**HOMEWORK – DES**

2011140138

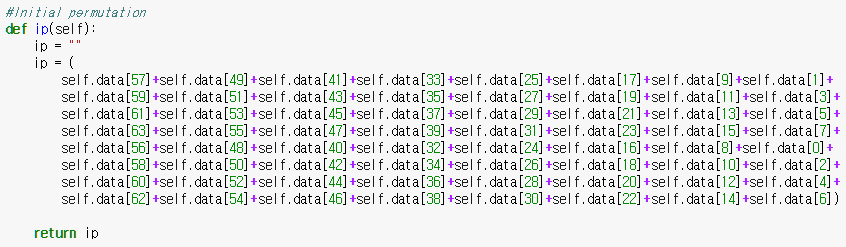
황 태림

1. **One round version of DES**

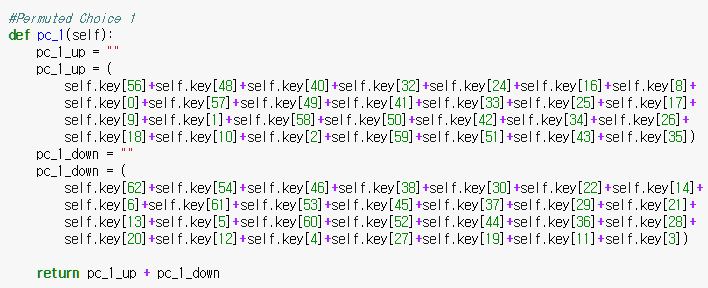
Input : 0123456789ABCDEF(key) / FEDCBA9876543210(plaintext)

* **Code(python)**

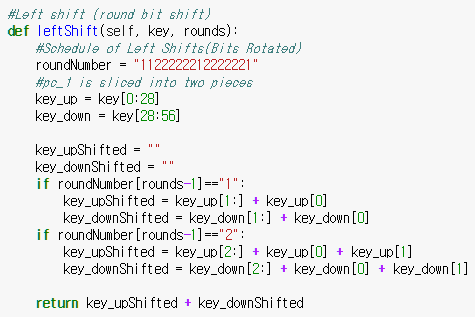
# Initial permutation



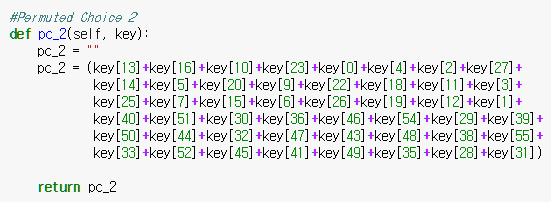
# Permuted Choice 1



# Left shift

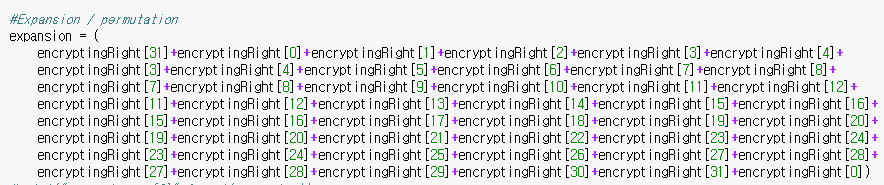


# Permuted Choice 2

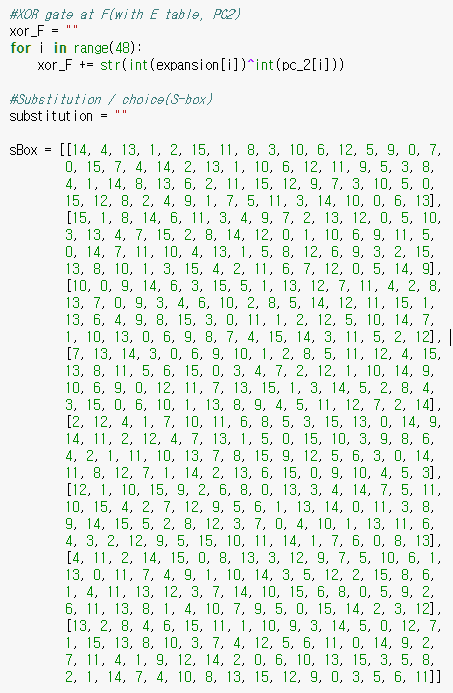


# Single round

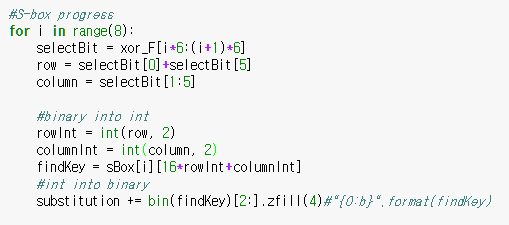
## Expansion / permutation



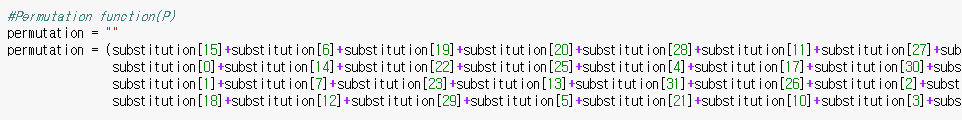
## XOR gate at F / Substitution (S-box)



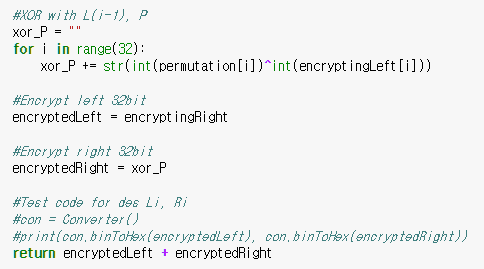
## S-box progress



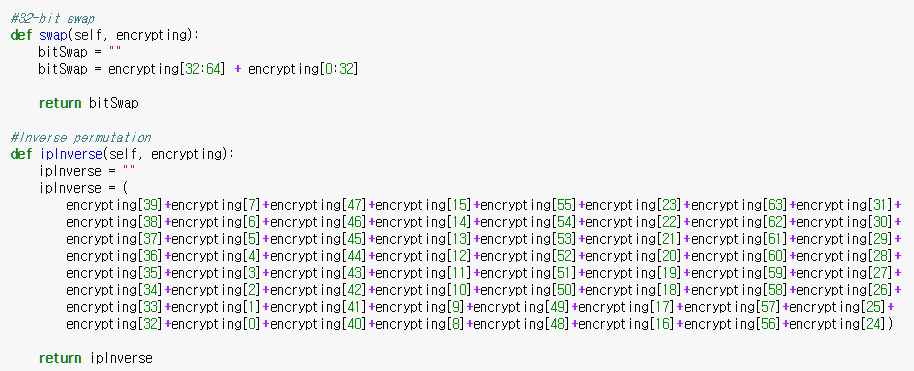
## Permutation function(P)



## XOR with L(i-1), P / return singleRound



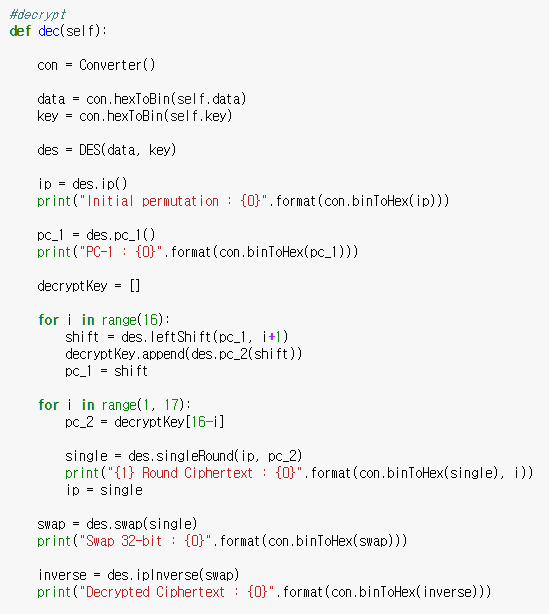
# 32bit swap / Inverse permutation



#DES encrypt



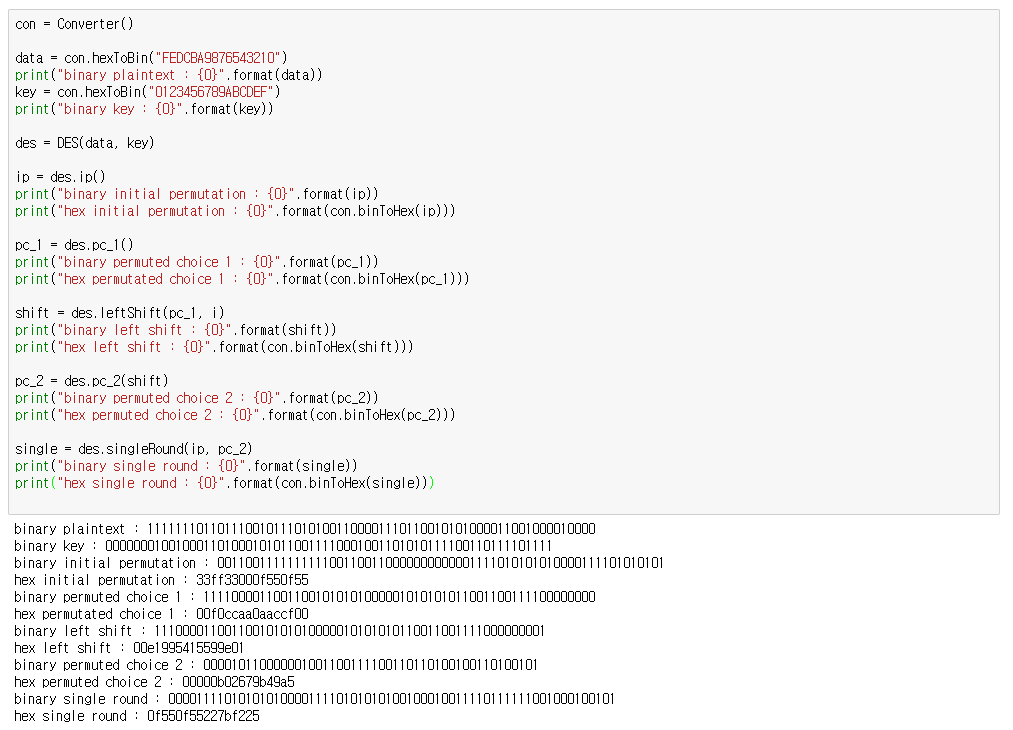
# DES decrypt



* **Description**

DES의 각 과정을 python으로 작성하였습니다. 각 table표는 python의 string이 배열처럼 사용할 수 있다는 점을 활용, 예로 64bit의 plaintext는 text[0]~text[63]까지의 index로 바뀌는 위치로 새로운 string을 만드는 방식을 사용했습니다.

* **Single round**



위와 같이 plaintext를 data에 입력하고, 이를 binary code로 변환 후 입력하여 총 64글자의 string binary code를 만들었습니다. key 또한 같은 처리를 하였습니다.

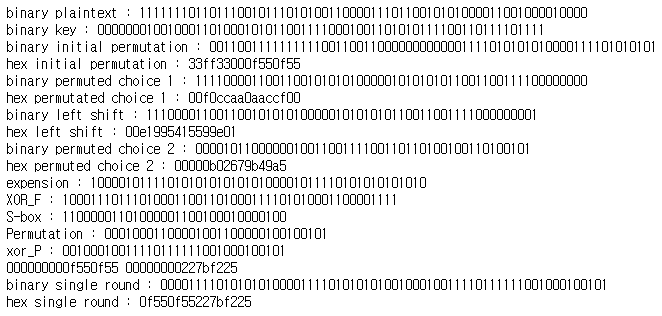
initial permutation 을 통해 ‘FEDCBA9876543210’ 는 ‘33ff33000f550f55’가 됩니다

key는 permuted choice 1을 통해 64bit->56bit 로 감소합니다. ‘0123456789ABCDEF’는 ‘00f0ccaa0aaccf00’이 됩니다.

이 값을 binary code 상에서 절반씩 나누고 left shift(round shift)를 해주면 ‘00e1995415599e01’의 값이 나옵니다.

이를 1차 변형된 initial permutation ciphertext와 XOR연산을 하기 위해 permuted choice 2 를 수행하면 ‘00000b02679b49a5’ 이 됩니다

다음 과정은 initial permutation ciphertext를 32bit 씩 쪼개고, R0은 L1, L0과 R0의 각 과정 처리 후 XOR 연산 결과는 R1로 만드는 round 입니다



expansion은 E table을 만드는데, 이를 통해 32bit -> 48bit가 된다

이 expansion된 48bit의 ciphertext와 permuted choice 2로 나온 key 48bit를 XOR연산(위에서 XOR\_F) 48bit의 ciphertext가 나온다

이 ciphertext를 S-box를 통해 처리하는데, 1, 6bit와 2~5bit로 나눠 각각 row, column으로 설정 후, 각 S-box에서 해당 열과 행을 찾아 10진법의 수를 다시 2진법으로 환산, 총 4bit X 8로 32bit의 S-box 값이 나온다

위의 결과는 permutation과정을 통해(위의 Permutation) L0과 연산될 ciphertext가 출력된다

L0과 permutation된 R0은 XOR를 통해(위의 xor\_P) 32bit의 최총 R1이 출력된다

L1은 R0이 그대로 내려온다

위의 그림에서 xor\_P 밑에 있는 두 hex값이 L1, R1이다

이 둘을 합쳐서 single round가 종료되면 위 그림의 hex single round 에 해당하는 ciphertext가 출력된다

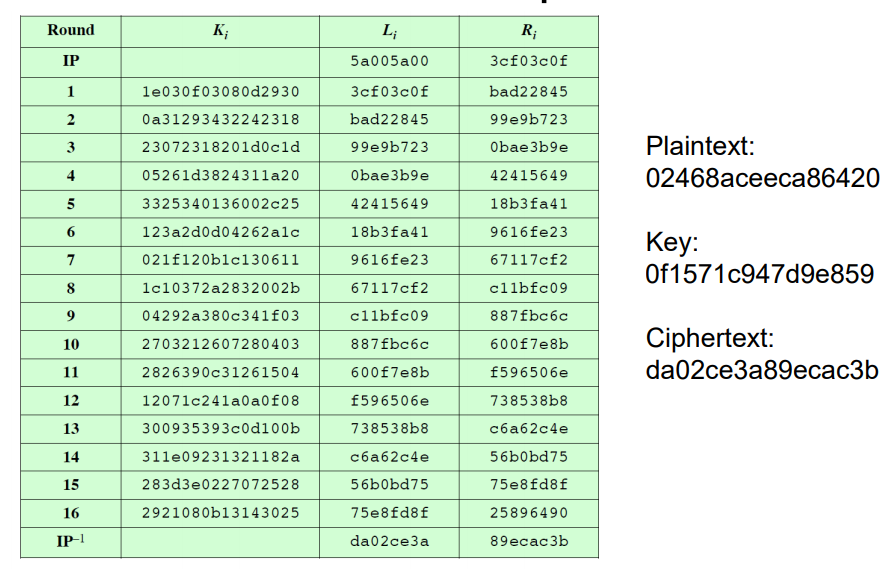
다음은 이 single round로 나온 값을 swap, inverse permutation 하는 과정이다

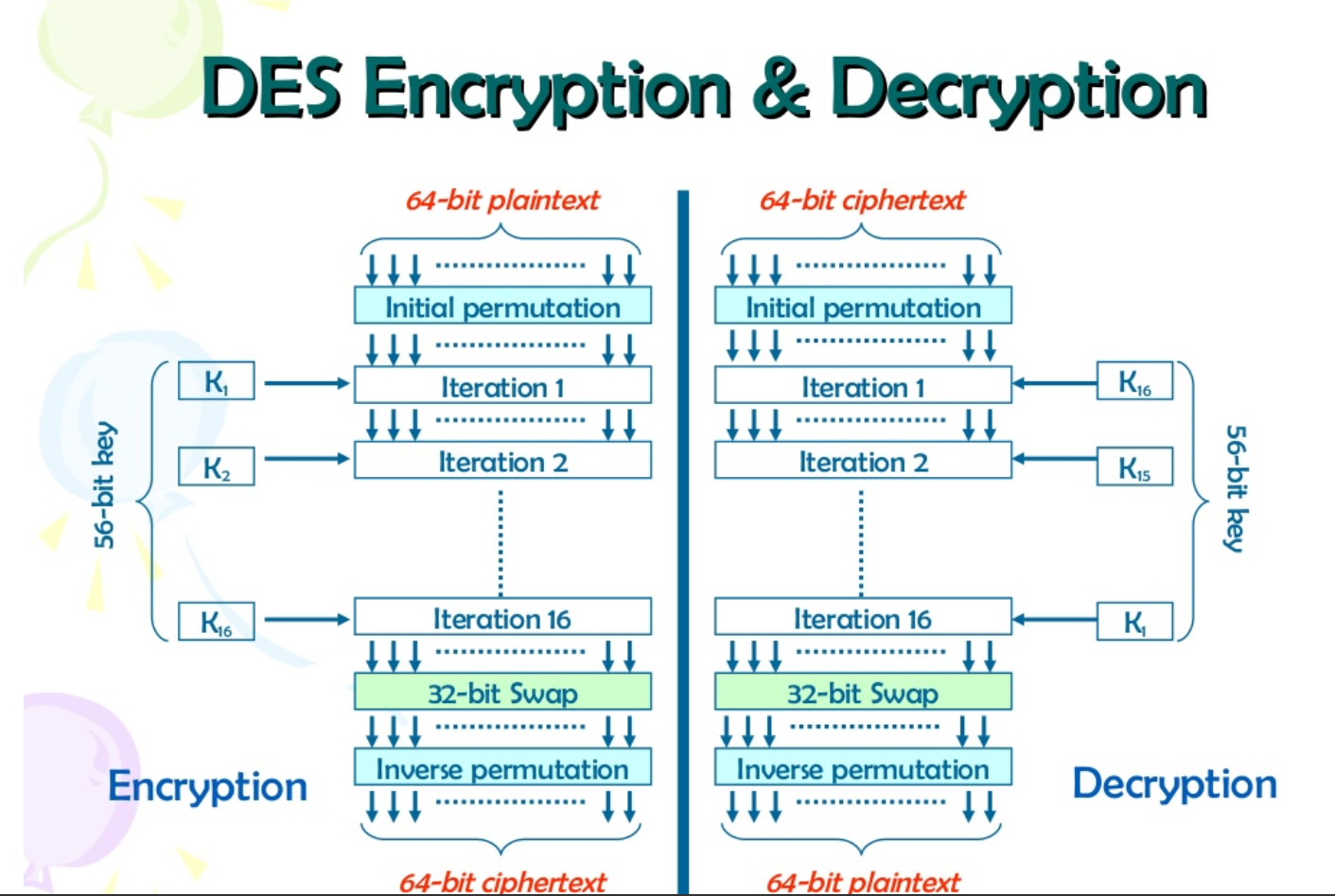


절반 씩 swap 후 inverse permutation을 처리하면 ‘bbdcab9836553604’의 값이 최종 출력된다

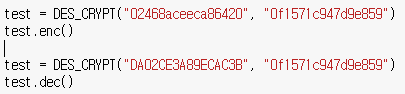
1. **Show that DES decryption is, in fact, the inverse of DES encryption**

강의 때 사용하신 PPT에 첨부된 DES의 예제 테이블 및 코드를 통해 DES decryption이 des encryption의 역임을 보인다



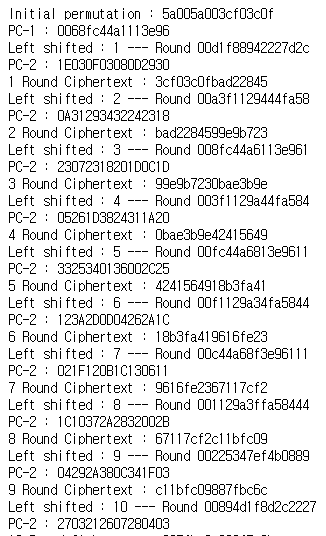


첫 번째는 예제 테이블이고, 두 번째는 DES의 encryption과 decryption의 과정 모식도이다. 주목할 점은 ciphertext를 encryption알고리즘과 동일하게 수행하는 대신, subkey만 각 encryption에서 나온 subkey의 역순으로 넣어주면 되는 점이다



다음과 같이 예제 테이블의 plaintext, key를 넣고 test.enc()를 통해 encrypt 하고, encrypt의 결과값인 ciphertext와 key를 넣고 test.dec()를 통해 decrypt 하였다

결과는 다음과 같다



initial permutation -> key modify(PC-1 -> left shifted -> PC-2) -> 각 round ciphertext -> swap -> encrypted ciphertext 의 과정이 나타난다

위의 예시로 사용한 표와 동일함을 볼 수 있다

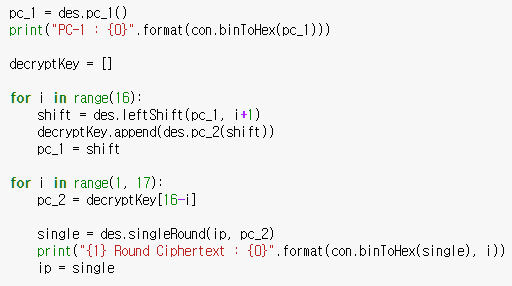
Ki를 표에서는 6bit씩 끊어 2bit씩 hex code로 바꾸는데, 이를 동일하게 구현하여 각 PC-2값 또한 일치함을 알 수 있다

다음은 decryption이다.

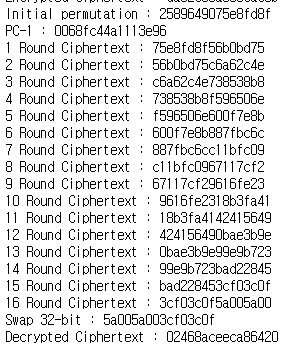
encryption과 과정이 동일하나, key modify를 하는 과정에서 먼저

PC-1 -> left shifted -> PC-2 를 한 후, 각 round에 맞게 left shift와 PC-2를 한 값들을

배열에 저장하였다



decryptKey에 저장된 각 키 값을 각각의 round 수행 시 역으로 (K16부터 차례로 round 1부터 K1까지 round 16 수행) XOR 연산을 수행하였다



decrypt된 결과는 원래의 plaintext이고, 각 round의 ciphertext도 encrypt 단계의 역이다

1. **결론**

single round 구현 및 전체 DES의 encryption, decryption을 수행하는 code를 작성하고, 이를 통해 각 과정에서 출력되는 subkey 및 ciphertext를 눈으로 확인하니 훨씬 더 해당 암호 알고리즘이 이해가 잘 되었다.

평소에도 AES나 DES는 암호화 방식으로 알고는 있었으나 실제로 암호화 되는 과정을 보면서 이를 해독하는 프로그램을 만드시는 분들이 더욱 대단하게 느껴졌다. 한 편으로, 코드 작성을 하고 나서는 어려웠던 수업 내용의 이해가 매우 잘 되어 기쁘다

긴 연휴기간에 작성한 코드와 과제이긴 하나, 매우 재미있었고, 결과 ciphertext가 출력되었을 때는 굉장히 뿌듯했다