CrypTech

A More Assured HSM Design and Obsessing About the Tool-Chain

Hardware Security Module

From Wikipedia:

A hardware security module (HSM) is a physical computing device that **safeguards and manages digital keys** for strong authentication and provides crypto processing. These modules traditionally come in the form of a plug-in card or an external device that attaches directly to a computer or network server.

Creative Commons: Attribution & Share Alike

HSMs Are Used For

- Principally, Lock-box for Private Keys for
  - DNSsec
  - RPKI
  - PGP
  - Corporate Authentication
  - TOR
  - Your use goes here
- Also,
  - Encryption / Decryption
  - VPNs
  - Source of Randomness

Creative Commons: Attribution & Share Alike

Creative Commons: Attribution & Share Alike

The Need

- Every week a new horror about Crypto/Privacy
- *der Spiegel's* revelation of the "SpyMall Catalogue"
- Compromises of, and trojans in, most network devices, servers, firewalls, ...
- **We are relying on HSMs designed and made by 42-eyes government contractors**
- Many people are not comfortable with this

Creative Commons: Attribution & Share Alike

Origins

- This effort was started at the suggestion of Russ Housley, Jari Arkko, and Stephen Farrell of the IETF, to **meet the assurance needs of supporting IETF protocols in an open and transparent manner.**

- **But this is NOT an IETF, ISOC, ... project**, though both contribute. As the saying goes, "We work for the Internet."

Creative Commons: Attribution & Share Alike

Goals

- **An open-source <u>reference design</u> for HSMs**
- Scalable, first cut in an FPGA and CPU, later allow higher speed options
- Composable, e.g. "Give me a key store and signer suitable for DNSsec"
- Reasonable assurance by being open, diverse design team, and an increasingly assured tool-chain

Creative Commons: Attribution & Share Alike

Open and Transparent

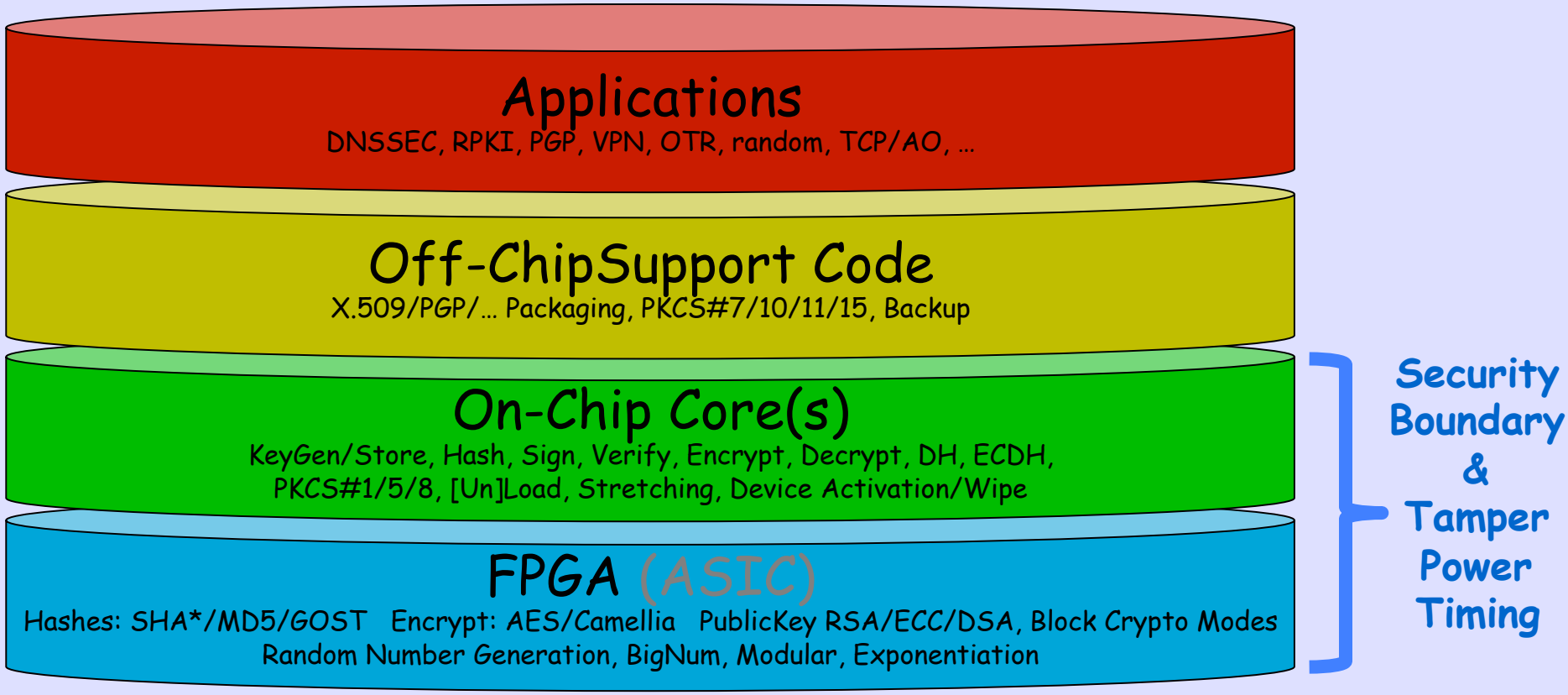
- The project is being run in a maximally **open, transparent** manner with traceability for all decisions etc.
- BSD and CC Licensed
- We do this in order to build trust in the project itself
- And **diverse**, engineering and funding

Creative Commons: Attribution & Share Alike

Funding (so far) From

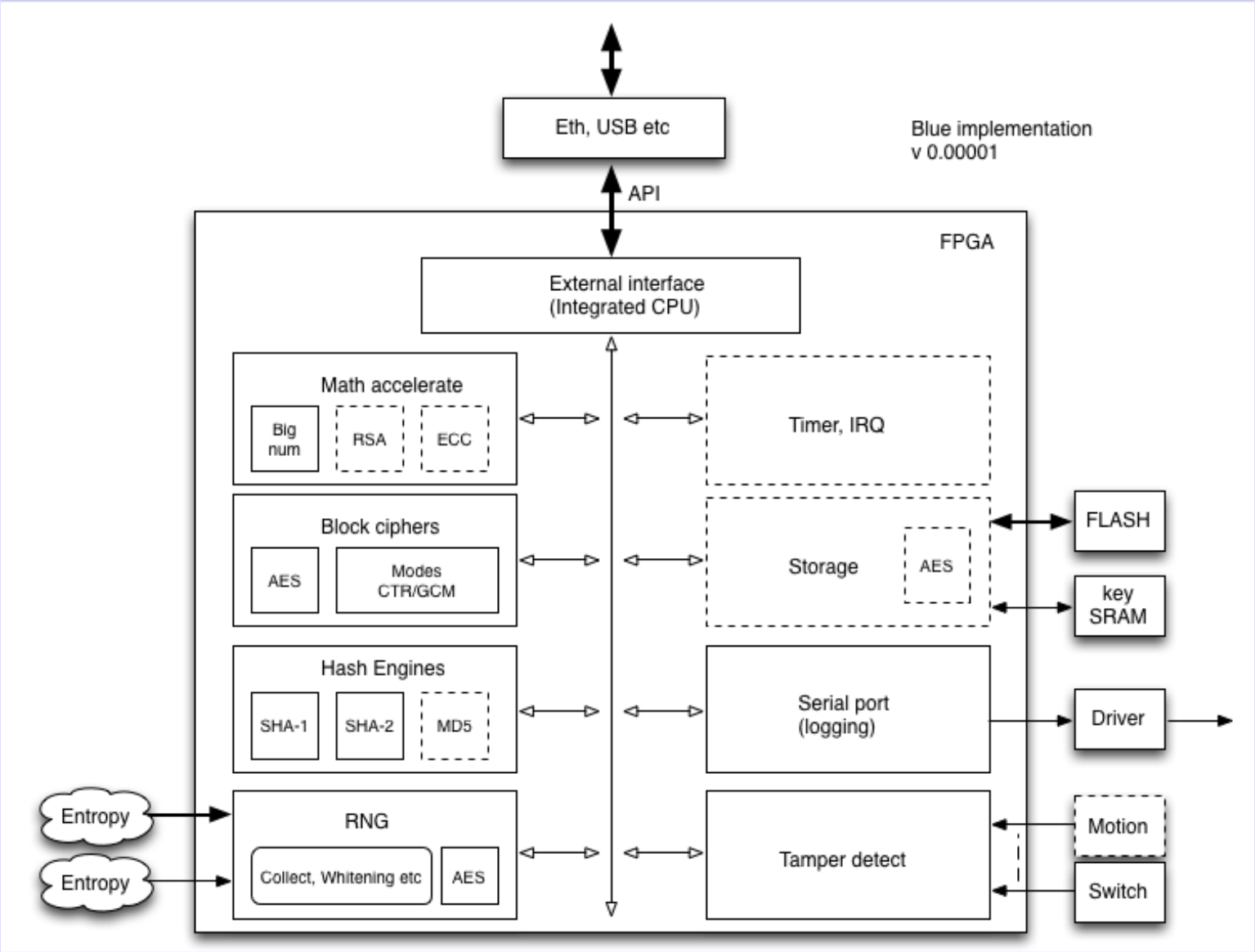
Creative Commons: Attribution & Share Alike

Layer Cake Model



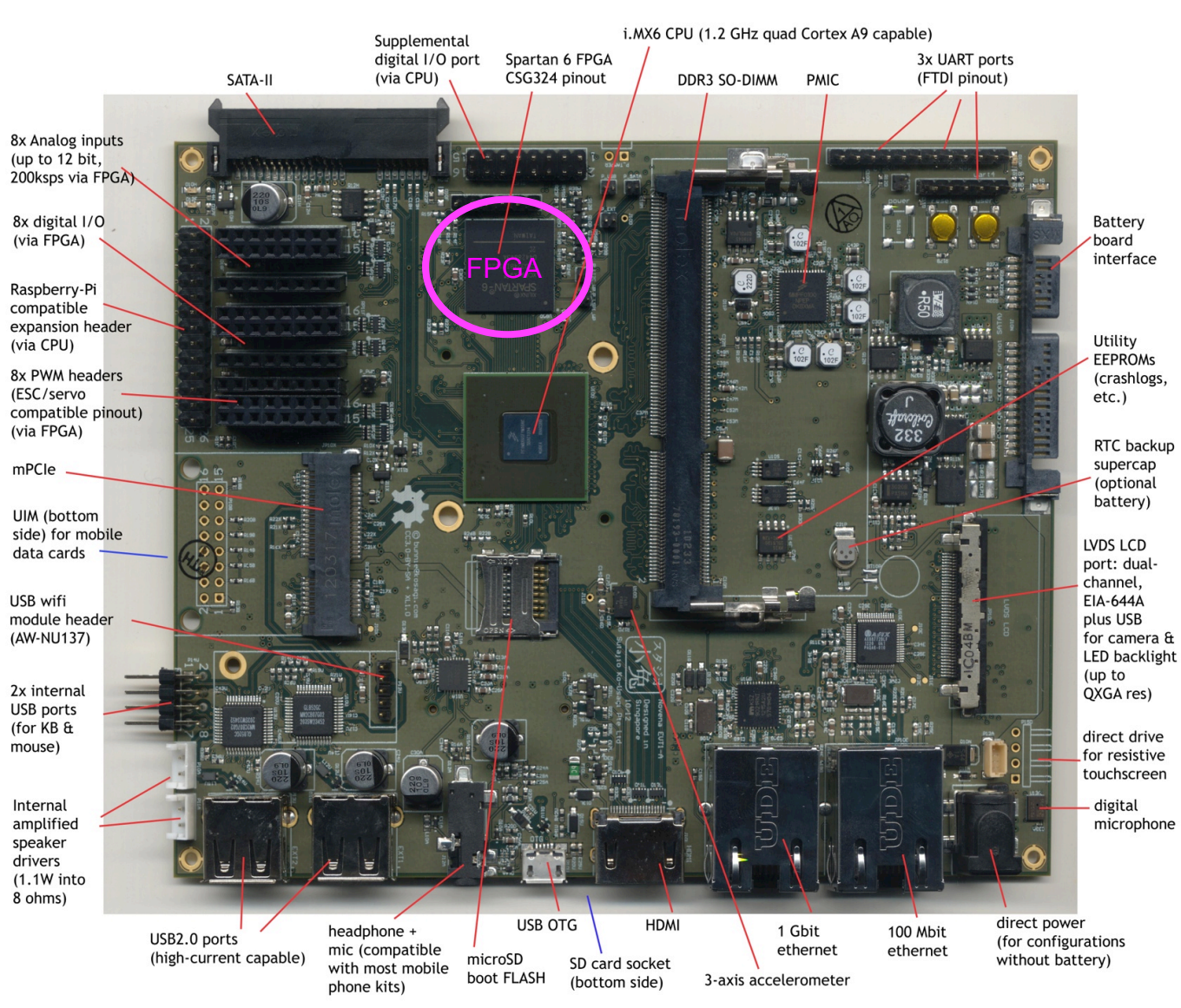
Creative Commons: Attribution & Share Alike

FPGA Cat Video



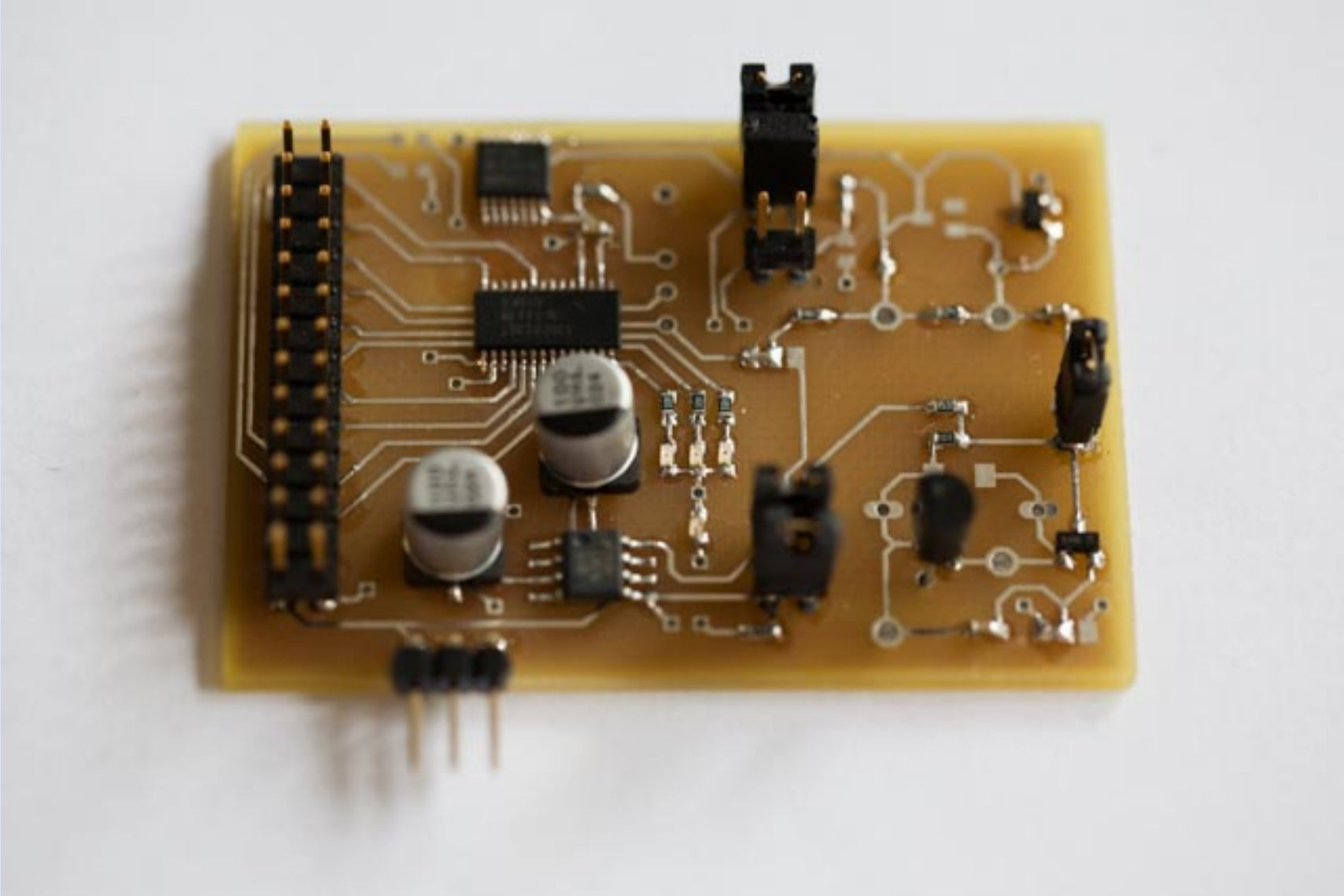
Creative Commons: Attribution & Share Alike

Novena Spartan 'Laptop'



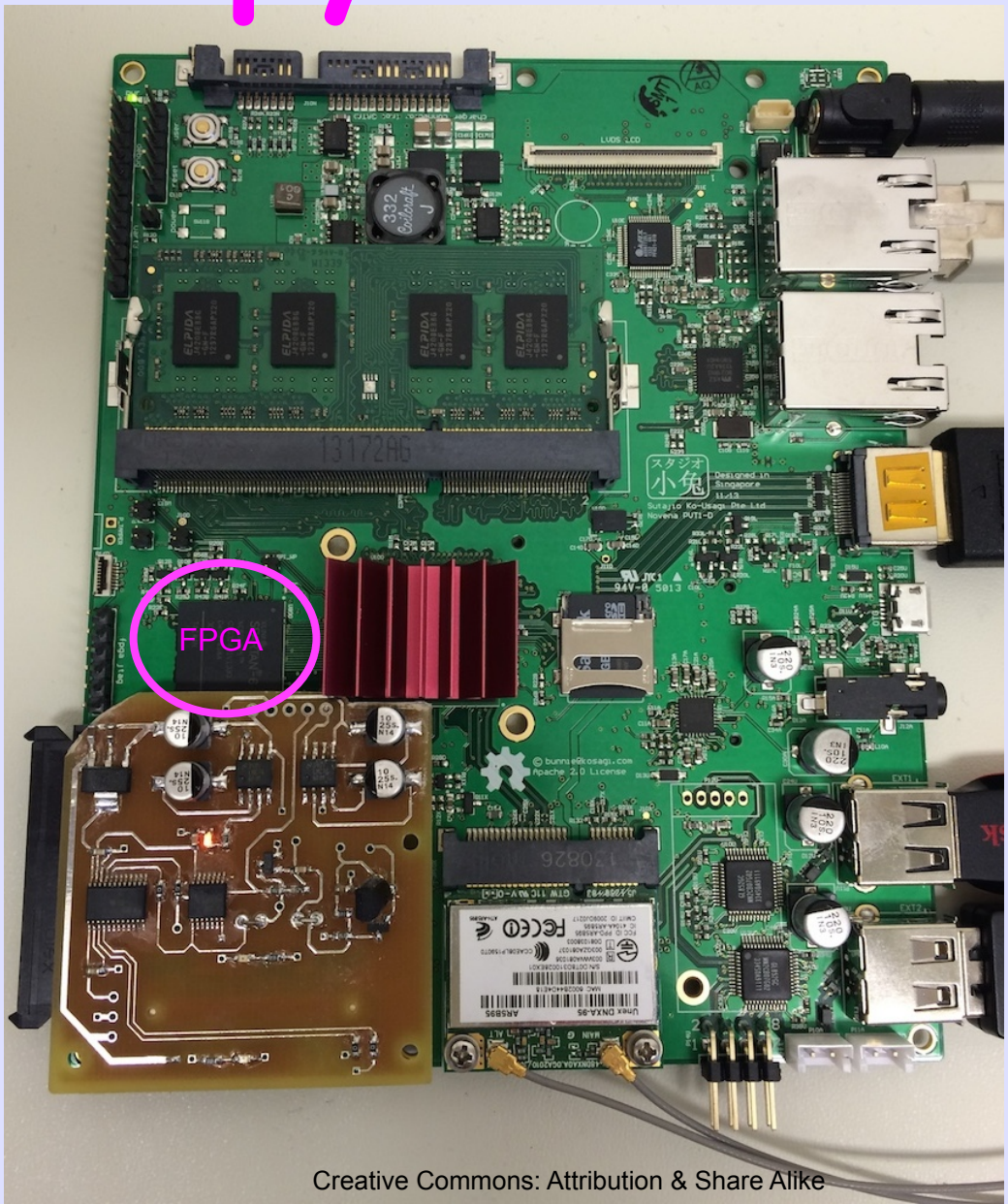
Creative Commons: Attribution & Share Alike

Entropy with Pi Pin-Out



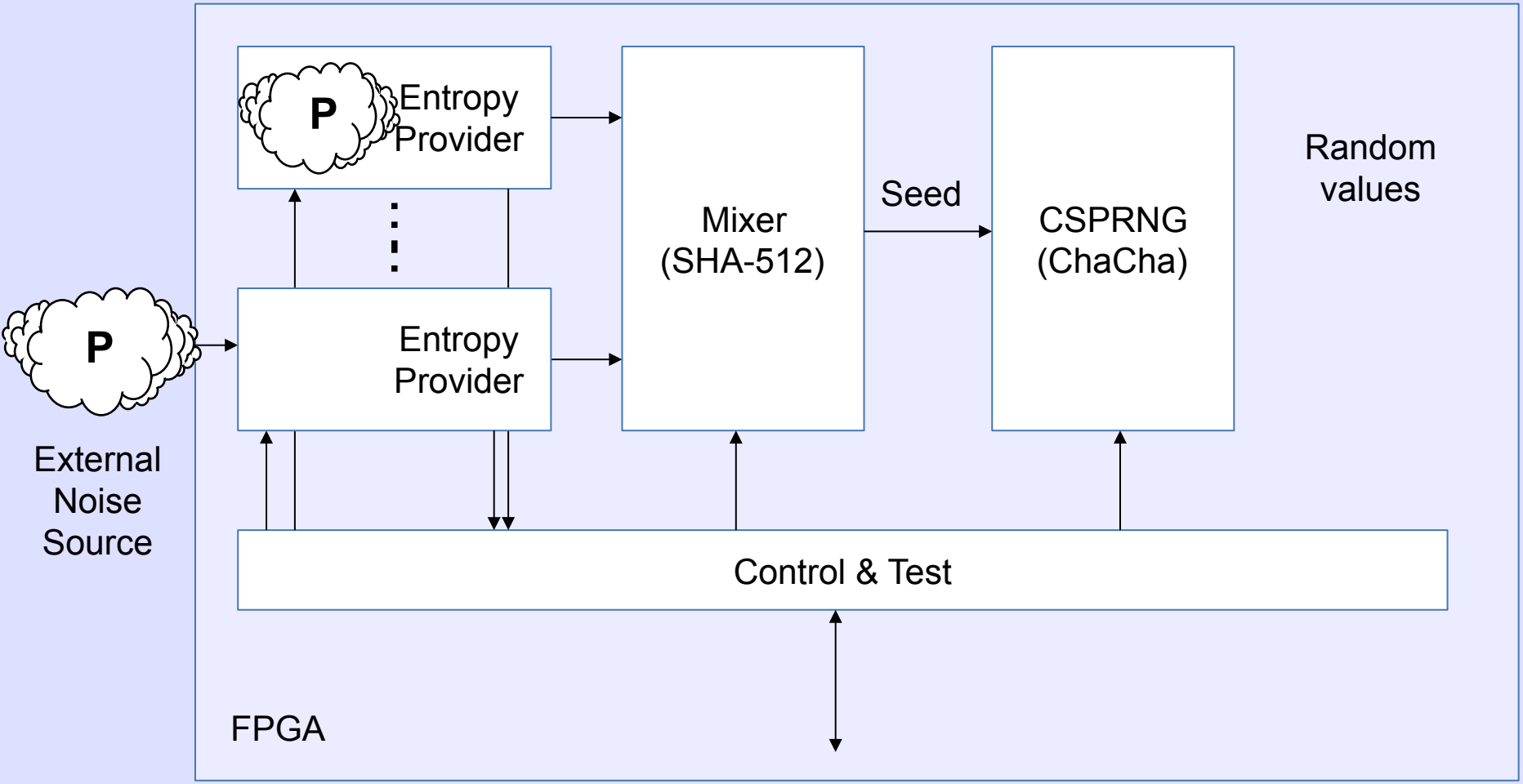
Creative Commons: Attribution & Share Alike

Entropy on Novena



Creative Commons: Attribution & Share Alike

The TRNG Architecture



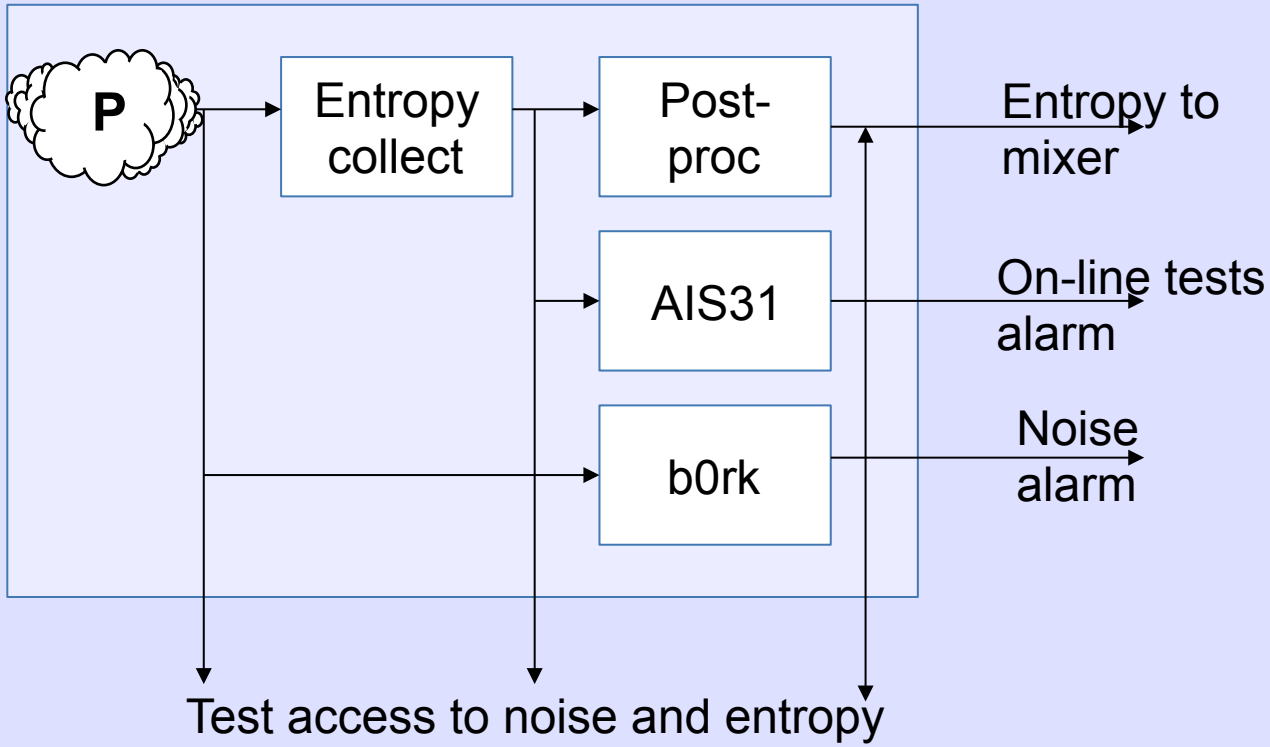
Creative Commons: Attribution & Share Alike

Test and Observability

- Two modes
  - Production mode (PM) and Test mode (TM)
- Observability of entropy sources in PM
- Continuous on-line testing in PM
- Injection in stages and complete chain in TM
- Generation of a small number of values in TM
- Allows test of all digital functionality including continuous tests.
- Full restart when moving from TM from/to PM

Creative Commons: Attribution & Share Alike

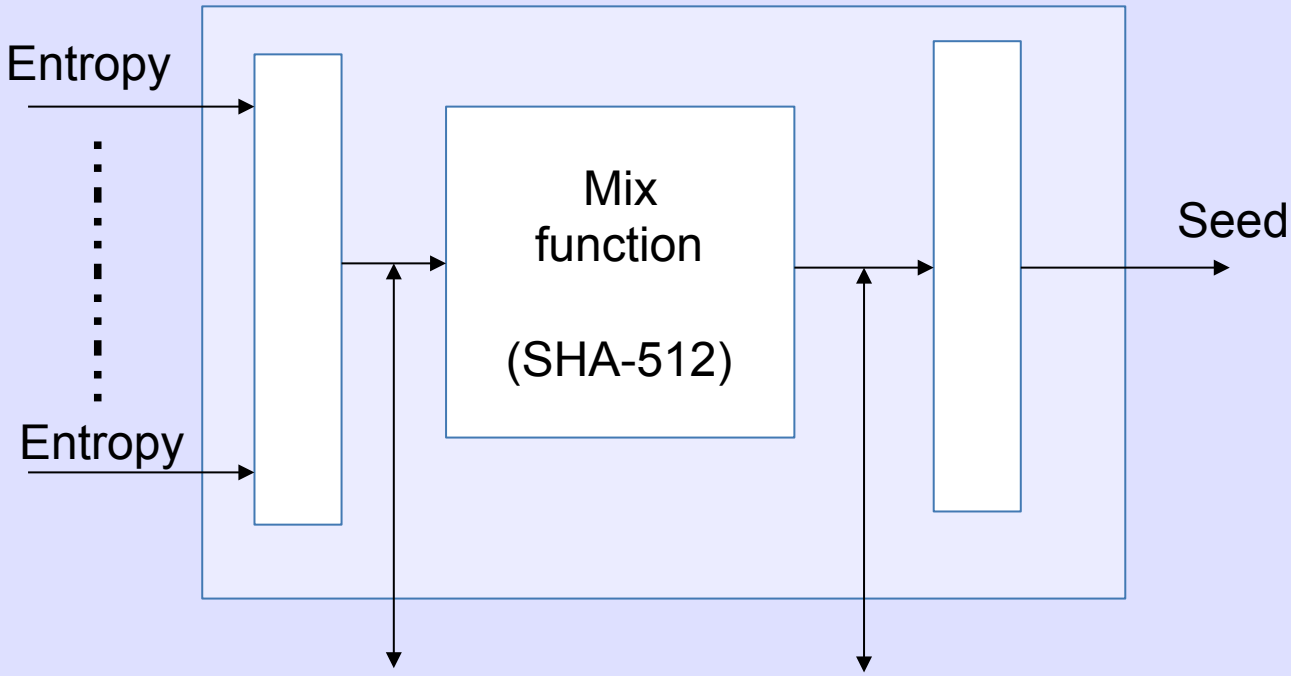
Observability & Test of Entropy Sources



- Extract for off-line comprehensive testing
- Inject for functional testing in test mode

Creative Commons: Attribution & Share Alike

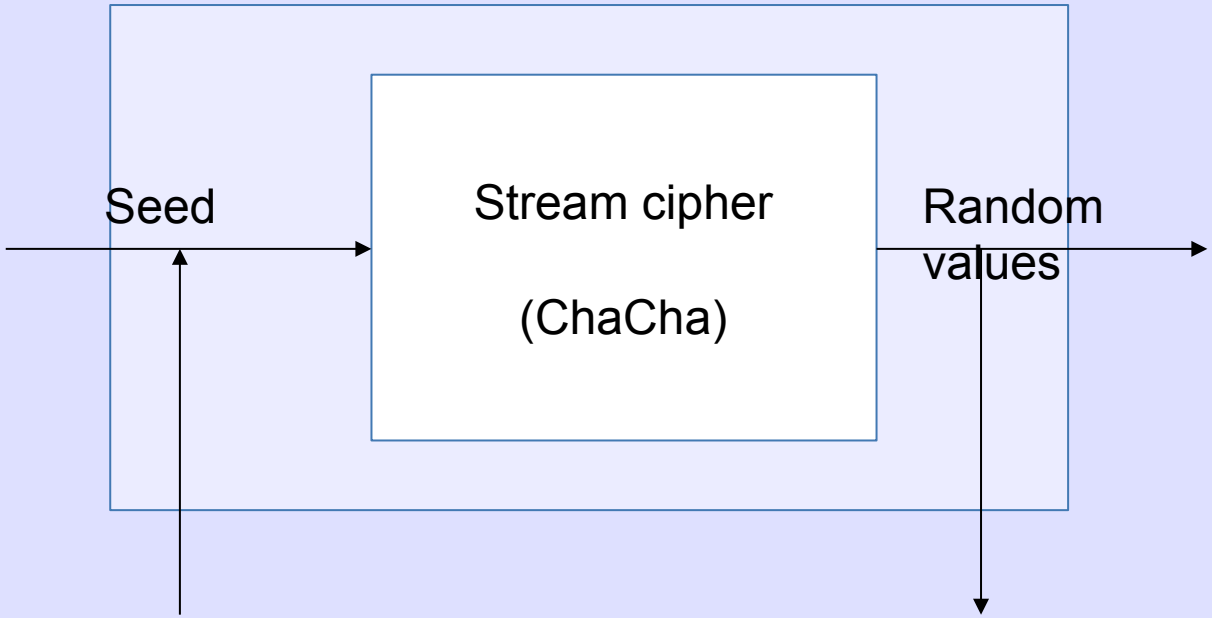
Observability & Test of Mixer



Inject for functional testing in test mode

Creative Commons: Attribution & Share Alike

Observability & Test of CSPRNG



Inject for functional testing in test mode

Creative Commons: Attribution & Share Alike

Some of the Fears

- ToolChain Poisoning
- Device Poisoning
- Side-Channel Attacks
- How can you tell if your vendor actually implemented CrypTech, and correctly?

Creative Commons: Attribution & Share Alike

The Tool Chain

- When my laptop's fan goes on, I think it is the NSA, GCHQ, Israelis, Chinese, … fighting to see who will own me
- **We have NO ASSURANCE of our tool set, from CPU to Kernel to Compiler to** …
- When constructing assurance-critical tools, we need to maximize assurance in the tools used to build them

Creative Commons: Attribution & Share Alike

The Compiler

- Ken Thompson's 1984 Turing Award paper *Reflections on Trusting Trust*

- A self-reproducing trap in the C compiler which "would match code in the UNIX "login" command. The replacement code would miscompile the login command so that it would accept either the intended encrypted password or a particular known password." You have been owned!

Creative Commons: Attribution & Share Alike

Double-Diverse Compilation

- In his 2009 PhD dissertation, David Wheeler explained how to counter the "trusting trust" attack by using the "Diverse Double-Compiling" (DDC) technique
- We can use this on GCC and clang to get somewhat assured compilers
- But you still have to inspect the source!

Creative Commons: Attribution & Share Alike

Critical Tool-Chain

- C compilers audited and built using DCC
- Audited kernel, libc, …
- Audited whole darn UNIX or Linux
- Audited Verilog compiler
- Audited FPGA download tools
- Audited test tools
- Trojan prevention & detection

Creative Commons: Attribution & Share Alike

HDL / Verilog

- But FPGAs/ASICs are programmed in a Hardware Definition Language, Verilog
- It is very hard to get an open Verilog compiler
- Verilog can not compile itself, so DDC is not applicable here, just a DCC C compiler
- We are working on methods of gaining trust in the FPGA tool chain

Creative Commons: Attribution & Share Alike

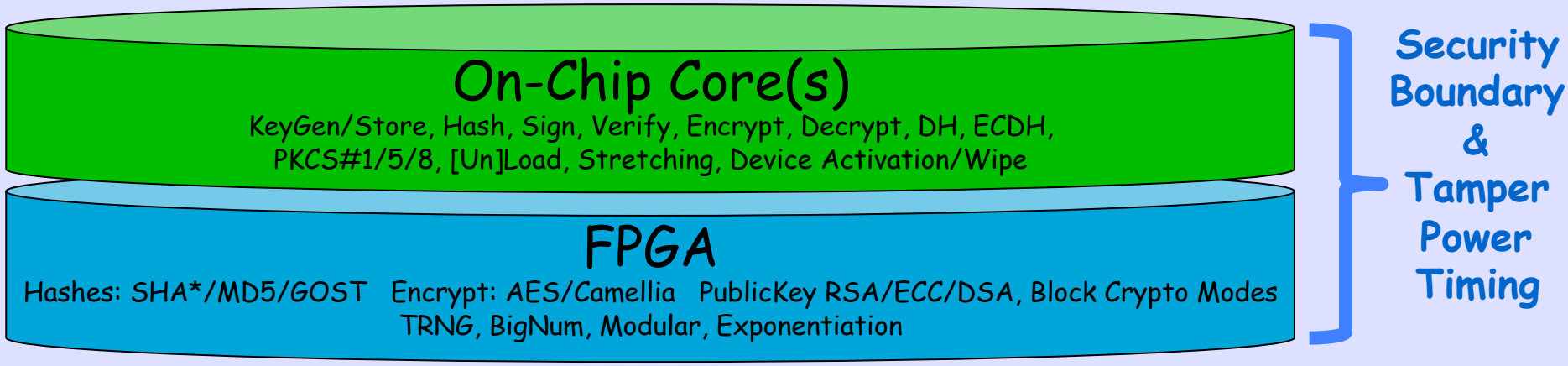
Side Channel & Tampering

- Exponentiation circuit **timing leaks** are exploitable remotely
- **Power leakage** is exploitable locally
- **Physical attack** detection critical
- Wipe key store if tampering detected
- On-board battery/capacitor to buy the time to wipe if unplugged from power
- Side-Channel attacks are the subject of entire conferences

Creative Commons: Attribution & Share Alike

Potting Boundary

- The FPGA/ASIC and accompanying Core(s) (ARM, whatever) are within the physically protected boundary of the chip carrier potting.
- We worry about side channel attacks, i.e. information leakage from how power is used, RF, data-dependent time to do an operation, etc.
- We worry about tampering, what if the chip is opened and attacked? So the potting includes tampering sensors and code to wipe all keys if tampering is detected.



Creative Commons: Attribution & Share Alike

Some Phases

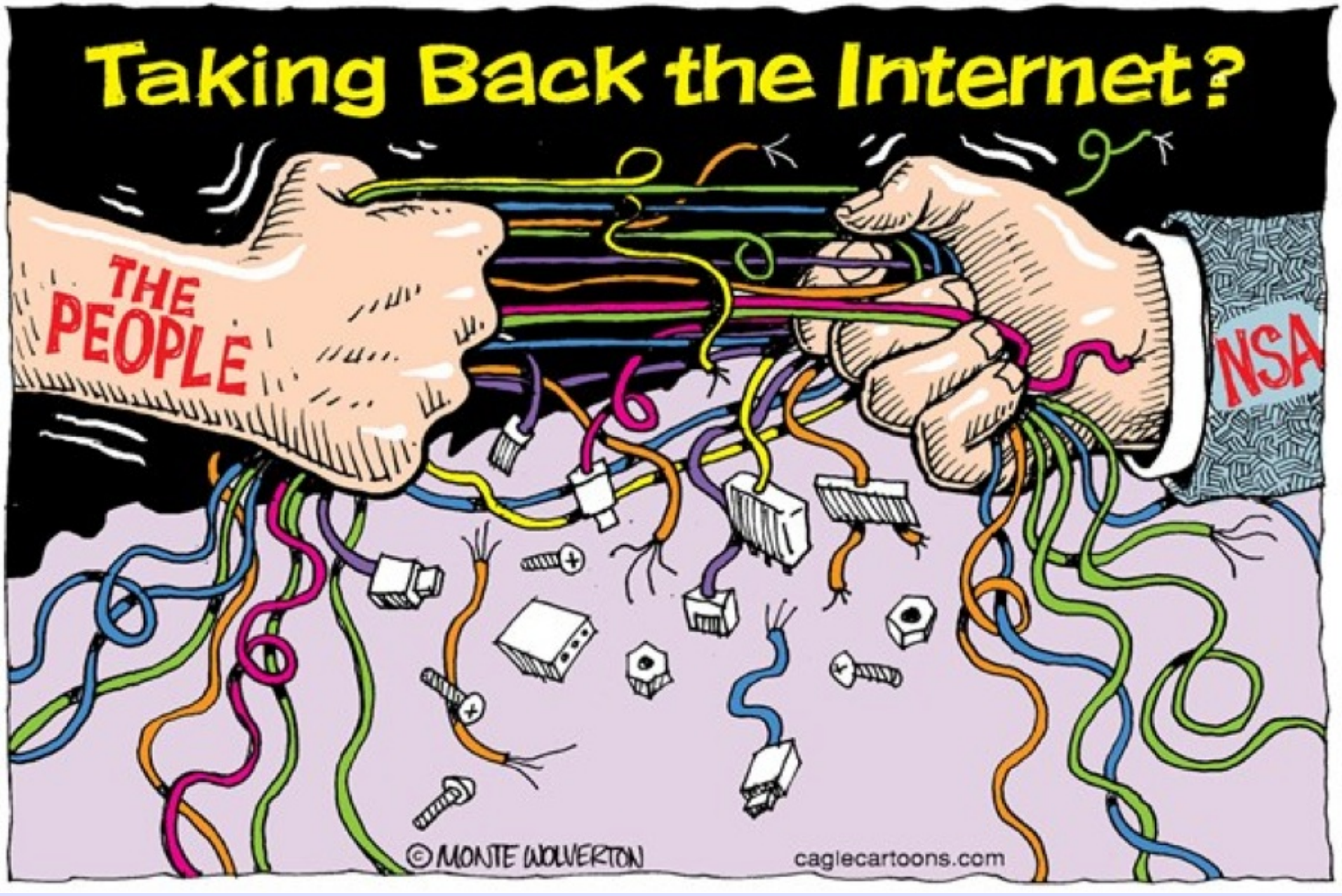
- First Year: Basic Design, not all cyphers, Tool-chain, not all protocols, prototype implementations on FPGAs and boards
- Second Year: Better Tool-chain, all needed cyphers, hashes, crypting, … and integration with some apps, DNSsec, RPKI, TLS, PGP, TOR
- Third Year: Solid packaging, ability to compose designs for use cases, etc.

Creative Commons: Attribution & Share Alike

A Few Related Projects

- Truecrypt audit: `http://istruecryptauditedyet.com` ☹
- OpenCores: `http://opencores.org`
- Icarus Verilog: `http://iverilog.icarus.com`
- Valgrind: `http://valgrind.org`
- clang+llvm: `http://clang.llvm.org`

Creative Commons: Attribution & Share Alike



Creative Commons: Attribution & Share Alike

https://cryptech.is/

Creative Commons: Attribution & Share Alike