HS1-SIV

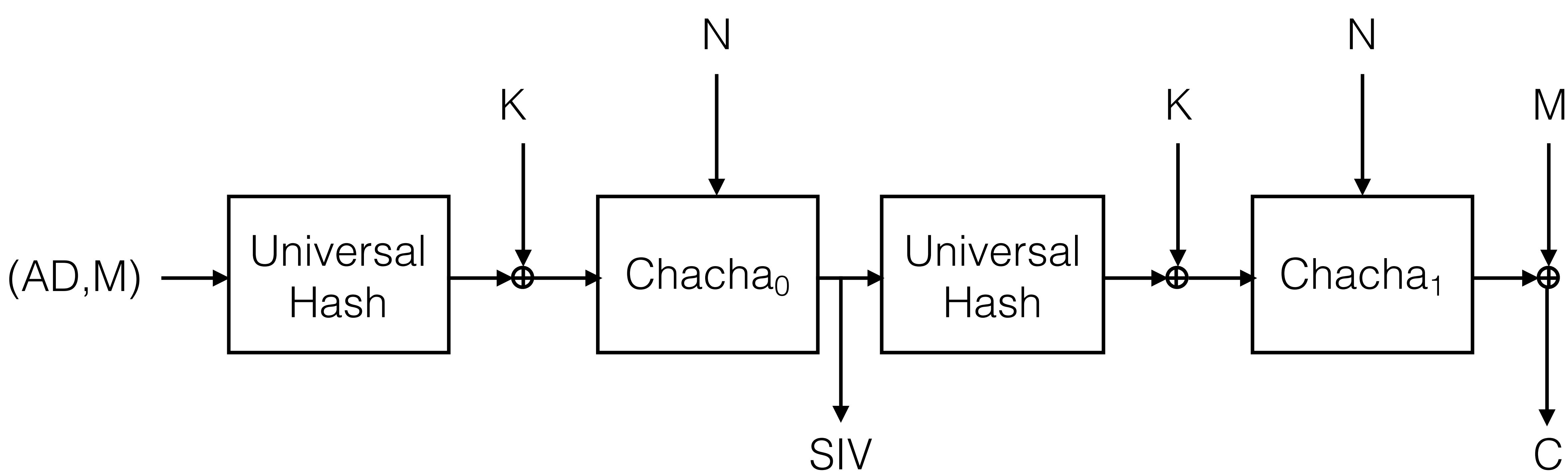
Ted Krovetz
Sacramento State

Goals:

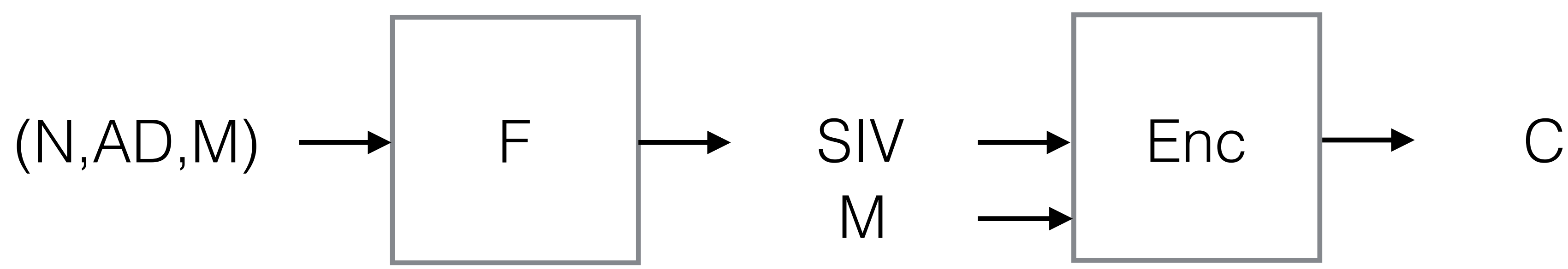
- Assemble from off-the-shelf, well-known parts
- Nonce reuse tolerant
- Balanced performance
- Provable security

Main Idea

- SIV for nonce reuse tolerance [RS06]
- Universal hash for input consumption
- Stream cipher for byte production



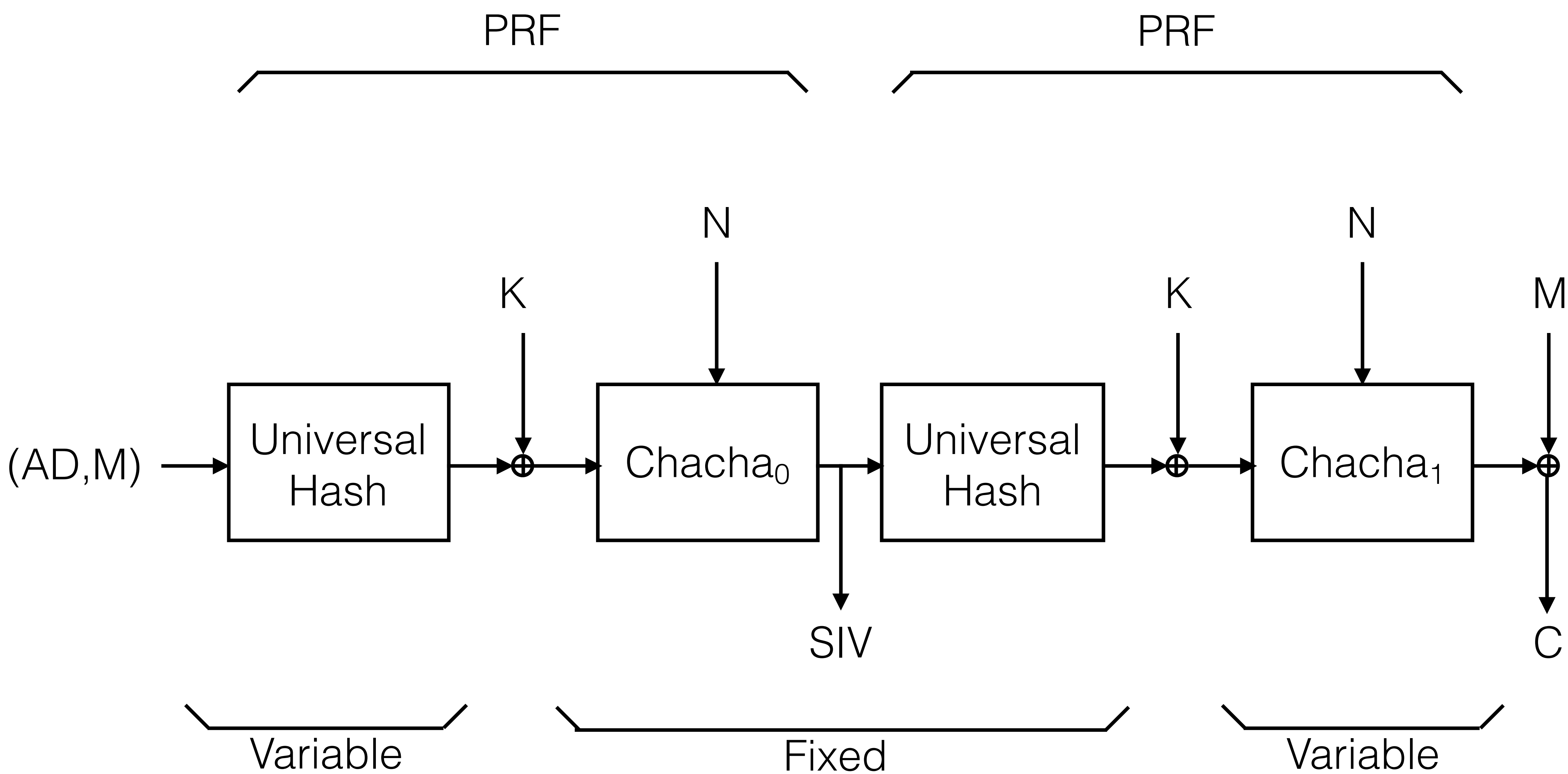
SIV



- Secure against nonce-respecting adversaries
- Easy to see when (N,AD,M) repeats
- Birthday bound: SIV repetitions

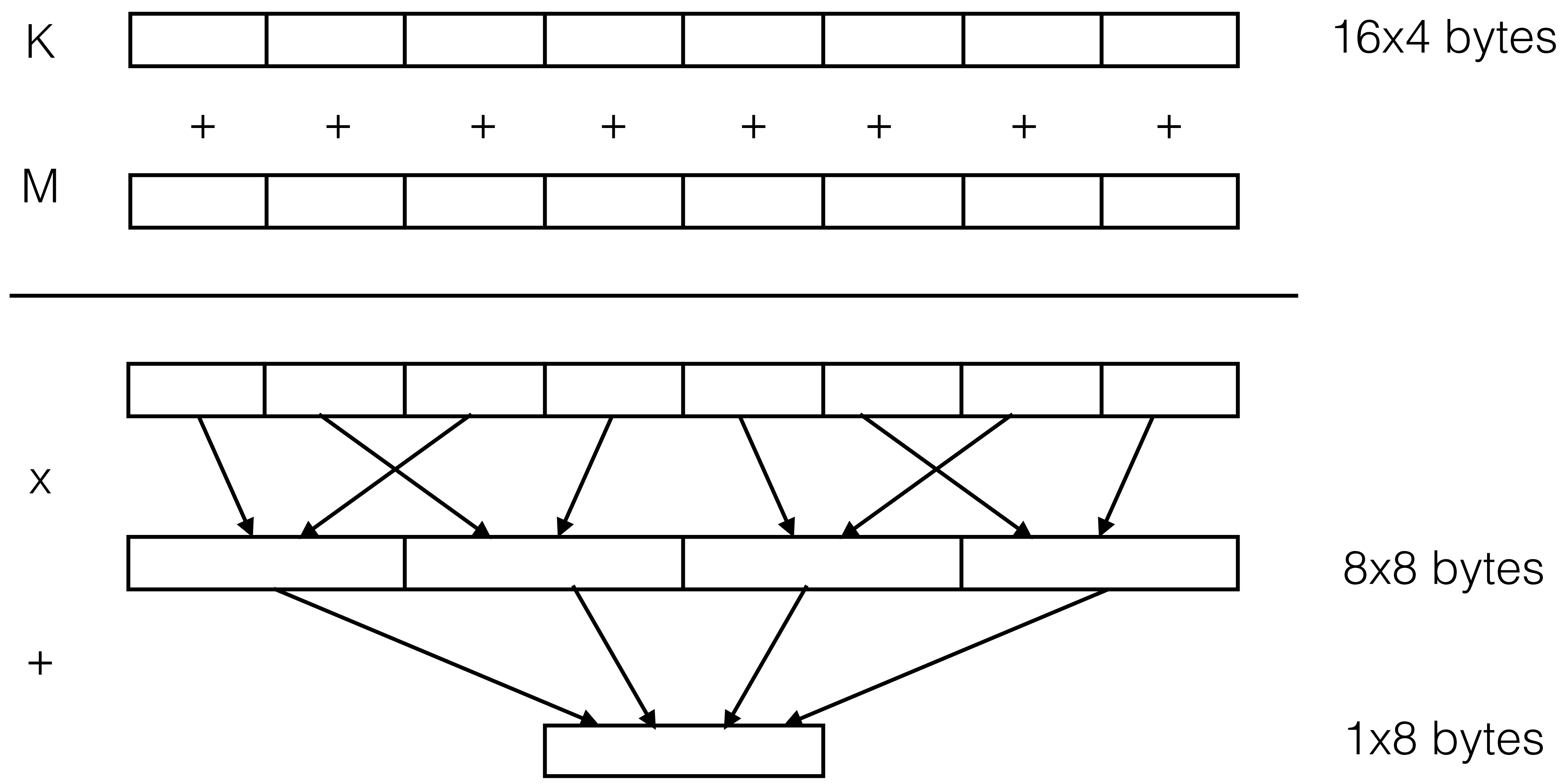
PRF-Only Version

- Let F be a VIL+VOL PRF
- SIV = F(Nonce || AD || M) [1..16]
- C = F(Nonce || SIV) [64..] ⊕ M
- Send (Nonce, C, SIV)


PRF
PRF
N
N
K
K
M
(AD,M)
Universal Hash
Chacha$_0$
Universal Hash
Chacha$_1$
SIV
C
Variable
Fixed
Variable

Universal Hash: NH

[BHKKR99]



- 64:8 compression. 2^{-32} collision probability.
- Efficient on vector and scalar CPUs.

[BHKKR99]

Universal Hash: Poly61

- After compression do poly-eval hash to 8 bytes.

- a=1
  while (bytes remain)
      M = next 64 bytes
      a = ak + (NH(M) mod 2^{60}) mod 2^{61}-1

- $2^{-28} + m2^{-67}$ collision probability hashing m bytes.
  2^{-28} when messages are each < 2^{38} bytes.

- 2^{-112} when done 4 times.

Security Levels

	Hash	Cipher	SIV	Security
hs1-siv-lo	2 x	Chacha8	64 bits	56 bits
hs1-siv	4 x	Chacha12	128 bits	112 bits
hs1-siv-hi	6 x	Chacha20	256 bits	168 bits

- Security: Adversary wins if
  Hash collision — birthday bound on "Security"
  SIV collision — birthday bound on "SIV"
  Chacha failure — Adv^{prf}(Chacha)

- Assumption: Chacha core is a prf.

Targeting 32-bit Ops

- 64-bit CPUs perform 32-bit ops well in vectors.
- 32-bit CPUs don’t perform 64-bit ops well.
- Maximize 32-bit operations (NH-32, Chacha)
- Minimize 64-bit operations (64-bit mult is rare)
- Targeting 32-bits provides balanced performance.

Performance

	MIPS32	Cortex-A9	Haswell
4 x HS1 Hash	16 cpb	5 cpb	0.8 cpb
Chacha12	20 cpb*	7 cpb*	0.8 cpb*
AES128	60 cpb*	23 cpb*	9 cpb* (0.6HW)

- Preliminary: unoptimized HS1 Hash in C.

  Romain Dolbeau reports 2 cpb for 2^{-168} security on Haswell (6 x HS1 Hash + Chacha20) in C.

* As reported by SUPERCOP (http://bench.cr.yp.to)

Questions

OCB AE API

- API developed by Rogaway and Krovetz for OCB

ae_encrypt(ctx, n, pt, ad, final) : returns ct

n	ad	final	Meaning
Yes	Yes	True	All-in-one
Yes	No	True	Reuse AD
Yes	Yes / No	False	New Incremntal
No	—	False	Continue
No	—	True	Finalize