

Krux and miniscript

an introduction to BIP379 and a little game

qlrd

Miniscript

Definition 1: BIP 379.

(...) a language for writing (a subset of) **Bitcoin Scripts** in a structured way, enabling analysis, composition, generic signing and more. [1]

Back to the basics

Bitcoin script

Definition 2: .

(...) an unusual stack-based language with many edge cases designed for implementing spending conditions consisting of various combinations of signatures, hash locks, and time locks. [1]

Bitcoin script

Common transactions from [2] and [3]

Comment	Unlock	Lock
P2PK	<sig> <pk>	OP_CHECKSIG
P2PKH	<sig> <pk>	OP_DUP OP_HASH160 <pkh> OP_EQUALVERIFY OP_CHECKSIG
Multisig 2-of-3	OP_0 <sigA> <sigB>	2 <pkA> <pkB> <pkC> 3 OP_CHECKMULTISIG

Bitcoin script

Freezing funds until a time in the future from [2]

Unlock	Lock
<sig> <pk>	<expiry time> OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP OP_HASH160 <pkh> OP_EQUALVERIFY OP_CHECKSIG

Bitcoin script

Timelock variable multisignature from [3]: 2-of-3 multisig; after 30 days 1-of-3 with a lawyers's signature; after 90 days the lawyer's signature.

Unlock	Lock
OP_0 <sigA> <sigB> OP_TRUE OP_TRUE	OP_IF OP_IF 2 OP_ELSE <30 days> OP_CHECKSEQUENCEVERIFY OP_DROP <sigD> OP_CHECKSIGVERIFY 1 OP_ENDIF <sigA> <sigB> <sigC> 3 OP_CHECKMULTISIG OP_ELSE <90 days> OP_CHECKSEQUENCEVERIFY OP_DROP <sigD> OP_CHECKSIG OP_ENDIF

The issue [1]

Given a combination of spending conditions, it is challenging to:

- find the most economical script to implement it;
- implement a composition of their spending conditions;
- find out what spending conditions it permits.

...

The motivation

Miniscript has a structure that allows composition: a representation for scripts that makes these type of operations possible.

Implementations

- Peter Wuille's reference implementation
- C++:
 - Bitcoin-core
- Rust:
 - rust-miniscript
 - Liana
- Go:
 - Tutorial: Understanding Bitcoin Miniscript - Part III
- Python:
 - Embit's miniscript.py
 - Krux (branch p2wsh_miniscript)
 - Krux (branch tr_miniscript)

Hands on: setup a single-inheritance scheme



Figure 1: Before start, download a Krux demo android app. https://github.com/odudex/krux_binaries/blob/main/Android/Krux_25.01.beta8_Android_0.2.apk

Hands on: setup a single-inheritance scheme (ii)



Figure 2: Before start, download Liana coordinator in your computer. <https://wizardsardine.com/liana/>

Hands on: setup a single-inheritance scheme (iii)



Figure 3: Before start, prepare in your computer the SeedQReader. It will be a helper for sign transactions. <https://github.com/pythcoiner/SeedQReader>

Hands on: setup a single-inheritance scheme (iv)

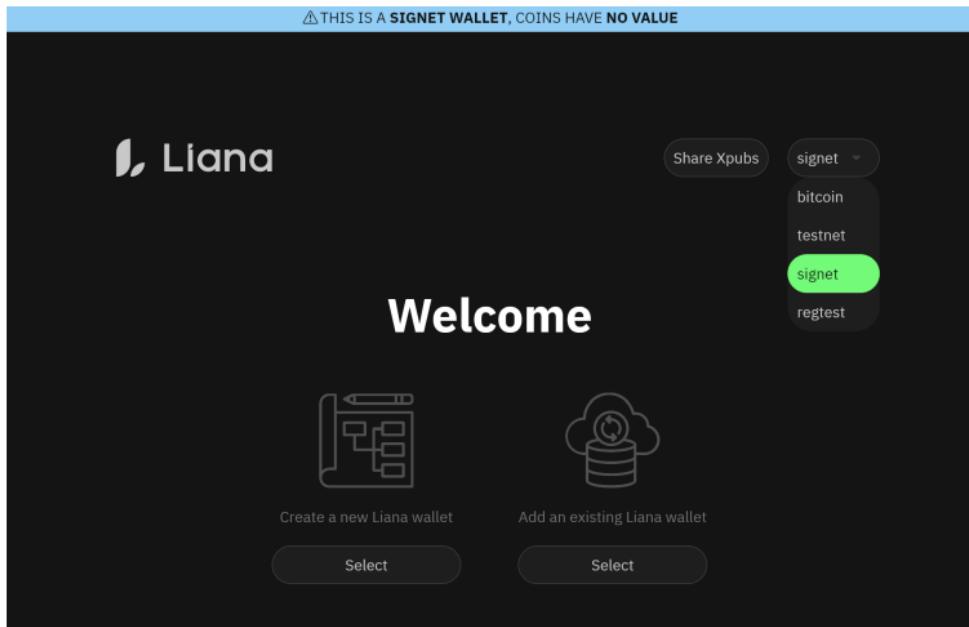


Figure 4: We will select **signet** to not risk our beloved sats.

Hands on: setup a single-inheritance scheme (v)

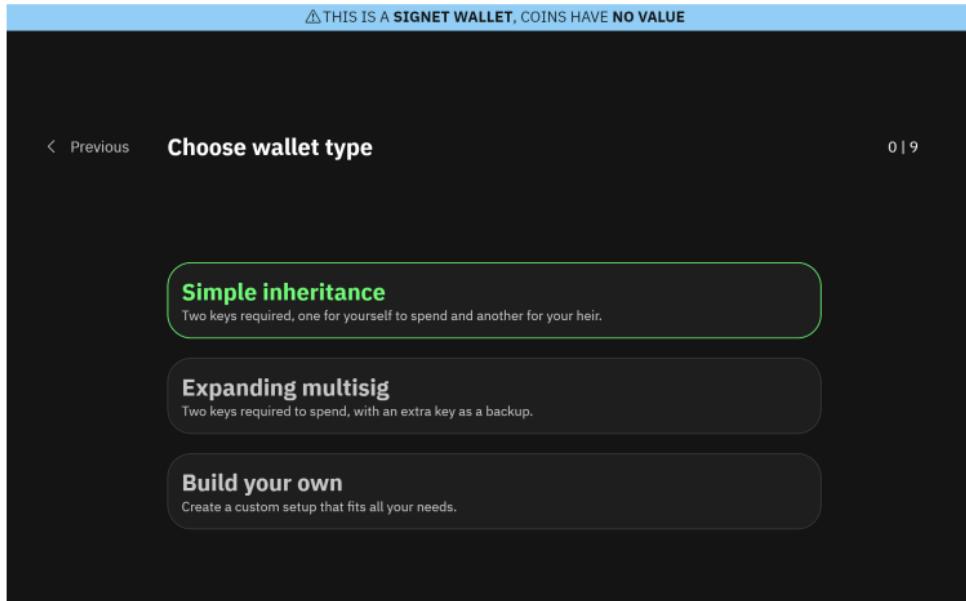


Figure 5: Selecting **Simple inheritance** scheme.

Hands on: setup a single-inheritance scheme (vi)

⚠ THIS IS A **SIGNET WALLET**, COINS HAVE NO VALUE

◀ Previous **Introduction** 1 | 9

Simple inheritance wallet

For this setup you will need 2 Keys: Your Primary Key (for yourself) and an Inheritance Key (for your heir). For security reasons, we suggest you use a separate Hardware Wallet for each key.

● Primary key ○ Inheritance key

You will always be able to spend using your Primary Key.
After a period of inactivity (but not before that) your Inheritance Key will become able to recover your funds.

The diagram shows two spending policies on a timeline. On the left, under 'Primary spending policy:', a green key icon is labeled 'Primary Key'. An arrow points from a green dot at the start of the timeline to another green dot further along, labeled 'CAN SPEND'. On the right, under 'Recovery spending policy:', a red key icon is labeled 'Inheritance Key'. At the start, a red dot is labeled 'CAN'T SPEND (TIMELOCKED)'. After a vertical line labeled 'After some time* of wallet inactivity', a green dot is labeled 'CAN SPEND'.

Primary spending policy:
Primary Key

Receipt of funds

After some time* of wallet inactivity

CAN SPEND

Recovery spending policy:
Inheritance Key

CAN'T SPEND (TIMELOCKED)

CAN SPEND

*The time range (timelock) for the activation of keys can be configured in the next step.

Next

Figure 6: An explanation how **Simple inheritance** scheme works.

Hands on: setup a single-inheritance scheme (vii)

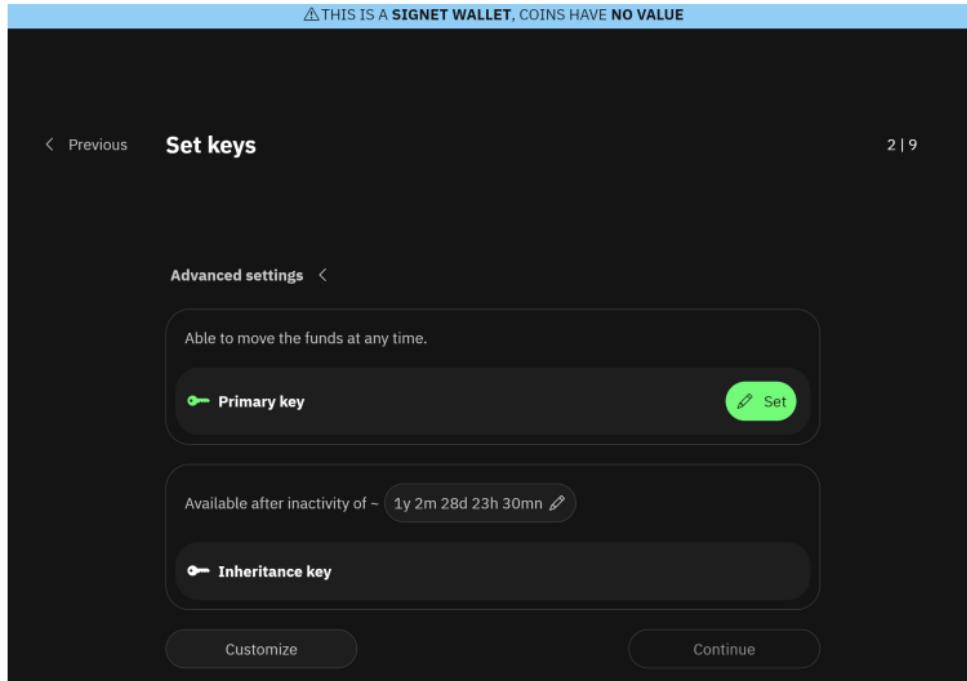


Figure 7: Liana's menu to setup **Simple inheritance**.

Hands on: setup a single-inheritance scheme (viii)

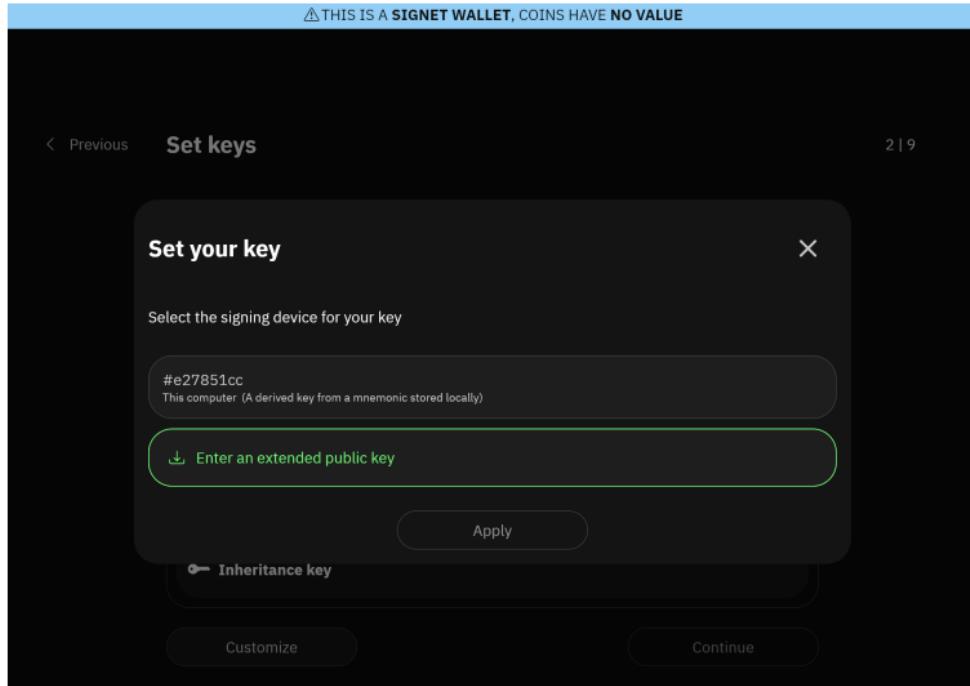


Figure 8: Liana's menu to setup **Simple inheritance**.

Hands on: setup a single-inheritance scheme (ix)

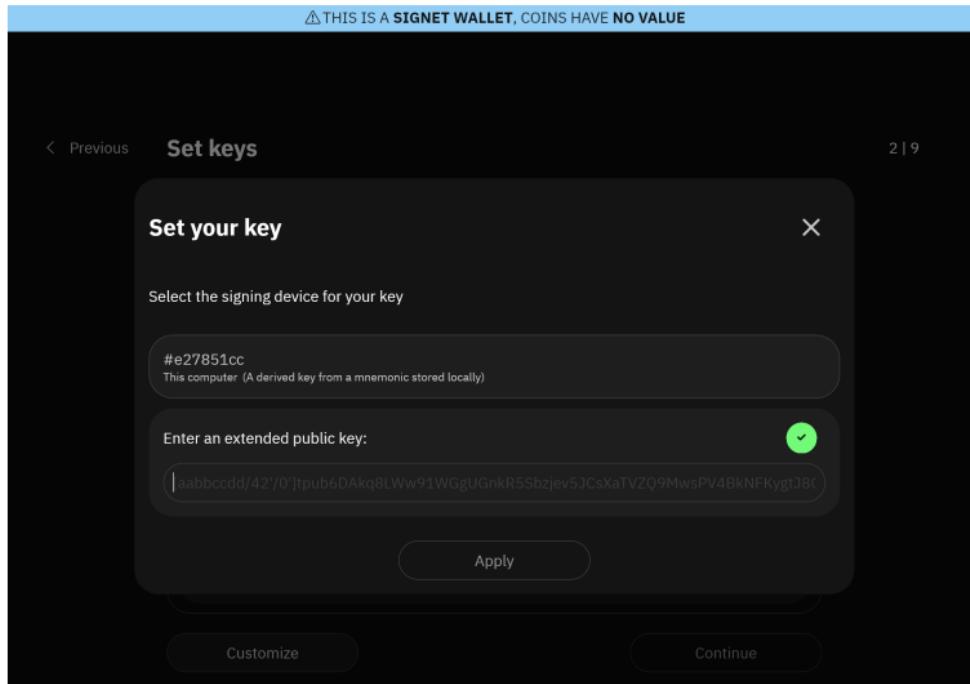


Figure 9: Liana's waiting for the first (key-expression + xpub) to setup **Simple inheritance.**

Hands on: setup a single-inheritance scheme (x)

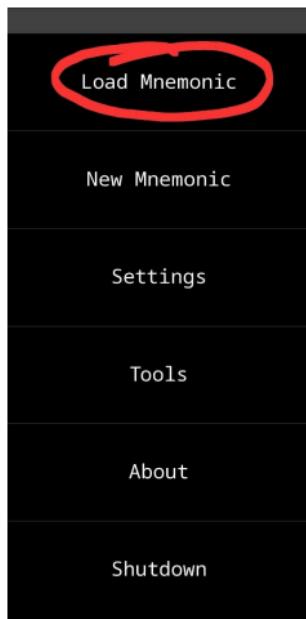


Figure 10: Load a previous created wallet on **Krux**.

Hands on: setup a single-inheritance scheme (xi)

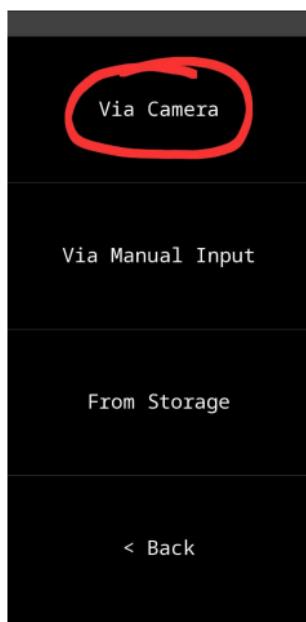


Figure 11: Select **Via Camera** to Load.

Hands on: setup a single-inheritance scheme (xii)

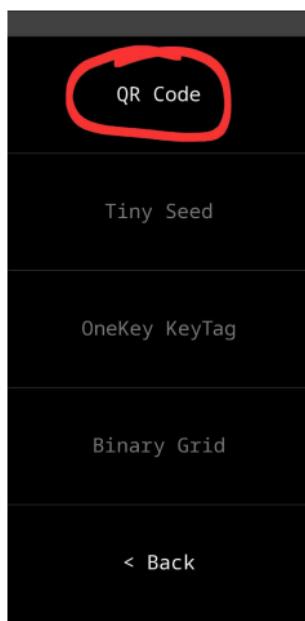


Figure 12: Select **QR Code** to scan.

Hands on: setup a single-inheritance scheme (xiii)

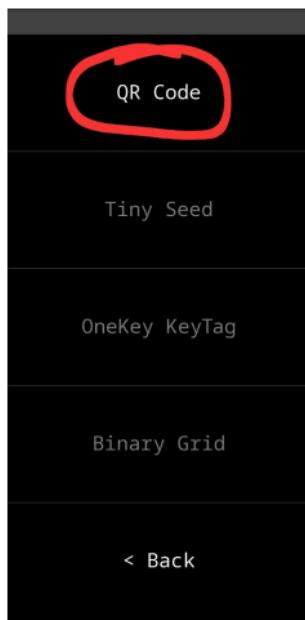


Figure 13: Select **QR Code** to scan.

Hands on: setup a single-inheritance scheme (xiv)



Figure 14: Scan it.

Hands on: setup a single-inheritance scheme (xv)



Figure 15: Decrypt a encrypted mnemoninc.

Hands on: setup a single-inheritance scheme (xvi)

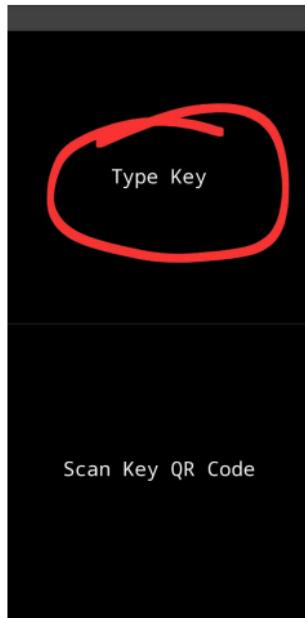


Figure 16: Select **Type Key** to type a decrypt key. Alternatively, you can scan a previous created QRCode key.

Hands on: setup a single-inheritance scheme (xvii)

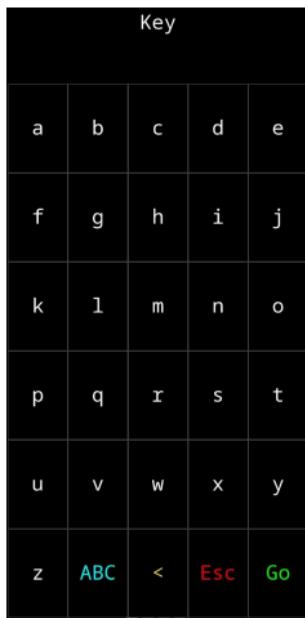


Figure 17: **Type Key** keyboard. Try something like 'test' or another one.

Hands on: setup a single-inheritance scheme (xviii)



Figure 18: 'test' was typed as the decrypt key.

Hands on: setup a single-inheritance scheme (xix)



Figure 19: Decrypted a double-mnemonic (see *, the first 12 words are a valid mnemonic; the last 12 are a valid mnemonic and the 24 words are another valid mnemonic).

Hands on: setup a single-inheritance scheme (xx)

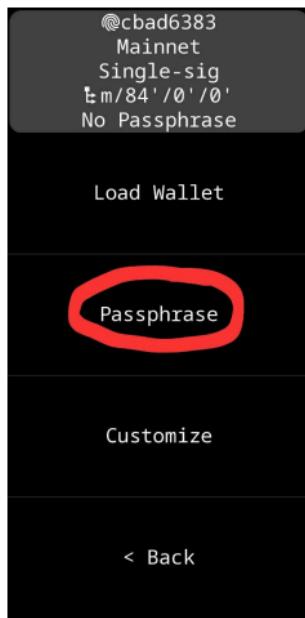


Figure 20: Loaded wallet secured by an optional BIP39 passphrase.

Hands on: setup a single-inheritance scheme (xxi)

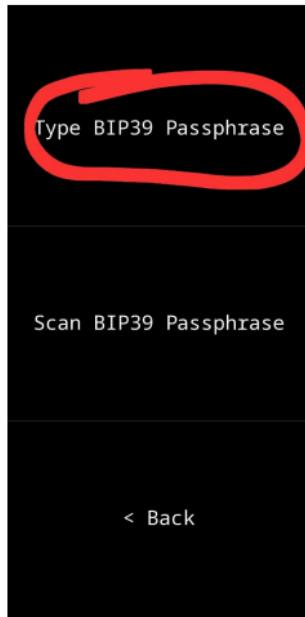


Figure 21: Select **Type BIP39 Passphrase** to access keyboard. Alternatively you can scan an QRCode encoded one.

Hands on: setup a single-inheritance scheme (xxii)



Figure 22: Select **Type BIP39 Passphrase** to access keyboard. Alternatively you can scan an QRCode encoded one.

Hands on: setup a single-inheritance scheme (xxiii)



Figure 23: Krux ask for passphrase confirmation. Remember that different passphrases leads to different wallets!

Hands on: setup a single-inheritance scheme (xxiv)

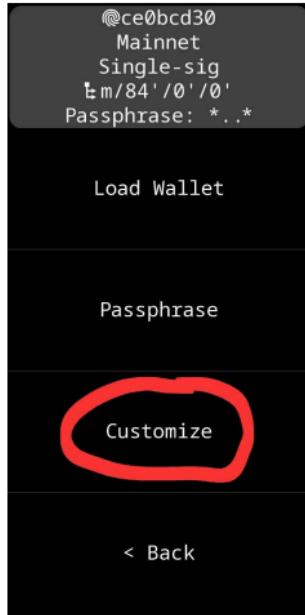


Figure 24: A different wallet was loaded (verify the upper checksum). But we still need to customize some stuffs.

Hands on: setup a single-inheritance scheme (xxv)

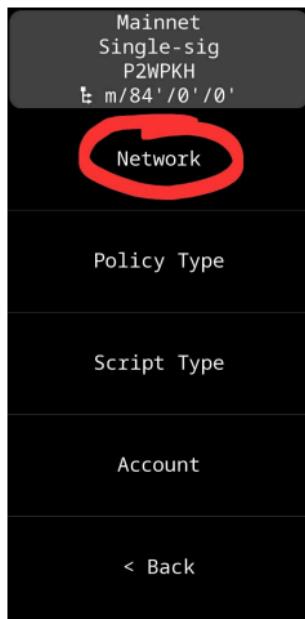


Figure 25: Let's change the network, since we're testing!

Hands on: setup a single-inheritance scheme (xxvi)



Figure 26: For signet wallet in Liana, we can use a testnet on Krux.

Hands on: setup a single-inheritance scheme (xxvii)

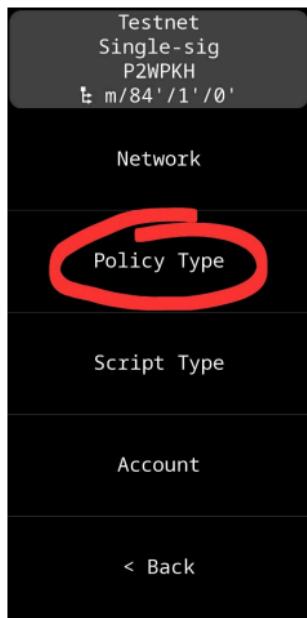


Figure 27: Let's change policy to be able to do a inheritance scheme.

Hands on: setup a single-inheritance scheme (xxviii)



Figure 28: Select miniscript policy.

Hands on: setup a single-inheritance scheme (xxix)

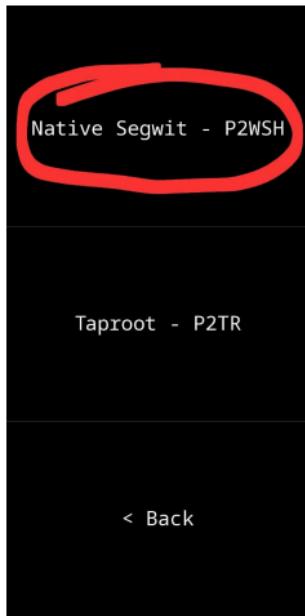


Figure 29: Then select the between BIP382 (wsh) or BIP386 (tr) descriptors.

Hands on: setup a single-inheritance scheme (xxx)



Figure 30: **Optional** you can edit your derivation path for the inheritance scheme.

Hands on: setup a single-inheritance scheme (xxxi)

Derivation Path		
m/48'/1'/0'/2'		
1	2	3
4	5	6
7	8	9
/	0	'
<	Esc	Go

Figure 31: **Optional** For educational purposes, let be the default
 $m/48'/1'/0'/2'$.

Hands on: setup a single-inheritance scheme (xxxii)



Figure 32: Now we can back to main logged menu.

Hands on: setup a single-inheritance scheme (xxxiii)

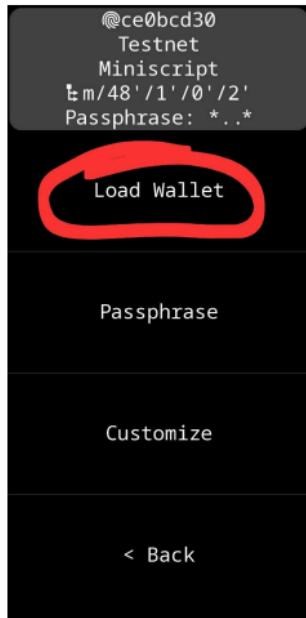


Figure 33: All done! 🎉 And load properly the wallet

Hands on: setup a single-inheritance scheme (xxxiv)

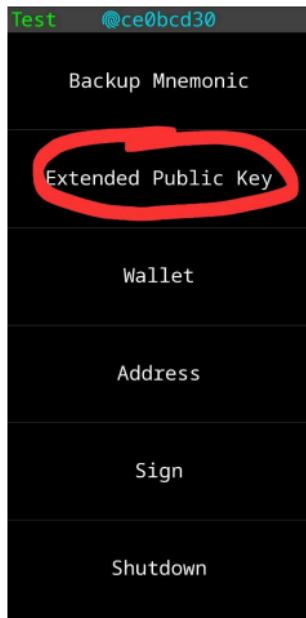


Figure 34: We need to load the key expression + tpub.

Hands on: setup a single-inheritance scheme (xxxv)

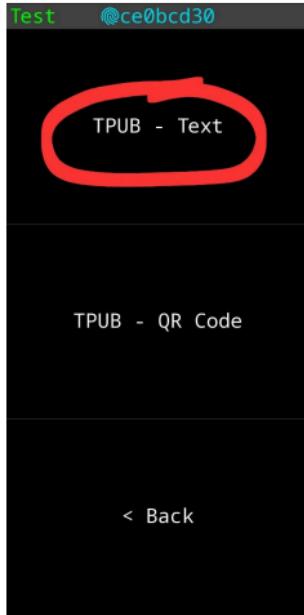


Figure 35: If you have a real krux device, it's recommended to load **TPUB - text**.

Hands on: setup a single-inheritance scheme (xxxvi)



Figure 36: For real devices, save the key expression + tpub into a SDCard.

Hands on: setup a single-inheritance scheme (xxxvii)

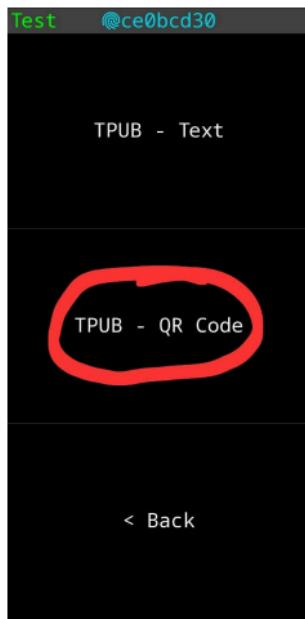


Figure 37: For demo app, press **TPUB - QRCode**.

Hands on: setup a single-inheritance scheme (xxxviii)

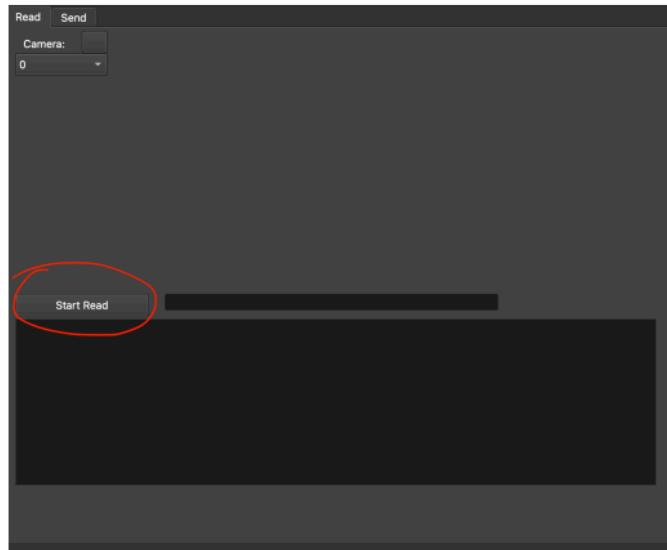


Figure 38: Open SeedQReader and scan the tpub in qrcode

Hands on: setup a single-inheritance scheme (xxxix)

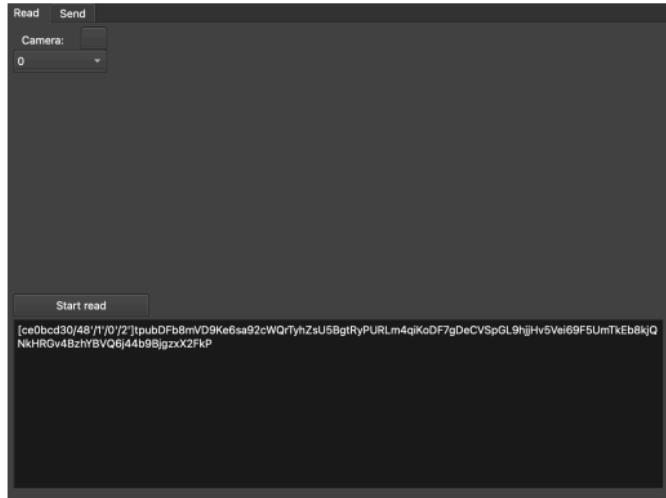


Figure 39: Now copy the output to Liana.

Hands on: setup a single-inheritance scheme (xl)

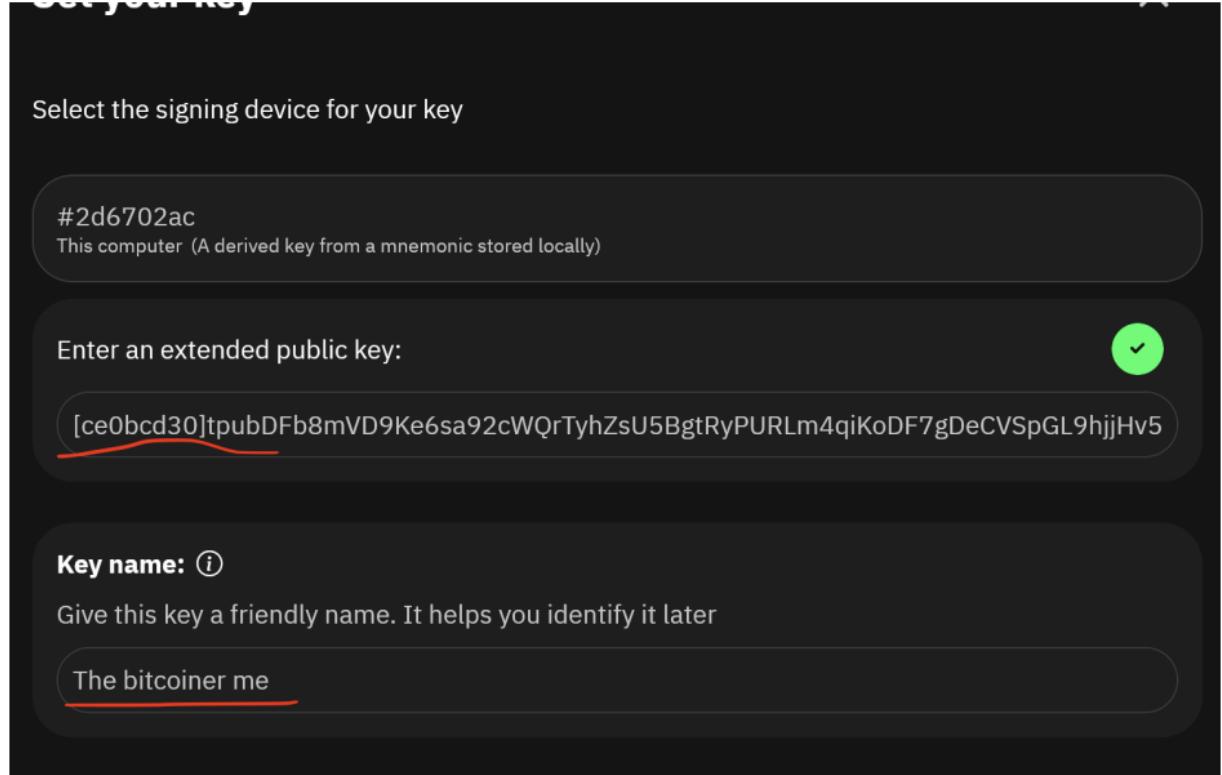


Figure 40: Type key fingerprint + tpub copied to Liana and add an Alias for it.
52/111

Exercise 1

Exercise 1

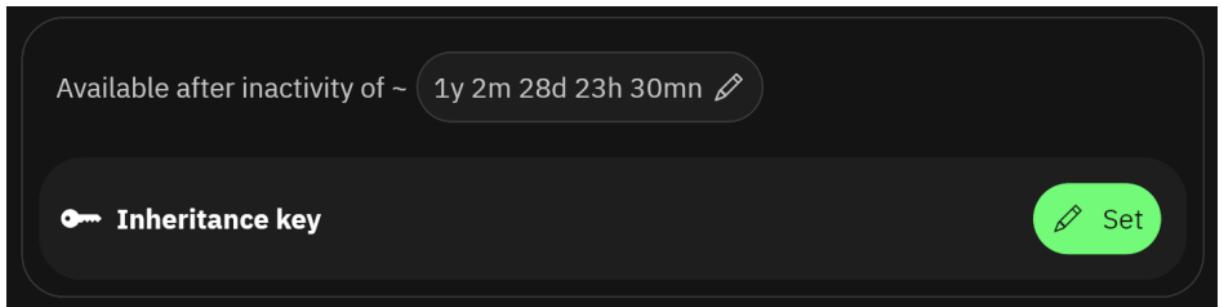


Figure 41: Set a timelock to your second key.

Exercise 2

Exercise 2

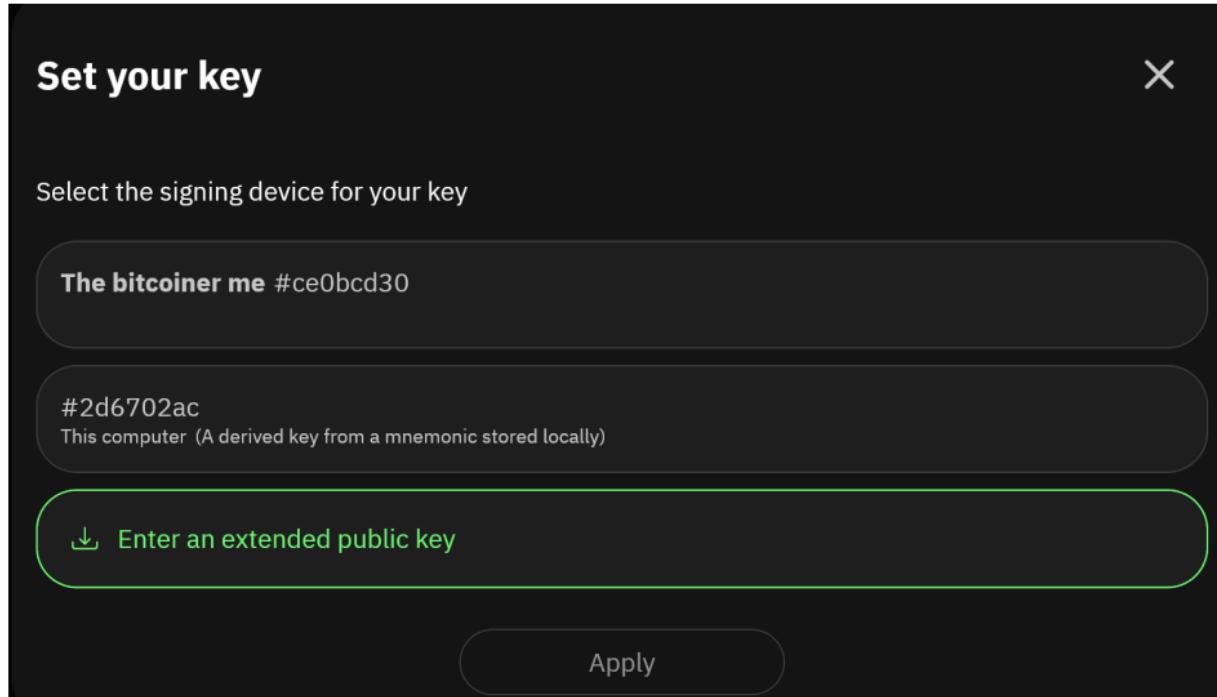


Figure 42: Now you will repeat all previous procedures to a heir key. For pedagogical purposes, you can use same mnemonic with another passphrase.

Backup your descriptor

Backup your descriptor

Advanced settings <

Able to move the funds at any time.

🔑 Primary key **The bitcoiner me**

✓  Edit

Available after inactivity of ~  1h

🔑 Inheritance key **the bitcoiner heir**

✓  Edit

Customize

Continue

Figure 43: Check if all is ok.

Backup your descriptor (ii)

The screenshot shows a dark-themed web page titled "Backup your wallet descriptor". At the top left is a "Previous" button, and at the top right is a "4 | 9" page number indicator. The main content area contains a message about the importance of backing up both the private key and the descriptor, followed by a "Learn more" link and a back arrow. Below this, there are two sections: "The descriptor:" containing a long hex string, and "The wallet policy:" containing a policy statement with a "Copy" button.

The descriptor is necessary to recover your funds. The backup of your key (via mnemonics, sometimes called 'seed words') is not enough. Please make sure you have backed up both your private key and your descriptor.

[Learn more](#) <

The descriptor:

```
wsh(or_d(pk([ce0bcd30]tpubDFb8mVD9Ke6sa92cWQrTyhZsU5BgtRyPURLm4qiKoDF7gDe
```

[Copy](#)

The wallet policy:

1 signature by **The bitcoiner me** can always spend this wallet's funds (Primary path)
1 signature by **the bitcoiner heir** can spend coins inactive for **6 blocks (~1h)** (Recovery pa

Figure 44: Check the policy and backup the miniscript descriptor on a SDCard to load it with Krux.

Backup your descriptor (iii)

```
wsh(  
    or_d(  
        pk([ce0bcd30/48'/1'/0'/2'])tpubDFb...FkP/<0;1>/*),  
        and_v(  
            v:pkh([73c9118b/48'/1'/0'/2'])tpub...CJR/<0;1>/*),  
            older(6)  
        )  
    )  
)#sefucc2h
```

Select a new node

Select a new node

The screenshot shows a dark-themed user interface for selecting a node. At the top center, it says "Choose backend". On the left, there's a "Previous" button. On the right, it says "5 | 9". Below the title, there are two main options: "Use your own node" and "Use Liana Connect". Each option has a detailed description and a "Select" button at the bottom.

Use your own node

Use your already existing Bitcoin node or automatically install one. The Liana wallet will not connect to any external server.

This is the most private option, but the data is locally stored on this computer, only. You must perform your own backups, and share the descriptor with other people you want to be able to access the wallet.

Select

Use Liana Connect

Use our service to instantly be ready to transact. Wizardsardine runs the infrastructure, allowing multiple computers or participants to connect and synchronize.

This is a simpler and safer option for people who want Wizardsardine to keep a backup of their descriptor. You are still in control of your keys, and Wizardsardine does not have any control over your funds, but it will be able to see your wallet's information, associated to an email address. Privacy focused users should run their own infrastructure instead.

Select

Figure 45: Select a proper node. For the workshop purpose, select Liana Connect.

Select a new node (ii)

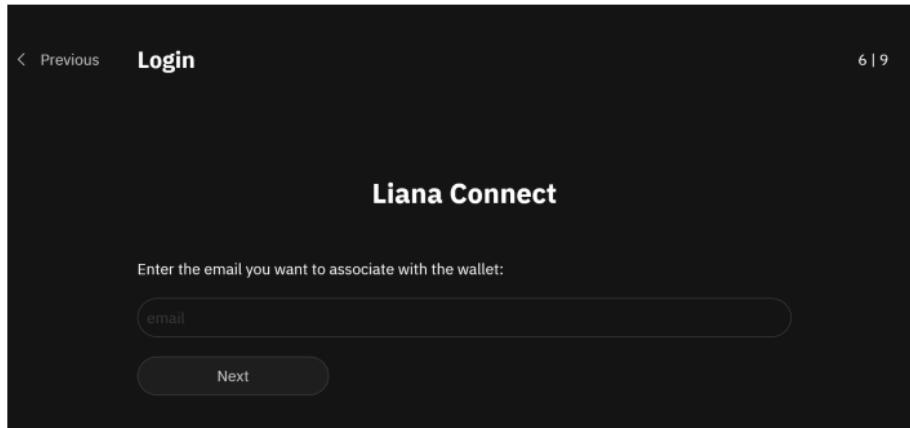


Figure 46: Put your email to receive an OTP.

Select a new node (iii)

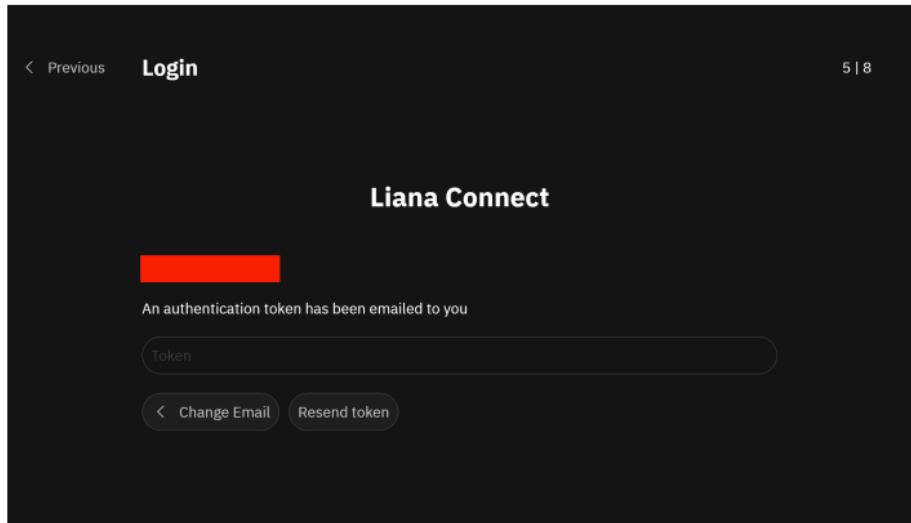


Figure 47: Type the received OTP.

Select a new node (iv)

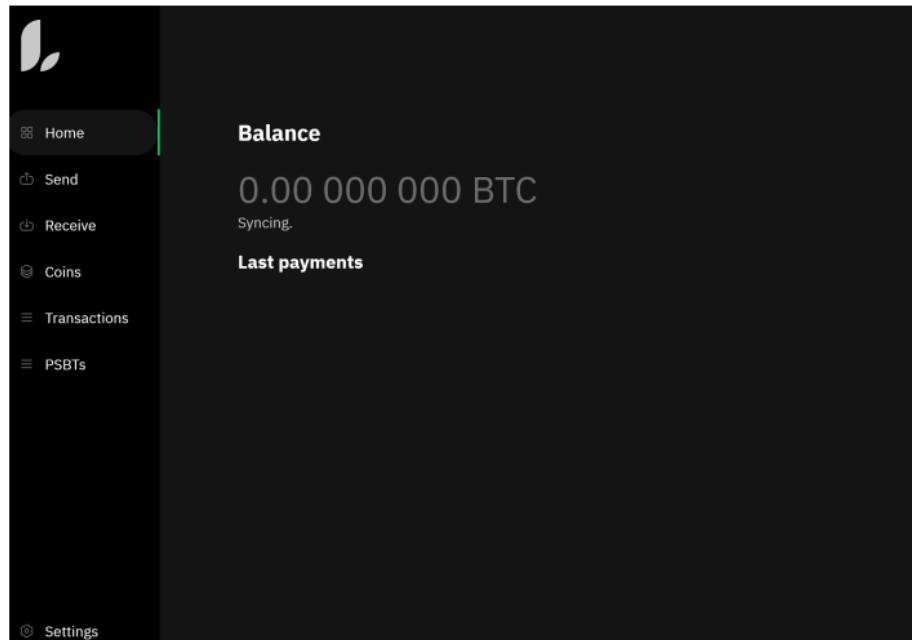


Figure 48: Fully loaded a **simple inheritance** wallet on Liana.

Receive coins

Receive coins

Before receive coins, we recomend to generate a QRCode for your descriptor with SeedQReader.

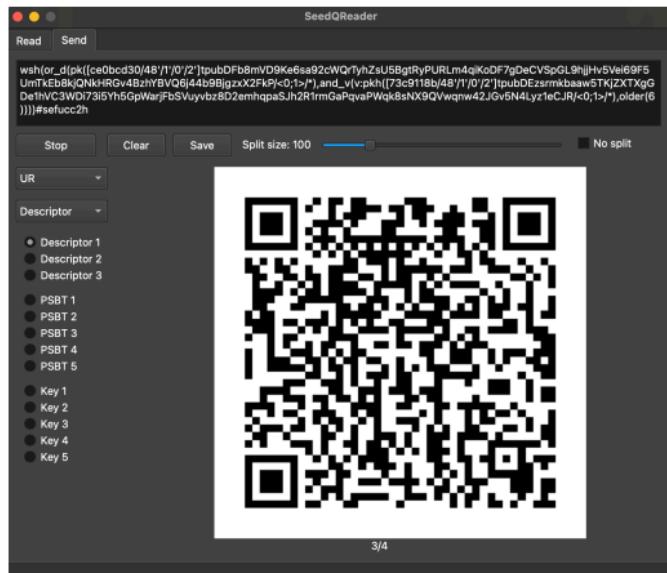


Figure 50: Like this one.

Receive coins (ii)

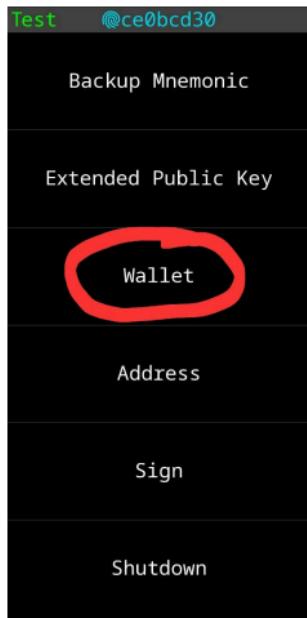


Figure 51: On a already loaded wallet, click on **Wallet**.

Receive coins (iii)

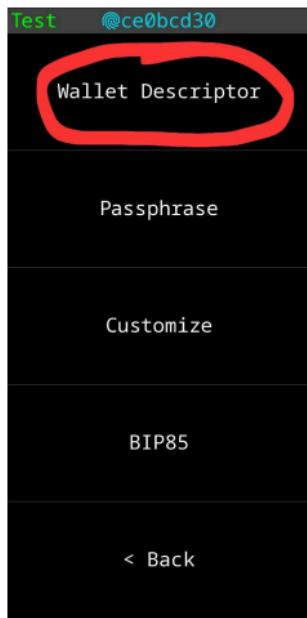


Figure 52: Select **Wallet Descriptor**

Receive coins (iv)



Figure 53: You will receive a warn to load a descriptor.

Receive coins (v)

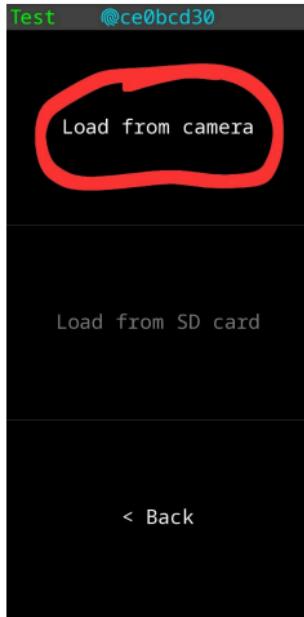


Figure 54: Select **Load from camera**. With a real Krux device, we recommend **Load from SD Card** and copy + paste the descriptor.

Receive coins (vi)

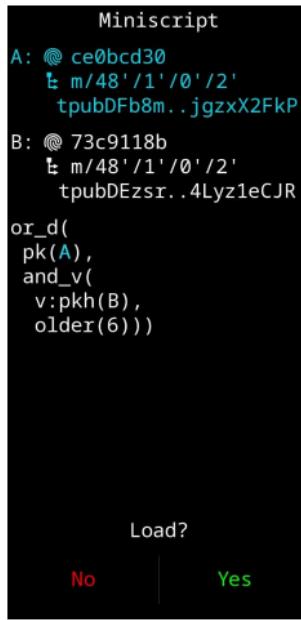


Figure 55: Once loaded, you will can see the same Liana's policy on Krux, where **A** (in blue) is your key and **B** is the heir key. The blue highlights that we loaded on Krux the **A** wallet.

Receive coins (vii)

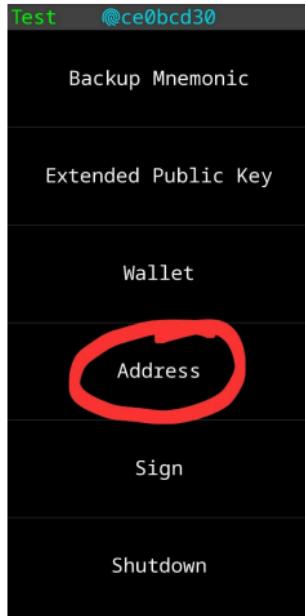


Figure 56: Now we start to check addresses to receive coin. Select **Address** on main menu.

Receive coins (viii)

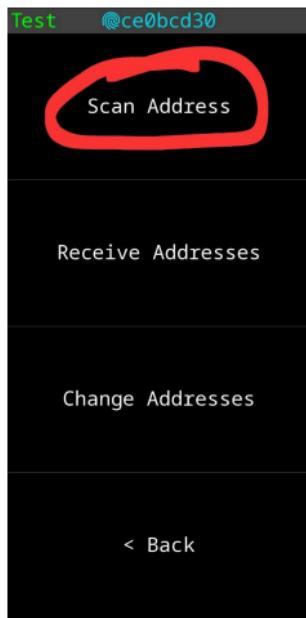


Figure 57: Select **Scan Address**.

Receive coins (ix)

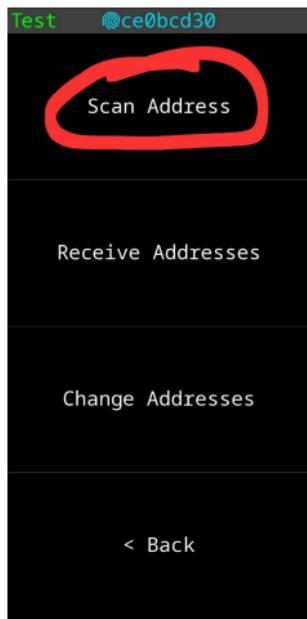


Figure 58: Select **Scan Address**.

Receive coins (x)

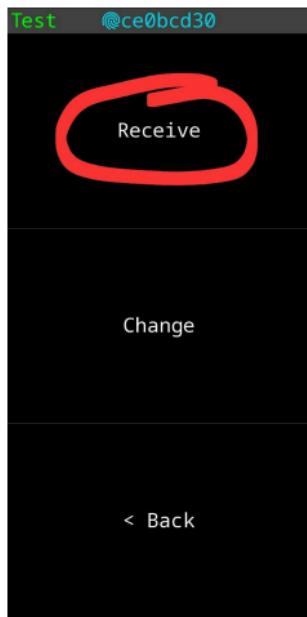


Figure 59: Select **Receive**. Now we will switch to Liana to scan some address.

Receive coins (xi)



Figure 60: Click **Receive coins**.

Receive coins (xii)

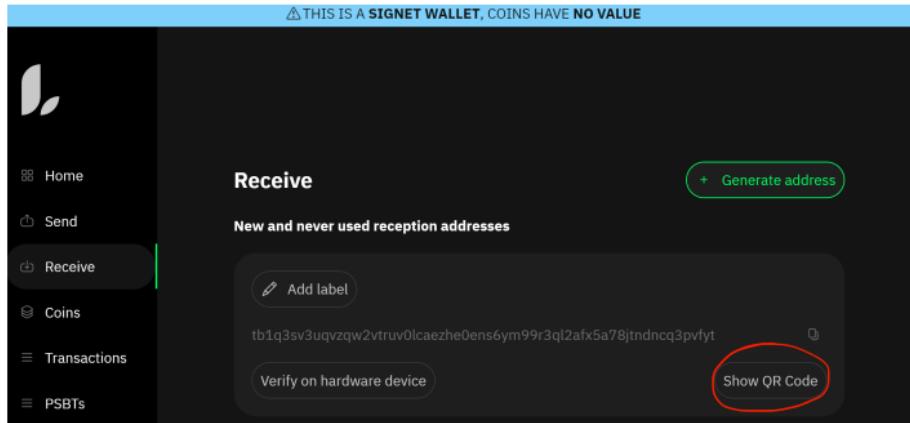


Figure 61: Then **Generate address**.

Receive coins (xiii)



Figure 62: Let's scan and back to Krux.

Receive coins (xiv)



Figure 63: Once scan is done, it will show a qrcode and the correspondent address. Click anywhere on screen and proceed.

Receive coins (xv)



Figure 64: Krux will show a warning about the address verification.

Receive coins (xvi)

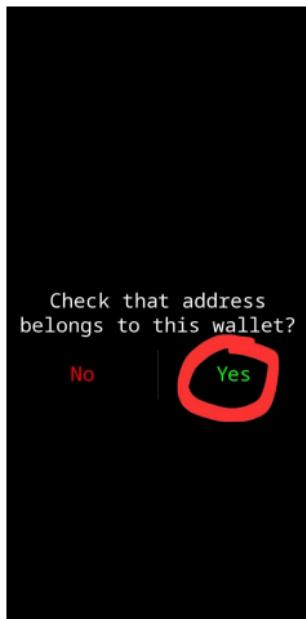


Figure 65: Krux will show a warning about the address verification.

Receive coins (xvii)



Figure 66: Krux verified that the Liana address is from a derived one on Krux.
We can deposit sats safely!

Receive coins (xviii)



Figure 67: Let's get some faucet coins on <https://signet25.bublina.eu.org/>
84/111

Receive coins (xix)

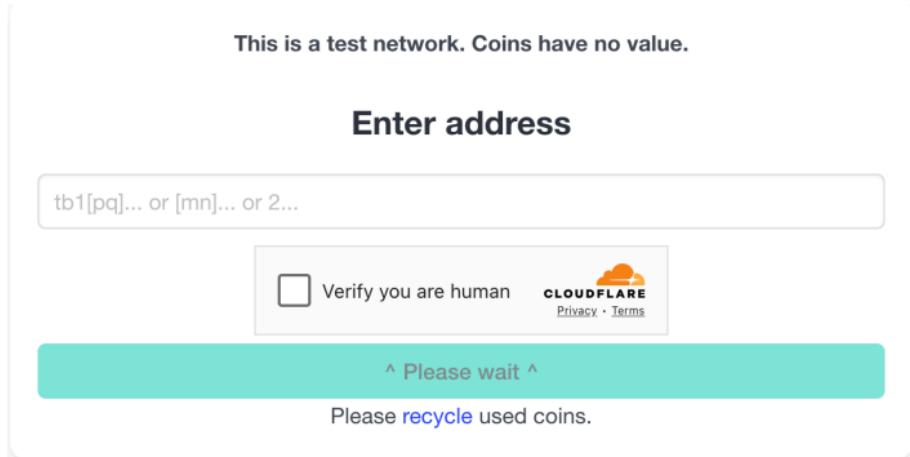


Figure 68: Copy and paste the Liana address here to receive

Receive coins (xx)

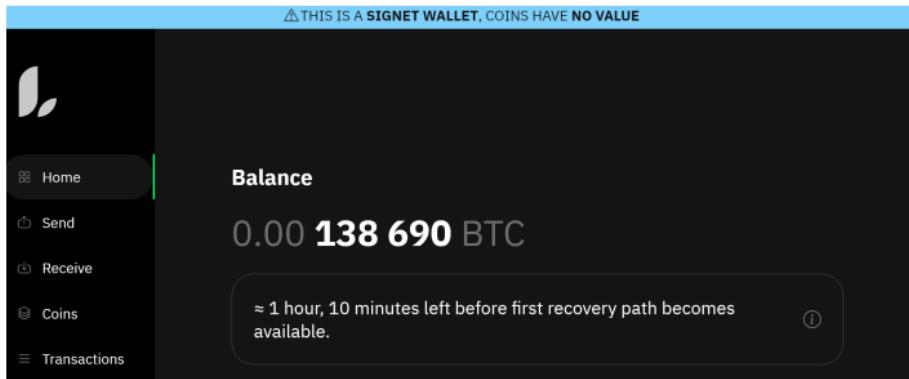


Figure 69: After confirmed on signet blockchain, your wallet will show that you can spend with first key and after sometime with the **first recovery path** (from the heir's key).

Spend with your key

Spend with your key

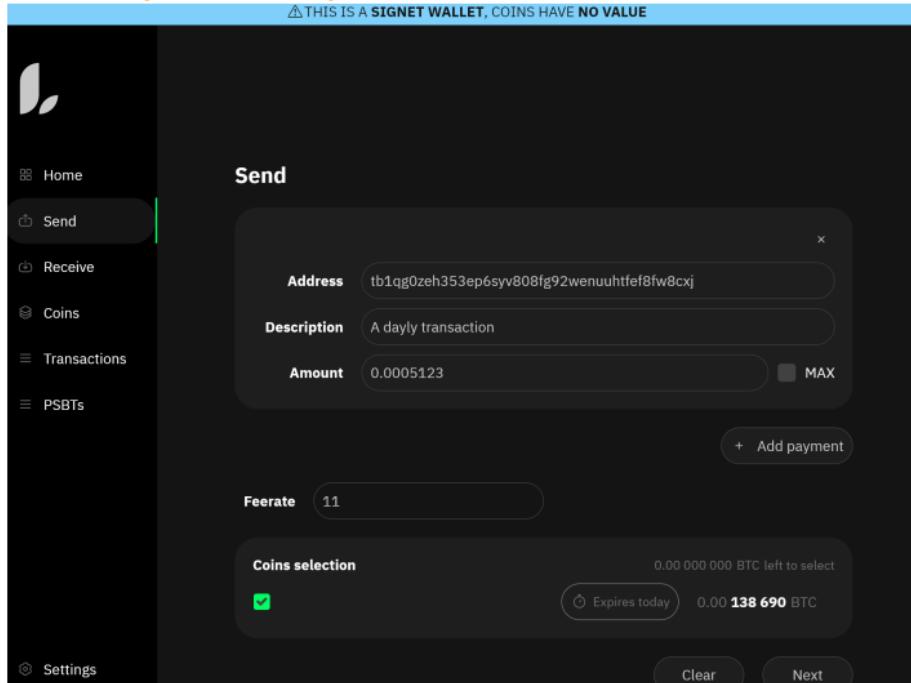


Figure 70: Select **Send**. The window will change to a form. Fill the form (like another coordinator), and select, at the bottom, the **coin selection**. Then click **Next**.

Spend with your key (ii)

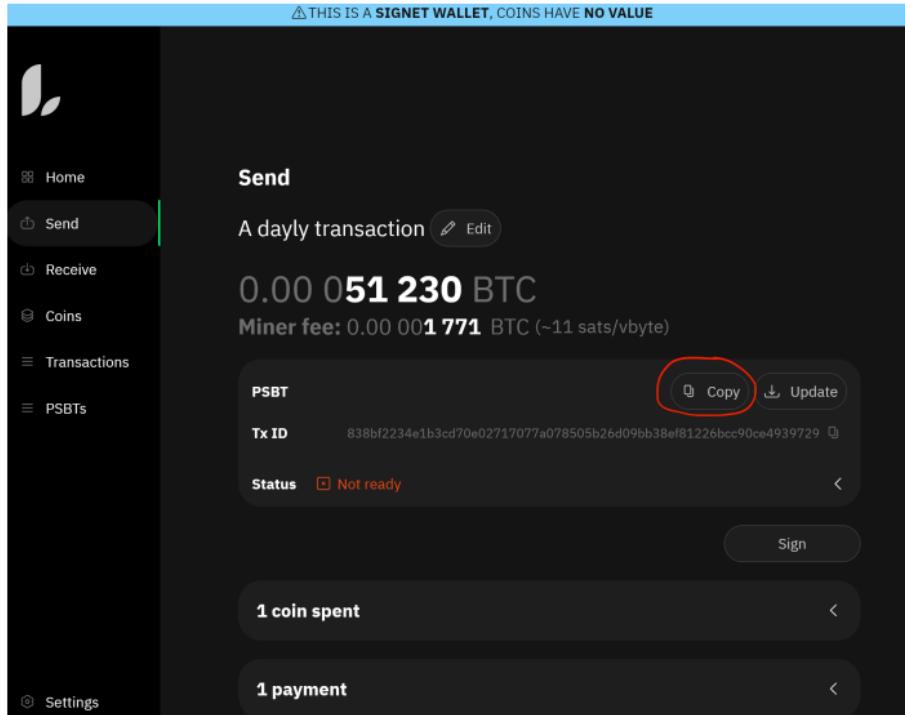


Figure 71: The window will change to a window with a unsigned transaction. To sign, we will copy the **PSBT** and paste it to SeedQReader.

Spend with your key (iii)



Figure 72: Select **PSBT-1**, paste the PSBT to text field and change to krux to sign it.

Spend with your key (iv)

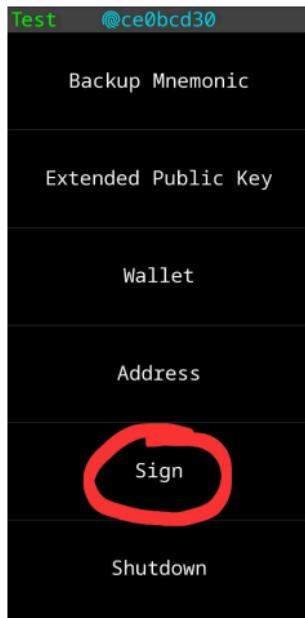


Figure 73: Click on **Sign**.

Spend with your key (v)

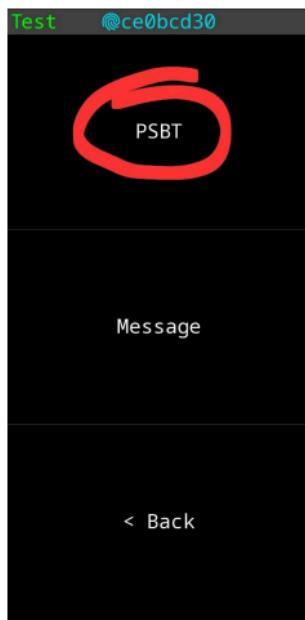


Figure 74: Click on **PSBT**.

Spend with your key (vi)

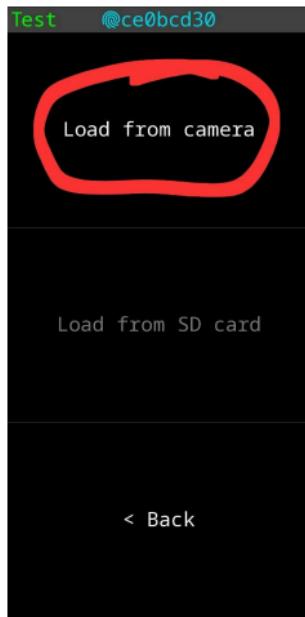


Figure 75: Click on **Load from camera** and scan from SeedQReader.

Spend with your key (vii)

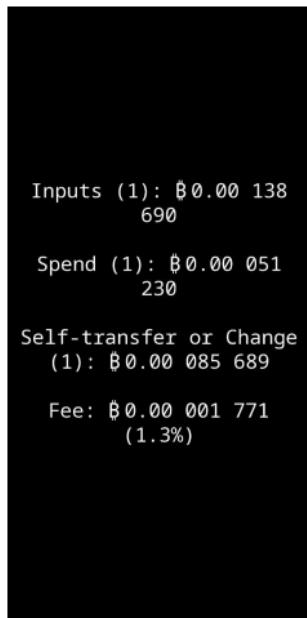


Figure 76: Once loaded, krux will show a summary of PSBT's contents.

Spend with your key (viii)



Figure 77: Click on screen and it will show the outpoints (the spent outpoint).

Spend with your key (ix)



Figure 78: Click on screen and it will show the outpoints (the change outpoint, if any).

Spend with your key (x)

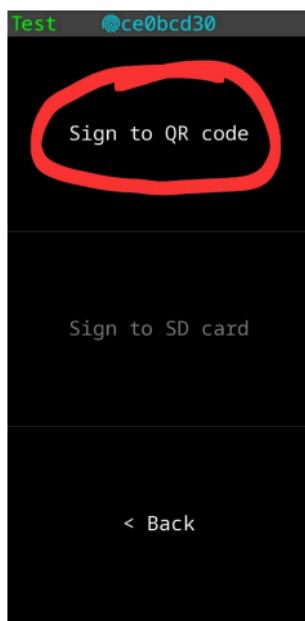


Figure 79: Now sign to a qrcode and use it to be scanned on SeedQReader.

Spend with your key (xi)

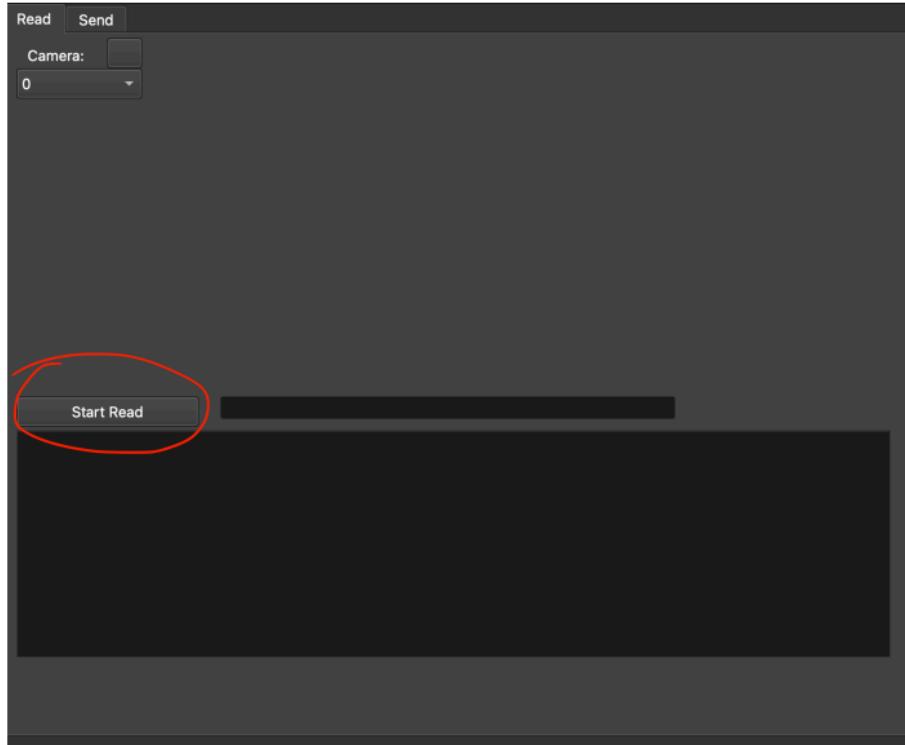


Figure 80: Let's scan the qrcode from krux

Spend with your key (xii)

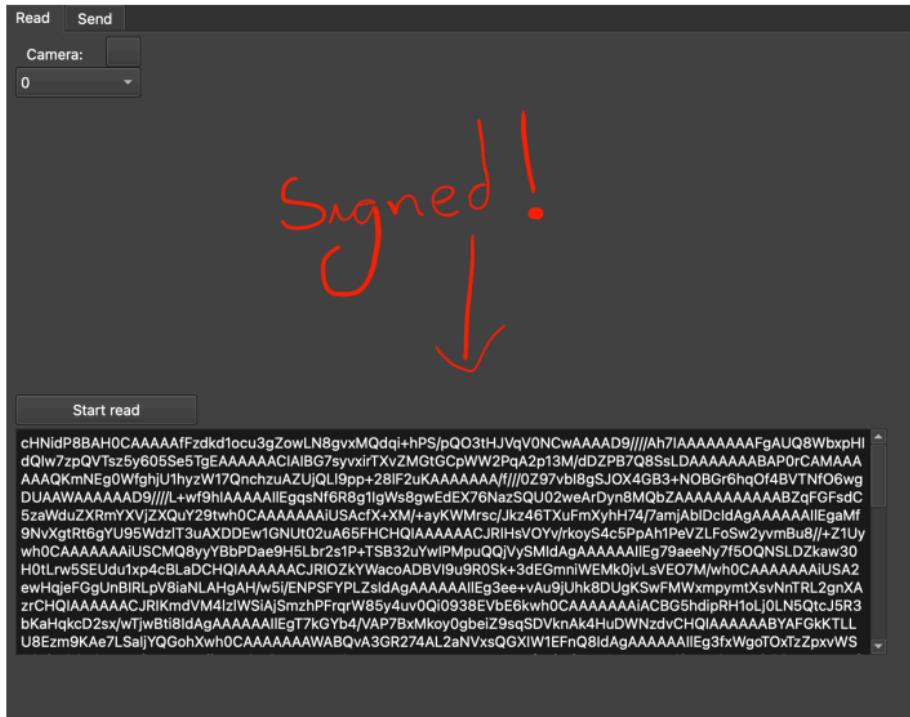


Figure 81: SeedQReader will output the signed PSBT. Copy it to Liana!

Spend with your key (xiii)

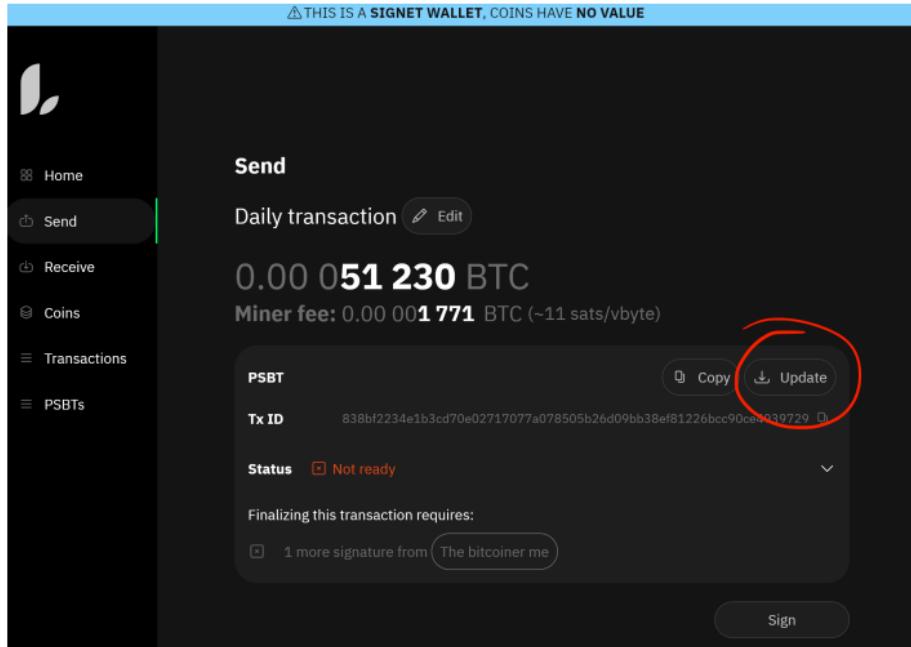


Figure 82: Click on **Update**.

Spend with your key (xiv)

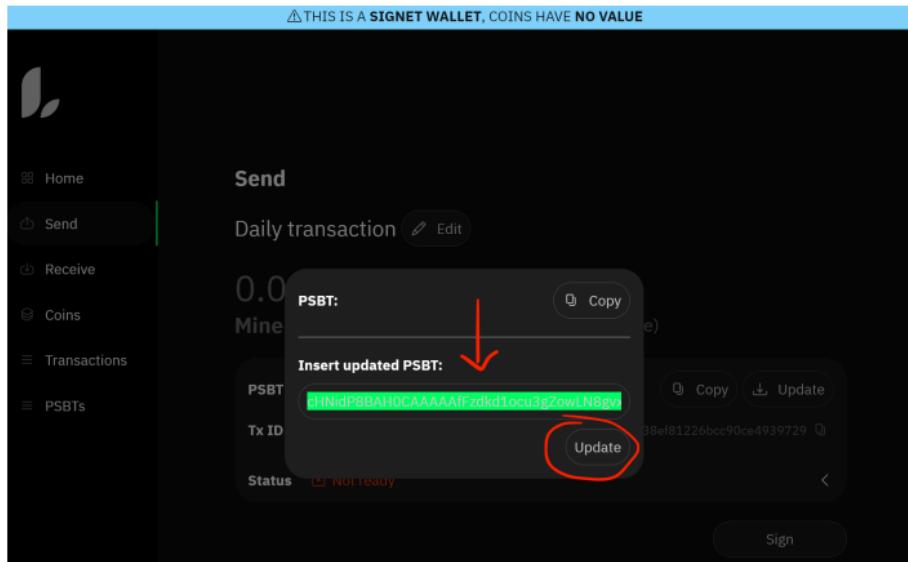


Figure 83: Paste the signed PSBT.

Spend with your key (xv)

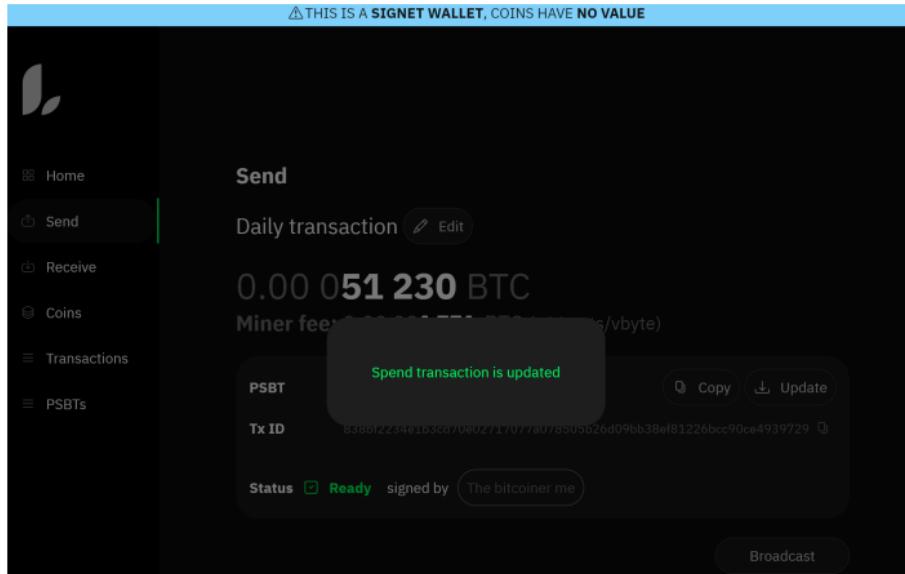


Figure 84: Liana updated to the signed transaction.

Spend with your key (xvi)

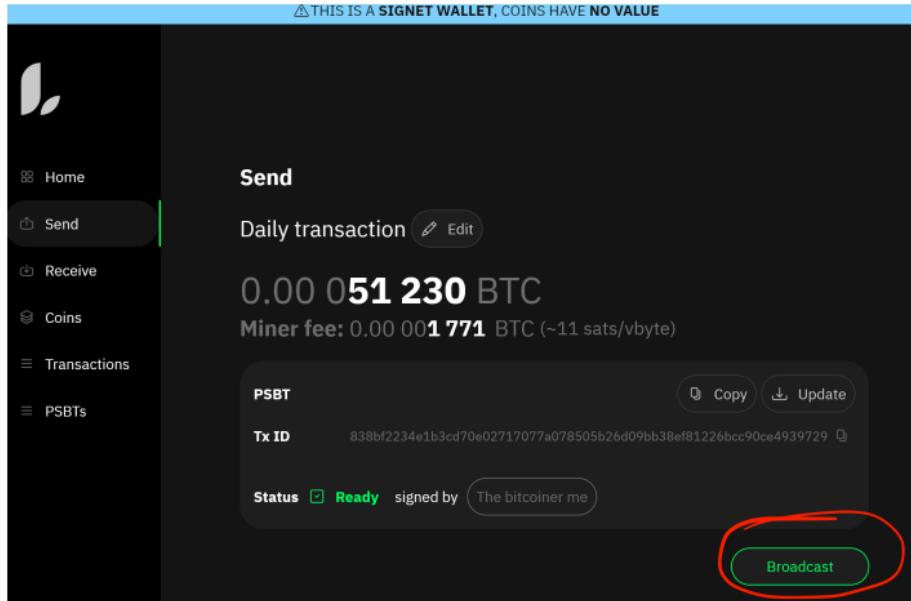


Figure 85: Liana show that the transaction is ready to broadcast.

Spend with your key (xvii)

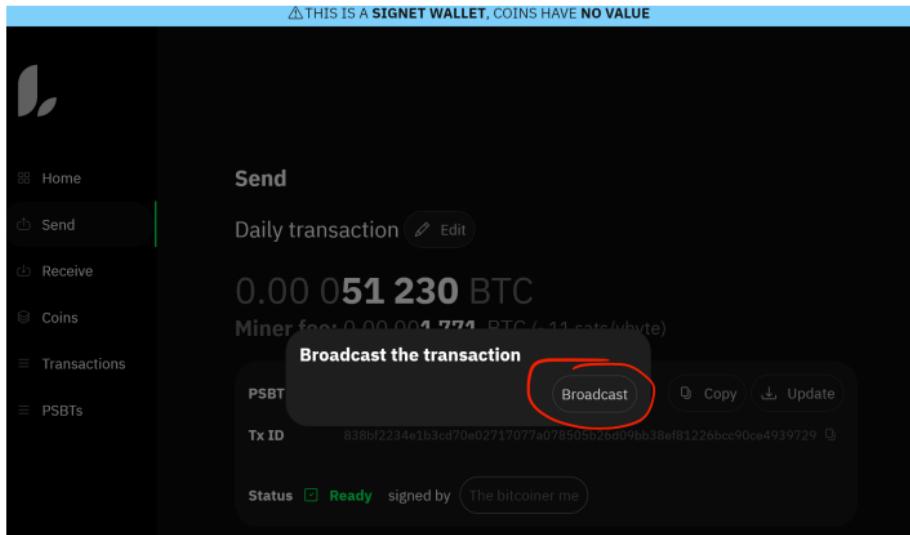


Figure 86: Click on broadcast to broadcast.

Spend with your key (xviii)

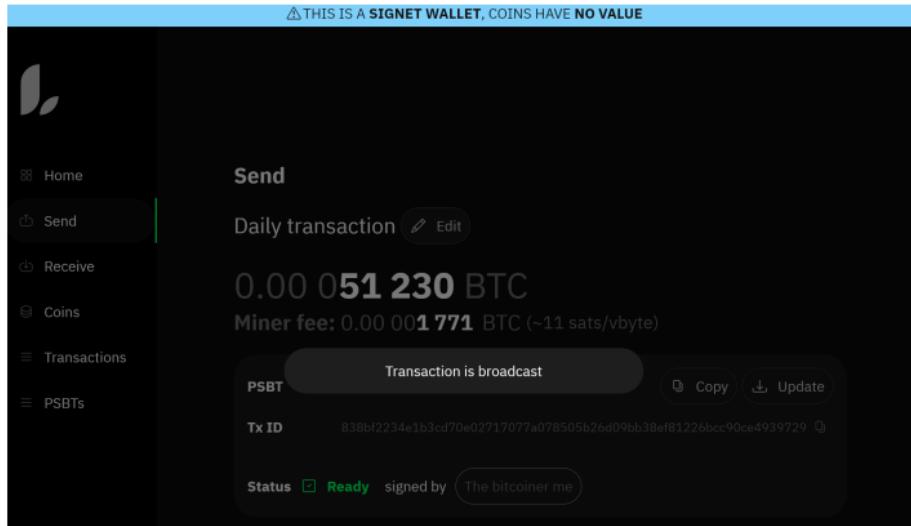


Figure 87: Liana confirmed the broadcast.

Spend with your key (xix)

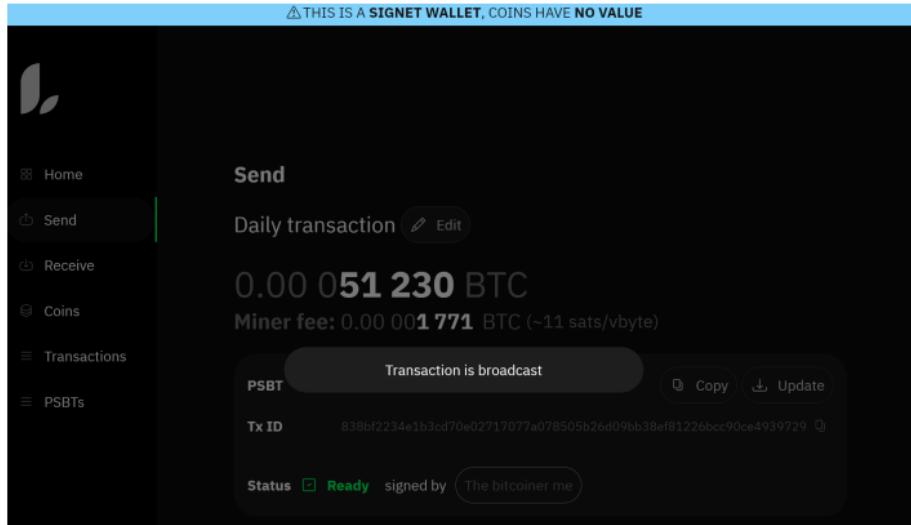


Figure 88: Liana now update wallet. Congrats!

Exercise 3: the game

Exercise 3: the game

The Workshop teacher added you as his heir and he previously added an encrypted key with a double-mnemonic. You must:

- decrypt the key;
- find which mnemonic is the correct;
- find-out the passphrase;
- index of the derivation path.
- spend the signet sats before everyone!

Thanks!

Bibliography

- [1] Bitcoin Improvement Proposals, “BIP 379: Miniscript Policy.” [Online]. Available: <https://github.com/bitcoin/bips/blob/master/bip-0379.md>
- [2] Bitcoin FAQ, “Script.” [Online]. Available: <https://en.bitcoin.it/wiki/Script>
- [3] A. M. Antonopoulos and D. A. Harding, “Mastering Bitcoin: Programming the Open Blockchain (Third Edition).” [Online]. Available: <https://github.com/bitcoinbook/bitcoinbook>