WebAssembly로 구현한 AES

한성대학교
안규황

HSU
HANSUNG
UNIVERSITY

목 차

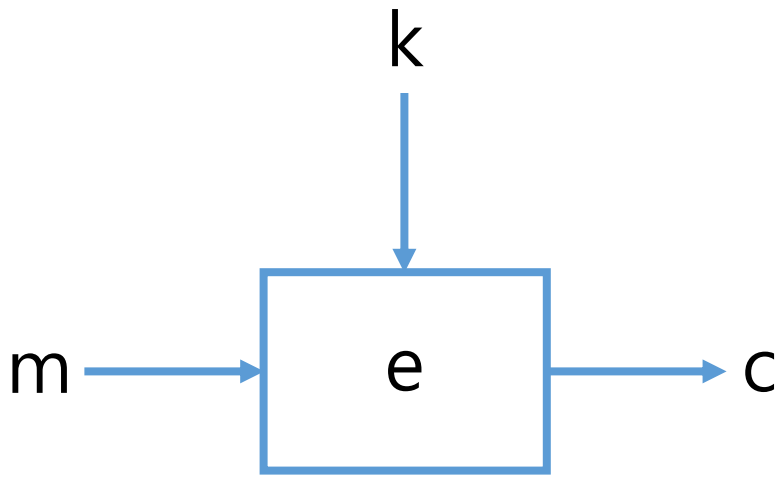
HSU
HANSUNG
UNIVERSITY

AES 소개

HSU HANSUNG UNIVERSITY

블록암호 정의

- 입력 블록 m, 출력 블록 c, 키 k



$$e: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$$
$$(m,k) \rightarrow \mathrm{c}$$

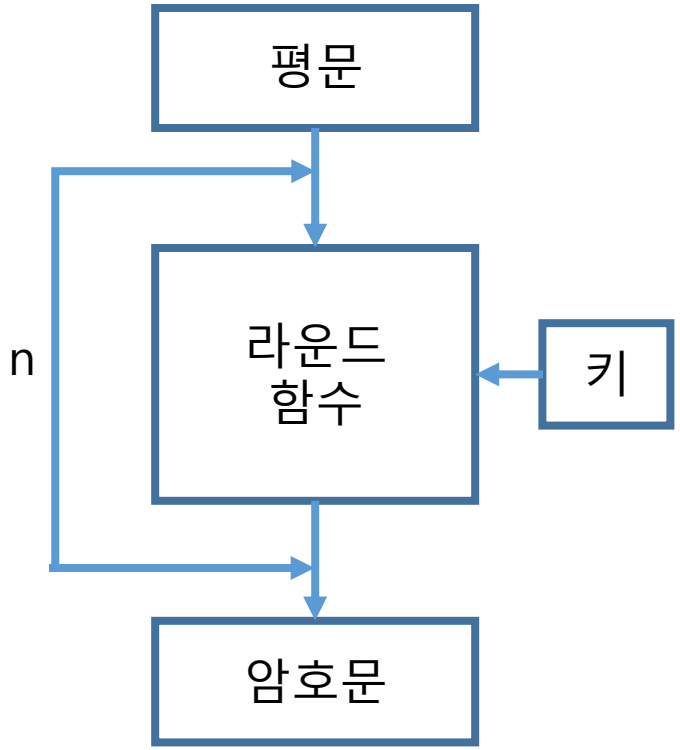
$$d: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$$
$$(c,k) \rightarrow \mathrm{m}$$

- 키 k가 주어질 때, e(*, k) = e_k 는 가역이고 e_k 와 그 역 d_k 의 계산이 쉬워야함
- 주어진 m과 c에 대해 e_k (m) = c를 만족하는 k를 계산하기 어려워야 함

HSU HANSUNG UNIVERSITY

블록암호 설계 원리

- 라운드 함수 반복 구조
  - Confusion과 Diffusion으로 구성 된 라운드 함수를 여러 번 반복하는 구조
- 각 라운드 함수가 독립적인 연산이 되도록 하기 위하여 각 라운드에 서로 다른 키 적용
  - k비트 비밀 키 → 여러 개의 라운드 키 생성을 위한 키 스케줄 함수 사용
- 알고리즘 설계, 구현, 안전성 분석의 용이성
- 잘 설계된 라운드 함수는 반복 될 수록 안전성 증가



HSU
HANSUNG
UNIVERSITY

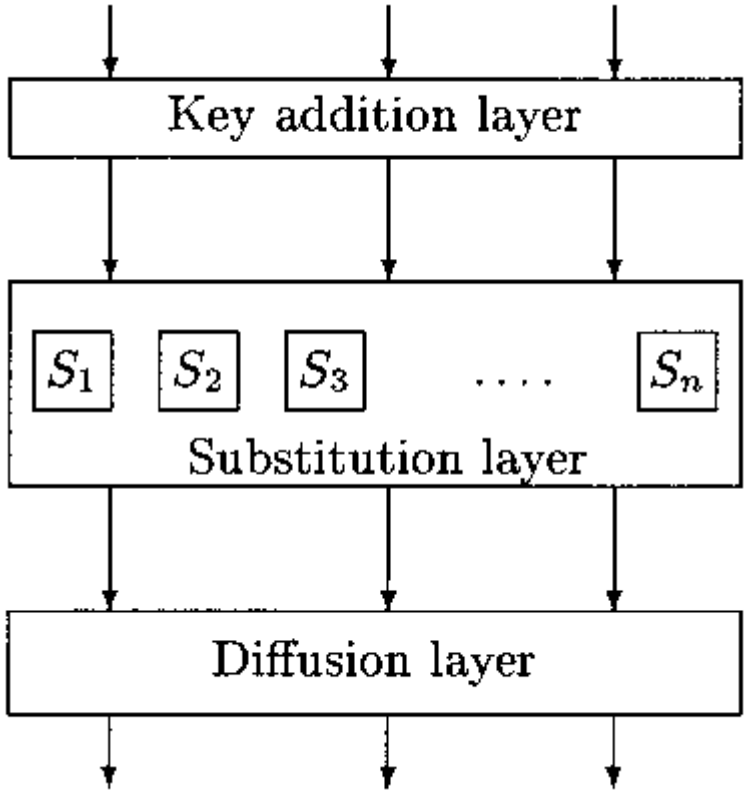
AES

- 미국 NIST 주관 공모사업으로 2001년 개발
  - 벨기에 암호학자 Rijmen, Daemen이 개발
- 규격
  - 구조 : SPN 구조
  - 블록길이 : 128비트
  - 키길이 : 128, 192, 256비트
  - 라운드 수 : 10, 12, 14라운드 (키 길이에 따라)
- 1 라운드 함수 구성
  - SubBytes (8비트 S-box 16개)
  - ShiftRows (Byte-wise rotations)
  - MixColum(4x4 MDS over GF(2^8))
  - AddRoundKey(128-bit key XOR)

HSU HANSUNG UNIVERSITY

블록암호(AES) 라운드 구조 – SPN 구조

- SPN: Substitution-Permutation Network
- 각 라운드 함수가 다음과 같이 구성 됨
  - Key ⊕ layer
  - Substitution layer (주로 S-box)
  - Permutation layer (주로 행렬 곱으로 구성)
- 복호화 과정은 암호화 과정의 역 연산
- AES에서 해당 구조를 사용



HSU HANSUNG UNIVERSITY

S-box

- 암호 알고리즘에 비선형성을 주기 위한 목적으로 사용됨
  - 작은 크기 (≤ 16bit)의 입출력을 가지는 비선형 함수
  - 암호 분석적 특성(선형, 차분 확률, 대수적 차수 등)을 고려하여 적용 필요
  - 경우에 따라 구현 특성도 고려할 필요 있음

- 라운드 키 덧셈 (xor/add)과 결합하여 confusion 효과를 줌

- 주로 8비트 또는 4비트 S-box가 사용됨

S-box

- AES에 사용 된 S-box
  - 8비트 입출력
  - GF(2^8)상의 곱셈에 관한 역함수와 8비트 입출력 affine 함수의 합성
    - $y = A[x^{-1}] + b$

- 암호학적 특성
  - 최대 차분 / 선형 확률 최소화
  - 대수적 특성 우수

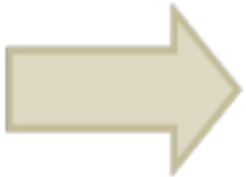
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

AES - State

16 바이트

E	A	S	Y	C	R	Y	P	T	O	G	R	A	P	H	Y
04	00	12	18	02	11	18	0F	13	0E	06	11	00	0F	07	18

(텍스트를 16진수로 표현)

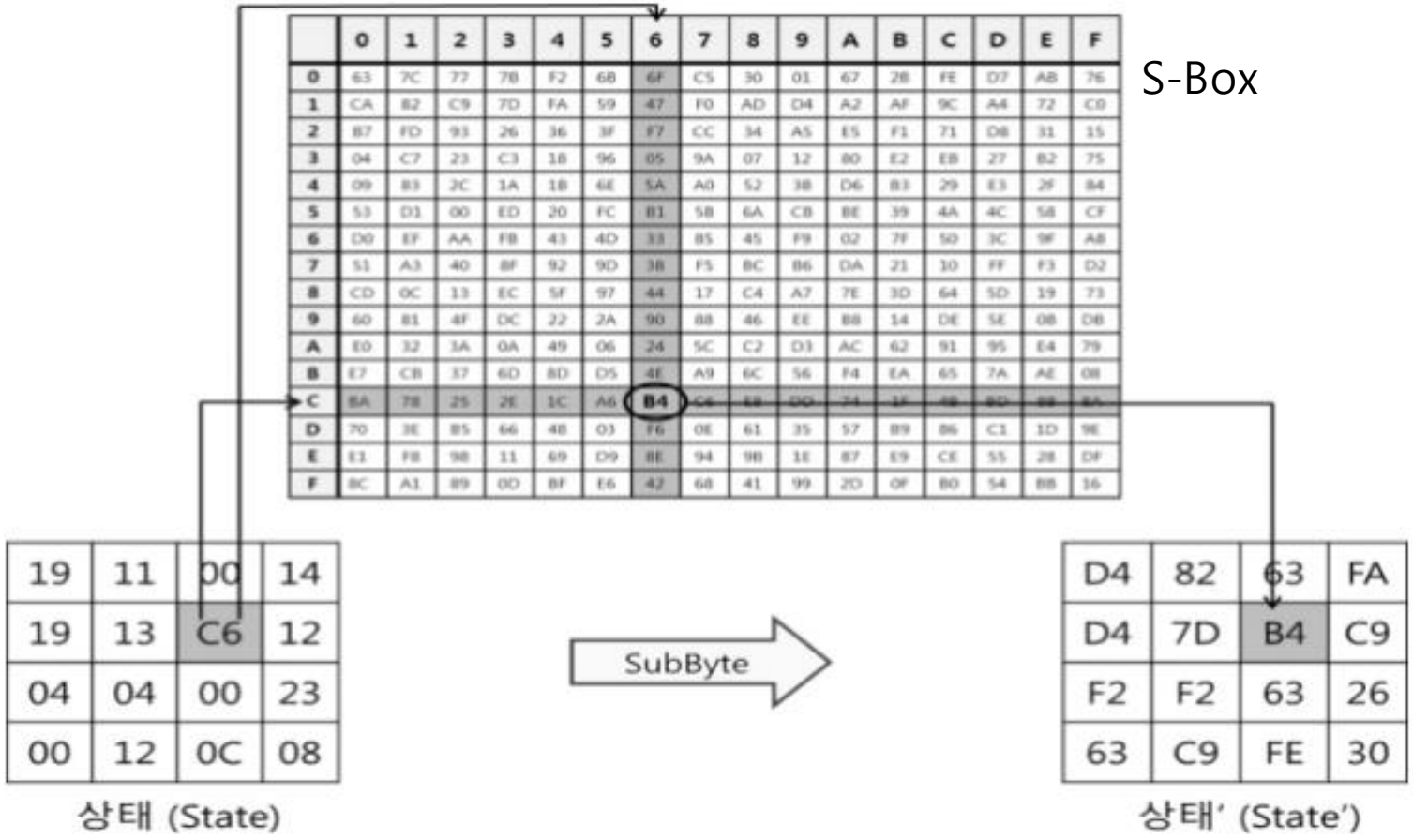


상태(State) 4×4

04	02	13	00
00	11	0E	0F
12	18	06	07
18	0F	11	18

HSU HANSUNG UNIVERSITY

AES - SubBytes



HSU HANSUNG UNIVERSITY

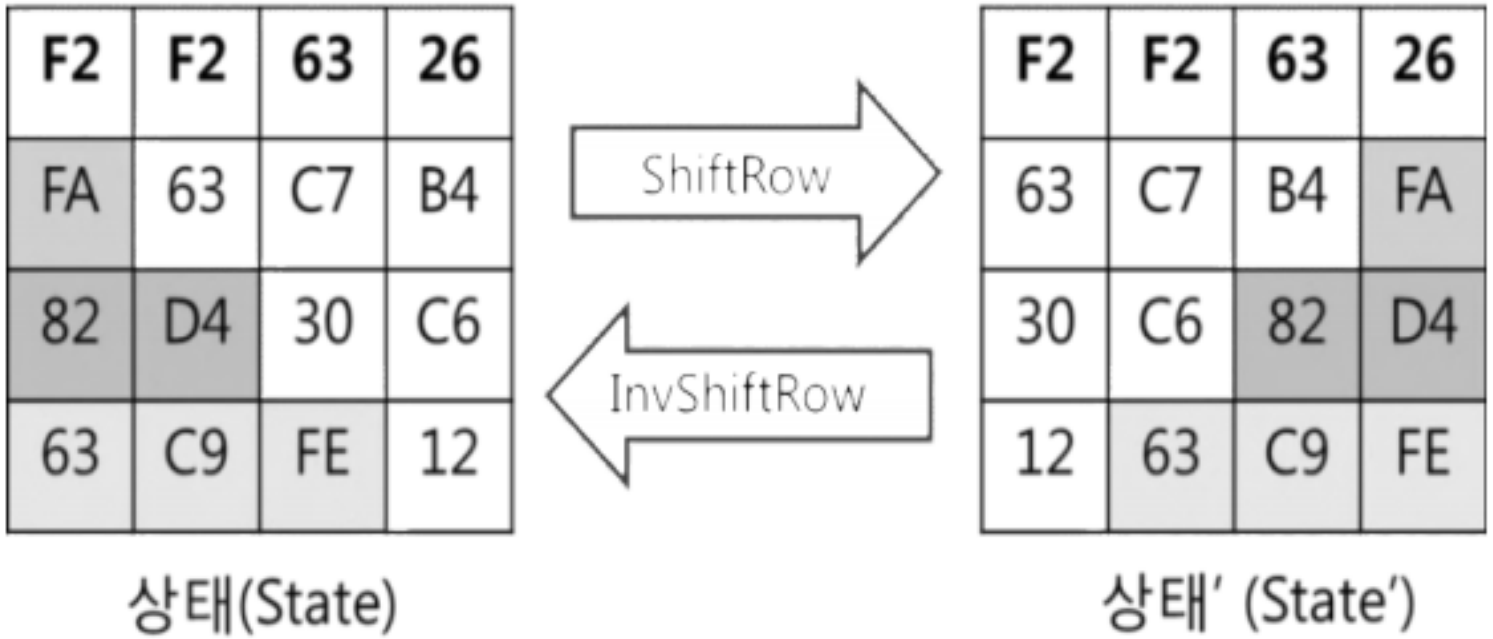
AES – SubBytes_Inverse_S-box

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 14. Inverse S-box: substitution values for the byte xy (in hexadecimal format).

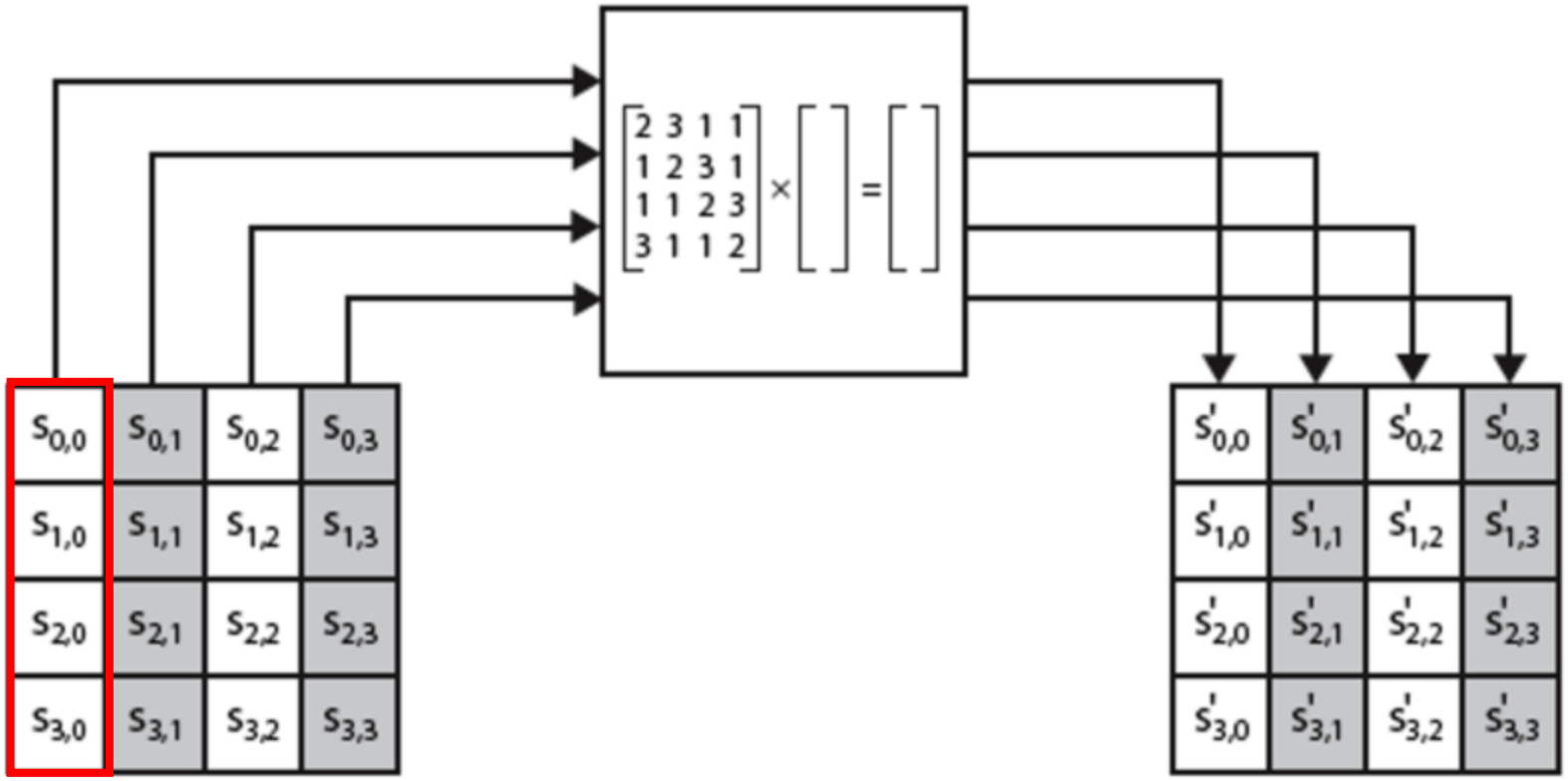
HSU HANSUNG UNIVERSITY

AES – ShiftRows & Inverse_ShiftRows



HSU HANSUNG UNIVERSITY

AES - MixColumns



HSU HANSUNG UNIVERSITY

AES – MixColumns_example

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus \{46\} \oplus \{A6\} = \{47\}$$

$$\{87\} \oplus (\{02\} \bullet \{6E\}) \oplus (\{03\} \bullet \{46\}) \oplus \{A6\} = \{37\}$$

$$\{87\} \oplus \{6E\} \oplus (\{02\} \bullet \{46\}) \oplus (\{03\} \bullet \{A6\}) = \{94\}$$

$$(\{03\} \bullet \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \bullet \{A6\}) = \{ED\}$$

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix} = \begin{bmatrix} \quad \\ \quad \\ \quad \\ \quad \end{bmatrix}$$

HSU HANSUNG UNIVERSITY

AES - MixColumns

$$\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

$$s'_{0,0} = (02 \cdot s_{0,0}) \oplus (03 \cdot s_{1,0}) \oplus (01 \cdot s_{2,0}) \oplus (03 \cdot s_{3,0})$$
$$s'_{1,0} = (01 \cdot s_{0,0}) \oplus (02 \cdot s_{1,0}) \oplus (03 \cdot s_{2,0}) \oplus (01 \cdot s_{3,0})$$
$$\vdots$$
$$s'_{3,3} = (03 \cdot s_{0,3}) \oplus (01 \cdot s_{1,3}) \oplus (01 \cdot s_{2,3}) \oplus (02 \cdot s_{3,3})$$

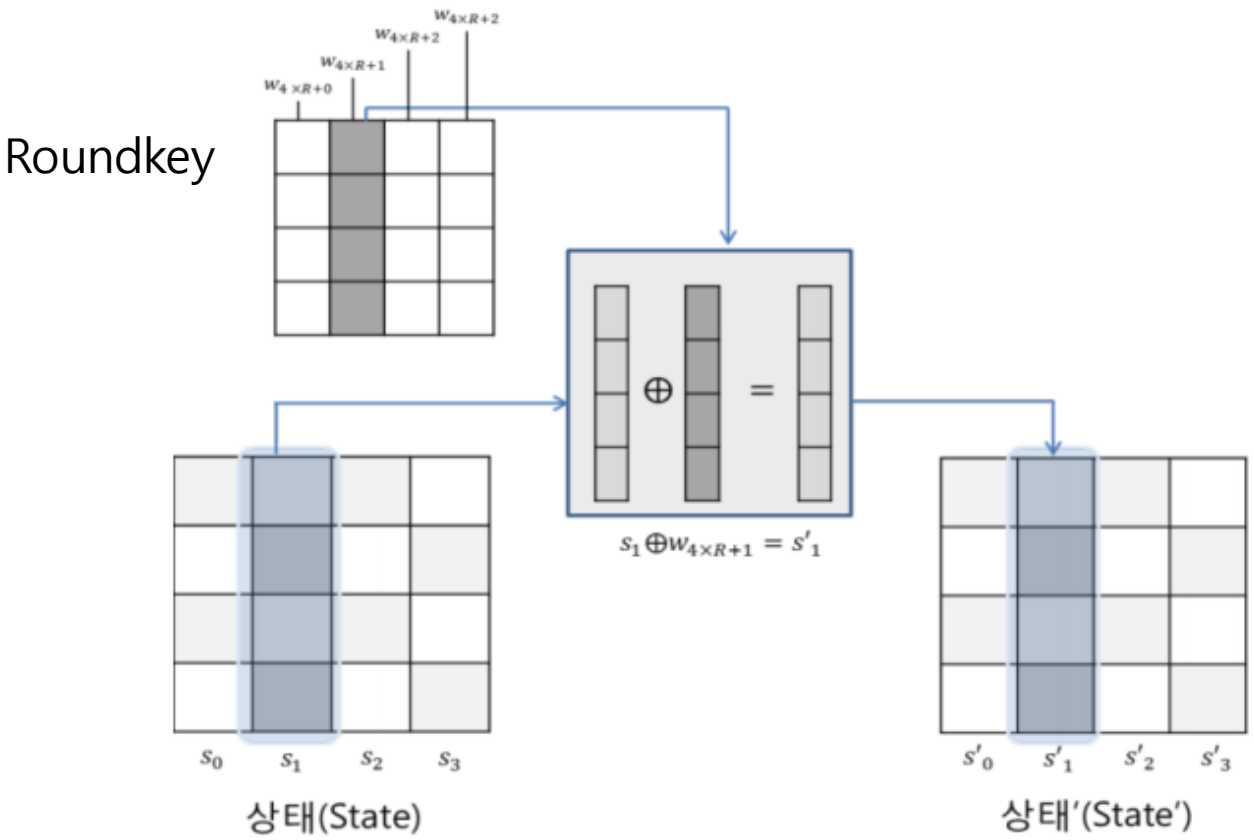
HSU HANSUNG UNIVERSITY

AES – Inverse_MixColumns

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}^{-1} = \begin{bmatrix} 0\mathrm{E} & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

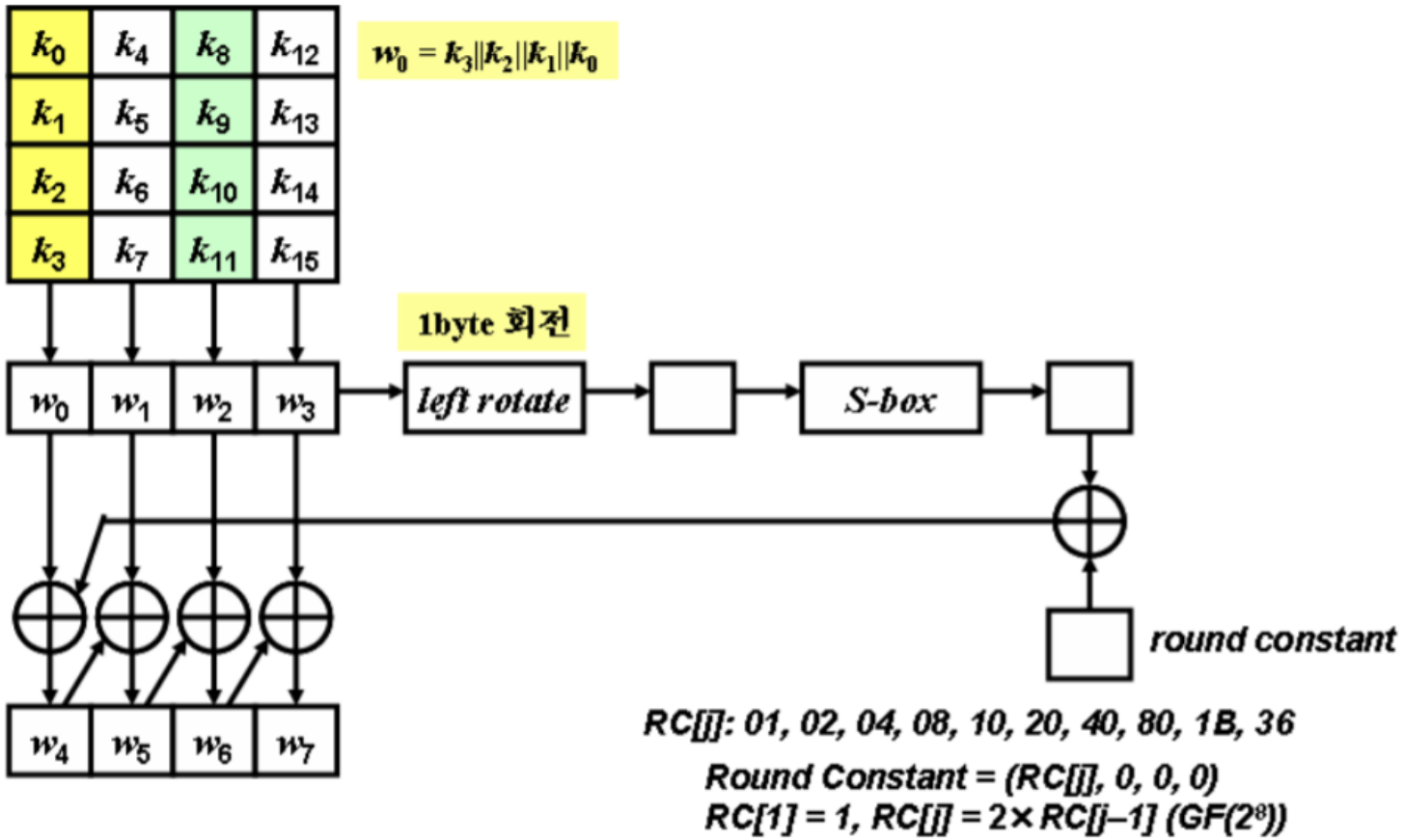
HSU HANSUNG UNIVERSITY

AES - AddRoundKey



HSU HANSUNG UNIVERSITY

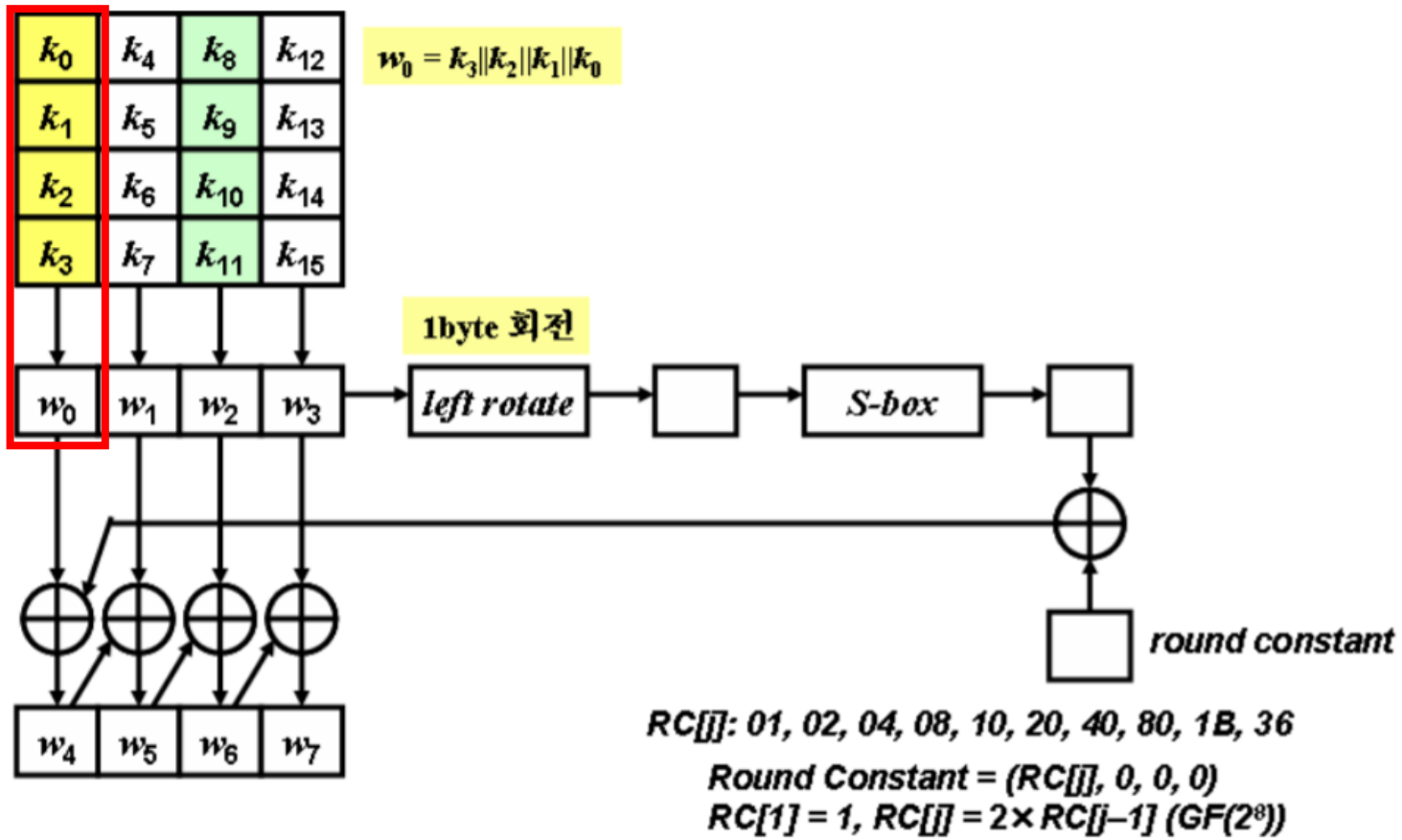
RoundKey - Scheduling



HSU HANSUNG UNIVERSITY

RoundKey - Scheduling

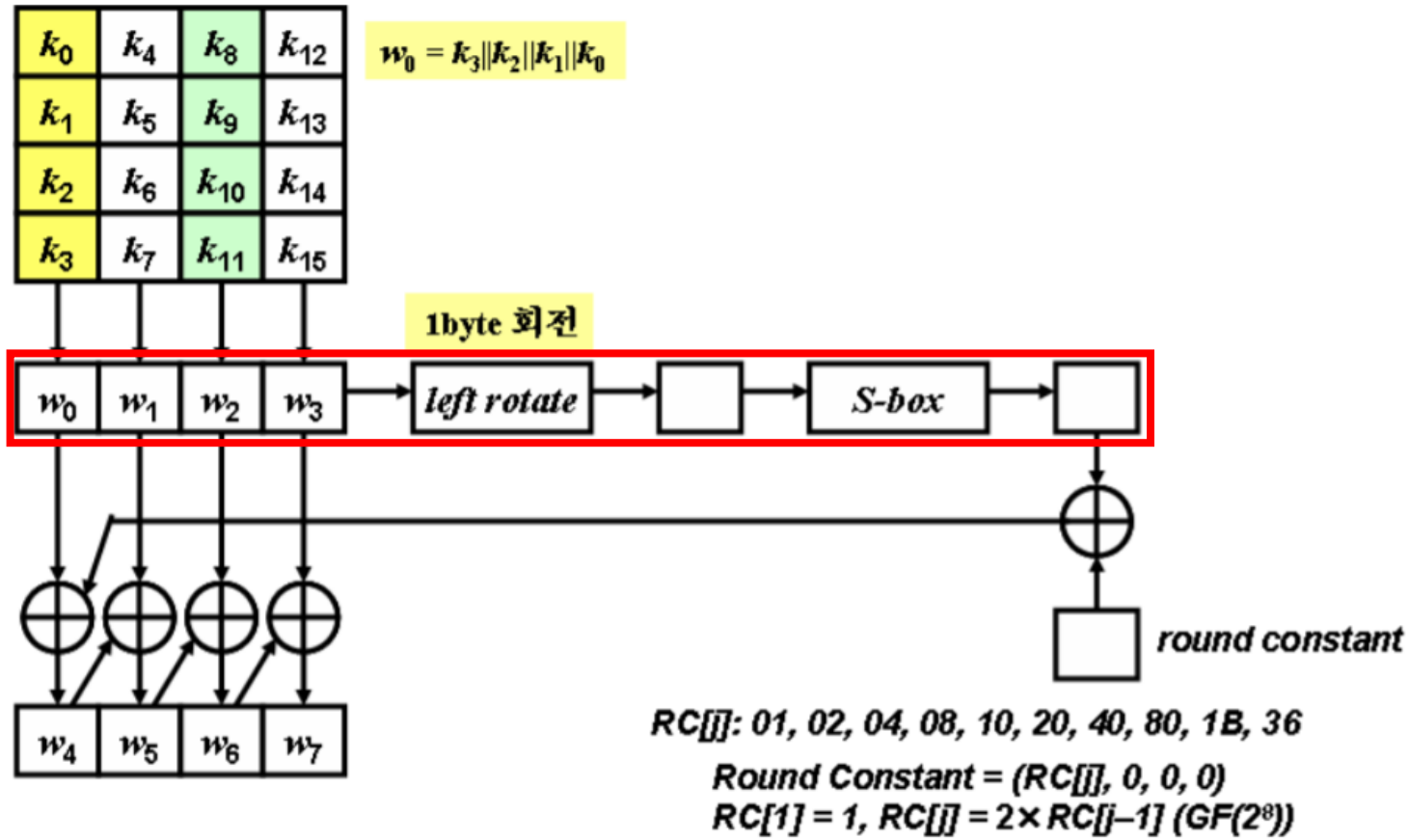
1st step



HSU HANSUNG UNIVERSITY

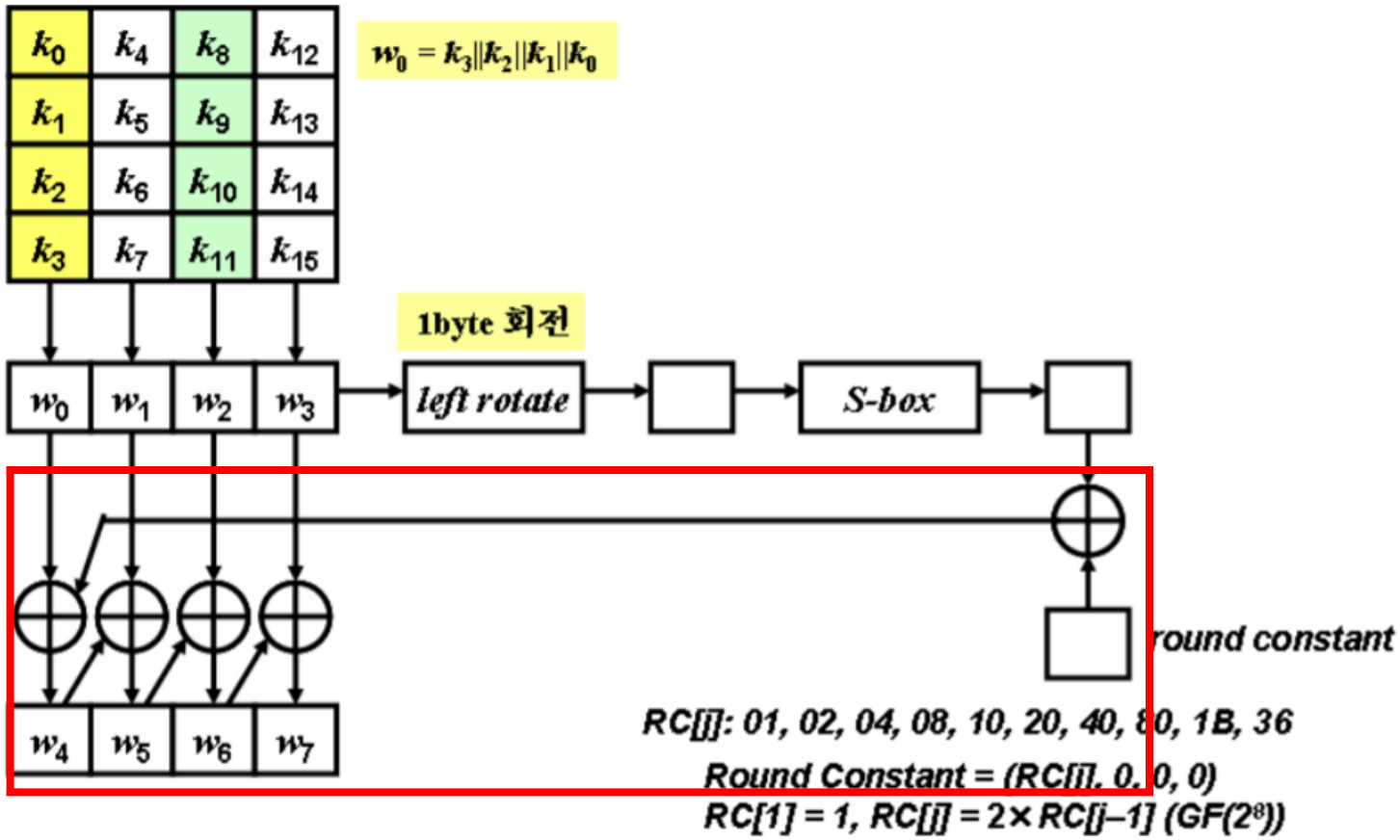
RoundKey - Scheduling

2nd step



HSU HANSUNG UNIVERSITY

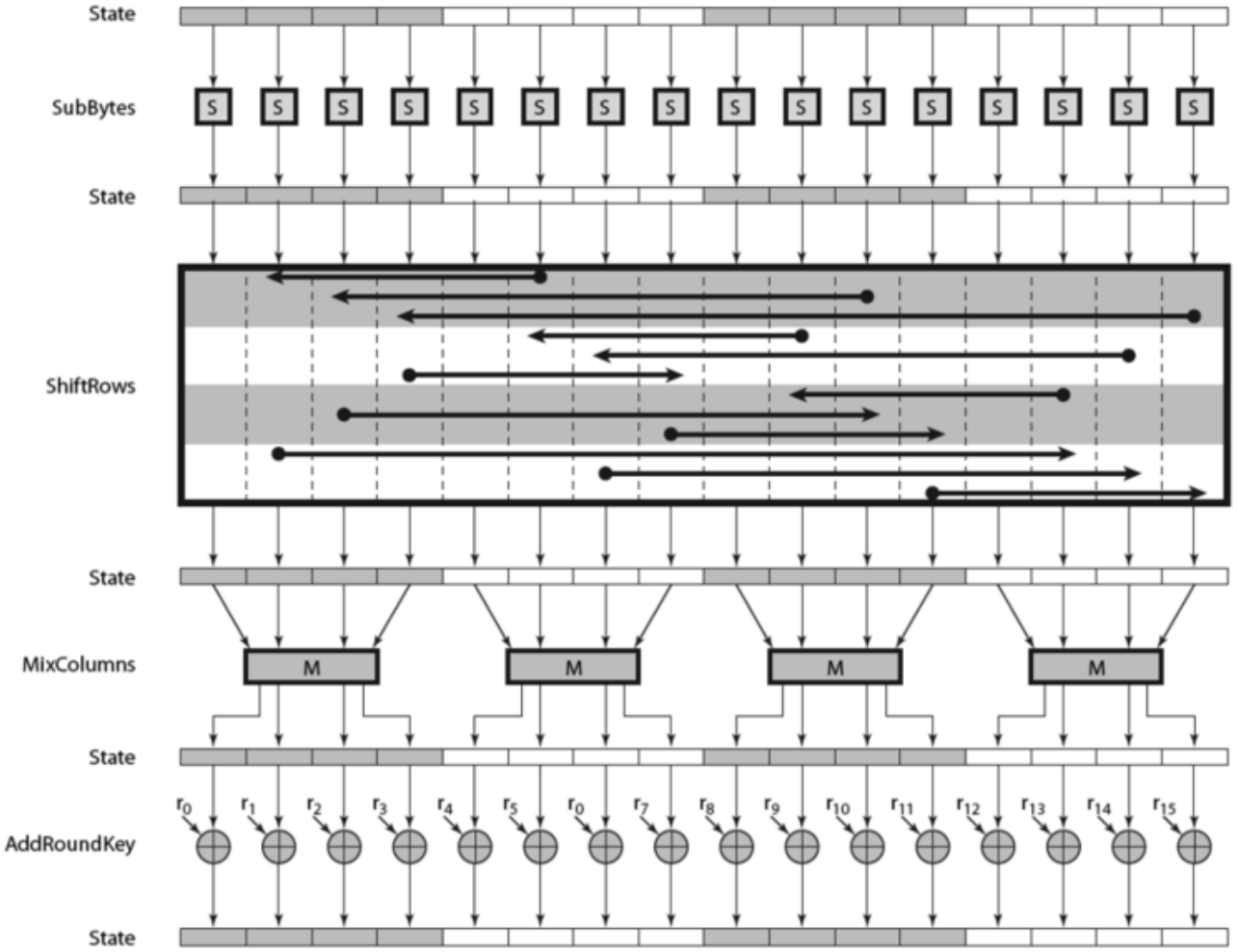
RoundKey - Scheduling



3rd step

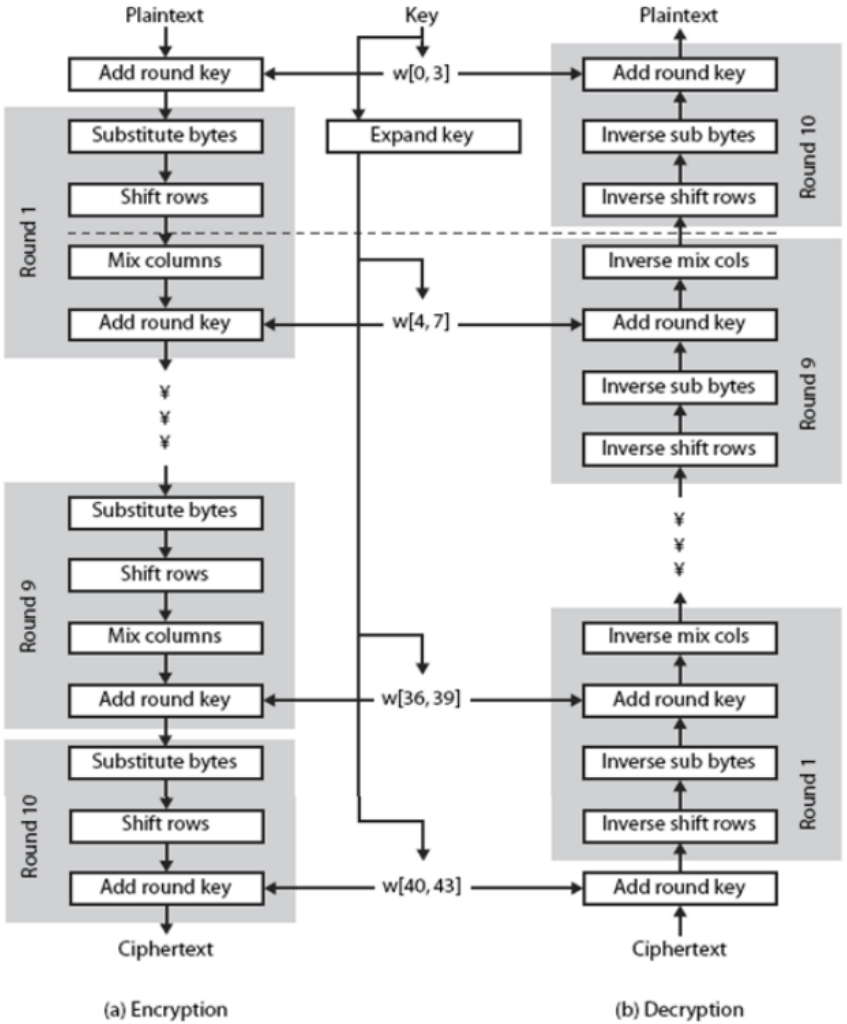
HSU HANSUNG UNIVERSITY

AES – 1 Round



HSU HANSUNG UNIVERSITY

AES 전체 구조



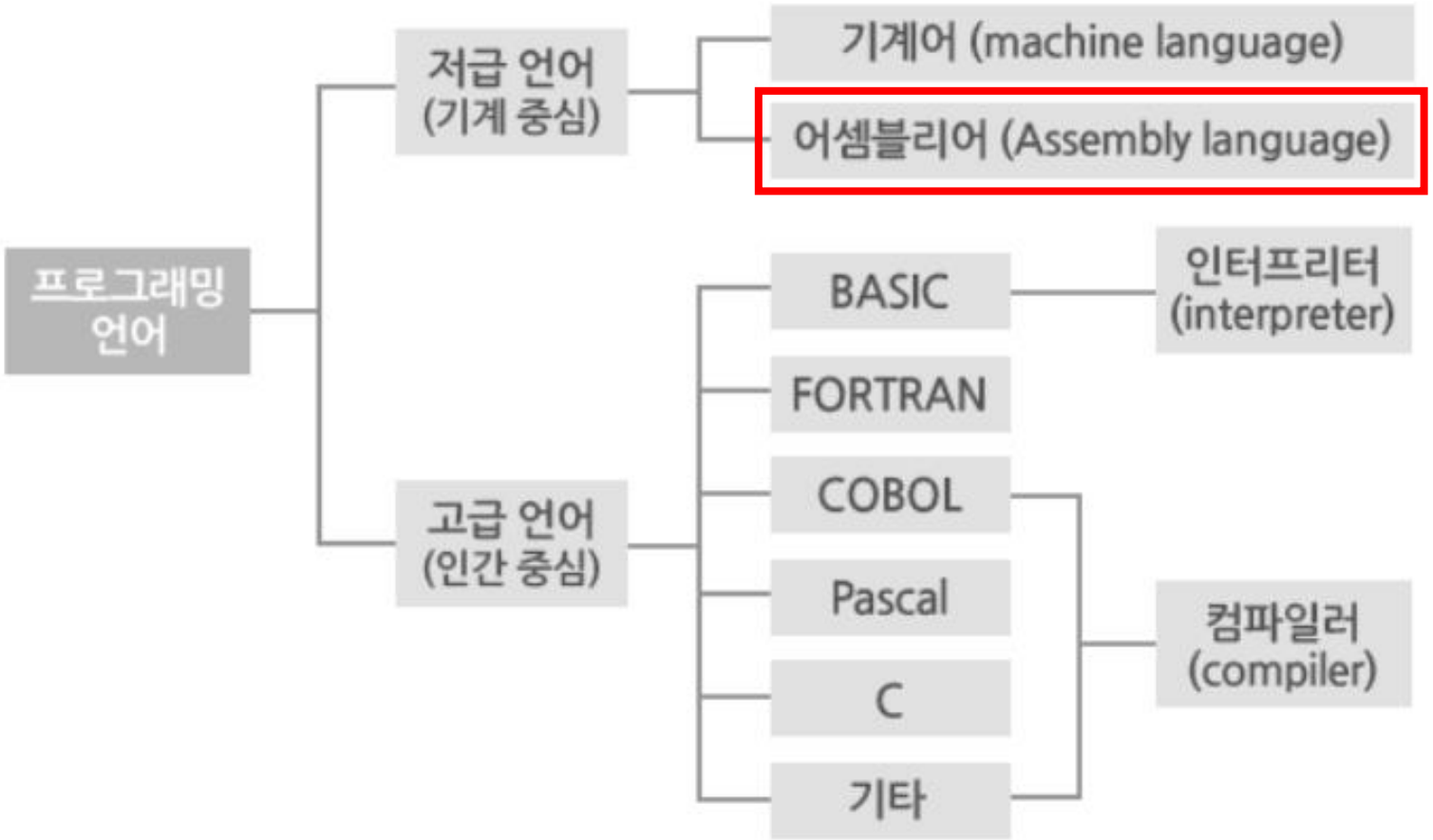
HSU HANSUNG UNIVERSITY

Assembly

HSU
HANSUNG
UNIVERSITY

프로그래밍 언어

프로그래밍 언어는 크게 기계 중심 혹은 인간 중심으로 나누어짐



HSU HANSUNG UNIVERSITY

어셈블리어란?

- 0과 1의 이진수로 프로그래밍을 하는 기계어는 컴퓨터가 바로 읽을 수 있다는 점 빼면 사용하기 아주 불편
- 이를 기계어보단 느리지만 인간이 그나마 이해할 수 있는 언어로 표현한 것을 어셈블리라 칭함
- 하드웨어와 스프트웨어의 가장 밑바닥에 있는 저급 언어

어셈블리어 특징

- 명령 실행 속도가 아주 빠름
- 매우 세밀한 프로그래밍 스킬이 필요
- 어셈블리어는 하드웨어 특성을 탄다
  → 하드웨어 종류에 따라 사용하는 어셈블리어도 달라짐

어셈블리어 장·단점

장점

- 프로그램의 실행 속도가 아주 빠름
- 프로그램의 크기가 매우 작음
- 어떤 기계에서도 사용 가능

단점

- 배우기 어려움
- 큰 프로그램을 구성하기 어려움
- 하드웨어별로 사용하는 어셈블리가 다름
  → 새로 배워야 함

HSU HANSUNG UNIVERSITY

Web Assembly

HANSUNG UNIVERSITY

웹어셈블리(WebAssembly, WASM)란?

- 구글, 마이크로소프트, 애플, 모질라가 소속된 웹 어셈블리 커뮤니티 그룹이 웹 성능을 향상하기 위해 2015년부터 개발한 웹브라우저 표준 포맷

- 웹 브라우저를 돌릴 수 있는 새로운 형식의 코드로 2017년 3월부터 모든 웹 브라우저에 도입

- 기존에는 javascript를 이용하여 웹을 구성했다면,
  웹어셈블리는 C/C++/Rust 등을 이용해 웹 페이지 구현 가능

웹 어셈블리 vs 자바스크립트

- 웹 어셈블리가 나오기전까지 웹 페이지를 표현하는데 사용할 수 있는 언어는 자바스크립트 밖에 존재하지 않았음

- 개발 언어에는 Low-Level 언어와 High-Level 언어가 존재함

  → Low-Level 언어는 컴퓨터가 그나마 이해하기 쉬운 언어 ex) C 언어
  High-Level 언어는 인간이 그나마 이해하기 쉬운 언어 ex) 자바스크립트

- 따라서, High-Level 언어일 수록 컴퓨터가 이해하기 힘들기 때문에 컴파일 하는데 오랜 시간이 걸림

HSU HANSUNG UNIVERSITY

웹 어셈블리 vs 자바스크립트

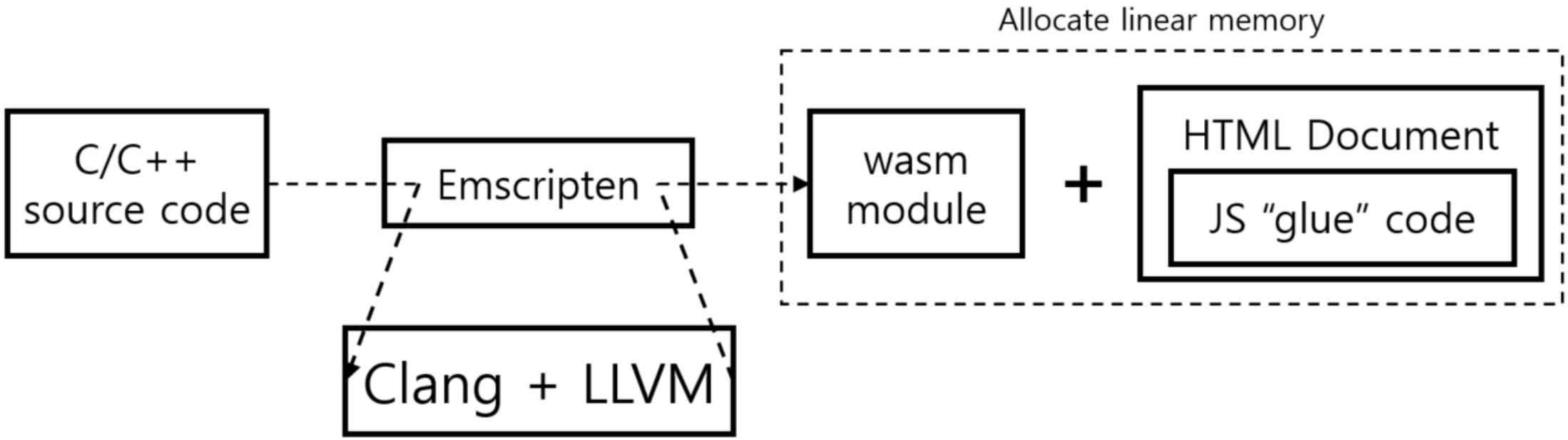
- 웹 어셈블리는 C/C++로 구현한 것을 선형메모리 상에서 '.wasm' 이라는 바이너리 파일로 컴파일한 후 웹 페이지 내 'glue' 라는 JS 영역에 이식하여 웹 페이지를 구성

- 웹 어셈블리가 개발됨에 따라 기존에 자바스크립트로 구현한 것 보다 상당한 속도로 웹 페이지를 구현할 수 있게 됨

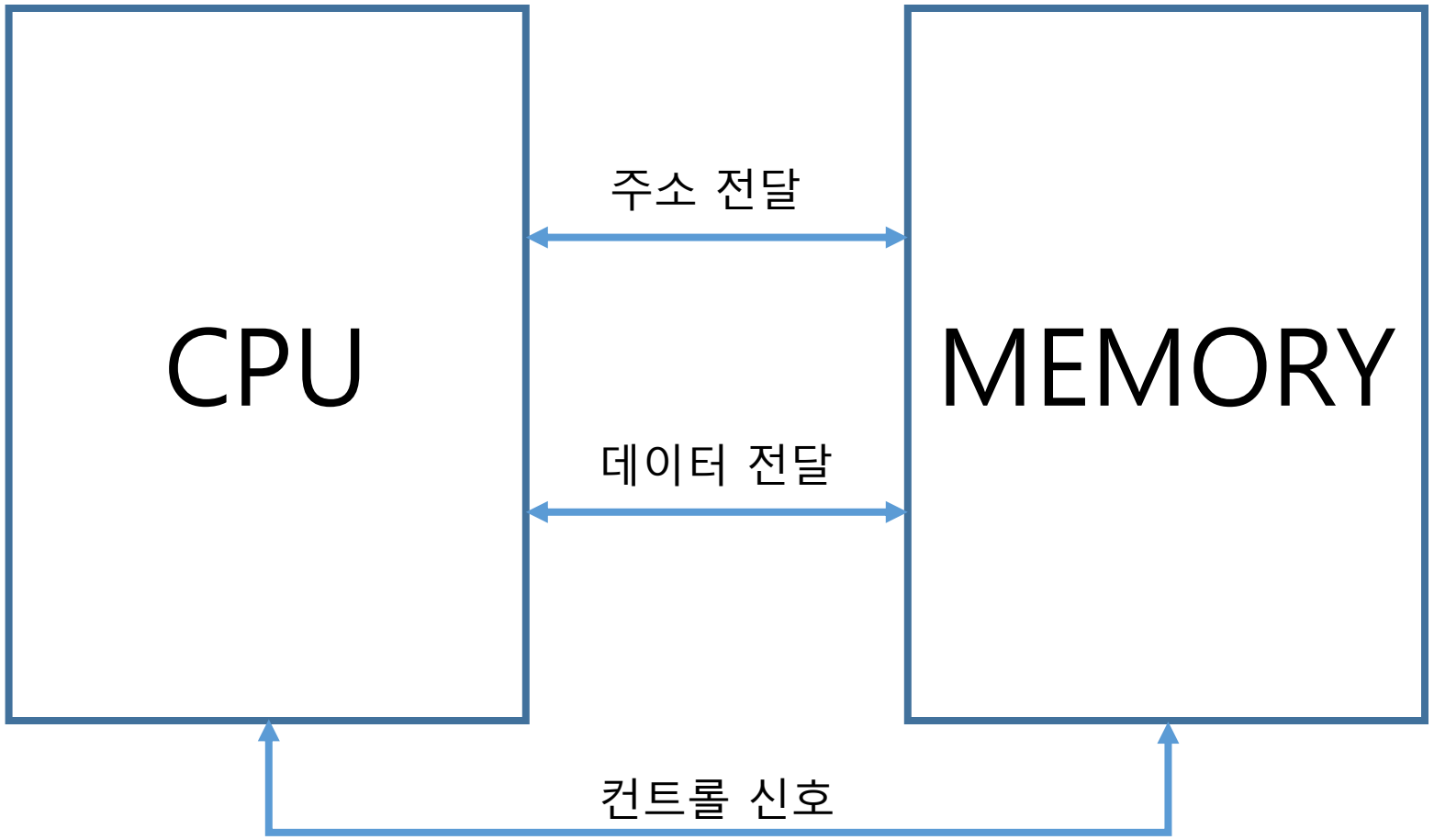
- 웹 어셈블리를 이용하여 암호를 구현하여 이를 검증하기 하고자 함

HSU HANSUNG UNIVERSITY

웹 어셈블리 구조

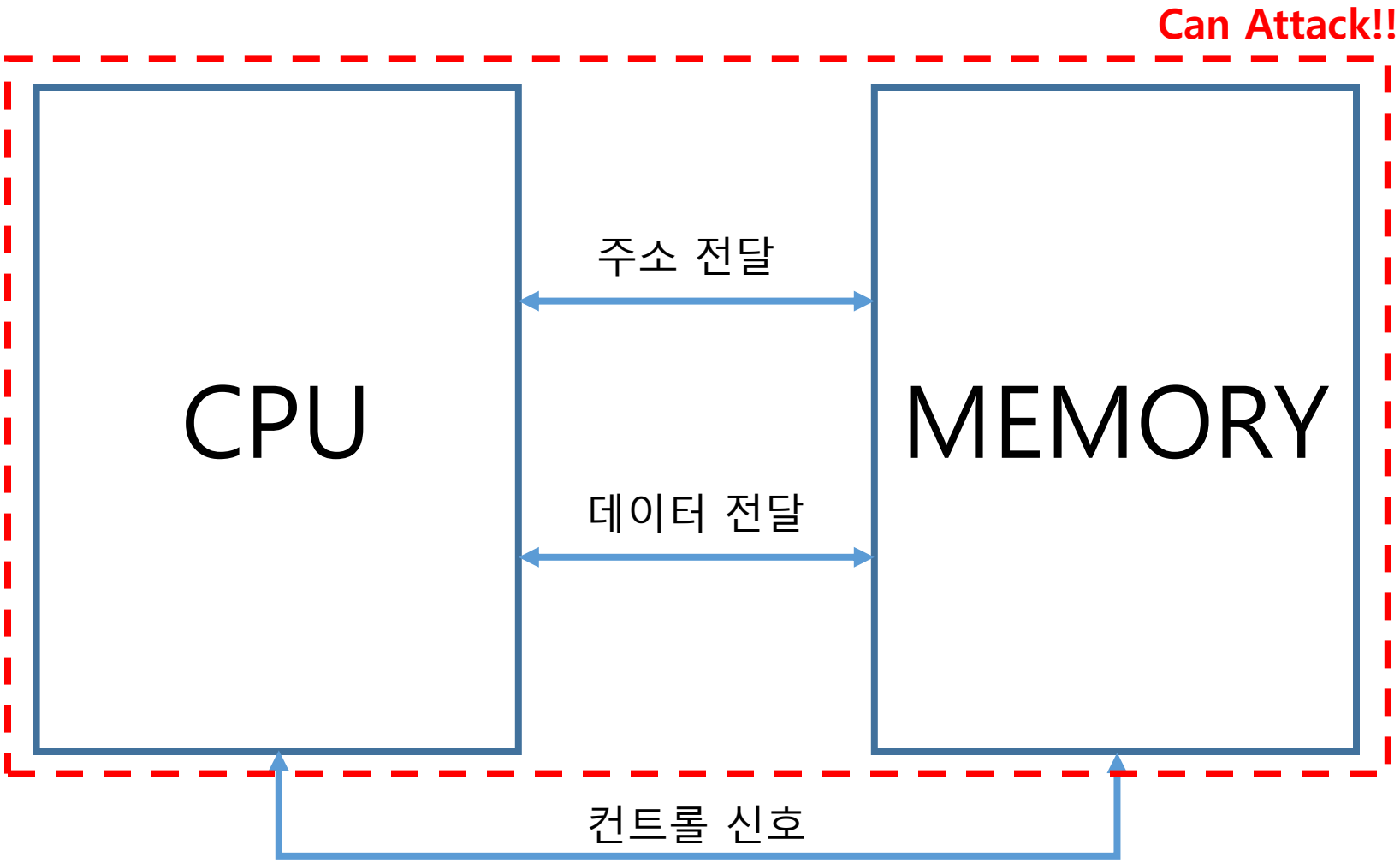

Allocate linear memory
C/C++
source code
Emscripten
Clang + LLVM
wasm
module
+
HTML Document
JS "glue" code

기존 메모리 구조



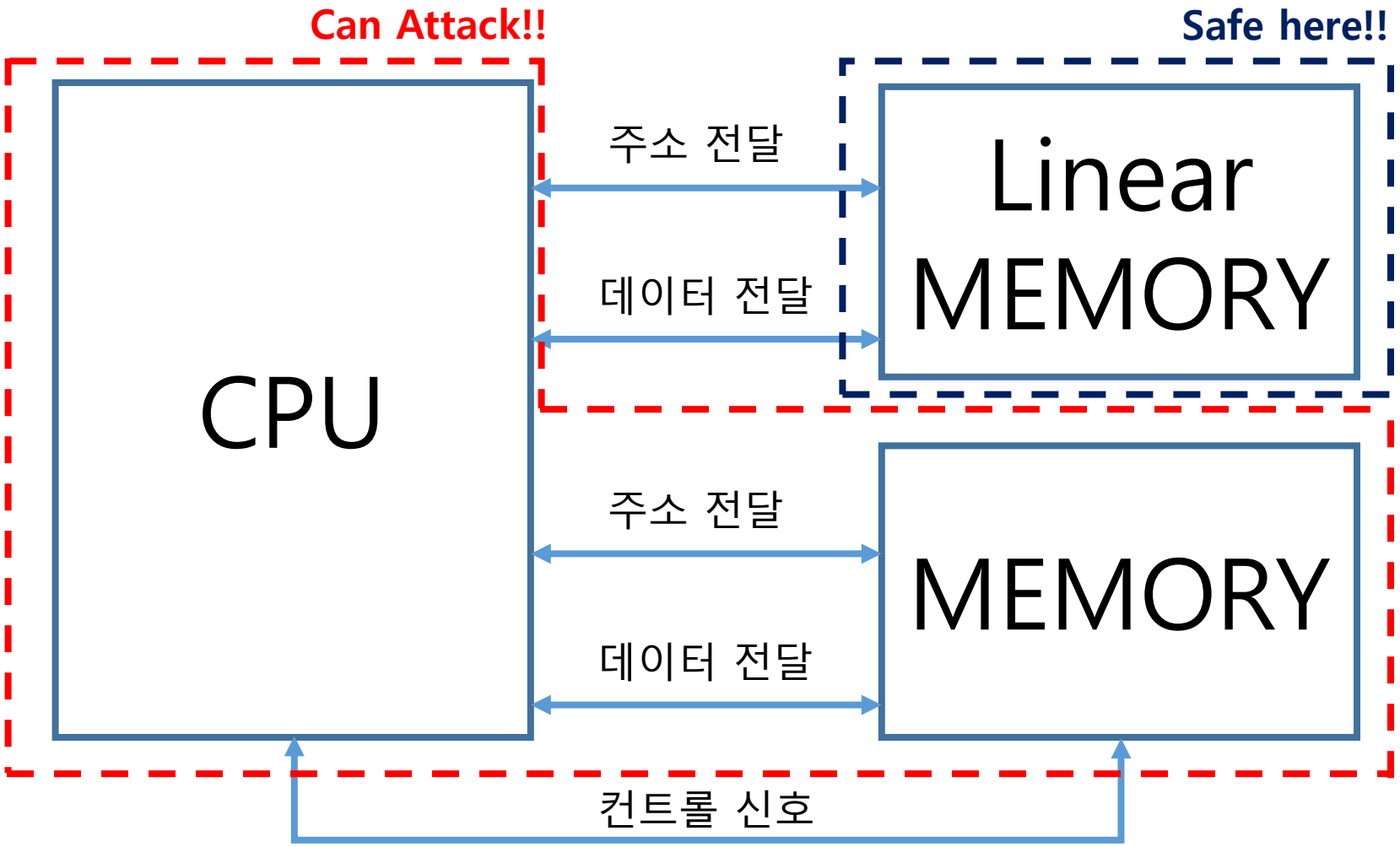
HSU
HANSUNG
UNIVERSITY

기존 메모리 접근 구조



HSU HANSUNG UNIVERSITY

웹어셈블리 메모리 접근 구조



HSU HANSUNG UNIVERSITY

성능 비교

HANSUNG UNIVERSITY

웹 어셈블리 & 자바스크립트를 이용한 AES 구현

구현 환경

OS	**MacOS 10.12.6**
Processor	Intel Core i5 2.7 GHz
IDE	Brackets 1.10
Web Browser	Chrome 69.0.3497.100

HSU HANSUNG UNIVERSITY

https://github.com/kyu-h/WebAssembly_AES_ECB

웹 어셈블리 & 자바스크립트를 이용한 AES 구현

최대 약 3.5배 성능 향상 그러나 최초로 부여된 선형 메모리 영역을 초과할 경우 오버헤드 발생

Language	Time(s)	Cpb
1회 암호화 하는데 수행된 퍼포먼스		
Web Assembly	0.0031	116,250
Javascript	0.011	412,500
100회 암호화 하는데 수행된 퍼포먼스		
Web Assembly	0.0218	817,500
Javascript	0.068	2,550,000
10,000회 암호화 하는데 수행된 퍼포먼스		
Web Assembly	2.0848	78,180,000
Javascript	0.619	23,212,500

HSU HANSUNG UNIVERSITY

결 론

HSU HANSUNG UNIVERSITY

결 론

- 웹 어셈블리로 구현 한 AES가 자바스크립트로 구현한 AES보다 약 3.5배 빠름
- 웹 어셈블리로 구현할 경우 보안성도 강화 됨
- 단, 웹 어셈블리를 구현하기 위해 할당 된 메모리 범주를 초과해서는 안됨 → 오버플로우 발생

HSU HANSUNG UNIVERSITY

Q&A

CONTACT: tigerk9212@gmail.com

HSU
HANSUNG
UNIVERSITY

Thank You!

HSU
HANSUNG
UNIVERSITY