5장 : 스트림 암호

정보보호이론

Spring 2015

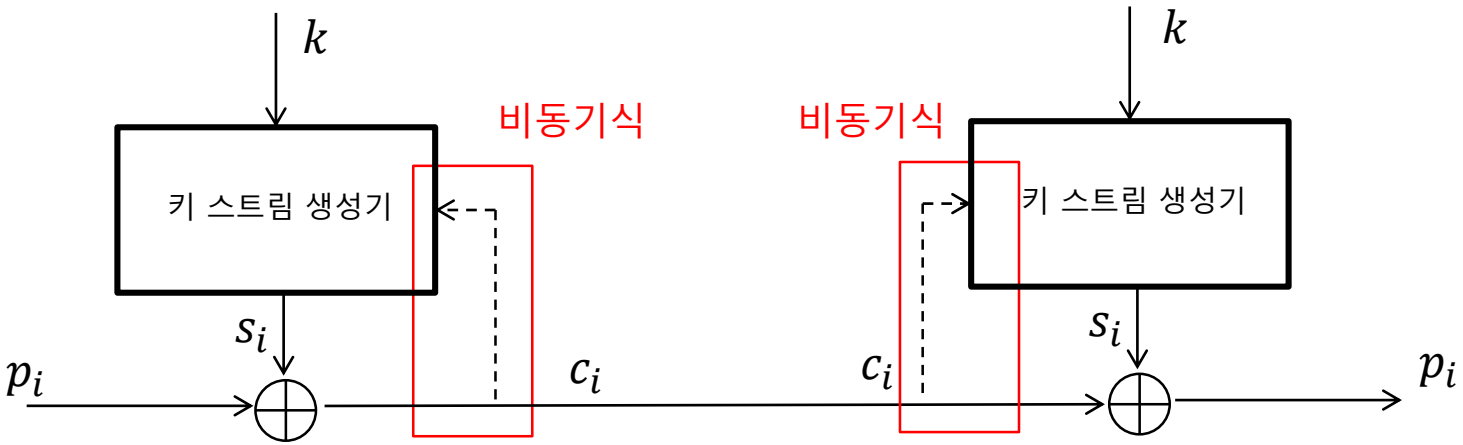
고려대학교
KOREA UNIVERSITY

5.1 스트림 암호와 블록 암호

- 블록 암호 vs 스트림 암호
  - 블록 단위로 암호화 vs 스트림 암호 :비트 단위로 암호화
- 스트림 암호
  - 패딩과 운영모드에 대한 개념 X
  - 빠른 암/복호화가 가능
  - 효율적인 ***"PRESENT"*** 나 ***"HIGHT"*** 와 같은 블록 암호가 존재

5.2 키 스트림 생성

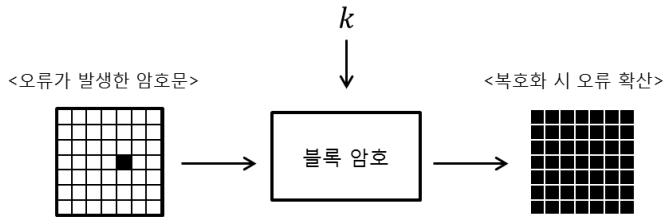
- **키 스트림**(Key Stream)으로부터 출력되는 1 비트가 평문 1비트와 XOR되어 암호문 1 비트를 생성
  - 스트림 암호의 안전성은 키 스트림이 어떻게 생성되는지에 따라 전적으로 결정
- 스트림 암호의 암호화 및 복호화 과정
  - 암호화 : $E_{s_i}(p_i) = p_i \oplus s_i = c_i$
  - 복호화 : $D_{s_i}(c_i) = c_i \oplus s_i = (p_i \oplus s_i) \oplus s_i = p_i$



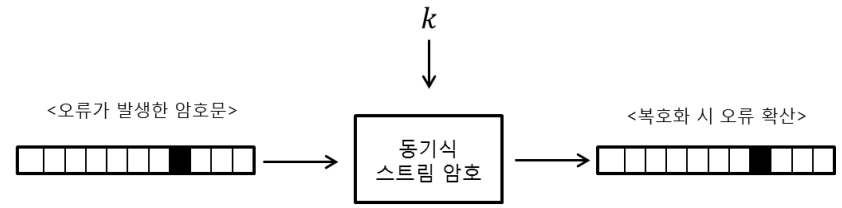
5.2 키 스트림 생성

- **동기식 스트림 암호**(Synchronous Stream Cipher)
  - 키 스트림의 생성이 키에만 의존 : OFB, CRT, OTP
- **비동기식 스트림 암호**(Asynchronous Stream Cipher)
  - CFB

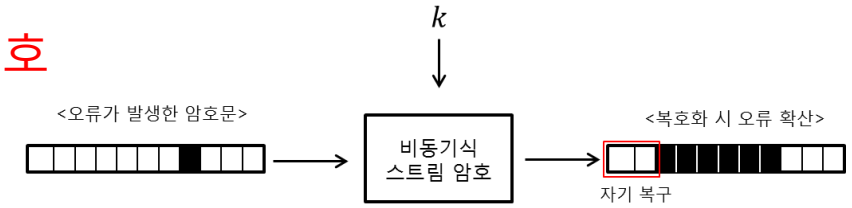
■ 스트림 암호의 오류 확산



동기식(Self-Synchronizing) 스트림 암호



비동기식(Self-Synchronizing) 스트림 암호



5.3 난수 생성

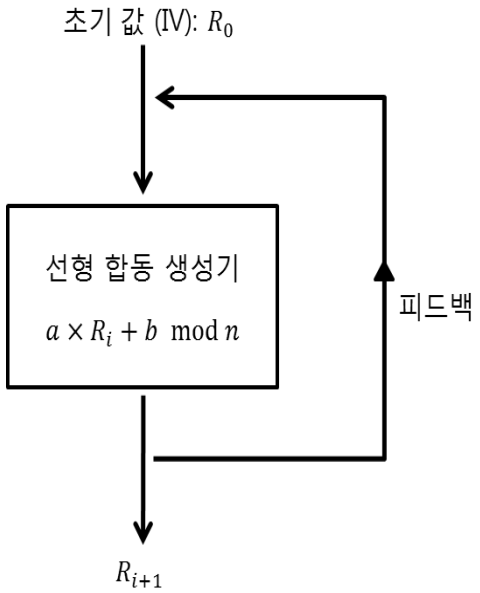
- 난수 생성기(True Random Number Generator, TRNG)
  - 비예측성 & 비결정성
  - 전자 저항에서 생성되는 열 잡음의 표본을 추출하거나, 방사선 관측기로부터 나오는 출력 값을 반복해서 사용 → 환경적 제약
- 의사 난수 생성기(PRNG: Pseudo Random Number Generator)
  - 초기값을 입력받아 계산되는 의사 난수열을 출력하며, 같은 입력 값에 대하여 같은 출력 값을 생성하는 결정적(Deterministic) 알고리즘 → 통계적 검증

	생성시간	결정성	주기성
TRNG	비효율적	비결정적	비주기적
PRNG	효율적	결정적	주기적

5.3 난수 생성

- PRNG 예) 선형 합동 생성기(Linear Congruential Generator)
  - $R_{i+1} \equiv (a \times R_i + b) \bmod n,\ 0 \le R_0 < n$: 초기 값
  - a, b를 비밀키로 공유, 법 n은 공개
  - $a = 3, b = 2, n = 17$

i	R_i	$a \times R_i + b$	R_{i+1}	i	R_i	$a \times R_i + b$	R_{i+1}
0	8	$3 \times 8 + 2$	9	10	3	$3 \times 3 + 2$	11
1	9	$3 \times 9 + 2$	12	11	11	$3 \times 11 + 2$	1
2	12	$3 \times 12 + 2$	4	12	1	$3 \times 1 + 2$	5
3	4	$3 \times 4 + 2$	14	13	5	$3 \times 5 + 2$	0
4	14	$3 \times 14 + 2$	10	14	0	$3 \times 0 + 2$	2
5	10	$3 \times 10 + 2$	15	15	2	$3 \times 2 + 2$	8
6	15	$3 \times 15 + 2$	13	16	8	$3 \times 8 + 2$	9
7	13	$3 \times 13 + 2$	7	17	9	$3 \times 9 + 2$	12
8	7	$3 \times 7 + 2$	6	18	12	$3 \times 12 + 2$	4
9	6	$3 \times 6 + 2$	3	…			



5.3 난수 생성

- PRNG 예) 선형 합동 생성기(Linear Congruential Generator)
  - 최대 주기를 갖기 위해서는 다음과 같은 조건을 고려

1. 모듈로 n의 크기에 따라 최대 주기가 결정되기 때문에, 효율성과 안전성을 고려하여 n을 선택(보통 컴퓨터가 한번에 데이터를 처리할 수 있는 워드의 크기만큼을 n으로 선택함)
2. n의 크기와 같은 주기를 갖기 위해서는 a를 n과 서로소인 수로 선택
3. b의 값은 주기에 영향을 미치진 않지만, 계산의 효율성을 위하여 일반적으로 0으로 사용

5.3 난수 생성

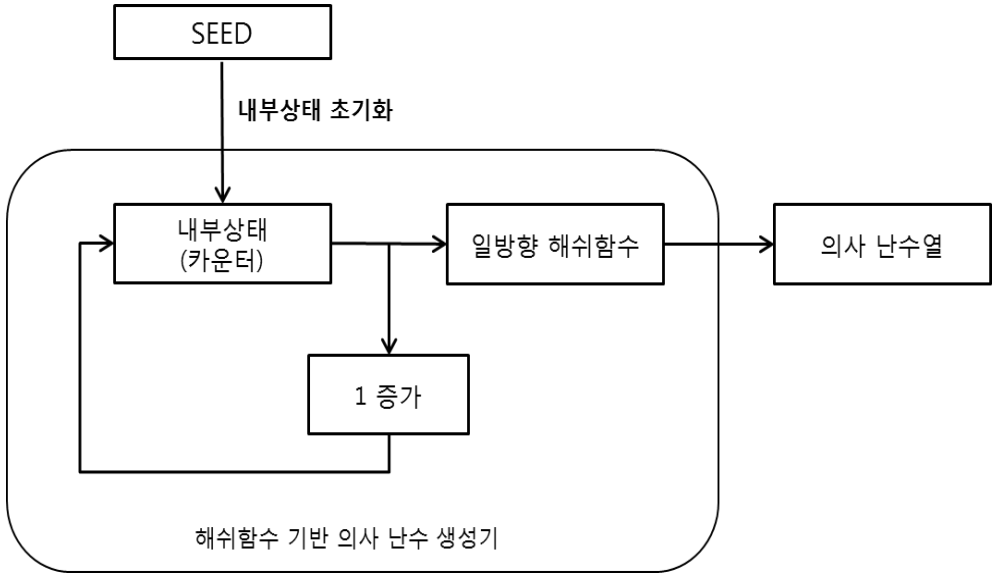
- 암호학적으로 안전한 의사 난수 생성기 (Cryptographically Secure Pseudo Random Number Generator, CSPRNG))
  - 예측불가능성
    1. CSPRNG가 생성하는 키 스트림 $s_{i+1}, s_{i+2}, \cdots, s_{i+k}$가 주어졌을 때, 다음 비트 s_{i+k+1}을 50% 이상의 확률로 예측하는 것이 계산적으로 불가능
    2. 주어진 키 스트림의 이전 비트인 $s_i, s_{i-1}, \cdots$ 중 한 비트를 50% 이상의 확률로 예측하는 것이 계산적으로 불가능
    - 컴퓨터 과학이나 공학분야에서는 예측불가능성 불필요
  - 난수 생성방법
    1. 일방향 해쉬 함수 이용
    2. 블록 암호 이용
    3. 수학적 난제에 기반한 생성 방법

5.3 난수 생성

- 일방향 해쉬 함수를 사용한 의사 난수 생성기
  - 일방향함수 $f(x) = y$
    - 주어진 x에 대하여 함수 $f(\cdot)$를 이용하여 y를 계산하는 것은 쉽지만 반대로 y가 주어졌을 때 함수 $f(\cdot)$를 이용하여 $x = f^{-1}(y)$를 계산하는 것은 어려운 함수
    - p, q: prime, $\quad g(p, q) = p \times q$
  - 해쉬함수 $f(x) = y$
    - 상이한 입력값에 고정된 길이의 출력값
    - 압축함수

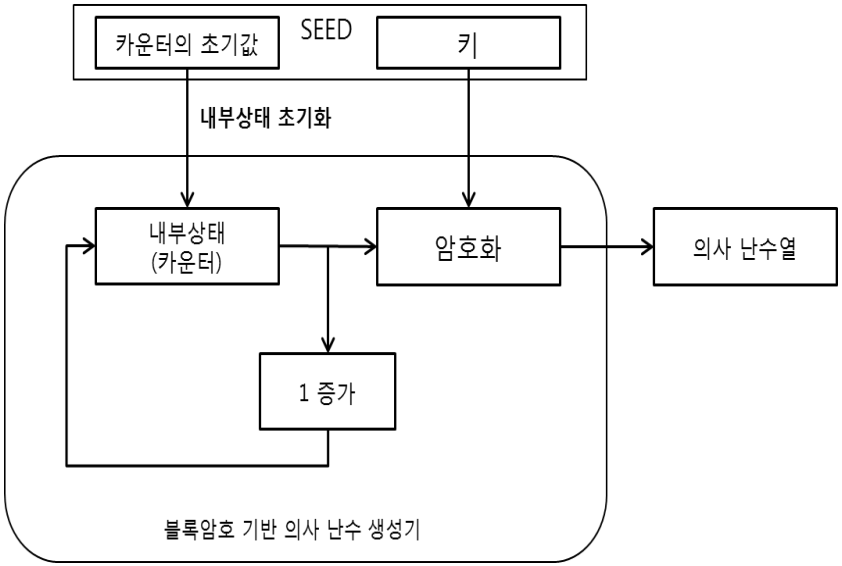
5.3 난수 생성

- 일방향 해쉬 함수를 사용한 의사 난수 생성기
  - 해쉬 함수의 일방향성에 의하여 해당 난수에 해당하는 카운터 값을 예상할 수 없음
  - 카운터 값을 알 수 없다면, 앞으로 생성될 난수열이나 이전에 생성된 난수열을 예측할 수 없게 되므로 앞에서 말한 CSPRNG를 위한 2가지 조건을 만족



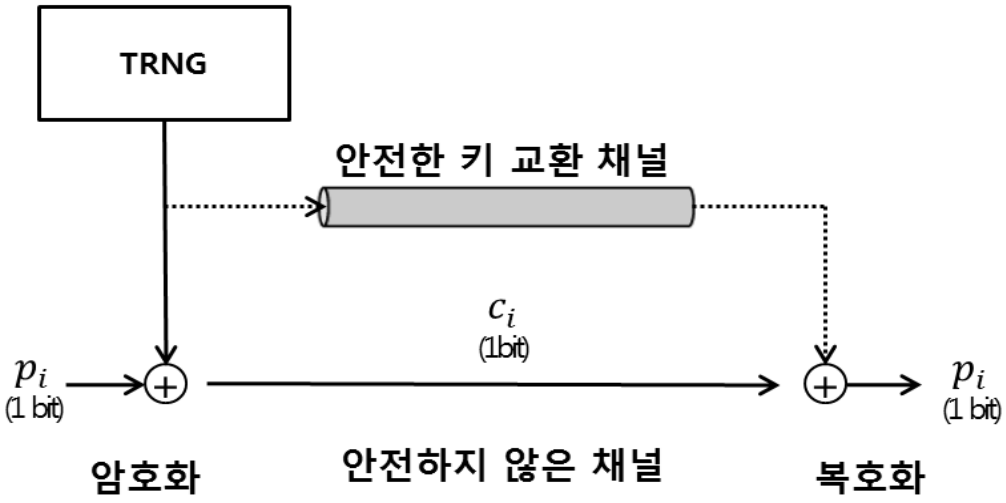
5.3 난수 생성

- 암호를 사용한 의사 난수 생성기
  - 암호 알고리즘을 이용하기 때문에 의사 난수열을 보고 카운터를 예상할 수 없음



5.4 이상적인 스트림 암호 : One-Time Pad (OTP)

- **완전 안전성 (Perfect Secrecy)**
  - 무한한 계산 능력을 지닌 공격자가 암호문을 관찰하고도 아무 정보도 얻을 수 없는 시스템
  - Pr[M=m|C=c] = Pr[M=m]



5.4 이상적인 스트림 암호 : One-Time Pad (OTP)

- OTP는 암호화 함수로 $c = m \oplus k$를 가지며, 복호화 함수는 $m = c \oplus k$인 스트림 암호로 다음 조건을 만족
  1. TRNG를 사용하여 키 스트림을 생성
  2. 키 스트림은 송신자와 수신자만이 공유
  3. 각 키 스트림 비트는 단 한 번만 사용

  - $c_i = m_i \oplus k_i$ (0 = 0 ⊕ 0 or 1 ⊕ 1, 1 = 0 ⊕ 1 or 1 ⊕ 0)
  - k_i 가 0일 확률과 1일 확률은 50%(i.e., TRNG가 생성) →공격자가 c_i 를 보고 m_i를 맞출 확률은 50%
    - Pr[M=m|C=c] = Pr[M=m]

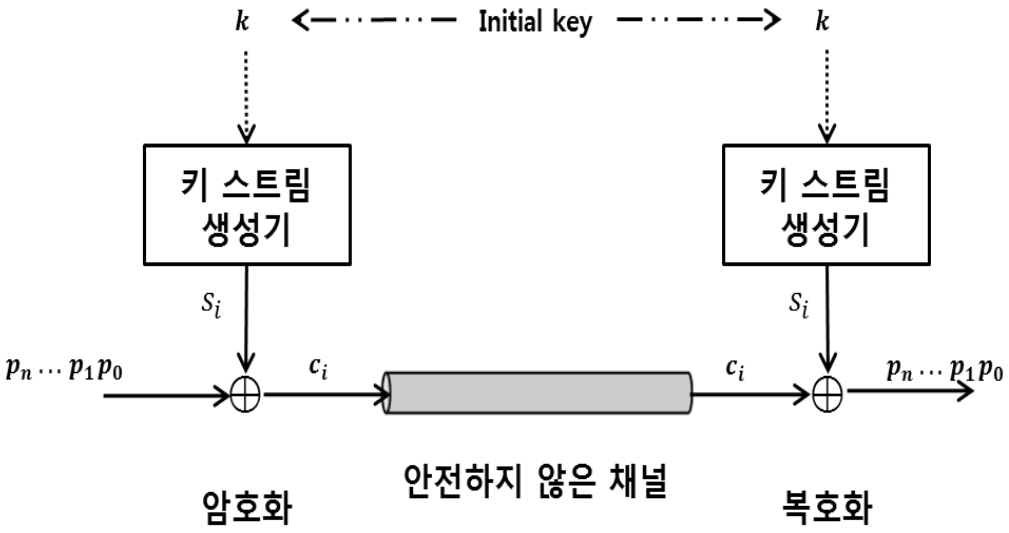
- 비현실적

5.5 실용적인 스트림 암호

- 키 스트림 생성기로 CSPRNG를 사용
  - CSPRNG는 결정적이기 때문에 수신자와 송신자가 안전하게 공유한 비밀키 k를 PRNG의 초기값으로 사용하여 동일한 키 스트림을 생성
  - PRNG를 사용하면 안전하기 못함
    - 평문의 일부분을 알면 전체 암호문을 모두 복호화 가능
    - $R_{i+1} = (a \times R_i + b) \bmod n$,
      - 비밀값은 a, b, R_0이고 n은 공개된 값
      - $R_1 = (s_1, \ldots, s_{10})$, $R_2 = (s_{11}, \ldots, s_{20})$, $R_3 = (s_{21}, \ldots, s_{30}), \ldots$
      - R_i는 $y_i = E_{si}(x_i) = x_i \oplus s_i$에서 사용되는 10개의 키 값을 생성
      - **단계 1.** 알고 있는 평문 30 비트 $x_1, \ldots, x_{30}$와 관측해서 얻은 암호문 30 비트 $y_1, \ldots, y_{30}$를 이용하여 $R_1 = (s_1, \ldots, s_{10})$, $R_2 = (s_{11}, \ldots, s_{20})$, $R_3 = (s_{21}, \ldots, s_{30})$를 얻는다.
      - **단계 2.** $R_2 \equiv (a \times R_1 + b) \bmod n$와 $R_3 \equiv (a \times R_2 + b) \bmod n$에서 $a \equiv (R_2 - R_3)/(R_1 - R_2) \bmod n$와 $b \equiv R_2 - R_1(R_2 - R_3)/(R_1 - R_2) \bmod n$ 계산

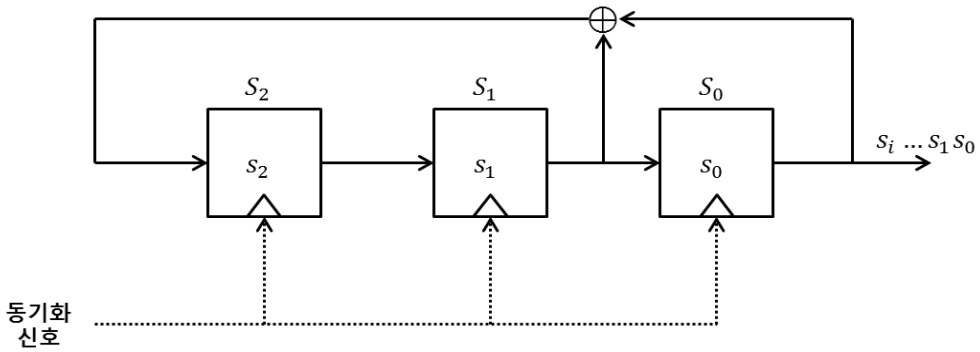
5.5 실용적인 스트림 암호

- OTP의 비현실성 → TRNG 대신 CSPRNG 사용
  - CSPRNG는 결정적이기 때문에 수신자와 송신자가 안전하게 공유한 비밀키 k를 PRNG의 초기값으로 사용하여 동일한 키 스트림을 생성



5.5 실용적인 스트림 암호

- 선형 피드백 쉬프트 레지스터(Linear Feedback Shift Register) 예)
  - $b_3 = b_1 \oplus b_0$

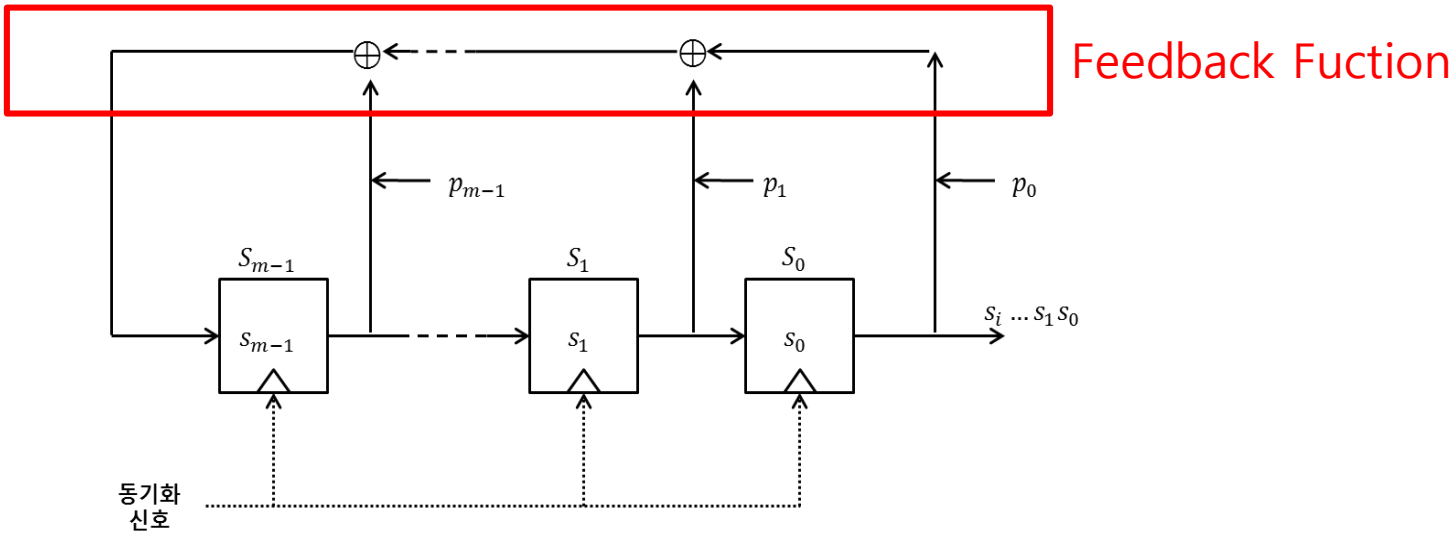


동기화 신호	S_2	S_1	S_0
0	1(=s_2)	0(=s_1)	0(=s_0)
1	0(=s_3)	1(=s_2)	0(=s_1)
2	1(=s_4)	0(=s_3)	1(=s_2)
3	1(=s_5)	1(=s_4)	0(=s_3)
4	1(=s_6)	1(=s_5)	1(=s_4)
5	0(=s_7)	1(=s_6)	1(=s_5)
6	0(=s_8)	0(=s_7)	1(=s_6)
7	1(=s_9)	0(=s_8)	0(=s_7)
8	0(=s_{10})	1(=s_9)	0(=s_8)
9	1(=s_{11})	0(=s_{10})	1(=s_9)
10	1(=s_{12})	1(=s_{11})	0(=s_{10})
...			

5.5 실용적인 스트림 암호

- 선형 피드백 쉬프트 레지스터(Linear Feedback Shift Register)
  - $s_m \leftarrow p_{m-1}s_{m-1} + \cdots + p_1 s_1 + p_0 s_0 \mod 2$
    - 피드백 회로가 활성화 또는 비활성화의 여부는 피드백 계수 $p_0, p_1, \ldots, p_{m-1}$의 값으로 표현
      - 즉 특성 다항식 $x^m + p_{m-1}x^{m-1} + p_{m-2}x^{m-2} + \cdots + p_0 x^0 = 0$으로 표현
      - 최대 주기는 $2^m - 1$

$x^m = p_{m-1}x^{m-1} + p_{m-2}x^{m-2} + \cdots + p_0 x^0$ 계수는 GF(2)



5.5 실용적인 스트림 암호

- **eSTREAM (유럽 연합의 eSTREAM 공모사업)**
  - 2008년 4월 총 3단계에 걸친 심사를 통하여 최종 스트림 암호가 당선
    - Profile1 : 높은 성능을 요구하는 소프트웨어 애플리케이션을 위한 스트림 암호
    - Profile2 : 제한된 자원(저장공간,게이트의 수, 전력량)을 가진 하드웨어 애플리케이션을 위한 스트림 암호

Profile 1 (SW)	Profile2 (HW)
HC-128	Grain v1
Rabbit	MICKEY v2
Salsa20/12	Trivium
SOSEMANUK	