generated by using the private key associated with the public key or XPUB to sign the first four lines. The signature should follow BIP-0322, legacy format accepted.
- The Signer calculates the Message Authentication Code (`MAC`) for the record. The first 16 bytes of the `MAC` serves as the Initialization Vector (`IV`) for the encryption.
- The Signer encrypts the key record with the `ENCRYPTION_KEY` and `IV`.
- The Signer encodes the `MAC` and the ciphertext into hexadecimal format, then concatenates the results: `(MAC || ciphertext)`.

Round 2

Coordinator

- The Coordinator gathers key records from all participating Signers. The Coordinator verifies that there are exactly `N` unique key records before the wallet setup session expires.
- For each key record, the Coordinator extracts the `MAC` from the data, sets `IV` to the first 16 bytes of the `MAC`, then decrypts the ciphertext using the `ENCRYPTION_KEY` and `IV`.
- The Coordinator verifies that the included `MAC` is valid given the plaintext.

- The Coordinator verifies that the key records have compatible specification versions.
- The Coordinator verifies that the included `SIG` is valid given the `KEY`.
- If all key records look good, the Coordinator fills in all necessary information to generate a descriptor record.
- The first line in the descriptor record must be the specification version (`BSMS 1.0` as of this writing). The second line must be a descriptor or a descriptor template. The third line must be a comma-separated list of derivation path restrictions. The paths must start with `/` and use non-hardened derivation. If there are no template or restrictions, it must say `No path restrictions`. The fourth line must be the wallet's first address. If there are path restrictions, use the first address from the first path restriction.
- The Coordinator calculates the `MAC` for the record. The first 16 bytes of the `MAC` serves as the `IV` for the encryption..
- The Coordinator encrypts the descriptor record with the `ENCRYPTION_KEY` and `IV`.
- The Coordinator encodes the `MAC` and the ciphertext into hexadecimal format, then concatenates the results: `(MAC || ciphertext)`.
- The Coordinator sends the encrypted descriptor record to all participating Signers.

Signer

- The Signer imports the descriptor record.
- The Signer extracts the `MAC` from the data, sets `IV` to the first 16 bytes of the `MAC`, then decrypts the ciphertext using the `ENCRYPTION_KEY` (derived from the open session) and `IV`.
- The Signer verifies that the included `MAC` is valid given the plaintext.
- The Signer verifies that it can support the included specification version.
- The Signer verifies that it can support the descriptor or descriptor template.
- The Signer checks that its `KEY` is included in the descriptor or descriptor template, using path and fingerprint information provided. The check must perform an exact match on the `KEY`s and not using shortcuts such as matching fingerprints, which is trivial to spoof.
- The Signer verifies that it is compatible with the derivation path restrictions.
- The Signer verifies that the wallet's first address is valid.
- For confirmation, the Signer must display to the user the wallet's first address and policy parameters, including, but not limited to: the derivation path restrictions, `M`, `N`, and the position(s) of the Signer's own `KEY` in the policy script. The total number of Signers, `N`, is important to prevent a `KEY` insertion attack. The position is important for scripts where `KEY` order matters. When applicable, all positions of the `KEY` must be displayed. The full descriptor or descriptor template must also be available for review upon user request.

- Parties must check with each other that all Signers have the same confirmation (except for the `KEY` positions).
- If all checks pass, the Signer must persist the descriptor record in its storage.

This completes the setup.

Encryption

The Token We define three modes of encryption.

1. `NO_ENCRYPTION` : the `TOKEN` is set to `0x00`. Encryption is disabled.
2. `STANDARD` : the `TOKEN` is a 64-bit nonce.
3. `EXTENDED` : the `TOKEN` is a 128-bit nonce.

The `TOKEN` can be converted to one of these formats:

- A decimal number (recommended). The number must not exceed the maximum value of the nonce.
- A mnemonic phrase using BIP-0039 word list. This would be 6 words in `STANDARD` mode. This encoding is not recommended in `EXTENDED` mode as it can result in potential confusion between seed mnemonics and `TOKEN` mnemonics.
- A QR code.
- Other formats.

The flexibility in the data format allows each Signer to customize the User Experience based on its respective capabilities.

Key Derivation The key derivation function is PBKDF2, with PRF = SHA512. Specifically:

```
DKey = PBKDF2(PRF, Password, Salt, c, dkLen)
```

Whereas:

- PRF = SHA512
- Password = "No SPOF"
- Salt = `TOKEN`
- c = 2048
- dkLen = 256
- DKey = Derived `ENCRYPTION_KEY`

Encryption Scheme The encryption scheme is AES-256-CTR.

```
MAC = HMAC-SHA256(HMAC_Key, hex-encoded TOKEN || Data)

IV = First 16 bytes of MAC

Ciphertext = AES-256-CTR-Encrypt(Plaintext, DKey, IV)

Plaintext = AES-256-CTR-Decrypt(Ciphertext, DKey, IV)
```

Whereas:

- DKey = `ENCRYPTION_KEY`
- HMAC_Key = SHA256(`ENCRYPTION_KEY`)
- Data = the plaintext, e.g. the entire key record in round 1 and the entire descriptor record in round 2

The `MAC` is to be sent along with the key and descriptor record, as specified above. Because it is a `MAC` over the entire plaintext, this is essentially an Encrypt-and-MAC form of authenticated encryption.

Descriptor Template

The output descriptor language only supports one-dimensional lists. This proposal introduces a descriptor template to represent multi-dimensional lists:

`XPUB/**`

Whereas `/**` can be replaced by any number of derivation path restrictions.

A descriptor template must be accompanied by derivation path restrictions. Signers should expand the template into concrete descriptors by replacing `/**` with the restrictions.

For example, the following template and derivation path restrictions:

- `wsh(sortedmulti(2,XPUB1/**,XPUB2/**))`
- `/0/*,/1/*`

Should translate to two concrete descriptors:

- `wsh(sortedmulti(2,XPUB1/0/*,XPUB2/0/*))`
- `wsh(sortedmulti(2,XPUB1/1/*,XPUB2/1/*))`

QR Codes

For signers that use QR codes to transmit data, key and descriptor records can be converted to QR codes, following the BCR standard.

Also refer to UR Type Definition for BIP44 Accounts and UR Type Definition for Bitcoin Output Descriptors for more details.

Compatibility

This specification is not backwards compatible with existing multisig implementations.

BSMS is opt-in, meaning existing multisig implementations can continue working as-is, with the caveat that they are likely to have various pitfalls. Some of the problems with existing solutions have been described in the #Motivation section.

To comply with this standard, a Signer must be able to persist the descriptor record in its storage.

To use BSMS for a multisig wallet, the user should wait until all participating Signers in the multisig have implemented BSMS.

Security

This proposal introduces two layers of protection. The first one is a temporary, secret `TOKEN`. The second one is the confirmation of the wallet's first address.

The `TOKEN` is used to encrypt the two rounds of communication between the Signer and the Coordinator. A `MAC` is also generated from the `TOKEN` and plaintext to authenticate the data being exchanged. The `TOKEN` is only needed during the setup phase, and can be safely discarded afterwards. It is not recommended to use the same `TOKEN` for multiple wallet creation sessions.

The wallet's first address, on the other hand, can be used to verify the integrity of the multisig configuration. An attacker who tampers with the multisig configuration must also change the wallet's first address. Parties must check with each other that all Signers confirm to the same address and policy parameters to reduce the chance of tampering.

Privacy

Encryption helps improve the privacy of the wallet by avoiding sharing keys and descriptors in plaintext.

If the parties wish to have stronger privacy, it is recommended to use a higher number of bits for the `TOKEN`, and to completely erase knowledge of the `TOKEN` after the multisig wallet has been set up.

Test Vectors

Mode: `NO_ENCRYPTION` with Public Keys

ROUND 1

- Coordinator
  - M-of-N: 1/2
  - ADDRESS_TYPE: NATIVE_SEGWIT
  - TOKEN: 0x00
- Signer 1
  - MASTER_KEY_FINGERPRINT: 59865f44
  - PRIVATE_KEY (m/48'/0'/0'/2'): L5TXU4SdD9e6QGgBjxeegJKxt4FgATLG1TCnFM8JLyEkFuyH
  - Public Key (m/48'/0'/0'/2'): 026d15412460ba0d881c21837bb999233896085a9ed4e5445bd637c10e5797
  - Legacy signature
  - `signer_1_key.bsms`:

```
BSMS 1.0
00
[59865f44/48'/0'/0'/2']026d15412460ba0d881c21837bb999233896085a9ed4e5445bd637c10e579768ba
Signer 1 key
H6DXgqkCb353BDPkzppMFpOcdJZlpur0WRetQhIBqSn6DFzoQWBtm+ibP5wERDRNi0bxxev9B+FIvyQWq0s6im4=
```

- Signer 2
  - MASTER_KEY_FINGERPRINT: b7044ca6
  - PRIVATE_KEY (m/48'/0'/0'/2'): KwT7BZDWjos4JAdfKi8NqF46Kj3rppTwN8KGhPbzmmugiZioF
  - Public Key (m/48'/0'/0'/2'): 030baf0497ab406ff50cb48b4013abac8a0338758d2fd54cd934927afa57cc20
  - Legacy signature
  - `signer_2_key.bsms`:

```
BSMS 1.0
00
[b7044ca6/48'/0'/0'/2']030baf0497ab406ff50cb48b4013abac8a0338758d2fd54cd934927afa57cc2062
Signer 2 key
H08mGNGN+NxX/snt+6eX2Q1HjjfDkOtotglshHi7xdsBdIrTVMCQbgQ5SdACNZOB2AJcifK11nJj43SvaitSemI=
```

ROUND 2

- Coordinator
  - `my_multisig_wallet.bsms`:

```
BSMS 1.0
wsh(sortedmulti(1,[59865f44/48'/0'/0'/2']026d15412460ba0d881c21837bb999233896085a9ed4e5445bc
No path restrictions
bc1quqy523xu3l8che3s8vja8n33qtg0uyugr9l5z092s3wa50p8t7rqy6zumf
```

Mode: `NO_ENCRYPTION`

ROUND 1

- Coordinator
  - M-of-N: 2/2
  - ADDRESS_TYPE: NATIVE_SEGWIT
  - TOKEN: 0x00
- Signer 1
  - MASTER_KEY_FINGERPRINT: 1cf0bf7e
  - PRIVATE_KEY (m/48'/0'/0'/2'): L3q1sg7iso1L3QfzB1riC9bQpqMynWyBeuLLSKwCDGkHkahB7M
  - XPUB (m/48'/0'/0'/2'): xpub6FL8FhxNNUVnG64YurPd16AfGyvFLhh7S2uSsDqR3Qfcm6o9jtcMYw
  - Legacy signature
  - `signer_1_key.bsms`:

```
BSMS 1.0
00
[1cf0bf7e/48'/0'/0'/2']xpub6FL8FhxNNUVnG64YurPd16AfGyvFLhh7S2uSsDqR3Qfcm6o9jtcMYwh6DvmcBF9q
Signer 1 key
```

```
IB7v+qi1b+Xrwm/3bF+Rjl8QbIJ/FMQ40kUsOOQo1SqUWn5QlFWbBD8BKPRetfo1L1N7DmYjVscZNsmMrqRJGWw=
```

- Signer 2
  - MASTER_KEY_FINGERPRINT: 4fc1dd4a
  - PRIVATE_KEY (m/48'/0'/0'/2'): L4JNkJfLBDyWfTLbKJ1H3w56GUMsvdfjCkzRo5RHXfJ6bdHqn
  - XPUB (m/48'/0'/0'/2'): xpub6EebMbEps7ZcV3FYEnddRsvrFWDrt2tiPmCeM7pPXQEmphvq9ZfJ1
  - Legacy signature
  - `signer_2_key.bsms`:

```
BSMS 1.0
00
[4fc1dd4a/48'/0'/0'/2']xpub6EebMbEps7ZcV3FYEnddRsvrFWDrt2tiPmCeM7pPXQEmphvq9ZfJ1LWFUDjf3vxCe
Signer 2 key
HzUa4Z76PFHMl54flIIF3XKiHZ+KbWjjxCEG5G3ZqZSqTd6OgTiFFLqq9PXJXdfYm6/cnL8IVWQgjFF9DQhIqQs=
```

ROUND 2

- Coordinator
  - `my_multisig_wallet.bsms`:

```
BSMS 1.0
wsh(sortedmulti(2,[1cf0bf7e/48'/0'/0'/2']xpub6FL8FhxNNUVnG64YurPd16AfGyvFLhh7S2uSsDqR3Qfcm6c
/0/*,/1/*
bc1qrgc6p3kylfztu06ysl752gwwuekhvtfh9vr7zg43jvu60mutamcsv948ej
```

Mode: `STANDARD` Encryption

ROUND 1

- Coordinator
  - M-of-N: 2/2
  - ADDRESS_TYPE: NATIVE_SEGWIT
  - TOKEN (hex): a54044308ceac9b7
    - TOKEN (decimal): 11907592390080907703
    - TOKEN (mnemonic): pipe acquire around border prosper swift
  - ENCRYPTION_KEY (hex): 7673ffd9efd70336a5442eda0b31457f7b6cdf7b42fe17f274434df55efa9839
- Signer 1
  - MASTER_KEY_FINGERPRINT: b7868815
  - PRIVATE_KEY (m/48'/0'/0'/2'): KyKvR9kf8r7ZVtdn3kB9ifipr6UKnTNTpWJkGZbHwARDCz5iZ3
  - XPUB (m/48'/0'/0'/2'): xpub6FA5rfxJc94K1kNtxRby1hoHwi7YDyTWwx1KUR3FwskaF6HzCbZMz
  - Legacy signature
  - `signer_1_key.bsms`:

```
BSMS 1.0
a54044308ceac9b7
[b7868815/48'/0'/0'/2']xpub6FA5rfxJc94K1kNtxRby1hoHwi7YDyTWwx1KUR3FwskaF6HzCbZMz3zQwGnCqdiFe
Signer 1 key
H8DYht5P6ko0bQqDV6MtUxpzBSK+aVHxbvMavA5byvLrOlCEGmO1WFR7k2wu42J6dxXD8vrmDQSnGq5MTMMbZ98=
```

- Signer 1 encryption
  - HMAC_KEY (hex): 3d4c422806ba8964c9ee45070cd675c024d96648a0ddb4001325818c84951de2
  - MAC (hex): fbdbdb64e6a8231c342131d9f13dcd5a954b4c5021658fa5afcb3fc74dc82706
  - IV (hex) : fbdbdb64e6a8231c342131d9f13dcd5a
  - CIPHERTEXT (hex): 53f491cfd1431c292d922ea5a5dec3eb8ddaa6ed38ae109e7b040f0f23013e89a89b4d
  - `signer_1_key.dat`:
    `fbdbdb64e6a8231c342131d9f13dcd5a954b4c5021658fa5afcb3fc74dc8270653f491cfd1431c292d9`
- Signer 2
  - MASTER_KEY_FINGERPRINT: eedff89a
  - PRIVATE_KEY (m/48'/0'/0'/2'): Kz1ijnkDXmc65NWTYdg47DDaQgSGJAPfhJG9Unm36oqZPpPY
  - XPUB (m/48'/0'/0'/2'): xpub6EhJvMneoLWAf8cuyLBLQiKiwh89RAmqXEqYeFuaCEHdHwxSRfzL
  - Legacy signature
  - `signer_2_key.bsms`:

```
BSMS 1.0
a54044308ceac9b7
[eedff89a/48'/0'/0'/2']xpub6EhJvMneoLWAf8cuyLBLQiKiwh89RAmqXEqYeFuaCEHdHwxSRfzLrUxKXEBap7nZS
Signer 2 key
H/IHW5dMGYsrRdYEz3ux+kKnkWBtxHzfYkREpnYbco38VnMvIxCbDuf7iu6960qDhBLR/RLjlb9UPtLmCMbczDE=
```

- Signer 2 encryption
  - HMAC_KEY (hex): 3d4c422806ba8964c9ee45070cd675c024d96648a0ddb4001325818c84951de2
  - MAC (hex): 383d05b7351a2cef7cca2850450f5efbbc4a3f8ea35707dda87a3692f0f2ebae
  - IV (hex) : 383d05b7351a2cef7cca2850450f5efb
  - CIPHERTEXT (hex): 71860b7c69f3a7665c3c3e85c45735bff78535a37ec6610b724627c73696820d519a92
  - `signer_2_key.dat`:
    `383d05b7351a2cef7cca2850450f5efbbc4a3f8ea35707dda87a3692f0f2ebae71860b7c69f3a7665c3`

ROUND 2

- Coordinator
  - `my_multisig_wallet.bsms`:

```
BSMS 1.0
wsh(sortedmulti(2,[b7868815/48'/0'/0'/2']xpub6FA5rfxJc94K1kNtxRby1hoHwi7YDyTWwx1KUR3FwskaF6H
/0/*,/1/*
bc1qhs4u273g4azq7kqqpe6vh5wfhasfmrq7nheyzsnq77humd7rwtkqagvakf
```

- Coordinator encryption
  - HMAC_KEY (hex): 3d4c422806ba8964c9ee45070cd675c024d96648a0ddb4001325818c84951de2
  - MAC (hex): 734ce791b466861945e1ef6f74c63faec590793de54831f0036b28d08714b71a
  - IV (hex) : 734ce791b466861945e1ef6f74c63fae
  - CIPHERTEXT (hex): 273cad18a5e1eff37dba6d850749594c9a3fd32b2069e8c69983ea269c5044b6bcaea2
  - `my_multisig_wallet.dat`:
    `734ce791b466861945e1ef6f74c63faec590793de54831f0036b28d08714b71a273cad18a5e1eff37db`

Mode: `EXTENDED` Encryption

ROUND 1

- Coordinator
  - M-of-N: 2/3
  - ADDRESS_TYPE: NESTED_SEGWIT
  - TOKEN for Signer 1 (hex): 108a2360adb302774eb521daebbeda5e
    - TOKEN (decimal): 219849024430335054234100711442034755 50
    - ENCRYPTION_KEY (hex): 63dc1e57dfdc21fa11109d5088be01fb8078a383d2296925ad2b7612b71
  - TOKEN for Signer 2 (hex): d3fabc873b98165254fe18a71b5335b0
    - TOKEN (decimal): 281769005132501859744421970528095647152
    - ENCRYPTION_KEY (hex): 3dc860a53471ec03af14617fef60921cf215b45a9d684462fa65b9d804ad3
  - TOKEN for Signer 3 (hex): 78a7d5e7549453d719150de5459c9ce5
    - TOKEN (decimal): 160378811550692397333855096016467696869
    - ENCRYPTION_KEY (hex): 62b90b4c08c03a0ee872e57aae73f9acfafb6cc09d20b5c9bc0bafaef3361

- Signer 1
  - MASTER_KEY_FINGERPRINT: 793cc70b
  - PRIVATE_KEY (m/48'/0'/0'/1'): L1ZEgZ4zNYxyNc8UyeqwyKW1UHVMp9sxwPgSi3s9SW8mc7Ks
  - XPUB (m/48'/0'/0'/1'): xpub6ErVmcYYHmavsMgxEcTZyzN5sqth1ZyRpFNJC26ij1wYGC2SBKYrg
  - Legacy signature
  - `signer_1_key.bsms`:

```
BSMS 1.0
108a2360adb302774eb521daebbeda5e
[793cc70b/48'/0'/0'/1']xpub6ErVmcYYHmavsMgxEcTZyzN5sqth1ZyRpFNJC26ij1wYGC2SBKYrgt9yariSbn7HI
Signer 1 key
ILG47LpCtjoD9UxL87jo5QFqA90t8g9fDQp/KBojdKgPPGB1pMx2bf9hPdORNZIOdCc/2+Gs6AOs3BEK9ubIuBw=
```

- Signer 1 encryption
  - HMAC_KEY (hex): 1162cdace4ac9fcde1f96924b93714143d057a701de83ebaed248d1c9154f9fd
  - MAC (hex): ea12776c73de4bd5ea57c2d19eb8e0be856ac0d7f5651f7b74be4563d61ba5b1
  - IV (hex) : ea12776c73de4bd5ea57c2d19eb8e0be
  - CIPHERTEXT (hex): a36f34232bff47a853092654a718fea4f5f57d6a1f3d38fede04e2414da12c90cefc24ef6
  - `signer_1_key.dat`:
    `ea12776c73de4bd5ea57c2d19eb8e0be856ac0d7f5651f7b74be4563d61ba5b1a36f34232bff47a8530`

- Signer 2
  - MASTER_KEY_FINGERPRINT: b3118e52
  - PRIVATE_KEY (m/48'/0'/0'/1'): L4SnPjcHszMg3Wi2YYxEYnzM2zFeFkFr5NcLZ18YQeyJwaSFbT
  - XPUB (m/48'/0'/0'/1'): xpub6Du5Jn6eYZE96ccmAc1ZTFPzdnzrvqfG4mpamDun2qZYKywoiQJMC
  - Legacy signature
  - `signer_2_key.bsms`:

```
BSMS 1.0
d3fabc873b98165254fe18a71b5335b0
[b3118e52/48'/0'/0'/1']xpub6Du5Jn6eYZE96ccmAc1ZTFPzdnzrvqfG4mpamDun2qZYKywoiQJMCbS3kWWMr6U3X
Signer 2 key
```

IDK4d/oO0pgfrwRu4Zb8vqlPEmJb9aKT1K2CCnI3RKepVAKs3fZsBrypcCdQfUy1TG/3O5vAR3gjldxcCA1Wzg8=

- Signer 2 encryption
  - HMAC_KEY (hex): 43a4e704bd1bade703023004b00290f1a7b005474a581d869a217068eedf3f57
  - MAC (hex): 4a3ff970d027010e83b4fbf2845a23907a301b3df692a9265e2ca679697ac718
  - IV (hex) : 4a3ff970d027010e83b4fbf2845a2390
  - CIPHERTEXT (hex): c8f4a6a6714eff90aa48cbefe6750c2ee3cc72182eb455e964f0ba59ada3ecd758c29f0fa
  - `signer_2_key.dat`:
    4a3ff970d027010e83b4fbf2845a23907a301b3df692a9265e2ca679697ac718c8f4a6a6714eff90aa4

- Signer 3
  - MASTER_KEY_FINGERPRINT: 842bd2ed
  - PRIVATE_KEY (m/48'/0'/0'/1'): L1ehZHpo2UFHc1yaBWDU4bKVycUwcU2TESm92wbfq6xK6qpZ
  - XPUB (m/48'/0'/0'/1'): xpub6Ex81KopPkEt9hJiWHabYy8LNsSR4A7sUQoFBk9dR8XxHrr4p9HrYW
  - Legacy signature
  - `signer_3_key.bsms`:

```
BSMS 1.0
78a7d5e7549453d719150de5459c9ce5
[842bd2ed/48'/0'/0'/1']xpub6Ex81KopPkEt9hJiWHabYy8LNsSR4A7sUQoFBk9dR8XxHrr4p9HrYWN3NCf5uwfop
Signer 3 key
IL77mML0xo/O9dJn0T5EpQLuyRPPrdpgVJbtsdAugW5iX0MQ3Ci0f8jVnXu68Xm07CYjYGKX8af72jmkQKhNud0=
```

- Signer 3 encryption
  - HMAC_KEY (hex): ab93ce7bf0f91c62a66d00ea9bf5e5c00b854ee2cfc2fb06f6eeff738abcdc26
  - MAC (hex): e82cfcccbd4bd4d3b76e28133eecd13f7362f4a8b4c4baa3e5f6ba2dfb4d69b8
  - IV (hex) : e82cfcccbd4bd4d3b76e28133eecd13f
  - CIPHERTEXT (hex): b44433f0b564ec35a1e71371f25844088084b47402c90d52fee7237167b58a60a28c23
  - `signer_3_key.dat`:
    e82cfcccbd4bd4d3b76e28133eecd13f7362f4a8b4c4baa3e5f6ba2dfb4d69b8b44433f0b564ec35a1e

ROUND 2

- Coordinator
  - `my_multisig_wallet.bsms`:

```
BSMS 1.0
sh(wsh(multi(2,[793cc70b/48'/0'/0'/1']xpub6ErVmcYYHmavsMgxEcTZyzN5sqth1ZyRpFNJC26ij1wYGC2SBK
/0/*,/1/*
3GzMtFXahiu4TpGNGFc4bHMvAcvz5vVQrT
```

- Send to Signer 1:
  - HMAC_KEY (hex): 1162cdace4ac9fcde1f96924b93714143d057a701de83ebaed248d1c9154f9fd
  - MAC (hex): 01bf557b6d44b3fbf07f8ec155cbdec42d85d856e174342563dd83b40ad7c025
  - IV (hex) : 01bf557b6d44b3fbf07f8ec155cbdec4
  - CIPHERTEXT (hex): 617ed25b4b8fd88b806cbebcc1731b071465514a805f7ba2de60e291bc9493f31aa0f9
  - `my_multisig_wallet_for_signer_1.dat`:
    01bf557b6d44b3fbf07f8ec155cbdec42d85d856e174342563dd83b40ad7c025617ed25b4b8fd88b806

- Send to Signer 2:
  - HMAC_KEY (hex): 43a4e704bd1bade703023004b00290f1a7b005474a581d869a217068eedf3f57
  - MAC (hex): 974ba77900c43c463dadaa6eaf24aaeb1b25b443cf155229b719bcbf8b343092
  - IV (hex) : 974ba77900c43c463dadaa6eaf24aaeb
  - CIPHERTEXT (hex): 86288c97a6341974a35015f97fbbc8db7655639c839fc438706f82fce36a82dd17e2d4c
  - `my_multisig_wallet_for_signer_2.dat`: `974ba77900c43c463dadaa6eaf24aaeb1b25b443cf155229b719bcbf8b34309286288c97a6341974a35`
- Send to Signer 3:
  - HMAC_KEY (hex): ab93ce7bf0f91c62a66d00ea9bf5e5c00b854ee2cfc2fb06f6eeff738abcdc26
  - MAC (hex): bb3c93b67d758f244de7ee73e5e61261cea6dff5b3852df8faf265cdf1c73dae
  - IV (hex) : bb3c93b67d758f244de7ee73e5e61261
  - CIPHERTEXT (hex): 7ac33bd9719a3cef6c68e09b3c9677565418933f188bbe50dc70f46329706732fe28ab
  - `my_multisig_wallet_for_signer_3.dat`: `bb3c93b67d758f244de7ee73e5e61261cea6dff5b3852df8faf265cdf1c73dae7ac33bd9719a3cef6c6`

Acknowledgement

Special thanks to Pavol Rusnak, Dmitry Petukhov, Christopher Allen, Craig Raw, Robert Spigler, Gregory Sanders, Ta Tat Tai, Michael Flaxman, Pieter Wuille, Salvatore Ingala, Andrew Chow and others for their feedback on the specification.

References

Related mailing list threads:

- https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-February/018385.html
- https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-April/018732.html