MD5 해시 알고리즘

https://youtu.be/fzYgD-ypCOI

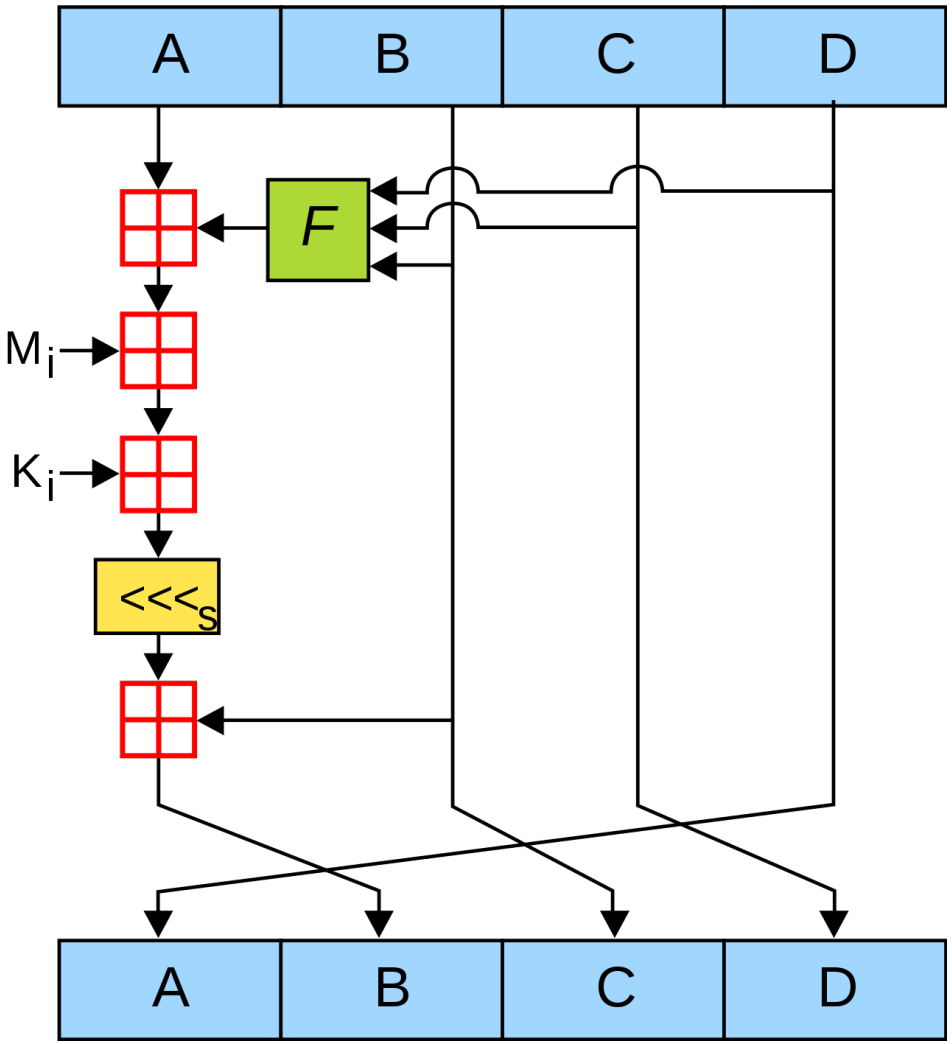
IT 융합공학부 송경주

MD5

- 1991년 MD4를 대체하기 위해 Ronald Rivest 가 제안한 해시 알고리즘
  - MD4 안전성이 위협 받을 가능성이 알려짐
- 128 비트의 해시 값을 출력함
- MD5는 기존 해시함수와 같이 데이터 무결성 확인을 위해 사용되었지만 취약점이 발견됨
  - 1996년 : 설계 상 결함 발견
  - 2004년 : 암호화 결함 발견
  - 2006년 : 노트북 한대로 1분 내에 해시 충돌을 찾을 수 있는 알고리즘이 발표됨
  - 2008년 : MD5 결함으로 SSL 인증서 변조가 가능해짐
- 따라서 현재 사용을 권하지 않지만 아직 사용하는 곳이 존재함

MD5

- 임의의 길이 메시지에 대해 모두 128 비트 고정 길이 해시를 출력함
- 입력된 메시지는 512의 배수로 패딩되어 512비트의 chunk 들로 나뉨 (16개의 32bit 워드)
- A, B, C, D : 32비트의 워드 → 총 128 비트 상태에서 동작
- A, B, C,D 레지스터는 Function, ADD, Shift 를 통해 연산이 진행됨



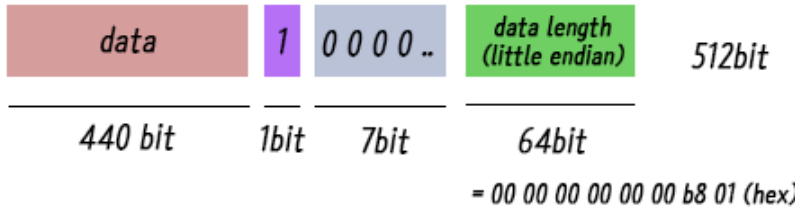
MD5

```
//Pre-processing:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit (bit, not byte) length of unpadded message as 64-bit little-endian integer to message
```

<메시지 패딩>

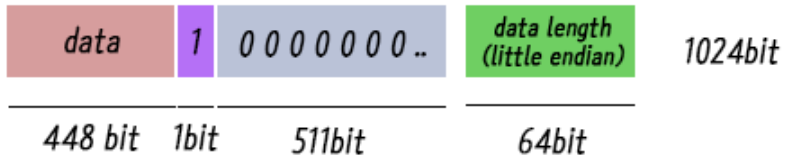
- 메시지를 512의 배수가 되도록 패딩 시킴
- 패딩 된 메시지는 512비트 단위로 나누어 입력됨
- M_i: i 번째 32 비트 블록

ex) input data 440bit

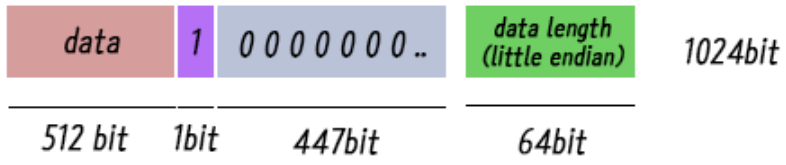


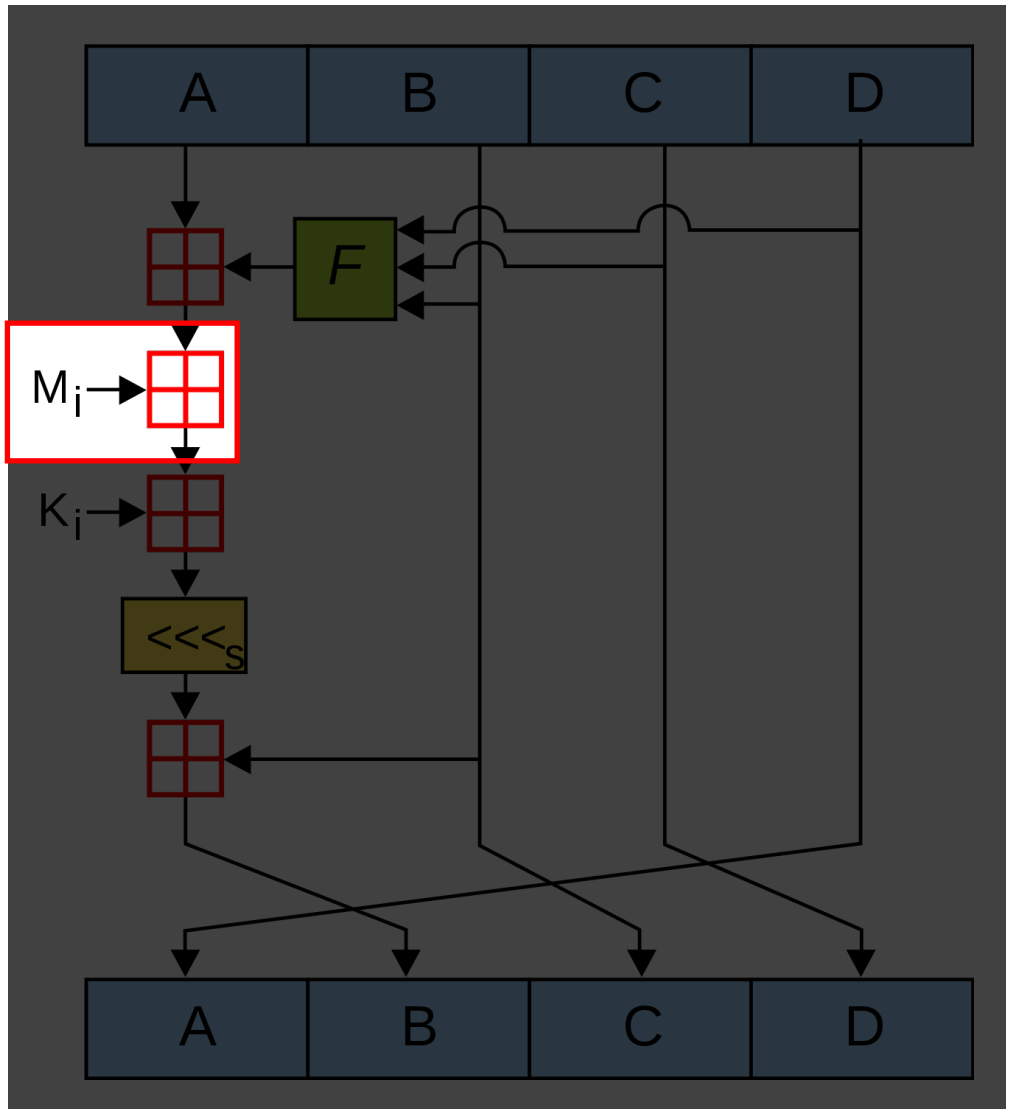
= 00 00 00 00 00 00 b8 01 (hex)

ex2) input data 448 bit



ex3) input data 512 bit



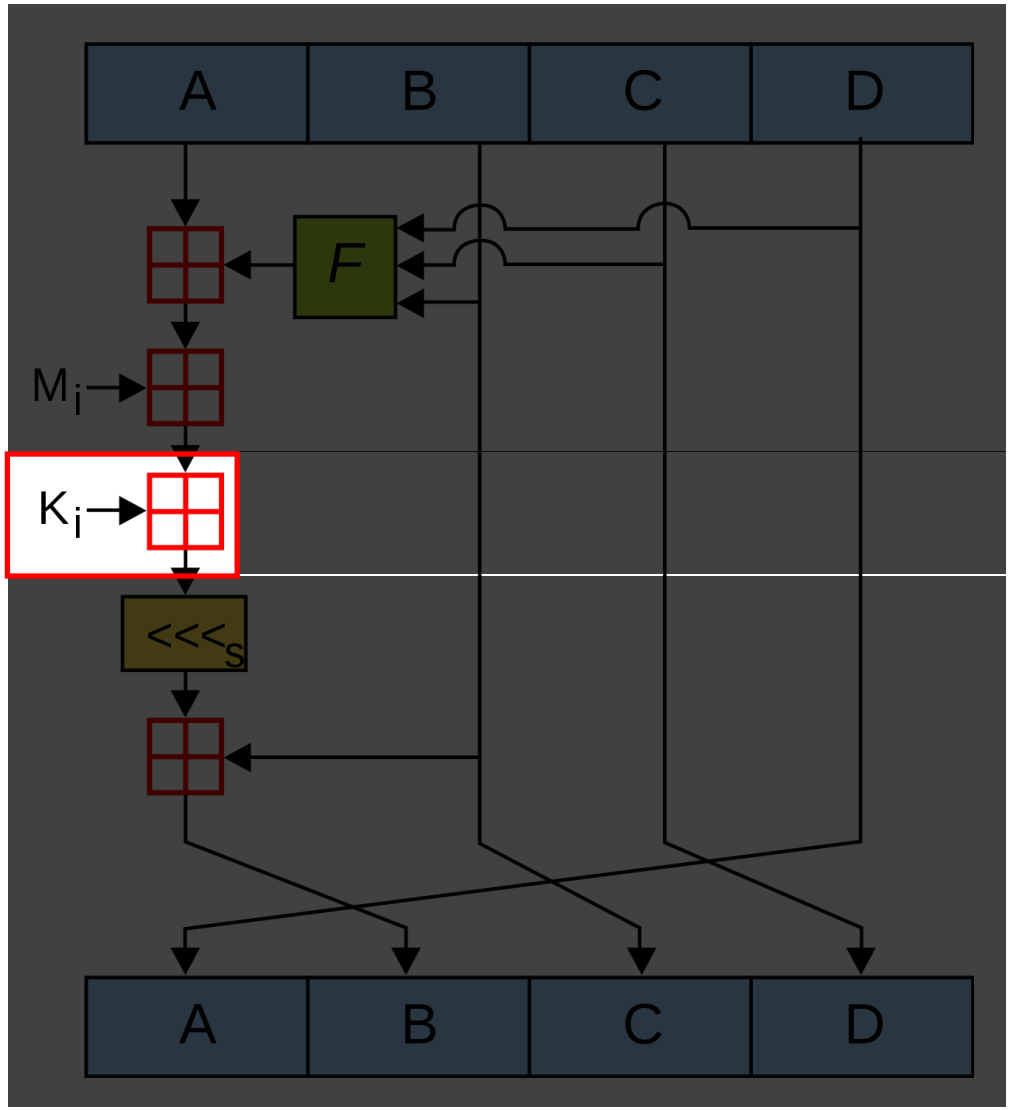


MD5

<초기값 설정>

```
# This list maintains the additive constant to be added in each processing step.
constants = [int(abs(math.sin(i + 1)) * 4294967296) & 0xFFFFFFFF for i in range(64)]
```

```
K = [0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,
     0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,
     0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,
     0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821,
     0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,
     0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,
     0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,
     0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a,
     0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,
     0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,
     0x289b7ec6, 0xea127fa, 0xd4ef3085, 0x04881d05,
     0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,
     0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,
     0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,
     0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,
     0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391 ]
```



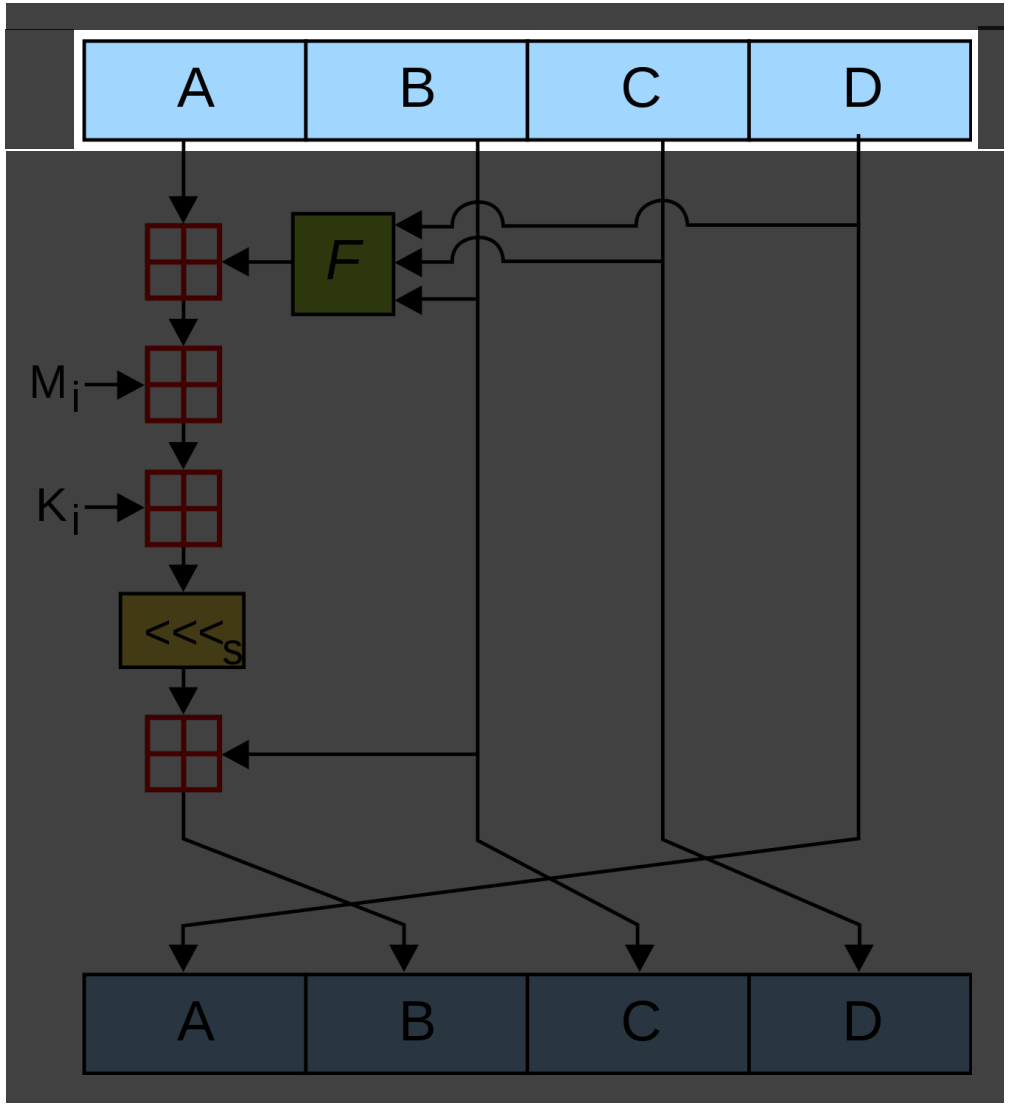
MD5

<초기값 설정>

- A, B, C, D : 각 32비트 워드 → 1개의 State
- 정해진 값으로 초기화 한 후 라운드 함수를 통해 업데이트를 진행함

```
//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476
```

```
//Initialize hash value for this chunk:
var int a := h0
var int b := h1
var int c := h2
var int d := h3
```



MD5

<내부 Function>

- 라운드마다 다른 연산이 진행됨

* $\oplus$: XOR, $\wedge$: AND, $\vee$: OR, $\neg$: NOT

- **0~15 라운드:**

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$

- **16~31 라운드:**

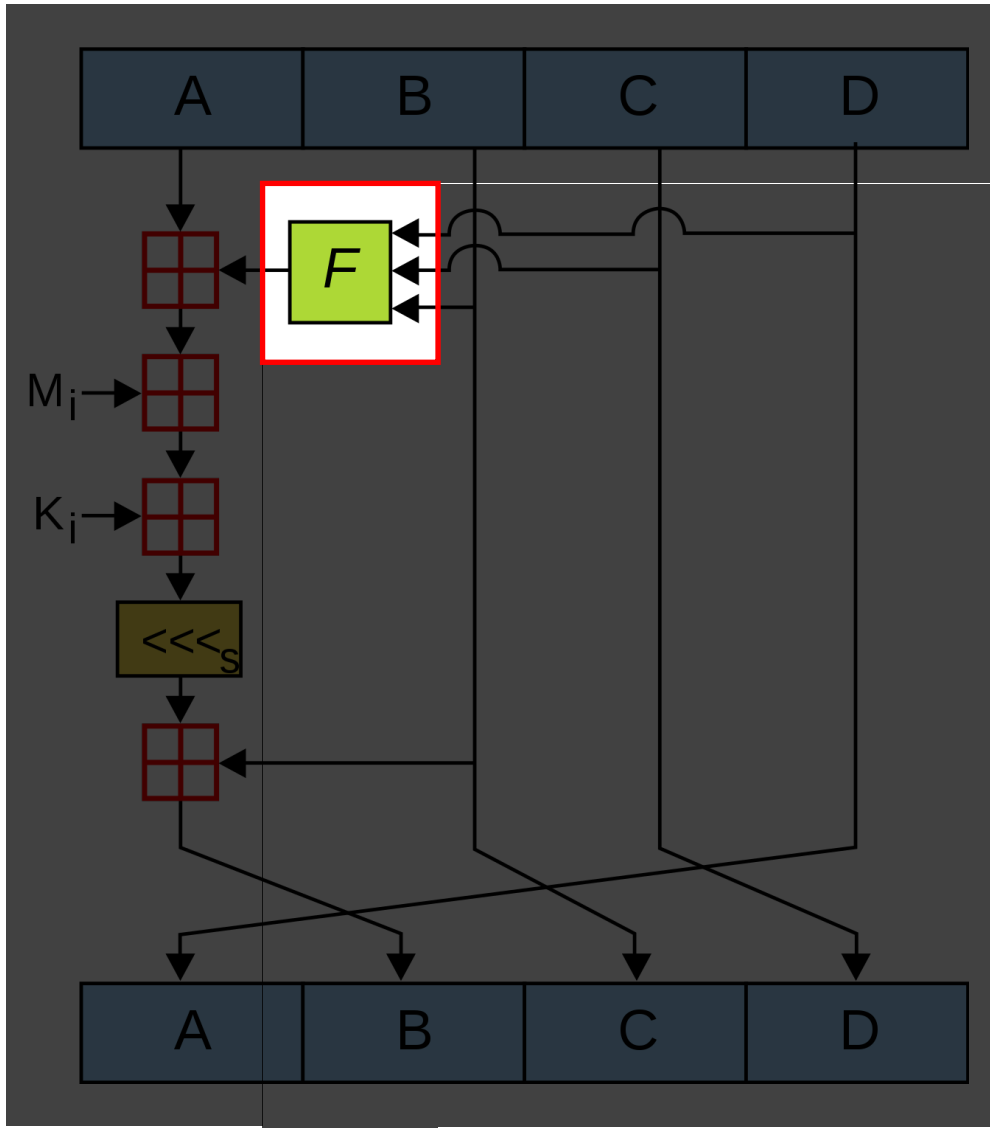
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$

- **32~47 라운드:**

$$H(B,C,D) = B \oplus C \oplus D$$

- **48~63 라운드:**

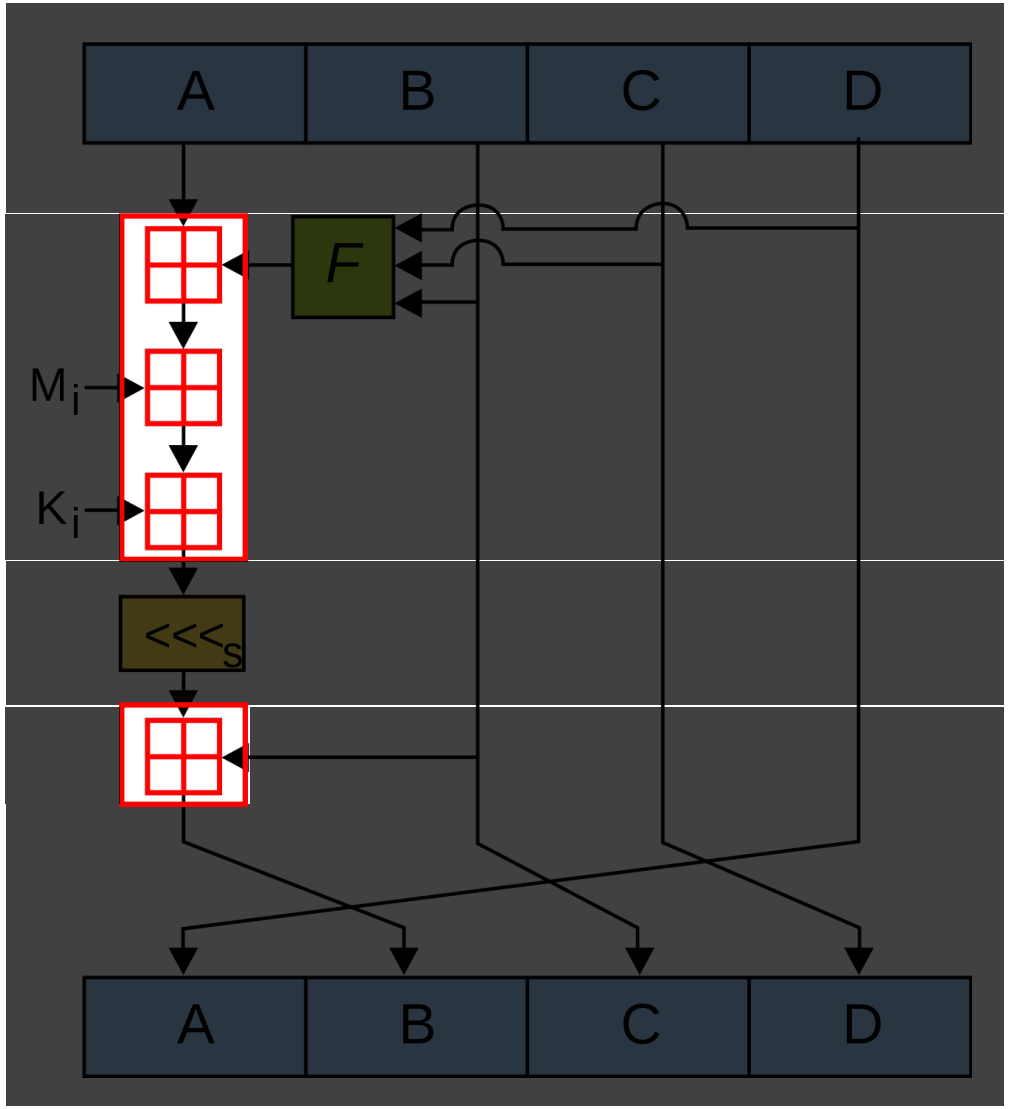
$$I(B,C,D) = C \oplus (B \vee \neg D)$$



MD5

<내부 연산>

- 모듈로 2^{32} 덧셈

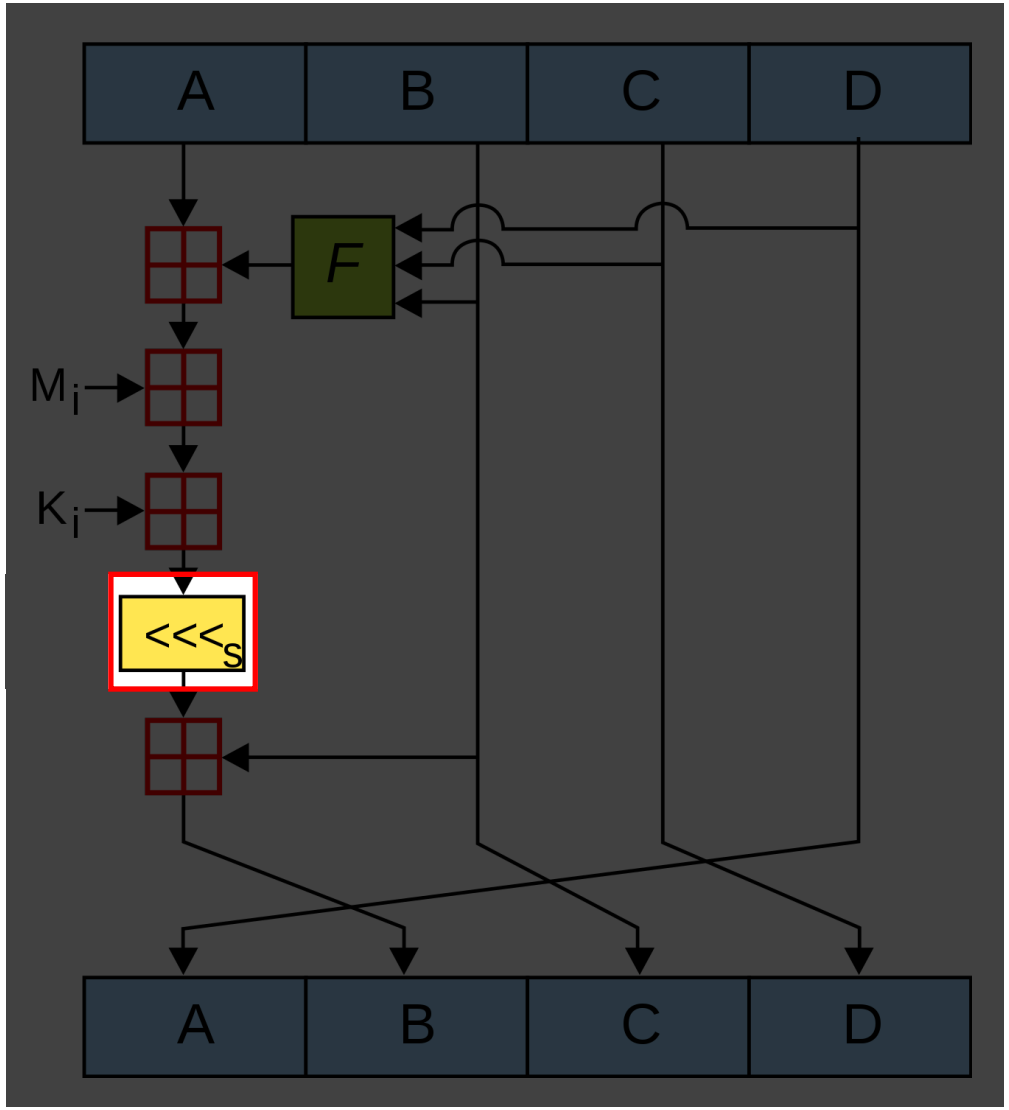


MD5

<내부 연산>

- Left rotation : 미리 주어진 Shift amounts에 따라 왼쪽으로 rotation 진행

```
//r specifies the per-round shift amounts
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}
```



MD5

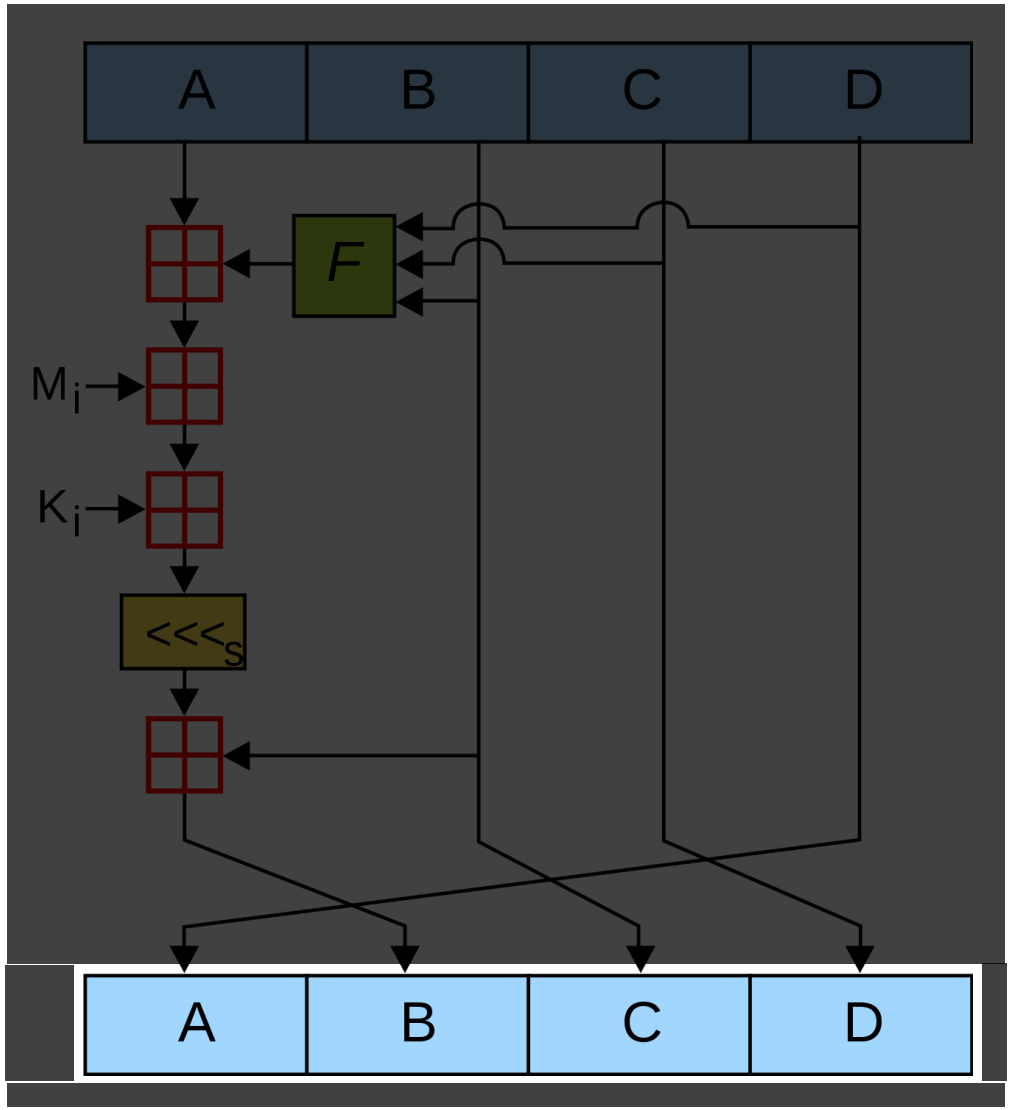
<해시 출력>

- 초기 해시 chunk에 최종 라운드를 마친 A, B, C, D 레지스터 값을 더해 줌

```
//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476
```

```
//Add this chunk's hash to result so far:
h0 := h0 + a
h1 := h1 + b
h2 := h2 + c
h3 := h3 + d
```

- 덧셈을 마친 h0 | h1 | h2 | h3 가 최종 해시 값



Q & A