JH 해시함수

정보컴퓨터공학과 권혁동

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

JH 해시함수

- 2011년에 제안된 알고리즘
- 내부에 **Generalized AES Design Methodology** 채택
  - Compress function으로 사용함
  - 블록 크기 64-byte
  - 8-dimentional
- 기존 해시 표준(SHA-256)과 동일 규격
  - JH-224, 256, 384, 512
- 병렬 구현 지원
  - SSE, AVX

JH 해시함수

강점

- 추가적인 변수가 필요하지 않음
  - 키 값이 상수로 되어있음
- 비트 슬라이스 구현이 가능함
  - 병렬 구현에 유리
- 차분 공격에 강함
  - 공격을 탐지하는데 능함

JH 해시함수

해시 과정

1. Padding
   - 메시지 M은 512-bit의 배수로 패딩
   - 규칙1: 메시지의 끝에 1을 붙이고 384 – 1 +(-l mod 512)개의 0을 삽입
   - 규칙2: 규칙1 이후로 128-bit로 표현된 l을 삽입
2. Parsing
   - 512-bit를 4개의 블록으로 파싱
   - 각 블록은 128-bit로 구성
3. Initialization
   - 초기 해시 $H^{(0)}$를 생성하기 위해 몇 가지 과정을 거침
   - $H^{(-1)}$: digest 길이 만큼 모든 값이 0으로 설정
   - $H^{(-1)}$을 Compress function에 통과시켜서 $H^{(0)}$ 생성
4. Finalization
   - $H^{(0)}$에서 정해진 횟수 만큼 compress를 반복하셔 해시 생성

JH 해시함수

- S-Box 구성
  - 두 종류의 S-Box를 사용
  - XOR로 구현이 가능하도록 설계
  - S-Box 보안성은 Lucifer 함수를 참고함

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

JH 해시함수

- Linear transformation L
- GF(2^4) 상에서 Maximum Distance Separable code 적용
  - MDS code는 데이터 인코딩 기법의 한 종류
- A, B, C, D 4-bit 워드가 있다고 가정할 시, L의 정의

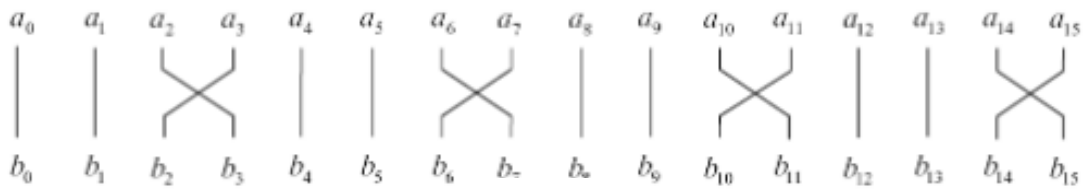
$$(C, D) = L(A, B) = (5 \bullet A + 2 \bullet B, 2 \bullet A + B).$$

- **비트 슬라이스 구현이** 가능함

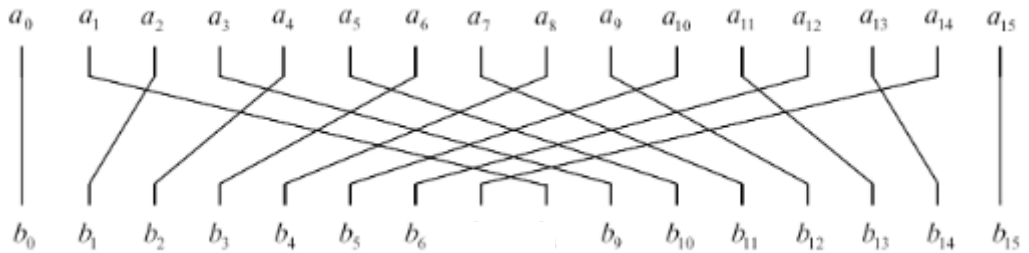
$$\begin{aligned}
D^0 &= B^0 \oplus A^1; & D^1 &= B^1 \oplus A^2; \\
D^2 &= B^2 \oplus A^3 \oplus A^0; & D^3 &= B^3 \oplus A^0; \\
C^0 &= A^0 \oplus D^1; & C^1 &= A^1 \oplus D^2; \\
C^2 &= A^2 \oplus D^3 \oplus D^0; & C^3 &= A^3 \oplus D^0.
\end{aligned}$$

JH 해시함수

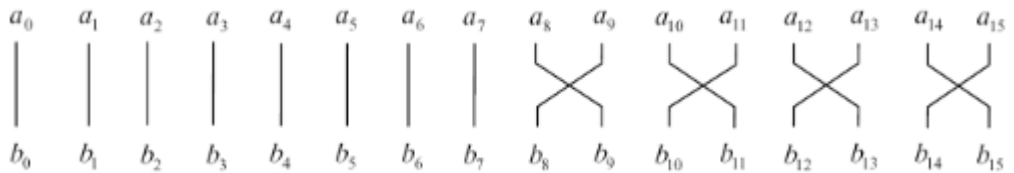
- Permutation
  - 4종류의 Permutation이 존재



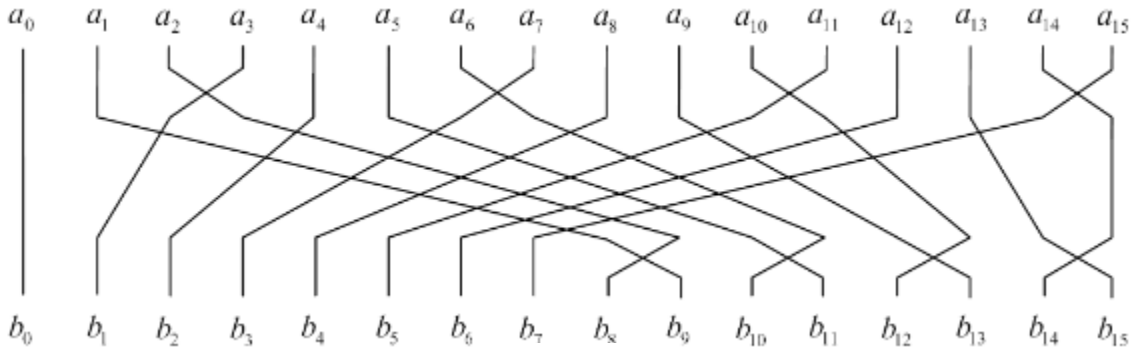
π_d



P'_d



ϕ_d



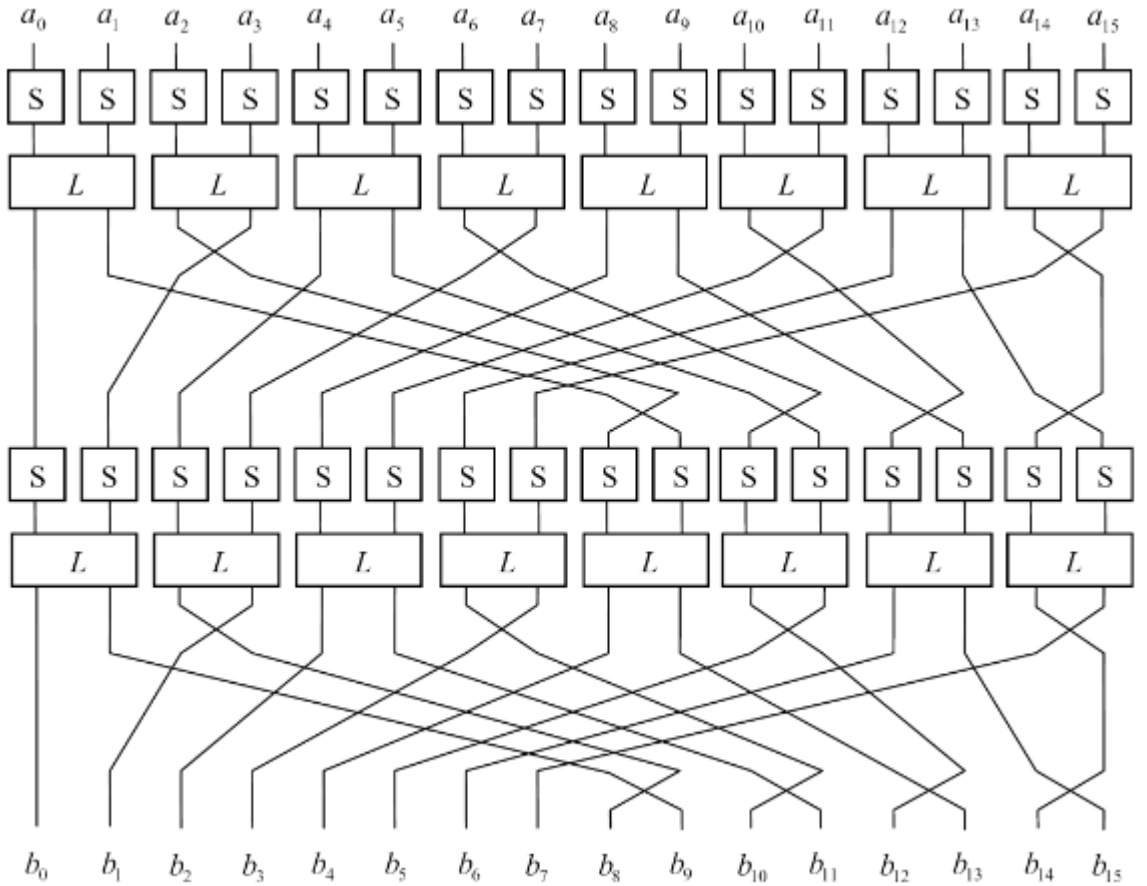
P_d

JH 해시함수

- Permutation
  - AES의 **Rotation row 과정과 거의 동일함**
  - 마찬가지로 AES의 디자인을 따왔기에 하드웨어 구현에도 적합
  - P_d는 다른 세 종류의 Permutation으로 구성됨
  - 따라서 세가지 Permutation만 구현하고 회로를 재사용하여 P_d구현 가능

JH 해시함수

- 라운드 함수 R_d의 구성
  - S, L, P로 구성 → AES와 유사



JH 해시함수

- 전단사 함수 E_d의 구성
  - 특정 라운드의 라운드 함수로 구성
  - Degroup은 마지막 단계에만 적용

$$\begin{aligned}&\text{for } i = 0 \text{ to } 2^{d-1} - 1,\\&\{\\&\quad q_{0,2i} = A^i \| A^{i+2^d} \| A^{i+2\cdot 2^d} \| A^{i+3\cdot 2^d};\\&\quad q_{0,2i+1} = A^{i+2^{d-1}} \| A^{i+2^{d-1}+2^d} \| A^{i+2^{d-1}+2\cdot 2^d} \| A^{i+2^{d-1}+3\cdot 2^d};\\&\}\end{aligned}$$

Grouping

$$\begin{aligned}&\text{for } i = 0 \text{ to } 2^{d-1} - 1,\\&\{\\&\quad B^i \| B^{i+2^d} \| B^{i+2\cdot 2^d} \| B^{i+3\cdot 2^d} = q_{6(d-1),2i};\\&\quad B^{i+2^{d-1}} \| B^{i+2^{d-1}+2^d} \| B^{i+2^{d-1}+2\cdot 2^d} \| B^{i+2^{d-1}+3\cdot 2^d} = q_{6(d-1),2i+1};\\&\}\end{aligned}$$

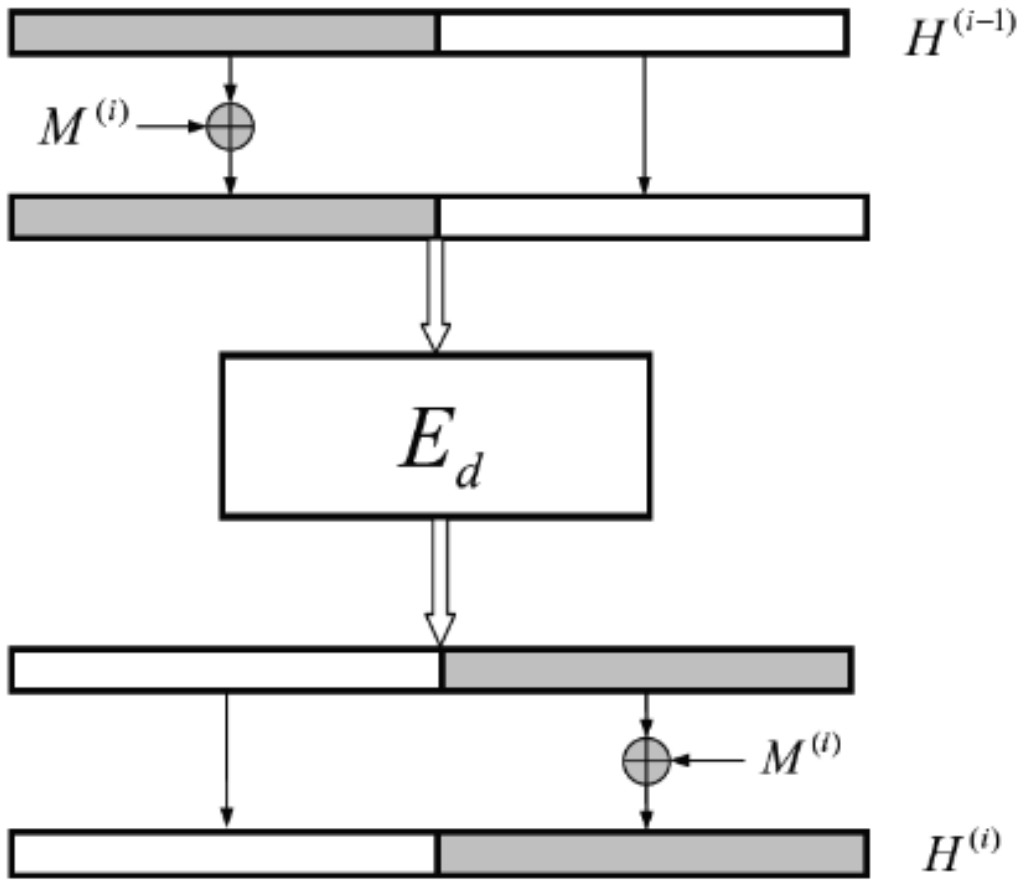
Degrouping

- 라운드 상수 C_d의 구성

$$\begin{aligned}&C_0^{(d)} \text{ is the integer part of } (\sqrt{2} - 1) \times 2^{2^d};\\&C_r^{(d)} = R_{d-2}(C_{r-1}^{(d)}, 0) \text{ for } 1 \le r < 6(d-1).\end{aligned}$$

JH 해시함수

- Compress function F_d
  - 메시지의 상위 절반을 해시 값과 XOR
  - E_d를 통과
  - 메시지의 하위 절반을 해시 값과 XOR



- F_8이 기본적인 JH에서 사용하는 함수

$$A^j = H^{(i-1),j} \oplus M^{(i),j} \quad \text{for } 0 \le j \le 511 ;$$
$$A^j = H^{(i-1),j} \quad \text{for } 512 \le j \le 1023 ;$$
$$B = E_8(A) ;$$
$$H^{(i),j} = B^j \quad \text{for } 0 \le j \le 511 ;$$
$$H^{(i),j} = B^j \oplus M^{(i),j-512} \quad \text{for } 512 \le j \le 1023 ;$$

JH 해시함수

- JH의 성능
- 8-bit 프로세서
  - 1152-byte 사전 연산 → ROM에 탑재
  - SSE2 bit-slice(CORE2): 16.8 cpb
  - 일반 bit-slice: 1344cpb
- Intel Core 2
  - T6600 2.2Ghz
  - 64-bit OS: 16.6 cpb(-O2)
  - 32-bit OS: 23.3 cpb(-O3)

	collision	second-preimage	preimage
JH-224	2^{112}	2^{224}	2^{224}
JH-256	2^{128}	2^{256}	2^{256}
JH-384	2^{192}	2^{384}	2^{384}
JH-512	2^{256}	$2^{512-\log_2 \bar{l}}$	2^{512}

Q & A