

How to setup BTC and XMR cold storage in Qubes OS

Qubes OS mini-summit 2021

Piotr Król





Piotr Król
3mdeb Founder & CEO

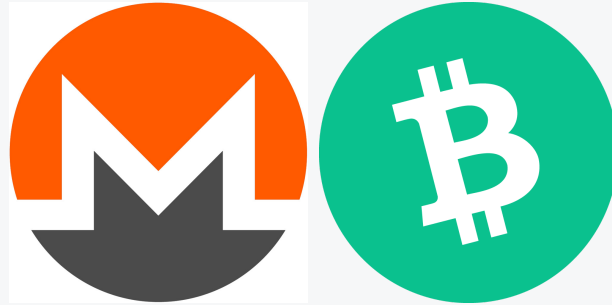
- coreboot contributor and maintainer
- Conference speaker and organizer
- Trainer for military, government and industrial organizations
- Former Intel BIOS SW Engineer
- 12yrs in business
- Qubes OS user since 2016
- C-level positions in





- coreboot licensed service providers since 2016 and leadership participants
- UEFI Adopters since 2018
- Yocto Participants and Embedded Linux experts since 2019
- Official consultants for Linux Foundation fwupd/LVFS project since 2020
- IBM OpenPOWER Foundation members since 2020
 - Our Firmware Engineer Michał is chair of SSWG since 2021

- Presentation goal
- Terminology
- Architecture
- BTC
 - wallet
 - btc-cs VM preparing
 - online watch-only wallet preparing
 - rx/tx coins
- Qrexec and Qubes RPC
- SendToSign and SignTxn RPC services
- BTC Demo
- XMR
 - a/a
- Future ideas
- Q&A

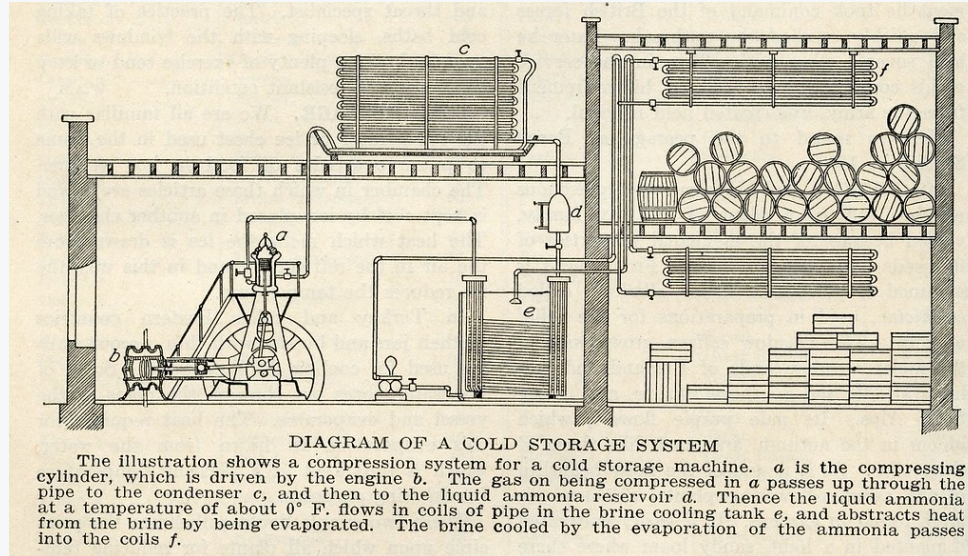


To demo offline wallet for BTC and XMR with Qubes OS

- ***Disclaimer:*** We are by no means cryptocurrency experts and you should not rely on this presentation as source of secure offline wallet configuration. Please consult domain experts. We are not responsible for any damage caused by using following information.

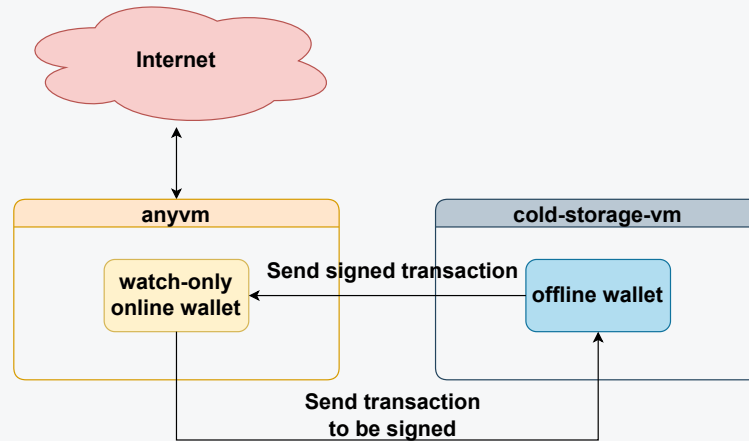
"cryptocurrency monero XMR" by bastamanography CC BY-NC-SA 2.0.

"File:Bitcoin Cash.png" by Amaury Sechet CC0 1.0



- **Cold storage** is an offline wallet used for storing cryptocurrency.
 - no remote access, reduced attack surface
 - public key is used for watching-only wallet
 - every transaction is signed in cold wallet

"Diagram of a Cold Storage System (1920)" by Eric Fischer CC BY 2.0



- Architecture consist of offline and online wallet
 - offline wallet can sign transactions before broadcasting online
 - offline wallet can generate payment requests
 - online wallet is watch-only to see transaction history
- Limitation
 - definitely not the setup for high frequency traders, although there is place for improvements



- Thomas Voegtlin in November 2011
- Wallet is written mostly in Python and its source code is available on Github: <https://github.com/spesmilo/electrum>
- For following tutorial we used v4.1.5 ApplImage version
- Meet our requirements
 - OS: Linux
 - Knowledge: Experienced User
 - License: Open Source
 - Lightweight
- We decided that Electrum meets all those criteria with great balance between privacy, transparency and feature-richness

- Minimal templates contain only most important packages
 - save resources
 - reduce attack surface

```
(dom0)$ sudo qubes-dom0-update qubes-template-debian-10-minimal
(dom0)$ qvm-prefs debian-10-minimal netvm sys-firewall
(dom0)$ qvm-run -u root debian-10-minimal xterm
```

- Install Electrum dependencies

```
(debian-10-minimal)$ apt update
(debian-10-minimal)$ apt install fuse
```

- Create bt-cs VM

```
(dom0)$ qvm-create --label black --property memory=128 --property maxmem=256
--template debian-10-minimal btc-cs
```

```
(trustedvm)$ export ELECTRUM_URL=https://download.electrum.org/4.1.5
(trustedvm)$ wget ${ELECTRUM_URL}/electrum-4.1.5-x86_64.AppImage
(trustedvm)$ wget ${ELECTRUM_URL}/electrum-4.1.5-x86_64.AppImage.ThomasV.asc
(trustedvm)$ wget ${ELECTRUM_URL}/electrum-4.1.5-x86_64.AppImage.sombernight_releasekey.asc
(trustedvm)$ export \
ELECTRUM_KEYS=https://raw.githubusercontent.com/spesmilo/electrum/master/pubkeys
(trustedvm)$ gpg --fetch ${ELECTRUM_KEYS}/ThomasV.asc
(trustedvm)$ gpg --fetch ${ELECTRUM_KEYS}/sombernight_releasekey.asc
```

- Fingerprint can be compared with <https://electrum.org/#about> and Github identity.

```
(trustedvm) $ gpg --verify electrum-4.1.5-x86_64.AppImage.ThomasV.asc \
electrum-4.1.5-x86_64.AppImage
(trustedvm) $ gpg --verify electrum-4.1.5-x86_64.AppImage.sombernight_releasekey.asc \
electrum-4.1.5-x86_64.AppImage
```

- trustedvm should be sufficiently trusted by user to verify signatures

- Copy Electrum from anyvm

```
(anyvm)$ qvm-copy electrum-4.1.5-x86_64.AppImage
```

- Choose btc-cs as target, and click OK
- Run btc-cs

```
(dom0)$ qvm-run btc-cs xterm
```

- In btc-cs you should be able to run Electrum

```
(btc-cs)$ cd ~/QubesIncoming/<anyvm>  
(btc-cs)$ chmod +x electrum-4.1.5-x86_64.AppImage
```

- Run Electrum in testnet

```
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet daemon -d  
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet create | tee seed  
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet load_wallet
```

```
[btc-cs] user@btc-cs: ~  
user@btc-cs:~$ ~/QubesIncoming/fw-dev/electrum-4.1.5-x86_64.AppImage --testnet list_wallets  
[  
  {  
    "path": "/home/user/.electrum/testnet/wallets/default_wallet",  
    "synchronized": false  
  }  
]  
user@btc-cs:~$ ~/QubesIncoming/fw-dev/electrum-4.1.5-x86_64.AppImage --testnet getinfo  
{  
  "auto_connect": true,  
  "blockchain_height": 1895039,  
  "connected": false,  
  "default_wallet": "/home/user/.electrum/testnet/wallets/default_wallet",  
  "fee_per_kb": null,  
  "path": "/home/user/.electrum/testnet",  
  "server": "testnet1.bauerj.eu",  
  "server_height": 0,  
  "spv_nodes": 0,  
  "version": "4.1.5"  
}  
user@btc-cs:~$
```

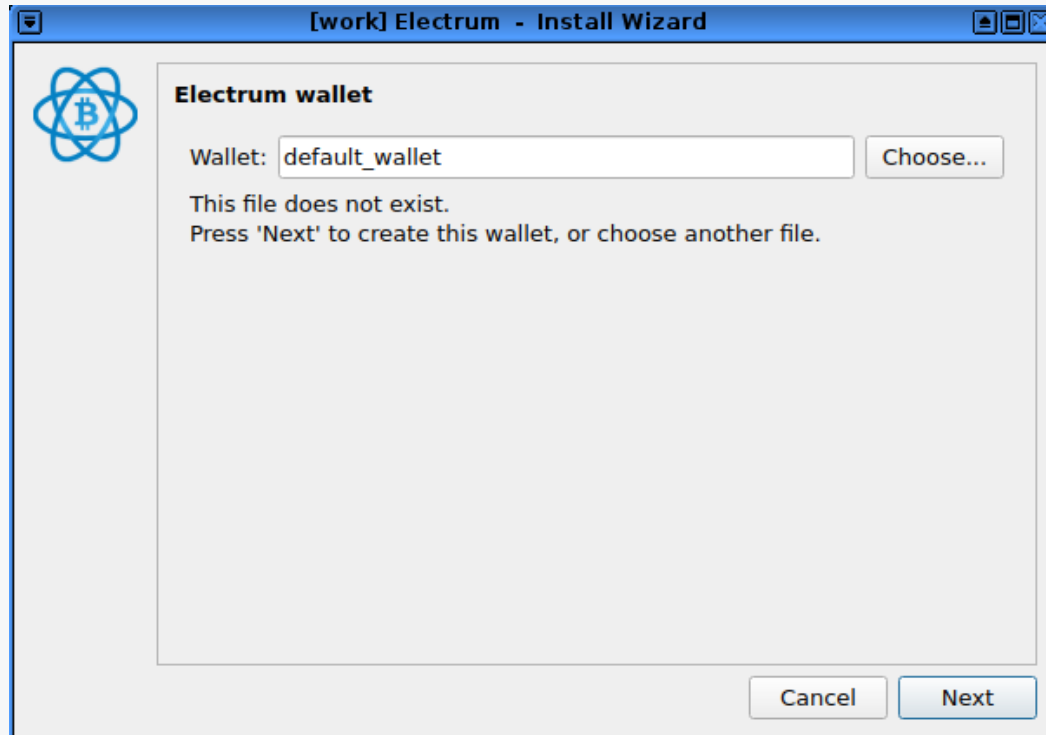
```
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet list_wallets  
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet getinfo
```

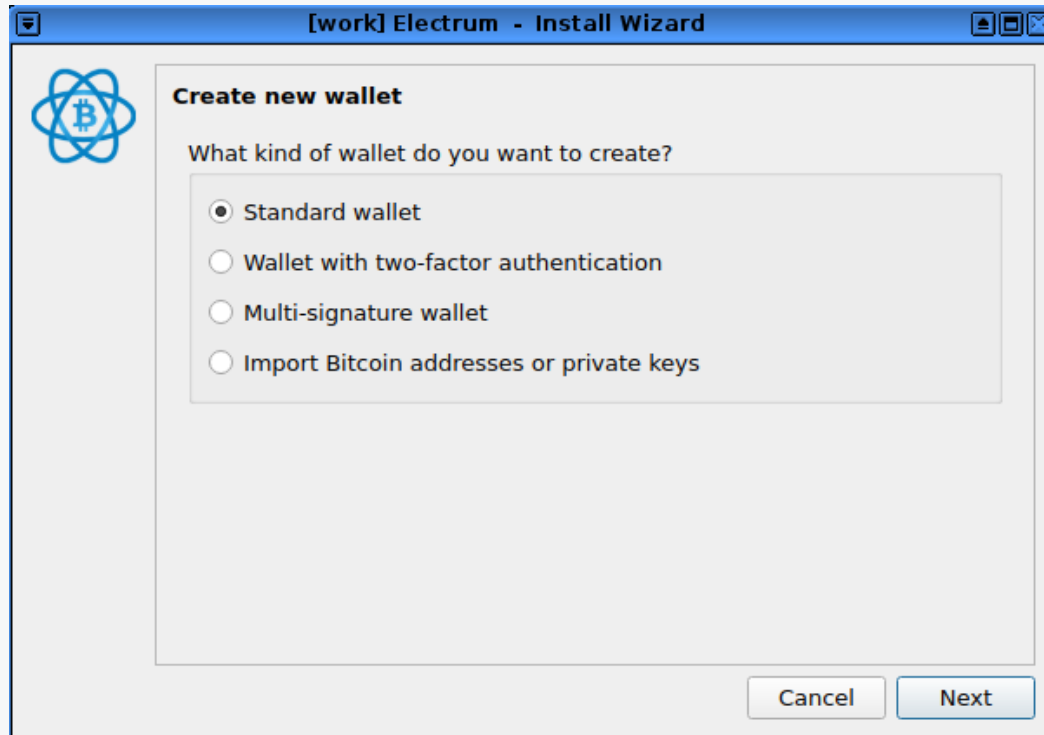


```
[btc-cs] user@btc-cs: ~  
user@btc-cs:~$ ~/QubesIncoming/fw-dev/electrum-4.1.5-x86_64.AppImage --testnet g  
etmpk | tee mpk  
vpub5VvVFBrNmV8AceFQueRccRo6p2fNCHvea85VTyAMKv15EALLRMaUbnhbTrh6xskoUHVh7qgJqHXT  
j2Yqi2DbAR545uqv5Ggurz6myHSqwAL  
user@btc-cs:~$ qvm-copy mpk  
sent 0/1 KB  
user@btc-cs:~$
```

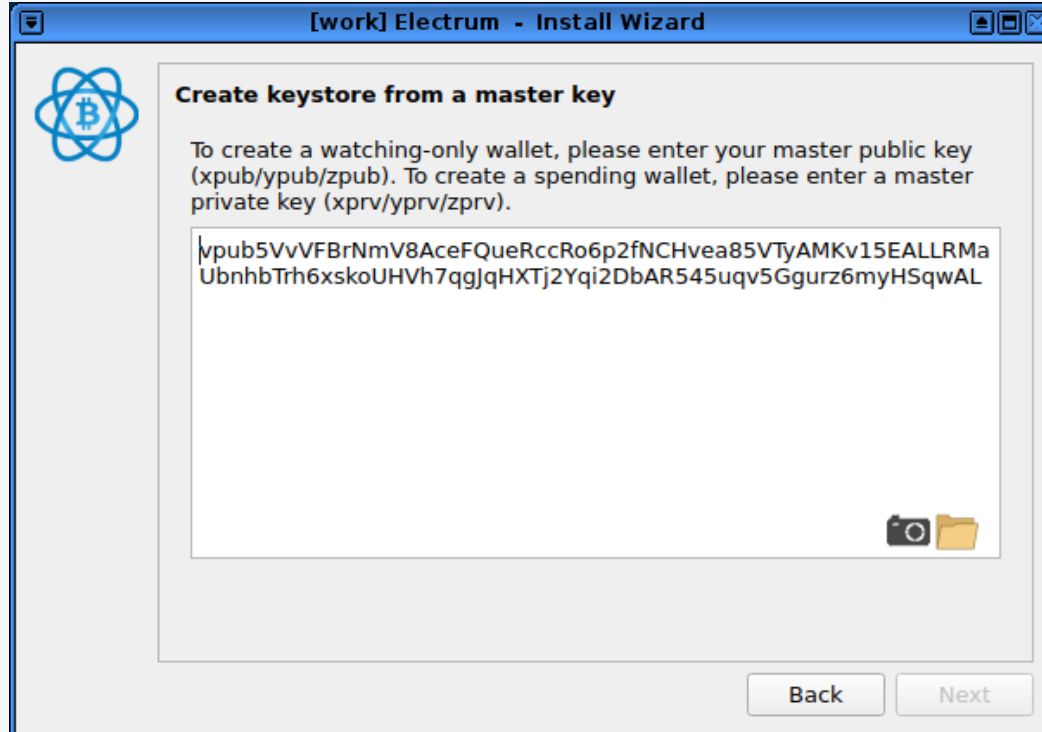
- Obtain Master Public Key and copy it to anyvm
- Run Electron GUI:

```
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet getmpk | tee mpk  
(btc-cs)$ qvm-copy mpk
```

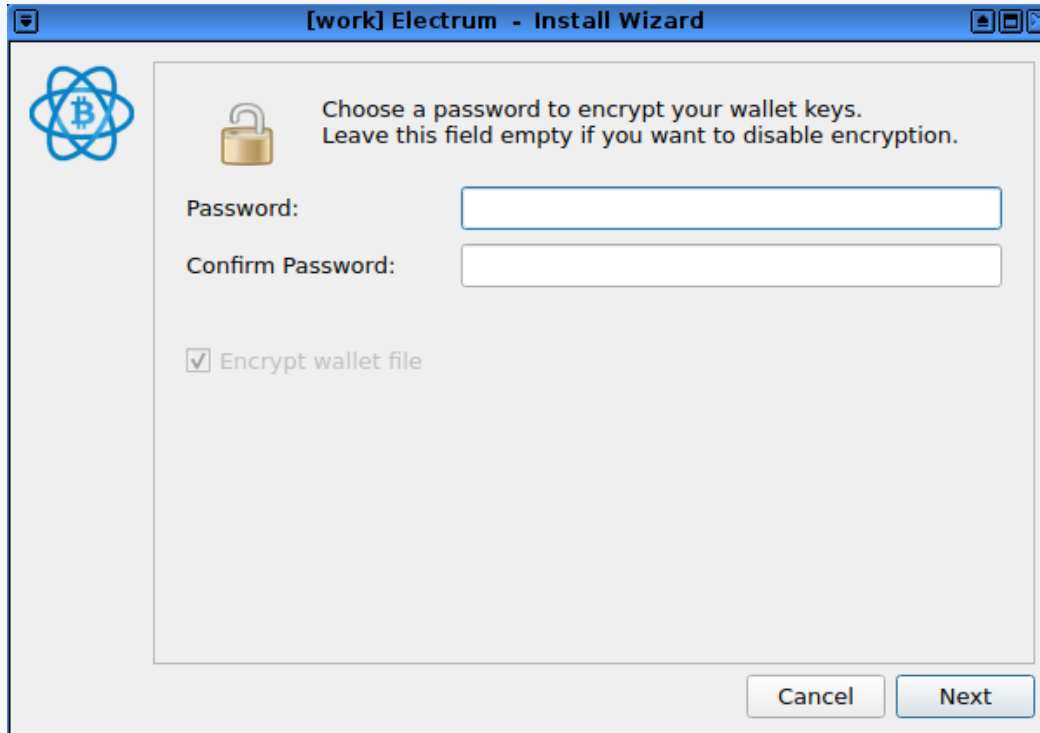










- Most probably Next will be not active, this is because of new line at end of MPK, when you delete it, Electrum can proceed



The screenshot shows the 'Electrum - Install Wizard' window. On the left is the Bitcoin logo. The main area contains a padlock icon and the text: 'Choose a password to encrypt your wallet keys. Leave this field empty if you want to disable encryption.' Below this are two text input fields labeled 'Password:' and 'Confirm Password:'. At the bottom left, there is a checked checkbox labeled 'Encrypt wallet file'. At the bottom right, there are 'Cancel' and 'Next' buttons.

[work] Electrum - Install Wizard

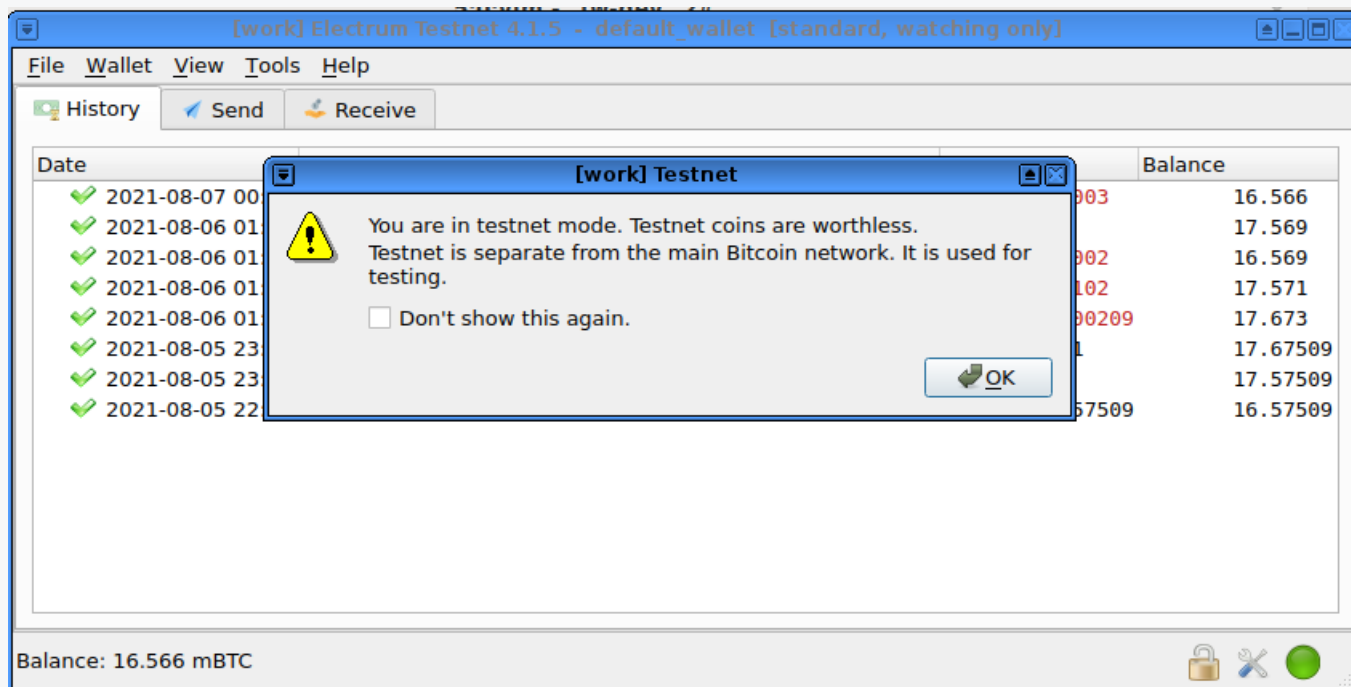
  Choose a password to encrypt your wallet keys.
Leave this field empty if you want to disable encryption.

Password:

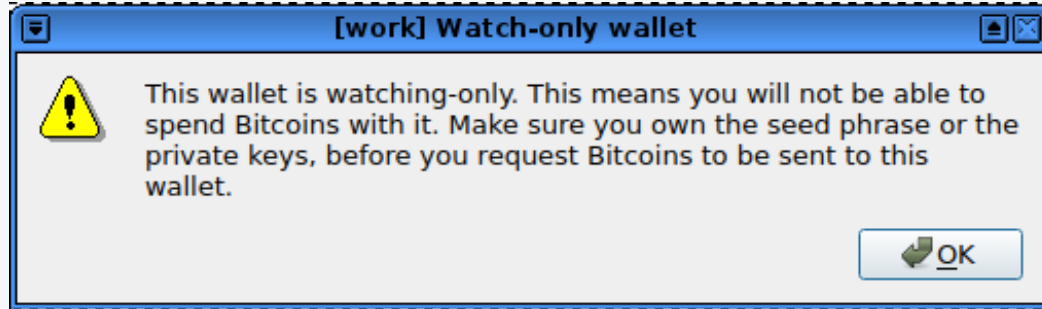
Confirm Password:

☒ Encrypt wallet file

Cancel Next



- Information about wallet working in testnet



- Information about watch-only mode of the wallet

- Create request payment transaction

```
(btc-cs)$ ./electrum-4.1.5-x86_64.AppImage --testnet add_request 0.001
{
  "URI": "bitcoin:tb1q22vafahlyg7ndx6t25qkl0nwle9x9pytn72z(...)",
  "address": "tb1q22vafahlyg7ndx6t25qkl0nwle9x9pytn72znd",
  "amount_BTC": "0.001",
  "amount_sat": 100000,
  "expiration": 3600,
  "is_lightning": false,
  "message": "",
  "status": 0,
  "status_str": "Expires in about 1 hour",
  "timestamp": 1628289375
}
```

- Go to testnet faucet e.g.: <https://testnet-faucet.mempool.co/> and send 0.001BTC to the address from request
tb1q22vafahlyg7ndx6t25qkl0nwle9x9pytn72znd
- You should receive tBTC in online wallet

- Click Pay... -> Send
- Export partial transaction using: Export -> Export to file from transaction menu
- Copy to btc-cs

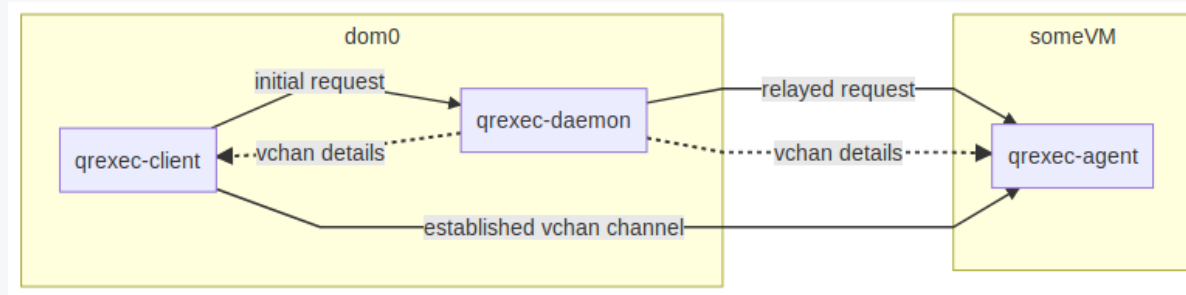
```
(anyvm)$ qvm-copy default_wallet-<txid>.psbt
```

- Sign transaction

```
(btc-cs)$ cat ~/QubesIncoming/<anyvm>/default_wallet-<txid>.psbt|base64 \  
| ./electrum-4.1.5-x86_64.AppImage --testnet signtransaction - > signed_<txid>.txn
```

- Transfer back to anyvm

```
(btc-cs)$ qvm-copy signed_<txid>.txn
```



- Qrexec framework implements communicating between domains
- It is built on top of vchan, Xen library providing data links between VMs
- communication between VMs is set up by dom0
- thanks to the framework RPC client/server are simple scripts
 - `qrexec-client-vm` makes RPC calls to target VM
 - call gets through dom0 and policy is checked
 - script in target VM is executed
 - stdin/stdout can be used to exchange data between client and target VMs

<https://www.qubes-os.org/doc/qrexec/>

```
(dom0)$ qvm-run -u root btc-cs xterm
```

- Create /etc/qubes-rpc/test.SignTxn in btc-cs

```
#!/bin/sh
ELECTRUM="/home/user/QubesIncoming/fw-dev/electrum-4.1.5-x86_64.AppImage"
argument=$(cat -)
if [ -z "$argument" ]; then
    echo "ERROR: No argument given!"
    exit 1
fi
${ELECTRUM} --testnet signtransaction -- "$argument"
```

- Create /etc/qubes-rpc/test.SignTxn in dom0

```
$anyvm btc-cs ask
```

- Create /etc/qubes-rpc/test.SendToSign in anyvm

```
#!/bin/sh  
cat "$1"  
exec cat >&${SAVED_FD_1}
```

Qubes OS mini-summit 2021: Electrum BTC cold wallet demo



- Developed by Monero Community
- Written mostly in C++ with source code available on Github:
<https://github.com/monero-project/monero>
- For the following tutorial we will use Oxygen Orion v0.17.2.0
- Meet our requirements
 - OS: Linux
 - Knowledge: Experienced User
 - License: Open Source
 - Lightweight

- Create xmr-cs based on debian-10-minimal template
- Signature verification according to XMR wallet documentation:
<https://monerodocs.org/interacting/verify-monero-binaries/>
- Copy XMR CLI wallet to xmr-cs
- Start daemon in stagenet mode

```
(xmr-cs)$ ./monerod --stagenet
```

- Create wallet by getting following instructions after running

```
(xmr-cs)$ ./monero-cli-wallet --stagenet
```

- Set following options in wallet

```
set ask-passsword 0
```

- Obtain address

```
(xmr-cs)$ ./monero-wallet-cli --stagenet \  
--wallet-file=/home/user/xmr_stagenet --password "" address \  
| grep address | tee address  
(xmr-cs)$ qvm-copy address
```

- Obtain private view key

```
(xmr-cs)$ ./monero-wallet-cli --stagenet \  
--wallet-file=/home/user/xmr_stagenet viewkey
```

- copy private view key from output and transfer to vm where watch-only wallet will be created

- Use CLI to create watch-only wallet

```
(anyvm)$ ./monerod --stagenet --prune-blockchain  
(anyvm)$ ./monero-wallet-cli --stagenet --generate-from-view-key xmr_watch-only_wallet
```

- When asked provide address and private view key
- Wait to synchronize stagenet blockchain
- After synchronizing blockchain you can start mining in monerod

```
start_mining <address>
```

- When XMR will be mined following message will appear:

```
2021-08-09 08:54:12.965 I Found block  
<155085fe8df9a587467a5e6cce82b61512fbaef90fb8e669468b2d288a9e10d0> at height  
895682 for difficulty: 177850
```

- balance command in wallet should give something like this:

```
Currently selected account: [0] Primary account  
Tag: (No tag assigned)  
Balance: 6.577195618595, unlocked balance: 0.000000000000 (56 block(s) to  
unlock) (Some owned outputs have missing key images - import_key_images needed)
```

- Because XMR has different mechanics then BTC it requires synchronization to keep track of balance on watch-only and cold storage
 - transfers can be monitored by export_outputs on watch-only and import_output on cold storage
 - spent can be monitored by export_key_images on cold storage and import_key_images on watch-only
- You have to wait to see some XMR on unlocked balance this may take time (in my case 1.5h)

```
Untagged accounts:
  Account      Balance      Unlocked balance      Label
*  0 56m3Uc    92.073751406255    13.154240698708    Primary account
-----
      Total    92.073751406255    13.154240698708
[wallet 56m3Uc]:
```

- Wallet run following to create transaction

```
[wallet 56m3Uc]: transfer 55LTR8KnIP4LQGJSptbYDacR7dz8RBFnsfAKMaMuwUX6aQbBcovzDPyrQF9KXF9tV(..) 1
Wallet password:

Transaction 1/1:
Spending from address index 0
Sending 1.000000000000. The transaction fee is 0.000063570000

Is this okay? (Y/Yes/N/No): Y
Unsigned transaction(s) successfully written to file: unsigned_monero_tx
[wallet 56m3Uc]:
```

- Copy unsigned_monero_tx to xmr-cs and sign

```
[wallet 56m3Uc]: sign_transfer
Wallet password:
Loaded 1 transactions, for 6.577195618595, fee 0.000063570000, sending 1.0000000
00000 to 55LTr8KniP4LQGJSptbYDacR7dz8RBFnsfAKMaMuwUNYX6aQbBcovzDPyrQF9KXF9tVU6Xk
3K8no1BywnJX6GvZX8yJsXvt, 5.577132048595 change to 56m3Uc9f96R1mfNE5htDMPRRVmd46
GcVZYiHaMzi8wXGYZUCqqMnKEegVF1VNYnpKFGD1uNGEY9JFWeKTVqFYnX3STsbr8W, with min rin
g size 11, dummy encrypted payment ID. 12 outputs to import. Is this okay? (Y/Y
es/N/No): Y
Transaction successfully signed to file signed_monero_tx, txid 186df9fe7deae2621
d9ccde522c742b324a08b752c4bc325c9e3fdde337cd0d2
[wallet 56m3Uc]:
```

- Copy newly created signed_monero_tx back to xmr-cs and submit transfer

```
[wallet 56m3Uc]: submit_transfer
Loaded 1 transactions, for 6.577195618595, fee 0.000063570000, sending 1.000000000000
0 to 55LTr8KniP4LQGJSptbYDacR7dz8RBFnsfAKMaMuwUNYX6aQbBcovzDPyrQF9KXF9tVU6Xk3K8no1By
wnJX6GvZX8yJsXvt, 5.577132048595 change to 56m3Uc9f96R1mfNE5htDMPRRVmd46GcVZYiHaMzi8
wXGYZUCqqMnKEegVF1VNYnpKFGD1uNGEY9JFWeKTVqFYnX3STsbr8W, with min ring size 11, dummy
encrypted payment ID. 14 key images to import. Is this okay? (Y/Yes/N/No): Y
Transaction successfully submitted, transaction <186df9fe7deae2621d9ccde522c742b324a
08b752c4bc325c9e3fdde337cd0d2>
You can check its status by using the `show_transfers` command.
[wallet 56m3Uc (out of sync)]:
```

- Please note signed_monero_tx file was wallet current working directory

```
(dom0)$ qvm-run -u root xmr-cs xterm
```

- Create /etc/qubes-rpc/test.SignXfer in xmr-cs

```
#!/bin/sh
XMR_WALLET="/home/user/monero-wallet-cli"
cat - > /home/user/unsigned_monero_tx
xterm -e "${XMR_WALLET} --stagenet --wallet-file=/home/user/xmr_stagenet --password '' sign_
cat signed_monero_tx
```

- Create /etc/qubes-rpc/test.SignXfer in dom0

```
$anyvm xmr-cs ask
```

- Create /etc/qubes-rpc/test.SendToSign in anyvm

```
#!/bin/sh
cat "$1"
exec cat >&${SAVED_FD_1}
```

Qubes OS mini-summit 2021: XMR cold wallet demo



- Confirmation of explicit amount spent in transaction to be signed
- Autostart wallet in daemon mode for real transactions
- Consider VM protection mechanisms
- Private key backups
- Disaster recovery
- Signing PBST vs TXN - recognize with what type of file we dealing with
- Salt stack automation of VM creation
- Combining presented configuration using multisig and keeping one of the keys in hardware wallet may improve security of the solution
- Offline wallet software update can be a problem
 - official suggestions saying about complete reinstall
- XMR: improve password handling
- XMR: use RPC instead of cli

Q&A