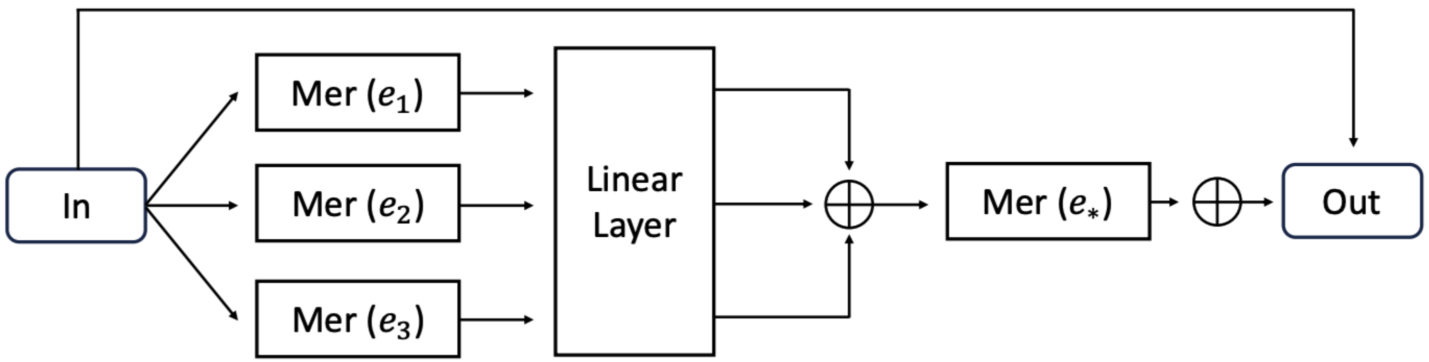
Quantum Circuit Implementation and Resource Analysis of AIM2

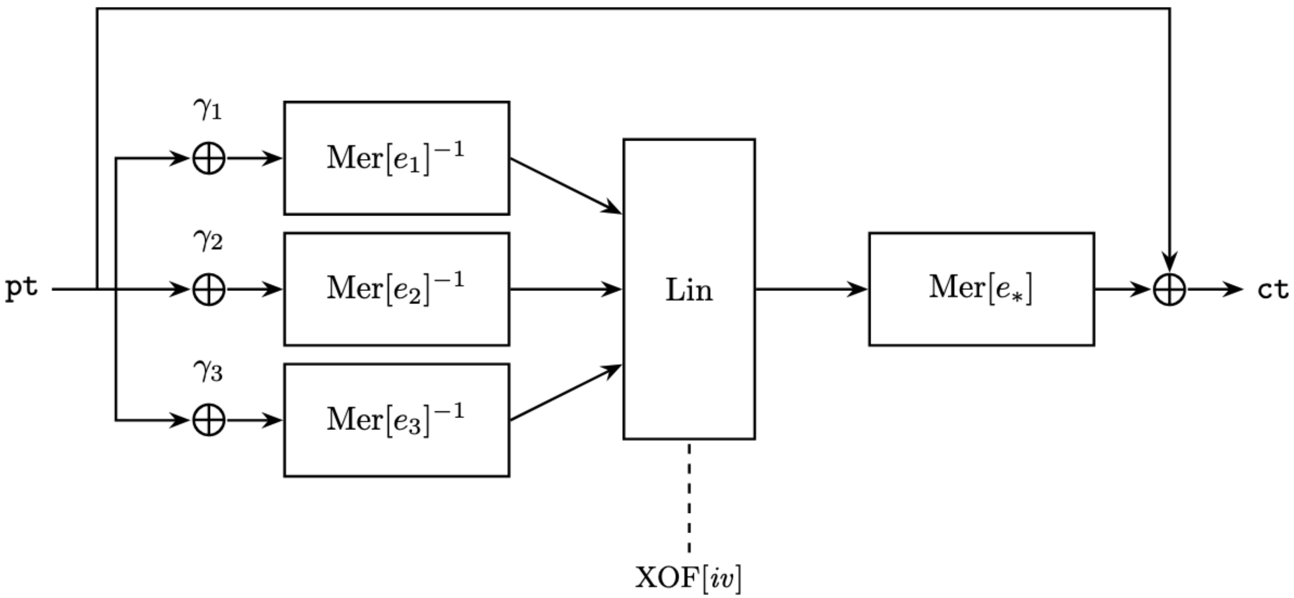
https://youtu.be/pM3z2Fa42IM

정보컴퓨터공학과 송경주

AIM vs AIM2

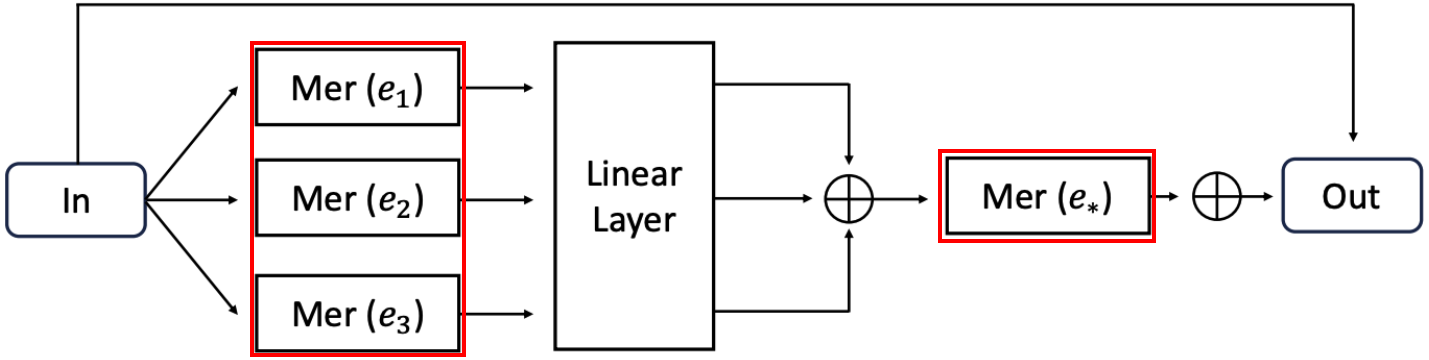


AIM

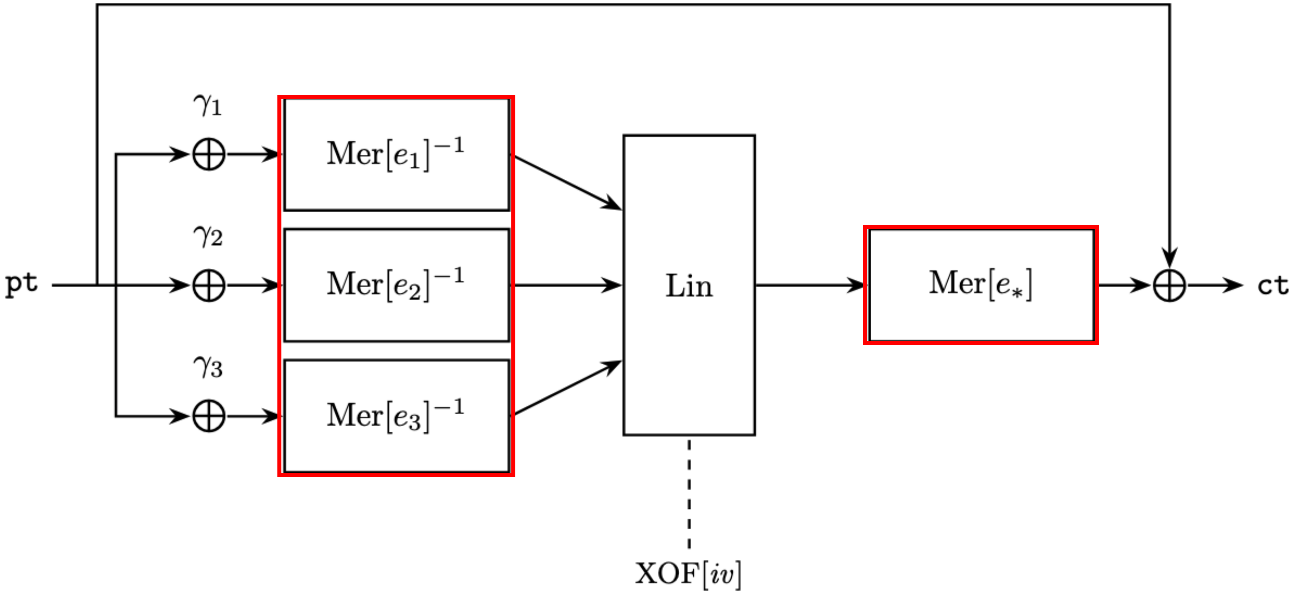


AIM2

AIM vs AIM2

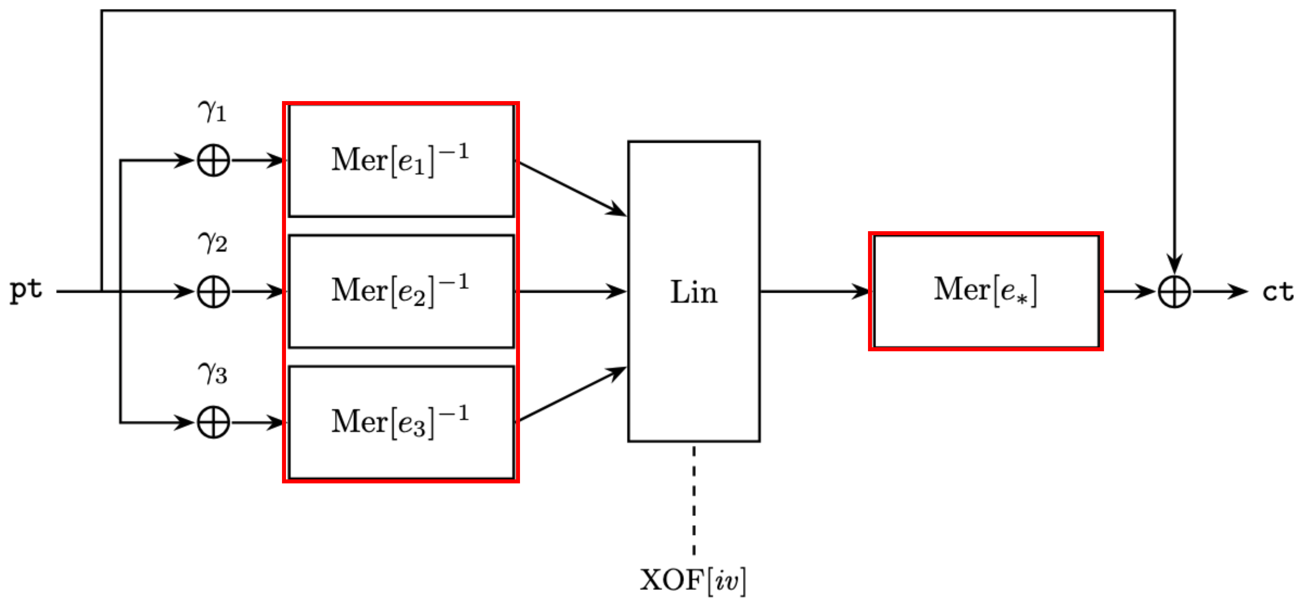


AIM



AIM2

AIM vs AIM2



AIM2

Mer

```c
// Mersenne exponentiation with e_star = 3
void GF_exp_mer_e_star(GF out, const GF in)
{
  GF t1 = {0,};

  // t1 = a ^ (2 ^ 2 - 1)
  GF_sqr_s(t1, in);
  GF_mul_s(t1, t1, in);

  // out = a ^ (2 ^ 3 - 1)
  GF_sqr_s(t1, t1);
  GF_mul_s(out, t1, in);
}
```

Mer^{-1}

```c
// t1 = in ^ 4
GF_sqr_s(table_d, in);
GF_sqr_s(t1, table_d);

// table_5 = in ^ 5
GF_mul_s(table_5, t1, in);
// table_6 = in ^ 6
GF_mul_s(table_6, table_5, in);
// table_a = in ^ 10 = (in ^ 5) ^ 2
GF_sqr_s(table_a, table_5);
// table_b = in ^ 11
GF_mul_s(table_b, table_5, table_6);
// table_d = in ^ 13
GF_mul_s(table_d, table_b, table_d);

// table_b = in ^ (0xb6), table_5 = in ^ (0xb5)
GF_sqr_s(t1, table_b);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(table_5, t1, table_5);
GF_mul_s(table_b, t1, table_6);

// t1 = in ^ (0xb6 d)
GF_sqr_s(t1, table_b);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(t1, t1, table_d);

// t1 = in ^ (0xb6d 6)
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(t1, t1, table_6);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5 b6)
for (i = 0; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_b);

// out = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5b6 b6d6dadb5)
for (i = 0; i < 36; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(out, t1, table_5);
```

```c
// t1 = in ^ (0xb6d6 d)
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(t1, t1, table_d);

// t1 = in ^ (0xb6d6d a)
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(t1, t1, table_a);

// t1 = in ^ (0xb6d6da d)
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_sqr_s(t1, t1);
GF_mul_s(t1, t1, table_d);

// table_5 = in ^ (0xb6d6dad b5)
for (i = 0; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(table_5, t1, table_5);

// t1 = in ^ (0xb6d6dadb5 b6)
GF_sqr_s(t1, table_5);
for (i = 1; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_b);

// t1 = in ^ (0xb6d6dadb5b6 b6d6dadb5)
for (i = 0; i < 36; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_5);
```

```c
// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5 b6)
for (i = 0; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_b);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6 b6d6dadb5)
for (i = 0; i < 36; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_5);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5 b6)
for (i = 0; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_b);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5b6 b6d6dadb5)
for (i = 0; i < 36; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_5);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5 b6)
for (i = 0; i < 8; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_b);

// t1 = in ^ (0xb6d6dadb5b6b6d6dadb5b6b6d6dadb5b6b6d6dadb5b6 b6d6dadb5)
for (i = 0; i < 36; i++)
{
  GF_sqr_s(t1, t1);
}
GF_mul_s(t1, t1, table_5);
```

AIM2 functions

- AIM2-I
  - $Mer^{-1}(49)$, $Mer^{-1}(91)$ / $Mer(3)$
- AIM2-III
  - $Mer^{-1}(17)$, $Mer^{-1}(47)$ / $Mer(5)$
- AIM2-V
  - $Mer^{-1}(11)$, $Mer^{-1}(141)$, $Mer^{-1}(7)$ / $Mer(3)$

Component quantum circuit

- **Multiplication**: out-of-place quantum circuit (Karatsuba algorithm [1])
- **Squaring**: In-place quantum circuit

→ **Depth 최적화 구조**

Field size 2^n	Operation	#CNOT	#1qCliff	$\#T$	T-depth	#Qubit	Full depth
$n = 128$	mul	29867	4374	15309	4	6561	78
	squ	205	-	-	-	131	127
$n = 192$	mul	85577	10206	35721	4	15309	94
	squ	301	-	-	-	195	196
$n = 256$	mul	115558	13122	45927	4	19683	180
	squ	401	-	-	-	261	253

[1] Jang, K., Kim, W., Lim, S., Kang, Y., Yang, Y., Seo, H.: Optimized implementation of quantum binary field multiplication with toffoli depth one. In: International Conference on Information Security Applications. pp. 251–264. Springer (2022)

Mer^{-1}- optimization

```
def invmer_exp1(eng, input, ancilla):  1 usage
    n = 128

    t1 = eng.allocate_qureg(n)
    table_a = eng.allocate_qureg(n)
    table_d_1 = eng.allocate_qureg(n)
    k = 127
    sqr_temp = [[eng.allocate_qubit() for _ in range(3)] for _ in range(k)]

    sqr_index = 0

    # t1 = in ^ 4
    with Compute(eng):
        copy(eng, input, table_d_1, n: 128)

        table_d_1 = Squaring_temp(eng, table_d_1, sqr_temp[sqr_index])
        sqr_index += 1

        copy(eng, table_d_1, t1, n: 128)
        t1 = Squaring_temp(eng, t1, sqr_temp[sqr_index])
        sqr_index += 1

    # table_5 = in ^ 5
    count1 = 0
    table_5_1 = []
    table_5_1, count1, ancilla = recursive_karatsuba(eng, t1, input, n, count1, ancilla)
    Reduction(eng, table_5_1)
```

계산된 중간 결과의 사용을 마침과 동시에 Inversion 연산에 필요한 요소들이 Forward 연산에 영향을 주지 않을 때 병렬로 inverse 동작

→ 큐비트 clean-up (내부에서 재사용 및 추후 linear component에서 재사용)

Mer^{-1}- optimization

```
t1_copy = eng.allocate_qureg(len(table_b))
copy(eng, table_b, t1_copy, len(table_b))

count4_1 = 0
table_b_2 = []
table_b_2, count4_1, ancilla = recursive_karatsuba(eng, table_b, table_6, n, count4_1, ancilla)
Reduction(eng, table_b_2)


count4_2 = 0
table_5_2 = []
table_5_2, count4_2, ancilla = recursive_karatsuba(eng, t1_copy, table_5_1, n, count4_2, ancilla)
Reduction(eng, table_5_2)

copy(eng, table_b, t1_copy, len(table_b))   # t1_copy: clean-up

Uncompute(eng)

# # table_d_1: clean-up
# # t1: clean-up
```

피 연산자가 중복될 경우, copy하여 병렬로 동작하도록 함.

Copy에 사용된 큐비트는 추후 재사용하므로 자원 소모 x
(2n개의 CNOT게이트 사용)

Linear component - optimization

```
invmer_exp1_output, clean_anc1, clean_anc2 = invmer_exp1(eng, state0, ancilla0)
invmer_exp2_output = invmer_exp2(eng, state1, ancilla1)

out = []

# linear component: affine layer
out = Matrix_Mul(eng, invmer_exp1_output, invmer_exp2_output, clean_anc1, clean_anc2)
```

```
def Matrix_Mul(eng, state0, state1, temp_U0, temp_U1):  1 usage

    Matrix_Mul_General(eng, state0, temp_U0, matrix_U0)

    out1 = eng.allocate_qureg(128)
    Matrix_Mul_General(eng, temp_U0, out1, matrix_L0)

    Matrix_Mul_General(eng, state1, temp_U1, matrix_U1)

    temp_L1 = eng.allocate_qureg(128)
    Matrix_Mul_General(eng, temp_U1, temp_L1, matrix_L1)

    for i in range(128):
        CNOT | (temp_L1[i], out1[i])

    return out1
```

Quantum resource for Mer

Table 3: Quantum resources required for the Mer of AIM2.

Cipher	Component	#CNOT	#T	#1qCliff	#Qubit	Full depth
AIM2-I	Mer(3)	68,508	30,618	8,748	8,754	410
AIM2-III	Mer(5)	240,325	107,163	30,618	25,911	1,061
AIM2-V	Mer(3)	205,924	91,854	26,244	26,254	668

Quantum resource for Mer^{-1}

Table 2: Quantum resources required for the Mer^{-1} of AIM2.

Cipher	Component	#CNOT	$\#T$	#1qCliff	#Qubit	Full depth
AIM2-I	$\mathsf{Mer}^{-1}(49)$	910,893	367,416	104,976	57,627	11,053
	$\mathsf{Mer}^{-1}(91)$	925,597	367,416	104,976	57,372	11,154
AIM2-III	$\mathsf{Mer}^{-1}(17)$	1,716,611	714,420	204,120	113,607	37,970
	$\mathsf{Mer}^{-1}(47)$	2,409,213	1,107,351	316,386	170,547	39,529
AIM2-V	$\mathsf{Mer}^{-1}(11)$	2,089,589	1,010,394	288,684	145,757	73,076
	$\mathsf{Mer}^{-1}(141)$	2,167,382	1,056,321	301,806	152,168	65,596
	$\mathsf{Mer}^{-1}(7)$	1,808,892	872,613	249,318	125,934	65,780

AIM vs AIM2

Table 5: Estimated quantum resources for the AIM2 quantum circuits.

Cipher	#CNOT	#1qCliff	$\#T$	#Qubit	Full depth	$FD \times M$
AIM-I [7]	358,754	39,430	137,781	25,299	3,499	88,521,201
AIM-III [7]	1,144,536	132,785	464,373	88,395	8,583	758,694,285
AIM-V [7]	1,486,100	157,588	551,124	108,072	16,857	1,821,769,704
AIM2-I (our)	1,921,984	218,768	765,450	119,763	11,861	1,420,508,943
AIM2-III (our)	4,403,948	551,439	1,928,934	300,819	41,041	12,345,912,579
AIM2-V (our)	6,371,395	866,052	3,031,182	476,613	74,759	35,631,111,267

AIM2 Grover's cost & post-quantum security

Table 6: Costs of the Grover's key search for AIM2

Cipher	Total gates	Total depth	Cost (complexity)	#Qubit	$FD \times M$
AIM2-I	1.09×2^{86}	1.14×2^{78}	1.24×2^{164}	119,764	1.04×2^{95}
AIM2-III	1.29×2^{119}	1.97×2^{111}	1.27×2^{231}	300,820	1.13×2^{130}
AIM2-V	1.92×2^{15}	1.79×2^{144}	1.72×2^{296}	476,614	1.63×2^{163}

Table 7: Comparison of the Grover's key search costs

Post-quantum Security	NIST'16 [14] (based on [3])	NIST'22 [15] (based on [8])	AIM2		
			-I	-III	-V
Level-1 (AES-128)	2^{170}	2^{157}	2^{164}		
Level-3 (AES-192)	2^{233}	2^{221}		2^{231}	
Level-5 (AES-256)	2^{298}	2^{285}			2^{296}

Q & A