Accelerating AES

with Vector Permute Instructions

Mike Hamburg

Stanford University

September 7, 2009

Fast AES Implementations

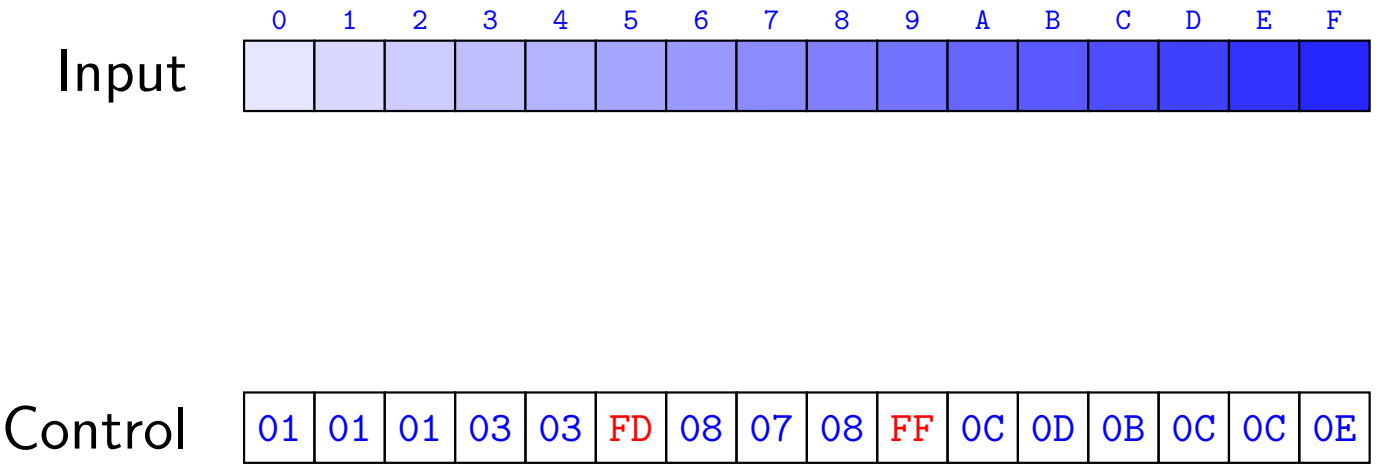
- Lookup table for (MixColumns $\circ$ S-box) [DR '99, . . . , BS '08]
  - Vulnerable to cache-timing attacks
- Composite fields for hardware [R '00, RDJKRR '02]
- Bitslicing [B '97, . . . , KS '09]
  - Very fast when encrypting ≥ 8 blocks in parallel

Fast AES Implementations

- Lookup table for (MixColumns ∘ S-box) [DR '99, . . . , BS '08]
  - Vulnerable to cache-timing attacks
- Composite fields for hardware [R '00, RDJKRR '02]
- Bitslicing [B '97, . . . , KS '09]
  - Very fast when encrypting ≥ 8 blocks in parallel
- Today: composite fields and vector permutations
  - Fast even for only one block

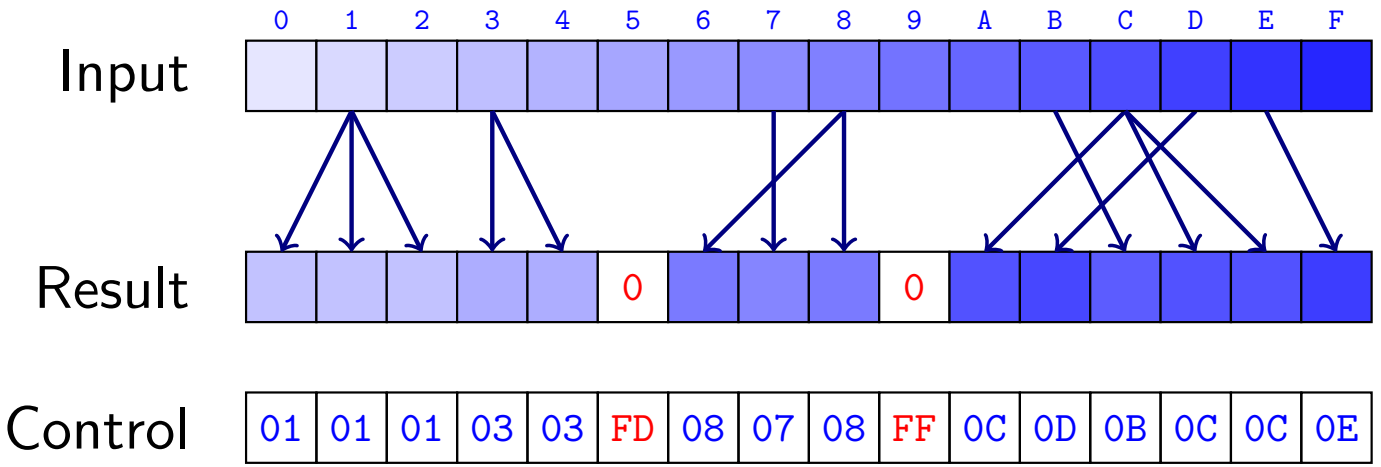
Vector permute instructions

- Available on AltiVec, ARM NEON, SSSE3



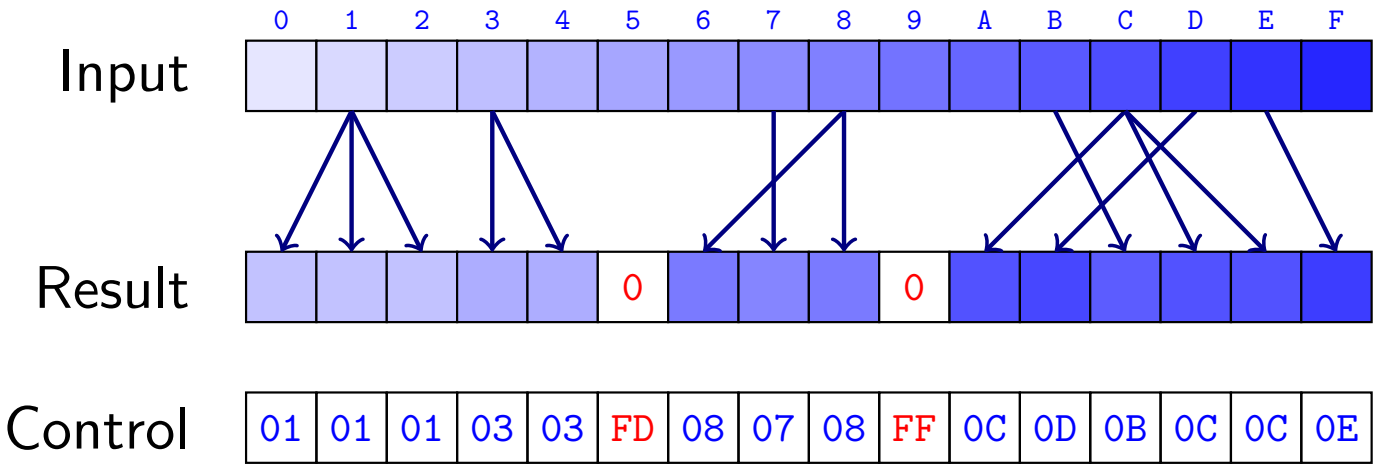
Vector permute instructions

- Available on AltiVec, ARM NEON, SSSE3



Vector permute instructions

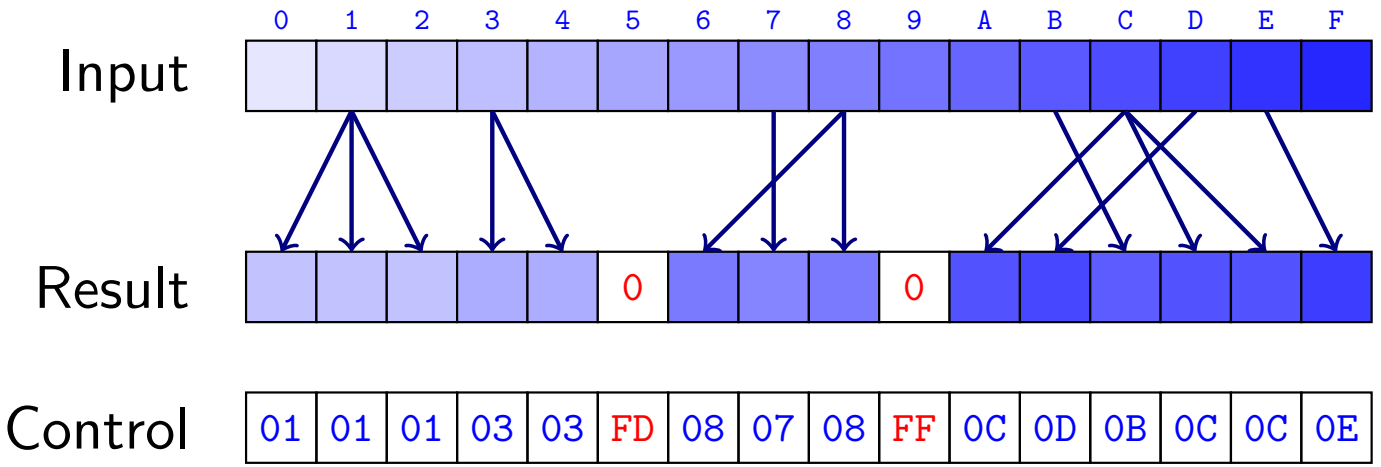
- Available on AltiVec, ARM NEON, SSSE3



- If control register is constant, acts as a permutation (ish)
- If input register is constant, acts as 16 lookups in parallel

Vector permute instructions

- Available on AltiVec, ARM NEON, SSSE3



- If control register is constant, acts as a permutation (ish)
- If input register is constant, acts as 16 lookups in parallel
- More powerful on AltiVec

Composite Fields

- AES uses inversion over $\mathbb{F}_{2^8}$
  - Too complicated to compute directly
- Decompose $\mathbb{F}_{2^8}$ as $\mathbb{F}_{2^4}[t]/(t^2 + t + \zeta)$
  - 16 elements – just right for vector permutes!
- Choice of basis is important!
  - We chose $(t,\ \bar{t} := t + 1)$

Computing the S-box

$$S(x) \quad = \quad \text{skew}\left(x^{254}\right) \; + \; \texttt{0x63}$$

Computing the S-box

$$S(x) = \text{skew}\left(x^{254}\right) + \texttt{0x63}$$

"free" with output tables

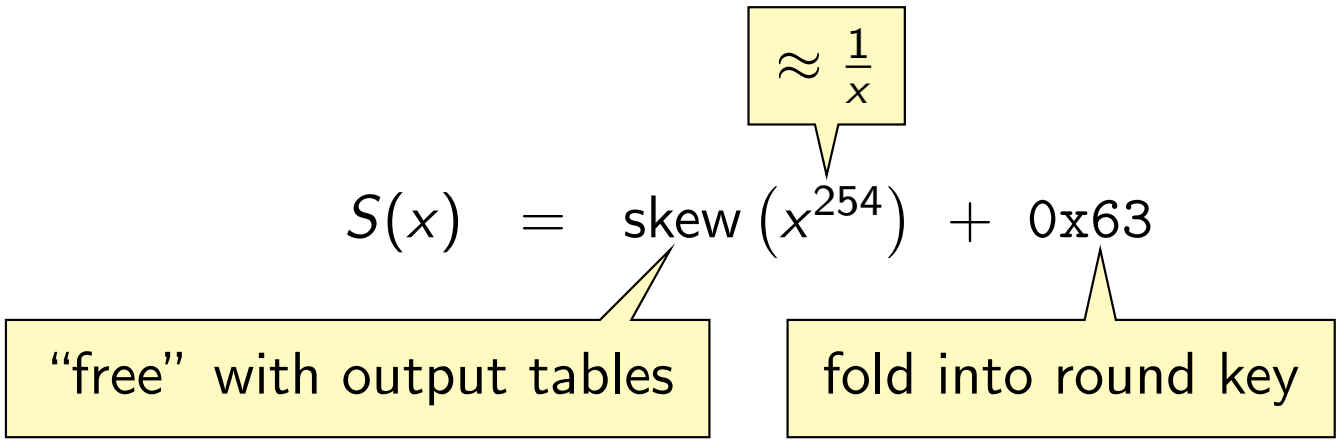
Computing the S-box

$$S(x) = \text{skew}\left(x^{254}\right) + \texttt{0x63}$$

$x^{254} \approx \frac{1}{x}$

"free" with output tables

Computing the S-box



Computing $\frac{1}{xt+y\bar{t}}$ — Classical Inversion

- Clear the denominator:

$$\frac{1}{xt + y\bar{t}} = \frac{x\bar{t} + yt}{xy + \zeta(x^2 + y^2)}$$

Computing $\frac{1}{xt+y\bar{t}}$ — Classical Inversion

- Clear the denominator:

$$\frac{1}{xt + y\bar{t}} = \frac{x\bar{t} + yt}{xy + \zeta(x^2 + y^2)}$$

- Actually implemented as:

$$\frac{x\bar{t} + yt}{\zeta(\sqrt{xy/\zeta} + x + y)^2}$$

Computing $\frac{1}{xt+y\bar{t}}$ — Classical Inversion

- Clear the denominator:

$$\frac{1}{xt + y\bar{t}} = \frac{x\bar{t} + yt}{xy + \zeta(x^2 + y^2)}$$

- Actually implemented as:

$$\frac{x\bar{t} + yt}{\zeta(\sqrt{xy/\zeta} + x + y)^2}$$

- Needs log tables!
  - Not enough space on x86

Computing $\frac{1}{xt+y\bar{t}}$ — Nested Inversion

- Multiply without multiplication:

$$\frac{xy}{x+y} = \frac{1}{\frac{1}{x} + \frac{1}{y}}$$

Computing $\frac{1}{xt+y\bar{t}}$ — Nested Inversion

- Multiply without multiplication:

$$\frac{xy \ + \ \zeta(x^2+y^2)}{(1+\zeta)x \ + \ \zeta y} \ = \ \frac{1}{\frac{1}{x} \ + \ \frac{1}{\zeta(x+y)}} \ + \ y$$

Computing $\frac{1}{xt+y\bar{t}}$ — Nested Inversion

- Multiply without multiplication:

$$\frac{1}{xt+y\bar{t}} = \cfrac{t+\zeta}{\cfrac{1}{\frac{1}{y}+\frac{1}{\zeta(x+y)}}+x} + \cfrac{\bar{t}+\zeta}{\cfrac{1}{\frac{1}{x}+\frac{1}{\zeta(x+y)}}+y}$$

Computing $\frac{1}{xt+y\bar{t}}$ — Nested Inversion

- Multiply without multiplication:

$$\frac{1}{xt+y\bar{t}} = \cfrac{t+\zeta}{\cfrac{1}{\frac{1}{y}+\frac{1}{\zeta(x+y)}}+x} + \cfrac{\bar{t}+\zeta}{\cfrac{1}{\frac{1}{x}+\frac{1}{\zeta(x+y)}}+y}$$

- Not as bad as it looks!

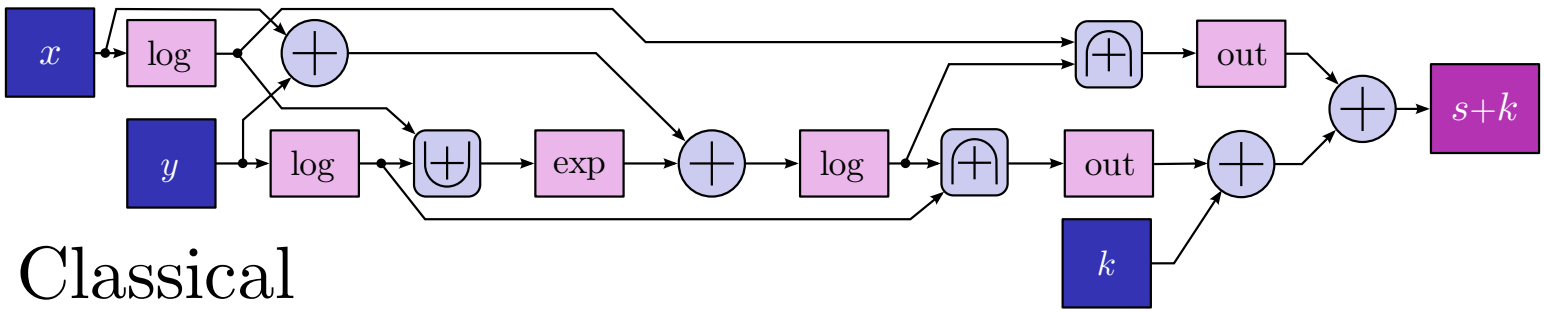
Computing $\frac{1}{xt+y\bar{t}}$ — Nested Inversion

- Multiply without multiplication:

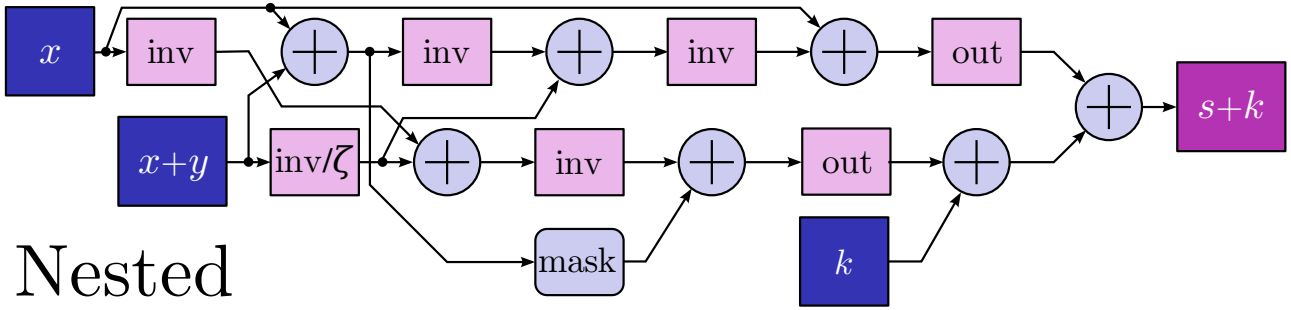
$$\frac{1}{xt+y\bar{t}} = \cfrac{t+\zeta}{\cfrac{1}{\frac{1}{y}+\frac{1}{\zeta(x+y)}}+x} + \cfrac{\bar{t}+\zeta}{\cfrac{1}{\frac{1}{x}+\frac{1}{\zeta(x+y)}}+y}$$

- Not as bad as it looks!
- Division by zero a problem
  - Use "∞ bit"

Computing $\frac{1}{xt+y\bar{t}}$ — Comparison



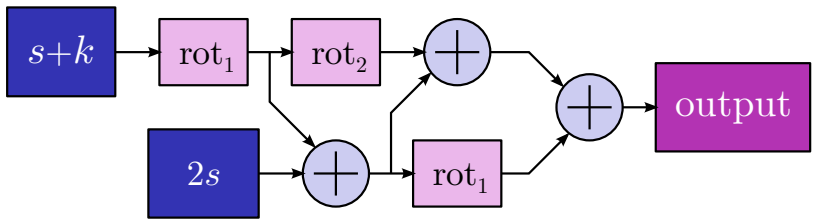
Classical



Nested

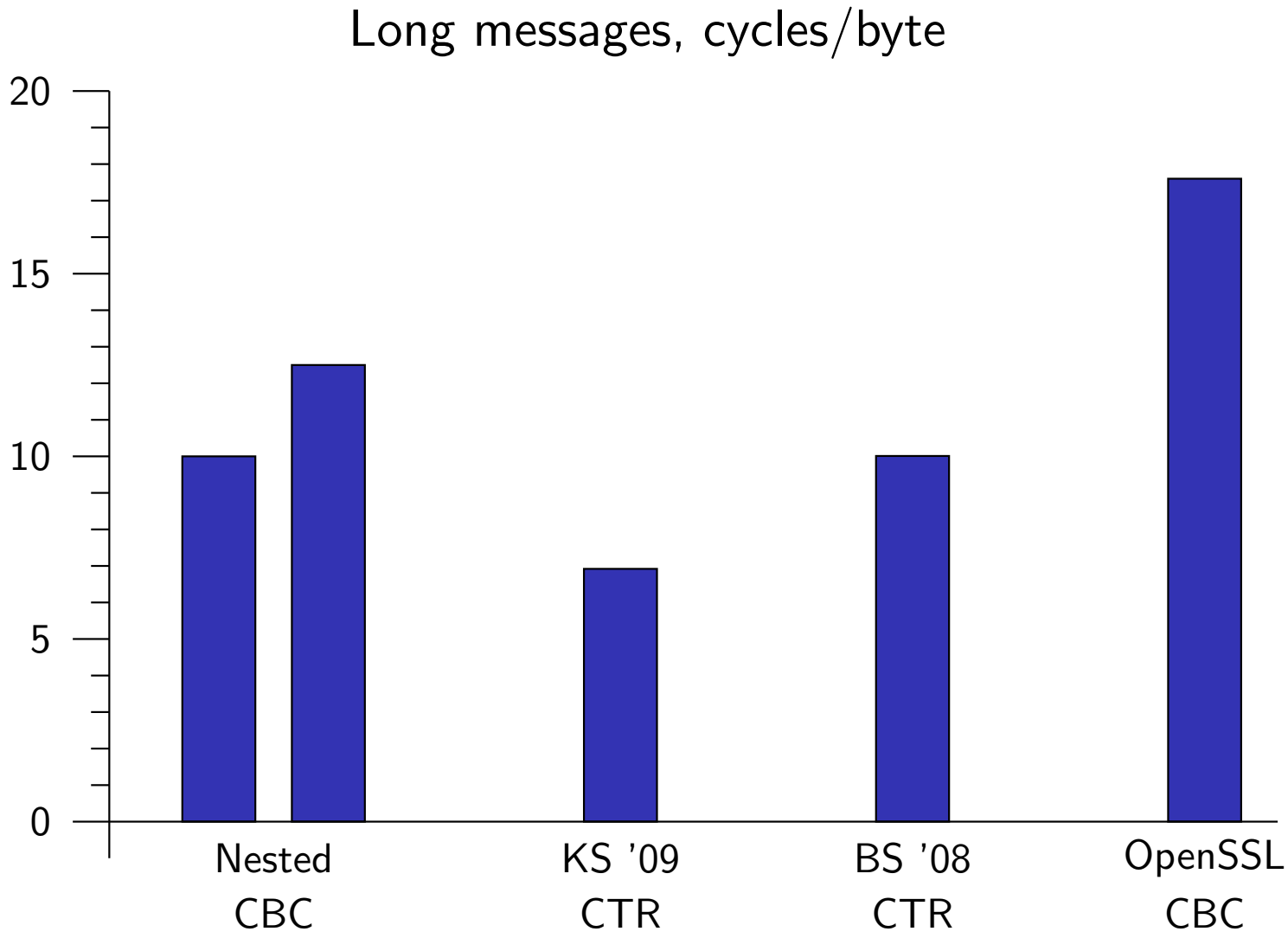
The Rest of the Cipher

- Don't compute ShiftRows
  - Fold into MixColumns permutations
- AddRoundKey before MixColumns
  - Modified key schedule
- MixColumns from $s + k$ and $2s$

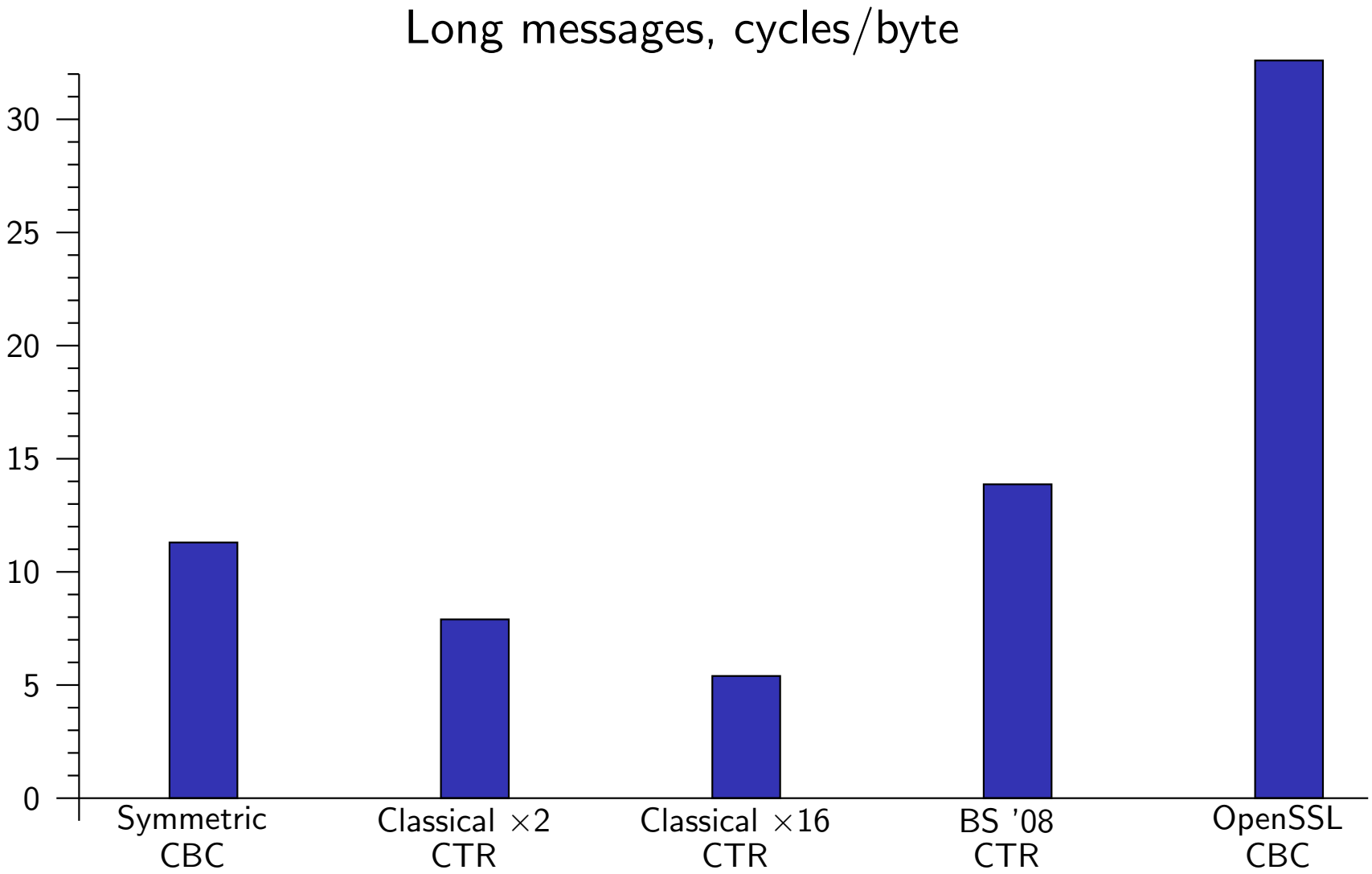


- Fold output basis shift into S-box

Performance – Nehalem



Performance – PowerPC G4e



Future Work

- Optimize more carefully for Intel
  - Up to 20% in microarchitectural tweaks
  - Up to 17% in CTR-mode caching
  - Byte-slicing?
- Implement other primitives
  - Camellia, LEX, Fugue, ...
- Package and distribute!

Fin

Questions?