The Hash Function JH

Hongjun Wu

Institute for Infocomm Research

Overview

1. Design objectives
2. JH compression function structure
3. The bijective function $\boldsymbol{E}_8$
4. Bit-slice implementation of $\boldsymbol{E}_8$
5. The hash function JH
6. The security analysis of JH
7. Contributions
8. Advantages of JH

1. Design Objectives

Hash function JH is design to achieve:

- strong security, with large security margin
- low cost security evaluation
- Efficient in hardware and software
  - use extremely simple structure and components

2. JH Compression Function Structure

$M^{(i)}$: m bits

$H^{(i)}$: $2m$ bits

New, simple

efficient

=> does not discard

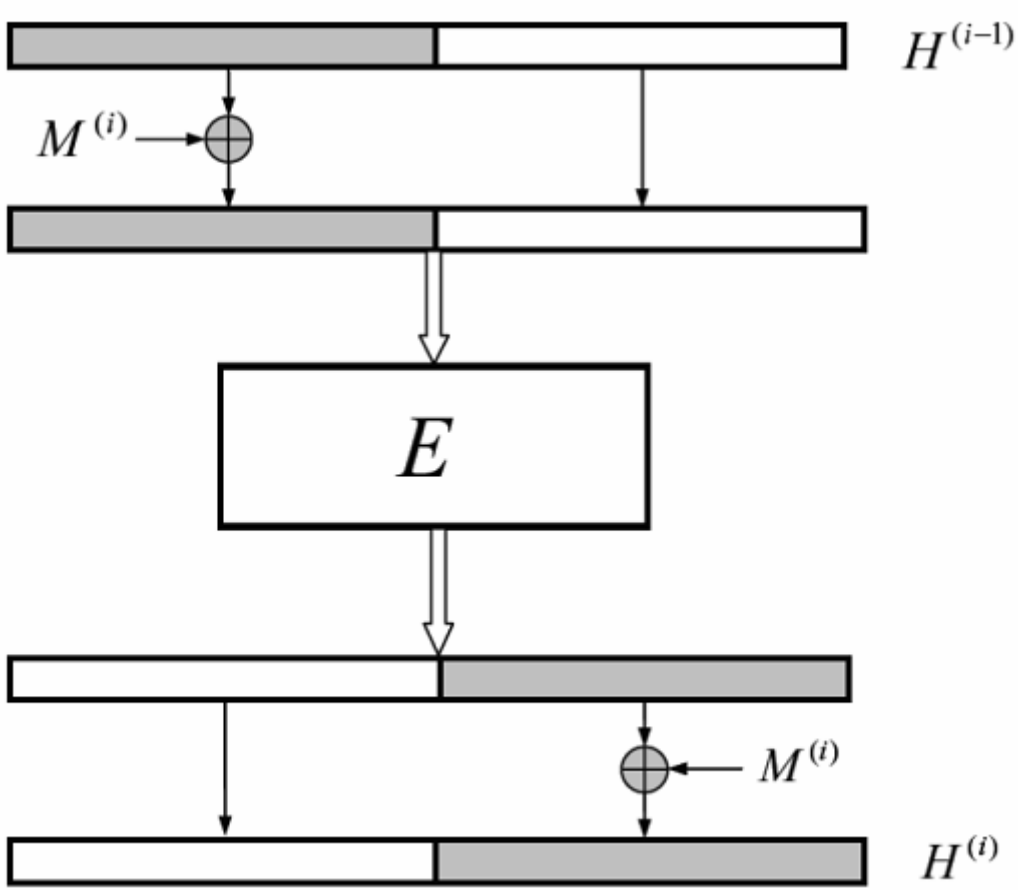
part of the output of ***E***

easy to analyze

=> no extra variables

being introduced into

the middle of ***E***



3. The Bijective Function E

Efficient Differential Propagation (EDP) design:

- SPN + MDS code (to a multi-dimensional array)
  - Confusion: Sbox layer
  - Diffusion: MDS code applied along the (i mod d)-th dimension in the i-th round (for a d-dimensional array)
- simple, efficient, easy to analyze

3. The Bijective Function *E*

- EDP is the generalization of the AES design
  - AES: SPN + MDS code applied to a two-dimensional array
    MDS applied to rows and columns alternatively

    (with row rotations, only apply MDS to columns)

    identical round functions => important for hardware efficiency

3. The Bijective Function E_8

Bijective function $\boldsymbol{E}_8$ –

EDP design: SPN + MDS code (to an 8-dimensional array)

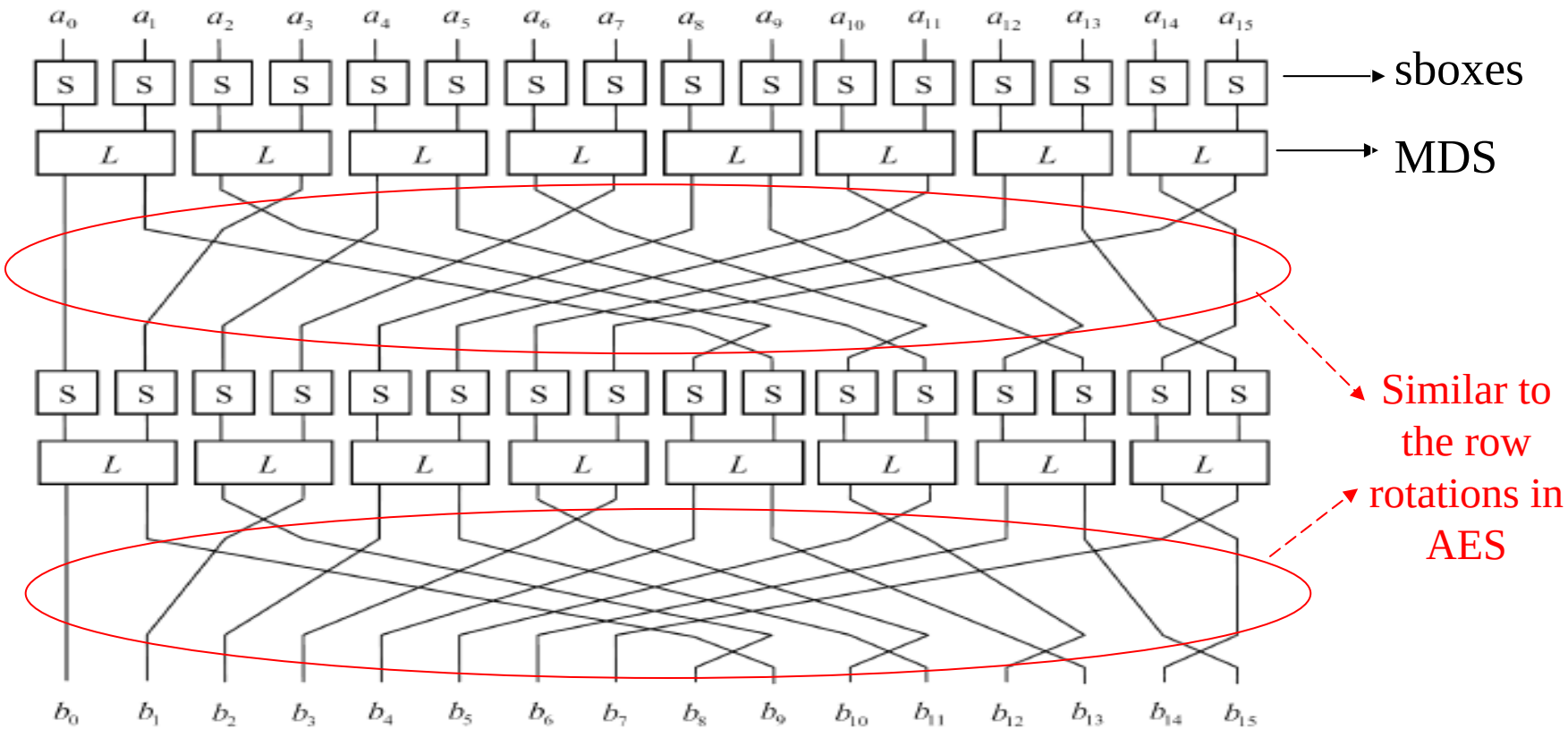
divide the 1024-bit input into 256 4-bit elements,
these 256 elements form an 8-dimension array

- Substitution: two 4-bit-to-4-bit Sboxes
  - Each round constant bit selects which Sbox is used
- Permutation: (4, 2, 3) MDS code over GF(2^4)
  - Applied along the ($\boldsymbol{i}$ mod 8)-th dimension in the $\boldsymbol{i}$-th round
- 35.5 rounds

3. The Bijective Function E_8

4-dimensional example $\boldsymbol{E}_4$ (two rounds, round constant not shown):

(identical round functions , except for different round constants)



3. The Bijective Function E_8

Sbox layer:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_0(x)$	9	0	4	11	13	12	3	15	1	10	2	6	7	5	8	14
$S_1(x)$	3	12	6	13	5	7	1	9	15	2	0	4	11	10	14	8

Each round constant bit selects which Sbox is used (similar of Lucifer)

Note: round constant bits not XORed to the intermediate value

Reason: conservative, to obtain a more complicated algebraic system, the cost for selecting Sbox is about 4 cycles/byte

3. The Bijective Function E_8

- The Sboxes are designed to satisfy 8 security requirements ……
- The two Sboxes can be computed with 20 binary operations (considering ANDNOT as a binary operation)

3. The Bijective Function E_8

Simple (4, 2, 3) MDS code over GF(2^4) (using polynomial $x^4 + x + 1$)

$$(C\,,D) = L(A\,,B) = (5 \bullet A + 2 \bullet B\,,\, 2 \bullet A + B)$$

L can be computed with 10 XOR operations

3. The Bijective Function E_8

36 round constants, each round constant is 256-bit

C_0 is the integer part of $(\sqrt{2}-1)\times 2^{256}$

C_i is generated as $C_i = R_6(C_{i-1})$

where R_6 is the round function of the 6-dimensional bijective function with round constants of R_6 being set as 0.

4. Bit-slice Implementation of E_8

The bit-slice implementation of $\boldsymbol{E_8}$ makes full use of the 128-bit SIMD architecture (powerful SIMD is available on many platforms):

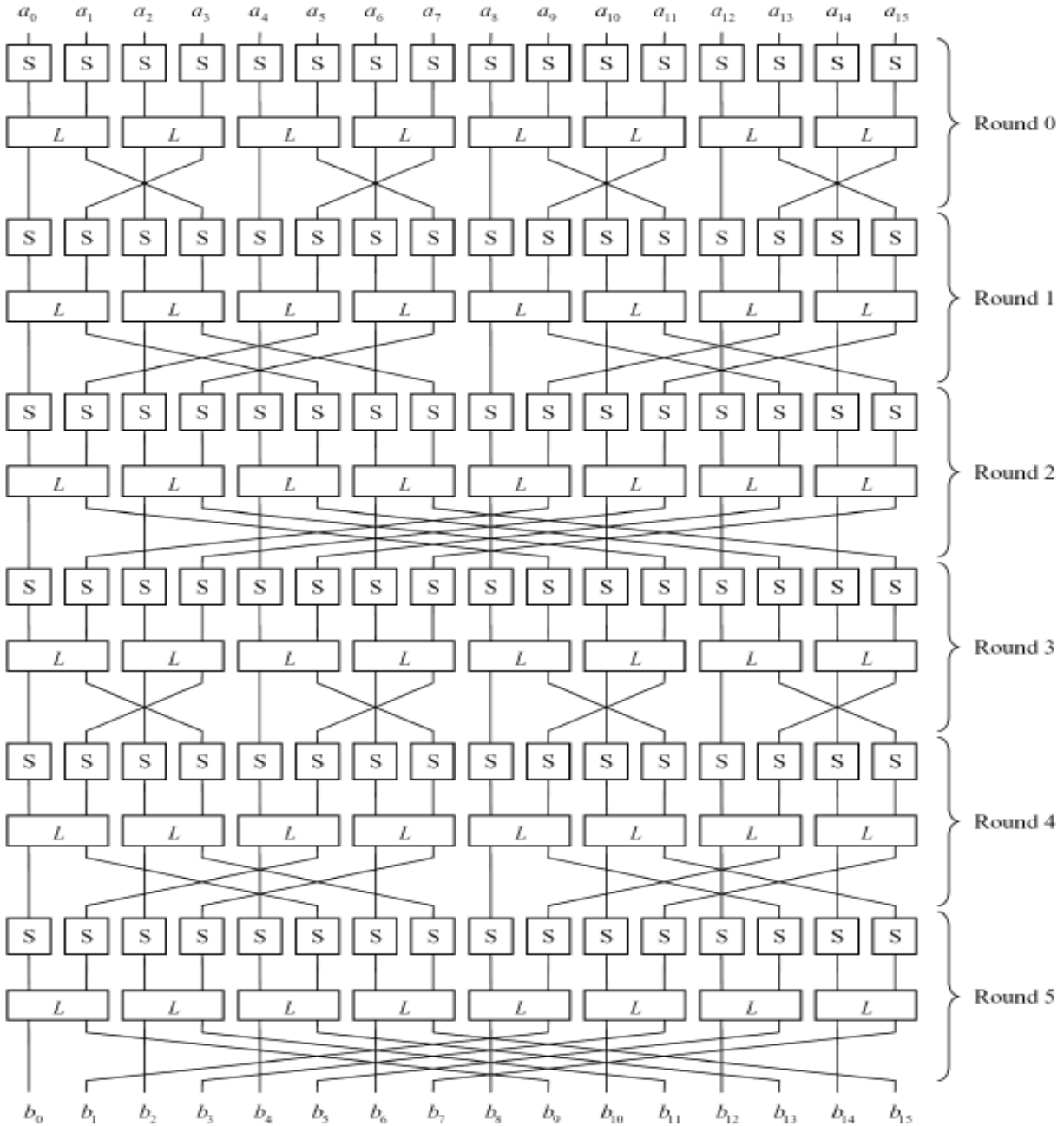
128 Sboxes can be computed in parallel
128 MDS codes can be computed in parallel

16.8 cycles/byte on 64-bit Core 2 processor,
21.3 cycles/byte on 32-bit Core 2 processor

Re-arrange the round function for bit-slice (4D example):

d-1 different round functions for bit slice;

identical round functions for hardware



5. The Hash Function JH

- Iterated construction
  - message block size: 512 bits
  - hash value: 1024 bits
- Pad the message with at least 512 bits (message length included)
- Different initial hash values for different digest sizes
- Truncate the 1024-bit final hash value to 224, 256, 384 or 512 bits to obtain message digest

6. Security Analysis of JH

It is easy to analyze the security of JH :

- simple compression function structure
  - no extra variables being introduced into the middle of the compression function
- SPN + MDS (to an 8-dimensional array)
  - low dimension function can be studied easily so as to analyze the high dimension function

6. Security Analysis of JH

Differential cryptanalysis

most powerful attack against hash function

a compression function in JH involves 9216 Sboxes.

any differential path in JH involves more than 600 active Sboxes, the large number of active Sboxes ensures that JH is strong against differential attack.

6. Security Analysis of JH

Differential cryptanalysis (contd.)

more than 600 active Sboxes,

effect of correlated active Sboxes:

each Sbox contributes $2^{-1.5}$ to the overall differential probability

=> the differential probability 2^{-900}

effect of message modification:

for collision search, even if assume that an attacker can control 16 rounds, there are still 336 active Sboxes

=> differential probability around 2^{-448} ;

much smaller than the required differential probability 2^{-256}

6. Security Analysis of JH

Algebraic Attack

- high order algebraic equations to thwart the direct algebraic attack and cube attack

- two Sboxes are selected by the random round constant bits

  => increase the complexity of the algebraic equations => increase the resistance against the future algebraic attack.

6. Security Analysis of JH

Security of JH :

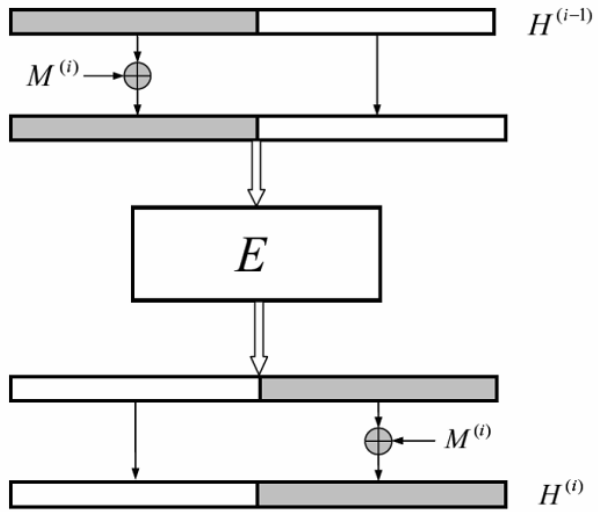
easy to analyze

large security margin

so far, no attack better than brute force

7. Contributions

- Proposed a new compression function structure
  - simple, efficient, easy to analyze



- Proposed EDP design-- the generalized AES design
  - SPN + MDS code (to a multi-dimensional array)
  - useful for designing block cipher with large block size
  - simple, efficient, easy to analyze

8. Advantages of JH

- Extremely simple structure and components
  - Simple compression function structure
  - EDP design : SPN + MDS
- Efficient on hardware and software
  - 16.8 cycles/byte on 64-bit Core 2 microprocessor,
  - 21.3 cycles/byte on 32-bit Core 2 microprocessor
- Easy to analyze
  - no variables being introduced into ***E***
  - SPN + MDS code to 8-dimension array
  - Easy to analyze the low dimensional function, then predict the security of the high dimensional function.
- Conservative design with large security margin

8. Advantages of JH

- JH will be more efficient on the incoming Intel microprocessors (2010)
  - Intel 256-bit Advanced Vector eXtensions (AVX), extension to SSE
  - 256 Sboxes can be computed in parallel

Thank you!

Q & A