# RC5 양자회로 구현

https://youtu.be/6595w2bUHKo

# RC5

- RC5 : 1994년 RSA security의 Rivest가 설계한 대칭키 암호
  - AES 후보 RC6는 RC5를 기반으로 설계된 암호

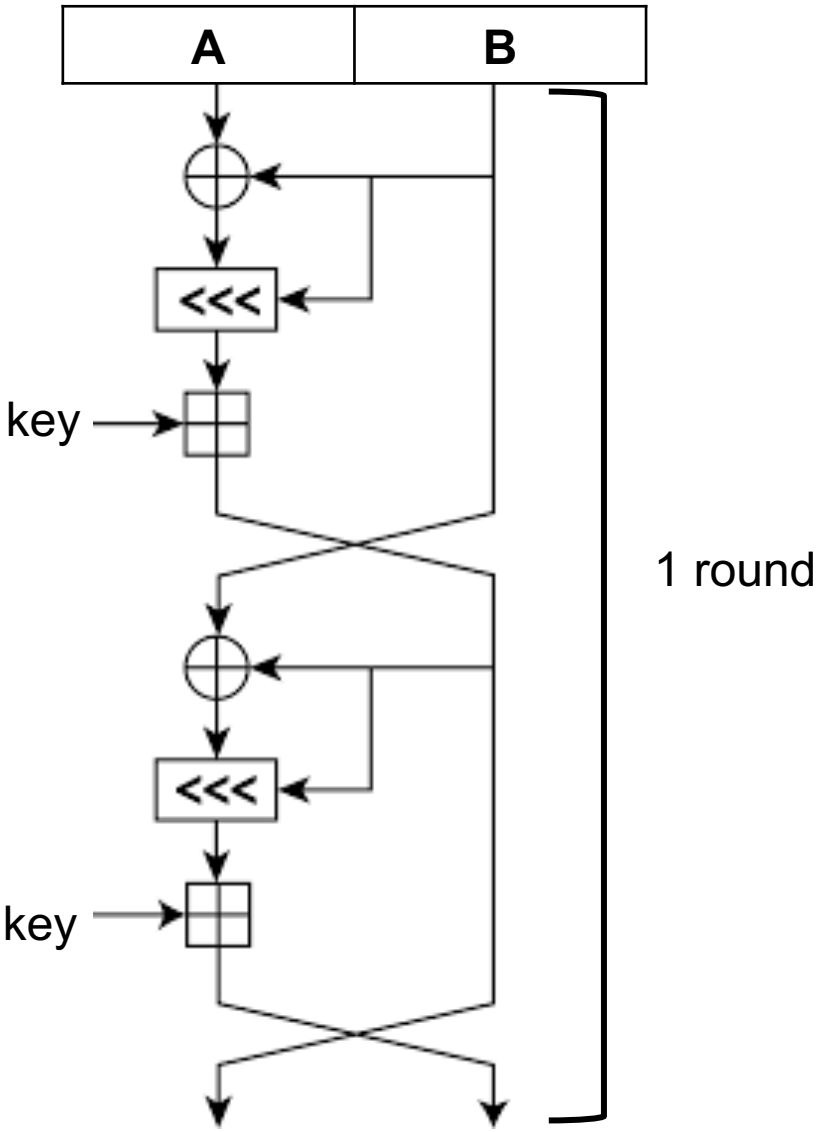
- 가변적인 블록사이즈(32,64,128) 키 사이즈(0~2040bits), 라운드 수(0~255)를 가짐.
  - 권장 블록 사이즈(64) 키 사이즈(128).

  - 사용자는 응용프로그램에 적합한 보안 수준을 선택
  - 사용자는 더 빠른 속도와 더 높은 보안 사이의 균형 선택
  - RC5는 메모리가 제한된 스마트 카드 또는 기타 장치에서 쉽게 구현될 수 있도록 낮은 메모리 요구

# RC5

Feistel 구조
  A =B = 32bits
  XOR, rotation,add 연산
  RC5는 데이터에 의존한 회전
-> 사전에 결정되지 않음.

Key schedule 단계:
  ex) 12라운드일 때?
128비트 키로 26개의 32비트(word) 배열 S 생성
  S 사이즈 = 2(라운드 수 +1)



# RC5 - c언어 구현

```c
#define w 32
#define r 12
#define b 16
#define t 26
#define c 4

unsigned int S[t];
unsigned int P = 0xb7e15163, Q = 0x9e3779b9;
//unsigned int c = (unsigned)(std::max(1, (cc
```

w = 워드
r = 라운드 수
b = 키 길이(byte) 16 x 8 = 128 (bits)
t = 확장된 키의 사이즈 2(r+1)
c = 워드 단위의 키 길이 32 x 4 = 128(bits)

```c
void RC5_SETUP(unsigned char *K) {
    unsigned  int i, j, k, u = w / 8, A, B, L[c];

    for(i = b - 1, L[c - 1] = 0; i != -1; i--)
        L[i / u] = (L[i / u] << 8) + K[i];
    for(S[0] = P, i = 1; i < t; i++)
        S[i]= S[i - 1] +Q;
    for(A = B = i = j = k = 0; k < 3 * t; k++, i = (i + 1) % t, j = (j + 1) %
        c) {
        A =  S[i] = ROTL(S[i] + (A + B), 3);
        B = L[j] = ROTL(L[j] + (A + B), (A + B));
    }
}
```

# RC5 - c언어 구현

```c
void RC5_ENCRYPT(unsigned  int *pt, unsigned  int *ct) {
    unsigned  int A = pt[0] + S[0], B = pt[1] + S[1];

    for(unsigned  int i = 1; i <= 12; i++) {
        A = ROTL(A ^ B, B) + S[2 * i];
        B = ROTL(B ^ A, A) + S[2 * i + 1];
    }
    ct[0] = A;

}
```

Key shedule에서 만든 배열 S를 가지고 연산

Left Rotation, add, XOR 연산 사용

```c
void RC5_DECRYPT(unsigned  int *ct, unsigned  int *pt) {
    unsigned  int B = ct[1], A = ct[0];

    for(unsigned  int i = 12; i > 0; --i) {
        B = ROTR(B - S[2 * i + 1], A) ^ A;
        A = ROTR(A - S[2 * i], B) ^ B;
    }
    pt[1] = B - S[1];
    pt[0] = A - S[0];
}
```

대칭키 암호이기때문에 Decryption에서는 역순으로 연산진행.

Right Rotation , sub,XOR 연산 사용

# RC5 - 양자회로 구현

```python
def RC5(eng):
    pt = eng.allocate_qureg(32)
    pt2 = eng.allocate_qureg(32)
    key = eng.allocate_qureg(128)
    S = eng.allocate_qureg(832)

    RC5_SETUP(eng,key,S)
    RC5_Encrypt(eng,pt,pt2,S)
```

S = 32 x 26 = 832(bits)

```python
def RC5_SETUP(eng,key,S):
    A = eng.allocate_qureg(32)
    B = eng.allocate_qureg(32)
    c = eng.allocate_qubit()

    SS=[0]*26
    SS[0] = 0xb7e15163
    QQ = 0x9e3779b9
    for i in range(1,26):
        SS[i] = (SS[i-1]+ QQ) & 0xFFFFFFFF

    for i in range(26):
        Round_constant_XOR(eng, S[32 * i:32 * (i + 1)], SS[i], 32)

    new_B = eng.allocate_qureg(32)
```

SS = 1word(32bits) x 26
S = 832 (bits)

# RC5 - 양자회로 구현

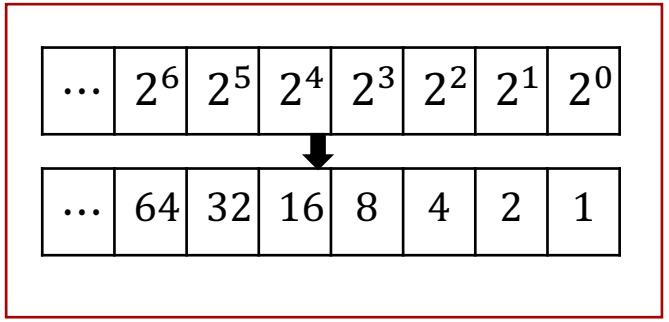
```
for i in range(78):
    CDKM(eng, B, S[(32 * (i % 26)):32 * ((i % 26) + 1)], c, 32)
    CDKM(eng, A, S[(32 * (i % 26)):32 * ((i % 26) + 1)], c, 32)
    A = S[(32 * (i % 26)):32 * ((i % 26) + 1)] = ROTL3(eng, S[(32 * (i % 26)):32 * ((i % 26) + 1)], 3)

    CDKM(eng, A, B, c, 32)
    CDKM(eng, B, key[32 * (i % 4): 32 * ((i % 4) + 1)], c, 32)
    left_rotation_a_b(eng,key[32 * (i % 4): 32 * ((i % 4) + 1)], B)

    if(i==0):
        copy(eng, key[32 * (i % 4): 32 * ((i % 4) + 1)], new_B, 32)
    else:
        CDKM_minus(eng,A,new_B,c,32)
        copy(eng,key[32 * ((i - 1) % 4): 32 * (((i - 1) % 4) + 1)],new_B,32)
        copy(eng,key[32 * (i % 4): 32 * ((i % 4) + 1)],new_B,32)

    B = new_B
```

```
def ROTL3(eng,A,n):
    new_A=[]
    for i in range(32-n,32):
        new_A.append(A[i])

    for i in range(29):
        new_A.append(A[i])

    return new_A

def left_rotation_a_b(eng, s, a_b):
    for i in range(5):
        with Control(eng, a_b[i]):
            for j in range(2 ** i):
                left_rotation_1bit(eng, s)

def left_rotation_1bit(eng, s):
    for i in range(31):
        Swap | (s[31 - i], s[30 - i])
```

Ex). i+1라운드일때

B(new_B) = key[i]

B(new_B) = A+B



# RC5 - 양자회로 구현

```
def RC5_Encrypt(eng,A,B,S):
    c = eng.allocate_qubit()

    CDKM(eng,S[0:32],A,c,32)
    CDKM(eng,S[32:64],B,c,32)

    for i in range(12): # (1,13):
        CNOT32(eng,B,A)
        left_rotation_a_b(eng,A,B)
        CDKM(eng,S[32*(2*i+2):32*(2*i+3)],A,c,32)
        CNOT32(eng,A,B)
        left_rotation_a_b(eng,B,A)
        CDKM(eng,S[32*(2*i+3):32*(2*i+4)],B,c,32)

    if(resource_check!=1):
        print_state(eng,A,32)
        print_state(eng,B,32)
```

## 회로 구현 양자 비용

| Cipher | Qubits | Clifford gates | T gates | Full depth |
|---|---|---|---|---|
| RC5 | 1122 | 394250 | 177051 | 256252 |

## Grover 공격 비용

| Cipher | Total gates | Total depth | Cost | Security |
|---|---|---|---|---|
| RC5 | $1.021 \cdot 2^{84}$ | $1.535 \cdot 2^{82}$ | $1.567 \cdot 2^{166}$ | Not achieved () |

# Q & A