# PBKDF1, 2
# 분석

정보보안암호수학과 20182209 김수빈

# PBKDF1 | Password-Based Key Derivation Function

- MD2, MD5, SHA-1인 해시함수 적용

- 파생된 키 길이는 제한

MD2, MD5 – 16byte
SHA-1 – 20byte

**함수 : DK = PBKDF1 (Password, Salt, iteration, dkLen)**

Options : Hash

Input :
Password
Salt (랜덤)
Iteration (반복횟수)
dkLen (키 길이)

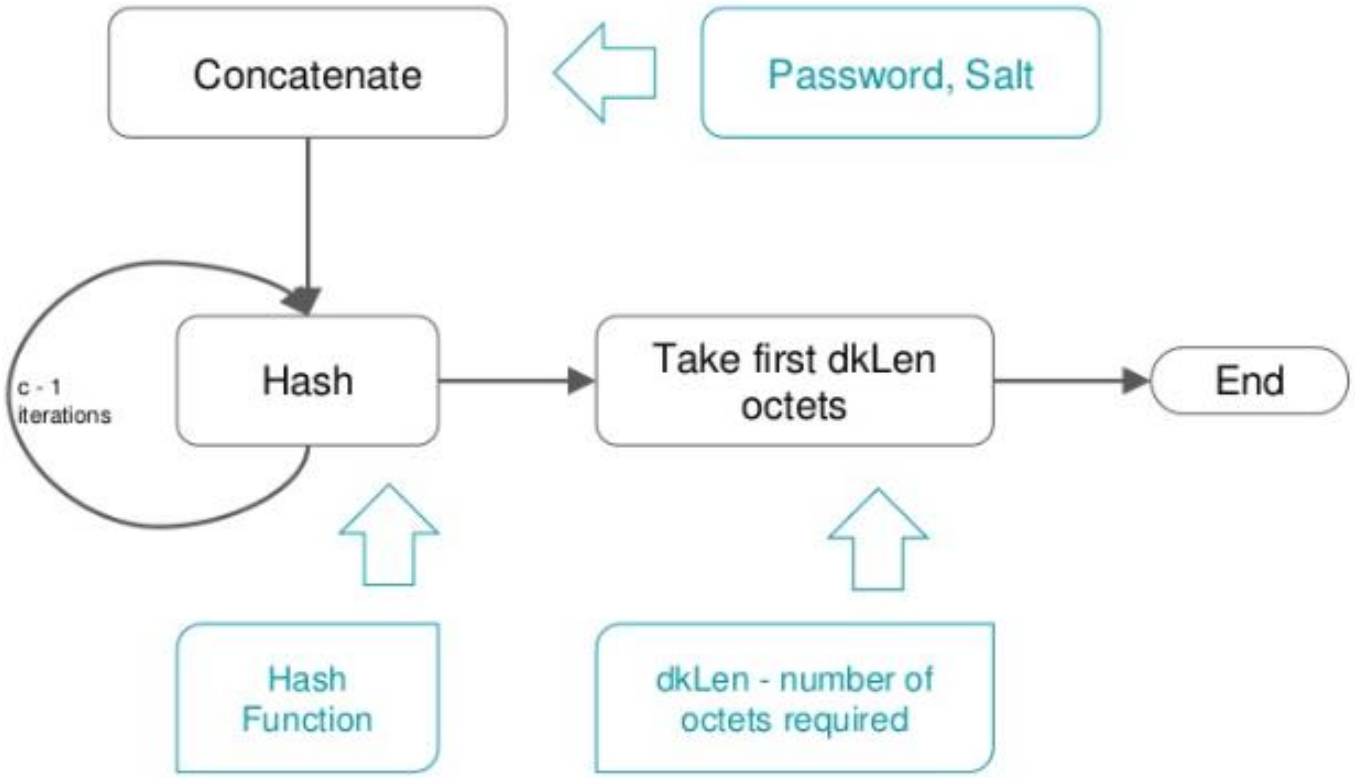
Output : DK (derived key)

Salt와 iteration은 공격자의 사전공격을 어렵게 하는 중요한 요소

[출처] PKCS #5 : Password-Based Cryptography Specification version 2.0

# PBKDF1 | Password-Based Key Derivation Function

Iteration = c




```
if (dkLen > 16)
   return print("derived key too long")

for ( i = 1 to c ){
    T_0 = P || S
    T_i = Hash(T_(i-1))
}
return DK = T_c<0 ··· dkLen-1>
```

# PBKDF2 | Password-Based Key Derivation Function

- 의사 난수 기능을 적용하여 키 파생

- 파생된 키 길이는 제한 없음
  그러나 파생된 키의 최대 유효 검색 공간은 기본 의사 난수 함수의 구조에 의해 제한 될 수 있다.

**함수 : DK = PBKDF2 (Password, Salt, iteration, dkLen)**

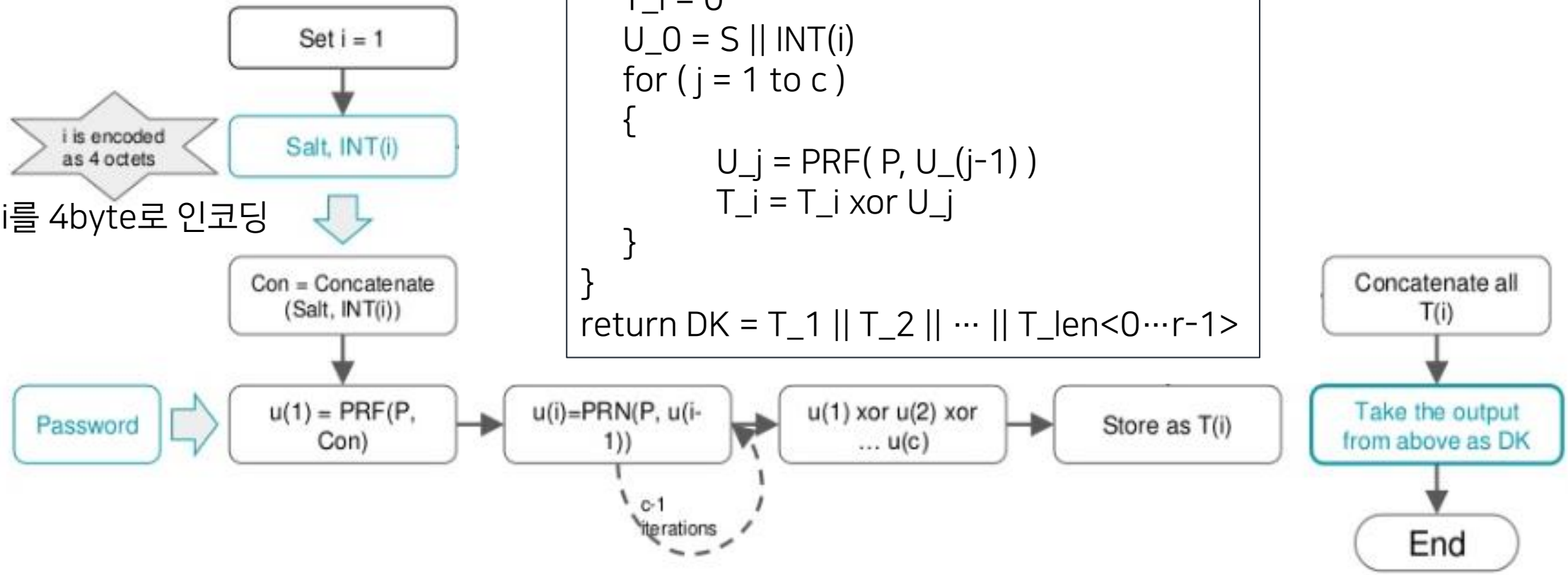
Options : PRF (의사 난수 함수, hlen : 의사 난수 함수 출력 길이)

Input :
Password
Salt (랜덤)
Iteration (반복횟수)
dkLen (키 길이, 최대길이 : (2^32 – 1) * hlen )

Output : DK (derived key)

# PBKDF2 | Password-Based Key Derivation Function

```
If (dkLen > (2^32 – 1) * hLen)
   return print(“derived key too long”)
len = dkLen / hLen
r = dkLen – (len – 1) * hLen
for (l = 1 to dkLen)
{
    T_i = 0
    U_0 = S || INT(i)
    for ( j = 1 to c )
    {
            U_j = PRF( P, U_(j-1) )
            T_i = T_i xor U_j
    }
}
return DK = T_1 || T_2 || ··· || T_len<0···r-1>
```




INT(i) : i를 4byte로 인코딩