Mycelium Wallet Backup

Creation Date:	November 9, 2019
Made With:	Mycelium Wallet 3.0.0.23
Backup Format:	Mycelium Backup 1.1
Active Records:	8
Archived Records:	0
Total Keys:	8
Total Addresses:	8

This document contains an encrypted backup of your Mycelium Wallet.

The backup contains sensitive data consisting of one or more private keys and an optional master seed. For your protection this data is encrypted with a 15 character random password.

The password was shown on display while creating the backup, and is different for every backup. It is not possible to restore the backup and access your bitcoins without the password.

Write the 15-character password and the checksum character from the display here:

☐☐☐ ☐☐☐ ☐☐☐ ☐☐☐ ☐☐☐ ☐

Alternatively you can write it down elsewhere. Keep it safe!

To import a key or a master seed in the Mycelium wallet you need to scan the corresponding QR code and enter the encryption password.

Note that the embedded PDF viewer in Windows 8 cannot display the QR codes properly.

Reign

Bitcoin Addresses

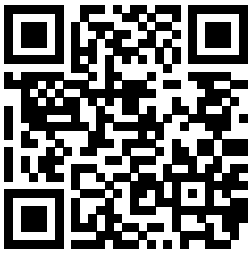
1J8bhbzuZNQwdnQgn
n7EL5Rg2LacZGbk4s

Encrypted Private Key

xEncEHICFQvjmisOqbtqF6PGJUXLZ1Q
FeZbtp798rpQG5fe6iwj2Xk9OB_T0SQ

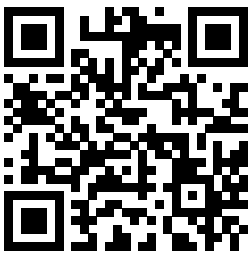
Nov 9, 2019 (2)

Bitcoin Addresses



12XtU1KXJKP4c3fyw
zghsf1Y7aJnLn7FRb

bc1qzr8epcekfrv3ks3r2
94kh769grpr77ed6jv2f5



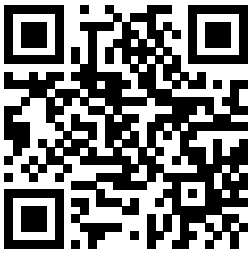
371RkXDcudLCA6BAJ
M4eFsKBoKtrbKS1e7

Encrypted Private Key

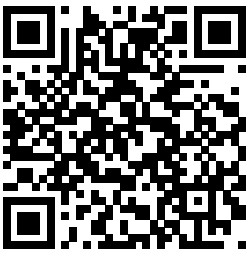
xEncEXICFQvjmoyTsXeLo9T0VN5miwn
CxhWm2bcCTCnKQ6cS37bYdPeQI6zUAA

Nov 9, 2019 (3)

Bitcoin Addresses



1KdN2bc9UXyaoziBC
XwMEaxTiTeDSb4v3w



bc1qe3fv42ph899nss08x
3cvm7n7vcdlx9j33ztq35

3KbhDKedxejZuJ7y4
niJdhhstpBG63tvNE

Encrypted Private Key

xEncEXICFQvjmgDL9ExQXayTvG9DX2z
XetAzi-GfnLRmbvFrz1ZZBARQNAPnAw

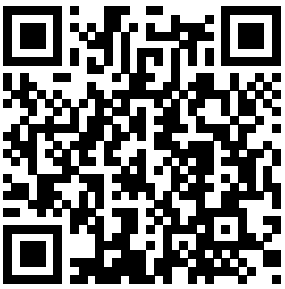
Bitcoin Addresses

1M1vC9fE3kZjrBCrD
zPTxwP7CaEuvE5yh7

bc1qmw8wk959tu5mlakec
jfxkhha8v8rl7mhvfhl43

3KF7P7hz1oCmqW5vX
vZVBZzneKKbZfXayw

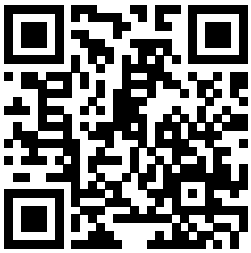
Encrypted Private Key



xEncEXICFQvjmtt0u2MEknG-SI4XdmM
yeZ43tYRDOsp1xE-PRsBmqqwdFqlecA

Nov 9, 2019 (5)

Bitcoin Addresses



1368VSWCowmsdagSx
Lh5pCdbtbVmG2smKo

bc1qzm5ftqx6vv3nup4k7
vfq932upzksnm7f04cqjw

3Mptgryk8i4AY3CPt
tBEW5sLvz6nyGTkBZ

Encrypted Private Key

xEncEXICFQvjmpRubh_qRlumOgym48m
tkU4_mO5EEC96LWbememVqF8XKUITyQ

Bitcoin Addresses

1rx5BNMFEGTr2Ap9
TnzSR36UUNvghjPBz

bc1qp9e9488mhnx6jt9xy
aa6nudgufcwykm7s8ly7m

3FTSu5GfSZPc9Mozh
2rAQCtPHf5Cbewa3A

Encrypted Private Key

xEncEXICFQvjmvVPJ4d6hYAGklMYW2L
c3sZO0gw5pHz0J-nSiVGJPxWvxzFV5w

MT 1 Bitcoin

Bitcoin Addresses

1rx5BNMFEGTr2Ap9
TnzSR36UUNvghjPBz

bc1qp9e9488mhnx6jt9xy
aa6nudgufcwykm7s8ly7m

3FTSu5GfSZPc9Mozh
2rAQCtPHf5Cbewa3A

Encrypted Private Key

xEncEXICFQvjmvVPJ4d6hYAGklMYW2L
c3sZO0gw5pHz0J-nSiVGJPxWvxzFV5w

The Mycelium Bitcoin Wallet performs the steps described below when decrypting and verifying an encrypted private key. The description is quite technical and allows a developer to create software that allows you to decrypt your private keys. This allows you to access your funds if the Mycelium software is no longer available. If you wish to read or review the implementation used by the Mycelium Bitcoin Wallet you can find it here:
https://github.com/mycelium-com/wallet/tree/master/public/bitlib/src/main/java/com/mrd/bitlib/crypto/MrdExport.java

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 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.

Decrypting the Content Data

The decryption function is 256-bit AES in CBC mode.

1. 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.
2. Split E into two 16-byte blocks E1 and E2
3. Do an AES block decryption of E1 into P1 using the derived AES key
4. X-or the initialization vector onto P1: P1 = P1 xor IV
5. Do an AES block decryption of E2 into P2 using the derived AES key
6. X-or E1 onto P2: P2 = P2 xor E1
7. The 32 byte plaintext data is the concatenation of P1 and P2: P = P1 || P2

If content is 000 or 001 the 32 bytes are a private key
If content is 010 the first 16 bytes are a master seed
If content is 011 the first 24 bytes are a master seed
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