Jade DIY

Costruisci il tuo Jade!

👤 Valerio Vaccaro

Spazio 21

📅 2025-10-24

👤 Valerio Vaccaro 👀

- 💻 Sviluppatore Bitcoin ed Esperto Hardware
- 🔥 Contributore a progetti Bitcoin open source
- ⚠️ Appassionato di hardware fai-da-te (DIY)
- ₿ Ingegnere Bitcoin e Liquid presso Blockstream

👤 https://www.linkedin.com/in/valeriovaccaro/

🐙 https://github.com/valerio-vaccaro/

Meme

⚠️ **Not your keys, not your Bitcoin!**

Licenza

May this presentation inspire you to become more self-sovereign!

Sommario

- 🔒 Cos'è un Hardware Wallet? (funzionalità principali, esempi, ecc.)
- 💻 Jade
- 🔥 Jade DIY (costruisci il tuo Jade)
- 🤖 Test semplice
- 💀 Argomenti avanzati 🚀
- 👀 Domande?

🔒 Cos'è un Hardware Wallet?

- **Un dispositivo dedicato progettato per custodire in modo sicuro le tue chiavi private Bitcoin**
- **Firma le transazioni in sicurezza sul dispositivo:** le tue chiavi private non escono mai dall’hardware, e puoi sempre verificare cosa stai firmando
- Può essere collegato a computer o smartphone, ma i segreti non vengono mai esposti
- Aggiunge un livello di sicurezza e controllo aggiuntivo ai tuoi fondi
- Protegge da malware, attacchi remoti e phishing
- 🔥 Rende la custodia autonoma dei Bitcoin sia **pratica che sicura**

Cos'è un hardware wallet?

- **Conserva in sicurezza** le tue chiavi private Bitcoin e le mantiene isolate dal tuo computer o telefono
- **Firma** le transazioni, permettendoti di controllare i dettagli direttamente sul display del dispositivo
- Permette di generare **chiavi pubbliche** e ricevere **indirizzi Bitcoin**
- Supporta la creazione di frasi di backup (mnemonici) con l’opzione di maggiore sicurezza tramite entropia esterna
- Aggiunge un ulteriore livello di protezione contro malware, attacchi remoti e phishing
- Rende la custodia autonoma dei Bitcoin pratica, sicura e accessibile a tutti

🔒 Cosa NON è un hardware wallet?

- **Non è una soluzione di backup** per i mnemonici; sei tu che devi gestire i backup
- **Non crea le transazioni**; per quello utilizzi un wallet software
- **Non è uno strumento di gestione portafoglio**; NON calcola i saldi né tiene traccia delle transazioni—questo è fatto dal software del wallet
- ⚠️ Puoi generare mnemonici sul dispositivo ma ...

🔒 Entropia

- 🎲 **Entropia**: misura della casualità/imprevedibilità
- Utilizzata per generare:
  - Frasi mnemoniche (BIP39)
  - Nonce
  - Output casuali, casualità nelle GUI e altro ancora
- ⚠️ La sicurezza dipende dall'entropia: Entropia debole = chiavi deboli, entropia prevedibile = chiavi prevedibili

🔒 Entropia

🔧 Fonti Hardware

- 🖥️ Generatore di numeri casuali hardware (TRNG)
- 🌡️ Sensori di temperatura
- ⚡ Rumore elettronico e radio
- 🕐 Jitter degli orologi

🖱️ Fonti Utente

- 👆 Movimenti nell’interfaccia grafica
- ⌨️ Tempi di arrivo dei messaggi
- 🕐 Intervalli nella pressione dei tasti
- 📷 Input della fotocamera

Ma come facciamo a sapere se l'entropia è veramente casuale?

Analisi qualità entropia

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import os

with open('random.bin', 'wb') as file:
    for i in range(1,1024*1024):
        seed = os.urandom(32)
        file.write(seed)
```

In [2]:
```
from hashlib import sha256

seed = 'pippo'.encode('utf-8')

with open('random2.bin', 'wb') as file:
    for i in range(1,1024*1024):
        seed = sha256(seed).digest()
        file.write(seed)
```

In [ ]:

Analisi qualità entropia

In [1]:
```
cat bible.txt | ent
```

```
Entropy = 4.596758 bits per byte.

Optimum compression would reduce the size
of this 4451368 byte file by 42 percent.

Chi square distribution for 4451368 samples is 68853960.24, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 86.2069 (127.5 = random).
Monte Carlo value for Pi is 4.000000000 (error 27.32 percent).
Serial correlation coefficient is 0.109292 (totally uncorrelated = 0.0).
```

In [2]:
```
cat random.bin | ent
```

```
Entropy = 7.999995 bits per byte.

Optimum compression would reduce the size
of this 33554400 byte file by 0 percent.

Chi square distribution for 33554400 samples is 249.31, and randomly
would exceed this value 58.87 percent of the times.

Arithmetic mean value of data bytes is 127.4935 (127.5 = random).
Monte Carlo value for Pi is 3.142053501 (error 0.01 percent).
Serial correlation coefficient is 0.000032 (totally uncorrelated = 0.0).
```

In [3]:
```
cat random2.bin | ent
```

```
Entropy = 7.999995 bits per byte.

Optimum compression would reduce the size
of this 33554400 byte file by 0 percent.

Chi square distribution for 33554400 samples is 252.71, and randomly
would exceed this value 52.87 percent of the times.

Arithmetic mean value of data bytes is 127.4897 (127.5 = random).
Monte Carlo value for Pi is 3.142155783 (error 0.02 percent).
Serial correlation coefficient is -0.000041 (totally uncorrelated = 0.0).
```

In [ ]:
```

```

🔒 Secure Elements

Alcuni hardware wallet includono un chip Secure Element (SE), altri no. Cos’è?

✅ Vantaggi:

- Chip hardware dedicati con resistenza fisica alle manomissioni
- Conservazione sicura delle chiavi crittografiche
- Operazioni crittografiche in ambiente isolato
- Protezione contro attacchi side-channel

❌ Ma ci sono anche svantaggi:

- Spesso chip closed-source
- Possibili backdoor/entropia non controllabile
- Costo aggiuntivo

Hardware Wallet & Secure Element

Gli hardware wallet possono includere (dal meno al più sicuro):

- 0 secure element
- 1 secure element
- Più di 1 secure element
- Secure element software e open source (come Jade)

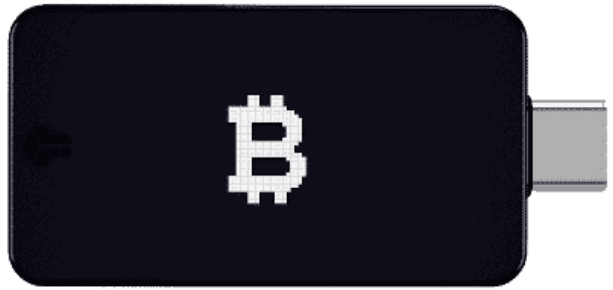
E poi, ci sono dispositivi che non memorizzano nulla... (stateless)

Esempi: Jade

- Hardware wallet open source sviluppato da Blockstream
- Supporta Bitcoin e Liquid Network
- Compatibile USB-C e Bluetooth
- Ampio schermo a colori, supporto QR code
- Progettato per privacy e uso air-gapped
- Ampia documentazione DIY

Esempi: BitBox02

- Hardware wallet open source di Shift Crypto
- Focalizzato su Bitcoin e buone pratiche di sicurezza
- Slider touch per PIN e navigazione
- Architettura MCU + secure element (interfaccia e codice totalmente open)
- Backup su MicroSD



Esempi: SeedSigner

- Hardware wallet Bitcoin fai-da-te completamente open source
- Usa componenti standard (Raspberry Pi Zero, fotocamera, display)
- Nessun secure chip dedicato; design “stateless”—nessun segreto memorizzato sul dispositivo
- Firma tramite QR code con fotocamera
- Obiettivo: massima trasparenza e hardware accessibile a basso costo
- Ideale per cold storage air-gapped e setup multisig

💻 Jade: Introduzione

- **Jade** è l'hardware wallet open source di Blockstream per Bitcoin e Liquid.
- Pensato per trasparenza, privacy e controllo totale dell’utente.
- DIY-friendly: si può assemblare secondo istruzioni open source o acquistare già pronto.
- Documentazione ricca per utenti, smanettoni e sviluppatori.
- Supportato da molti wallet e librerie software.

💻 Jade: Funzionalità Hardware Sicure

- Ampio display a colori ad alta risoluzione, chiaro per revisione transazioni e prompt
- USB-C e Bluetooth per massima flessibilità
- Fotocamere e QR per l’operatività completamente air-gapped
- Aggiornamenti/firma anche via chiavetta USB (solo Plus)

💻 Jade: Open Source e Auditabile

- Firmware e hardware design 100% open source
- Build riproducibili: verifica che i binari corrispondano al codice ispezionabile pubblicamente
- Supporta Bitcoin e asset Liquid nativamente
- Forte comunità di maker supporta nuove board

💻 Jade: Backup, Recupero e Compatibilità

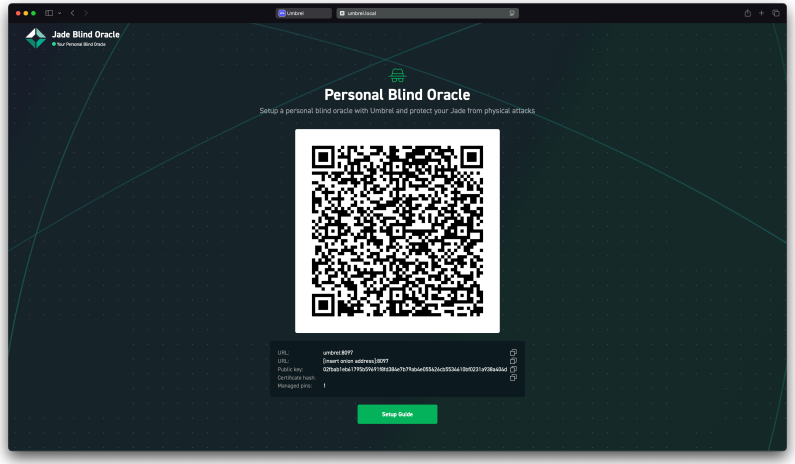
- Backup mnemonic standard a 12 o 24 parole
- Supporto passphrase (BIP39) per “plausible deniability”
- Supporta mnemonici con entropia esterna
- Onboarding, recupero e backup guidati semplici (QR/on-screen keyboard)
- Compatibile con i più diffusi software wallet (BlueWallet, Sparrow, Specter, Electrum, Green, ecc.)

💻 Jade: Funzioni Avanzate & Multisig

- Completo supporto ai workflow multisig e custodia collaborativa (istituzioni e gruppi)
- Implementa BIP85
- Firmware frequentemente aggiornato per supportare le ultime novità Bitcoin
- Integrazione con **Anti-Exfil** per evitare firme malevole

💻 Jade + Blind PIN Server

- Supporta **Blind PIN Server** (o Blind Oracle)
- Un blind oracle funziona come “secure element virtuale”: mantiene il meccanismo di decrittazione fuori dal dispositivo, rendendo Jade invulnerabile all’estrazione fisica della chiave. Diversamente dai dispositivi secure element classici, che tengono tutto sul device stesso.
- Di default, Jade comunica con il blind oracle di Blockstream; puoi anche usarne uno tuo.



💻 Open Source

Perché l’Open Source è importante:

- 🔍 Codice auditabile da chiunque
- 🐛 Vulnerabilità trovate più velocemente
- 👥 Contributi dalla comunità
- 🛡️ Nessun backdoor nascosto

Rischi del Closed Source:

- ❓ Pratiche di sicurezza sconosciute
- 🕵️ Funzionalità nascoste
- 🔒 Vendor lock-in
- 💣 Vulnerabilità agli attacchi supply chain

⚠️ **Ma open source non basta da solo.**

💻 Build Riproducibili 🚀

Le build riproducibili sono un processo in cui lo stesso codice sorgente, compilato nello stesso ambiente, produce sempre lo stesso output binario. Questo permette a chiunque di verificare che un certo binario sia stato creato da codice open e non modificato.

✅ Benefici

- Sicurezza della supply chain
- Verifica delle build
- Validazione della comunità
- Rilevamento di malware

⚠️ **Ma ottenere build riproducibili richiede molto impegno.**

Jade DIY

Lo stesso firmware può girare sulle seguenti board:

- M5Stack Core Gray/Black/Fire e anche Core 2/S3
- M5StickC-Plus e Plus 2
- LilyGO TTGO T-Display
- LilyGO T-Display S3
- LilyGO T-Display S3 Pro Camera
- LilyGO t-Watch S3
- LilyGO T-Camera Plus
- ESP32-cam
- ...e molte altre.

Trovi il firmware di test in questa pagina.

LilyGO T-Display-S3 Pro Camera

Prepara il sistema

Questo tutorial è stato sviluppato su una Debian 13 AMD64. Può funzionare allo stesso modo anche su altri sistemi Linux e su macOS.

```
sudo apt update
sudo apt upgrade
sudo apt install git python3-pip python3-venv cmake
```

Hai i privilegi? Se no, come root...

```
/sbin/adduser my_user dialout  # per accedere alle seriali
/sbin/adduser my_user docker   # per usare docker
/sbin/adduser my_user sudo     # per sudo, se serve
```

Clona il codice

Puoi clonare il codice dal repository:

```
git clone --recursive https://github.com/Blockstream/Jade.git
cd $HOME/Jade
```

Per installare l'ultima versione stabile, puoi prendere l'ultimo tag disponibile, nel nostro caso 1.0.36! (Controlla con `git tag`.)

```
git checkout 1.0.36
git submodule update --init --recursive
```

Struttura del codice

Il codice è scritto in C con toolchain e librerie ESP-IDF. La struttura è divisa in varie cartelle:

- **configs**: File di configurazione SDK (sdkconfig) per la compilazione di Jade su vari hardware e setup di test.
- **components**: Componenti custom ESP-IDF usati dal build system di Jade. Estendono il framework base con funzionalità specifiche Jade come update/diff firmware.
- **diy**: Adattamenti hardware DIY per Jade.

Struttura del codice

- **main**: Cartella principale del firmware Jade, punto di ingresso ESP-IDF. Contiene i sorgenti C/C++, header e script di build per la logica wallet.
- **tools**: Script di utilità e tool Python per preparazione, build e deployment firmware.

È anche offerta una libreria Python per interagire con Jade.

🔥 Usare Docker

Dalla repo clonata, costruisci le immagini e accedi alla shell.

```
docker-compose up -d
docker-compose exec dev bash
```

Copia la configurazione e flasha la tua board.

```
cp configs/sdkconfig_jade.defaults sdkconfig.defaults
idf.py flash --port /dev/ttyACM0
```

⚠️ Veloce, ma serve Docker!

🔥 Installa l’ambiente

Installa le dipendenze:

```
mkdir ~/esp
cd ~/esp
git clone -b v5.4 --recursive https://github.com/espressif/esp-idf.git
cd ~/esp/esp-idf
git checkout 67c1de1eebe095d554d281952fde63c16ee2dca0
./install.sh --enable-gdbgui esp32 esp32s3
python ~/esp/esp-idf/tools/idf_tools.py install qemu-xtensa
```

Trovi branch e commit nel README di progetto e nel Dockerfile.

⚠️ Ogni volta che ti servono i comandi esp, esegui:

```
. $HOME/esp/esp-idf/export.sh
```

Configura

```
cp configs/sdkconfig_display_ttgo_tdisplays3procamera.defaults sdkconfig.defaults
```

Aggiungi (se manca) per silenziare i messaggi debug:

```
CONFIG_LOG_DEFAULT_LEVEL_NONE=y
```

e rimuovi

```
CONFIG_DEBUG_MODE=y
```

Oppure puoi usare

```
./tools/mkdefaults.py ./configs/sdkconfig_display_ttgo_tdisplays3procamera.defaults NDEBUG
```

🔥 Build/Flash

Ora puoi finalmente compilare il firmware e flashare il device.

```
idf.py flash --port /dev/ttyACM0
```

Se non funziona, controlla di avere i privilegi sulla seriale (ad es. aggiungiti al gruppo dialout).

Test semplice: Introduzione

Sparrow Wallet è un potente wallet Bitcoin per desktop, ideale per chi cerca privacy, sicurezza e versatilità:

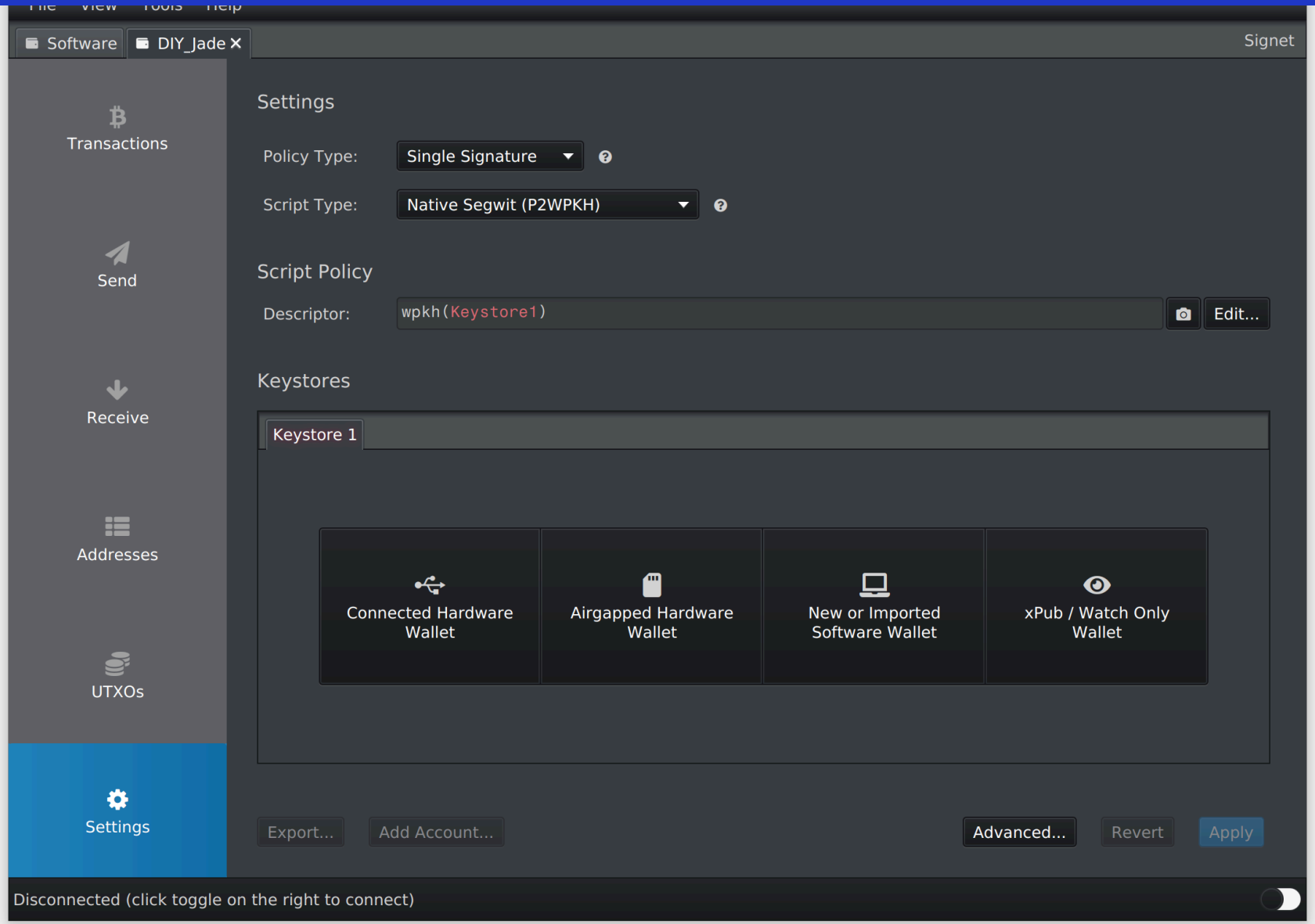
- **Open Source & focalizzato su auto-sovranità**
- **Supporta hardware wallet airgapped**: inclusi dispositivi DIY come Jade, Specter, Passport
- **Funzionalità avanzate**: multisig, coin control, script personalizzati, PSBT
- **Funziona su Testnet, Signet e Mainnet**
- **Ottima Interfaccia**: UI intuitiva per gestione indirizzi, UTXO, coin selection

Test semplice: Introduzione

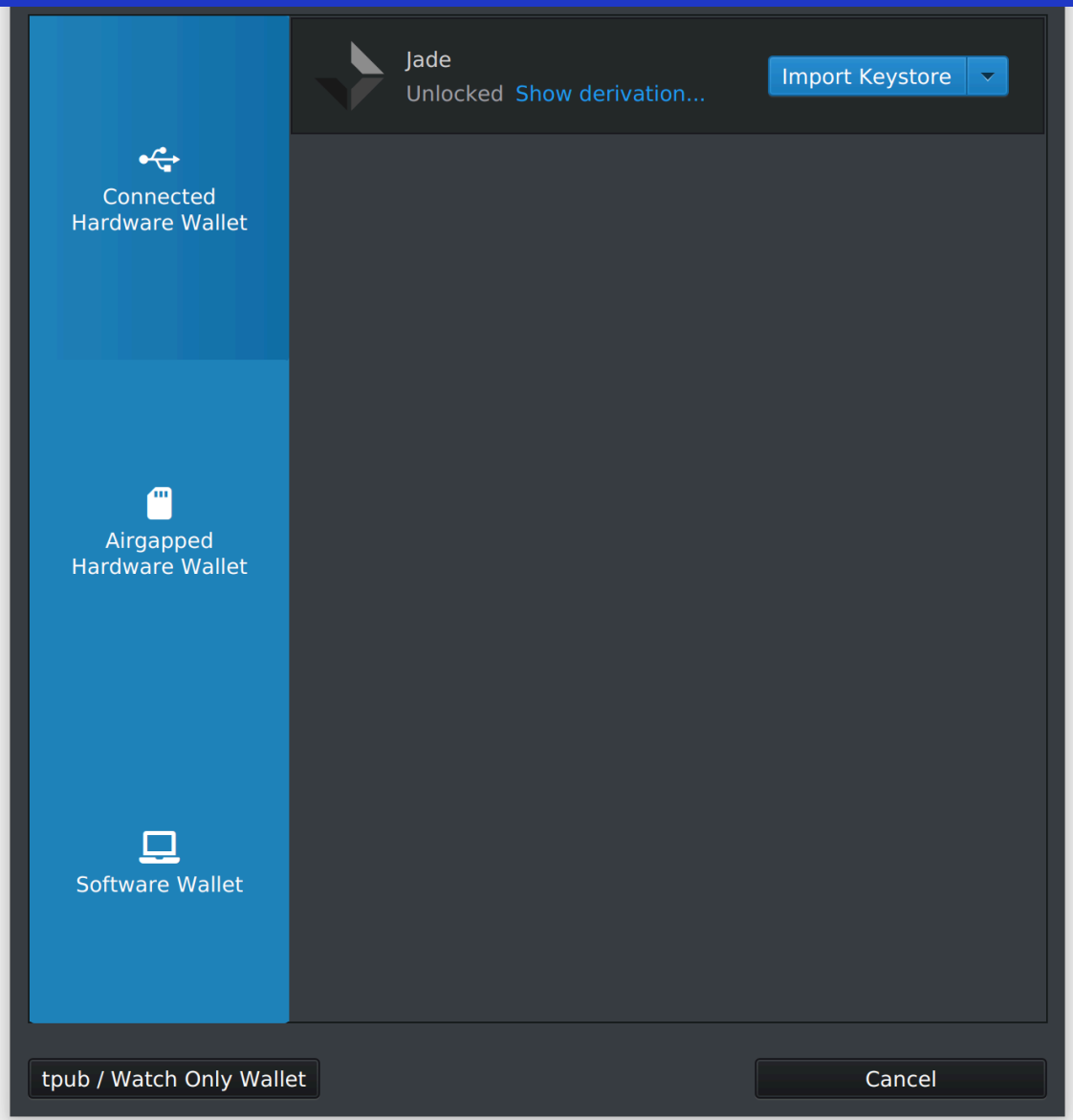
Signet è una rete pubblica di test Bitcoin, pensata per sperimentare in sicurezza senza rischiare veri bitcoin:

- **“Bitcoin finti”** vengono usati su signet—senza valore, ottenibili gratis
- **Più sicuro per test**: a differenza di testnet, i blocchi su signet sono firmati e affidabili, meno spam e instabilità
- **Stesse funzionalità di mainnet**: puoi provare software/dispositivi Bitcoin in uno scenario simile al mainnet
- **Perfetto per sviluppo wallet, test firmware, giocare con nuovi strumenti**

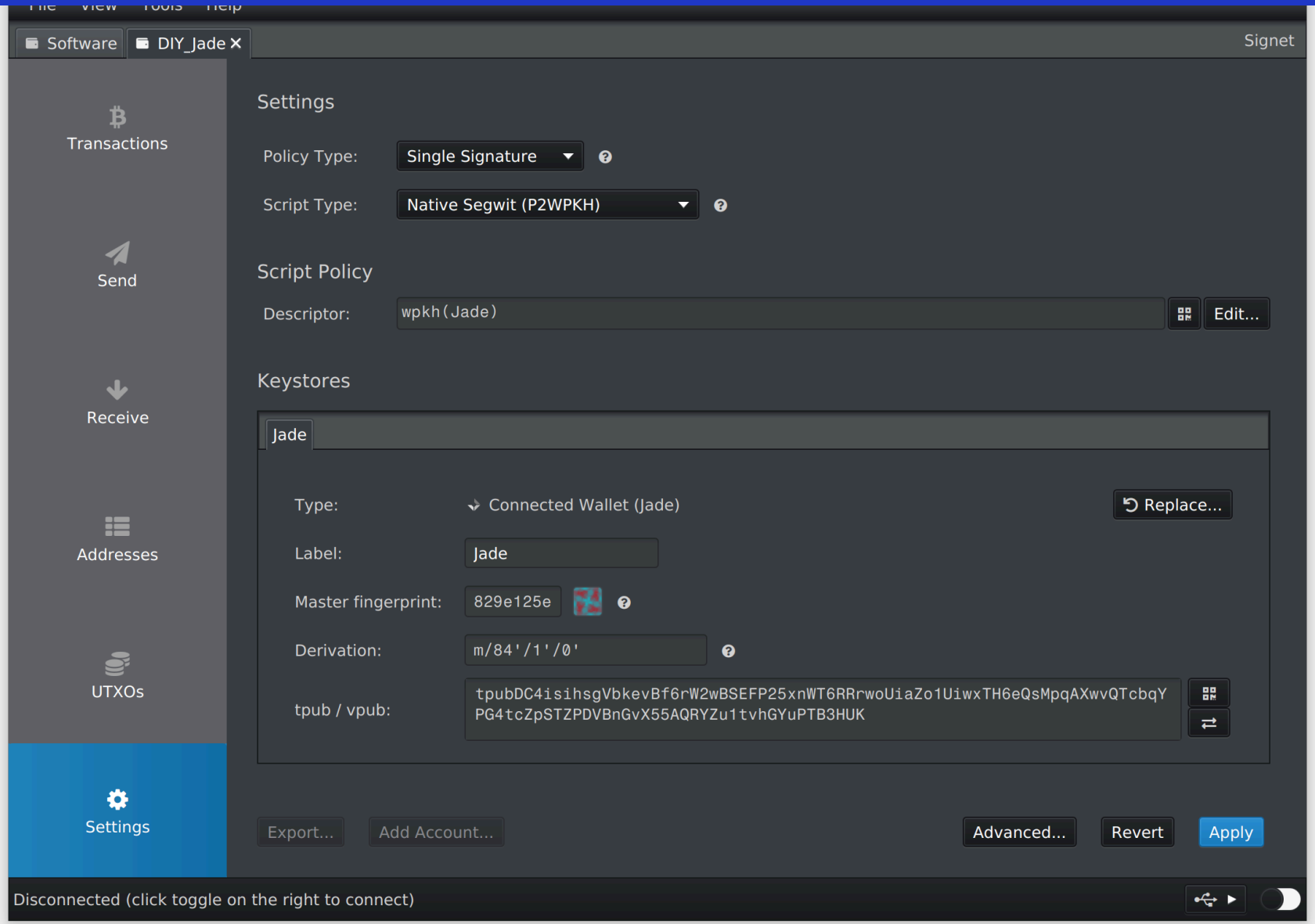
Test semplice: Carica mnemonico



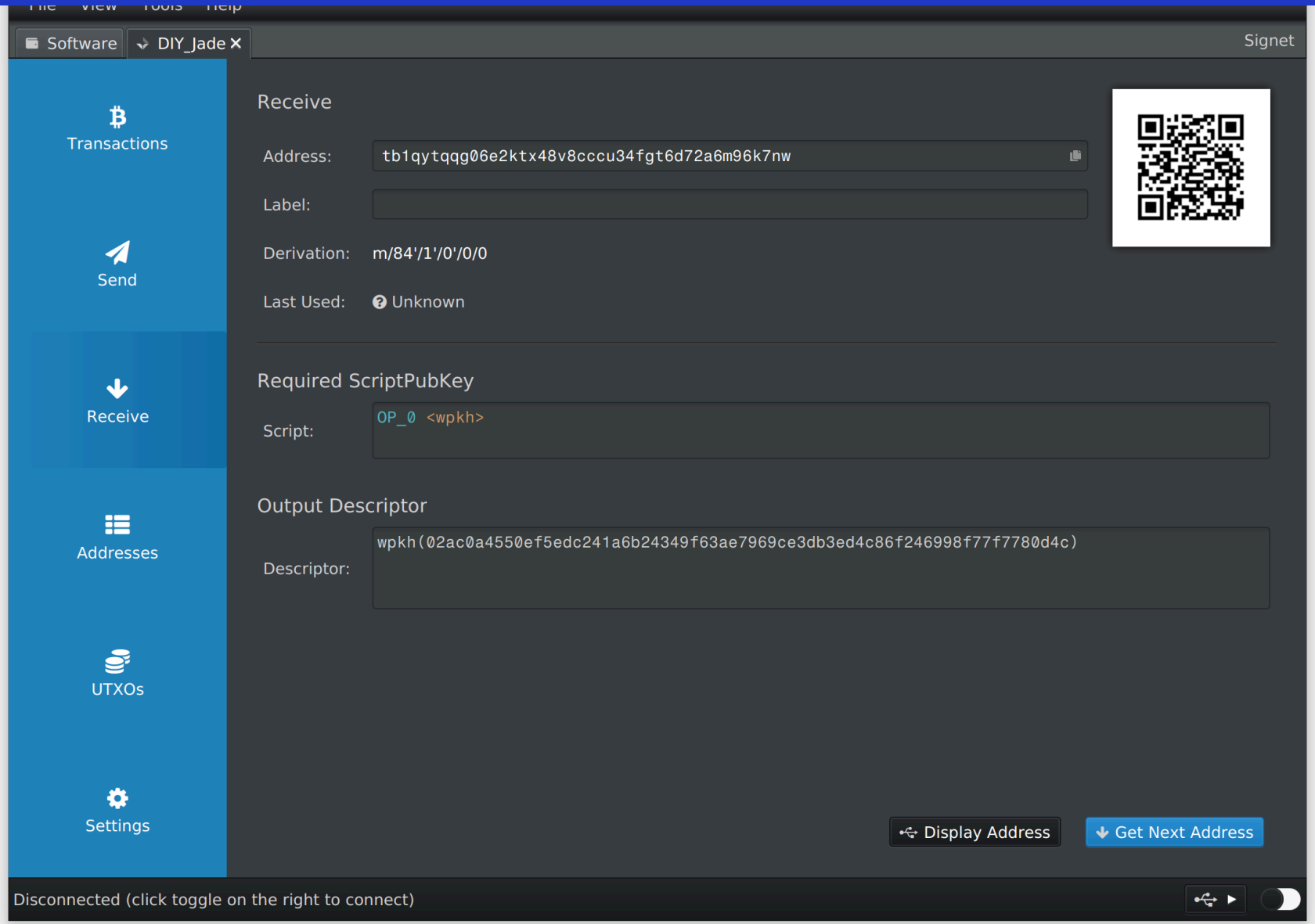
Test semplice: Crea nuovo firmatario



Test semplice: Crea wallet



Test semplice: Ricevi



Test semplice: Spendere

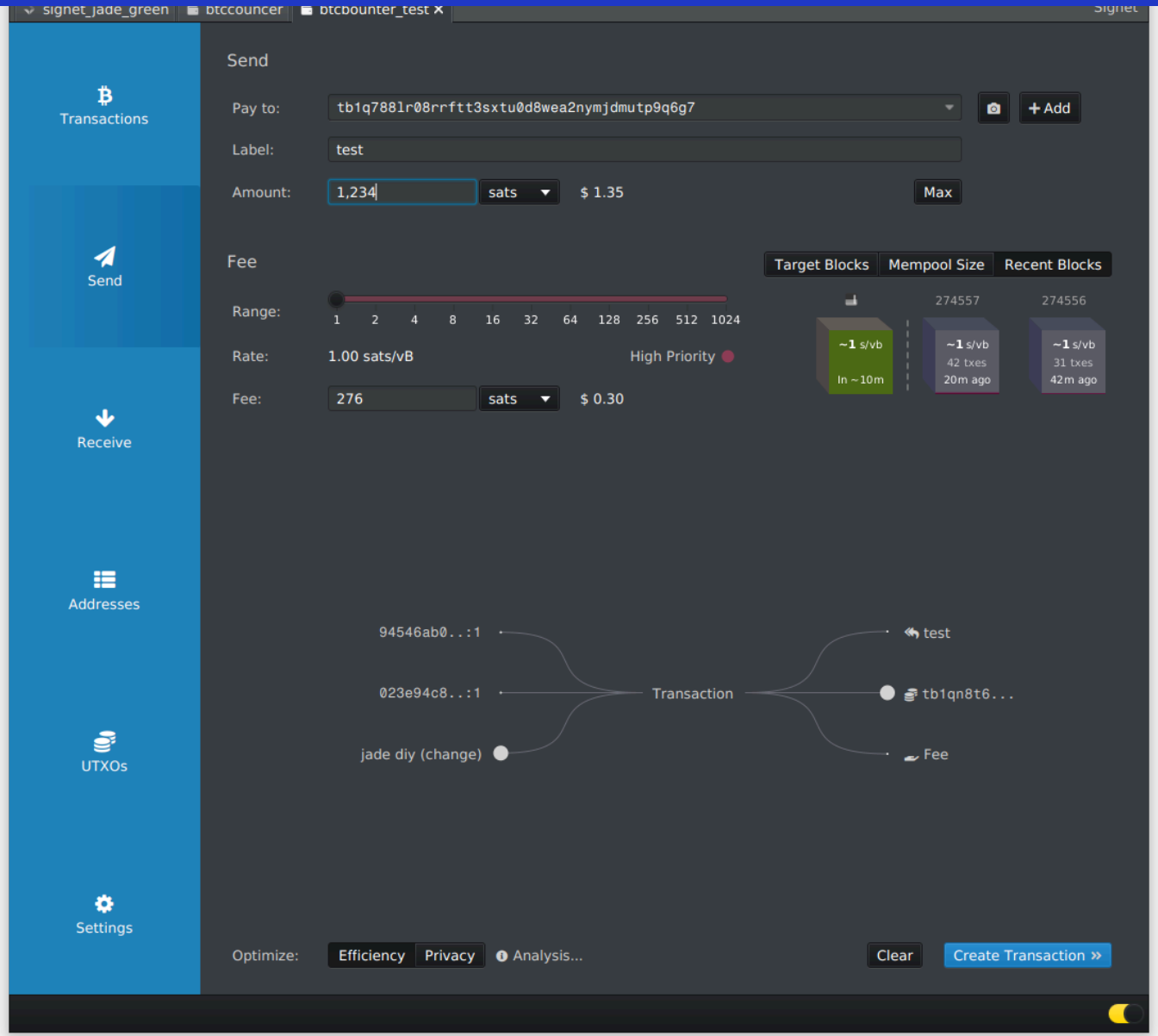
Esercizio:

- Crea una transazione che invii 1234 sats a un indirizzo.
- Firma la transazione usando Jade e controlla le informazioni fornite.
- Trasmetti la transazione firmata.

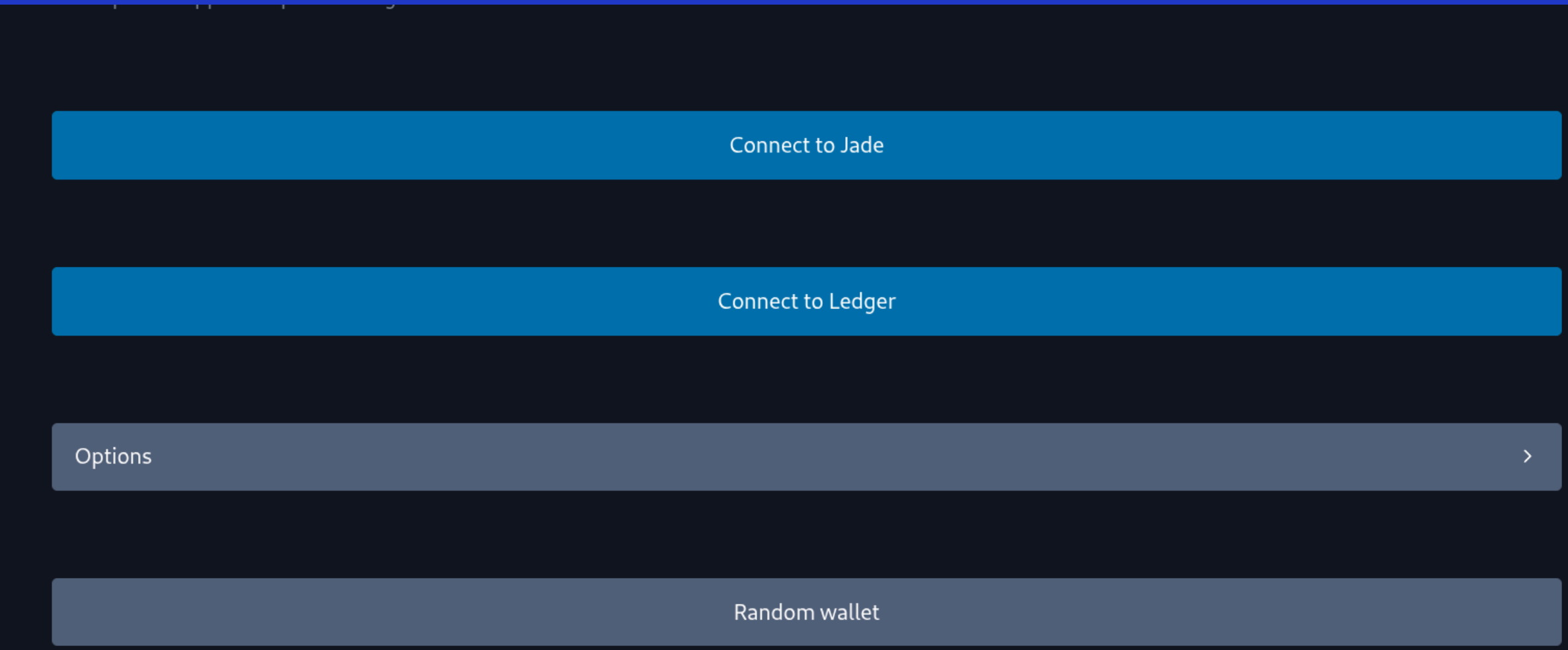
Esercizio 2:

- Crea una transazione con più output.
- Firma la transazione usando Jade e controlla le informazioni fornite.
- Trasmetti la transazione firmata.

Test semplice: Spendere



Test semplice: Liquid Web Wallet 🚀



💀 Bootloader 🚀

Sugli hardware wallet DIY senza un bootloader installato in modo sicuro, esiste un grave rischio chiamato **firmware swap attack**:

- Un attaccante può collegarsi via USB/seriale e **sostituire il firmware** con una versione malevola che esporta chiavi, salta controlli di sicurezza o usa indirizzi errati.
- **Il device parte con il firmware malevolo**, spesso indistinguibile agli occhi dell’utente.
- Così l’attaccante può ricevere i tuoi fondi/informazioni o il device compromesso salva i segreti nella memoria flash.

💀 Bootloader 🚀

Come difendersi:

- Usa secure boot hardware e cifratura flash.
- **Assicurati che il bootloader sia veramente protetto in scrittura** (es: ONE-TIME PROGRAMMABLE/eFuses).
- Quando possibile, usa solo device che permettano di verificare in ogni momento integrità di bootloader e firmware.

Ecco perché la sicurezza del bootloader è *cruciale* nei device DIY!

💀 Bootloader 🚀

Abilitare secure boot e criptare la flash sono passi fondamentali per evitare modifiche non autorizzate al firmware e furti di dati.

Attenzione:

- Conserva la chiave di firma in un luogo sicuro, perderla impedisce aggiornamenti futuri.
- Futuri update firmware (inclusi OTA) chiederanno questa chiave per generare aggiornamenti validi.

Per iniziare, genera la tua chiave di firma:

```
espsecure.py generate_signing_key --version 2 ../jade-diy-v2.pem
```

⚠️ QUESTA CHIAVE È ESTREMAMENTE IMPORTANTE!

💀 Bootloader 🚀

Modifica la configurazione aggiungendo

```
CONFIG_ESP32_DISABLE_BASIC_ROM_CONSOLE=y
CONFIG_SECURE_DISABLE_ROM_DL_MODE=y
CONFIG_SECURE_BOOT_SIGNING_KEY=<PERCORSO_TUA_CHIAVE_FIRMA>
CONFIG_SECURE_BOOT=y
CONFIG_SECURE_FLASH_ENC_ENABLED=y
CONFIG_SECURE_FLASH_ENCRYPTION_MODE_RELEASE=y
CONFIG_ESP32_REV_MIN_3=y
```

e rimuovi

```
CONFIG_ESP32_REV_MIN_1=y
```

Oppure usa

```
./tools/mkdefaults.py ./configs/sdkconfig_display_ttgo_tdisplays3procamera.defaults NDEBUG SECURE
```

💀 Bootloader 🚀

🔨 Ora puoi buildare un bootloader sicuro con la nuova configurazione.

```
idf.py bootloader
```

✍️ Adatta questo comando o controlla i log.

```
esptool.py --chip esp32s3 --port=(PORT) --baud=(BAUD) --before=default_reset --after=no_reset --no-stub write_flash --flash_mode dio --flash_freq 80m --flash_size keep 0x0 ./build/bootloader/bootloader.bin
```

💀 Bootloader 🚀

🔨 Ripeti per buildare il firmware completo.

```
idf.py build
```

✍️ Adatta questo comando o controlla i log.

```
esptool.py --chip esp32s3 -b 460800 --before default_reset --after no_reset --no-stub write_flash --flash_mode dio --flash_size keep --flash_freq 80m 0x8000 build/partition_table/partition-table.bin 0x1a000 build/ota_data_initial.bin 0x20000 build/jade.bin
```

oppure semplicemente

```
idf.py flash
```

💀 Bootloader 🚀

Se hai abilitato secure boot con quelle impostazioni, per aggiornare il firmware dovrai usare OTA.

Ad esempio, usando il jade.bin dalla cartella /build:

```
python jade_ota.py --noagent
```

💀 Blind Pin Server 🚀

Un modello alternativo di sicurezza rispetto ai chip secure element tradizionali.

- Il segreto necessario a decriptare il seed del wallet è conservato su un server “blind oracle”.
- Il server vede solo un hash del PIN utente + una nonce casuale. Non accede mai alle chiavi o agli indirizzi.
- Dopo PIN corretto, il wallet chiede al blind oracle la parte mancante del segreto, tramite canale cifrato (ECDH).
- Ottenuto il segreto, il seed viene sbloccato per autorizzare operazioni.
- Dopo tre PIN errati, sia device sia server cancellano i dati sensibili: senza seed di recupero il wallet è inutilizzabile.
- Gli utenti avanzati possono gestire un proprio server blind oracle, senza dipendere da Blockstream.

💀 Blind Pin Server 🚀

Vantaggi:

- Architettura completamente trasparente e open source
- Elimina costi e limiti dei secure element proprietari
- Forte protezione da attacchi fisici o estrazione chiave
- Possibilità per l’utente di hostare il proprio blind oracle

Già disponibile su Umbrel e come app Docker Compose! Altre integrazioni in arrivo.

💀 Blind Pin Server Protocol v2: Flussi di messaggi 🚀

Il Blind Pin Server Protocol v2 è un modo sicuro e privato per sbloccare i segreti del wallet tramite oracle remoto, senza mai esporre seed o chiavi—nemmeno durante lo sblocco!

Ecco un protocollo semplificato, tutti i messaggi sono cifrati.

💀 Blind Pin Server Protocol v2: Flussi di messaggi 🚀

- **Il client genera una coppia di chiavi ECDH** e stabilisce una sessione TLS col server.
- **SET del PIN** via POST:
  Il device invia:
  - `public_key`: chiave pubblica ephemeral Curve25519 del client
  - `encrypted_secret`: segreto wallet cifrato, decriptabile solo da chi ha PIN + chiave device
  - `pin_hash`: (hash Argon2 di PIN utente e nonce casuale)
- **Risposta server:**
  - `device_id`: identificatore univoco del dispositivo (usato nelle richieste successive)

💀 Blind Pin Server Protocol v2: Flussi di messaggi 🚀

- **GET del PIN** via POST:
- Il client invia:
  - `device_id`
  - `public_key`: chiave ephemeral della sessione
  - `pin`: PIN utente (processato con Argon2id+nonce e inviato come hash, mai come PIN raw)
- **Il server verifica l'hash PIN:**
  - Se corretto e nei tentativi permessi:
    - Ritorna `encrypted_secret` (verrà decriptato dal device con chiavi e PIN)
  - Se errato: incrementa il contatore tentativi.

⚠️ **Dopo 3 PIN sbagliati, il server cancella i dati del wallet.**

Bibliografia

- Codice sorgente Jade
- Codice sorgente Pin server
- Video, Windows-based
- Video, Linux-based

👀 Domande?

Su Telegram: @valeriovaccaro

Progetto Satoshi Spritz

- 💻 Federazione di gruppi locali bitcoiners
- 🎓 Eventi gratuiti e orientati alla privacy
- 🤖 SOLO BITCOIN
- 📖 Formazione sulla self-sovereignty (auto-sovranità)
- 🍕 Ogni settimana, Satoshi Spritz Connect online

https://satoshispritz.it

https://t.me/SatoshiSpritzConnect

₿ Officine Bitcoin

- 🤝 Comunità italiana di Bitcoiners, totalmente gratuita
- 🤖 SOLO BITCOIN
- 🎓 Focus su didattica e sviluppo progetti
- 📋 Progetti:
  - 💼 Sviluppo nodi Bitcoin
  - 👨‍🏫 Uso Hardware Wallet
  - 💻 Filosofia open source
  - 🤝 Installazione Debian
  - ...e molto altro

https://officinebitcoin.it