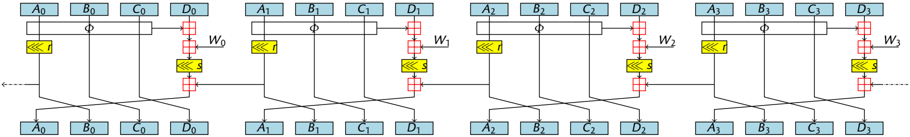
SIMD Is a Message Digest

Gaëtan Leurent, Pierre-Alain Fouque, Charles Bouillaguet

École Normale Supérieure
Paris, France

`http://www.di.ens.fr/~leurent/simd.html`

First SHA-3 Conference



Main Features of SIMD

- Security
  - Strong message expansion
  - Proof of security against differential cryptanalysis
- Parallelism
  - Small scale parallelism (inside the compression function): good for hardware / software with SIMD instructions
  - Can use two cores: message expansion / compression
- Performance
  - Very good on high-end desktops: 11 cycles/byte on Core2
  - Good if SIMD instructions are available: *SSE* on x86, *AltiVec* on PowerPC, *IwMMXt* on ARM,*VIS* on SPARC...
  - Drawback: no portable efficient implementation.

General Design

Merkle-Damgård-like iteration

Davies-Meyer-like compression function

Feistel-based block cipher

Two versions:

	Message block size m	Internal state size p
SIMD-256	512	512
SIMD-512	1024	1024

can be truncated (*e.g.* SIMD-224, SIMD-384)

Outline

Introduction

Description

- Mode of operation
- Compression Function
- Message Expansion

Security

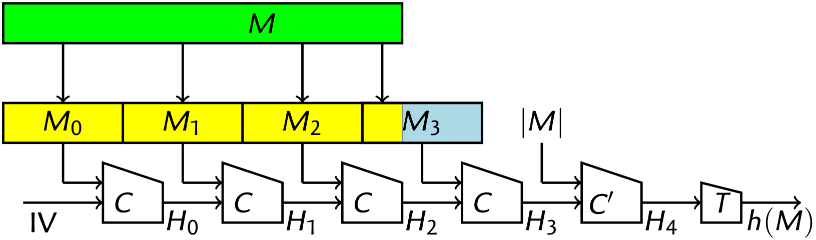
- Resistance to Differential Cryptanalysis

Implementation

- Performance

Iteration mode

The iteration mode is based on ChopMD (*a.k.a.* wide pipe).

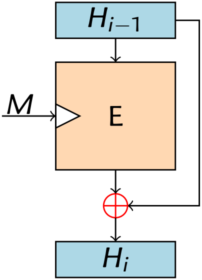


- Pad with zeros
- Use the message length as input of the last block: quite constrained, kind of blank round
- Tweaked final compression function (*i.e.* prefix-free encoding)
- Security proof: indifferentiable up to 2^n

How to build a compression function?

Two inputs: H_{i-1} hard to control / M easy to control

Davies-Meyer:

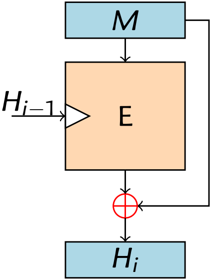


$$H_i = E_M(H_{i-1}) \oplus H_{i-1}$$

differential attack on C

$\rightsquigarrow$ related key attack on E

Matyas-Meyer-Oseas:



$$H_i = E_{H_{i-1}}(M) \oplus M$$

differential attack on C

$\rightsquigarrow$ differential attacks E

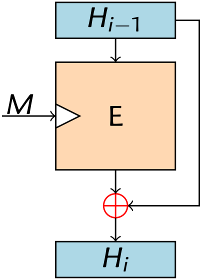
Message expansion

can reduce control over M

How to build a compression function?

Two inputs: H_{i-1} hard to control / M easy to control

Davies-Meyer:



$$H_i = E_M(H_{i-1}) \oplus H_{i-1}$$

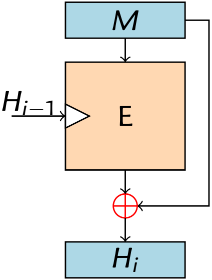
differential attack on C

related key attack on E

Message expansion

can reduce control over M

Matyas-Meyer-Oseas:

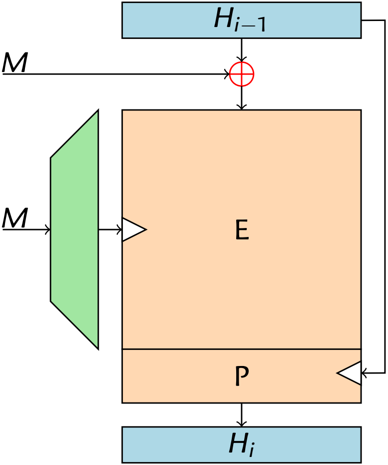


$$H_i = E_{H_{i-1}}(M) \oplus M$$

differential attack on C

differential attacks E

The Compression Function



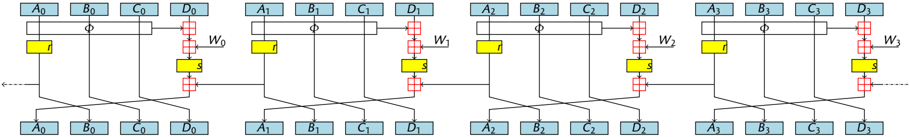
Modified Davies-Meyer mode.

- XOR M in the beginning: *no message modifications*
- Use some more Feistel rounds as the feed-forward: *avoids some fixed points and multiblock attacks*
- Same security proofs as DM: *good if E if good*

Feistel-based cipher

Strong message expansion

The Feistel Round



- 4 parallel Feistel ladders (8 for SIMD-512) with 32 bit words
- 4 (expanded) message words enter each round
- Interaction between the Feistel ladders via the permutations $p^{(i)}$
- Constants hidden in the message expansion

$$A_j^{(i)} = \left(D_j^{(i-1)} \boxplus W_j^{(i)} \boxplus \phi^{(i)}(A_j^{(i-1)}, B_j^{(i-1)}, C_j^{(i-1)})\right)^{s^{(i)}} \boxplus \left(A_{p^{(i)}(j)}^{(i-1)}\right)^{r^{(i)}}$$

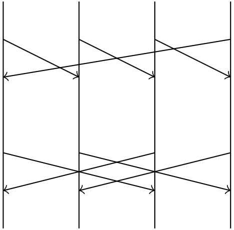
$$B_j^{(i)} = A_j^{(i-1)\;r^{(i)}} \qquad C_j^{(i)} = B_j^{(i-1)} \qquad D_j^{(i)} = C_j^{(i-1)}$$

Round Parameters

Rotations and
Boolean functions:

$\phi^{(i)}$	$r^{(i)}$	$s^{(i)}$
IF	π_0	π_1
IF	π_1	π_2
IF	π_2	π_3
IF	π_3	π_0
MAJ	π_0	π_1
MAJ	π_1	π_2
MAJ	π_2	π_3
MAJ	π_3	π_0

Permutations:
chosen for maximal diffusion



$p(j) = j + 1$

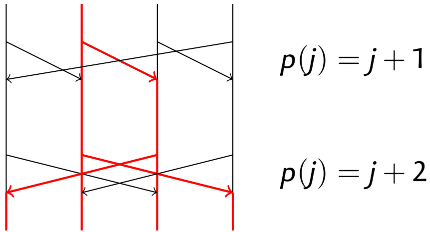
$p(j) = j + 2$

Round Parameters

Rotations and
Boolean functions:

$\phi^{(i)}$	$r^{(i)}$	$s^{(i)}$
IF	π_0	π_1
IF	π_1	π_2
IF	π_2	π_3
IF	π_3	π_0
MAJ	π_0	π_1
MAJ	π_1	π_2
MAJ	π_2	π_3
MAJ	π_3	π_0

Permutations:
chosen for maximal diffusion



The Message Expansion

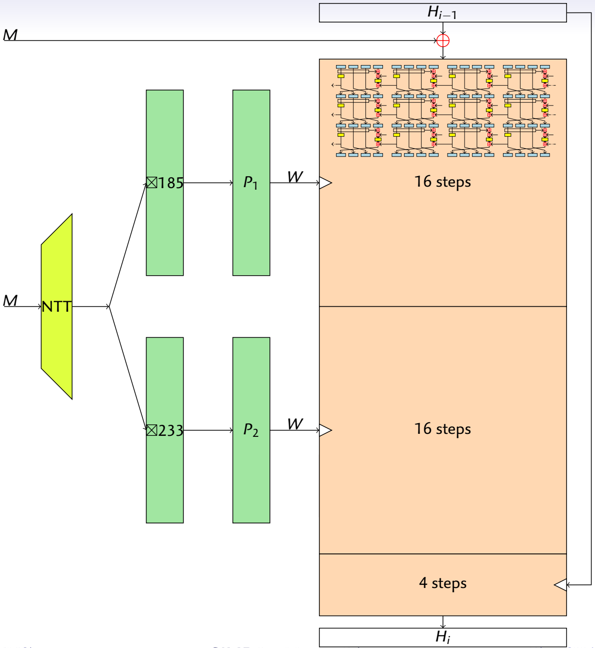
	Message block	Expanded message	Minimal distance
SIMD-256	512 bits	4096 bits	520 bits
SIMD-512	1024 bits	8192 bits	1032 bits

Provides resistance to differential attack

Based on (error correcting) codes with a good minimal distance

Concatenated code:

- outer code gives a high word distance
- inner code gives a high bit distance


H_{i-1}
M
M
NTT
⊠185
⊠233
P_1
P_2
W
W
16 steps
16 steps
4 steps
H_i

Outer Code

Reed-Solomon code

Interpret the input (k words) as a polynomial of degree $k-1$ over some finite field

Evaluate on n points ($n > k$)

MDS code: minimal distance $n-k+1$

	k	n	d
SIMD-256	64	128	65
SIMD-512	128	256	129

Efficiency:
- Compute with an FFT algorithm
- Use the field $\mathbb{F}_{257}$

Add a constant part: affine code

Inner code

We encode the output words of the NTT twice,
through two different inner codes.

Very efficient codes, with a single 16-bit multiplication.

$$
\begin{aligned}
I_{185} : \mathbb{F}_{257} &\mapsto \mathbb{Z}_{2^{16}} \\
x &\rightarrow 185 \boxtimes \widetilde{x} \qquad \text{where } -128 \leq \widetilde{x} \leq 128 \text{ and } \widetilde{x} = x \pmod{257} \\
I_{233} : \mathbb{F}_{257} &\mapsto \mathbb{Z}_{2^{16}} \\
x &\rightarrow 233 \boxtimes \widetilde{x} \qquad \text{where } -128 \leq \widetilde{x} \leq 128 \text{ and } \widetilde{x} = x \pmod{257}
\end{aligned}
$$

The magic constants 185 and 233 give a minimal distance of 4 bits.
(also for signed difference)

Security of SIMD

The mode of operation is indifferentiable.

No generic multicollision attack, second-preimage on long messages,
or herding attack

Any attack has to use some property of the block cipher.

The most obvious property is to find differential trails.

Security Proof: Attacker goal

We model a differential attacker:

Attacker game

Choose a message difference Δ

Build a differential path u $\quad$ v

Find a message M s.t. $(M, M + \Delta)$ follows the path

At each step there is a probability p that the path is followed
i.e. there are c conditions, $c = -\log_2(p)$.

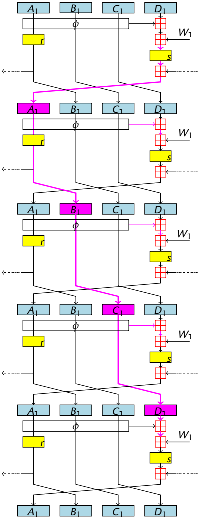
We want to show that $c \geq 128$.

Differential attacks

Two possible differentials:

- XOR difference: specifies which bits are modified
  - Easy to use
  - No condition for carry on bit 31
  - (limited number due to the inner code)
- Signed difference: specifies which bits go up or down
  - More powerful:
  - Used by Wang *et al.* to break MD4, MD5, SHA-1, HAVAL, ...
  - No condition when differences cancel out in $\boxplus$
  - Less conditions on the Boolean functions
  - Need a condition for the sign of bit 31

State Differences



We consider a single isolated difference bit in the state.

One condition to control the carry when the difference is introduced

Three conditions for the Boolean functions

Security Proof: Attacker game

We will ask the adversary to play an easier game:

Simplified adversary

- You have 520 differences in the expanded message (δW)
- You want to get rid of them by placing differences in the state (δA):
  - Each δA can consume some δW
  - But it costs you some conditions

The adversary is looking for a set of δA's with a good exchange rate.

He wins if the rate is less that $1/4$.

Adversary I: No control over the message differences

Adversary I

1. Choose a message difference of minimal weight
2. Find a path connecting the δW's

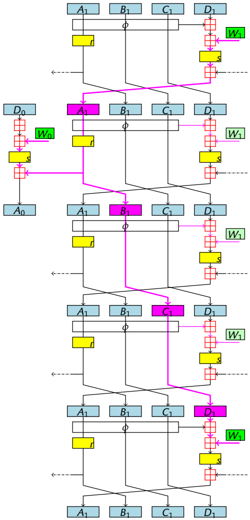
If the message difference has no other property,
Most of the δW will introduce a δA, *i.e.* 4 conditions.

Realistic if optimal message pairs (minimal weight difference) are hard to find.

Exchange rate: $4/1$. FAIL. ($p \approx 2^{-2048}$)

Lesson: the adversary need some control over the extended message.

Adversary II: Local Collisions



Adversary II

1. Choose a set δA
2. Use the neighbours of this δA as δW

If the state difference are isolated, $c \approx 4\delta A$.

Realistic if optimal message pairs are not so easy to find.

$\delta W \leq 6\delta A$
Exchange rate: $4/6$. FAIL. ($p \approx 2^{-340}$)

Lesson: the adversary needs to combine local collisions.

Adversary III: Combining Local Collisions

With a signed difference, many conditions can be avoided when two differences enters the same ϕ.

Exchange rate as low as $1/4.5$. WIN? ($p \approx 2^{-113}$)

We expect that it is impossible to choose a possible δW and a matching δA that achieve this exchange rate.

Can we prove it?

We modelled this game as a linear integer program.

The solver proved that there is no solution with less than 130 conditions (and counting).

Adversary III: Combining Local Collisions

With a signed difference, many conditions can be avoided when two differences enters the same ϕ.

Exchange rate as low as $1/4.5$. WIN? ($p \approx 2^{-113}$)

We expect that it is impossible to choose a possible δW and a matching δA that achieve this exchange rate.

Can we prove it?

We modelled this game as a linear integer program.

The solver proved that there is no solution with less than 130 conditions (and counting).

Adversary III: Combining Local Collisions

With a signed difference, many conditions can be avoided when two differences enters the same ϕ.

Exchange rate as low as $1/4.5$. WIN? ($p \approx 2^{-113}$)

We expect that it is impossible to choose a possible δW and a matching δA that achieve this exchange rate.

Can we prove it?

We modelled this game as a linear integer program.

The solver proved that there is no solution with less than 130 conditions (and counting).

Proof summary

The adversary:

- Chooses the message difference and
  the *expanded* message difference independently
- Can place the differences arbitrarily in the inner code
- Uses a signed diference

His optimal strategy:

- Use only local collisions (no error propagation)
- Locate the state differences next to each other
  to avoid most conditions.

Then, any differential path has at least 130 conditions.
(that includes pseudo-near-collision paths)

SIMD instructions

The NTT and the Feistel ladder can be parallelized using SIMD instructions.

Single Instruction, Multiple Data

A	1	2	3	4
B	5	5	5	5
$A+B$	6	7	8	9

Available on most architectures:

- *x86* MMX (64-bit registers), SSE (128-bit registers)
- *PPC* Altivec (128-bit registers)
- *ARM* IwMMXt (64-bit registers)
- *Sparc* VIS (64-bit registers)

Performance Overview

Message expansion *vs.* Feistel: 50/50

No need for 64-bit arithmetic

Efficient on some embedded architectures: ARM Xscale, x86 Atom

About 80% of the throughput of SHA-1 with a good SIMD unit (Core2, Atom, G4)

SIMD units are improved in each generation of processors

Performance in cycle/byte

Architecture		SHA-1/256/512			Scalar SIMD-256/512		Vector SIMD-256/512	
Core2	32 bits	11	21	63	90	118	12	13
	64 bits	9	16	13	63	85	11	12
K10	32 bits	12	18	64	80	125	17	
	64 bits	9	17	13	65	85	16	
P4	32 bits	19	89	147	170	210	32	43
K8	32 bits	12	19	65	90	135	25	
	64 bits	9	18	14	66	88	26	
Atom	32 bits	24	46	133	220	280	25	
G4	32 bits	12	23	78	125	166	16	23
ARM		22	38	138	200	260	46	

See eBASH for more accurate figures...

Conclusion

SIMD is

- Built on the MD/SHA legacy
- secure (mode of operation and compression function)
- Fast on the reference platform: 11-13 cycles/byte