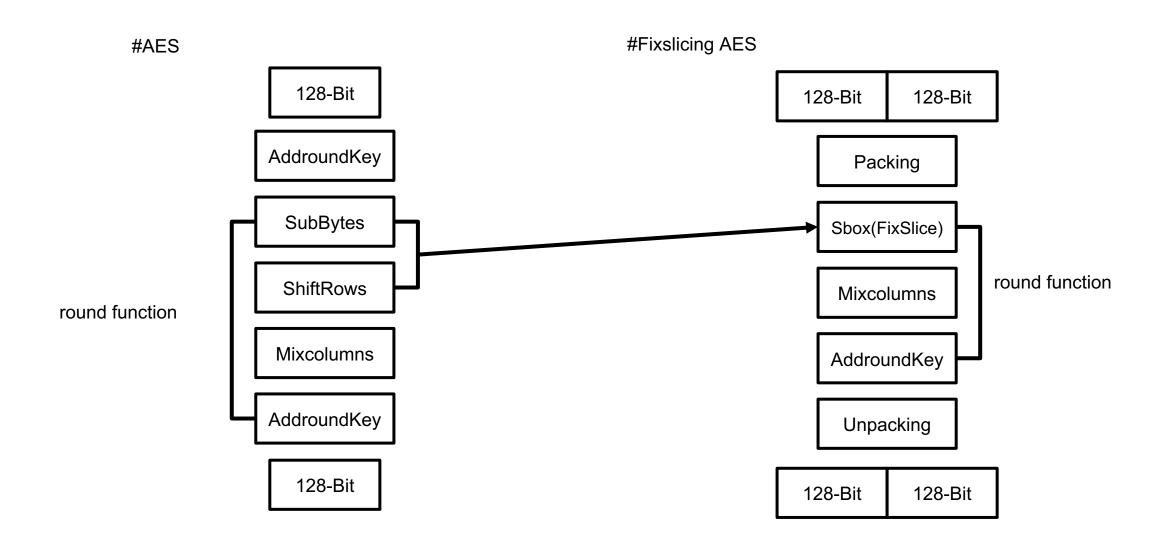
## FixSlicing AES

https://youtu.be/E2DKupsDGok





#### FixSlicing AES



### FixSlicing AES – Packing

				ro	w 3				 row 0								
	colu	mn 0	colu	mn 1	colu	mn 2	colu	nn 3	 colı	ımn 0	colu	mn 1	colu	mn 2	colu	mn 3	
	block 0	block 1	block 0	block 1	block 0	block 1	block 0	block 1	 block 0	block 1	block 0	block 1	block 0	block 1	block 0	block 1	
$R_0$	$b_{24}^{0}$	$b^1_{24}$	$b_{56}^{0}$	$b_{56}^{1}$	$b_{88}^{0}$	$b_{88}^{1}$	$b_{120}^0$	$b^1_{120}$	 $b_0^0$	$b_0^1$	$b_{32}^{0}$	$b_{32}^{1}$	$b_{64}^{0}$	$b_{64}^{1}$	$b_{96}^{0}$	$b_{96}^{1}$	
:	:	:	:	:	:	:	:	:	 :	÷	:	:	:	:	:	:	
$R_7$	$b_{31}^0$	$b^1_{31}$	$b_{63}^{0}$	$b_{63}^{1}$	$b_{95}^{0}$	$b_{95}^1$	$b_{127}^0$	$b^1_{127}$	 $b_7^0$	$b_7^1$	$b_{39}^{0}$	$b_{39}^1$	$b_{71}^0$	$b^1_{71}$	$b_{103}^0$	$b^1_{103}$	

<u>1100</u>	0011	1011	1111	0100	0001	0001	0000	c3bf4110
1100	0011	1011	1111	0100	0001	0001	0000	-
1000	0111	1010	0110	1001	0101	1110	0100	87a695e4
1000	0111	1010	0110	1001	0101	1110	0100	
0011	1011	1100	1010	1111	0010	0110	1000	3bcaf268
0011	1011	1100	1010	1111	0010	0110	1000	
0000	0000	0000	0000	0000	0000	0000	0101	5
0000	0000	0000	0000	0000	0000	0000	0101	
				ti.	,			
1111	0000	1111	1100	0011	1100	0011	0000	f0fc3c30
1100	0000	0000	1100	1100	1100	0011	1100	c00ccc3c
0000	1100	1111	0000	0000	1100	0011	1100	0cf00c3c
0000	1100	1100	0000	0011	1100	1100	0000	0cc03cc0
0000	1100	1100	1100	0000	0000	0000	1100	0ccc000c
0011	0000	1111	0000	0011	0000	0011	0011	30f03033
1111	1100	1111	1100	0000	1100	0000	0000	fcfc0c00
1111	1100	1100	0000	1111	0000	0000	0011	fcc0f003
	1100 1000 0011 0011 0000 0000 1111 1100 0000 0000 0000 0011 1111	1100 0011 1000 0111 1000 0111 0011 1011 0011 1011 0000 0000 0000 0000 1111 0000 1100 0000 0000 1100 0000 1100 0001 1000 011 0000 1111 1100	1100 0011 1011 1000 0111 1010 1000 0111 1010 0011 1011 1100 0011 1011 1100 0000 0000 0000 0000 0000 0000 1111 1100 0000 1100 0000 1100 1100 0000 1100 1100 0001 0000 1111 1111 1100 1111	1100 0011 1011 1111 1000 0111 1010 0110 1000 0111 1010 0110 0011 1011 1	1100 0011 1011 1111 0100 1000 0111 1010 0110 1001 1000 0111 1010 0110 1001 0011 1011 1100 1010 1111 0011 1011 1100 1010 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000  1111 0000 1111 1100 0011 1100 0000 1100 1100 0000 0000 1100 1100 1100 0000 0011 0000 1111 0000 0011 1111 1100 1111 1100 0000	1100 0011 1011 1111 0100 0001 1000 0111 1010 0110 1001 0101 1000 0111 1010 0110 1001 0101 0011 1001 1011 1100 1010 1111 0010 0011 1011 1100 1010 1111 0010 0000 0000 0000 0000 0000 0000 0000 0000	1100 0011 1011 1111 0100 0001 0001 1000 0111 1010 0110 1001 0101 1110 1110 1001 0111 1110 1110 1001 1111 1110 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 1111 1110 0000 1110 0000 1110 1111 1110 1111 1110 0000 1110 0000 1110 1111 1110 0000 1110 0000 1110 1111 1110 0000 1110 0000 0011 1111 1110 0000 1110 0000 0011 1111 1110 0000 1110 0000 1110 0000 0011 1111 1110 0000 1110 0000 0011	1100 0011 1011 1111 0100 0001 0001 000

#### FixSlicing AES – Packing

#### COLUMN

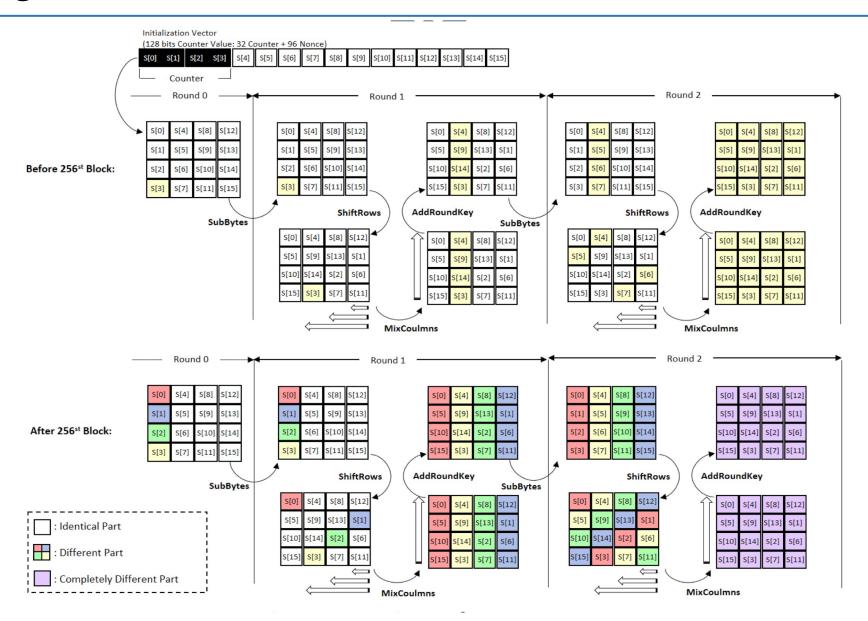
0	1	2	3
A0	A4	A8	A12
A1	A5	A9	A13
A2	A6	A10	A14
A3	A7	A11	A15
	A0 A1 A2	A0 A4 A1 A5 A2 A6	A0 A4 A8 A1 A5 A9 A2 A6 A10

ROW	0	1	2	3

0	В0	B4	B8	B12
1	B1	B5	B9	B13
2	B2	В6	B10	B14
3	В3	В7	B11	B15

			I	<b>30</b>	W3	}				ROW2						ROW1						ROW0										
	CC		CC 1	DL	CO			OL 3	CC		C	DL 1	CO		C		CC		C	) J		DL 2	CO		CC		C	OL 1	CC	- 1	C(	
stat	e[0]																															
	A 0 3	B 0 3	A 0 7	B 0 7	A 1 1	B 1 1	A 1 5	B 1 5	A 0 2	B 0 2	A 0 6	B 0 6	A 1 0	B 1 0	A 1 4	B 1 4	A 0 1	B 0 1	A 0 5	B 0 5	A 0 9	B 0 9	A 1 3	B 1 3	A 0 0	B 0 0	A 0 4	B 0 4	A 0 8	B 0 8	A 1 2	B 1 2

#### FixSlicing AES – CTR



#### FixSlicing AES – CTR

```
void aes128_encrypt_ffs(unsigned char *ctext0, unsigned char *ctext1,
                    const unsigned char* ptext0, const unsigned char* ptext1,
                    const uint32_t* rkeys_ffs) {
    uint32_t state[8];
                                          // 256-bit internal state
   packing(state, ptext0, ptext1);
                                          // packs into bitsliced representation
    ark(state, rkeys_ffs);
                                          // key whitening
    sbox(state);
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 8);
    sbox(state);
                                         // 2nd round
    mixcolumns_1(state);
                                         // 2nd round
    ark(state, rkeys_ffs + 16);
                                        // 2nd round
    sbox(state);
                                         // 3rd round
    mixcolumns_2(state);
                                        // 3rd round
    ark(state, rkeys_ffs + 24);
                                        // 3rd round
    sbox(state);
                                        // 4th round
                                        // 4th round
    mixcolumns_3(state);
    ark(state, rkeys_ffs + 32);
                                        // 4th round
    sbox(state);
                                        // 5th round
                                         // 5th round
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 40);
                                        // 5th round
    sbox(state);
    mixcolumns_1(state);
                                        // 6th round
    ark(state, rkeys_ffs + 48);
                                        // 6th round
    sbox(state);
                                         // 7th round
    mixcolumns_2(state);
                                        // 7th round
    ark(state, rkeys_ffs + 56);
                                        // 7th round
    sbox(state);
                                         // 8th round
                                        // 8th round
    mixcolumns_3(state);
   ark(state, rkeys_ffs + 64);
                                        // 8th round
    sbox(state);
                                        // 9th round
    mixcolumns_0(state);
                                         // 9th round
    ark(state, rkeys_ffs + 72);
                                        // 9th round
    sbox(state);
                                         // 10th round
    double_shiftrows(state);
                                        // 10th round (resynchronization)
    ark(state, rkeys_ffs + 80);
    unpacking(ctext0, ctext1, state);
                                        // unpacks the state to the output
```

```
void aes128_encrypt_ffs(unsigned char *ctext0, unsigned char *ctext1,
                    const unsigned char* ptext0, const unsigned char* ptext1,
                    const uint32 t* rkeys_ffs) {
    uint32_t state[8];
                                          // 256-bit internal state
    packing(state, ptext0, ptext1);
                                          // packs into bitsliced representation
    ark(state, rkeys_ffs);
                                          // key whitening
    sbox(state);
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 8);
    sbox(state);
                                        // 2nd round
    unpacking(ctext0, ctext1, state);
                                        // unpacks the state to the output
                                          // 2nd round
                                          // 2nd round
                                          // 3rd round
      mixcolumns 2(state):
                                          // 3rd round
                                         // 3rd round
                                          // 4th round
      mixcolumns 3(state);
                                          // 4th round
                                          // 4th round
                                          // 5th round
      mixcolumns_0(state);
                                          // 5th round
      ark(state, rkeys_ffs + 40);
                                          // 5th round
                                         // 6th round
                                          // 6th round
                                         // 6th round
      sbox(state);
                                          // 7th round
                                          // 7th round
      mixcolumns_2(state);
                                          // 7th round
                                          // 8th round
      mixcolumns_3(state);
                                          // 8th round
```

#### FixSlicing AES – CTR

	cou	nt =	00 00	0 00 0	00		c	ount =	= 00 0	0 00 0	)1	с	ount =	= 00 0	0 00 0	)2		count	= 00 0	0 00	ff		count = 00 00 00 00					
	5	0	1	2	3		s	0	1	2	3	s	0	1	2	3	s	0	1	2	3		s	0	1	2	3	
(	) (	06	d8	61	c2		0	06	ea	61	c2	0	06	fa	61	c2	0	06	f5	61	c2		0	0	4	8	12	
1	1	f9	d9	46	b1		1	f9	d9	e8	b1	1	f9	d9	c5	b1	1	f9	d9	62	b1		1	1	5	9	13	
2	2 !	5c	38	57	e0		2	5c	38	57	9f	2	5c	38	57	31	2	5c	38	57	a6		2	2	6	10	14	
3	3 (	01	03	59	0f		3	0e	03	59	0f	3	65	03	59	0f	3	94	03	59	Of		3	3	7	11	15	
	cou	nt =	00 0	0 00 0	00		С	ount =	00 0	0 01 0	00	с	ount =	00 0	0 0a 0	10	с	ount =	00 0	00 0	0							
	5	0	1	2	3		s	0	1	2	3	s	0	1	2	3	S	0	1	2	3							
(	) (	06	d8	61	c2		0	06	d8	50	c2	0	06	d8	bb	c2	0	06	d8	58	c2							
1	1	f9	d9	46	b1		1	f9	d9	46	1c	1	f9	d9	46	2d	1	f9	d9	46	c1							
2	2 !	5c	38	57	e0		2	44	38	57	e0	2	01	38	57	e0	2	98	38	57	e0							
	3 (	01	03	59	Of		3	01	bd	59	0f	3	01	6d	59	Of	3	01	0e	59	Of							
	cou	nt =	00 0	0 00 0	00		c	ount =	00 0	1 00 0	00	с	ount =	00 0	a 00 0	10		count	= 00 f	f 00 0	)							
	5	0	1	2	3		s	0	1	2	3	s	0	1	2	3	s	0	1	2	3							
(	) (	06	d8	61	c2		0	06	d8	61	13	0	06	d8	61	53	0	06	d8	61	69							
1	1	f9	d9	46	b1		1	62	d9	46	b1	1	8d	d9	46	b1	1	59	d9	46	b1							
2	2 !	5c	38	57	e0		2	5c	31	57	e0	2	5c	00	57	e0	2	5c	03	57	e0							
- 13	3 (	01	03	59	Of		3	01	03	19	0f	3	01	03	85	Of	3	01	03	2e	Of							
		00	00 0			l		ount =			0.0		ount =					count			10.							
	+	0	1	2	3		S	0	1	2	3	s	0	1	2	3	S	0	1	2	3							
(		06	d8	61	c2		0	70	d8	61	c2	0	ee	d8	61	c2	0	d2	d8	61	c2							
	-	f9	d9	46	b1		1	f9	fd	46	b1	1	f9	7f	46	b1	1	f9	b6	46	b1							
2		5c	38	57	e0		2	5c	38	f9	e0	2	5c	38	12	e0	2	5c	38	2f	e0							
3	3 (	01	03	59	Of		3	01	03	59	8d	3	01	03	59	ee	3	01	03	59	af							

#### FixSlicing AES – CTR 구헌

```
void Pre_Table(unsigned char* IV, const uint32_t* rkeys_ffs, u8 pretable[4][256][4])
   u8 temp_IV[16];
   for(int i=0; i<16; i++) {
        if(i<4) temp_IV[i] = IV[i]+1;</pre>
        else temp_IV[i] = IV[i];
   for(int i=0; i<256; i+=2){
        uint32_t state[8];
        u8 temp_1[16] = \{0x0, \};
        u8 temp_2[16] = \{0x0, \};
        packing(state, IV, temp_IV);
        ark(state, rkeys_ffs);
                                               // key whitening
        sbox(state);
        mixcolumns_0(state);
        ark(state, rkeys_ffs + 8);
                                               // 1st round
        sbox(state);
                                                // 2nd round
        unpacking(temp_1, temp_2, state);
        for(int j=0; j<4; j++){
            if(j==0){
                pretable[j][i][0] = temp_1[3];
                pretable[j][i][1] = temp_1[4];
                pretable[j][i][2] = temp_1[9];
                pretable[j][i][3] = temp_1[14];
                pretable[j][i+1][0] = temp_2[3];
                pretable[j][i+1][1] = temp_2[4];
                pretable[j][i+1][2] = temp_2[9];
                pretable[j][i+1][3] = temp_2[14];
```

```
void aes128_encrypt_ffs(unsigned char *ctext0, unsigned char *ctext1,
                    const unsigned char* ptext0, const unsigned char* ptext1,
                    const uint32 t* rkeys_ffs) {
    uint32_t state[8];
                                          // 256-bit internal state
    packing(state, ptext0, ptext1);
                                          // packs into bitsliced representation
    ark(state, rkeys_ffs);
                                          // key whitening
    sbox(state);
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 8);
    sbox(state);
                                        // 2nd round
                                        // unpacks the state to the output
    unpacking(ctext0, ctext1, state);
                                          // 2nd round
     mixcolumns_1(state);
                                         // 2nd round
                                          // 3rd round
      mixcolumns_2(state);
                                          // 3rd round
                                         // 3rd round
                                          // 4th round
     mixcolumns_3(state);
                                          // 4th round
                                         // 4th round
                                          // 5th round
                                          // 5th round
      mixcolumns_0(state);
      ark(state, rkeys_ffs + 40);
                                         // 5th round
                                         // 6th round
                                          // 6th round
                                         // 6th round
     sbox(state);
                                          // 7th round
                                          // 7th round
     mixcolumns_2(state);
                                         // 7th round
                                          // 8th round
                                          // 8th round
      mixcolumns_3(state);
```

#### FixSlicing AES – CTR 구현

```
void aes128_encrypt_ffs(unsigned char *ctext0, unsigned char *ctext1,
                    const unsigned char* ptext0, const unsigned char* ptext1,
                    const uint32_t* rkeys_ffs) {
    uint32_t state[8];
                                          // 256-bit internal state
   packing(state, ptext0, ptext1);
                                          // packs into bitsliced representation
    ark(state, rkeys_ffs);
    sbox(state);
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 8);
                                          // 1st round
    sbox(state);
                                         // 2nd round
    mixcolumns_1(state);
                                        // 2nd round
    ark(state, rkeys_ffs + 16);
                                        // 2nd round
    sbox(state);
                                        // 3rd round
    mixcolumns_2(state);
                                        // 3rd round
    ark(state, rkeys_ffs + 24);
                                        // 3rd round
    sbox(state);
                                        // 4th round
                                        // 4th round
    mixcolumns_3(state);
    ark(state, rkeys_ffs + 32);
                                        // 4th round
                                        // 5th round
    sbox(state);
                                        // 5th round
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 40);
                                        // 5th round
    sbox(state);
                                        // 6th round
    mixcolumns_1(state);
                                        // 6th round
    ark(state, rkeys_ffs + 48);
                                        // 6th round
    sbox(state);
                                        // 7th round
    mixcolumns_2(state);
                                        // 7th round
    ark(state, rkeys_ffs + 56);
                                        // 7th round
    sbox(state);
                                        // 8th round
                                        // 8th round
    mixcolumns_3(state);
    ark(state, rkeys_ffs + 64);
                                        // 8th round
    sbox(state);
                                        // 9th round
    mixcolumns_0(state);
                                        // 9th round
    ark(state, rkeys_ffs + 72);
                                        // 9th round
    sbox(state);
                                        // 10th round
    double_shiftrows(state);
                                        // 10th round (resynchronization)
    ark(state, rkeys_ffs + 80);
    unpacking(ctext0, ctext1, state);
                                        // unpacks the state to the output
```

```
void ctr_aes128_encrypt_ffs(unsigned char *ctext0, unsigned char *ctext1,
                   const unsigned char* ptext0, const unsigned char* ptext1,
                   const uint32 t* rkeys ffs) {
    uint32 t state[8];
                                          // 256-bit internal state
    packing(state, ptext0, ptext1);
                                          // packs into bitsliced representation
                                            // key whitening
      sbox(state);
                                          // 1st round
     mixcolumns_0(state);
                                          // 1st round
                                          // 2nd round
    mixcolumns_1(state);
                                        // 2nd round
                                       // 2nd round
    ark(state, rkeys_ffs + 16);
    sbox(state);
                                        // 3rd round
    mixcolumns_2(state);
                                        // 3rd round
    ark(state, rkeys_ffs + 24);
                                       // 3rd round
    sbox(state);
                                        // 4th round
   mixcolumns_3(state);
                                        // 4th round
    ark(state, rkeys_ffs + 32);
                                       // 4th round
    sbox(state);
                                        // 5th round
                                        // 5th round
    mixcolumns_0(state);
    ark(state, rkeys_ffs + 40);
                                       // 5th round
    sbox(state);
                                       // 6th round
                                        // 6th round
    mixcolumns_1(state);
    ark(state, rkeys_ffs + 48);
                                       // 6th round
    sbox(state);
                                        // 7th round
    mixcolumns_2(state);
                                        // 7th round
    ark(state, rkeys_ffs + 56);
                                       // 7th round
                                        // 8th round
    sbox(state);
    mixcolumns_3(state);
                                        // 8th round
    ark(state, rkeys_ffs + 64);
                                       // 8th round
    sbox(state);
                                        // 9th round
    mixcolumns 0(state);
                                        // 9th round
    ark(state, rkeys_ffs + 72);
                                       // 9th round
                                        // 10th round
    sbox(state);
    double shiftrows(state);
                                        // 10th round (resynchronization)
    ark(state, rkeys_ffs + 80);
                                       // 10th round
    unpacking(ctext0, ctext1, state);
                                        // unpacks the state to the output
```

#### FixSlicing AES – CTR 성능

- 성능 테스트
  - 1MB 암호화 1000번 반복
  - ECB: 약 43초
  - CTR:약36초
  - 약 7초 정도의 차이 발생

```
clock_t start, end;

start = clock();

aes128_keyschedule_ffs(rk32, key8, key8);

for(int loop=0; loop<1000; loop++){

    for(int i=0; i<L00P_LEN; i++){

        aes128_encrypt_ffs(ct8_1, ct8_2, IV_0, IV_0, rk32);

        for(int i=0; i<16; i++) ct8_1[i] ^= pt8_1[i];

        for(int i=0; i<16; i++) ct8_2[i] ^= pt8_1[i];

    }

}

end = clock();

printf("타이더 : %.2f",((float)(end - start) / CLOCKS_PER_SEC));
```

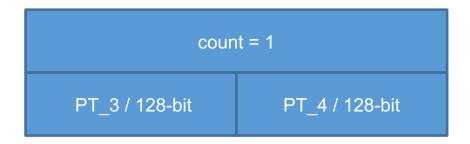
```
aes128_keyschedule_ffs(rk, key, key);
Pre_Table(IV_CNT1, rk, pretable);
for(int loop=0; loop<1000; loop++){
    for(int i=0; i<L00P_LEN; i++){
        u8 temp_1[16], temp_2[16];
        pre_state(pretable, IV_CNT1, IV_CNT2, temp_1, temp_2);
        ctr_aes128_encrypt_ffs(&ct_1[i*16], &ct_2[i*16], temp_1, temp_2, rk);
        IV32_1[0] += 2;
        IV32_2[0] += 2;
    }
    for(int i=0; i<PT_LEN; i++) ct_1[i] ^= pt_1[i];
    for(int i=0; i<PT_LEN; i++) ct_2[i] ^= pt_2[i];
}
end = clock();
printf("E|O|D| : %.2f",((float)(end - start) / CLOCKS_PER_SEC));
```

#### 질문...

count = 0	count = 1
PT_1 / 128-bit	PT_2 / 128-bit

count = 2	count = 3
PT_3 / 128-bit	PT_4 / 128-bit

coun	nt = 0
PT_1 / 128-bit	PT_2 / 128-bit



# Q&A