CrypTech

October 2018

Barcelona

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.

HSMs are used for

- Principally, Lock-box for Private Keys for
  - DNSsec
  - RPKI
  - Web PKI
  - Tor
  - Corporate Authentication
- Also,
  - Encryption / Decryption
  - VPNs
  - Source of Randomness

Many Flavors and Sizes

But Can We Trust Them?

Broken Despite Gov Issues CC Evaluation & NIST FIPS 140-2 Validation

The Need

- It seems that every week we hear about a new Crypto/Privacy horror
  - Compromises, malware, backdoors, …
- We are relying on HSMs, but we have no insight into what is inside
- Many people are not comfortable with this


OBAMA MEETS WITH CHINA'S DICTATOR
ANY COMMENT ON YOUR OUTRAGEOUS CYBER-SPYING?
ASSOCIATED PRESS
IS THAT QUESTION FOR ME, OR HIM?
MIKE THOMPSON

Origins

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

However,

- This effort is NOT an IETF or ISOC project, though both contribute.
- As the saying goes, "We work for the Internet."

Goals

- An open-source reference design for HSMs, not a manufactured product.
- Scalable, first cut in an FPGA and CPU
  - Maybe a higher speed options later
- Composable: "*Give me a key store and signer suitable for DNSsec*"
- Reasonable assurance from by open design and diverse development team
  - Over time, increasing assured tool-chain

CrypTech Project

- An Open Design, not a Product
- Open – everything (documentation, design, code)
- BSD, CC license for everything developed by CrypTech
- Diverse development team and reviewers
- Support for transparency, testing, …
- Diverse funding sources: Comcast, Google, .SE, SUNET, PIR, ISOC, Afilias, RIPE, IANA, Cisco, and more

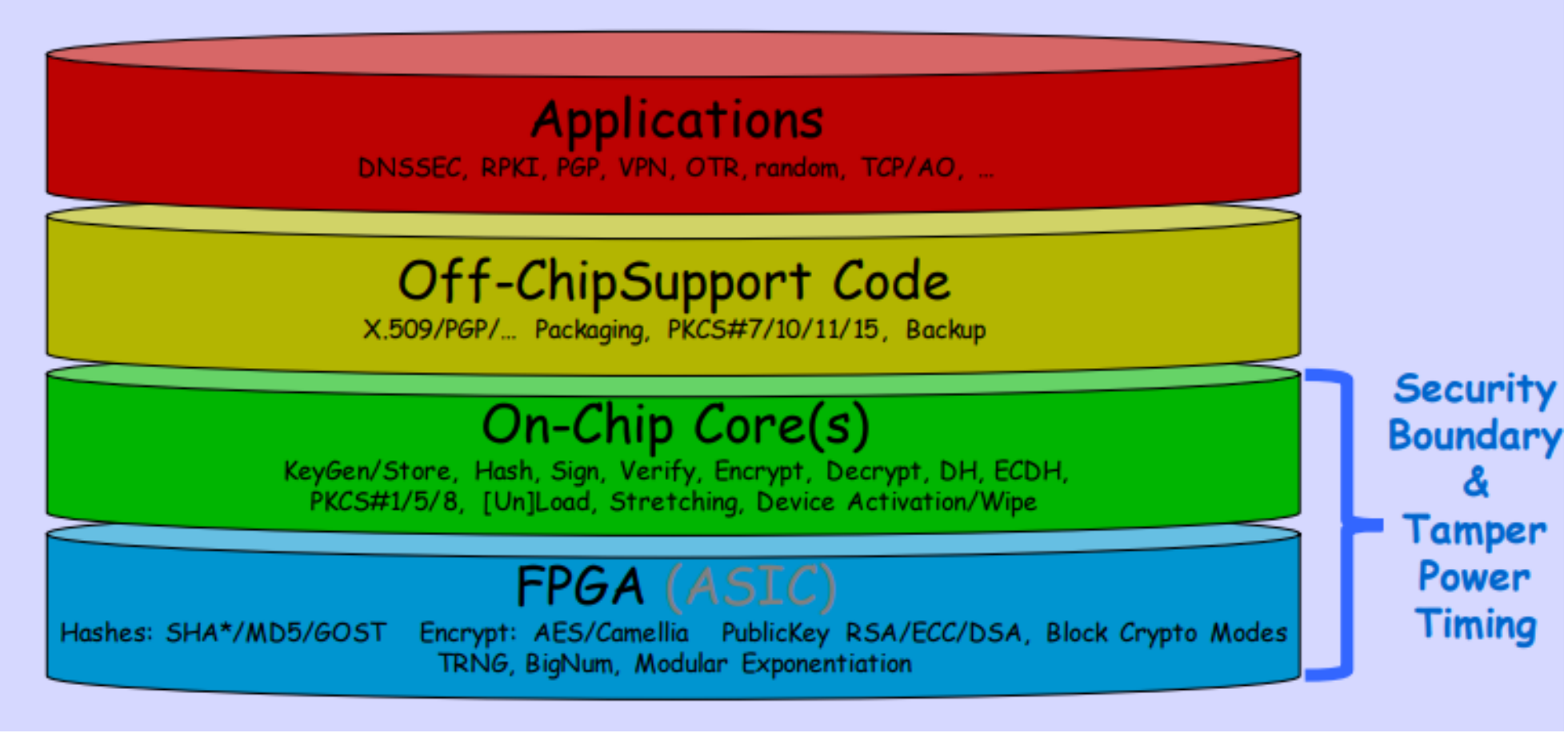
Diverse Engineering

- Verilog – Goteborg & Moscow
- Hardware Adaption Layer (HAL) - Boston
- Software, PKCS#11 – Boston
- TRNG advice from Germany and US
- Hardware Design and Build – Moscow & Stockholm
- DNSsec – Goteborg & Stockholm
- Engineering coordination – Tokyo & Stockholm

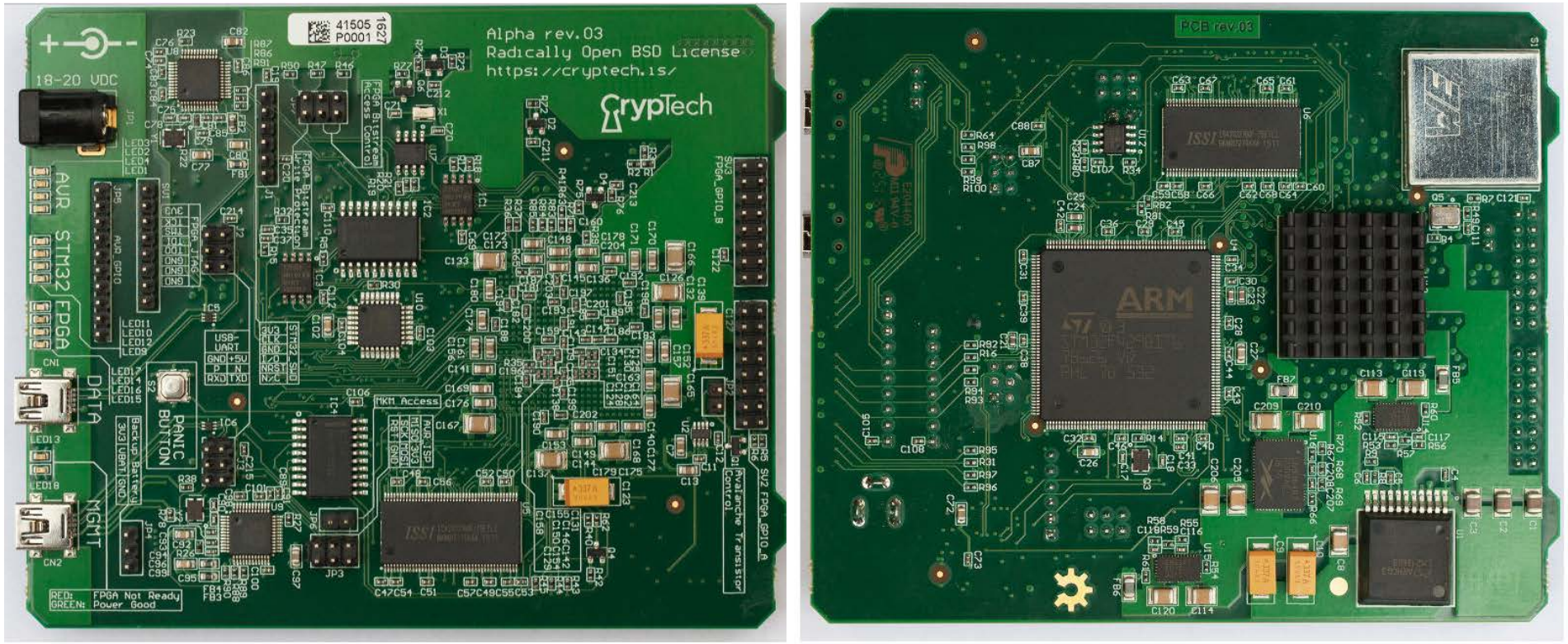
First practical instantiation

- DNSSEC signer
  - Works with OpenDNSSEC, BIND, Knot, and PowerDNS
  - Supports both RSA and ECDSA
  - Available soon: product offering from Diamond Key Security with CrypTech inside
- In final testing: Hash-based Signatures
- Under development: Ed25519

Layer Cake Model



CrypTech Alpha Board



General Core Design

- Plain Verilog 2001 compliant RTL code
- FPGA vendor and FPGA/ASIC agnostic design
  - No explicitly instantiated technology specific macros
- All cores are independent co-processors
  - Cores do not share resources
  - Load data and configure, start core and wait for ready signal
- 32-bit memory like interface
  - Implemented by core wrapper
  - API structured similarly for all cores

2018 Accomplishments (1 of 2)

- Performance Improvements
  - Revising and updating implementation to improve performance
  - Some of that work has led also to improved security (FPGA implementation of AES keywrap, for example)
- Hash-based Signatures
  - David McGrew's hash-based signature document: https://datatracker.ietf.org/doc/draft-mcgrew-hash-sigs
  - Quantum resistant signature scheme
  - Potential uses in signing code update

2018 Accomplishments (2 of 2)

- Ed25519
  - Edwards-curve signature algorithm
  - Crypto implementation done, working on drivers
  - Could implement x25519 without a lot of additional effort if needed
- External Security Code Audit
  - Completed in September of this year
  - Cure53 report is on our website: https://cryptech.is/2018/10/external-security-audit-completed/
  - Identified vulnerabilities should be fixed by year-end

2019 Plans

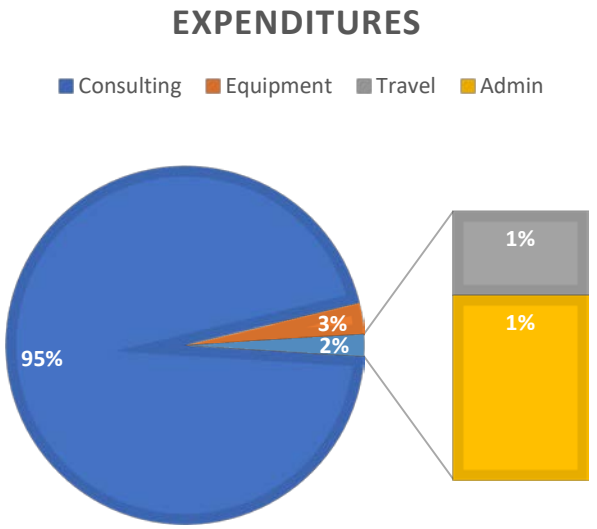
- Likely an update to the hardware
- Continued improvements to security
- Continued focus on performance improvements

- Supported commercial product offering available from Diamond Key Security

- Welcome input on new use cases

Financials

- We've raised about USD 1.9 million
- We've spent about USD 1.7 million
- Additional financial support is most welcome



CrypTech: Thanks to our Funders:

DuckDuckGo

Backup Slides:

Core Walk Through

SHA-1

- Implements SHA-1 as specified in FIPS 180-2
- Iterative, one cycle/round
  - - 82 cycles/block with setup and finish
- Block expansion (W mem) implemented using sliging window with 16 separate 32-bit registers
- Test benches for w_mem, core and top level
  - Using NIST test vectors

SHA-256

- Implements SHA-256 as specified in FIPS 180-4
- Iterative, one cycle/round
  - 66 cycles/block with setup and finish
- Block expansion (W mem) implemented using sliding window with 16 separate 32-bit registers
- Test benches for w_mem, core and top level
  - Using NIST test vectors
- Heavily tested with SW on the Novena
- Used for DNSSEC

SHA-512

- Implements SHA-512/x (FIPS 180-4)
- Including SHA-512/224, SHA-512/256, SHA-512/384 and SHA-512
- Iterative, one cycle/round
  - 82 cycles/block with setup and finish
- Block expansion (W mem) implemented using sliding window with 16 separate 64-bit registers
- Support for work factor processing with up to 2**32-1 iterations/block
- Test benches for w_mem, core and top level
  - Using NIST test vectors
- Heavily tested with SW on the Novena
- Used in Cryptech as mixer in the TRNG

AES (1 of 3)

- As specified in FIPS 197
  - Support for 128 and 256 bit keys
- Iterative, four cycles/round
  - 42 cycles/block with setup and finish for AES-128
  - 58 cycles/block with setup and finish for AES-256

AES (2 of 3)

- Key expansion performed before any block processing
  - 10 cycles for 128 bit keys, 14 cycles for 256 bit keys
- Separate encipher and decipher data paths
  - Decipher can be removed for modes where only encipher is needed (e.g., CTR mode)
  - Encipher and decipher share key expansion

AES (3 of 3)

- Four sbox ROMs
  - Shared between encipher data path and key expansion
- Test benches for key expansion, data paths, core and and top level
  - Using NIST test vectors and vectors by Sam Trenholme
  - See http://www.samiam.org/key-schedule.html
- Cryptech implements AES Key Wrap to protect private keys
  - RFC 5649
  - See https://tools.ietf.org/html/rfc5649

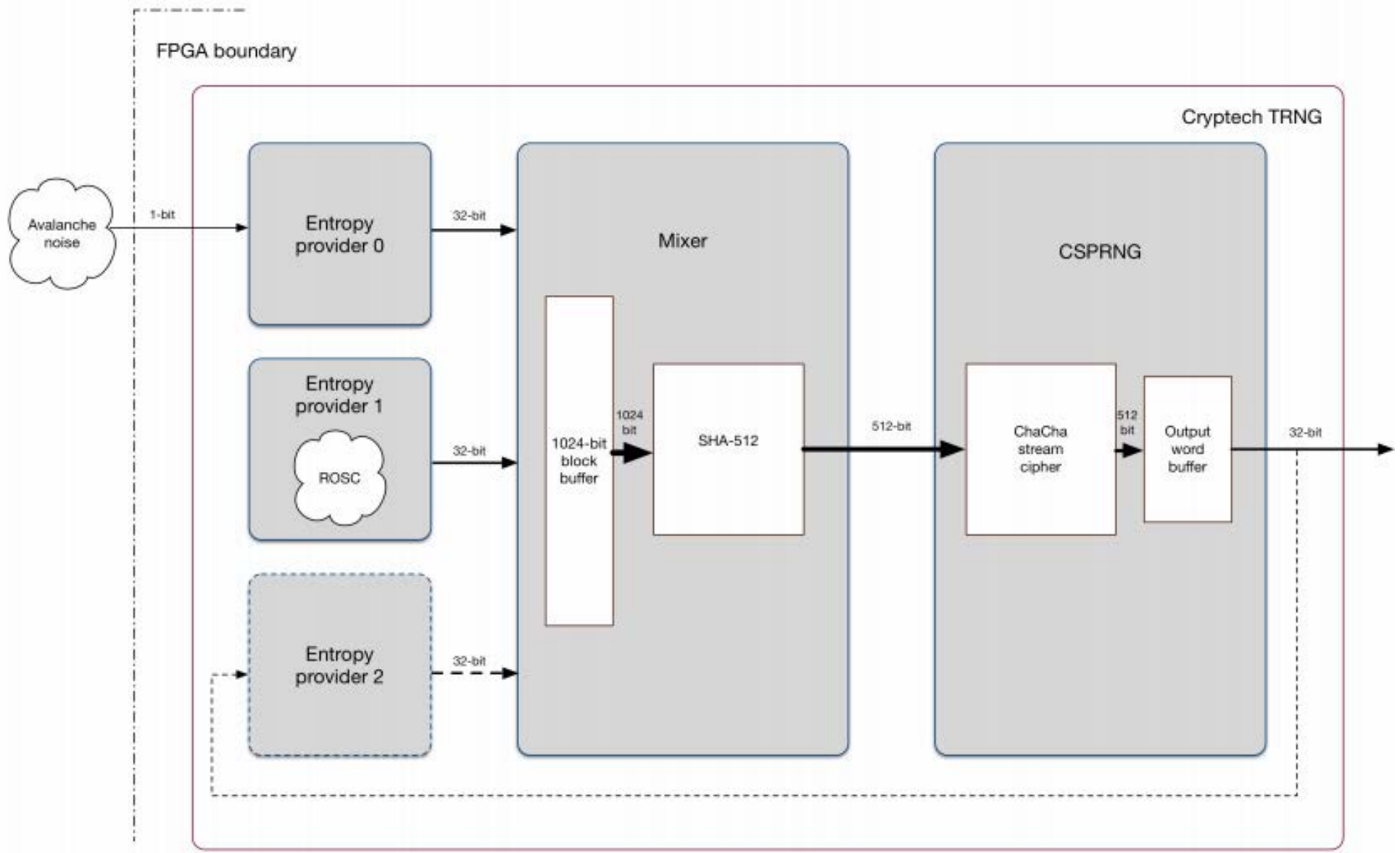
ChaCha (1 of 2)

- Implements the ChaCha stream cipher
  - http://cr.yp.to/chacha/chacha-20080128.pdf
  - Support for 128 and 256 bit keys
  - Support for up to 32 rounds
  - Support for settable 64-bit initial counter value
- Iterative, two cycles/double round
  - 42 cycles/block with setup and finish for ChaCha20

ChaCha (2 of 2)

- Testbenches for core and top level
  - Using DJB test vectors and generated test vectors
  - https://tools.ietf.org/html/draftstrombergson-chacha-test-vectors-00
- Used in Cryptech as CSPRNG in the TRNG
  - With 256 bit key and 24 rounds
  - Key, block, IV and initial counter as seed

TRNG (1 of 4)


FPGA boundary
Cryptech TRNG
Avalanche noise
1-bit
Entropy provider 0
Entropy provider 1
ROSC
Entropy provider 2
32-bit
32-bit
32-bit
Mixer
1024-bit block buffer
1024 bit
SHA-512
512-bit
CSPRNG
ChaCha stream cipher
512 bit
Output word buffer
32-bit

TRNG (2 of 4)

- Sub system using multiple cores
  - Avalanche noise entropy provider core
  - Ring oscillator entropy provider core
  - SHA-512 core used as entropy mixer
  - ChaCha core used as CSPRNG

TRNG (3 of 4)

- Modular architecture
  - Support for adding more entropy sources
  - Support for replacing SHA512 in mixer and ChaCha in CSPRNG with other cores
- Support for observability and testing and of all parts and output
  - Extract raw noise and entropy from the sources
  - Inject test vectors and extract results to allow verification of functionality
  - Planned support for on-line testing and alarms for entropy sources and CSPRNG

TRNG (4 of 4)

- Scalable performance and security
  - Number of rounds (default 24) can be configured via API
  - Reseed frequency settable and can be forced via API
  - Can generate ~500 Mbps @50 MHz clock frequency
  - Can instantiate multiple ChaCha cores (seeded separately) to scale performance to multiple Gbps performance
- Tested using ent, diehard, dieharder, and several custom tools
  - TBytes of data generated and tested so far