TREZOR
The Hardware Bitcoin Wallet

Pavol Rusnak
stick@satoshilabs.com

Chaos Communication Congress
30c3, Hamburg
Problem: private keys security/safety

- end user computer security
- compromised computers
- untrusted computers
- rigged clients
- data (wallet) loss
- disasters, hard-drive failures
- naive reinstalls
- failing to do proper backups
Solution

HARDWARE

WALLETS!
Hardware Wallet Ideas

Wishes
Proposals
Results
Real needs
KISS

- USB gadget (HID)
- OLED display
- ok/cancel buttons
- no batteries
- no radio
What's inside?

- ARM Cortex-M3 microcontroller
  - STM32F205
  - 120 MHz
  - 512 KiB Flash
  - 128 KiB RAM
  - HW RNG *
- 128x64 0.96” OLED display
Raspberry Pi

- same OLED display
- USB HID to Serial
- prototyping platform
  - Python
  - rapid development
- follows the same logic
Modus Operandi (1)

- generate initial entropy
- allow its easy backup
- use this entropy to derive master private key and master public key
  „generators“
- send master public key to computer
Modus Operandi (2)

- computer prepares transaction and sends to TREZOR
- (gaps with keys indices instead of signatures)
- TREZOR uses master private key to generate needed private keys from indices
- TREZOR sends signed transaction back to computer
  - which will broadcast it to the network
- private keys never leave the device!
Generate Entropy

- use HW RNG to generate entropy A (e.g. 256 bits)
- request entropy B from computer (e.g. 256 bits)
- use both entropies to generate final entropy while proving that external entropy was used - e.g. \( E = \text{SHA256}(A \ || \ B) \)
- more complex schemas suggested by Timo Hanke & Ilja Gerhardt
Mnemonic code (for backups)

- convert entropy to string of words aka „mnemonic sentence“
  "immense uphold skin recall avoid cricket brush pill next home require friend"

- use entropy directly to generate master private key
Mnemonic code BIP-0039

- convert entropy to string of words aka „mnemonic sentence“
  
  "immense uphold skin recall avoid cricket brush pill next home require friend"

- use entropy directly to generate master private key

- use PBKDF2 to generate master private key
  
  - PRF = HMAC-SHA512
  
  - Password = mnemonic sentence
  
  - Salt = „mnemonic“ || user's secret

  - c = 4096 ; dkLen = 512 bits
Hierarchical Deterministic Wallets

Master Node

WALLETS / ACCOUNTS

-Wallet chains

Addresses

Depth = 0

Depth = 1

Depth = 2

Depth = 3
Hierarchical Deterministic Wallets

- BIP-0032 by Pieter Wuille; CKD uses HMAC-SHA512
- abstract concept, lots of possibilities
  - master node – accounts – chains – addresses
  - master node – cointype – accounts – addresses
  - master node – HQ – local branches – accounts – addresses
  - master node – cryptocurrencies / SSH / FDE / challenge response / etc.
- wallet token => identity token!
ECDSA Signatures

- ECDSA requires random nonce during signing (256-bit for Bitcoin)
- using same nonce twice for signing different messages using the same particular key => leak
- 27c3 fail0verflow: Console Hacking – PS3 hack
- August 2013: Android Java RNG vulnerability in SecureRandom
  - 59+ BTC stolen
Deterministic ECDSA Signatures

- August 2013: RFC 6979 (Java, Go, python-ecdsa since 0.9)
- HMAC_DRBG seeded with private key and message
- great news!
  - avoids problem described in the previous slide
  - enables unit testing of signatures
  - proof that TREZOR does not leak master private key in nonce
Integration

- existing desktop clients
  - Multibit, Electrum, Armory
- mobile clients
- webwallets via native browser plugin
Thank you!

info@bitcointrezor.com

www.bitcointrezor.com

github.com/trezor