

Secure GCM Implementation on AVR

김경호

<https://youtu.be/N8zlh9U5vEY>

Contents

Galois/Counter Mode(GCM)

Polynomial Multiplication

Secure Binary Multiplication

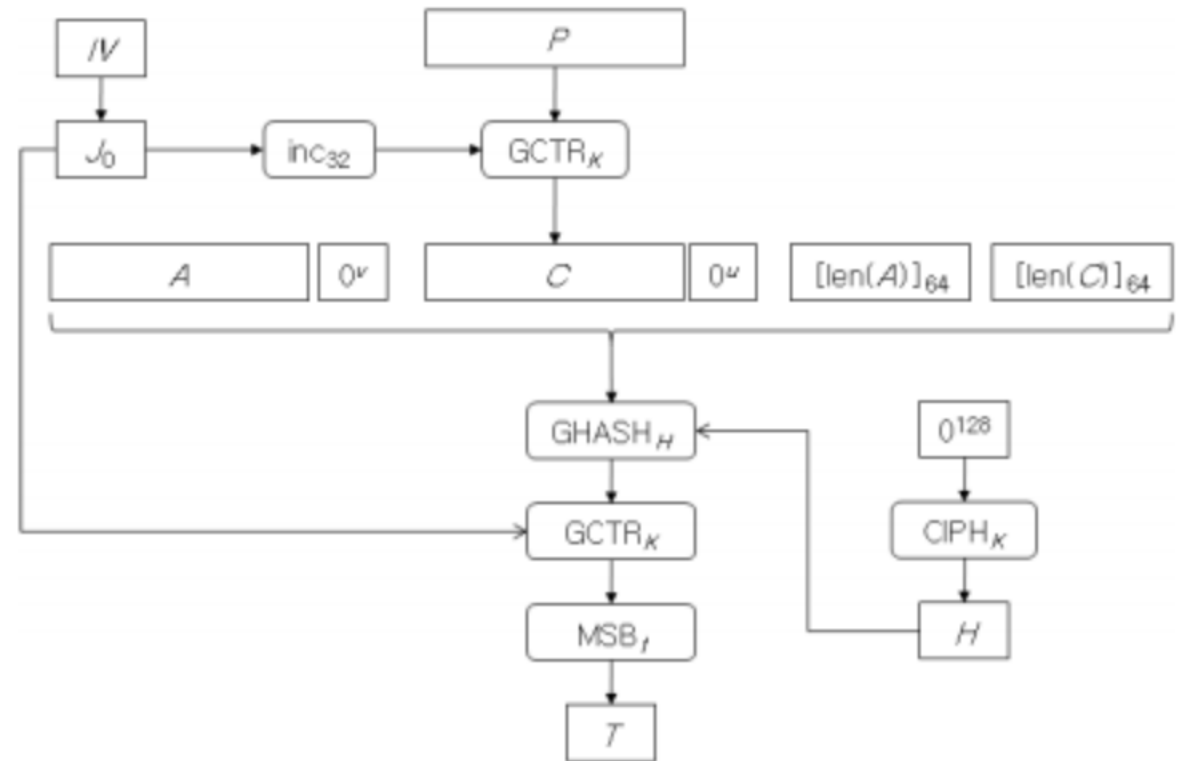
FACE-LIGHT

Conclusion



AES Galois/Counter Mode(GCM)

- 현재 가장 많이 사용하는 암호화 모드
- AES CTR 암호화 + Galois Message Authentication Code(GMAC)
- GMAC을 이용한 무결성 검증
- 128bit 블록 암호에서 사용
- $GF(2^{128})$ 곱 사용



Galois field

- 유한개의 원소를 가지는 field
- 한정된 비트를 가지는 컴퓨터에서 사용하기 적합
- $GF(x) \rightarrow x$ 의 크기 내 값에 각종 연산결과에 의한 값이 순환하는 Field
- 2진수 bit 연산 $\rightarrow GF(2^x)$
- Irreducible polynomial(기약 다항식)에 따라 연산 값 순환

- Ex) $GF(4)$
 - X^2+X+1

Addition					Multiplication					Division				
+	0	1	α	$1+\alpha$	\times	0	1	α	$1+\alpha$	x/y	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$	0	0	0	0	0	0	—	0	0	0
1	1	0	$1+\alpha$	α	1	0	1	α	$1+\alpha$	1	—	1	$1+\alpha$	α
α	α	$1+\alpha$	0	1	α	0	α	$1+\alpha$	1	α	—	α	1	$1+\alpha$
$1+\alpha$	$1+\alpha$	α	1	0	$1+\alpha$	0	$1+\alpha$	1	α	$1+\alpha$	—	$1+\alpha$	α	1


Galois field

- AES Mixcolumns 연산은 $GF(2^8)$
- Irreducible polynomial = $x^8 + x^4 + x^3 + x + 1$
- 8bit에서 유한체를 만들기 위하여 $x^8 = x^4 + x^3 + x + 1$ 를 XOR
- $x^4 + x^3 + x + 1 \rightarrow 0001\ 1011$
- 8bit를 초과할 경우 0001 1011(0x1b)를 XOR

45	43	9f	a8
9f	d2	40	7f
aa	43	33	9f
31	d8	71	15

ab	70	a7	40
b4	e1	3b	f9
c6	64	2a	cd
98	ff	2b	29

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02



45
9f
aa
31

<AES Mixcolumns>

Polynomial Multiplication

- GF를 사용하는 암호에서 Binary Multiplication 사용
- 곱셈은 연산시간이 크기 때문에 최적화 효율성
- Look Up Table을 이용한 Polynomial Multiplication
- 특정 비트 값에 따라 연산을 수행하는 Block-Comb method

LUT 사용 Polynomial Multiplication

Algorithm 3 Lopez et al. polynomial multiplication [17]

Require: Operand A and B

Ensure: Result R

- 1: $LUT \leftarrow U \cdot B$ where $U \in [0 \sim 15]$
 - 2: $R \leftarrow A_{HIGH} \cdot B$ with LUT and higher 4-bit of A
 - 3: $R \leftarrow R \ll 4$
 - 4: $R \leftarrow R \oplus A_{LOW} \cdot B$ with LUT and lower 4-bit of A
 - 5: **return** R
-

- LUT를 이용한 Polynomial Multiplication 경우 연산 결과가 저장된 LUT에 반복적으로 접근하는 동안 전력 소모를 분석하여 상관관계 분석을 통해 Key 값이 노출

Block-Comb Method

Algorithm 1 Constant exclusive-or operation with NOP [25]

Require: Operand A

Ensure: Result R

```
1: if exclusive operation is required then
2:    $R \leftarrow R \oplus A$            {normal exclusive-or operation}
3: else
4:   NOP                               {no operation}
5: end if
6: return  $R$ 
```

Algorithm 2 Constant exclusive-or operation with zero variable [25]

Require: Operand A

Ensure: Result R

```
1: if exclusive operation is required then
2:    $R \leftarrow R \oplus A$            {normal exclusive-or operation}
3: else
4:    $R \leftarrow R \oplus 0$          {exclusive-or operation with zero value}
5: end if
6: return  $R$ 
```

- 특정 bit의 값에 따라 연산을 수행(1이면 연산, 0이면 pass)
- 해당 방법은 Timing attack에 취약 -> NOP, Zero를 이용하여 해결
- But.. Nop, Zero 연산은 전력 소모가 다른 연산에 비해 작기때문에 전력파형 분석을 통해 취약점 생성

Secure Binary Multiplication

Algorithm 5 Masked Block Comb on 32-bit

Require: 32-bit wise operands A and B

Ensure: Result $C \leftarrow A \cdot B$

```
1: for  $i$  from 7 by 1 to 0 do
2:   for  $j$  from 3 by 1 to 0 do
3:      $BIT \leftarrow A[j] \& (1 \ll i)$ 
4:      $\{MASK, T0\} \leftarrow 0 - BIT$ 
5:     for  $k$  from 3 by 1 to 0 do
6:        $C[k + j] \leftarrow C[k + j] \oplus (B[k] \& MASK)$ 
7:     end for
8:   end for
9:    $C \leftarrow C \ll 1$ 
10: end for
11: return  $C$ 
```

1. BIT 변수에 $A[j]$ 의 i 번째 bit set
2. BIT이 1이면 $MASK = 0xff$, 0이면 $MASK = 0$
3. $MASK = 0xff$ 이면 동일한 결과 출력
4. $MASK = 0$ 이면 XOR 0 -> 결과에 지장 x
5. 결과 1비트 왼쪽 shift 연산 후 8번 반복

Table 1: AVR program codes for masked block comb multiplication

LDI BIT, 0X80	SUB ZERO, BIT	AND M0, BIT	EOR C0, M0
AND BIT, A0	SBC BIT, BIT	AND M1, BIT	EOR C1, M1
CLR ZERO	MOVW M0, B0	AND M2, BIT	EOR C2, M2
	MOVW M2, B2	AND M3, BIT	EOR C3, M3

Karatsuba Algorithm

- 효율적인 연산을 통해 곱셈 연산을 최적화하는 알고리즘

- 기본적인 곱셈 알고리즘

$$x = x_1 B^m + x_0$$

$$y = y_1 B^m + y_0$$

$$xy = (x_1 B^m + x_0)(y_1 B^m + y_0)$$

$$xy = x_1 y_1 B^{2m} + (x_1 y_0 + x_0 y_1) B^m + x_0 y_0$$

Ex) 1234 * 5678

$$12\ 34 = 12 \times 10^2 + 34$$

$$56\ 78 = 56 \times 10^2 + 78$$

$$z_2 = 12 \times 56 = 672$$

$$z_0 = 34 \times 78 = 2652$$

$$z_1 = (12 + 34)(56 + 78) - z_2 - z_0 = 46 \times 134 - 672 - 2652 = 2840$$

$$\text{결과} = z_2 \times 10^{2 \times 2} + z_1 \times 10^2 + z_0 = 672 \times 10000 + 2840 \times 100 + 2652 = \mathbf{7006652}$$

- Karatsuba Algorithm

$$z_2 = x_1 y_1$$

$$z_0 = x_0 y_0$$

$$\begin{aligned} z_1 &= (x_1 y_1 + x_1 y_0 + x_0 y_1 + x_0 y_0) - x_1 y_1 - x_0 y_0 \\ &= x_1 y_0 + x_0 y_1 \end{aligned}$$

$$z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$$

Karatsuba Block Comb

Algorithm 6 64-bit Karatsuba Block Comb

Require: 64-bit wise operands A and B

Ensure: Result $C \leftarrow A \cdot B$

- 1: $L \leftarrow A[3, 0] \times B[3, 0]$
 - 2: $H \leftarrow A[7, 4] \times B[7, 4]$
 - 3: $M \leftarrow (A[7, 4] \oplus A[3, 0]) \times (B[7, 4] \oplus B[3, 0])$
 - 4: $M \leftarrow M \oplus L \oplus H$
 - 5: $C \leftarrow (H \ll 64) \oplus (M \ll 32) \oplus L$
 - 6: **return** C
-

$$z_2 = A[7,4] \times B[7,4]$$

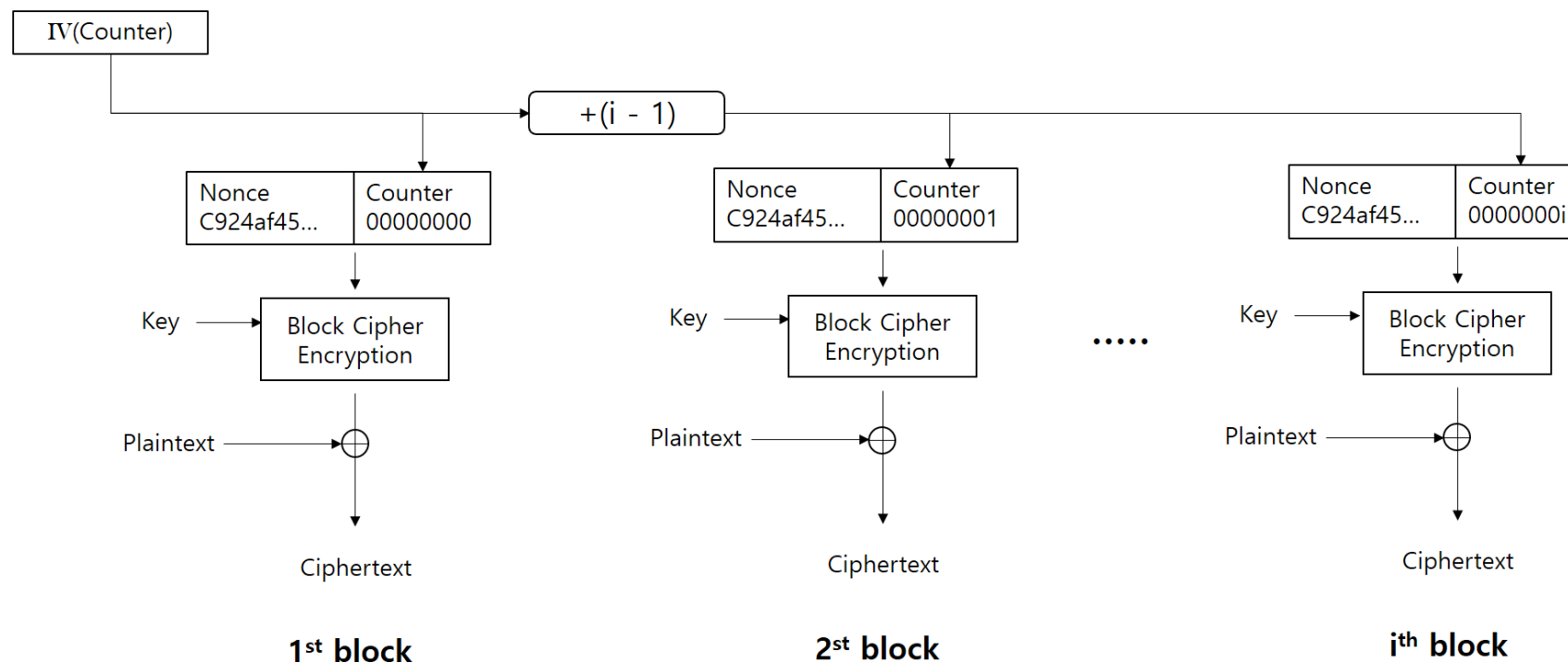
$$z_0 = A[3,0] \times B[3,0]$$

$$z_1 = (A[7,4] + B[3,0]) \times (B[7,4] + B[3,0]) - z_2 - z_0$$

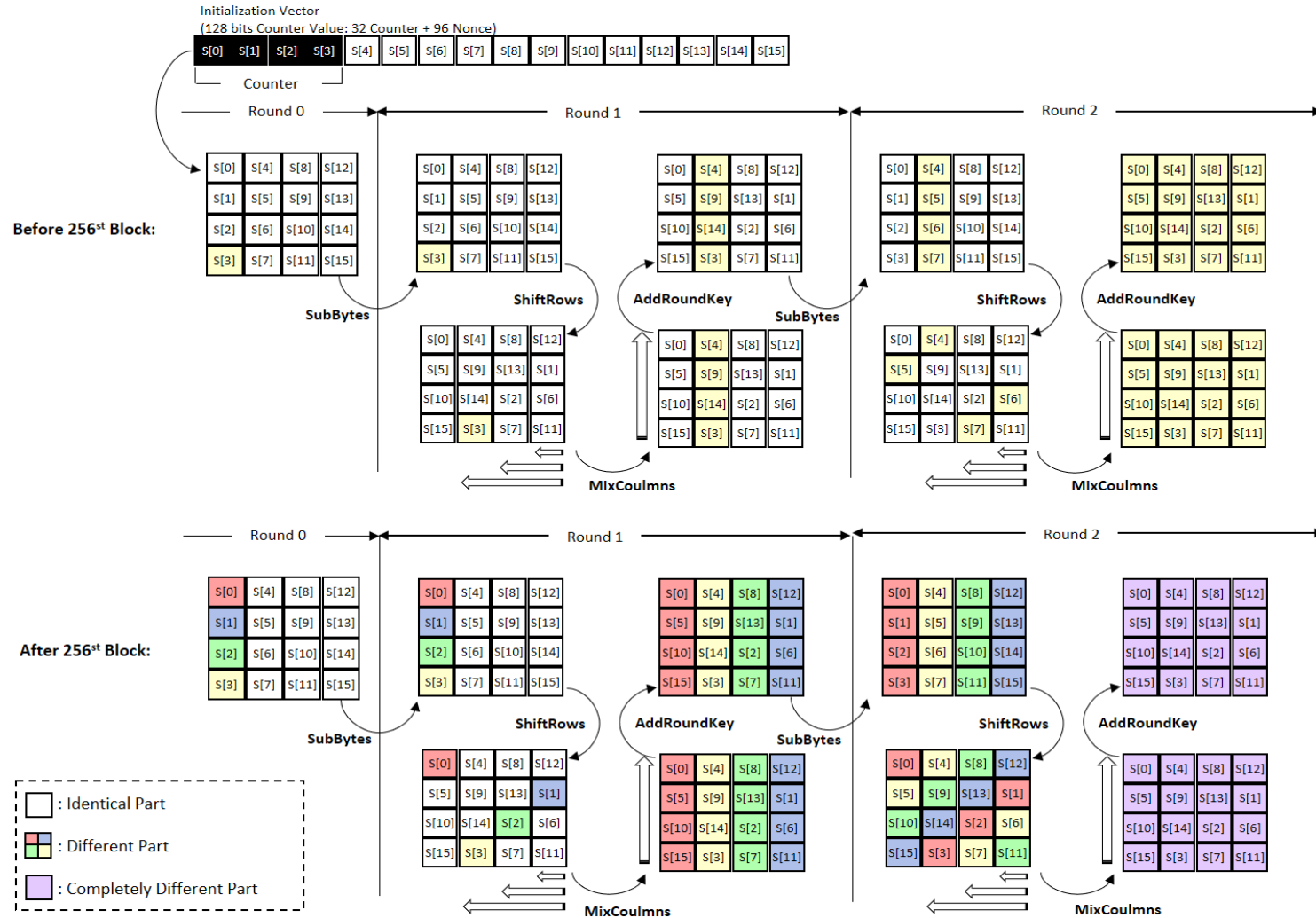
$$C = z_2 * 2^6 + z_1 * 2^5 + z_0$$

AES-CTR

- 128bits IV(Initial Vector)
 - 96bits Nonce
 - 32bits Counter
- Counter만 1씩 증가



FACE-LIGHT



Conclusion

- Polynomial Multiplication 최적화를 이용한 GHASH 함수 최적화
- FACE-LIGHT를 이용한 AES-CTR 함수 최적화
- 새로운 아이디어..?

Q & A

