

The Hodl Guide

This is a draft, all feedback is appreciated!

Bitcoin gives you the possibility to store value in ways earlier generations only could dream about. A common man can store value in a more secure and trust minimized way than any king throughout history. If done right, no one even needs to know what you own.

I wrote this guide in my own quest to becoming a first-class Bitcoin citizen. A first-class Bitcoin citizen has control of his own keys, runs his own full node to validate all transactions and does this in a secure and private way. I'm no technical expert myself, so this guide only uses tools that's been properly battletested throughout the years. The goal of the guide is to find a middle ground in the constant battle between usability and security. More bitcoin has probably been lost because of a complicated process with paper wallets than has been stolen. This guide uses HD-wallets with mnemonic seeds (12-24 regular English words) for your private keys that reduce the risk of fatal mistakes. In the end of the guide, you'll have a reliable and secure **cold storage!**

Who is this for?

The guide is for everyone who wants to store their bitcoin in a secure way (you don't need any specific technical knowledge). You should have a basic understanding of what a private key is and why it's important to store it in a secure way. The guide is flexible and can be utilized in different ways. The full setup is with two different hardware wallets and one backup key created with the Tails operating system. The most time consuming, and slightly more technical, part is related to Tails. You could skip that part for an easier (but less secure) setup. If you don't trust, or want to buy, hardware wallets you could do the whole setup with 3 keys in Tails. If you make modifications to the guide, you'll have to adjust accordingly. The full setup should take 2-3 hours, excluding download and sync-times. The guide is meant for normal users and not individuals that might be extra targeted for specific attacks (but they can probably use parts of the guide).

Privacy

The guide gives you the opportunity to create a very private storage solution as well. Those parts are optional (but recommended). They don't directly affect the security of your private keys. But they reduce the risk of being the victim of a targeted attack (virtual or physical). The privacy parts are marked with a **[P]**. Privacy isn't black and white and every situation is different. I would recommend to at least try a few of the methods that might work for you. It'll probably payoff in the long run. The best security for your bitcoin is probably that no one knows that you own them in the first place. If you want two deep dives about why privacy is very important, check out <https://medium.com/human-rights-foundation-hrf/privacy-and-cryptocurrency-part-i-how-private-is-bitcoin-e3a4071f8fff> and <https://en.bitcoin.it/wiki/Privacy>.

At the end of the guide, you'll have a set of private keys and a plan to store those in a secure way. That includes the possibility for people you trust to access them (if they collaborate with each other) in case of an emergency.

Cold Storage

This is a guide for Cold Storage. Only put funds that you don't need to access frequently in cold storage. If you have more than a few \$1000 worth of bitcoin a good starting point could be to store 70-90% in cold storage, 10-30% in a hardware wallet and 0-10% in a hot wallet on your phone or computer (lightning etc).

I, or no one else, is responsible for your funds. This guide gives you the tools, but you have to evaluate and perform all the steps yourself. Only you are responsible for your bitcoin. Grow a spine and start loving it!

More Reading

For more reading on private keys and other approaches for cold storage, see:

<https://github.com/rustyrussell/bitcoin-storage-guide>

Or for even more depth, take a look at:

<https://glacierprotocol.org/>

Contributions

This guide has one goal. To create a solid cold storage. Any critique or improvement proposal to the guide is highly appreciated. Feel free to open a pull request or contact me on Github or Twitter ([@HelgeHunding](#)).

Preparations

Mandatory Hardware Requirements for Full Setup

- 1 USB-drive with 8GB+ memory. Consider buying a new one, used only for this purpose.
- 2 different Hardware-wallets (Trezor, Ledger, Coldcard etc). Make sure the hardware-wallets are from two different manufacturers (for example one Trezor and one Ledger). You can use existing devices if you already own some (but they shouldn't be used for other purposes after you're done). If you're buying new ones, make sure to only buy from the official store. It's always sensitive to have Bitcoin-related stuff shipped to your home. Consider having it delivered to a UPS access point or a similar service.
- 1 internet connected computer. Could be your normal computer. Should have at least 1 USB-slot and a camera (for reading QR-codes). If you only use one computer, the computer needs to have a 64-bit processor.

- 1 Smartphone (or digital camera with screen)

Optional Equipment and Hardware

- 1 extra computer. This will make the process of working with Tails easier. If you have an old computer lying around, consider using it and making it an eternally quarantined computer (or buy a cheap used computer). That's a computer that's going to be eternally quarantined (only used for this purpose and never used for anything else or connected to the internet again). If you are using an eternally quarantined computer it's a good idea to format/reinstall the computer to get it as fresh as possible. You can even wipe the operating system of the computer since we are using Tails. Before wiping the OS, try it out with Tails. You might have to update some drivers for it to work.
 - The computer needs 1 USB-slot, a camera and needs to have a 64-bit processor. If you're not sure check the computers properties or if you're buying a used computer, you can search for the computer's processor on <http://www.cpu-world.com>. "Data width" should be 64 bit.
- Home safe. For safe, waterproof and fireproof storage for one of the Hardware Wallets.
- TerraSlate paper (<https://www.amazon.com/dp/B076JKVNWY/>), waterproof, heat resistant and tear-resistant paper.
- Envelopes to store sensitive information in.
- Cryptosteel or similar product that stores a private key on a metal plate (then you don't need TerraSlate).
- Tamper-resistant seals or tape.

First steps

An important part of the guide (and a great skill to have) is to know how to validate digital signatures. We are going to do a validation of this guide as a practice (and to control the guide).

To start, download my signature `hundingsbane.asc`, the guide `hodl-guide.pdf` and the detached signature `hodl-guide.pdf.sig` in the [validation-folder](#). We are going to make sure that the one controlling the public key `hundingsbane.asc` (me) signed the document `hodl-guide.pdf`.

To verify what we downloaded, we need GnuPG (<https://gnupg.org/>). The implementation varies for different OS:

Windows: Download and install the latest version of Gpg4win <https://www.gpg4win.org>. If you don't want to donate, click bank transfer on the download page to access the download. You only need to install GnuPG and Kleopatra. Start Kleopatra once finished.

macOS: Download and install the latest version of GPG Suite <https://gpgtools.org/>

Linux: GnuPG comes pre-installed with Linux distributions.

An easy way to verify a digital signature is to use a terminal (the command line). In all examples, what's written to the terminal is everything after the `$` sign (and examples that's specific for Windows uses the symbol `>`).

For example: `$ cd` Means that you'd write `cd` to the command line (`cd` is a command that changes the active directory). Usually you can paste text to a terminal with `ctrl+v` or with a right click on the mouse. Another useful shortcut is to use the arrows up and down to toggle between previously executed commands. If you're stuck, you can usually kill a process with `Ctrl+C` or `Ctrl+Z`.

To start, we need to change the active directory with `$ cd`.

Windows: Open `Powershell` (search for it or use `Win+R`, type `powershell` and hit enter)

macOs: Click the Searchlight (magnifying glass) icon in the menu bar and type `terminal`. Select the Terminal application from the search results.

Linux: Varies, on Ubuntu, press `Ctrl+Alt+T`

Change the current directory to the one where the 3 downloaded files are located, for example:

Windows > `cd C:\Users\User1\Downloads`

macOS and Linux \$ `cd $HOME/Downloads`

To be able to verify the signature, import `hundingsbane.asc` into your local GPG installation:

```
$ gpg --import hundingsbane.asc
```

Now use the “detached signature” to check that the `.pdf` file was signed with the signing key we imported:

```
$ gpg --verify hodl-guide.pdf.sig hodl-guide.pdf
```

Hint: If the name of the file and the signature is the same, you don't need to write the name of the file (you could use only `$ gpg --verify hodl-guide.pdf.sig` in our example).

The expected output should be something like:

```
gpg: Signature made 03/01/19 16:32:53 W. Europe Standard Time
gpg:          using RSA key FF541B4EE4E6D84593011D403D27D7D359A0E4A9
gpg: Good signature from "Helge Hundingsbane" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: FF54 1B4E E4E6 D845 9301 1D40 3D27 D7D3 59A0 E4A9
```

There's three key points to look at when verifying a signature. When the signature was made (03/01/19 in the example above). It should be around the time that the file was uploaded.

That it's a `Good signature` and what the primary key fingerprint is (in this case `FF54 1B4E E4E6 D845 9301 1D40 3D27 D7D3 59A0 E4A9`). To check that the fingerprint indeed seems to belong to me, you could check my Twitter and my Keybase. Signers usually keep their fingerprint in their social profiles. For more critical downloads (like Electrum or Bitcoin

Core) it's a good idea to do a search online on the fingerprint. That should give results on forums etc, that's hard to fake.

[Twitter](#), [Keybase](#)

If someone was trying to fake the guide and change even one character in hodl-guide.pdf, it would result in a bad signature when you verified it with hundingsbane.asc. If you get a bad signature when validating something, you should stop and investigate further before doing anything else with the file. You can open hodl-guide.pdf and select a few parts of the text and compare it to what's written here on Github. If it's not the same, you should stop and investigate further. You can continue with the pdf or on Github.

Now that we have this knowledge, we can start generating private keys! Before moving on, if you don't have Bitcoin Core installed and synced, you might want to start that process now.

[P] Bitcoin Core

We don't want to go through all this trouble to secure our keys and then rely on trusted third parties for validations. That's why our first option should be to use Bitcoin Core to validate all transactions. Bitcoin core is the main software implementation of the Bitcoin protocol. By running your own node, you're also helping the network and making it more decentralized. Check out the bonus guide, [Install and optimize Bitcoin Core](#)

Note, the Bitcoin blockchain size is approaching 300 GB. It could take several days to download the whole blockchain. If storage is an issue, you could use "pruned mode". That way, you'd discard old transactions and only need to store as little as 15 GB of data. You won't be able to help new nodes connect, but you'd still validate everything yourself. If you're on a limited data plan, downloading 300 GB of data might still be an issue. In that case you'd have to rely on third parties for validating and broadcasting transactions and can skip this step.

Generating the first two keys

Time to generate the first keys!

A few things to keep in mind when generating your private keys. This is valuable information and a lot of devices are "spying" on us in one way or the other. If possible, generate all private keys in a room where other electronic devices connected to the internet are turned off. Make sure that your actions aren't visible from windows or doors and that no home security camera records your actions.

Try to keep smartphones turned off and/or in another room during the process. They are hard to secure and can record sound/video without you noticing (computers can of course do the same). Generally, use your common sense to make sure that nothing can record your actions.

Generate keys with Hardware Wallets A and B

The first two keys are generated with your two different hardware wallets. Make sure that the hardware wallets are from two different manufacturers. In that case, if a fatal flaw is found in one of the wallets, your funds are still safe. You can simply move your funds to a new multi-sig contract and replace the flawed key.

Follow the setup procedure recommended from each manufacturer. Protect the devices with a pin (use two different pins for the two devices). We are going to write down the pins on information packages later. The pin is only to protect you if someone gets physical access to the device and isn't super important (but don't use 0000 as pin because of that..). Use PINs you can remember yourself.

Update the firmware on the hardware wallets if you don't have the latest version.

We are going to protect the seed in our hardware wallets with different passwords/passphrases. The method for how to use a password with your seed is different for each manufacturer, check their guides (for example [Ledger](#)). On Ledger it needs to be setup on the device (you can use a temporary passphrase since we won't use it much) and with Trezor you can do it in Electrum later. You don't need to set it up on the devices now, generate the passwords now and put it into the devices when we construct the multi-sig wallet.

Most vulnerabilities that's been detected in hardware wallets would've been stopped with a strong password. We humans are pretty terrible at generating random passwords. So, it's probably safer to generate a password with a password manager on your computer then trying to come up with a password yourself. You could use Lastpass, KeypassX or a similar service. A password manager is a great place to store moderately sensitive information in (like public keys and even more sensitive information like the password that protects the seed, but never put your seed on a "hot" computer). If you are given the option, generate a password without symbols that can be confused (big o and zero etc). If you don't want to use an online service like Lastpass, you could use an encrypted secure note stored on a USB instead. The important part is that this information should be available if your house burns down. We are referring to this as the `digital note` from now on.

I would recommend a password containing symbols from (0-9, a-z, A-Z) and with a length of at least 15 characters. That would give you a password with ~80-bit entropy (on average, it would require 2^{80} guesses to crack the password). Use two different passphrases for your two different hardware wallets. If you already have two old hardware wallets with seeds that you're sure has been setup in a secure manner, you could use those. But make sure they're protected with a strong passphrase (preferably use a new passphrase for this purpose).

We are calling the first Hardware wallet, `Hardware wallet A` (used with `password A`) and the second one `Hardware wallet B` (used with `password B`). It doesn't matter which wallet is which, but make sure to keep track of what you select so you don't mix them later and make sure which private key belongs to which wallet. In your `digital note`, store the passwords like this:

```
PWA: your_password_A
PWB: your_password_B
```

Optional: Depending on your memory (add hints so you can remember the PINs).

Create information packages

Apart from the private keys, we are going to create 3 physical information packages on a different piece of paper. If you want more detail on exactly why we are doing this and how they are being used, look at [Key storage](#).

On each information package, write a short instruction for how to access the funds. Something like this:

```
The Hodl Guide Github, multi-sig 2 of 3
```

That should give enough information for someone else to do a search online for how to retrieve funds in case of an emergency.

Mark the three info packages A, B and C .

While writing down critical information, make sure to be extra careful with symbols that can be confused (like big o and 0, I and small L, etc). The best solution is to not use them at all.

On info package A, write the pin to hardware wallet A `PIN_A: pin_hw_a`.

On info package B, write your first password for the seed `PWA: your_password_A` and the pin to hardware wallet B `PIN_B: pin_hw_b`.

On info package C, write the second password for the seed `PWB: your_password_B`

You should now have:

- Hardware wallet A and its private key (private key 1).
- Hardware wallet B and its private key (private key 2).
- Info package A, B and C.
- A digital note in a password manager or on a USB (that's going to be store in another location), containing `your_password_A` and `your_password_B` (never put anything else from your seed on a computer connected to internet).

Store all information in a secure way during the rest of the process (don't leave your notes lying around visibly).

Generate the last key with Tails

We are generating the last key with Electrum on a computer booted with Tails, <https://tails.boum.org/>. Tails is a live operating system that's built upon Debian (a Unix-like operating system). It's booted from a USB-stick and only uses the computers RAM-memory. That means that all sensitive information is erased when the USB is ejected (and your computer will start with your usual operating system like nothing happened).

Download Tails

Go to <https://tails.boum.org/install/index.en.html> and select your operating system. If you don't have an old copy of Tails, select "Install from {Your operating system}". Select "Let's go" and download the USB image at step 1.1 to a directory on your computer. At the download page, make sure to download `tails-signing.key` and `tails-amd64-3.12.1.img.sig` (at the "OpenPGP signature for the Tails 3.12.1 USB image" link, the exact name should change for future versions) and place the files in the same directory as the `.img` file. Wait until the USB image is downloaded.

Verify signatures

We need to verify what we downloaded (and that's why we downloaded the last two files). As we are verifying the download ourselves, skip step 1.2 "Verify your download using your browser". The browser extension is probably fine, but it's a great practice to do it yourself as the process can be used for other files as well. And browser extensions always comes with a risk of leaking personal information, so always be cautious with browser extensions (no idea if this particular extension does that, probably not).

It's the same procedure as before (check the [Preparations](#) for more information about validating signatures).

Start a terminal window (like Powershell on Windows).

Change the current directory to the one where the 3 downloaded files are located, for example:

```
$ cd $HOME/Downloads
```

To be able to verify the signature, import the Tails-signing key into your local GPG installation:

```
$ gpg --import tails-signing.key
```

Now use the "detached signature" to check that the `.img` file was signed with the signing key we imported:

```
$ gpg --verify tails-amd64-3.12.1.img.sig tails-amd64-3.12.1.img (make sure to change the file name on both places if using a newer version). The verification can take a while.
```

Expected output should be something like:

```
gpg: Signature made 01/28/19 18:44:16 W. Europe Standard Time
gpg: using RSA key FE029CB4AAD4788E1D7828E8A8B0F4E45B1B50E2
gpg: Good signature from "Tails developers (offline long-term identity key)
<tails@boum.org>" [unknown]
gpg: aka "Tails developers <tails@boum.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: A490 D0F4 D311 A415 3E2B B7CA DBB8 02B2 58AC D84F
```


Subkey fingerprint: FE02 9CB4 AAD4 788E 1D78 28E8 A8B0 F4E4 5B1B 50E2

The important part is the date, Good signature and Primary key fingerprint A490 D0F4 D311 A415 3E2B B7CA DBB8 02B2 58AC D84F This ensures that the .img file was signed with a key with the fingerprint A490 D0F4... at a date, not too long before the release. If we do an online search, one of the first matches was [@Tails_live](#) on what seems to be a legit Twitter account and who has the fingerprint in the bio.

Another match is a post from 2015 on the official webpage that announces a change to this key and we can also see several old posts from Reddit. We can be almost certain (not possible to do much more without meeting the signer IRL and get the signature confirmed that way) that this release was indeed signed by the developers. This process would catch a scenario where a malicious actor had taken control over the webpage and uploaded a bad file and a bad signing key. If you get a bad signature or another fingerprint, stop and investigate further before installing anything.

If everything is good, go ahead and create the boot USB. This is going to be used to create the last private key.

Flash Tails to USB

The easiest way is to follow the instructions on <https://tails.boum.org/install> for your operating system.

So, insert the USB you are going to use and follow the instructions at “2/5 Install Tails”. In Mars 2019 that means downloading, running and flashing the USB with Etcher for Windows and macOS and with GNOME Disks for Linux. When step 2/5 is completed, you’ve got two choices.

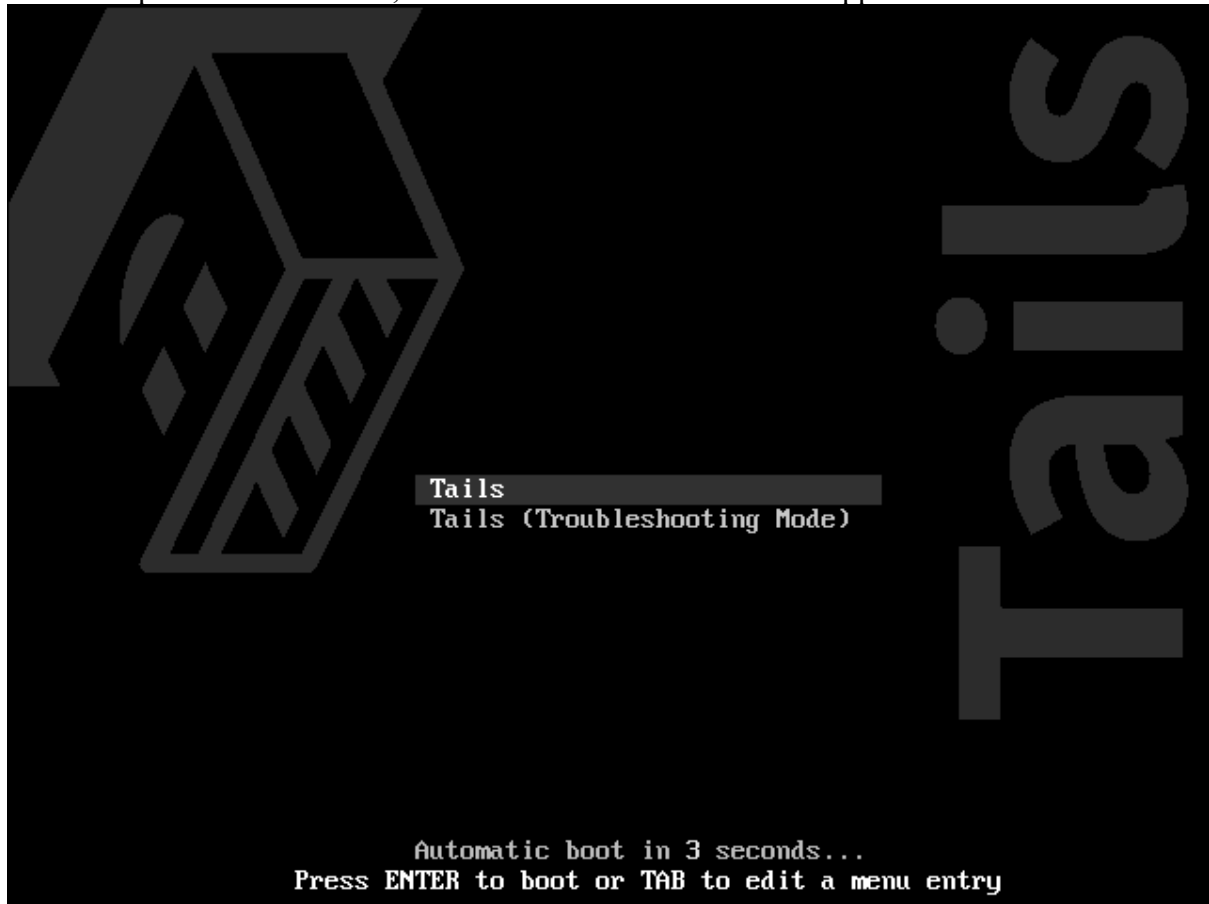
If you use one computer. Tails is going to be started on the computer you are reading this on. In that case you’re going to have to print or write down the rest of the of the instructions on this page. You could also bring the instructions up on another device (keep them in flight mode and don’t let any cameras see any screens or private keys). Make sure no other USB drives is connected to the computer.

If you are using two computers. Tails is going to be started on the other computer and you can keep the instructions on the main computer. The other computer could be a normal computer that's going to be used for other things after the process. Or for extra security, an eternally quarantined computer used only for this purpose. Make sure no other USB drives is connected to the computer.

Start computer on Tails

Go ahead and make sure that the USB with Tails is inserted in the right computer. Start (or restart) the computer.

If the computer starts on Tails, the "Boot Loader Menu" should appear:



Tails should be selected and start automatically after a few seconds. If it does, follow the instructions on the screen for starting (setting up language and keyboard) and start Tails.

Otherwise, restart the computer and bring the boot menu up. The key for bringing the boot menu up differs from manufacturer to manufacturer.

On PC, It's usually Esc or F12. Usual keys used for the boot menu on different manufacturers:

Manufacturer	Key
Acer	Esc, F12, F9
Asus	Esc, F8
Dell	F12
Fujitsu	F12, Esc
HP	Esc, F9
Lenovo	F12, Novo, F8, F10
Samsung	Esc, F12, F2
Sony	F11, Esc, F10
Toshiba	F12
others...	F12, Esc

Choose the USB-stick with Tails in the boot menu. If that doesn't work, you might have to change settings in Bios (turn off secure start, change start order, disable fast start etc).

Check the troubleshooting guide at the install page at Tails for more information. There could be issues with certain computers or graphic cards, check the troubleshooting guide to see if there's an easy fix.

On Mac, Immediately press-and-hold the Option key (Alt key) until a list of possible start-up disks appears. If the computer starts on Tails, the Boot Loader Menu appears and Tails starts automatically after 4 seconds.

When you get past the boot loader menu, follow the instructions on the screen to start Tails.

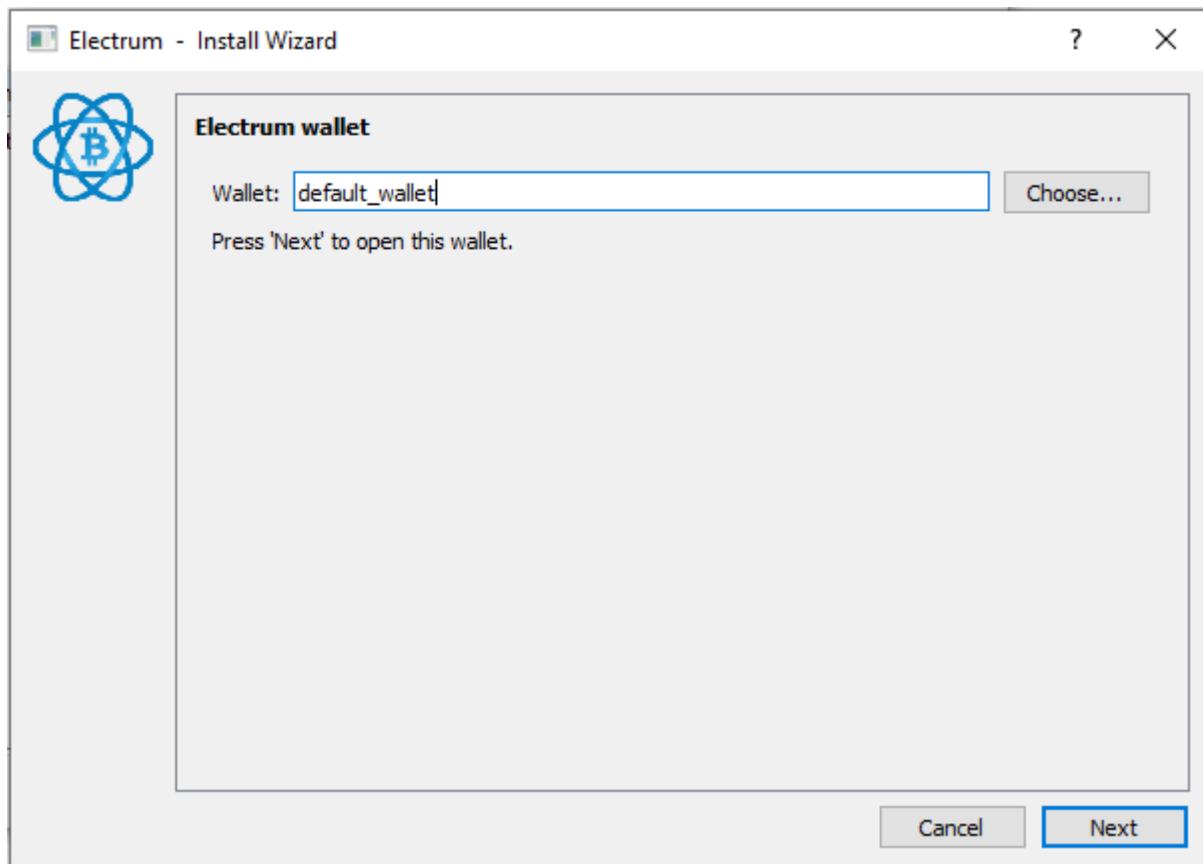
Note: Tails can be pretty slow if you use an old computer, make sure it's really stuck and not only slow before troubleshooting.

Generating the private key with Electrum


Once started, make sure WiFi is disconnected (arrow in upper right corner, should say "Wi-Fi not connected").

Go to Applications (upper left corner)>Internet>Electrum Bitcoin Wallet. Click on `Launch` if a window about persistence appears.

The Electrum – Install Wizard should appear. The name of the wallet isn't important (it will be deleted), so "default_wallet" is ok. Click Next:



On the next step, let “Standard wallet” be selected (we are only interested in generating a key). Click Next:




Create wallet

What kind of wallet do you want to create?

- Standard wallet
- Wallet with two-factor authentication
- Multi-signature wallet
- Import Bitcoin addresses or private keys

Cancel Next

On the next step, let “Create a new seed” be selected. Click Next:



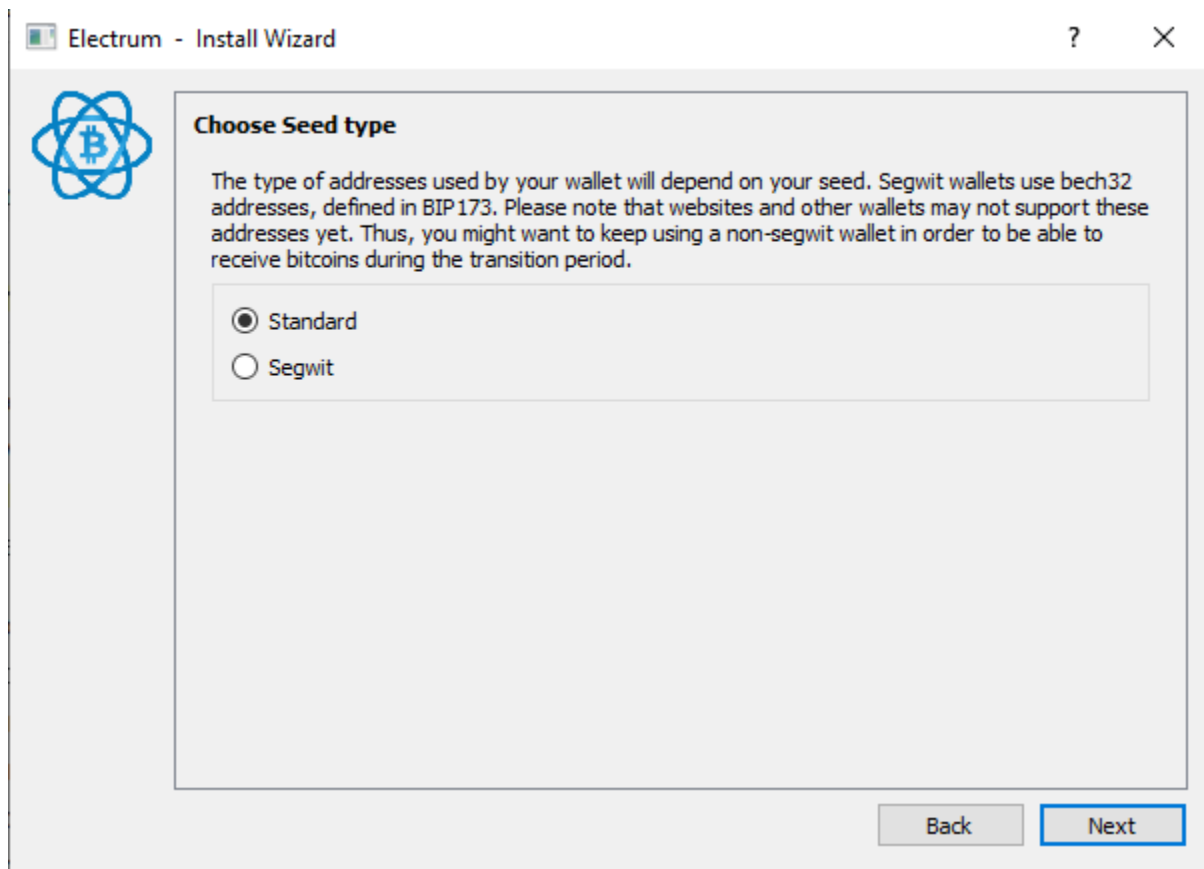
Keystore

Do you want to create a new seed, or to restore a wallet using an existing seed?

- Create a new seed
- I already have a seed
- Use a master key
- Use a hardware device

Back Next

On the next step, let standard be selected. Click Next:



You should now see 12 words, this is your private key! Electrum use 12 words for it's seed, most hardware wallets use 24. Today the difference doesn't really matter, the security is plenty (and we have two other keys with 24 words). Write the 12 words you see on your screen on a paper of good quality. Then click Next:



Your wallet generation seed is:



ghost easy cup drift boss drop basket category smoke tissue enrich patch

Options

Please save these 12 words on paper (order is important). This seed will allow you to recover your wallet in case of computer failure.

WARNING:

- Never disclose your seed.
- Never type it on a website.
- Do not store it electronically.

Back

Next

Confirm your seed by typing all the words you wrote down in the blank field. When finished, click Next:



Confirm Seed

Your seed is important! If you lose your seed, your money will be permanently lost. To make sure that you have properly saved your seed, please retype it here.



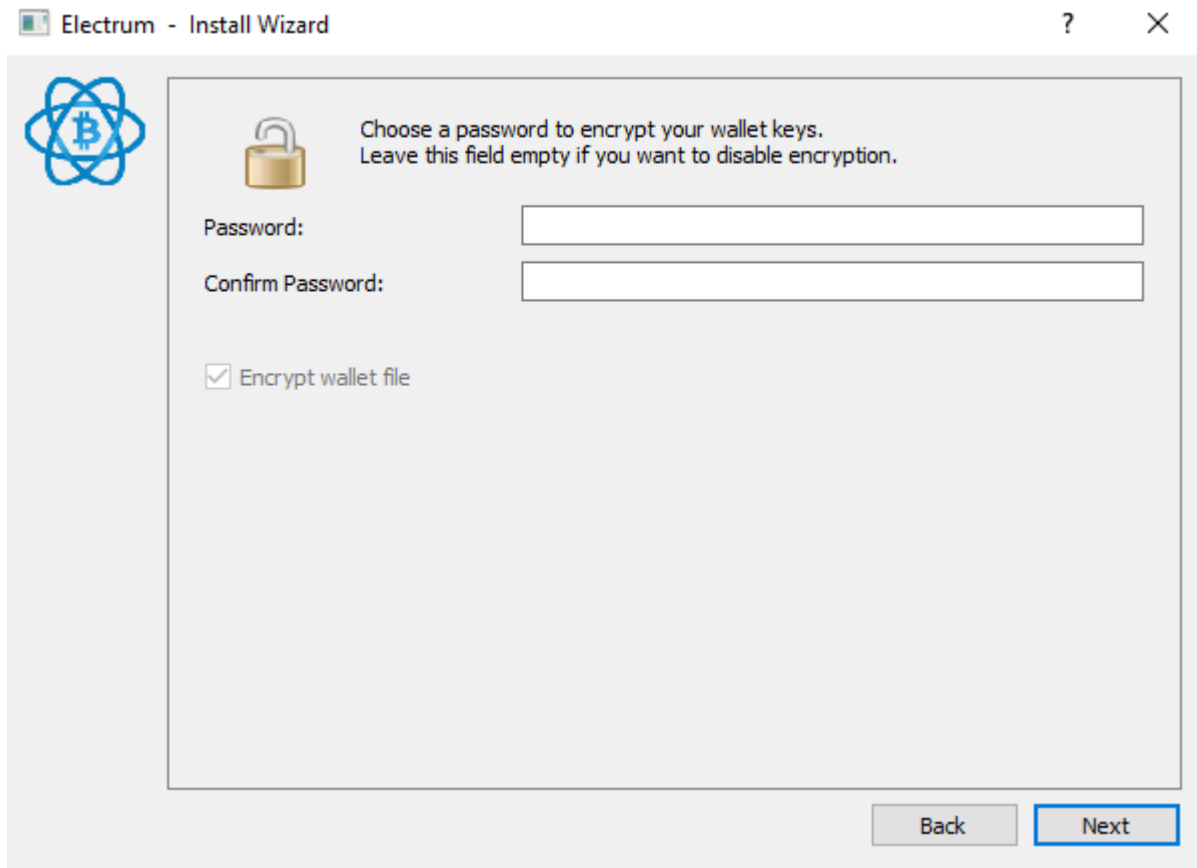
ghost easy cup drift boss drop basket category smoke tissue enrich patch |

Back

Next

We are done with the seed for now, put it away during the rest of the process so it's not visibly lying around.

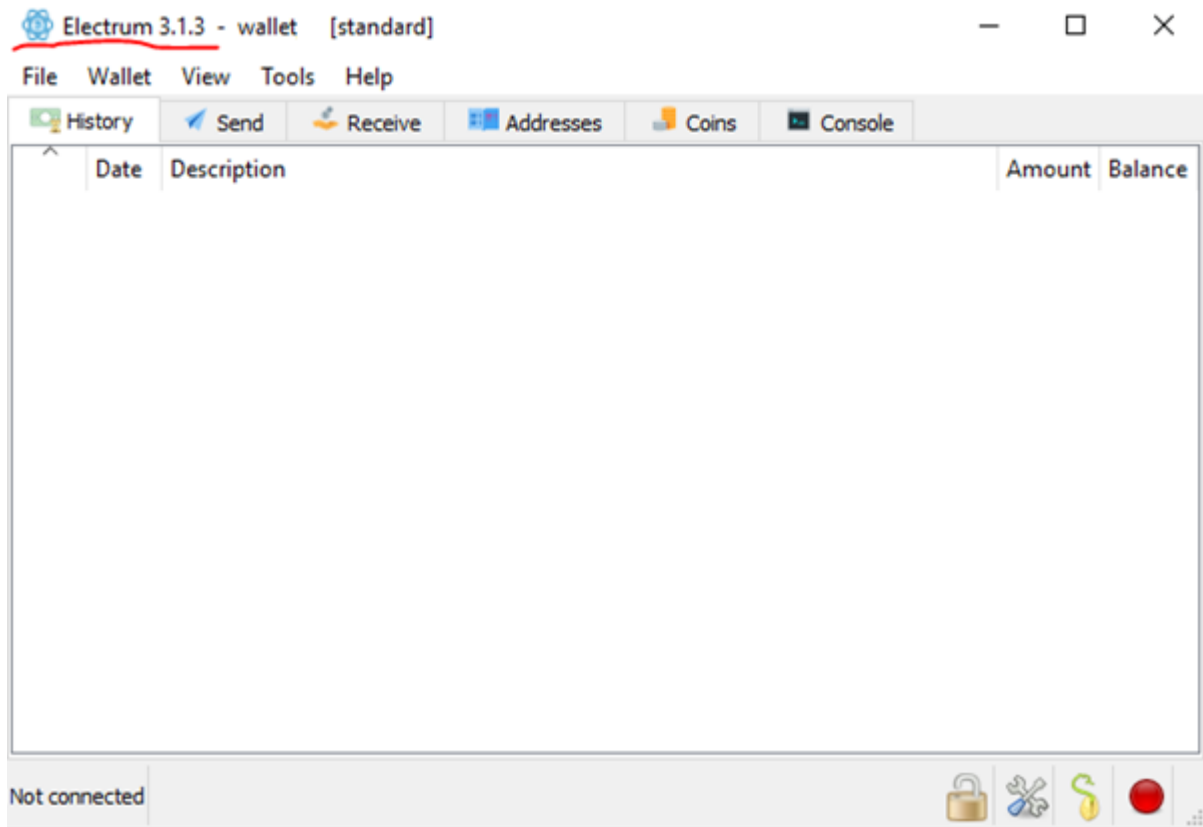
You should now be asked for a password. This is for protecting the wallet file and we don't need that (since everything is deleted once finished), so leave it blank and click Next:



Electrum is now generating addresses, it can take a while before the main window shows up. Once the main window loads, the key and all necessary information is now generated!

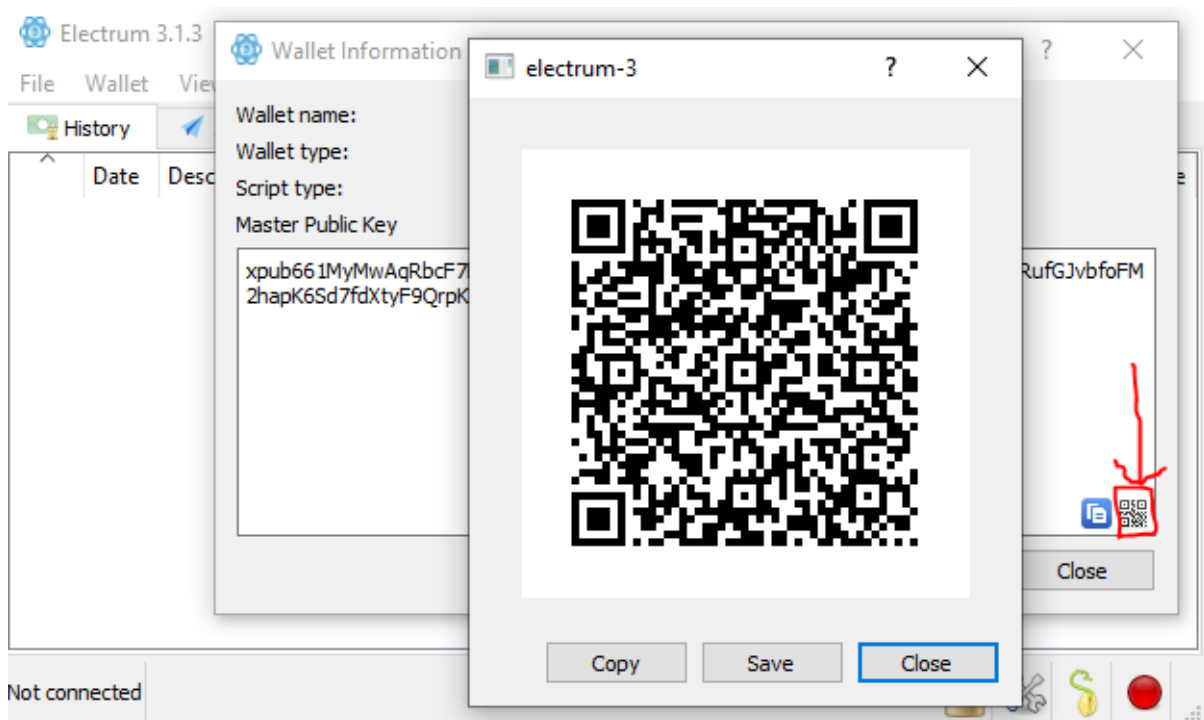
Before we move on, we need to check what version of Electrum we are running and what our master public key is.

Start with the version. Check what version of Electrum that's running on Tails. For example, Tails 3.12.1 comes with Electrum 3.1.3. Remember this or note it down (not sensitive information).



Now we need to copy the master public key. The master public key is used to construct our multi-sig wallet later. To show the master public key in Electrum, go to `Wallet>Information`.

We need to copy this to our live system where we'll construct the multi-signature wallet. But we don't want to put another USB in to our system at this point (reduce any risk of information about our private key leaking). Your public master key doesn't really affect your bitcoin's security (no one can steal your funds with a public key). But all your bitcoin-addresses can be generated from the master public key (in a multi-sig you would need all 3 public keys). So, for privacy, it should be treated with care. But it isn't as sensitive as a private key. The key can be represented as a QR code. In the bottom right corner, click "Show as QR-code":



It's now safe to bring other electronic devices near the computer that generated the private key. So, you can turn your cell phone on, but put it in flight mode so nothing is uploaded to any cloud service (or use a digital camera). We are going to use several cameras, so double check that no private keys are lying around. With your cell phone, take a photo of the QR-code that represents the master public key.

If you use one computer, remove the tails boot USB from the computer and restart the computer on your regular OS. You can keep Tails running if you use two computers, but close Electrum.

Create a multi-sig wallet with Electrum

We are using Electrum on our main computer to construct the multi-signature wallet.

Download and verify Electrum

On your main computer (or regular OS) go to <https://electrum.org/#download>

For our multi-sig to work on both systems, you might need the same version of Electrum as the one used in Tails. But first, we need the signing key of Electrum developer Thomas Voegtlin. Scroll down to the bottom of the page and click on the "Public Key" link (you can skip this on Linux and use `gpg --import ThomasV.asc`):

https://electrum.org/#download

Linux	Install dependencies:	<code>sudo apt-get install python3-pyqt5</code>
	Download package:	<code>wget https://download.electrum.org/3.3.4/Electrum-3.3.4.tar.gz</code>
	Verify signature:	<code>wget https://download.electrum.org/3.3.4/Electrum-3.3.4.tar.gz.asc</code> <code>gpg --verify Electrum-3.3.4.tar.gz.asc</code>
	Run without installing:	<code>tar -xvf Electrum-3.3.4.tar.gz</code> <code>python3 Electrum-3.3.4/run_electrum</code>
	Install with PIP:	<code>sudo apt-get install python3-setuptools python3-pip</code> <code>python3 -m pip install --user Electrum-3.3.4.tar.gz[fast]</code>
Windows	Install PyQt5	
	Install Electrum-3.3.4.zip	In the electrum directory, run 'python3_run_electrum'
OSX	Install Homebrew	
	Install dependencies	<code>'brew install qt5 pyqt5 gettext'</code>
	Install Electrum-3.3.4.zip	In the electrum directory, run 'python3_run_electrum'

How to verify GPG signatures

GPG signatures are a proof that distributed files have been signed by the owner of the signing key. For example, if this website was compromised and the original Electrum files had been replaced, signature verification would fail, because the attacker would not be able to create valid signatures. (Note that an attacker would be able to create valid hashes, this is why we do not publish hashes of our binaries here, it does not bring any security).

In order to be able to verify GPG signatures, you need to import the public key of the signer. Electrum binaries are signed with ThomasV's [public key](#). On Linux, you can import that key using the following command: `gpg --import ThomasV.asc`. Here are tutorials for [Windows](#) and [MacOS](#). When you import a key, you should check its fingerprint using independent sources, such as [here](#), or use the [Web of Trust](#).

Note for Windows users: Electrum binaries are often flagged by various anti-virus software. There is nothing we can do about it, so please stop reporting that to us. Anti-virus software uses heuristics in order to determine if a program is malware, and that often results in false positives. If you trust the developers of the project, you can verify the GPG signature of Electrum binaries, and safely ignore any anti-virus warnings. If you do not trust the developers of the project, you should build the binaries yourself, or run the software from source. Finally, if you are really concerned about malware, you should not use an operating system that relies on anti-virus software.

That should take you to a page with the public key, use `Ctrl+S` and save the file `ThomasV.asc` on your computer.

Go back to <https://electrum.org/#download> and check what the latest release is. Chances are that the latest release is newer than the one used in Tails. If that's the case we need an older release, click "Previous releases":

https://electrum.org/#download

ELECTRUM Bitcoin Wallet

Warning: Versions of Electrum older than 3.3.3 are vulnerable to a [phishing bitcoin-stealing malware](#). Do not download Electrum upgrades from any a

Latest release: **Electrum-3.3.4**

[Release notes](#) [Previous releases](#)

Sources and executables are signed by [ThomasV](#).

Windows builds are reproducible, and signed by several developers. See the list [here](#)

Note: Old versions of Windows might need to install the KB2999226 Windows update.

That should take you to <https://download.electrum.org/>. Go to the folder with the same version number as the one in Tails (for example 3.1.3 with Tails 3.12.1). Download the file for your OS (tar.gz for Linux, -setup.exe for Windows and .dmg for macOS) and make sure to download the corresponding .asc file as well. Put the files in the same folder as ThomasV's signing key:

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory	-	-	-
Electrum-3.1.3.0-release.apk	2018-04-18 17:17	16M	
Electrum-3.1.3.0-release.apk.asc	2018-04-18 17:17	801	
Electrum-3.1.3.tar.gz	2018-04-18 17:11	2.7M	Linux
Electrum-3.1.3.tar.gz.asc	2018-04-18 17:11	801	
Electrum-3.1.3.zip	2018-04-18 17:11	3.0M	
Electrum-3.1.3.zip.asc	2018-04-18 17:11	801	
electrum-3.1.3-portable.exe	2018-04-18 17:11	20M	
electrum-3.1.3-portable.exe.asc	2018-04-18 17:11	801	
electrum-3.1.3-setup.exe	2018-04-18 17:11	14M	WINDOWS
electrum-3.1.3-setup.exe.asc	2018-04-18 17:11	801	
electrum-3.1.3.dmg	2018-04-18 17:12	23M	macOS
electrum-3.1.3.dmg.asc	2018-04-18 17:12	801	
electrum-3.1.3.exe	2018-04-18 17:12	20M	
electrum-3.1.3.exe.asc	2018-04-18 17:12	801	

Note, if you're already using a newer version of Electrum. Downgrading the version might make wallets created with newer versions temporary unusable. Once done with the cold storage, you can upgrade to a newer version of Electrum and everything should work again. Your funds aren't at risk, you can always recover your funds (or import to an older version) with your recovery seed.

Once downloaded we need to verify the download in the same way we verified Tails. Open a terminal (for example Powershell on Windows). Change the current directory to the one where the 3 downloaded files are located:

```
$ cd $HOME/Downloads
```

Import the signing key from ThomasV into your local GPG installation:

```
$ gpg --import ThomasV.asc
```

Now use the .asc to check that the Electrum installer was signed with the signing key we imported:

```
$ gpg --verify electrum-3.1.3-setup.exe.asc electrum-3.1.3-setup.exe (make sure to change the file name if using a different version).
```

The verification can take a while.

Expected output should be something like:

```
gpg: assuming signed data in 'electrum-3.1.3-setup.exe'  
gpg: Signature made 04/18/18 17:10:44 W. Europe Daylight Time  
gpg:          using RSA key 2BD5824B7F9470E6  
gpg: Good signature from "ThomasV <thomasv1@gmx.de>" [unknown]  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg:          There is no indication that the signature belongs to the  
owner.  
Primary key fingerprint: 6694 D8DE 7BE8 EE56 31BE D950 2BD5 824B 7F94 70E6
```

The signing was made the same day as the release was uploaded (should be around the same time). It's a Good signature. A search online on 6694 D8DE 7BE8 EE56 31BE D950 2BD5 824B 7F94 70E6 seems to confirm that this key belongs to Thomas V. We are good to go. Install Electrum and follow the instructions on screen.

[P] Setup Electrum to run over Tor

Electrum use servers that's run by volunteers to validate and broadcast transactions. Anyone can start a server and if you don't specify a server, you'll be connected to one randomly. This is terrible for privacy (and has been used for phishing attacks). If you don't use Tor or a VPN you're essentially giving a random server your IP-address and all the bitcoin addresses you're asking for.

There's several companies that specialises in chain analysis to deanonymize addresses and we can assume that they're running several Electrum Servers. If you bought your bitcoin on an exchange that use KYC (know your customer), you can assume that your private data will be leaked sooner or later. If you don't do something about it, risk is that almost every transaction you do can be linked to you. A first good step is to use Tor. It uses "onion routing" to hide your true IP-address. You can also use a VPN, both solutions will hide your real IP-address from random servers. With a VPN, you'll trust the provider (all traffic goes through them) which is probably safe as long as Bitcoin is legal in your jurisdiction (but you never know).

Using Electrum with Tor should be a fairly straightforward process. If you don't have Tor, go to <https://www.torproject.org/projects/torbrowser.html> and download the latest version of Tor Browser for your OS. You should now know how to verify digital signatures. So, download

the signature (.asc) for the file you download as well. You can import the Tor signing key with the command:

```
gpg --keyserver pool.sks-keyservers.net --recv-keys 0x4E2C6E8793298290
```

Then use:

```
gpg --verify {your_file.asc}
```

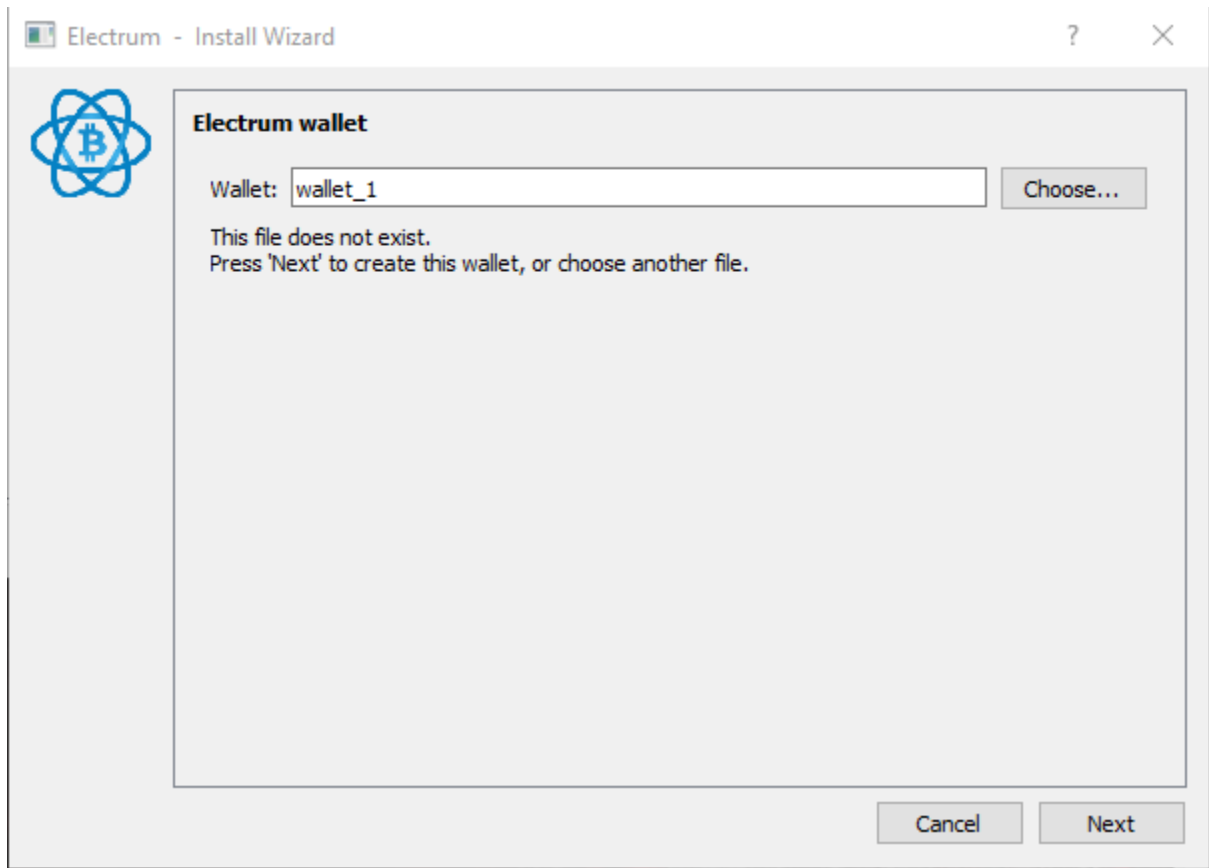
Make sure the output is similar to this:

```
gpg: assuming signed data in 'torbrowser-install-win64-8.0.6_en-US.exe'  
gpg: Signature made 02/12/19 14:26:17 W. Europe Standard Time  
gpg:          using RSA key EB774491D9FF06E2  
gpg: Good signature from "Tor Browser Developers (signing key)  
<torbrowser@torproject.org>" [unknown]  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg:          There is no indication that the signature belongs to the  
owner.  
Primary key fingerprint: EF6E 286D DA85 EA2A 4BA7  DE68 4E2C 6E87 9329 8290  
Subkey fingerprint: 1107 75B5 D101 FB36 BC6C  911B EB77 4491 D9FF 06E2
```

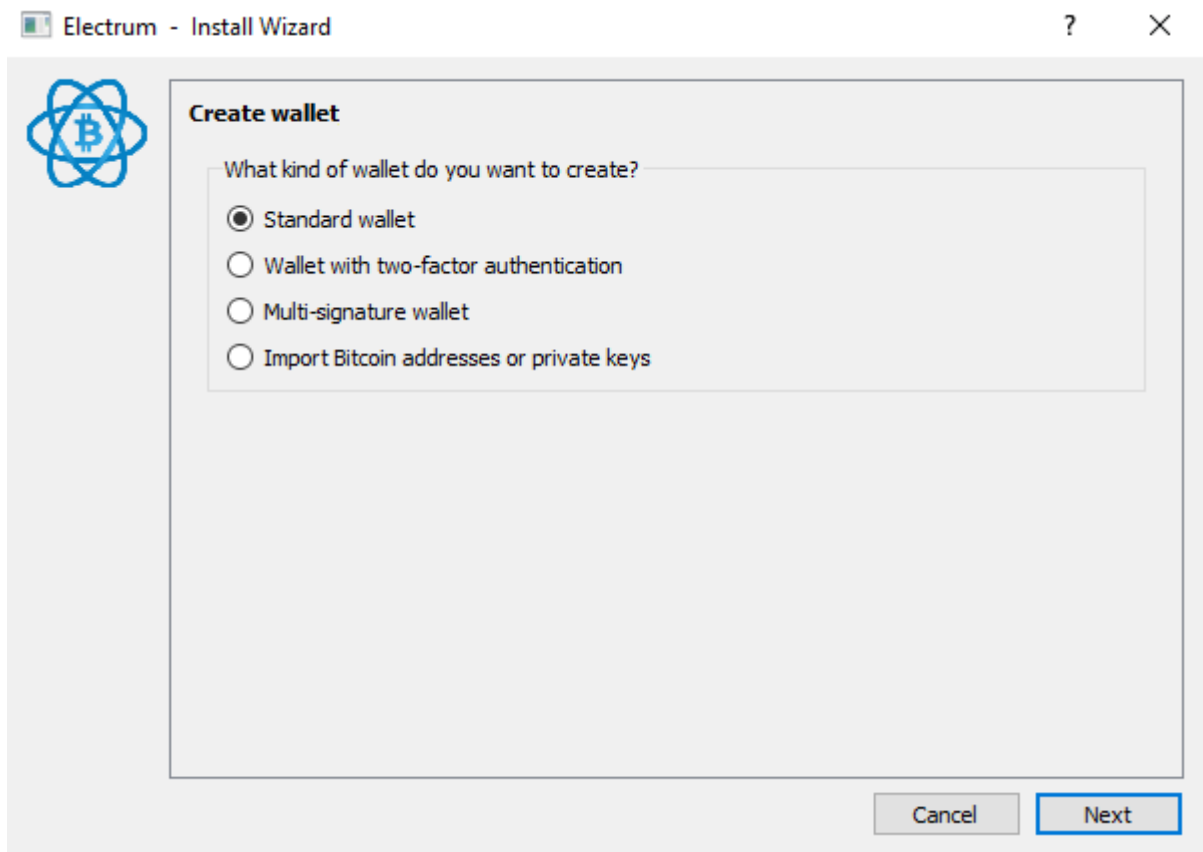
When Tor is installed, start Tor. That could be done in two ways. You can launch the browser (it's much easier to use then a few years ago) or only start Tor (like tor.exe on WIndows). Only Tor should be located at .\Tor Browser\Browser\TorBrowser\Tor. You won't notice anything if you launch Tor without the browser (nothing visible will start).

Start Electrum. If this is the first-time starting Electrum, you'll have to create a wallet first (skip these steps and open a wallet if you already have one). The wallet you create can be a "dummy" wallet that you delete after the process. We only want to access the settings.

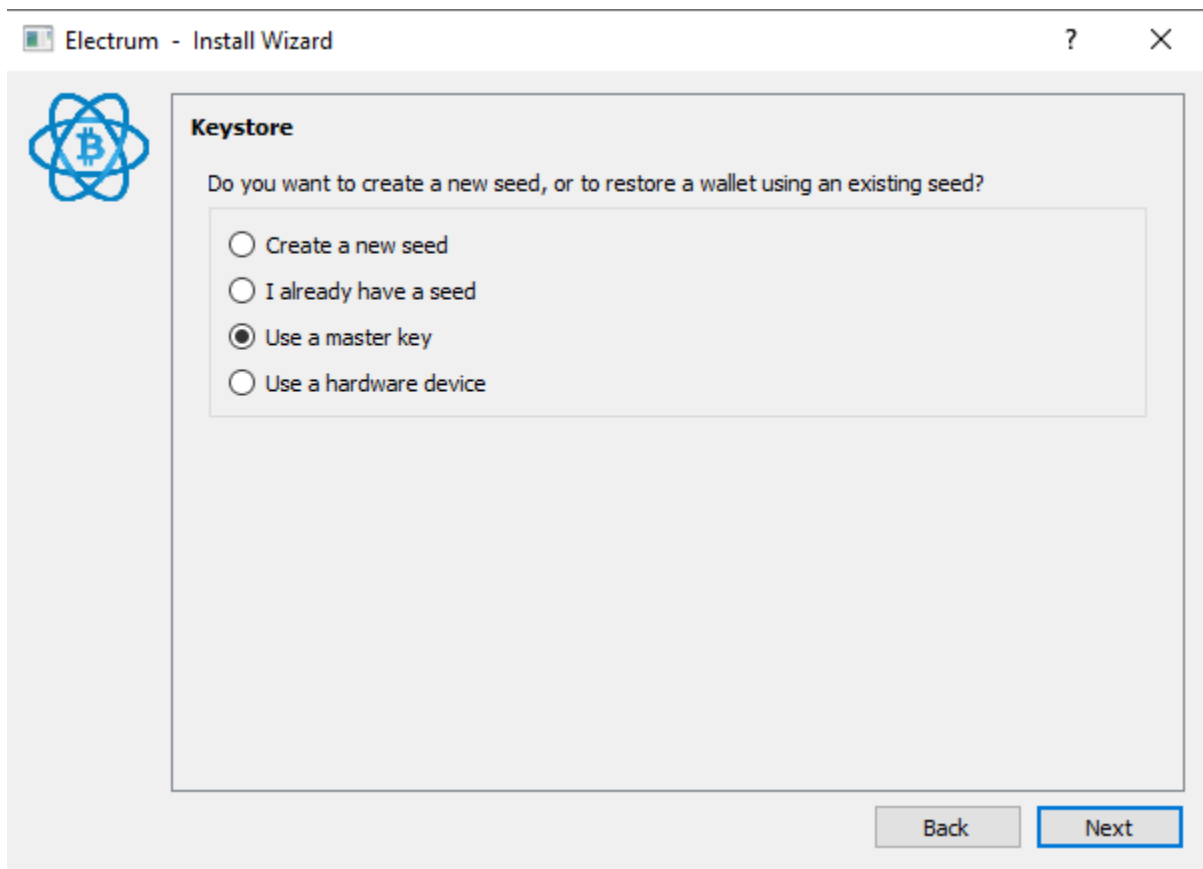
Pick a name for the wallet, click Next:



Let Standard Wallet be selected, click Next:

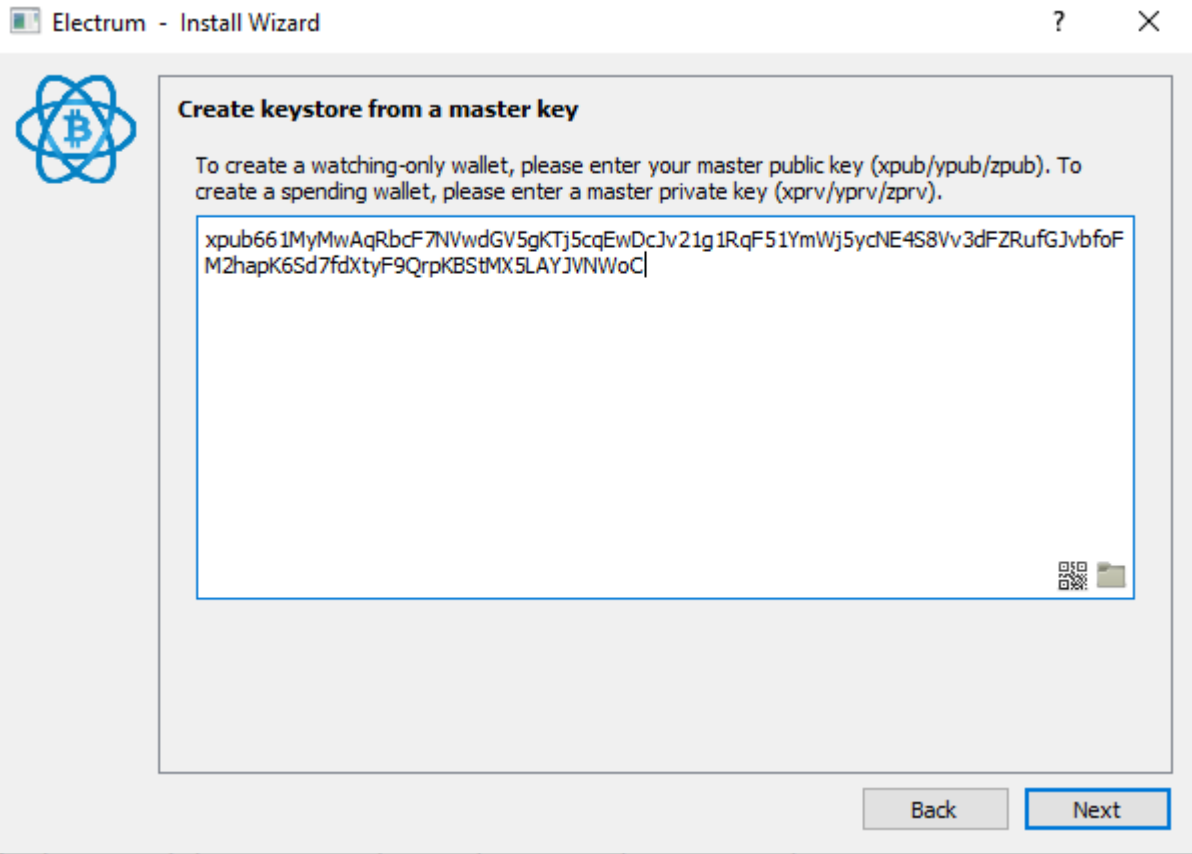


Change to "Use a master key" and click Next:

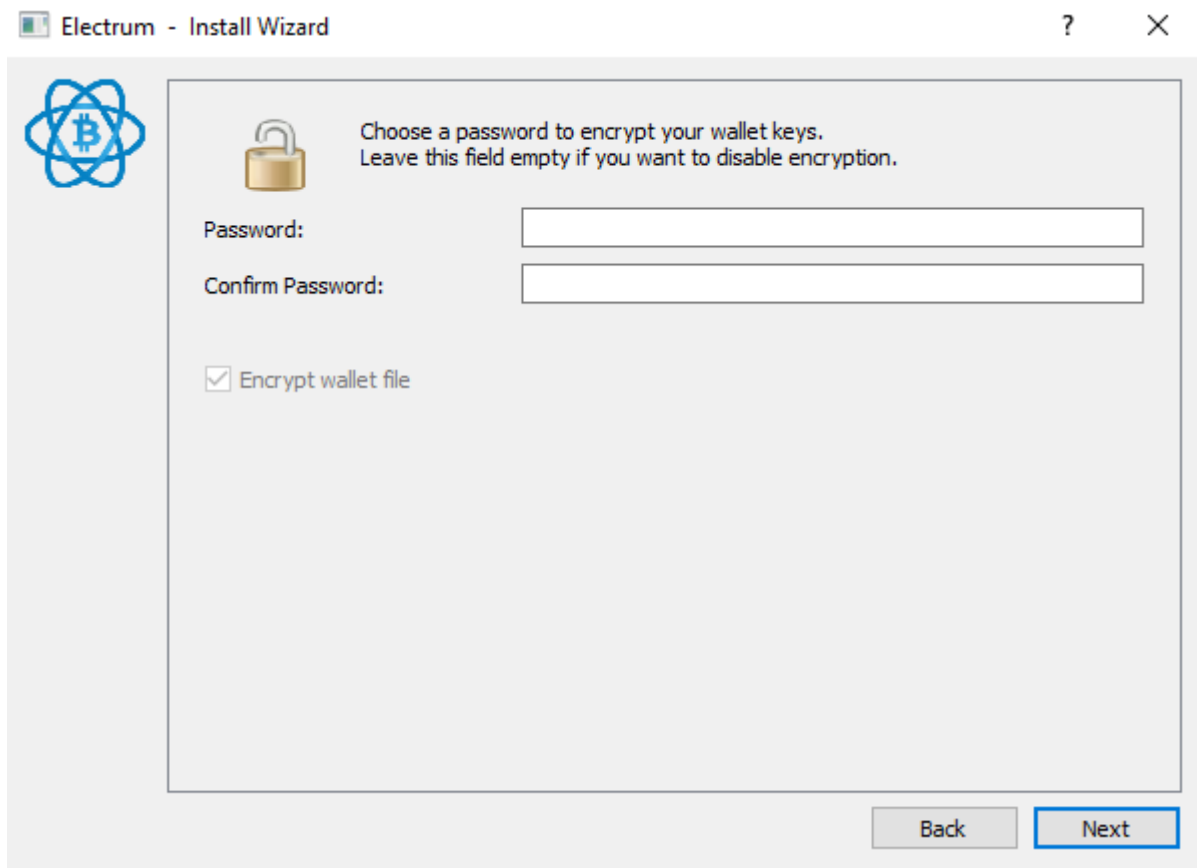


Since we only want a dummy wallet to access the settings, paste this key in the field and click Next:

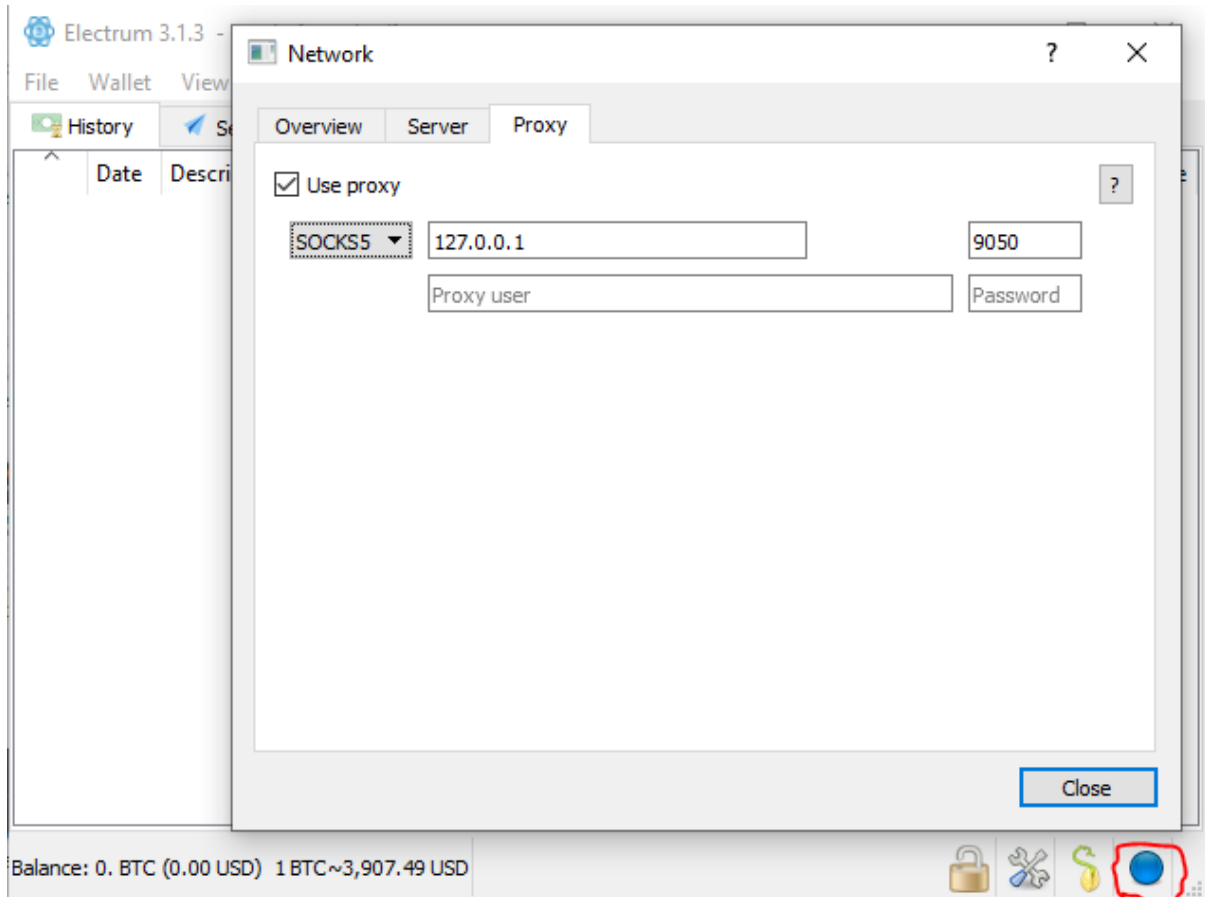
```
xpub661MyMwAqRbcF7NVwdGV5gKTj5cqEwDcJv21g1RqF51YmWj5ycNE4S8Vv3dFZRufGJvbfoF  
M2hapK6Sd7fdXtyF9QrpKBStMX5LAYJVNWoC
```

We don't need a password for this wallet, leave the field blank and click Next:



Electrum should now start. Go to `Tools>Network`. Change the tab to `Proxy`. Select "Use Proxy" and make sure `SOCKS5` is `127.0.0.1` and that the port is `9050`:

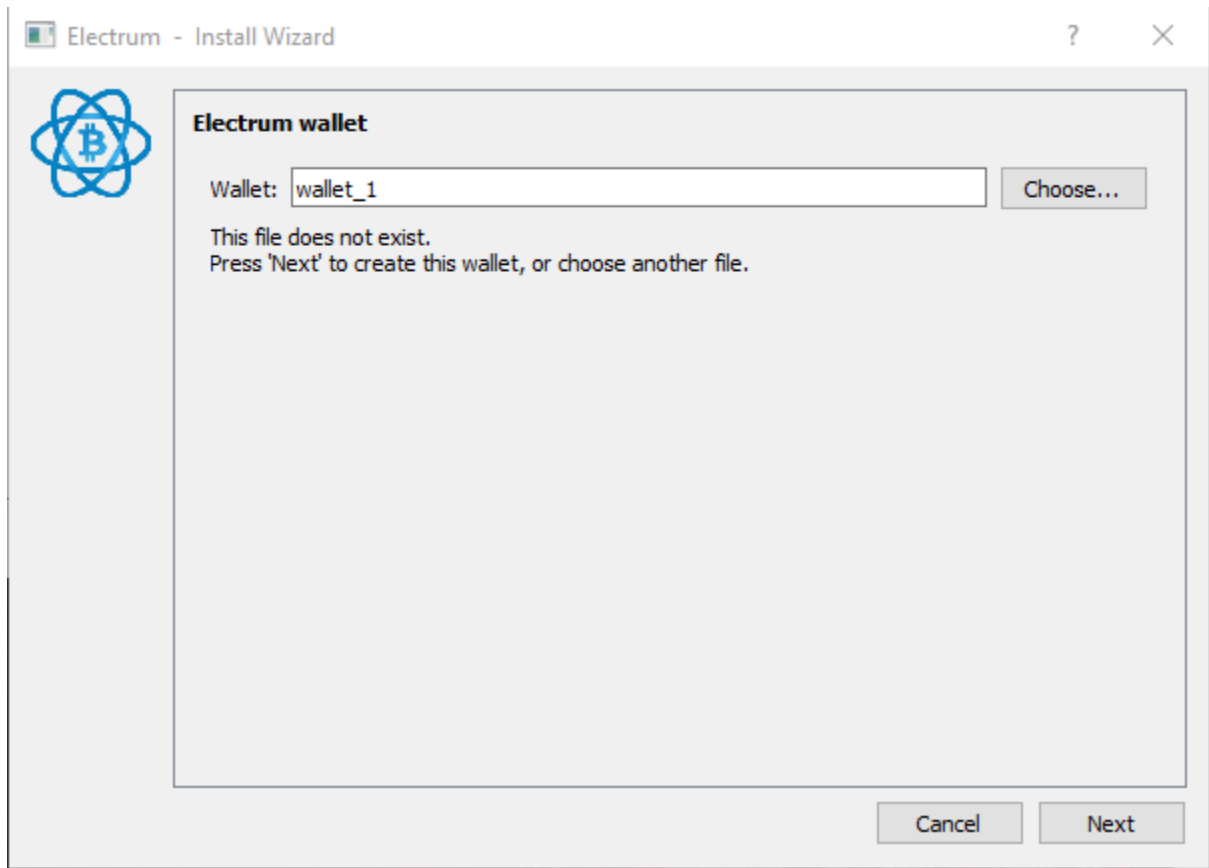


Close the network dialog. The circle in the bottom right corner should now be blue and not green. You have now configured Electrum to run over Tor. Electrum won't connect to anyone unless Tor is running (even if you restart it).

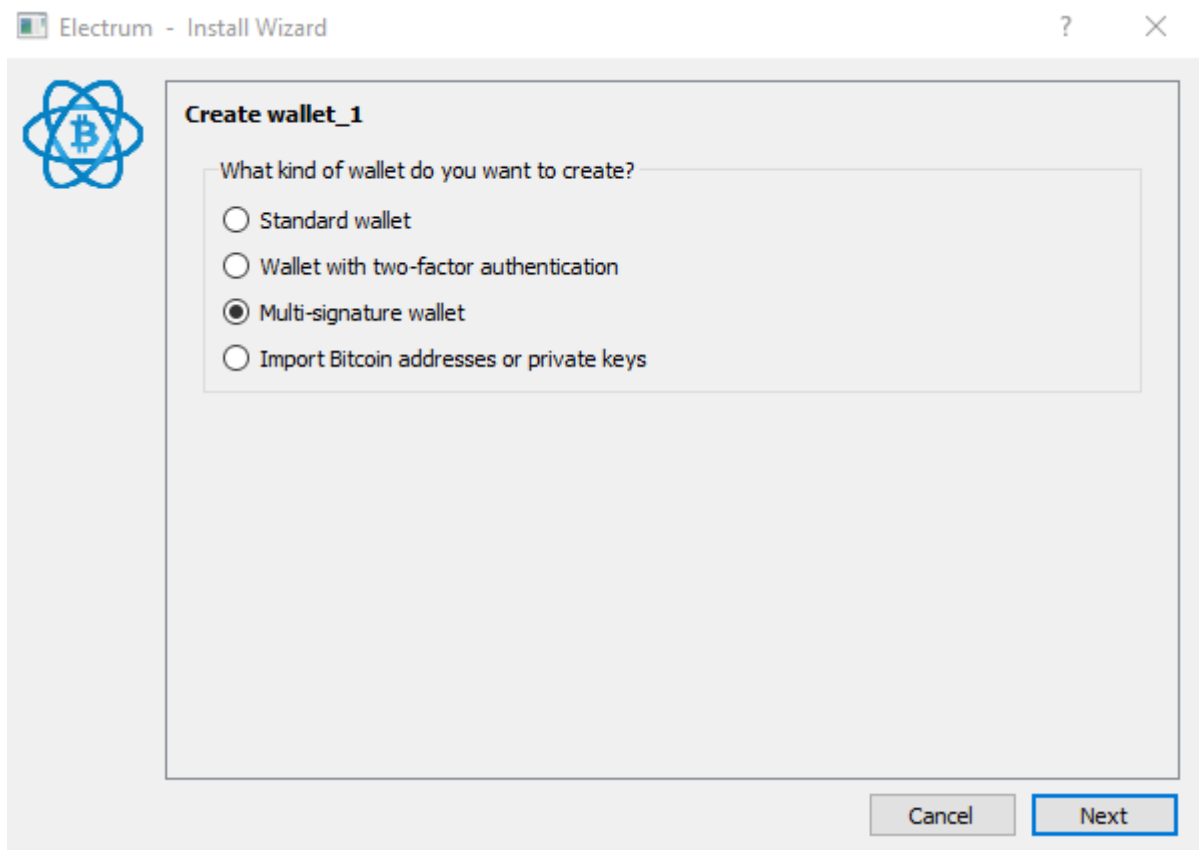
Create the multi-sig wallet

We are now going to create our multi-sig wallet. If you already have a Electrum Wallet open, go to `File>New/Restore` (or use `Ctrl+N`). Otherwise start Electrum, the install wizard should be launched automatically.

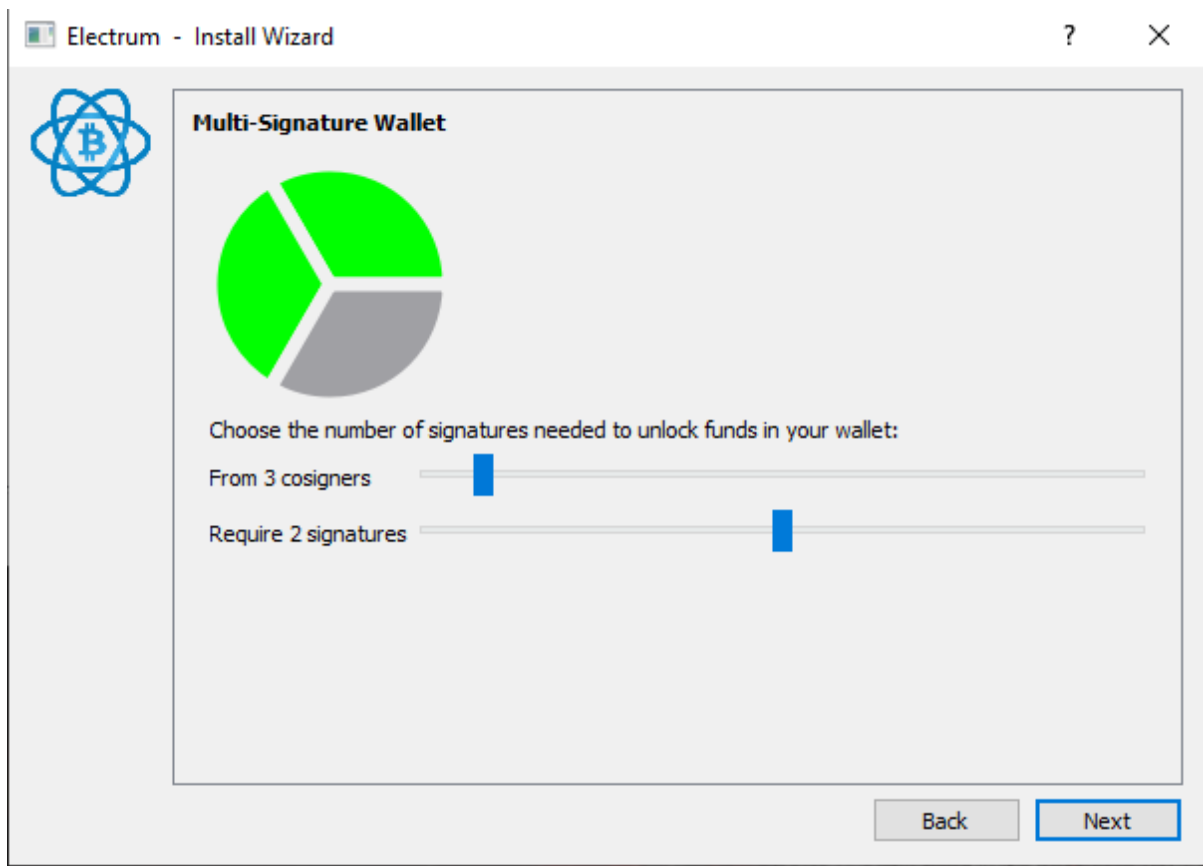
Pick a name for your multi-sig wallet and click Next:



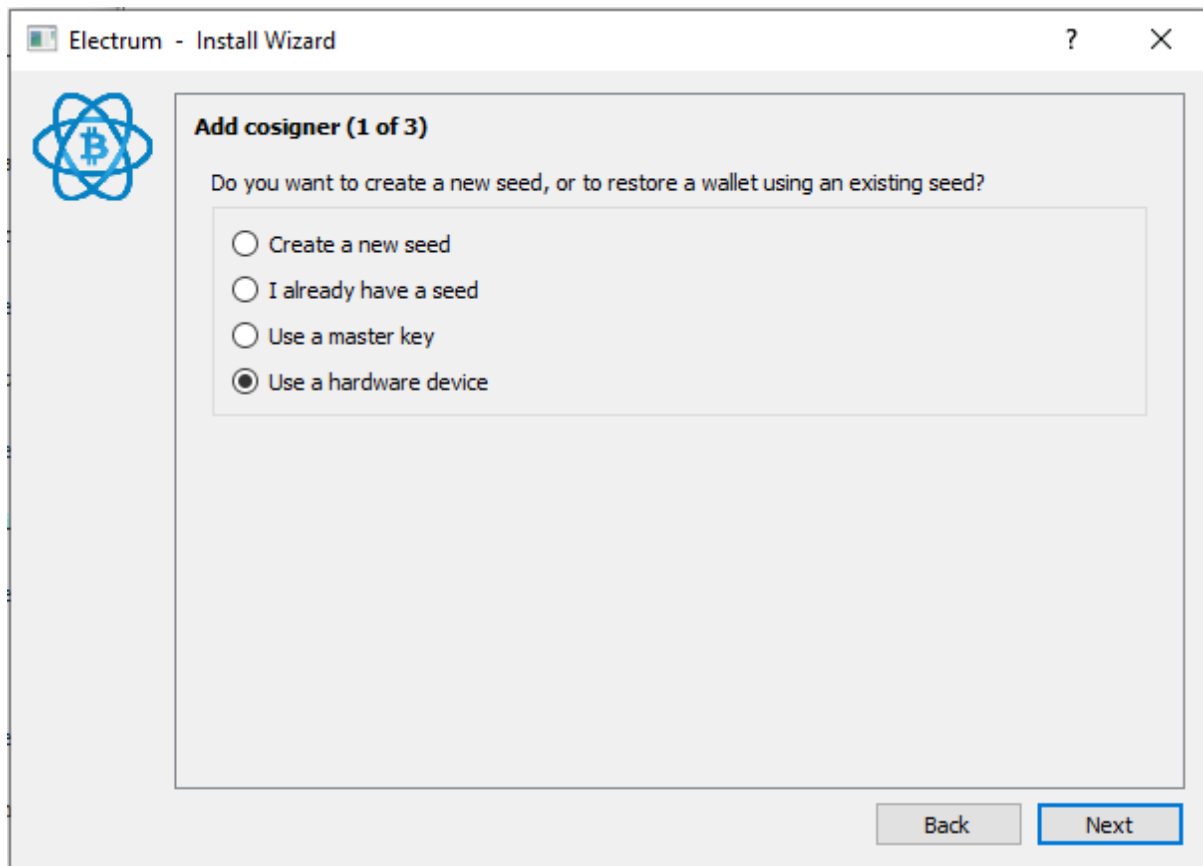
Select "Multi-signature wallet" and click Next:



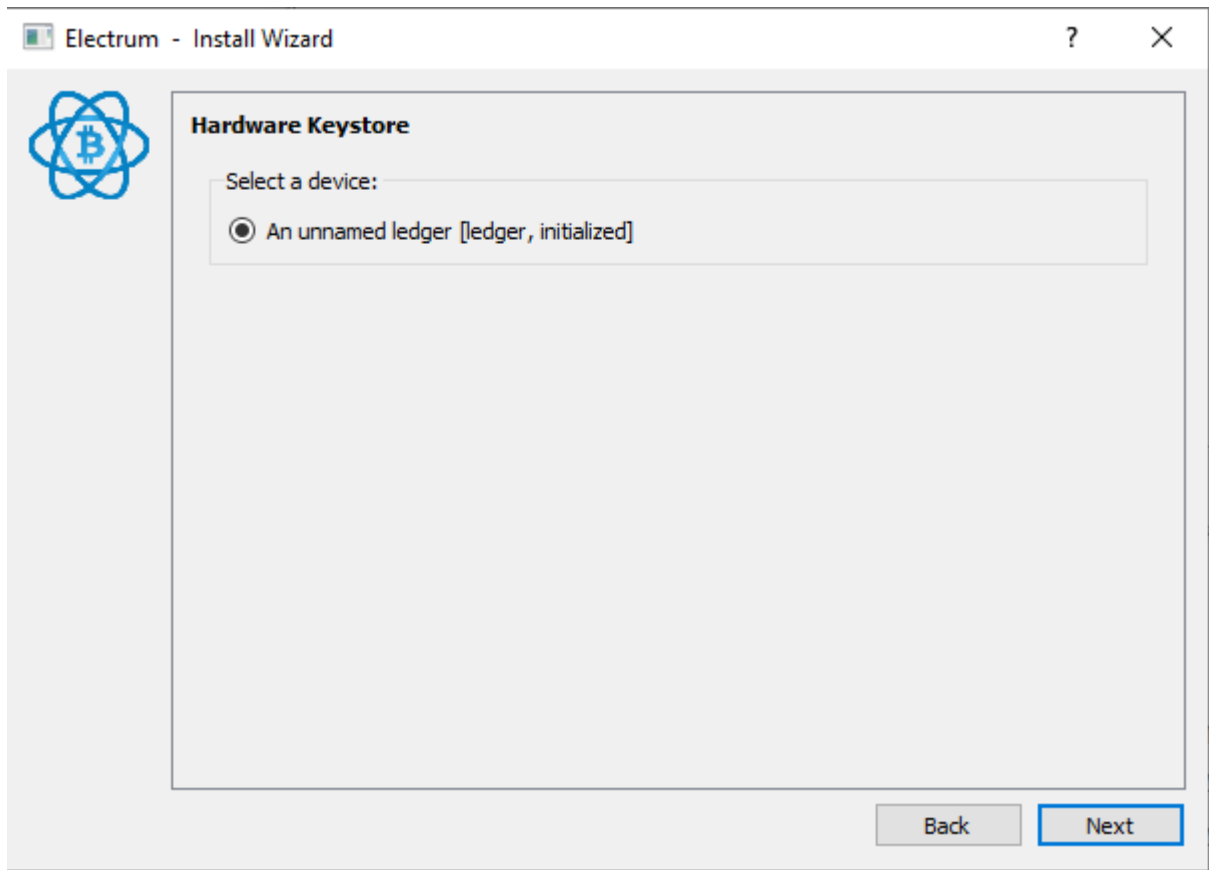
Change the first slider to 3 cosigners (with 2 signatures required) and click Next:



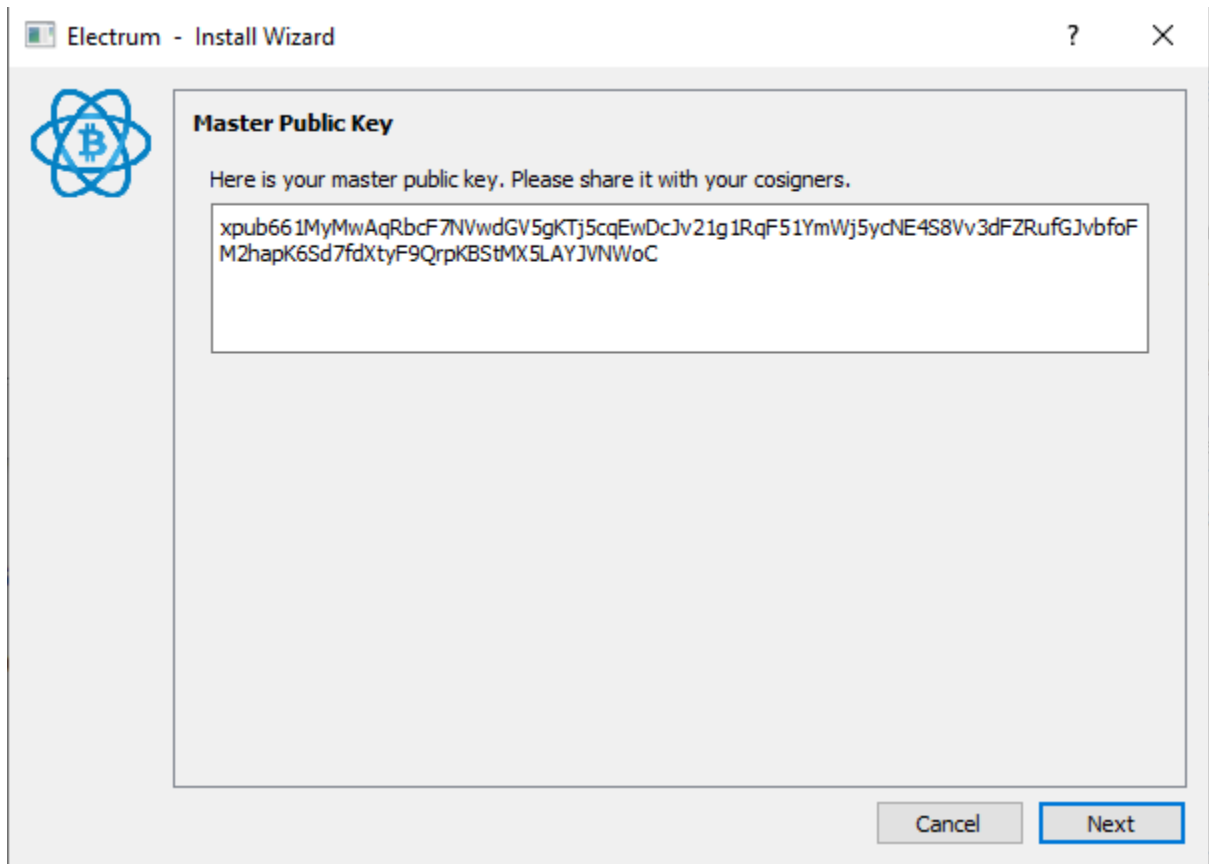
We are now going to construct our multi-sig. Start with `Hardware Wallet A`. If you use a wallet, like Ledger, where the password is written on the device. Make sure the password is active before moving on. Select `Use a hardware device` and click Next:



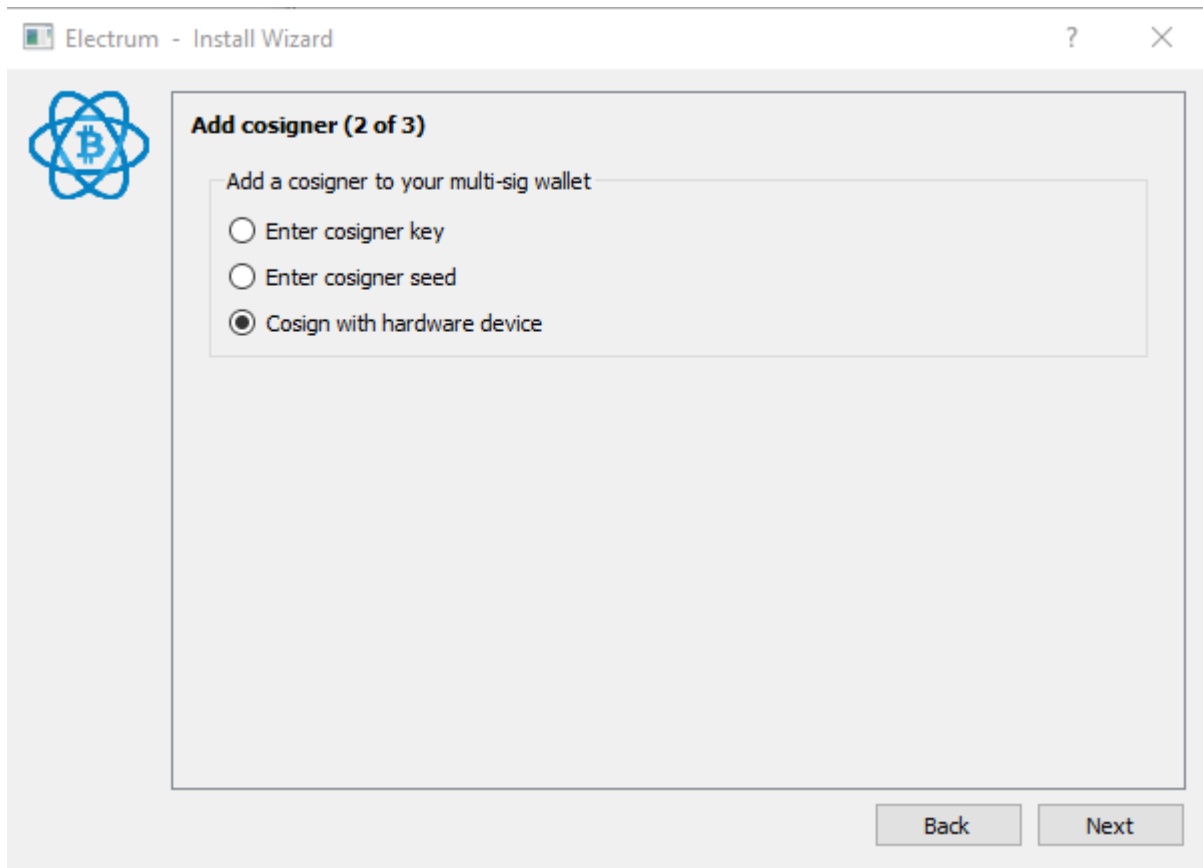
Electrum should detect your hardware wallet and show its name. If detected, click Next (otherwise, rescan by clicking Next). If using a hardware wallet where the pin and password is entered on the computer (like Trezor), enter the pin and password A after you've clicked Next:



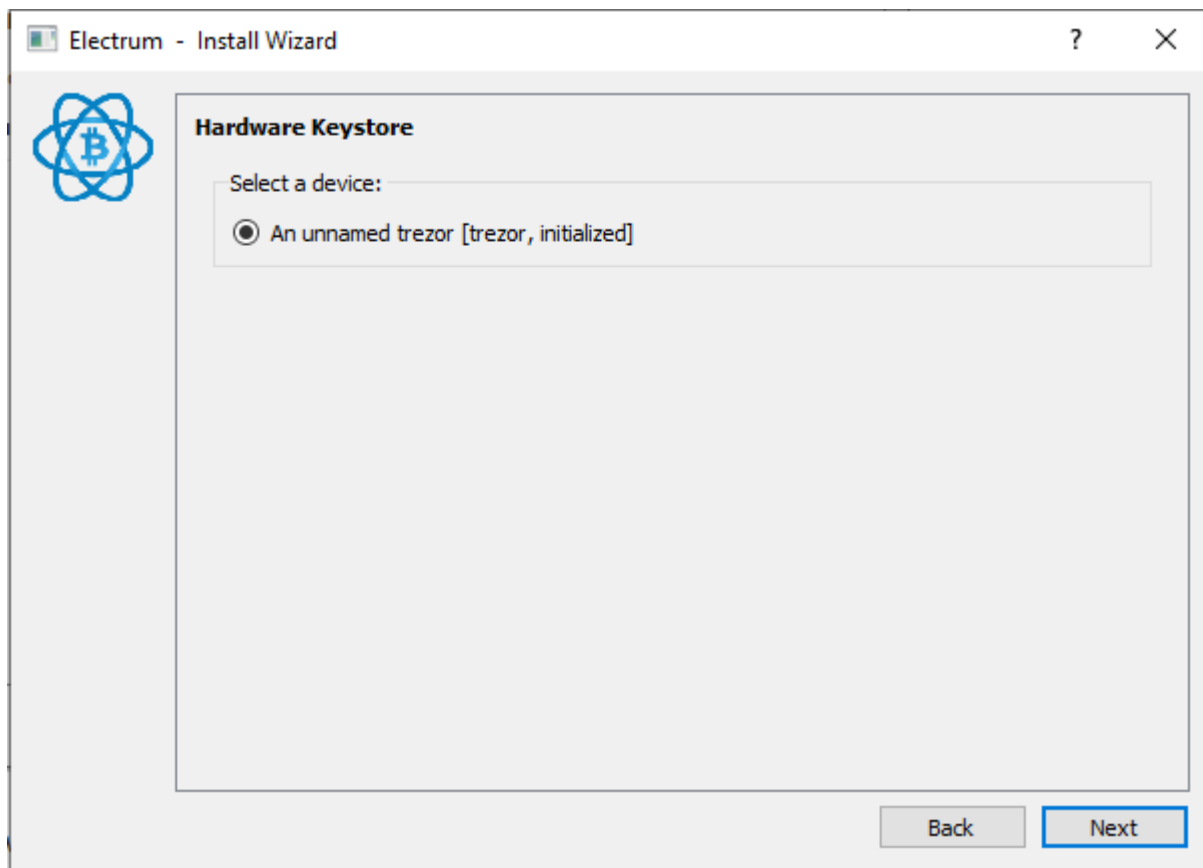
The next window shows your Master Public Key, we will access this later in Electrum and can skip it now, click Next:



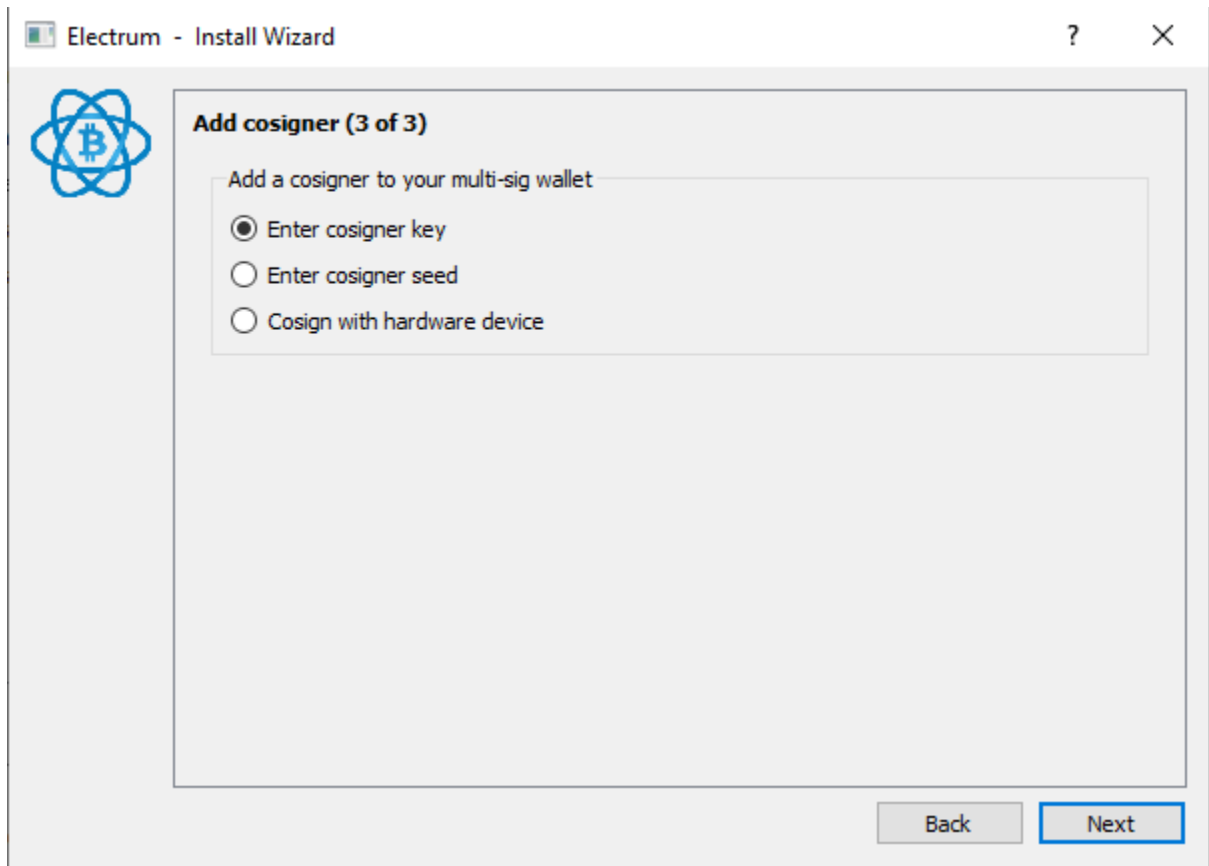
We are now going to add key 2. Remove Hardware Wallet A and insert Hardware Wallet B and repeat the process you used for key 1. If it's a wallet with a physical pin like Ledger, enter the pin (and make sure that it uses password B). Select Cosign with hardware device and click Next.



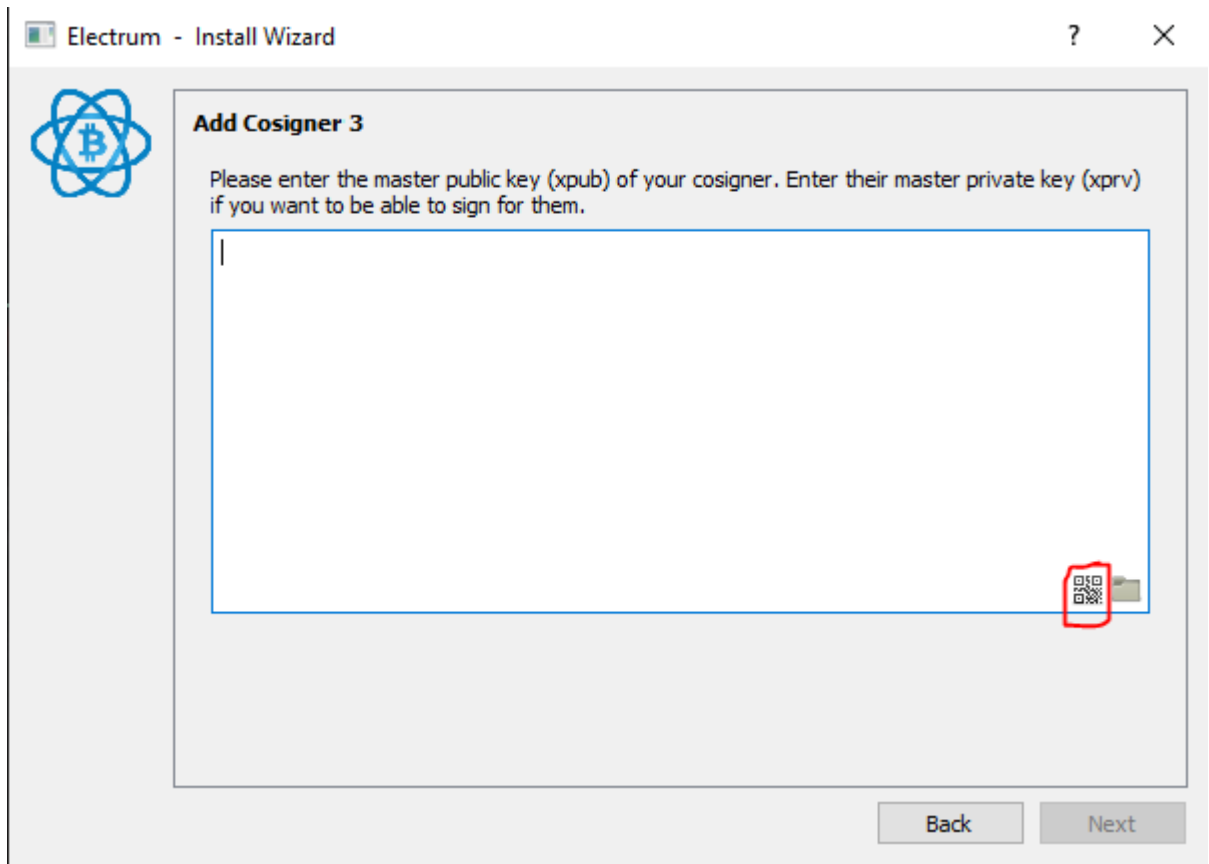
Electrum should detect your hardware wallet and show its name. If detected, click Next (otherwise, rescan by clicking Next). If using a hardware wallet where the pin and password is entered on the computer (like Trezor), enter the pin and password B after you've clicked Next:



The next window should be where you add cosigner 3 of 3. We are now going to use the key we created with Tails. Select “Enter cosigner key” and click next:



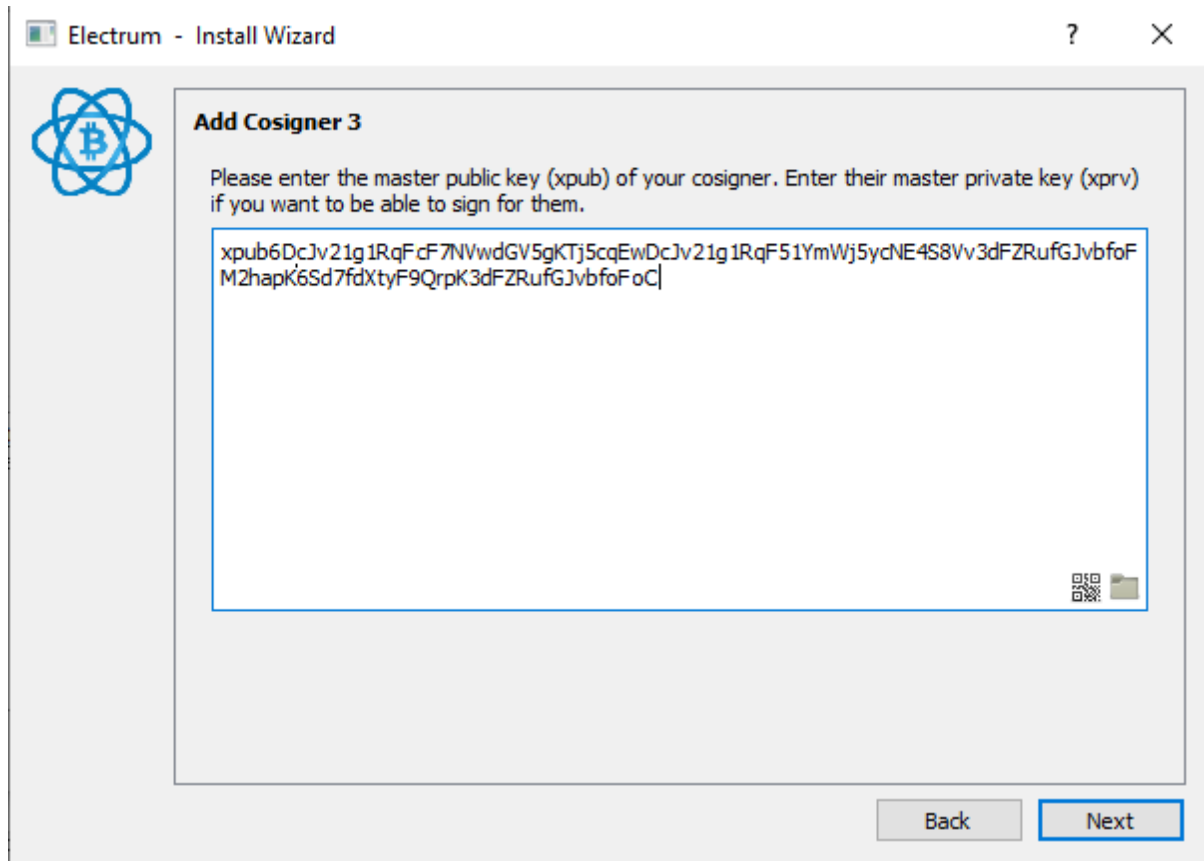
Click the QR-code in the bottom right corner and scan the QR-code on your phone:



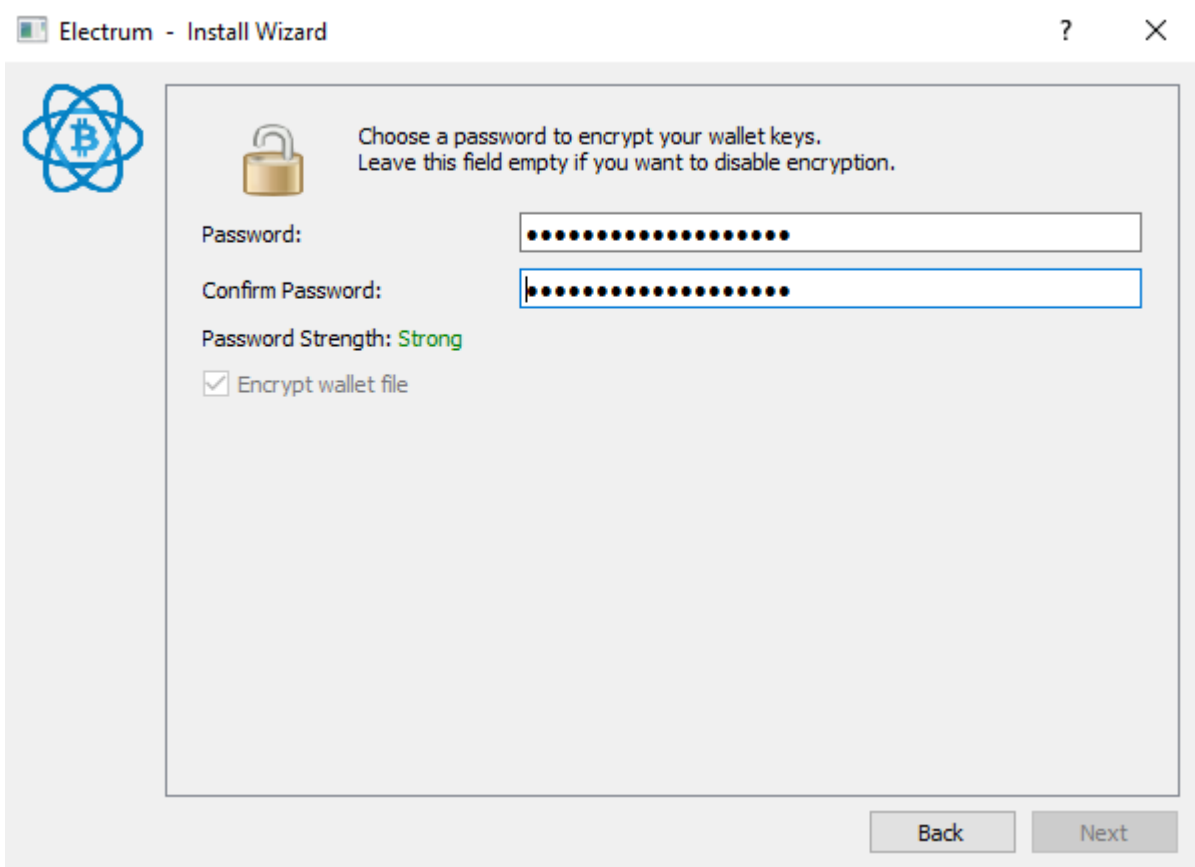
That should bring up your xpub . . . key

If you have an error with scanning the QR-code, check the Troubleshooting guide [Scan QR-code with Electrum](#)

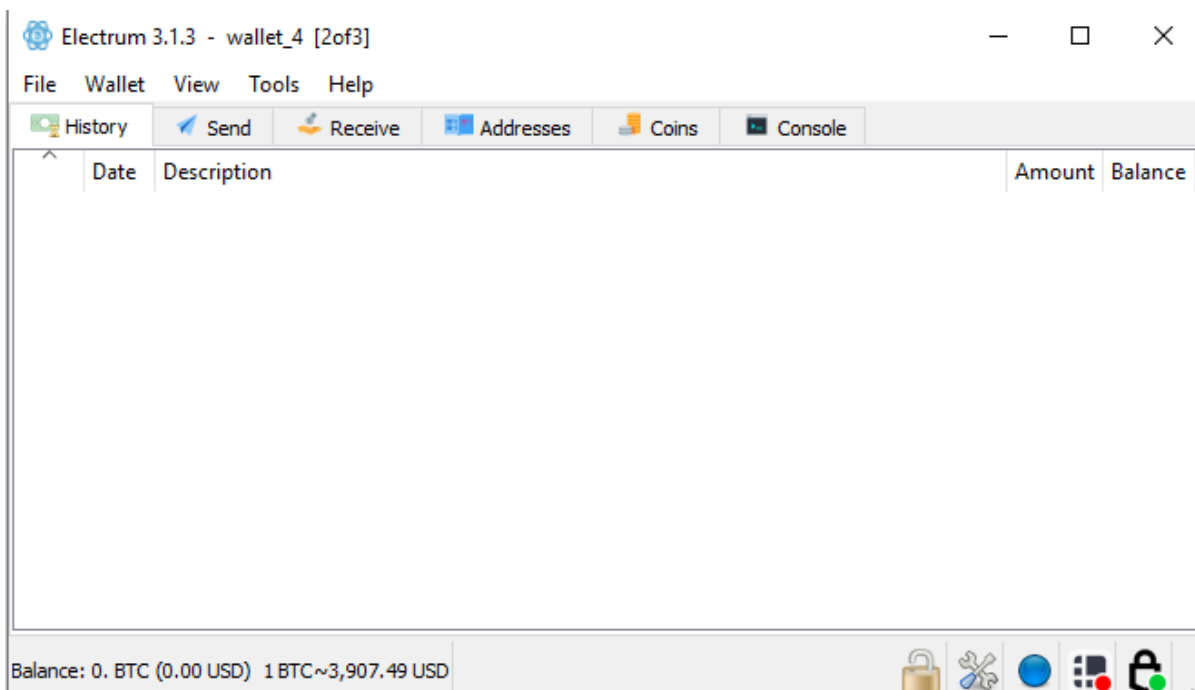
When the QR-code is in the field, click Next:



Your wallet is now created and you should be asked for a password to encrypt your wallet. This is for your master public keys that's stored on your computer and it's a good idea to protect that with a password. Pick a strong password, preferably generated by a password-manager. You will need this password to open the wallet in Electrum (but not for restoring your funds, you can always restore your funds with your seeds + the seed passwords). You can store this password in LastPass, KeepassX or similar managers or use a password you'll remember. Enter the password, confirm it and click Next:

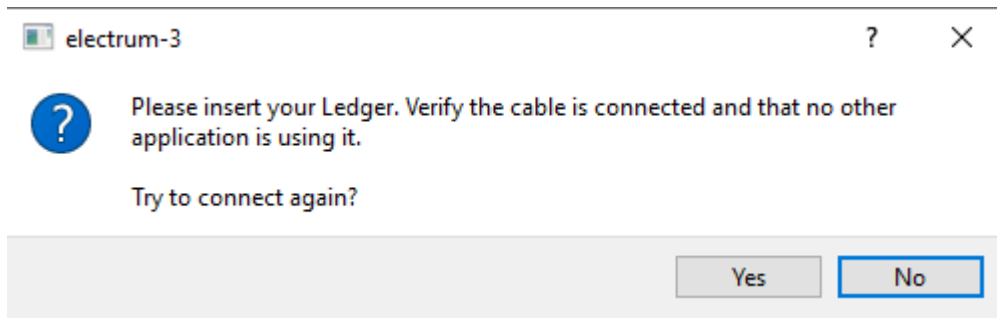


You should now see the following:



Congratulations, your wallet is created!

If you disconnected one, or both, of your Hardware Wallets. You'll probably get a message like this:



You can simply click `NO`

If you use Electrum over Tor (the circle in the bottom right corner should be blue), you have a pretty solid setup and can use this as it is. Electrum won't connect unless you use Tor and you will hide your real IP-address from any servers. But you still rely on third parties for validation and broadcasting. It's fine for our test deposit and we can improve this later.

We are going to deposit a small amount of bitcoin to one of our addresses to make sure everything works. In the beginning of 2019, transactions are practically free. So, a few \$ worth of bitcoin is plenty to try it out. The amount should cover 3 transaction fees.

Note: Always think twice before depositing funds to your cold storage. I highly recommend to properly mix any coins deposited to cold storage (especially if they are from an exchange with KYC). This is only a test deposit to a one-time address, so mixing is not that important here. You can deposit from any normal wallet and can think about mixing later.

Deposit Program

This can be used every time you want to deposit funds to your cold storage.

Open your multi-sig wallet in Electrum and enter the password that unlocks the wallet. Go to Receive and copy the "Receiving address".

Note: If you used Zbar to scan your QR-code, there might be a problem with copy and pasting. A restart of Electrum should fix that.

Go to another wallet (like Wasabi Wallet or another Electrum wallet) and send your bitcoin to the receiving address. You should see the unconfirmed balance almost immediately.

Optional: If you want to know that your backup works, you might want to go through the trouble of doing a full restore of your wallet. Do this by opening a wallet in Electrum, go to File>Open (or Ctrl+O). Delete your wallet file and close the wallet you deleted if it's open. Go back to the first step of [create the multi sig wallet](#) and set it up exactly the same way as you did before (same password etc). But this time use another source of information. If you used your secure note for the passwords the first time, use the information in the information packages this time. When the wallet is recovered, your funds should be there and you can be sure that your process works.

Withdrawal program

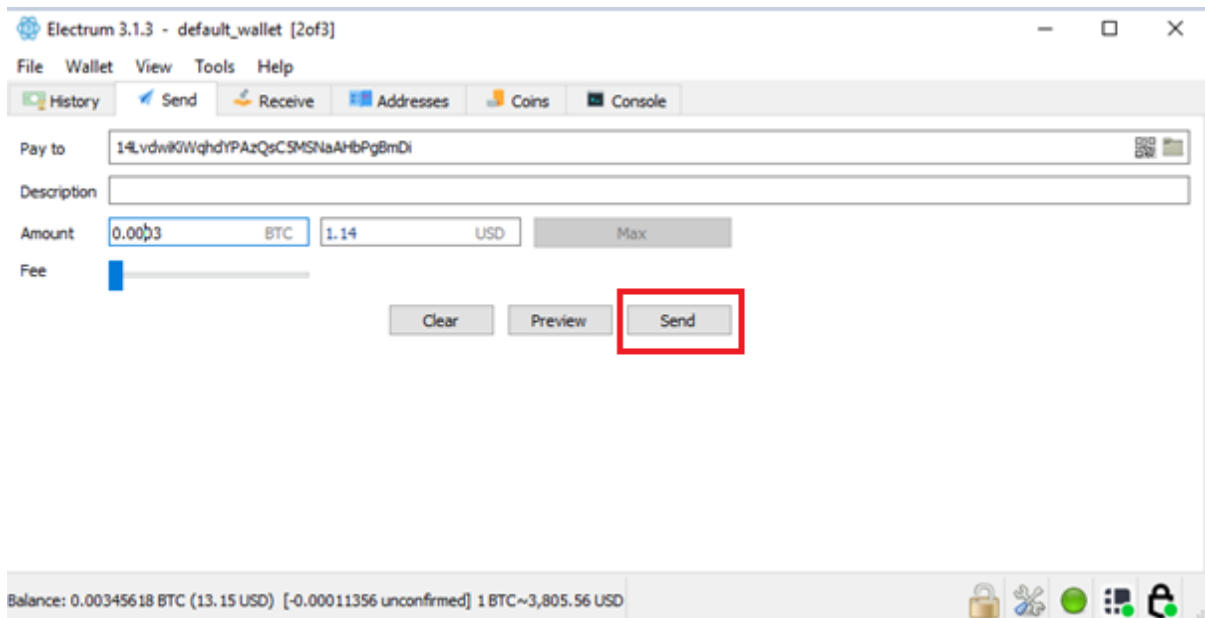
This can be used when you want to withdraw funds from your cold storage.

[P] If you don't know about coin control and have 100% control of your unspent outputs. Never use funds in your cold storage for day to day spending and don't transfer funds from your cold storage directly to someone that knows your real name or address (like a friend or an exchange). If you connect your real name and/or physical address with addresses in your cold storage, that could be used to "cluster" addresses together and reveal what addresses you control. A good rule of thumb is to use Wasabi Wallet, or a similar service, to mix all funds going into cold storage and all funds going out of cold storage. *Note:* Some exchanges might treat bitcoins involved in coin join transactions as suspicious and deny your deposit. You have to decide yourself between privacy and ease of selling.

If this is the first time withdrawing from the wallet, we're going to do two test withdrawals (and we don't have to think about mixing). The first one with your two hardware wallets. The second one is with your third backup key and one hardware wallet. That procedure is only necessary if you lose one private key or its password and the corresponding hardware wallet. Normally, use the method described in "Withdrawal method 1".

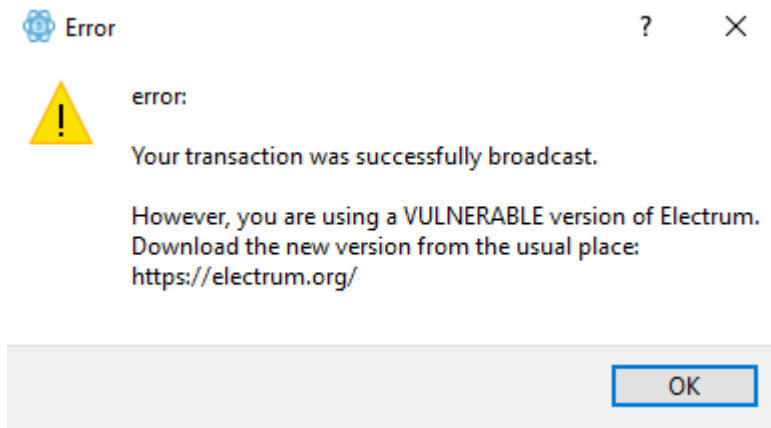
[Withdrawal method 1](#)

Open your wallet in Electrum. Connect your 2 hardware wallets. You can connect both at the same time or one at a time. If you use them one at a time, start with Hardware Wallet A. On the Send tab, create an ordinary transaction with half of your test amount to an address you control (like an address in Wasabi Wallet or in another Electrum Wallet). Once the transaction is constructed click `Send`:



Sign and confirm the transaction with one Hardware Wallet at the time, start with Hardware Wallet A. If your hardware wallet has a screen, control the information (like address and amount) on the screen.

Your transaction should be broadcasted and you'll probably get a message like this:

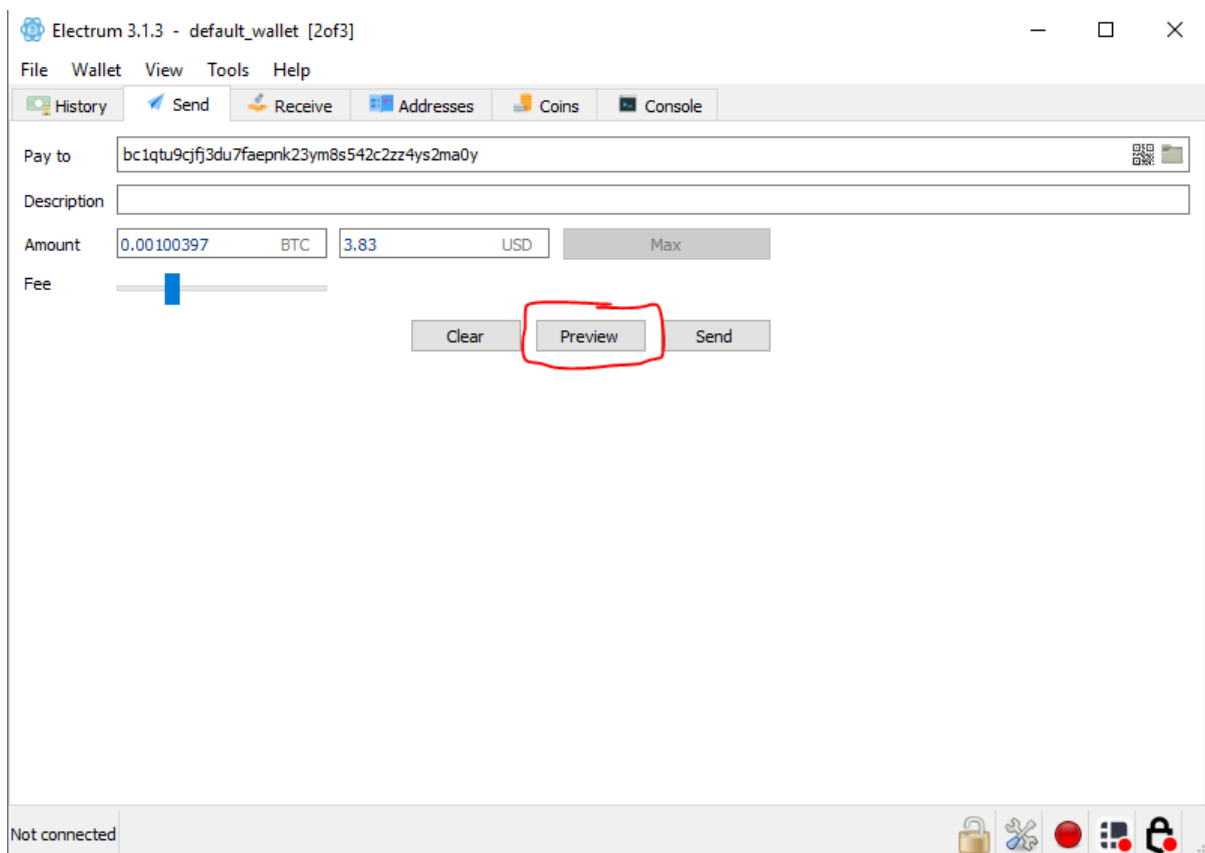


Click OK, we can update once we are done with Tails

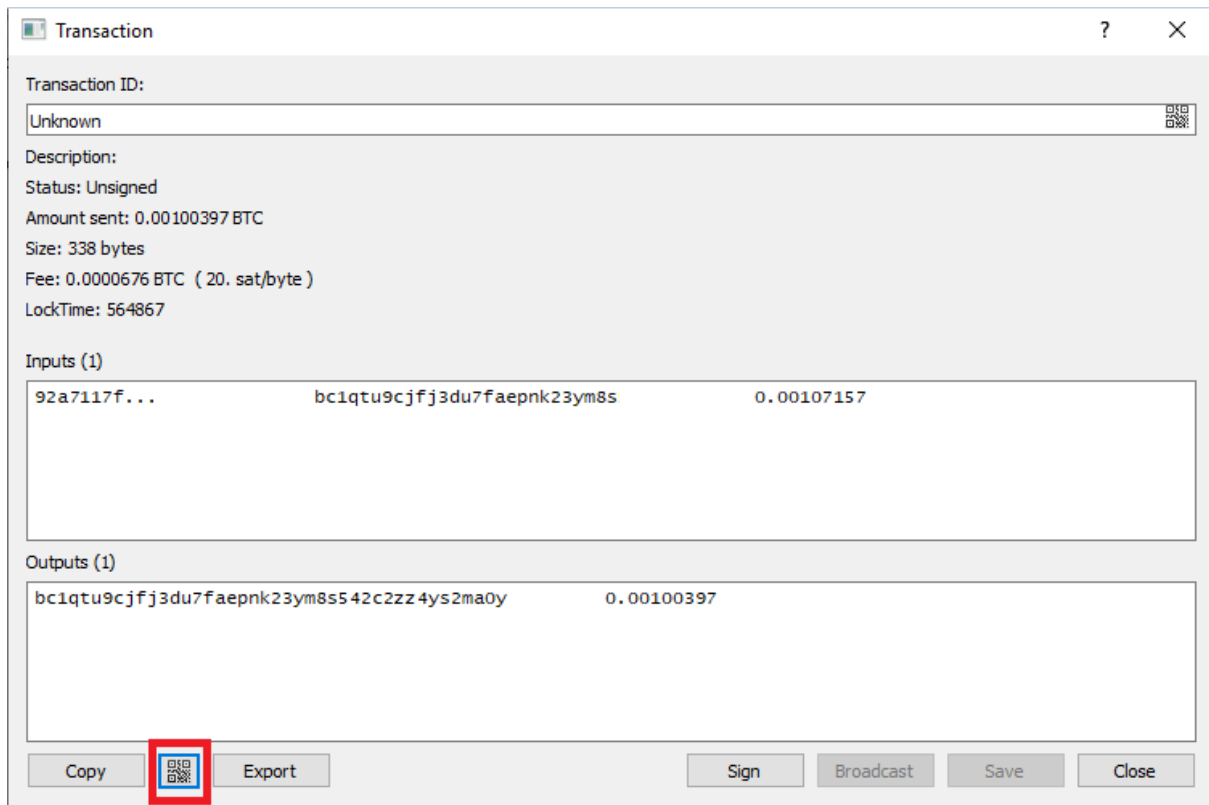
Withdrawal method 2

This is the backup method and should only be needed in case you lose some of your keys. But it's a good check to make sure our third key works. If you are doing this at a later date, remember that you'll probably need the same version of Electrum running on your computer as in Tails.

In Electrum, go to send, enter an address to another wallet you control. Select the rest of the test amount you have left in the wallet, pick a fee and select “preview”:



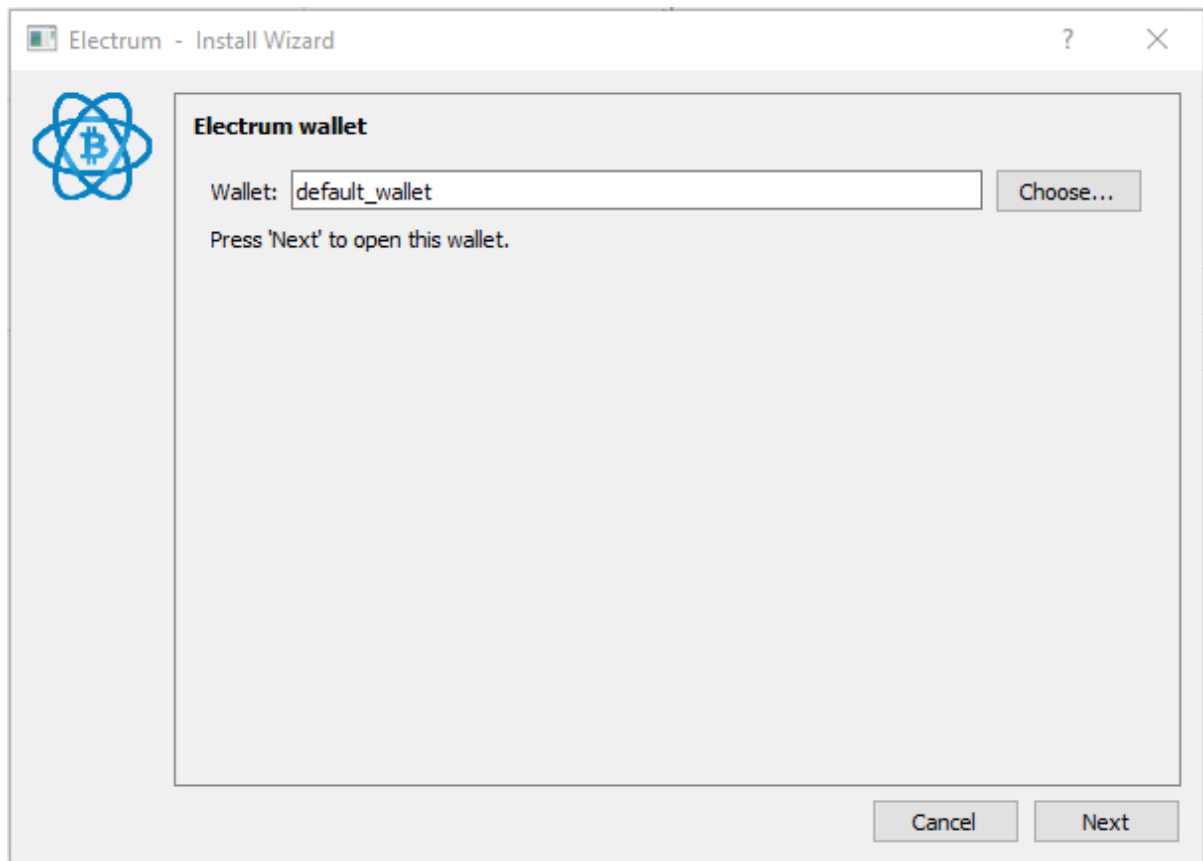
In the preview window, select the QR-code in the bottom left corner:



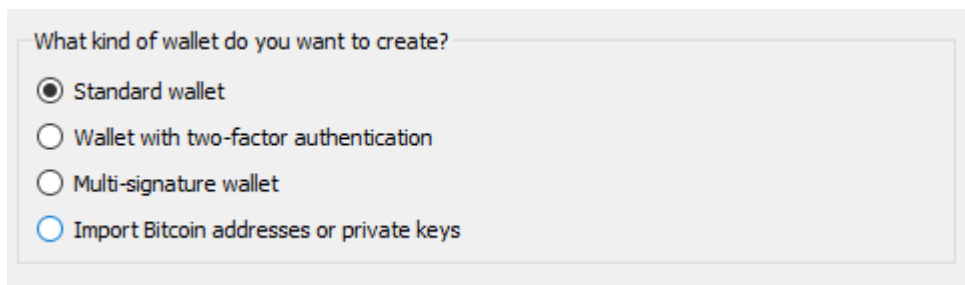
That should bring up the QR-code. Take a photo of the QR-code with your phone. You can close the Transaction dialog.

We are now going back to Tails. So, either go to your second computer or restart your main computer on Tails. We are going to handle a private key, make sure to follow the same procedure that you used when generating the keys (that no one can see what you do). In Tails, launch Electrum like before. If you already have Electrum running create a new wallet by going to `File>New/Restore` (we are going to create the exact same wallet as before as a check that our seed works).

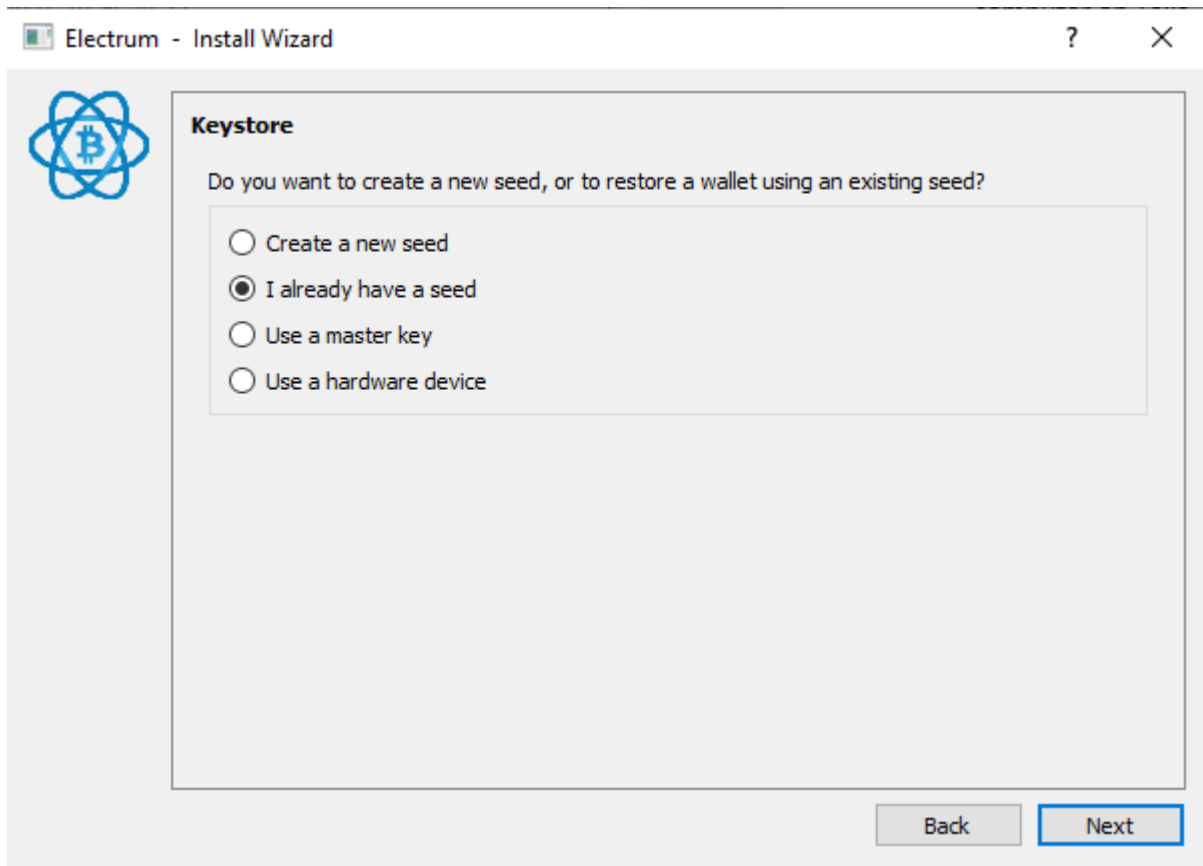
Click next at the first window:



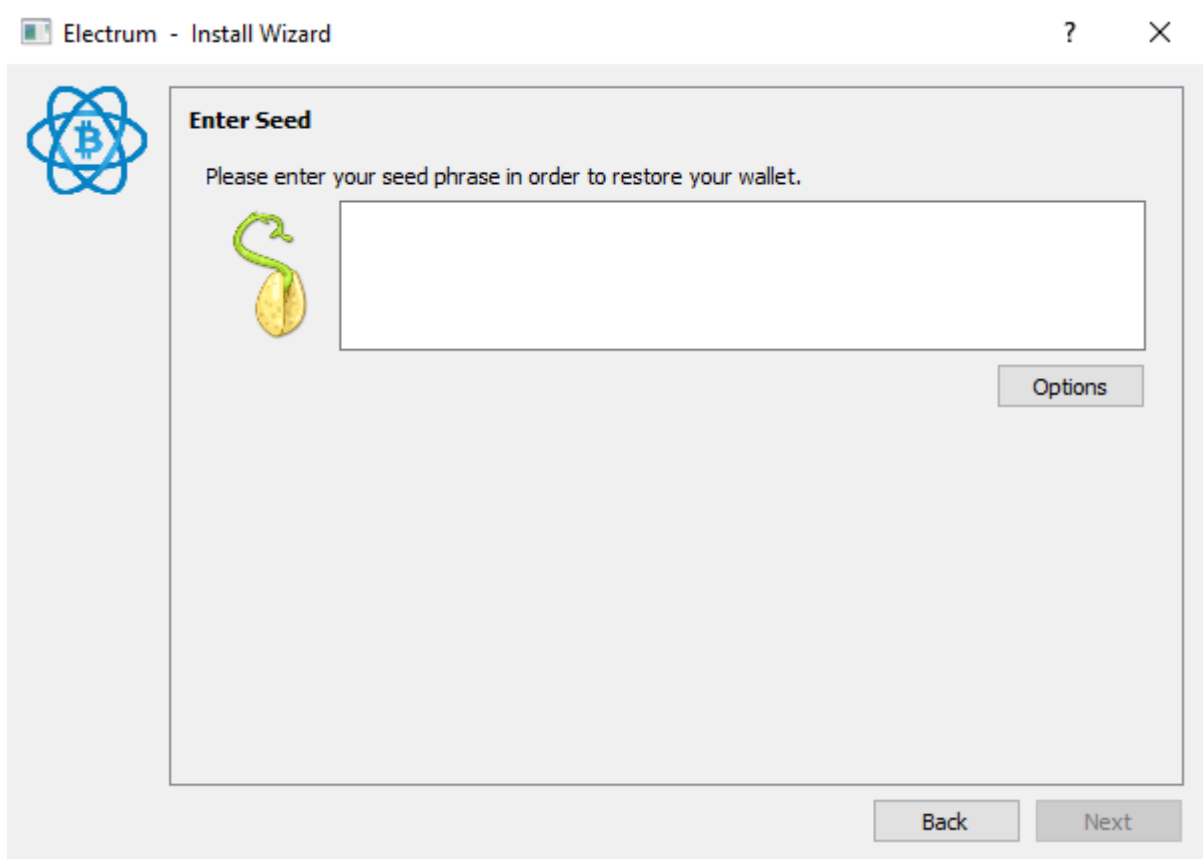
We only need the signature, so keep “standard wallet” selected and click Next:



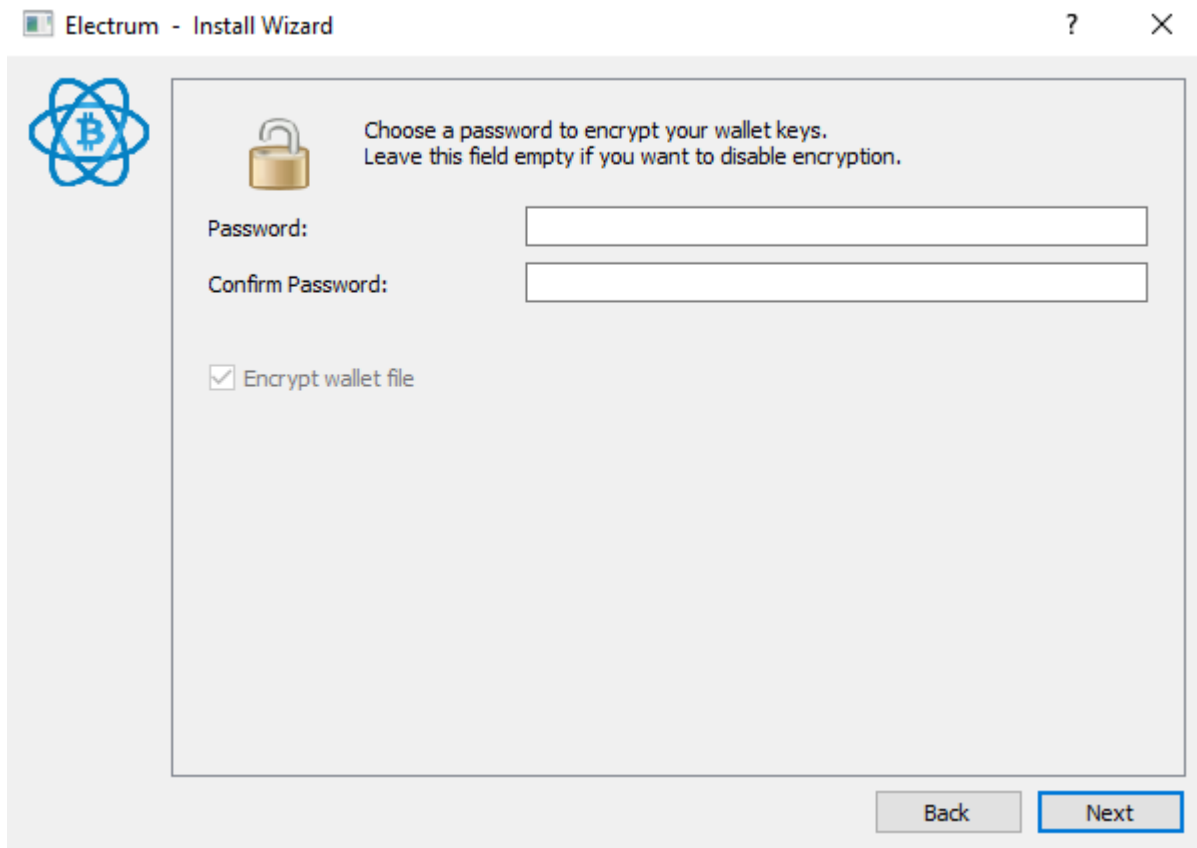
Select “I already have a seed” and click Next:



Enter your 12-word seed on the next screen (again, this is valuable information):



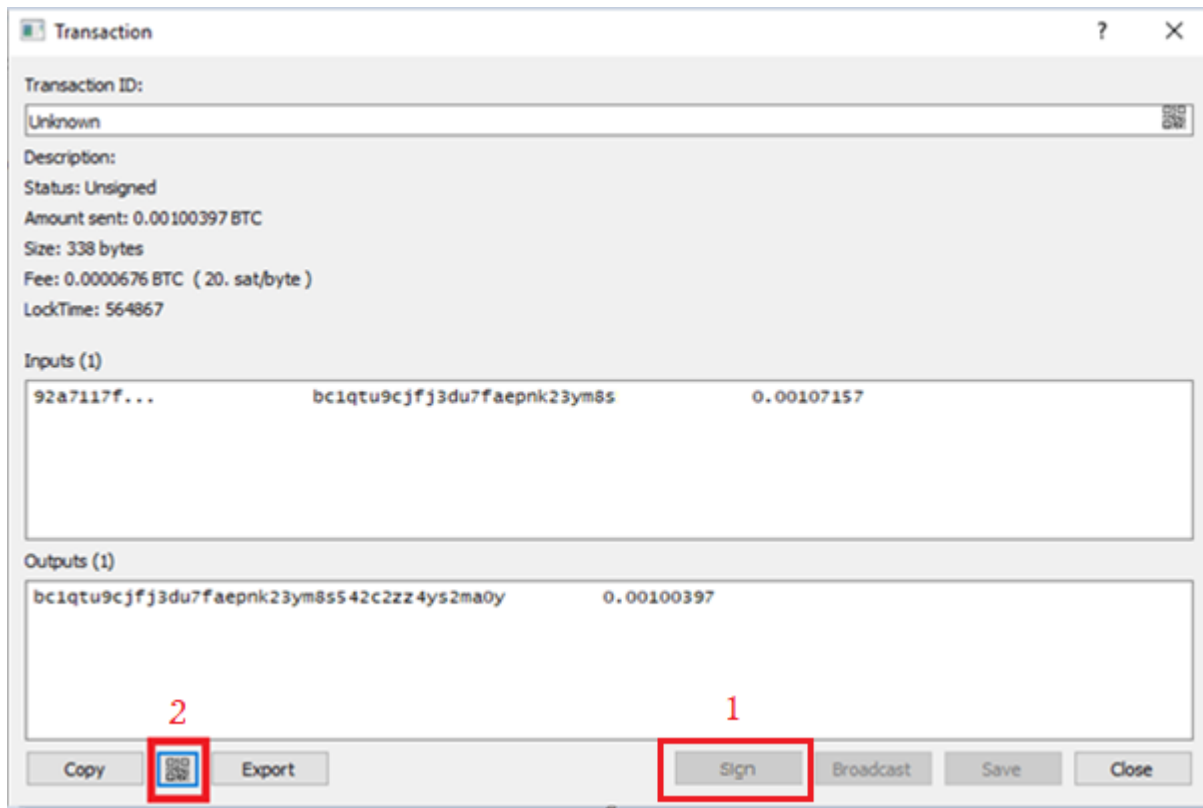
Leave the password field empty (the wallet file will be deleted once finished):



The wallet should now load. Go to `Tools>Load Transaction>From QR code` Scan the QR code on your phone.

This should bring up the same Transaction Window that you had on your main OS. If you get an error, check that the Electrum version is the same in Tails as on your main computer.

Start by first clicking “Sign”. That should Sign the transaction. Then click the QR-code:



Take a photo of the QR-code with your phone.

You can now close Electrum and Tails (we won't be using it anymore). Remove the Tails USB from the computer to delete all information.

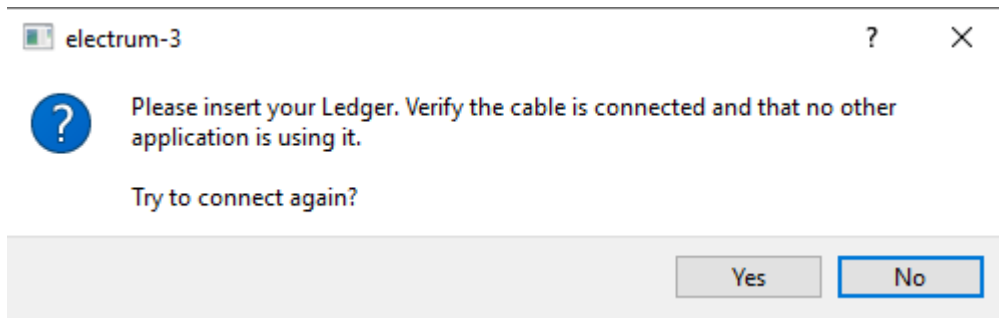
Go back to your main computer and open your multi-sig wallet in Electrum. Go to `Tools>Load Transaction>From QR code` and scan the last QR-code. You might have to start Zbar and select a camera for the camera to work (and it might be slow to start, so try 2-3 times).

Note, if you are using another way to scan the QR-codes. Scan the QR-code outside of Electrum. In Electrum, go to `Tools>Load Transaction` and paste the text.

That should bring up the transaction window.

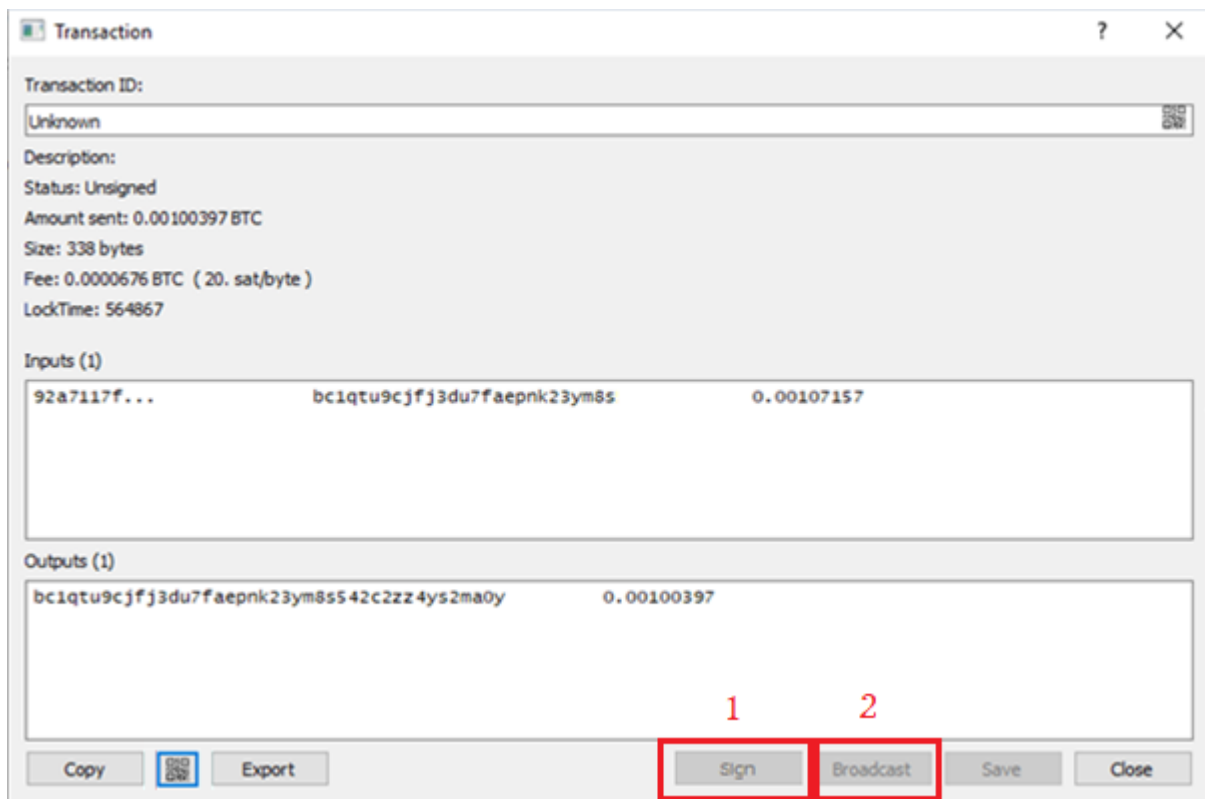
Connect one of your hardware wallets and click sign. Follow the instructions on your Hardware Wallet, control the address and the amount, and sign the transaction.

If you are using Hardware Wallet B to sign, you'll probably get a message like this:

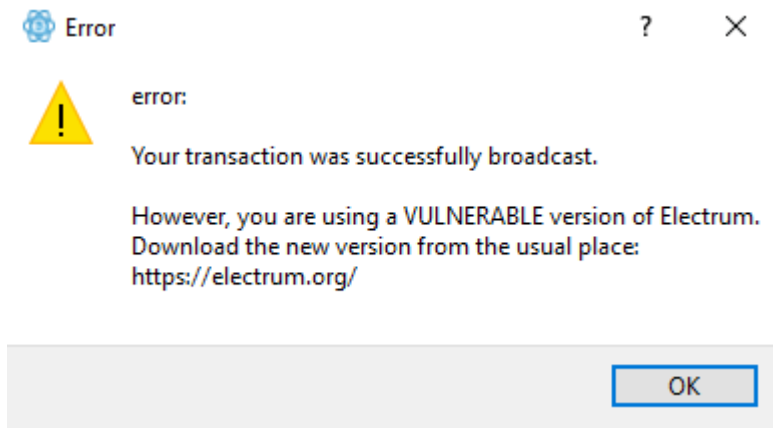


Select **No** and wait for Electrum to detect Hardware Wallet B. Once detected, click **Sign** and follow the instructions on your Hardware Wallet (this process can be rather slow). Once signed, wait for Electrum to calculate everything.

Once calculated, click **Broadcast**:



Your transaction should be broadcasted and you'll probably get a message like this:



Since we are done with Tails, feel free to update to the latest version whenever you like.

Update information packages

Before moving on, we need to fill in the last information in the information packages.

With your wallet open in Electrum, go to `Wallet>Information`.

In your `secure` note add all master public keys, like this:

```
MPK1: xpub668...
MPK2: xpub78e...
MPK3: xpub87t...
```

In `information package A` add the master public key from cosigner 3 This is like backup for the backup but should still be treated with care. It's easy to miss one character if typing it by hand. Consider printing it on a printer, if you own one, and attaching it to the note. Otherwise double check what you print:

```
MPK3: xpub668...
```

In `information package C` add the master public key from cosigner 2:

```
MPK2: xpub78e...
```

That's it! Finish by deleting all the pictures of QR-codes on your phone or camera.

Store your keys

You should now have the following information:

- **Package 1:**
 - Private key 1
 - `Information package A` containing `PIN_A: pin_hw_a` and `MPK3: master_pub_key_3`

- **Package 2:**
 - Private key 2
 - Information package B containing PWA: your_password_A and PIN_B: pin_hw_b
- **Package 3:**
 - Private key 3
 - Information package C containing PWB: your_password_B and MPK2: master_pub_key_2
- **4:**
 - Hardware Wallet A (containing private key 1)
- **5:**
 - Hardware Wallet B (containing private key 2)
- **6:**
 - Secure note containing:
 - PWA: your_password_A
 - PWB: your_password_B
 - MPK1: master_pub_key_1
 - MPK2: master_pub_key_2
 - MPK3: master_pub_key_3

Since two of your private keys are both on paper and in hardware wallets, this is almost like a 2 of 5 signature scheme. This means that you could lose 3 pieces and still be able to restore your funds (as long as you have your secure note). So how should you handle this information? One option is to make the funds unavailable to access from one place only. Then you'd need to store the packages at five locations. We are going with a more flexible setup in our base case where you always can access your funds from home.

Base case for storage:

1. Package 1 (private Key 1 and information package A) at someone you trust. This package is a little bit more important than the other two as it's easier to restore funds with this package.
2. Package 2 (private Key 2 and information package B) at someone else you trust or in a different location than your home.
3. Package 3 (private Key 3 and information package C) in a vault or a safe deposit box at a bank or at a specialized company.
4. Hardware Wallet A and Hardware Wallet B in your home (preferably in a safe). This gives you the possibility to spend from your cold storage at any time.
5. Digital note, encrypted and stored on a server you can access away from your home or on a USB stored in another location than your home.

You can of course modify this to fit your situation (only use persons you trust, only use vaults etc). But you shouldn't store the private keys in the same building or store more than your two hardware wallets at home.

This is a solid setup. The persons you trust can't access your funds or even see your balance. You can control your funds from your home. If one hardware wallet breaks, you only need

one other key to access your funds. If someone steals your hardware wallets, they would still need both of your PINs and the master public key for the last key to spend anything. The keys stored in other locations needs to be combined with 2 other keys to spend.

If you are giving your packages to other persons. Try to deliver them yourself, never put anything in an e-mail. In the worst case, mail the packages one at a time with snail mail and confirm that they are delivered properly. Tell the persons that are holding the packages what it is and who they need to contact in case of an emergency where they need to access the funds.

It's a good idea to store everything in envelopes and seal with tape (so you'd notice if it's been open). Avoid writing "Bitcoin" or something on the envelopes, they should look as normal as possible. Control your keys from time to time (like once a year). Move your funds to a new multi-sig wallet if one of the keys are lost or compromised.

Finish up by sealing the USB you used for Tails with tape (so you don't use it for something else) and if you used an eternally quarantined computer, put a tape on the ethernet-port and mark it so you or, someone else, don't use it by accident.

Everything is now set to start to HODL with a peace of mind!

Worst case scenarios

So, what are the worst case scenarios? How vulnerable are you to theft or loss and how are your trusted persons going to access your funds in case of an emergency?

Your funds are most vulnerable to "the \$5 wrench attack". That means, if someone breaks into your house and threatens you until you give them your bitcoins. This is why



Feel free to talk about Bitcoin, but never talk about how much bitcoin you own (that's not very classy anyways). That's the best defence against this type of attack. Even if you couldn't access your funds from your home, an attacker could hold you hostage until you pay. You could use multiple wallets and different PINs on your hardware wallets to reduce the risk of losing your main stash in an attack.

Another bad case is if your computer is compromised. If an attacker got access to your secure note or your wallet file in Electrum, they would be able to derive all your addresses. If they had this access, it's very possible that they could be able to get your name and address as well. If this attacker attacks you in your home, multiple wallets wouldn't help since the attacker would know how much you owned. This is truly a worst case scenario and we do what we can to reduce this risk. If you use Tor and/or your full node you reduce the risk compared to simply giving remote servers a lot of this information. But it's still a risk we need to acknowledge.

Important to remember, your funds aren't at risk only because your computer is compromised. As long as one of the hardware wallets do what they are designed to do (not leak the private key) and the last private key was generated properly in Tails, your funds are safe. An attacker would need you to give them access to the funds (physical attack or social engineering).

If the people you trust with your packages were going to collude against you, they would need all 3 packages. If the ones holding package 1 and 2 tried to spend, they wouldn't have password B. If person 1 got access to package 3 she wouldn't have password A and if person 2 had access to package 3 she wouldn't have the public key for key 1 (needed to construct the multi-sig).

What if someone needed to access your funds in case you are hospitalized or worse?

The worst case would be if your house burned down and you and your hardware wallets with it. All three keys stored in other places would then be needed to access your funds. The multi-sig contract could be reconstructed by combining all private keys (+ the seed passwords) or two private keys and the last master public key (in the right order).

If you are in a coma or die (without your house burning down) and a trusted person needs to access your funds, access to package 1 is needed. If we assume that the person can get access to your hardware wallets (drill the safe or whatever is needed). The person can combine its package with either package 2 and the hardware wallets or package 3 and the hardware wallets. They can of course combine it with package 2 and 3 and get access as well.

What about loss of keys?

If your house burns down with your hardware wallets in it, but you survive, you can construct the multi-sig wallet with your secure note and any 2 of the private keys.

If your computer crashes and you lose access to your electrum wallet file and you somehow lose your secure note, you can reconstruct the wallet with your hardware wallets and package 1 or 3.

[P] Improvements

To be a true Bitcoin first class citizen you need to run your own Bitcoin full node and validate all transactions yourself. That way you don't rely on trusted third party servers and as a bonus improves your privacy.

If you run a Bitcoin Core node you can use “Electrum Personal Server” to connect it with Electrum. This can be a little bit tricky for a non-technical user (especially on Windows and Mac). I've created guides for users on Windows and Mac in the bonus section. You can find the guides here: [Windows](#), [Mac](#). Linux user can watch a tutorial here: <https://www.youtube.com/watch?v=1JMP4NZCC5g> (not my tutorial) or follow the official documentation. You can read more about the project on <https://github.com/chris-belcher/electrum-personal-server>. Once done you can use Electrum as usual, but without relying on someone else for verifying and broadcasting transactions. You can use it for more “day-to-day” spending as well. You can connect a hardware wallet to Electrum or use a hot-wallet (private key stored on the computer) and verify all transactions yourself.

The second-best option is to use your Bitcoin full node to add watch addresses. You simply add all addresses you want to keep track of and your node checks for any activity on those addresses. If you have a full node running, this is very simple. Check the bonus section for how to [add watch addresses in Bitcoin Core](#). You'd still have to connect to random Electrum servers to broadcast transactions, but that isn't terrible if you use Tor or a VPN. The crucial part is to validate every transaction you receive.

Another very important aspect of privacy is what traces you leave on chain. A very common example is that you buy bitcoin on an exchange that use KYC (know your customer). They then have all your personal information. If you then transfer your funds directly from the exchange to your cold storage, they would still know what bitcoin you most likely own. There are multiple other ways your funds can be linked to you and you never know who has this information. An easy way to break this link is to mix your coins with a technique called coinjoin. This has often been done by central parties, I would advise against those services. You put your trust in them, both with your privacy and with your keys as they could steal your bitcoin. A better alternative is to use a service that never control our keys. Two alternatives are [Joinmarket](#) and Wasabi Wallet. You can find a guide and best practises for [Wasabi Wallet](#) in the bonus section.

Note: Some exchanges treat coins involved in coin join transactions as suspicious. If you do a coin join transaction and then tries to transfer the coins to an exchange, they might reject your deposit. You have to decide yourself about privacy vs ease of selling.