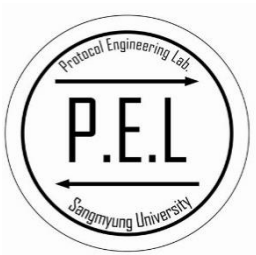


네트워크 보안 에센셜

2장 랜덤 넘버와 스트림

Sein Myung (sein@pel.smuc.ac.kr)

Protocol **E**ngineering **L**ab., **Sangmyung** University

목 차

- 랜덤넘버 (Random Number)
- 진성 랜덤 넘버 (True Random Number)
- 의사 랜덤 넘버 (Pseudo Random Number)
- 스트림 암호 (Stream Cipher)
- RC4 (Rivest Chipher 4)
- 암호블록의 운용 (Operation of Block Cipher)

랜덤넘버

- **랜덤 넘버 (Random Number)**
  - **무작위적으로 추출되어 예측 불가능한 숫자열**
  - **대칭 암호 스트림 키 생성 에 사용**
  - **무작위성과 예측 불가능성을 가져야함**

랜덤넘버

- 랜덤 넘버 (Random Number)
  - 무작위성(Randomness)
    - 수열이 균등분포하고 독립성이 있음
      - 균등분포 (Uniform Distribution)
        - 1과 0이 비트열에 나타나는 빈도가 균일함을 의미
      - 독립성 (Independence)
        - 수열에서 추출한 부분 어떤 부분 수열도 다른 수열로부터 추론될 수(연관성이) 없어야 함
    - 통계적으로 랜덤 한다고 여겨지는 수열을 암호화 알고리즘 설계에 이용

      문제가 너무 어렵거나, 정확한 답을 내는데 시간이 오래 걸릴 경우 무작위 방법으로 원하는 수준의 해답을 찾아냄

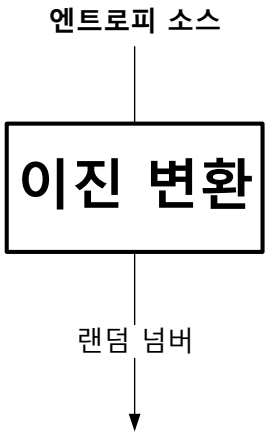
      무작위 방법 : 소수 생성시 너무 큰 소수(10^{150})는 제곱근(10^{75}) 이하의 홀수로 나누는 방법을 이용

랜덤넘버

- 랜덤 넘버 (Random Number)
  - 예측불가능성 (Unpredictability)
    - 수열의 일부를 보고 그 다음에 이어지는 수를 예측할 수는 없어야 함
    - 진성랜덤넘버일 경우에는 수열에 나타나는 모든 수가 통계적으로 수열 안의 다른 수와 독립이지만 진성 랜덤넘버는 컴퓨터로 구현이 불가능

진성 랜덤 넘버

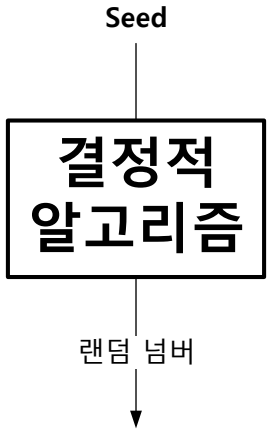
- 진성 랜덤 넘버 (True Random Number)
  - 입력 값이 실제로 랜덤한 정보
  - 컴퓨터에서 물리적으로 얻을 수 있는 랜덤정보
    - 엔트로피 소스(Entropy Source) : 키 입력 타이밍 패턴, 디스크 전기작용, 마우스 움직임, 시스템 클록의 순간 값 등
  - TRNG (True Random Number Generator)
    - 알고리즘의 입력으로 엔트로피 소스들의 조합을 사용하여 랜덤한 바이너리를 출력함



의사 랜덤 넘버

- 의사 랜덤 넘버 (Pseudo Random Number)
  - 입력 값이 고정된 특정 값, 이를 종자(Seed)라 함
  - Seed 값에 따라 난수가 결정됨

    결정적 알고리즘에 의해 Seed가 같으면 같은 난수를 출력
  - Seed를 모르면 난수를 예측할 수 없음
  - Seed를 예측할 수 없어야 함(고유 값)
  - 철학적으로 완전한 난수는 아니지만 무작위성 테스트에 걸리지 않게 설계



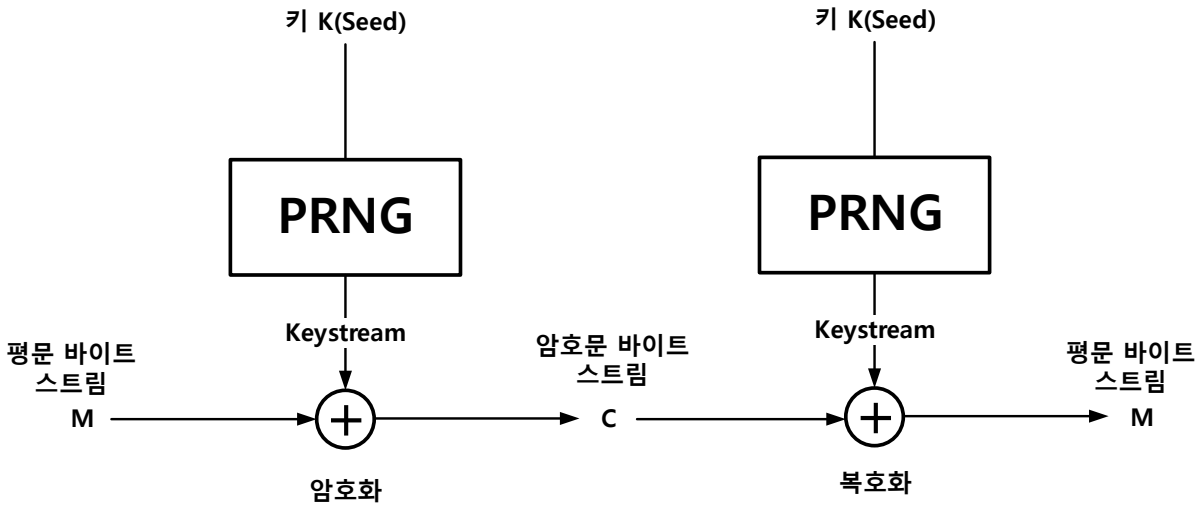
  - PRNG (Pseudo Random Number Generator)
    - 무한 비트열을 생성하기 위해 사용되는 알고리즘
    - 대칭 스트림 암호에 이용
  - PRF (Pseudo Random Number Function)
    - 고정된 길이의 의사 랜덤 비트열을 생성에 사용되는 함수

의사 랜덤 넘버

- 의사 랜덤 넘버 (Pseudo Random Number)
  - 알고리즘의 설계
    - 특정 목적 알고리즘
      - 의사 랜덤 비트 스트림을 생성하기위해 특정한 목적만을 위한 알고리즘
    - 기존 암호 알고리즘을 이용
      - 암호학적 알고리즘은 입력을 난수로 만드는 효과가 있음, 설계 하려는 의사 난수 알고리즘에 필요한 사항임
        대칭블록 암호가 생성하는 암호문이 특정 패턴을 가지면 해독의 빌미가 됨
        따라서 암호학적 알고리즘은 TRNG의 핵심 역할을 할 수 있음
    - 어떤 방식을 사용해도 암호학적으로 강한 PRNG를 만들수 있음
      - 특정 목적 알고리즘은 운영체제에서 일반적 용도로 제공할 수도 있다, 이를 사용해 암호화를 하거나 인증을 하고 있다면, 그 코드를 그대로 이용해서 PRNG를 만들어 낼 수 있음

스트림 암호

- 스트림 암호 (Stream Chiper)
  - 입력되는 요소를 연속적으로 처리하여 한번에 한 요소씩 출력 함
  - 적합한 암호구조가 있지만 블록암호, 스트림 암호 둘다 모든 응용프로그램에서 사용가능
  - 스트림 암호 구조
    - 설계시 비트, 바이트 보다 큰 자료를 처리할 수 있도록 만듦
      - 보통 평문 1바이트 단위로 암호화
    - PRNG의 출력인 키 스트림(Keystream)과 평문을 XOR연산하여 암호문을 만들어냄



스트림 암호

- 스트림 암호 (Stream Chiper)
  - 스트림 암호 설계시 주의할점
    - 키스트림 주기가 커야함
      - PRNG는 결국에는 반복적으로 나타나는 비트 스트림을 만들어 냄
      - 이 반복주기가 길면 길수록 해독이 어려우므로 충분히 길어야 함
    - 키스트림은 가능하면 진성랜덤 스트림의 특성에 근사해야 함
      - 0과 1의 개수가 거의 동일해야 함 (균등분포)
      - 바이트 단위라면 256가지의 가능한 모든 값이 (거의)균일한 횟수로 출력 되어야 함
    - 입력 키의 길이가 충분히 길어야함
      - 입력되는 키의 값이 전수공격에 견디도록 충분히 길어야 함 (최소한 128bit)

스트림 암호

- **스트림 암호 (Stream Chiper)**
  - **스트림 암호의 장점**

    메시지의 길이가 블록의 정수배가 되도록 패딩할 필요가 없고, 실시간으로 사용할 수 있음

    동일 길이의 키를 사용하는 블록암호 만큼의 보안성을 유지할 수 있음

    블록암호보다 속도가 빠르고 적은 양의 코드를 사용

  - **스트림 암호의 단점**

    키의 재사용에 취약함

    두개의 평문을 동일한 키스트림으로 암호화 할 시 암호해독이 단순해짐

    두개의 암호문을 XOR연산시 두 평문을 XOR 한 값이됨

    ^ : XOR연산 A,B : 평문

    K : 키 C1,C2 : 암호문

    A^K=C1, B^K=C2

    C1^C2 = (A^K) ^ (B^K) = A^B ^ K^K = A^B

    평문이 문자열, 신용카드번호 등의 특성이 알려진 바이트 스트림 이면 암호해독을 성공적으로 할 수 있음

RC4

- **RC4 알고리즘의 특징(Rivest Cipher 4)**
  - **1987년 RSA Security 에서 Ron Rivest가 설계한 스트림 암호**
  - **바이트 단위로 작동됨, 다양한 크기의 키(1~256byte)를 사용**

    1994년 9월 익명의 제보자가 인터넷의 사이버 펑크 익명 리메일러 목록에 RC4 알고리즘을 올림
  - **알고리즘은 랜덤 치환에 기초해서 만들어짐, 스트림의 주기는 10^{100}을 상회**
  - **하나의 바이트를 출력하기 위해 8에서 16번의 연산이 필요**
    - **복잡도를 의미하며, 이 암호는 소프트웨어 상에서 매우 빠르게 동작함**

RC4

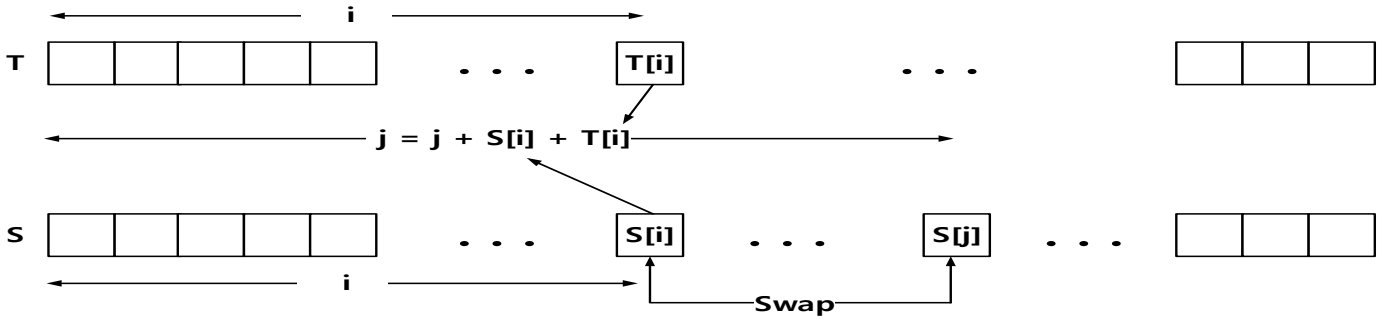
- **RC4 알고리즘의 동작**

```
Initialization

For( i=0; i<256; i++ ) {
    S[i] = i;                          // S는 256개의 byte벡터 배열, 0~255의 값을 가짐
    T[i] = K[ i mod keylen ];          // 256byte의 T에 키 값을 스케줄
}
```

```
Initial Permutation of S

j=0;
For( i=0; i<256; i++ ) {
    j = ( j + S[i] + T[i] ) mod 256;   // 이전 j값+i벡터 값+스케줄된 i번째 키 값 -> 키값으로 치환을 복잡하게
    Swap(S[i], S[j]);                  // 단순 값 교환
}
```



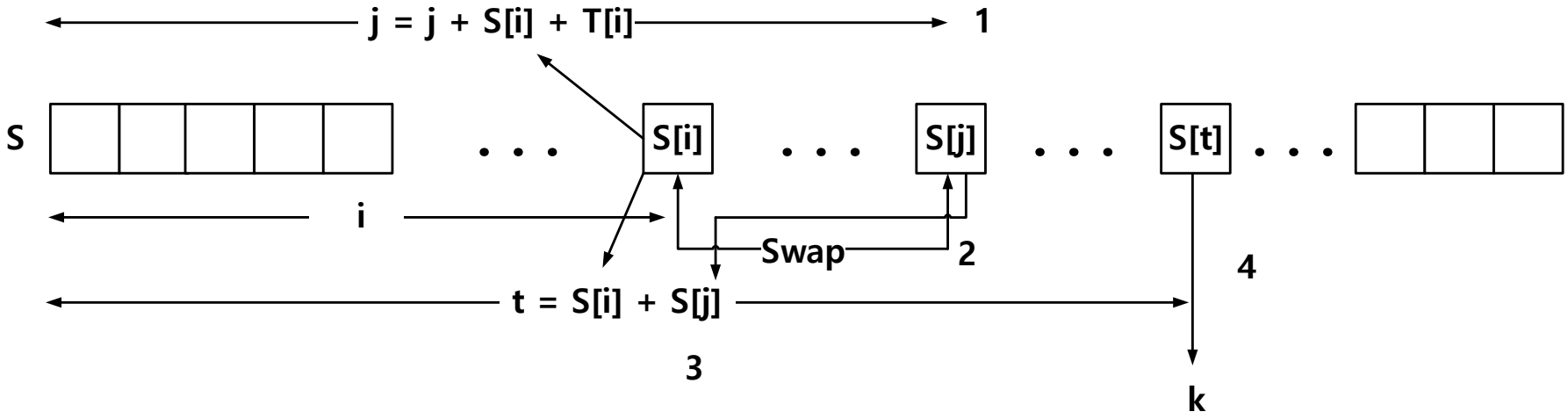
RC4

- **RC4 알고리즘의 동작**

```
Stream Generation

i,j=0;
while(true) {                          // 스트림 반복
    i = ( i+1 ) mod 256;               // i 는 0~255 를 반복
    j = ( j+S[i] ) mod 256;            // 이전 j값 + i번째 S값 -> j값으로 무작위 선택
    Swap( S[i], S[j] );                // 단순 값 교환으로 스트림의 규칙성을 깸

    t = ( S[i]+S[j] ) mod 256;
    k = S[t];                          // k는 스트림 출력
}
```



RC4

- RC4 알고리즘의 특징(Rivest Cipher 4)
  - RC4 알고리즘의 강도
    - 여러 논문에서 RC를 공격하는 방법에 대해 분석중
    - 여러 공격중 어느 것도 적당한 길이의 키(128bit)에 대해 RC4를 실제로 공격 할 수 없음
    - 키값이(키 생성에) 문제가 없다면 RC4 알고리즘은 현재까지 안전하다고 알려짐

암호 블록의 운용

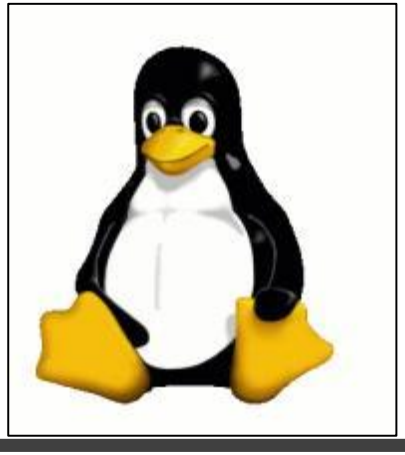
- 블록 암호의 운용
  - 알고리즘마다 입력되는 블록의 크기가 다르므로 구체화가 필요
  - NIST(National Institute of Standards and Technology)에서 5가지 운용 모드(Mode of operation)를 정의
  - 5가지 모드는 블록암호가 사용되는 모든 암호 응용에 사용할 수 있도록 만들어짐
  - 전자 코드북 모드, 암호 블록 체인 모드, 암호 피드백 모드, 카운터 모드 등을 설명

암호 블록의 운용

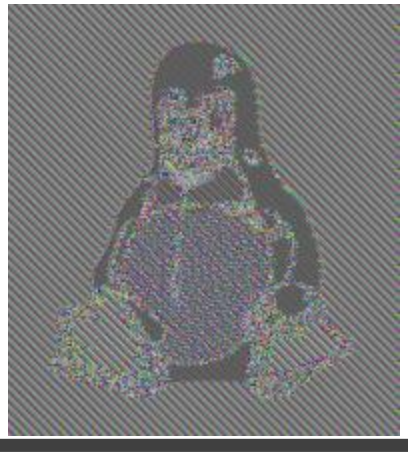
- 전자 코드북 모드 (ECB:Electronic Codebook Mode)
  - b비트 크기의 블록으로 다뤄지고 각 블록을 동일한 키로 암호화 함
  - 하나의 키가 주어지면 각각의 b비트 평문 블록에 대해 유일하게 하나의 암호 블록이 대응되기 때문에 코드북 이란 이름이 붙음
  - 가장 간단한 운용모드

  - ECB 모드의 취약점
    - 동일 평문 블록에 대해 같은 암호문 출력 (평문 반복이 암호문의 반복)

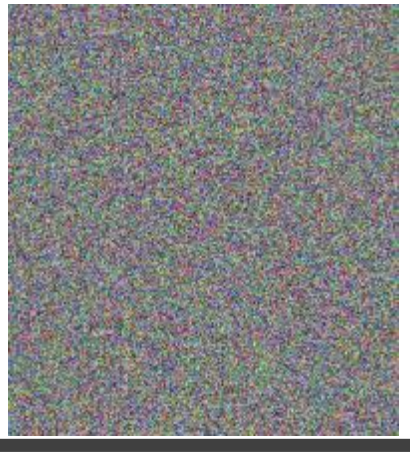
평문



ECB 모드



다른 모드



암호 블록의 운용

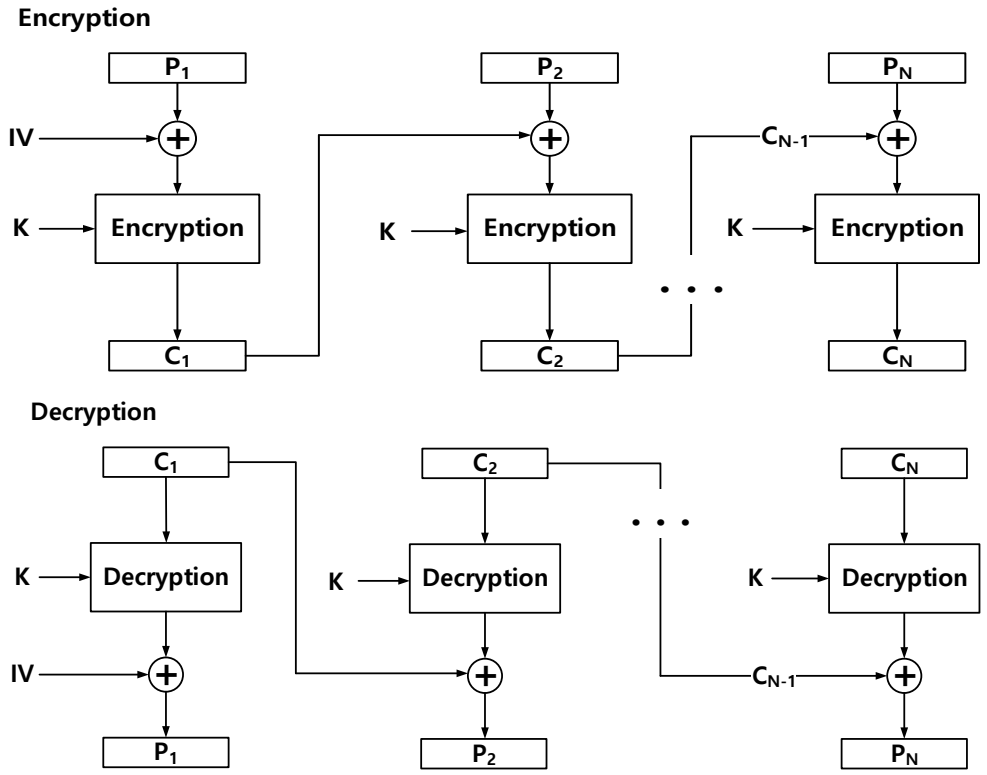
- 암호 블록 체인 모드 (CBC:Cipher Block Chaining Mode)
  - 1976 IBM에 의해 개발
  - 각 블록에서 동일한 키를 사용
  - 현재의 평문과 직전의 암호문을 XOR한 후 입력으로 사용
  - IV(Initialization Vector)로 초기 암호문을 생성
  - CBC의 장점
    - 평문의 반복이 숨겨짐
  - CBC의 단점
    - 송신자와 수신자는 IV를 알고 있고, 보안성을 강화 하기위해 IV또한 키처럼 보호 해야 함 (IV전송시 ECB암호를 이용해서 보낼 수 있음)
    - 오류가 확산됨

암호 블록의 운용

- **암호 블록 체인 모드 (CBC:Cipher Block Chaining Mode)**
  - **CBC의 동작과 오류 확산**

    암호화시 평문 하나에 문제가 생기면 이후 전체에 영향

    복호화시 암호문 하나의 영향은 현재 평문과 다음 평문에 영향



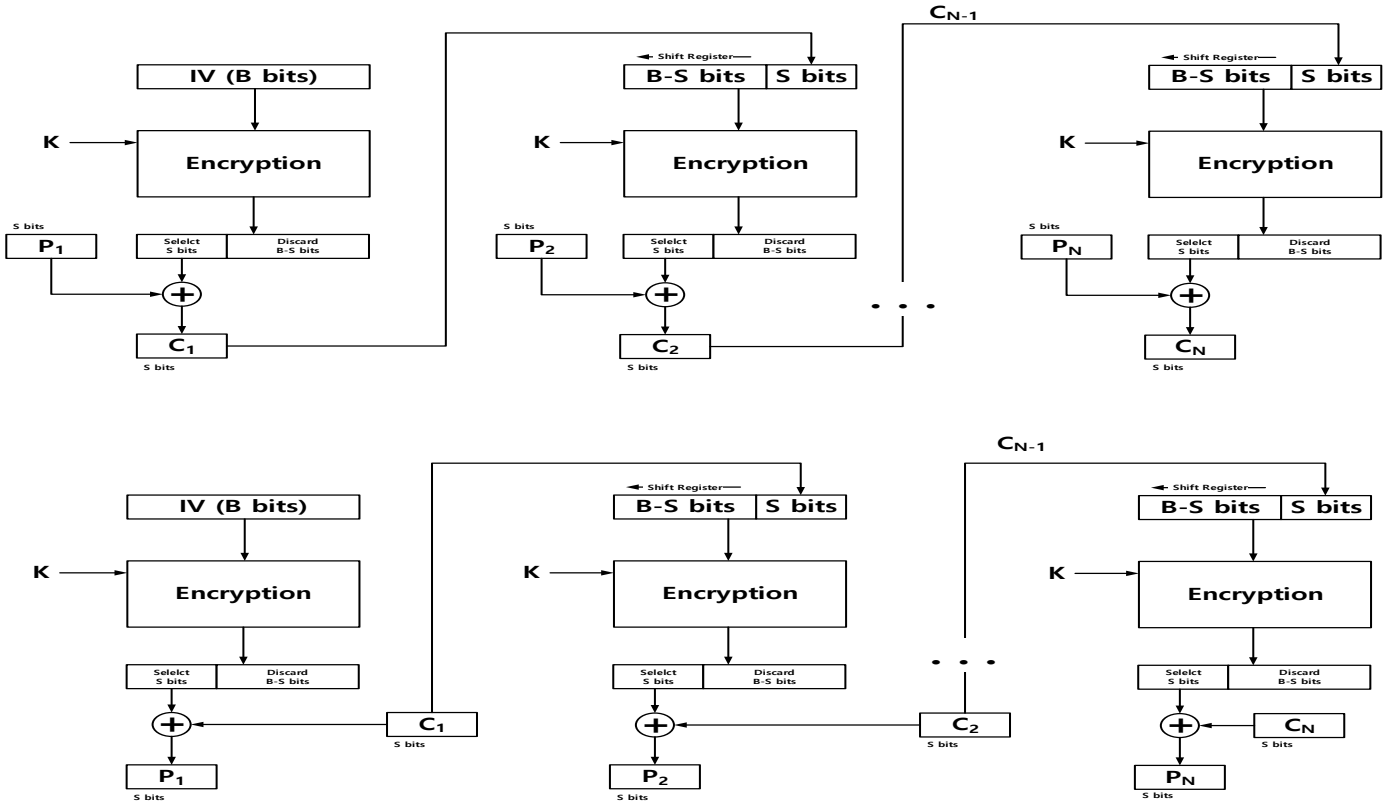
암호 블록의 운용

- 암호 피드백 모드 (CFB:Cipher Feedback Mode)
  - 이전출력에 의해서 다음 입력이 변화
  - 어떤 블록 암호도 스트림 암호로 바꿀 수 있음

    메시지의 길이가 블록의 정수배가 되도록 패딩하지 않음
  - 한 문자를 전송하는 경우 암호화되는 순간 바로 전송 가능 (보통 8byte단위)
  - 암호 함수의 입력은 초기값(IV) 를 갖는 시프트 레지스터, 각 단계마다 피드백을 받음
  - CFB의 장점
    - 블록 배수 패딩이 불필요
    - 복호화시 선택 접근 가능
  - CFB의 단점
    - 오류가 확산됨

암호 블록의 운용

- 암호 피드백 모드 (CFB:Cipher Feedback Mode)
  - CFB의 동작과 오류 확산
    - CBC와 같은 오류 확산

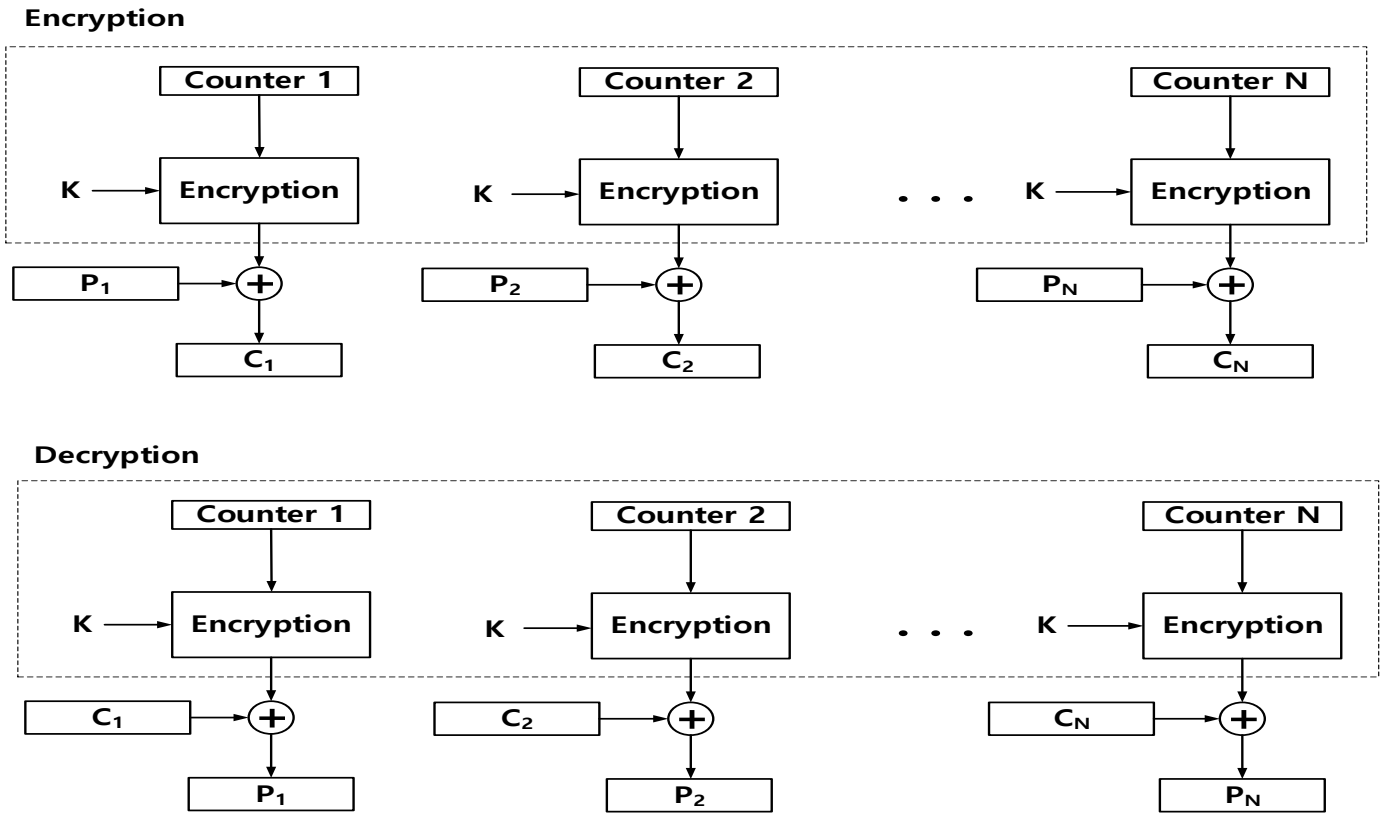


암호 블록의 운용

- **카운터 모드 (CTR:Counter Mode)**
  - **ATM(Asynchronous Transfer Mode)네트워크 보안과 IPSec(IP Security)에 응용되면서 최근에 관심이 늘어났지만 이 모드는 오래 전부터 제안되어 왔음**
  - **평문과 동일한 크기의 카운터 사용**
  - **카운터 값이 암호화할 블록별로 다름**
    - **특정b비트의 초기 카운터값을 정하고, 1증가후 2^b 로 모듈로 하여 단계별 카운트 생성**
  - **카운터를 암호화해서 평문과 XOR연산을 해 암호블록을 만들고, 복호화 시에 동일한 카운터 값을 이용함**

암호 블록의 운용

- **CTR모드의 동작**
  - **사전처리, 병렬처리, 랜덤접근이 가능**
  - **체인은 이용하지 않음**



암호 블록의 운용

- 암호블록 모드 정리

종류	병렬처리	랜덤 접근	오류 확산	평문패턴 노출	초기화 벡터
ECB	암호화, 복호화	암호화, 복호화	대응 블록	Y	불필요
CBC	복호화	복호화	암호화시 전체	X	필요
CFB	복호화	복호화	암호화시 전체	X	필요
CTR	암호화 복호화	암호화, 복호화	대응 블록	X	카운터