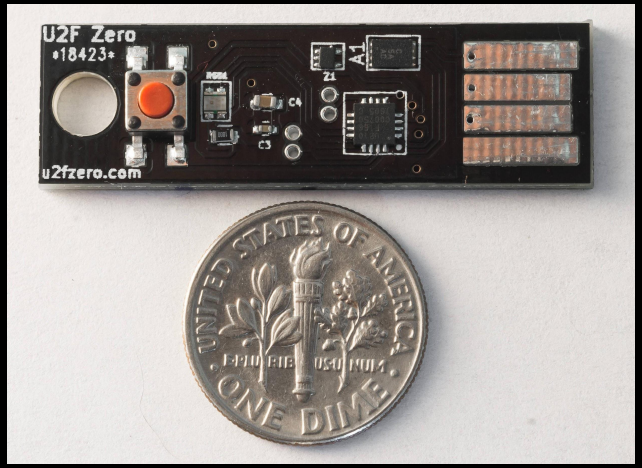
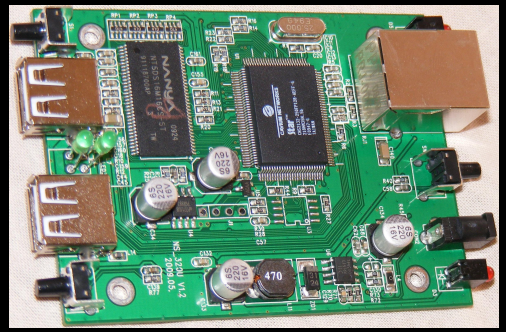
U2F Zero: a story on (cheap, open source, secure) authentication



Conor Patrick (twitter: @_conorpp)

How it starts

- Started in the AMP Lab
- Always wanted to make one of these ----->



- Got sucked into “cyber”
  - Learning more about secure embedded systems
    - *IOT is cool right*?
  - U2F, secure crypto
  - Started using 2FA after Defcon
  - Started reading 2FA protocols

- U2F Zero: my first engineering project?
  - Today’s goal: learn about design & implementation

What I'll cover

- Authentication
  - Challenge response
  - U2F protocol
- Secure design
  - Requirements for an authentication key
  - Importance of being random
  - Crypto requirements, key storage
- Implementation
  - Picking out the parts
  - Making a PCB
  - Software design
  - Cost overview
- Next steps
- Demo & trivia

Authentication

Most of you are familiar with this

pass0Rdh@ck3r

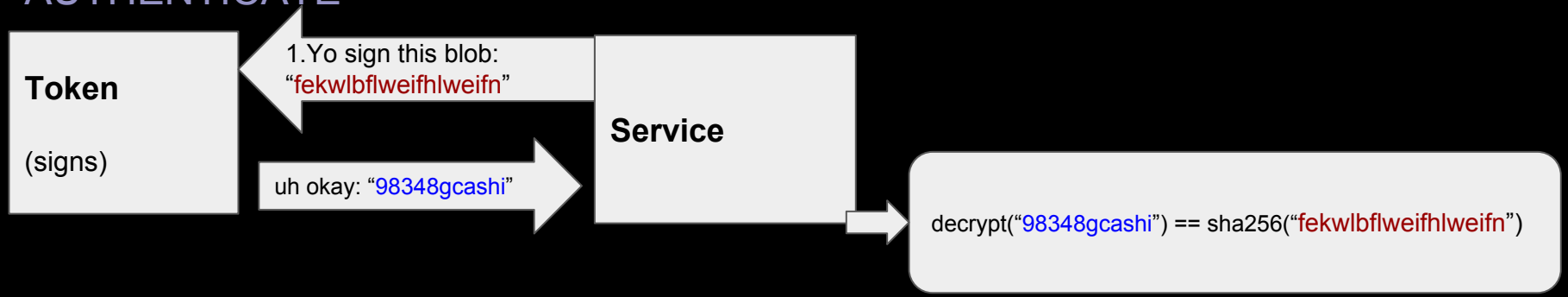
1. Something you know
2. Something you have
3. Something you are

- 2FA means using 2 of these to authenticate

U2F: Universal 2 Factor (protocol)

- Largely adopted, developed by FIDO Alliance
- Based on challenge response:
- Leaves implementation details open
  - Designed for small devices
  - Describes message formats for USB, NFC, and Bluetooth

AUTHENTICATE

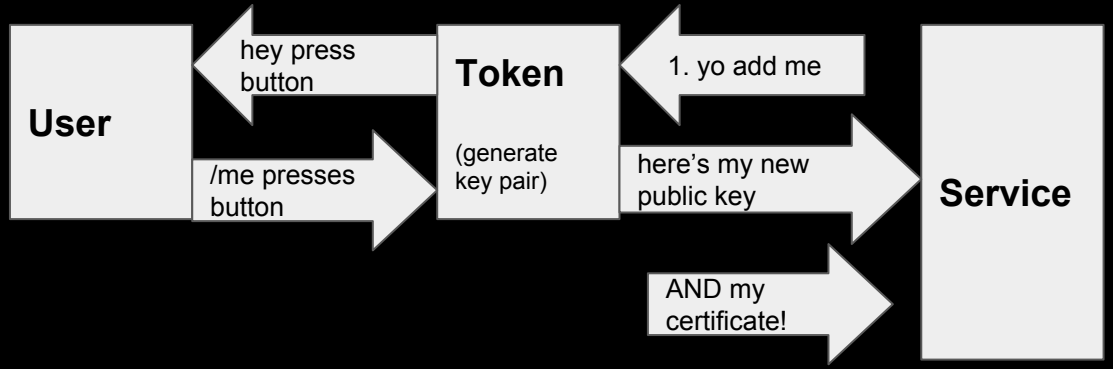


^THIS is ECDSA

-

U2F: a little more than challenge response...

REGISTER



- ^ must be done before a service can authenticate
- Certificate is used for attestation

Let’s make a U2F Token

- Need to support AUTHENTICATE and REGISTER
  - Compute power
  - Good random numbers
- The token should not be clonable
  - Tamper resistant storage for private keys
- It should be more secure than a phone app.
- Goal: only way to compromise factor is to physically steal key.

- 1st step: pick out a microcontroller or some secure element

On the topic of good random numbers

- Random on embedded is hard.
- Why not just sample a floating ADC?
  - biasing
- Cryptographically secure RNG's (DRNG)
  - Based on a tamper resistant secret state
  - Uses crypto primitives for output and reseeding.
  - Fast
- True RNG's (TRNG)
  - E.g. samples a flip flop biased between logic 1 and logic 0
  - Has feedback circuitry to prevent environment biases
  - slow

Secure microcontroller Market

Secure meaning:

- Accelerated ECC crypto, hardware RNG, tamper resistance

Good options:

NXP Semiconductors

infineon

ST Electronics
A company of ST Engineering

Atmel

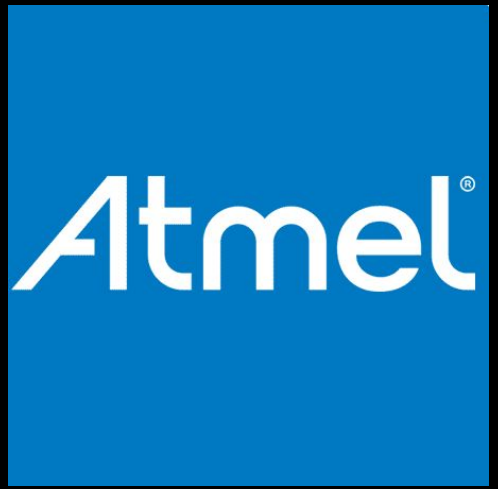
Honorable mentions but not quite there:

MICROCHIP

Problems purchasing secure microcontrollers...

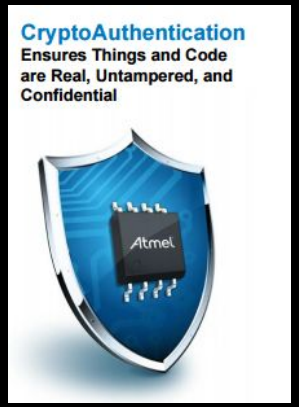
You really have to pierce the corporate walls:

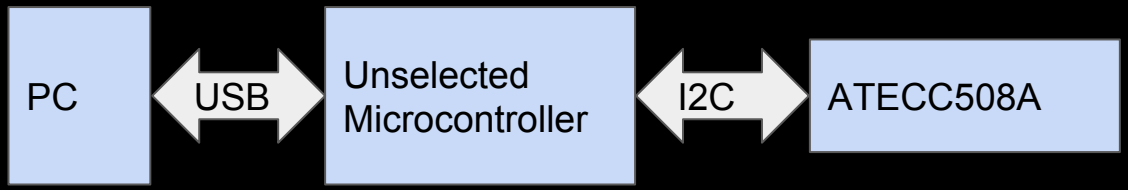
Anything Atmel sells is on Digikey



Atmel's ATECC508A (secure element)

- I2C peripheral
  - ECC P256 Acceleration
  - HW RNG
  - Tamper resistant storage
  - Monotonic counters
  - Internal clock, fault detection



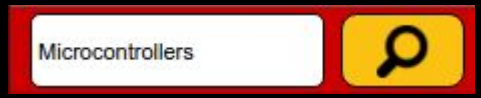


- All that's left: Microcontroller that does USB

How to pick a microcontroller that does 1 thing...

Go here ->

And do this ->



Pick wherever your 1 thing shows up->

Filter & sort by cheapest!

Image	Digi-Key Part Number	Manufacturer Part Number	Manufacturer	Description	Quantity Available	Unit Price USD	Minimum Quantity
	336-3408-5-ND	EFM8UB10F8G-C-QFN20	Silicon Labs	IC MCU 8BIT 8KB FLASH 20QFN	135 - Immediate	0.86000	1

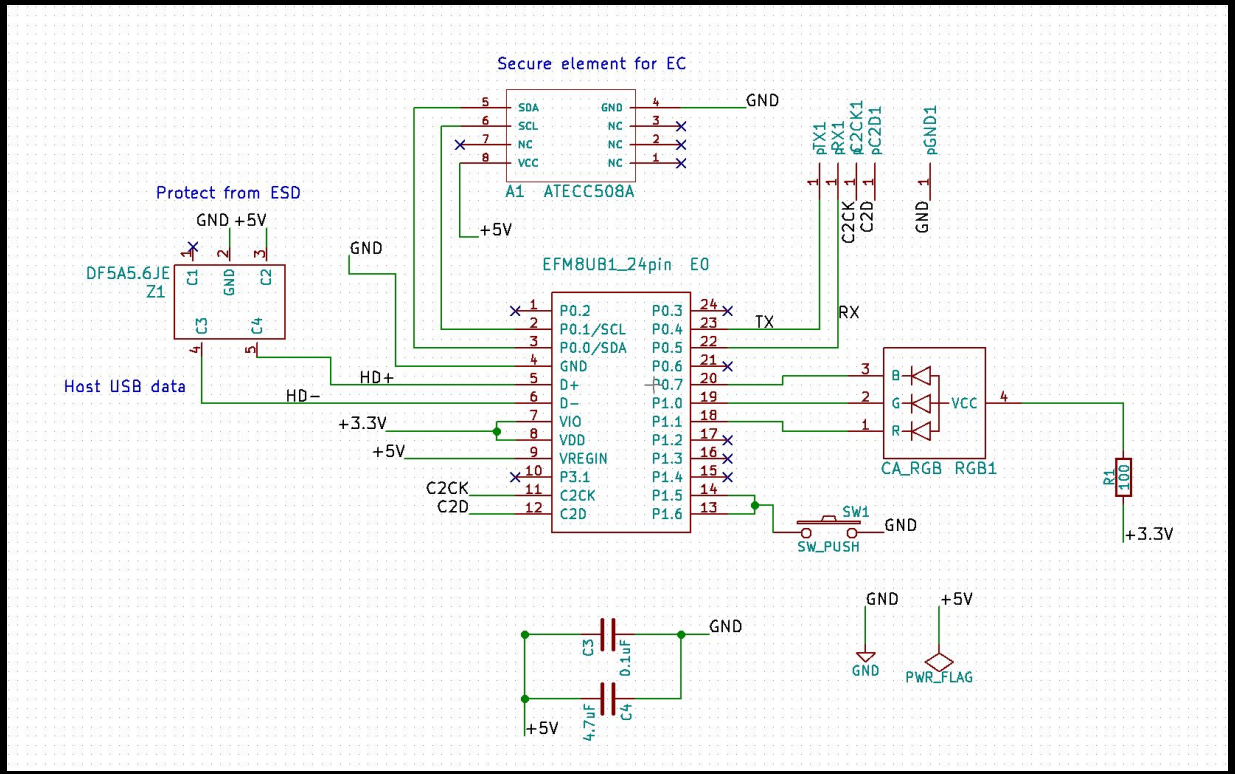
EFM8UB1

- 8 bit microcontroller by Silicon Labs
  - Internally uses the 8051 core (embedded core by Intel 1980's)
- Costs $0.60 - $1.00
- Has either 8 KB or 16 KB of memory
  - How much does U2F need? Don't know!
- Supports USB and I2C.
- Small enough to fit on USB token (20-28 pins)

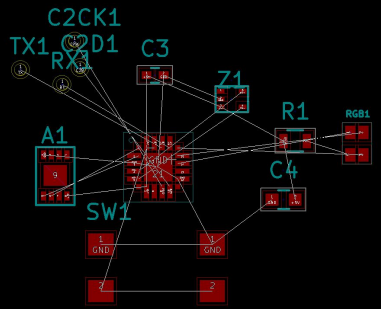
- No security features and no need

We have parts let’s make a schematic

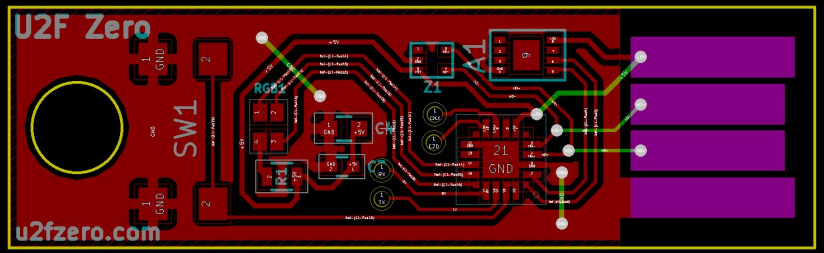
Logical connections between parts



Logical connections -> netlist -> layout

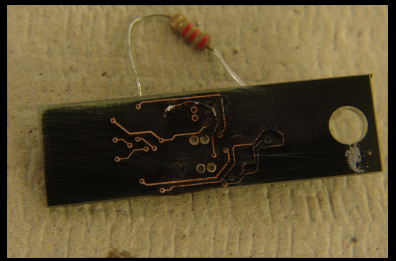


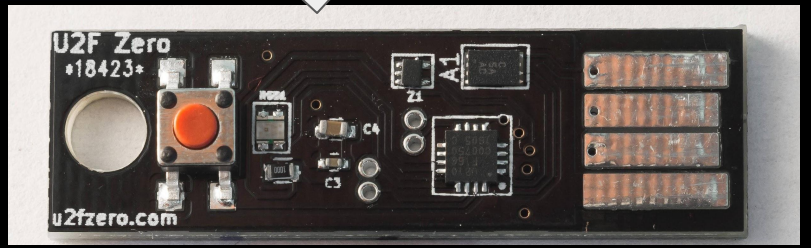
Convert rats nest into your final PCB



Get it fabbed ~$1 per PCB

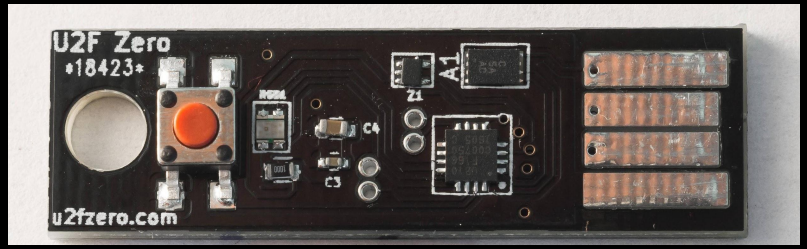
Nothing works the first try..

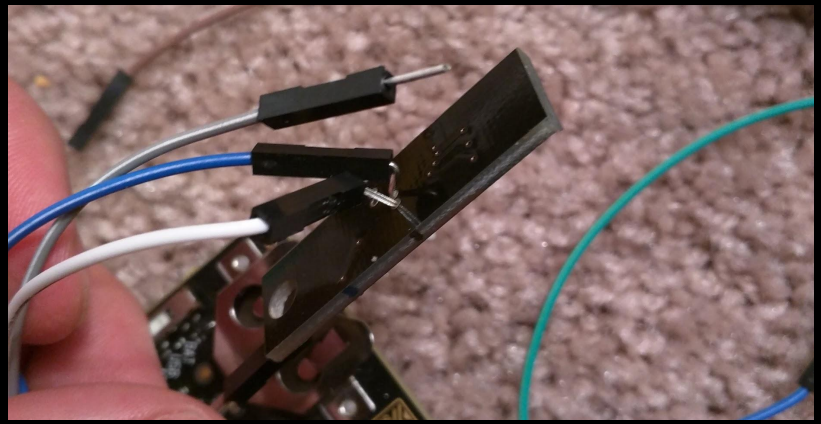




Programming

- Left two holes for programming pins and two holes for serial debugging



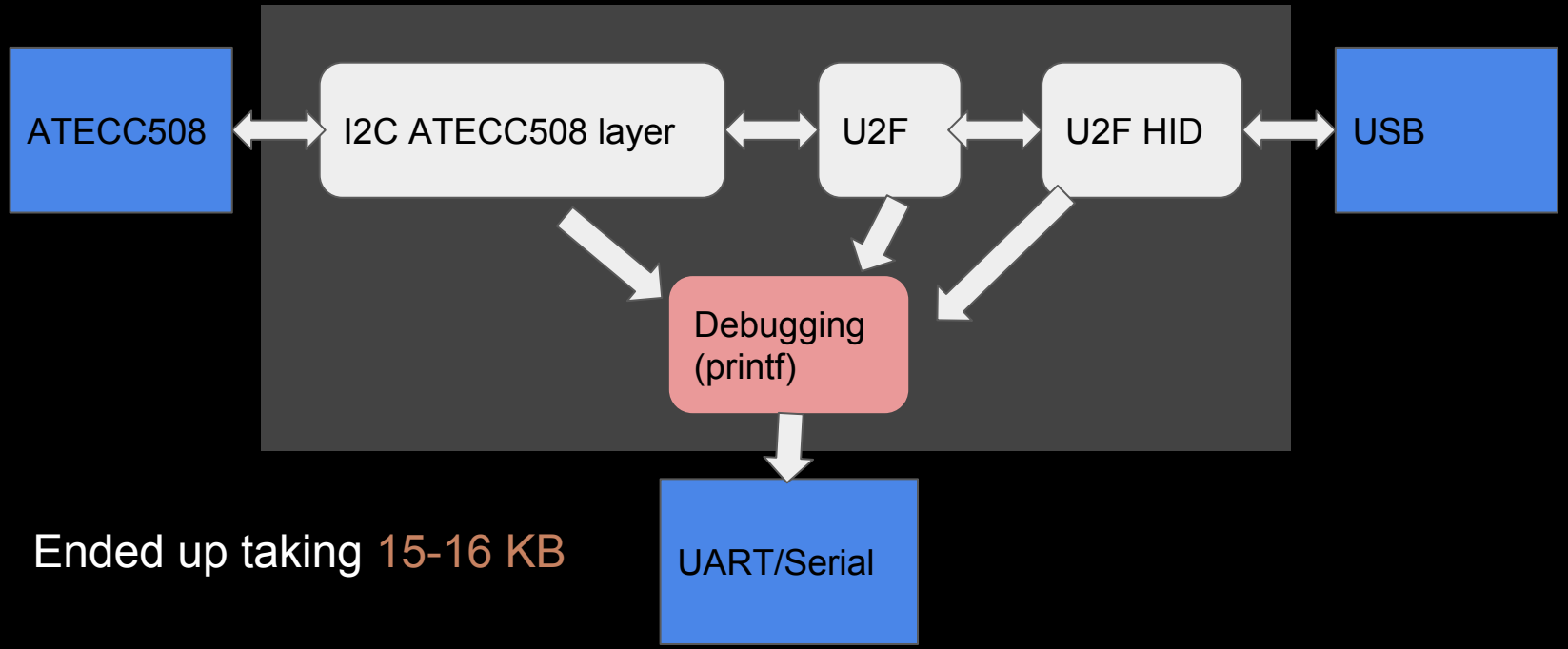


Steps:

1. Program with setup build to configure and lock ATECC508A
2. Generate a key pair on device for attestation.
3. Sign public key with my root key -> cert.
4. Reprogram with live build with the new cert.

Software

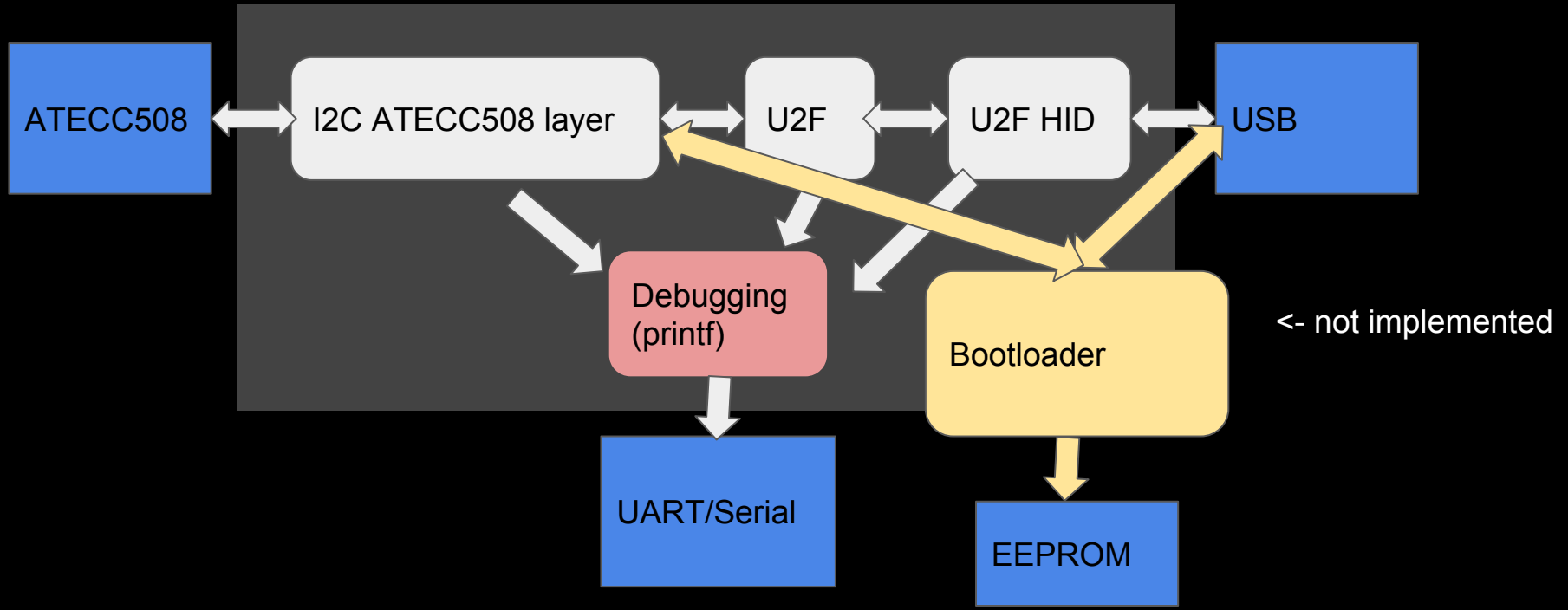
- Only have 8-16 KB of program memory and 1000 KB of RAM
- Need to write:



- Ended up taking 15-16 KB

One 1KB left...

Could add a bootloader so it can support updates from USB.



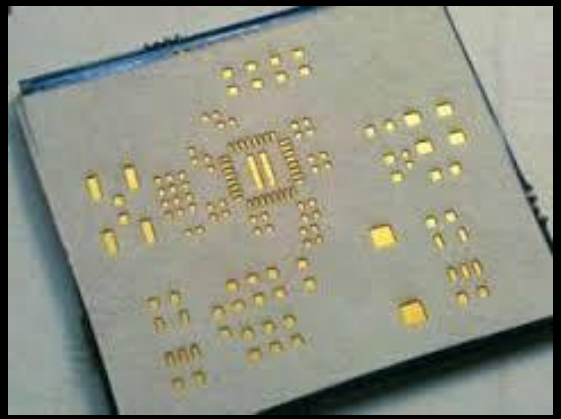
Costs

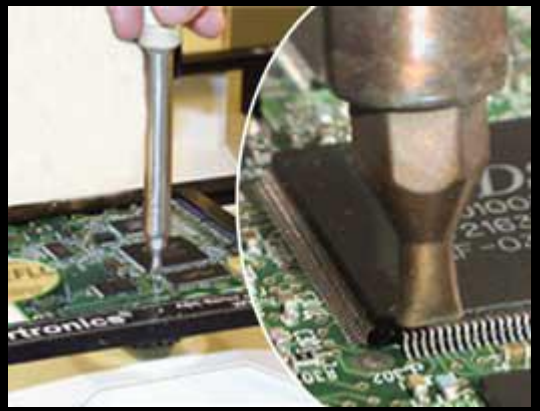
50 PCB’s from DirtyPCB’s for $48

file chosen	2	50	5x5 +$	
Thickness	**Coating**	**Stencil**	**Size (quantity 10+)**	**$48**
2.0mm +$	HASL	None	1.2 x 4.2 cm	

8 Surface mount parts from Digikey: $2.70

Tools for assembly: PCB stencil, Air gun, solder paste





Next steps

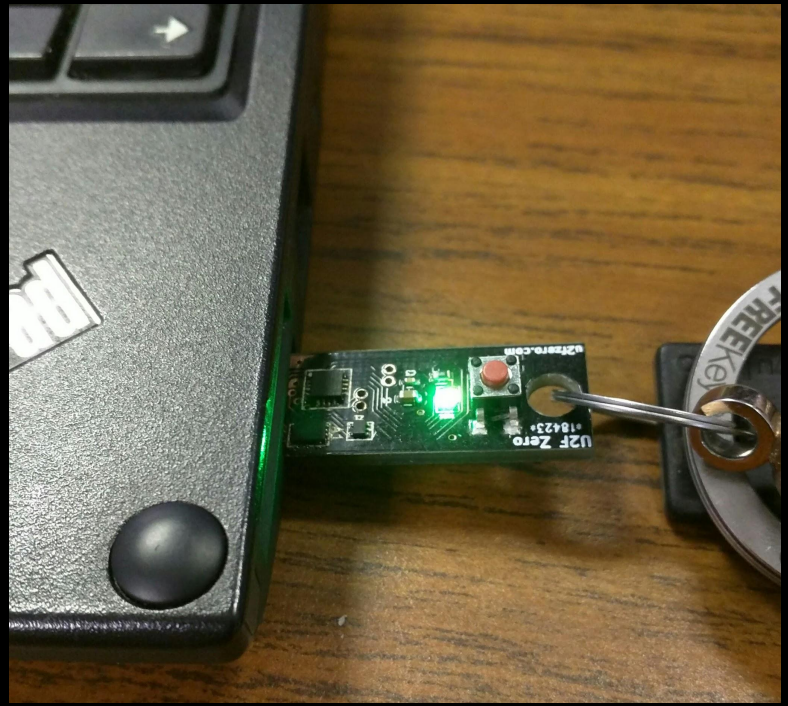
- Open source it publicly
- People can easily build their own
- Plan to get them fab'd and assembled externally
  - Sell them online or at VT

- Use parts that are easier to solder. Make programming holes nicer.
- Look into adding NFC or Bluetooth support for use with phones.
- Look for a fingerprint reader to replace button -> lost/stolen keys are safe.

- Anyone interested?

Demo

- Register with Duo for VT 2 factor



Contact (& source & trivia)

You can get source, documentation, design files here: **linx.li/u2f.zip**

Conor Patrick (twitter: @_conorpp)

https://conorpp.com

I appreciate any feedback on presentation here or in IRC