Efficient Hashing using the AES Instruction Set

Joppe Bos[1] Onur Özen[1] *Martijn Stam*[2]

[1]Ecole Polytechnique Fédérale de Lausanne

[2]University of Bristol

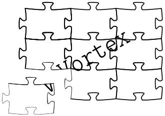
Nara, 1 October 2011

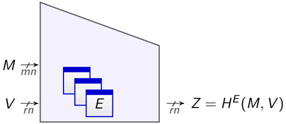
Outline

Outline

Motivation

AES-based vs. AES-instantiated Blockcipher-based





AES-Based Hashing [BBGR09]
(several SHA-3 candidates)

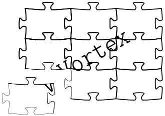
Use AES as a blackbox
(blockcipher-based hashing)

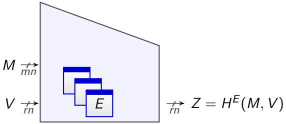
AES in a nutshell

- The US encryption standard (standardized by NIST in 2001)
- 128-bit block-size version of the Rijndael blockcipher (designed by Daemen & Rijmen)

Motivation

AES-based vs. AES-instantiated Blockcipher-based





AES-Based Hashing [BBGR09]
(several SHA-3 candidates)

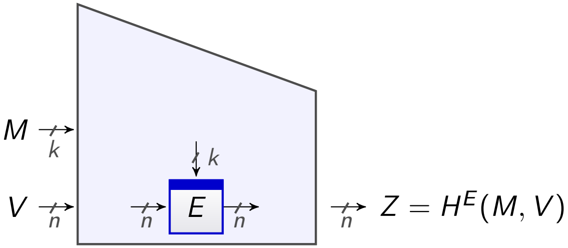
Use AES as a blackbox
(blockcipher-based hashing)

Why is this interesting?

1. AES-NI Instruction Set promises considerable speedup
2. Blockcipher-based hashing relatively well understood
   with many security proofs in ideal cipher model (ICM)

Blockcipher-Based Hashing

The principal idea

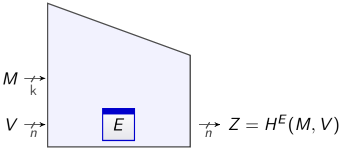


$$E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

- Blockcipher with k-bit key, operating on n-bit blocks.
- Compression function H^E from $n + k$ bits to n bits (input consists of k bits message and n bits chaining variable).

Blockcipher-Based Hashing

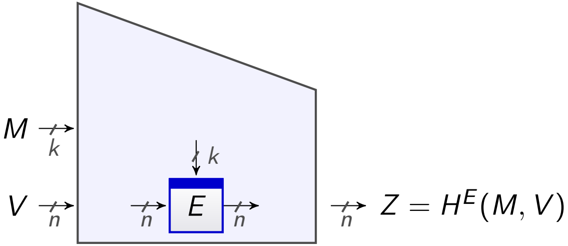
Using AES



Blockcipher E	**Block-size** n **(bits)**	**Key-size** k **(bits)**	**Number of Rounds**
AES-128	128	128	10
AES-256	128	256	14
Rijndael-256	256	256	14

Blockcipher-Based Hashing

The principal idea, revisited

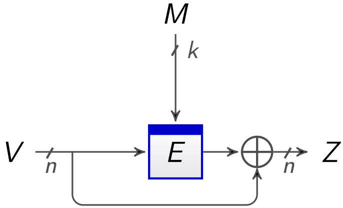


$$E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

- Examples include MD5, SHA family, plus the (generic) PGV compression functions.

Blockcipher-Based Hashing

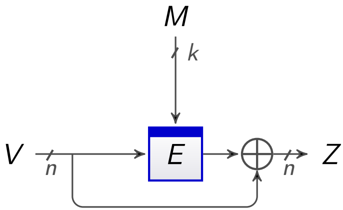
The principal idea, revisited



- Examples include MD5, SHA family, plus the (generic) PGV compression functions.
- For instance the Davies–Meyer construction.

Blockcipher-Based Hashing

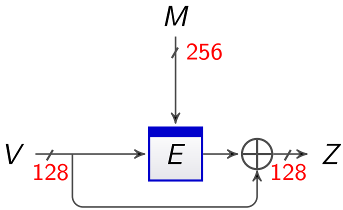
The principal idea, revisited



- Assuming E is ideal, Davies–Meyer is optimally collision resistant.

Blockcipher-Based Hashing

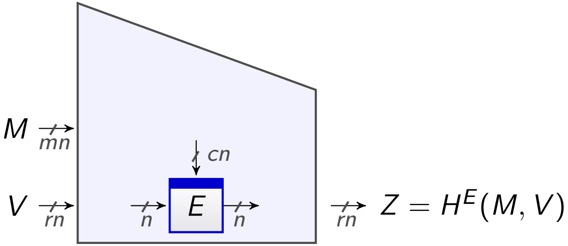
The principal idea, revisited



- Assuming E is ideal, Davies–Meyer is optimally collision resistant.
- When instantiated with e.g. AES-256,
  it takes 2^{64} operations to find a collision. Insufficient!

Blockcipher-Based Hashing

The principal idea, revisited

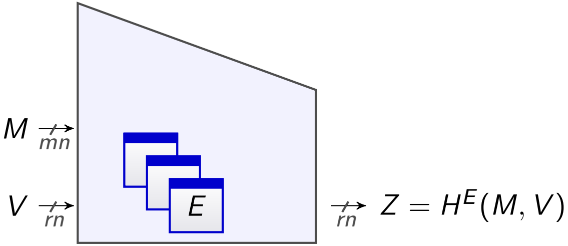


$$E : \{0,1\}^{cn} \times \{0,1\}^n \to \{0,1\}^n$$

- Blockcipher with *cn*-bit key, operating on *n*-bit blocks.
- Compression function H^E from $(r+m)n$ bits to rn bits (using multiple calls to E) where $r > 1$.

Blockcipher-Based Hashing

The principal idea, revisited

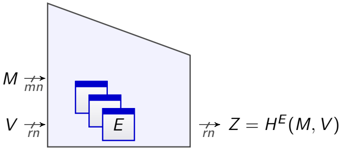


$$E : \{0,1\}^{cn} \times \{0,1\}^n \to \{0,1\}^n$$

- Blockcipher with cn-bit key, operating on n-bit blocks.
- Compression function H^E from $(r+m)n$ bits to rn bits (using multiple calls to E) where $r > 1$.

Blockcipher-Based Hashing

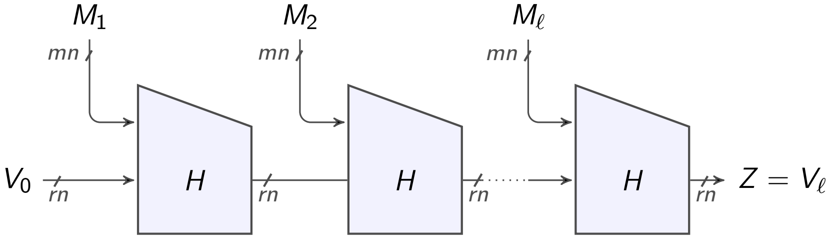
Using AES



Blockcipher *E*	**Block-size (bits)**	**Key-size (bits)**	**Number of Rounds**
AES-128	128	128	10
AES-256	128	256	14
Rijndael-256	256	256	14

Iterated Hash Functions

Merkle-Damgård Transformation



MD-Iteration

From $H : \{0,1\}^{(m+r)n} \to \{0,1\}^{rn}$ to $\mathcal{H}^H : (\{0,1\}^{mn})^* \to \{0,1\}^{rn}$

Multi-Block Length Blockcipher-Based Schemes

This Work: A Performance Comparison

Blockcipher	**Variable-key Constructions**	**Fixed-key Constructions**
AES-128	MDC-2, MJH, Peyrin et al.(I)	LP362
AES-256	Abreast-DM, Hirose-DBL, Knudsen–Preneel, MJH-Double, QPB-DBL, Peyrin et al.(II)	n.a.
Rijndael-256	Davies–Meyer	LP231, LANE*, Luffa*, Shrimpton–Stam

Related Key Attacks (RKA) on AES

The ugly

A formal definition of related key attacks [BK03,AFPW11]

Related Key Attacks (RKA) on AES

The ugly

A formal definition of related key attacks [BK03,AFPW11]

The bad

AES-192 and AES-256 are susceptible to meaningful RKA [BK09,BKN09]

- Casts doubt on modelling AES-192 and AES-256 as ideal ciphers.
- Davies–Meyer[AES-256] fails optimal security for certain beyond-birthday properties.

Related Key Attacks (RKA) on AES

The ugly

A formal definition of related key attacks [BK03,AFPW11]

The bad

AES-192 and AES-256 are susceptible to meaningful RKA [BK09,BKN09]

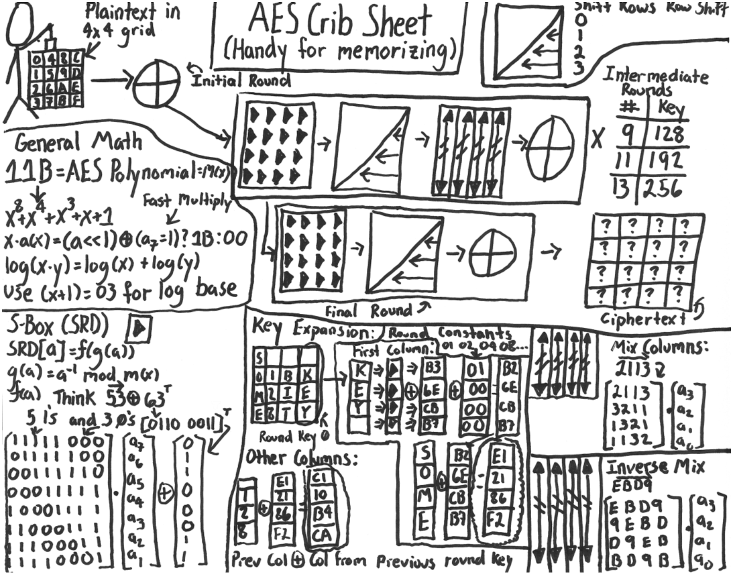
- Casts doubt on modelling AES-192 and AES-256 as ideal ciphers.
- Davies–Meyer[AES-256] fails optimal security for certain beyond-birthday properties.

The good

No identified weaknesses against any of the schemes considered in this talk

Outline

AES and Rijndael



(Created by Jeff Moser)

AES-NI

- **Goal:** Fast and secure AES encryption and decryption
- **Available Platforms:** Intel Westmere-based (2010) and Sandy Bridge processors (2011), AMD Bulldozer-based processors (2011)

Useful New AES Instructions

- `AESENC` performs a single round of encryption.
- `AESENCLAST` performs the last round of encryption.
- `AESKEYGENASSIST` is used for generating the round keys.

(For decryption available `AESDEC`, `AESDECLAST` and `AESIMC`)

Finally, `PCLMULQDQ` performs carry-less multiplication of two 64-bit operands to an 128-bit output.

Intel AES-NI Sample Library

For Intel Core i5 650 (3.2 GHz with AES-NI).

Blockcipher	**Key Schedule**	**1-Encryption (Seq. modes)**	**4-Encryption (Par. modes)**
	cycles (cycles/byte)		
AES-128	99.0 (6.2)	64.0 (4.0)	83.2 (1.3)
AES-256	124.5 (7.8)	86.4 (5.4)	108.8 (1.7)

Timing Modes of Encryption [G10,GK10,MMG10]

- Refers to CBC, ECB, etc.
- Intricate interleaving of `AESENC` calls.
- Key Scheduling is performed only once.
- Not included in the encryption timings.

AES-NI Timings for Hashing

Extensions (results in cycles, compiled using both gcc and icc)

Major Overhead: Frequent key-scheduling!

Blockcipher	1K	2K	3K	4K	1E	2E	3E	4E
AES-128	97.7	126.1	163.4	226.7	60.2	60.6	67.7	84.7
AES-256	125.5	147.2	202.6	287.2	82.0	83.0	93.6	113.9
Rijndael-256	291.6	316.6	412.6	570.3	182.9	219.2	281.4	352.6

AES-NI Timings for Hashing

Extensions (results in cycles, compiled using both gcc and icc)

Major Overhead: Frequent key-scheduling!

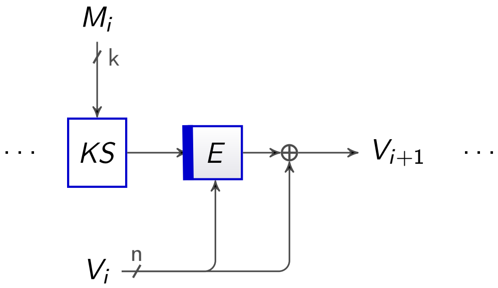
Blockcipher	1K	2K	3K	4K	1E	2E	3E	4E
AES-128	97.7	126.1	163.4	226.7	60.2	60.6	67.7	84.7
AES-256	125.5	147.2	202.6	287.2	82.0	83.0	93.6	113.9
Rijndael-256	291.6	316.6	412.6	570.3	182.9	219.2	281.4	352.6

Blockcipher	1K1E	2K2E	3K3E	4K4E	1K2E	1K3E	1K4E
AES-128	107.4	149.2	200.0	269.9	120.1	135.3	137.8
AES-256	152.8	178.1	249.7	337.9	154.0	158.4	164.9
Rijndael-256	285.3	407.5	620.5	867.3	312.0	373.3	463.7

Outline

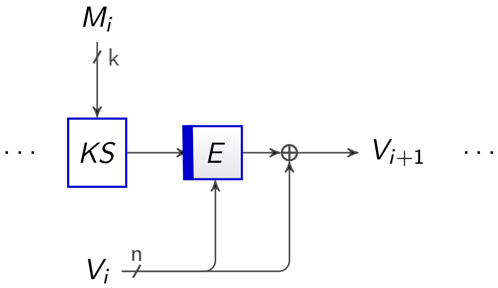
Davies–Meyer

Using Rijndael-256, $n = k = 256$



Davies–Meyer

Using Rijndael-256, $n = k = 256$

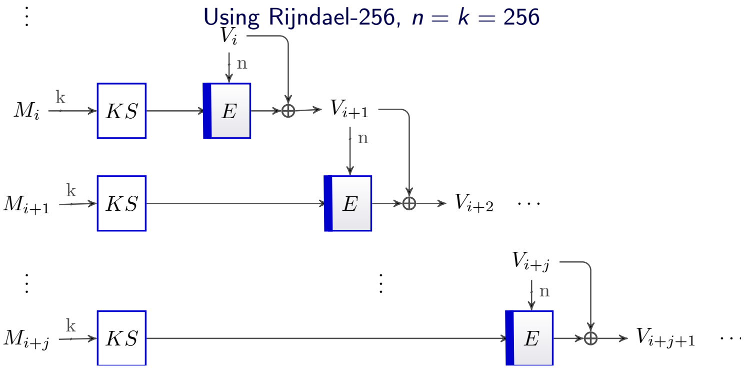


Conventional Implementation

- Requires one key-schedule and one encryption call (possibly round functions interleaved for each call).
- The performance can be estimated with **1K1E**.

Davies–Meyer

Using Rijndael-256, $n = k = 256$

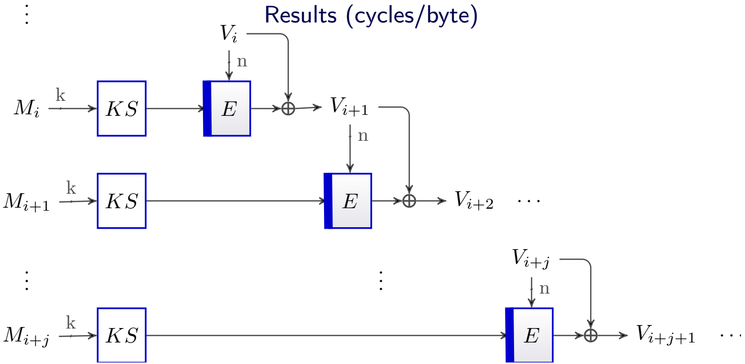


Optimized Implementation (for MD-iteration)

- Run the j key-schedules in parallel followed by j encrpytion calls.
- $j = 4$ gives the most efficient result.
- The performance can be estimated to be in [**4K4E**,**4K**+**4**×**1E**].

Davies–Meyer

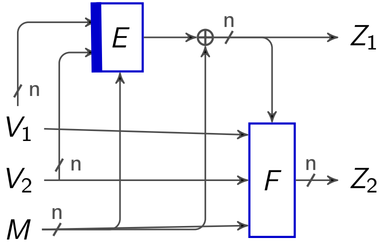
Results (cycles/byte)



Compression Function	Conventional		Optimized	
	Estimate	Achieved Speed	Estimate	Achieved Speed
Davies–Meyer	8.9	8.9	[6.8, 10.2]	8.7

Quadratic-Polynomial-Based DBL

Using AES-256



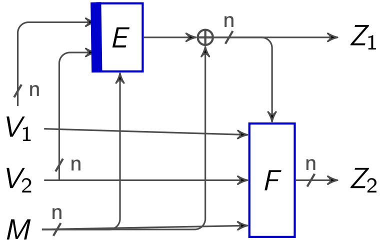
$$F(M, V_1, V_2, Z_1) = Z_1(V_2 Z_1 + V_1) + M$$

Evaluating F

- Requires on $GF(2^n)$ finite field multiplications.
- Relies on the PCLMULQDQ instruction.

Quadratic-Polynomial-Based DBL

Using AES-256



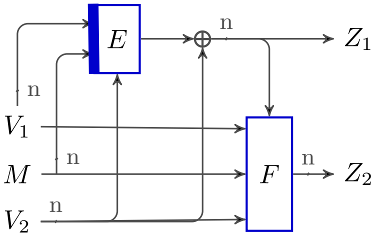
$$F(M, V_1, V_2, Z_1) = Z_1(V_2 Z_1 + V_1) + M$$

Conventional Implementation

- Calls the (full) compression function iteratively.
- Requires one key-schedule, one encryption call followed by two (full) finite field multiplications.
- The performance can be estimated with **1K1E**$+\epsilon$ where ϵ stands for the time required for multiplications.

Quadratic-Polynomial-Based DBL

Swapping the Inputs



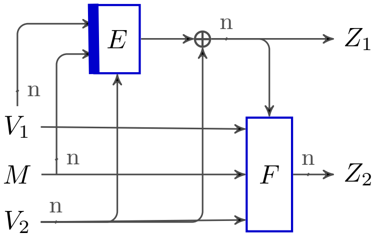
$$F(M, V_1, V_2, Z_1) = Z_1(V_1 Z_1 + M) + V_2$$

Optimized Implementation (for MD-iteration)

- Interleaves the key-scheduling of round $i+1$ with the two (sequential) finite field multiplications of round i.
- The predicted performance of QPB-DBL is based on the **1K1E**$+\epsilon$ setting where ϵ stands for the time required for multiplications.

Quadratic-Polynomial-Based DBL

Results (cycles/byte)



$$F(M, V_1, V_2, Z_1) = Z_1(V_1 Z_1 + M) + V_2$$

Compression Function	Conventional Estimate	Conventional Achieved Speed	Optimized Estimate	Optimized Achieved Speed
QPB–DBL	$9.5 + \epsilon$	15.8	$9.5 + \epsilon$	14.1

Our Timings

(cycles/byte)

Algorithm	Building Block	Key Scheduling	Predicted Speed Range	Achieved Speed
Abreast-DM	AES-256	two	$11.1 + \epsilon$	11.21
DM	Rijndael-256	one	$[6.8, 10.2]$	8.69
Hirose-DBL	AES-256	one, shared	9.6	9.82
Knudsen–Preneel	AES-256	four	10.6	10.58
LANE*	Rijndael-256	fixed	11.7	11.71
LP231	Rijndael-256	fixed	$12.6 + \epsilon$	13.04
LP362	AES-128	fixed	$11.8 + \epsilon$	12.09
Luffa*	Rijndael-256	fixed	$8.8 + \epsilon$	10.22
MDC-2	AES-128	two	$[9.3, 11.7] + \epsilon$	10.00
MJH	AES-128	one, shared	$6.6 + \epsilon$	7.45
MJH-Double	AES-256	one, shared	$4.1 + \epsilon$	4.82
QPB-DBL	AES-256	one	$9.5 + \epsilon$	14.12
Peyrin et al.(i)	AES-128	three, shared	$[12.5, 16.3]$	15.09
Peyrin et al.(ii)	AES-256	three, shared	$[7.8, 10.7]$	8.75
Shrimpton–Stam	Rijndael-256	fixed	12.6	12.39

Outline

Conclusion

For Intel Core i5 650 (3.2 GHz with AES-NI).

1. Fast instantiations of provably secure bc-based hash functions, using AES-NI achieving between 4 and 15 cycles per byte. (vs. SHA-256: 13.90 and SHA-512: 10.47).
2. MJH-Double is the overall speed champion (but its concrete security bound is lacking).
3. For **blockcipher-based** compression functions, DM is the fastest algorithm with optimal security
4. In the **permutation-based** setting, the fastest is Luffa*.
5. Slightly changing the compression function can lead to performance benefits without sacrificing provable security.