Mycelium Gold Bitcoin

> Technical Instructions

Access via Mycelium Software

To access the funds stored in your coin, you need to import an **encrypted private key** into Mycelium Bitcoin Wallet for Android, Darxx for iOS or any other wallet that supports this format. The private key is password protected. Without a password, it is impossible to access bitcoins.

To import keys into Mycelium or Darxx, you need to scan the QR code of the private key from the coin surface and enter the 15-character encryption password plus the checksum symbol. To do this, go to {”+" button} —> {Advanced & More} —> {Scan} in the Android app; for iOS, launch Darrx, go to {My Darxx} —> {Wallet}, select {Add Mycelium Gold Bitcoin account} and follow the instructions.

If you would like to read or review the implementation used by Mycelium Bitcoin Wallet, you can find it here: github.com/mycelium-com/wallet-android

Tracking Bitcoins

To keep track of your funds, you can import only your MGB **public key** into Mycelium Wallet and use it as a "watch-only" account. You cannot spend funds from such an account, but you can see balance and transactions.

Access Without Mycelium Software

Parsing the QR Code

1. Scan the QR code to get a Base64 encoded string.
2. Decode the Base64 encoded string to get exactly 46 bytes. The Base64 variant used is designed for URLs as specified in RFC 4648, section 5.
3. The first 3 bytes are the magic cookie `0xC4 0x49 0xDC: decoded[0...2]`
4. The next 3 bytes are the header bytes: `H = decoded[3...5]`
5. The next 4 bytes is the random salt: `SALT = decoded[6...9]`
6. The next 32 bytes are the **encrypted private key**: `E = decoded[10...41]`
7. The next 4 bytes are the checksum: `C = decoded[42...45]`

Decoding the 3 Header Bytes

Regard the header as an array of 24 bits and decode the following values:

version = XXXX???? ???????? ????????:

must be 1

network = ????X??? ???????? ????????:

0 = prodnet, 1 = testnet

content = ?????XXX ???????? ????????:

000 = private key with uncompressed **public key**

001 = private key with compressed **public key**

010 = 128 bit master seed

011 = 192 bit master seed

100 = 256 bit master seed

HN = ???????? XXXXX??? ????????:

0 <= HN <= 31

Hr = ???????? ?????XXX XX??????:

1 <= Hr <= 31

Hp = ???????? ???????? ??XXXXX?:

1 <= Hp <= 31

reserved = ???????? ???????? ???????X:

must be zero

AES Key Derivation

1. Make the user enter a 15-character password using characters A-Z, all in upper case. Convert the characters to 15 bytes using normal ASCII conversion. An implementations may use additional checksum characters for password integrity. They are not part of the AES key derivation.
2. Run scrypt using parameters ***N = 2^HN, r = Hr, p = Hp*** on the password bytes and SALT, to derive 32 bytes.
3. The 32 output bytes are used as the 256-bit AES key used for decryption.
4. ~~The next 3 bytes are the header bytes:~~ `H = decoded[3...5]`

Decrypting the Content Data

1. The decryption function is 256-bit AES in CBC mode.
2. Generate the AES initialization vector (IV) by doing a single round of SHA256 on the concatenation of SALT and C, and use the first 16 bytes of the output.
3. Split E into two 16-byte blocks E1 and E2.
4. Do an AES block decryption of E1 into P1 using the derived AES key.
5. X-or the initialization vector onto P1: P1 = P1 xor IV
6. Do an AES block decryption of E2 into P2 using the derived AES key.
7. X-or E1 onto P2: ***P2 = P2 xor E1***
8. The 32 byte plaintext data is the concatenation of P1 and P2: ***P = P1 || P2***
9. If content is 000 or 001 the 32 bytes are a private key.
10. If content is 010 the first 16 bytes are a master seed.
11. If content is 011 the first 24 bytes are a master seed.
12. If content is 100 the 32 bytes are a master seed.

Verifying the Checksum

1. Convert the generated bitcoin address to an array of ASCII bytes

2. Do a single SHA256 operation on the array of bytes

3. The checksum is the first 4 bytes of the output

4. Verify that the calculated checksum equals C. If a wrong password was entered the checksums will not match.

Cryptographic Functions Used

AES - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

SHA-256 - http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

scrypt - http://www.tarsnap.com/scrypt/scrypt.pdf

---

https://000.mycelium.com/