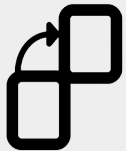
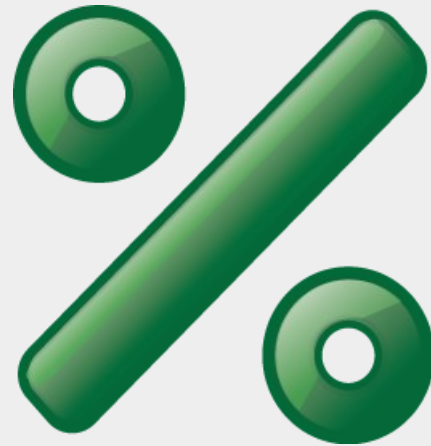


Understanding Offline Wallets...and Building One!



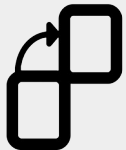
Wallet Usage?

- How many of you have used:
 - A hardware wallet?
 - An old-school paper wallet or raw keypair?



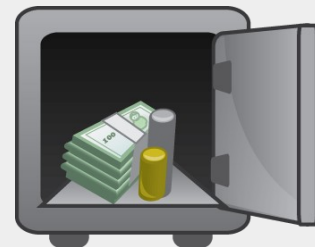
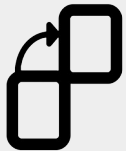
A Little About Me...

- Software Engineer @ Microsoft in Pittsburgh
- Run <https://chaintuts.com> creating cryptocurrency & blockchain related tutorials
 - Articles, videos, and code projects
 - On YouTube, Twitter, Github
 - Support: Patreon, Crypto, Spreadshirt Apparel
- Focus is on understanding & teaching core concepts



First, A Quick Disclaimer

- I'm not a cryptography/security expert, but I am a software engineer
- Security is a Complex, evolving topic
- This is a proof-of-concept for building open hardware, open source cryptocurrency tools

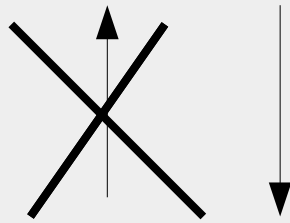


How Addresses are Generated



0x12351bc143badf2348fe38e8f8b785b...

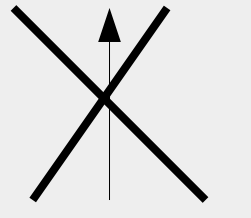
PRIVATE KEY



Elliptic Curve
(secp256k1)

0x04135981abcd7f7a7d7b7c720....

PUBLIC KEY



“Double hash” (SHA-256
and RIPEMD160)
And Base58check encoding

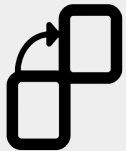
1MT3uNoFLP82j2aSD5Qtibm2kXJ7RWumAM

ADDRESS
(PUBLIC KEY
HASH)



What Wallets Do

- Generate and Secure private keys & addresses
- Scan the blockchain for relevant transactions
 - Full node – downloads the whole chain and validates *all* blocks & transactions (signatures, other protocol rules)
 - SPV wallet – downloads address-relevant data from full nodes and validates some info
 - Some wallets use a 3rd party API to get data



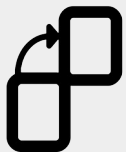
What Wallets Do

- Sign and broadcast new transactions
 - For Bitcoin and other UTXO chains – composes a transaction with spendable “outputs” as new “inputs” to the transaction
 - Sends out transaction to network for validation



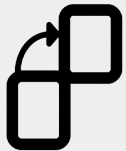
The Importance of Private Keys

- Ultimately, the private keys are the most important bits of data that wallets generate & store
 - Not your keys, not your coins – *anyone* with the keys can spend funds from another wallet
 - All other data like addresses, balances, transactions are *public* blockchain data
 - Critical that keys are both:
 - Securely generated
 - Securely stored



Types of Wallets

- Custodial Wallets – such as Exchanges
- Online Wallets – Bitcoin.com, Coinomi, Bitcoin Core full node, etc.
- Offline Wallets



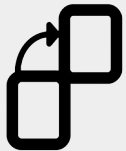
Offline Wallets

- Most common, modern wallets are purpose built *hardware wallets*
 - KeepKey, Trezor, Ledger, etc.
 - Both generate keys and sign transactions
- Simplest form: an encoded keypair
 - Paper wallets
 - Keypairs engraved in metal, etc.



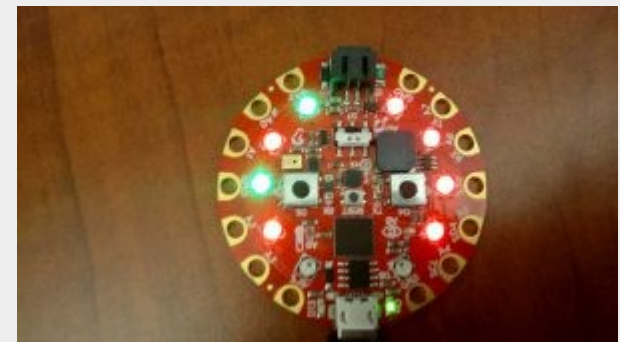
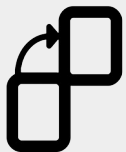
Why Offline Wallets?

- Robust security – not vulnerable to network attacks
 - Keys have to be *physically* stolen
 - Key generation is difficult to observe
 - Can't create/sign a malicious tx
- Resistant to data loss from computing failures (phone in toilet, HDD failure, loss/theft from daily use)



The Idea

- It would be super cool to generate real, offline keys with my own code
- I had some basic microcontroller experience from another project (visualizing proof-of-work)
- Let's build something with readily available, “easy” to code products



Proof-of-work simulator
On the Circuit Playground

The Challenges

- Need a platform with enough memory to store, run code
 - First experiments shows the M0 platform was insufficient
- Cryptographic primitives – need:
 - Hashing algorithms such as SHA-256
 - ECDSA - secp256k1
- MUST have cryptographically secure RNG



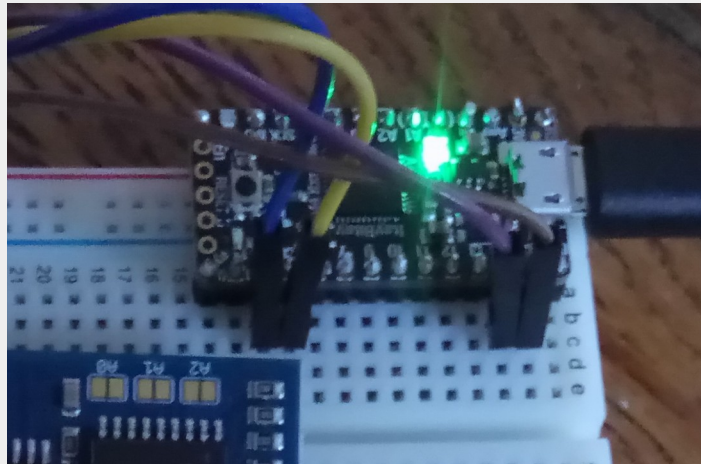
The Challenges

- MUST have cryptographically secure RNG
 - Many “random” occurrences in code are not truly random
 - Games, etc. use “pseudorandom” algorithms
 - Usable, but predictable
 - For cryptography, we MUST use a system that generates true randomness
 - Using environmental noise, etc.

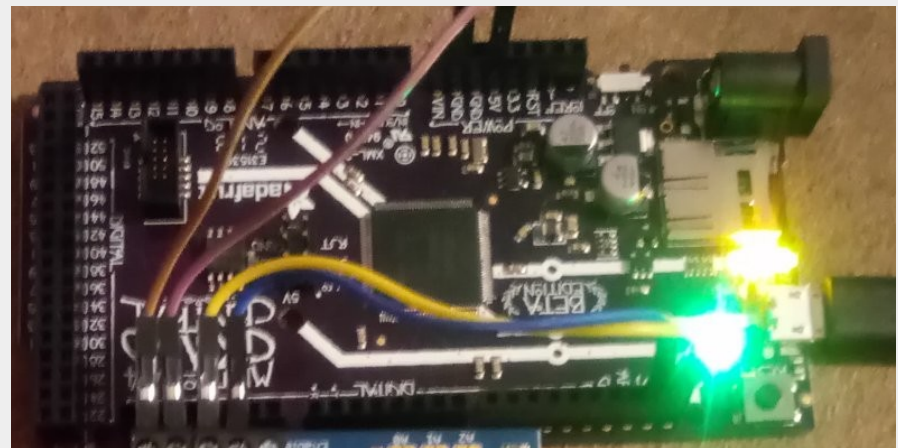


Answering the Challenges

- Memory, power
 - Decided on the Adafruit M4 platforms
 - Grand Central, ItsyBitsy tested



ItsyBitsy M4



Grand Central M4



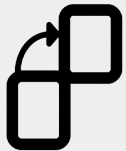
Answering The Challenges

- How to get Crypto Primitives? - Use the Open Source!
 - Standalone Python libs are hard to find and run on micro platform
 - Trezor firmware is open source, standalone C code
 - Ported Trezor code (more on that later)



Answering The Challenges

- Need for true random number generation
 - Platform choice fixed this for us!
 - M4 microcontrollers used all have built in true random number generation built in
 - Other option *could* have been to read data from an accelerometer or used other suitable environmental noise



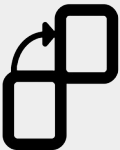
Putting It Together

- The stack:
 - Custom CircuitPython module, C
 - Application, CircuitPython
 - Device: M4 processor with I2C Character LCD screen (mostly used) or a receipt printer
 - Supported currencies: BTC, BCH, ETH, LTC, DGB



Putting It Together

- Custom CircuitPython Module
 - Gets CRNG seed passed in from Python layer
 - Hashes seed into private key, generates pubkey, returns encoded keypair
 - Chain-appropriate address encoding
 - WIF or HEX privkey



```
josh@Josh-Asus: ~/circuitpython/ports/atmel-samd
File Edit View Search Terminal Help
josh@Josh-Asus:~/circuitpython/ports/atmel-samd$
josh@Josh-Asus:~/circuitpython/ports/atmel-samd$
josh@Josh-Asus:~/circuitpython/ports/atmel-samd$ make BOARD=grandcentral_m4_express
Use make V=1, make V=2 or set BUILD_VERBOSE similarly in your environment to increase build verbosity.

717016 bytes free in flash out of 1024000 bytes ( 1000.0 kb ).
244796 bytes free in ram for stack out of 262144 bytes ( 256.0 kb ).

Converting to uf2, output size: 614400, start address: 0x4000
Wrote 614400 bytes to build-grandcentral_m4_express/firmware.uf2.
josh@Josh-Asus:~/circuitpython/ports/atmel-samd$
```

```
josh@Josh-Asus:~/circuitpython/shared-module/bitaddr$ ./test
DSu9C81NQCqmM2X4Ga99ULWarn8yH7TNKV
5JCtUcQv5w1Jj3SEeX1J8PpM2ByWPxzTexXHXoBdFoYKEWEERoi
```

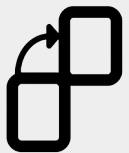
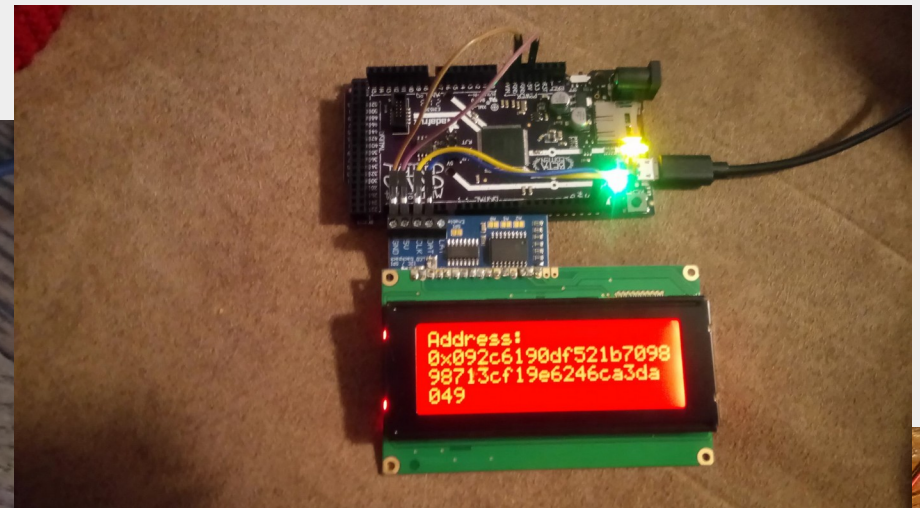
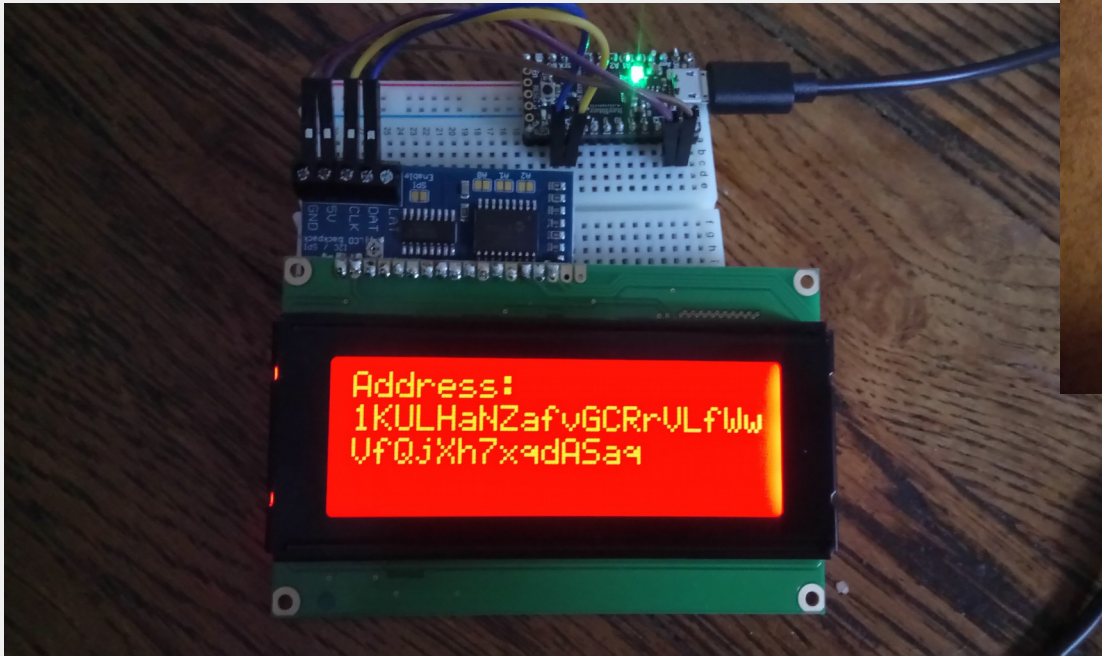
Putting It Together

- CircuitPython layer
 - Generates entropy (Python `os.urandom`, which uses the built in CS RNG)
 - Calls custom module code to get privkey, address
 - Displays on LCD screen or prints to receipt printer, etc. using Adafruit libraries



uBitAddr

<https://github.com/chaintuts/ubitaddr/tree/development>



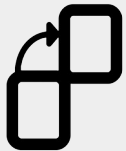
Potential Pitfalls

- Single keypairs are out-of-favor/legacy
 - MUST make sure to copy correctly, double check before sending funds
 - Encoding is not human friendly
 - Data easily corrupted (water damage, bad handwriting, character distinctions)
 - Change can cause issues



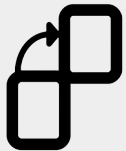
Potential Pitfalls

- UI/UX
 - Code only shows keypair once (generates a new one on restart)
 - Doesn't provide any security recommendations
- Cryptography foot-guns
 - Used trusted implementations and manually tested to mitigate this



Potential Future Improvements

- Biggest: Upgrade to BIP39/BIP44 compatible seed phrases instead of single keypairs
 - Give user a seed phrase for safe backup
 - Generate new addresses as needed
 - Even if we use it for one address from tree (similar to old impl), backup is easier and safer
- UI/UX improvements
 - Warnings & security reqs, store keypairs?, etc.



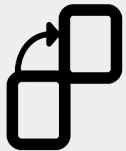
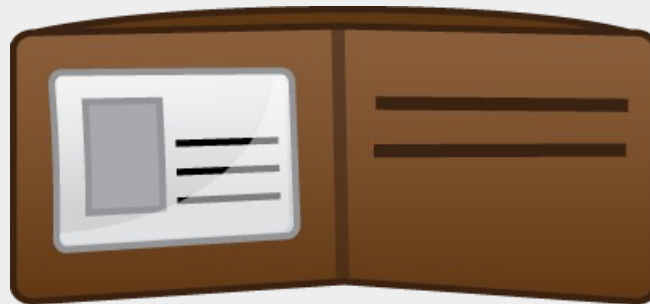
Potential Future Improvements

- From a code perspective:
 - Update module/firmware code for CP5
 - Move to pure Python implementation?
 - Unit testing to catch regressions and edge cases
- Open to ideas – always trying to improve!



The Importance of Offline Wallets

- As Andreas A. says – Not your keys, not your coins!!
- Keys must be safely generated and stored offline for online storage
- Can this be done with open hardware & open source – this experiment says yes!!



Questions?

