스트림 암호 기술

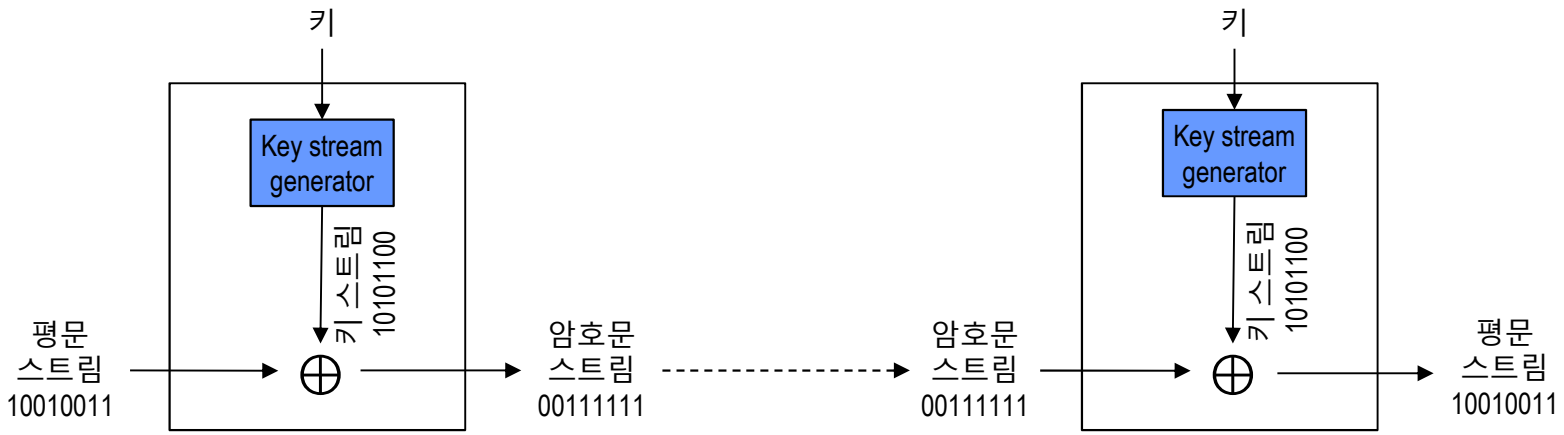
목차

- 스트림 암호 기술 개요
- RC4
- LFSR 기반 스트림 암호 기술

스트림 암호 기술 개요

스트림 암호 기술

- 일반적으로 한 번에 평문 한 바이트씩 암호화 수행 (비트 단위 혹은 한 바이트보다 큰 단위로도 설계 가능)
- Key stream generator의 출력인 키 스트림과 평문 스트림을 XOR하여 암호문 스트림 생성
- 일반적으로 블록 암호 시스템보다 빠르게 동작
- 실시간 처리에 효율적
- 2018년 TLS 1.3에 ChaCha20 포함 (IETF RFC 8446)
- RC4, A5/1, ChaCha20



10010011 평문 스트림
⊕ 10101100 키 스트림
00111111 암호문 스트림

00111111 암호문 스트림
⊕ 10101100 키 스트림
10010011 평문 스트림

RC4

Reference: (Stallings, 2014; Forouzan, 2008)
Reference: http://people.csail.mit.edu/rivest/pubs/RS14.pdf

RC4 개요

- Ronald Rivest에 의해 설계됨(1987년)
- 여러 데이터 통신 및 네트워크 프로토콜에 사용됨 → SSL/TLS, IEEE 802.11 무선랜
- Byte-oriented stream cipher → 평문의 한 바이트와 키스트림의 한 바이트를 XOR하여 암호문의 한 바이트 생성
- 비밀키 크기 → 1~256 바이트
- 2015년 TLS에서의 RC4 사용 금함 (IETF RFC 7465)

RC4 알고리즘(입력: 비밀키 key), N=256인 경우

① 256 바이트 크기 state 배열 S를 생성하여 0, 1, 2, …, 255의 값으로 초기화 → S[0]=0, S[1]=1, S[2]=2, …, S[255]=255
② S의 초기 순열 생성 → key 배열의 값을 활용하여, S 내 값들의 교환 작업 256회 수행 (이후 비밀키 미사용)
③ S의 현재 (순열) 상태에 기반하여 S의 순열 생성 후 S 내 하나의 원소 선택하여 키스트림으로 출력하는 작업 반복

RC4

Reference: (Stallings, 2014; Forouzan, 2008)
Reference: http://people.csail.mit.edu/rivest/pubs/RS14.pdf

```
public class Test {
	public static void main(String[] args) {
		int key[]= {0,0,0,0};
		int P[]= {0,1,2,3,4,5,6,7,8,9};
		int N=256;
		int C[]=RC4(N, key, P);
		System.out.println(Arrays.toString(C));
	}
	private static int[] RC4(int N, int[] key, int[] P) { // key:비밀키, P:평문
		int C[]=new int[P.length];
		int S[]=new int[N];
		for (int i=0; i < N; i++) S[i]=i; // state 배열을 0, 1, 2, ..., N-1의 값들로 초기화
		for (int i=0, j=0; i < N; i++) { // key 배열을 이용하여 state 배열 shuffling
			j=(j+S[i]+key[i % key.length]) % N;
			swap(S, i, j); // state 배열 내 S[i], S[j]의 값 교환
		}
		for (int i=0, j=0, t=0;  t < P.length; t++) { // 평문 바이트 크기만큼 반복
			i=(i+1) % N;
			j=(j+S[i]) % N;
			swap(S, i, j); // state 배열 내 S[i], S[j]의 값 교환
			int k=S[(S[i]+S[j]) % N]; // 키스트림의 다음 바이트로 사용될, state 배열 내 하나의 값을 선택
			C[t]=P[t] ^ k; // 평문 내 다음 바이트를 키스트림의 다음 바이트와 XOR 수행하여 암호문 스트림의 한 바이트 결정
		}
		return C;
	}
	private static void swap(int[] S, int i, int j) { int t=S[i];  S[i]=S[j];  S[j]=t; }
}
```

RC4 알고리즘(입력: 비밀키 key), N=256인 경우

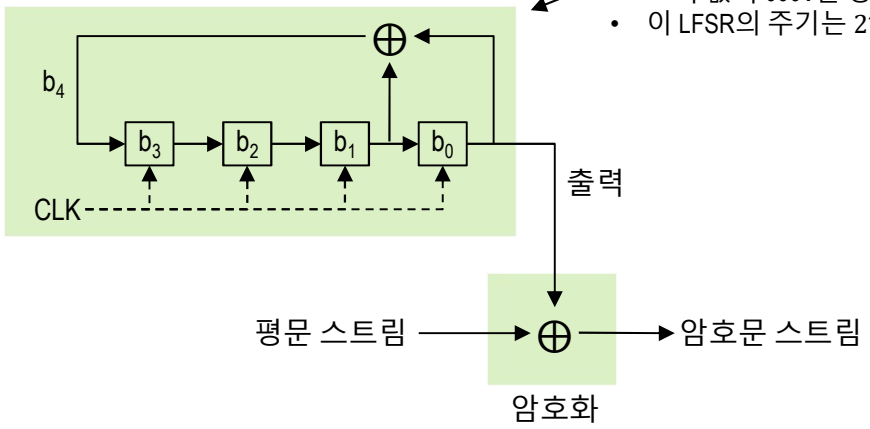
① 256 바이트 크기 state 배열 S를 생성하여 0, 1, 2, …, 255의 값으로 초기화 → S[0]=0, S[1]=1, S[2]=2, …, S[255]=255
② S의 초기 순열 생성 → key 배열의 값을 활용하여, S 내 값들의 교환 작업 256회 수행 (이후 비밀키 미사용)
③ S의 현재 (순열) 상태에 기반하여 S의 순열 생성 후 S 내 하나의 원소 선택하여 키스트림으로 출력하는 작업 반복

LFSR 기반 스트림 암호 기술

Reference: (Stallings, 2014; Forouzan, 2008; Paar & Pelzl, 2010)

LFSR (Linear Feedback Shift Register, 선형 피드백 쉬프트 레지스터)
- 이전 상태의 선형 함수(예: XOR)로 그 입력 값이 결정되는 쉬프트 레지스터
- 플립플롭(flip-flop)과 피드백 경로로 구성 → 하드웨어 구현 용이
- LFSR 기반 스트림 암호 기술의 예 → A5/1
- m 비트 LFSR의 최대 주기(maximum period)는 $2^m - 1$
- LFSR의 비트가 모두 0인 상태에 도달한 경우 이후 LFSR의 상태 변경 없음

A5/1
- 3개 LFSR들(각각 19, 22, 23 비트 크기)로 구성됨
- GSM 이동통신에서 휴대폰과 기지국 간 음성 암호화를 위해 사용됨



- 4개 플립플롭(4-bit shift register)과 피드백 경로로 구성
- 초기 값이 **0001**인 경우 출력 **100010011010111** 반복
- 이 LFSR의 주기는 $2^4 - 1 = 15$

클럭	b_3	b_2	b_1	b_0
초기값	0	0	0	1
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	1	0	0	1
5	1	1	0	0
6	0	1	1	0
7	1	0	1	1
8	0	1	0	1
9	1	0	1	0
10	1	1	0	1
11	1	1	1	0
12	1	1	1	1
13	0	1	1	1
14	0	0	1	1
15	0	0	0	1

특성다항식(characteristic polynomial)
- LFSR의 피드백 함수를, 계수가 $GF(2)$에서 정의되는, 특성다항식으로 표현
- 위 LFSR은 $x^4 + x + 1$으로 표현 가능

$$b_4 = 0 \cdot b_3 + 0 \cdot b_2 + 1 \cdot b_1 + 1 \cdot b_0$$
$$x^4 = 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$
$$x^4 = x + 1$$
$$x^4 + x + 1 = 0$$

최대 주기를 갖지 않는 LFSR 예
- 다항식 $x^4 + x^3 + x^2 + x + 1$에 대응하는 LFSR은 5의 주기를 가짐

References

- Behrouz A. Forouzan, Cryptography and Network Security, McGraw-Hill, 2008
- William Stallings, Cryptography and Network Security: Principles and Practice, Sixth Edition, Prentice Hall, 2014
- Christof Paar, Jan Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010
- 김명환, 수리암호학개론, 2019
- 정민석, 암호수학, 경문사, 2017
- 최은미, 정수와 암호론, 북스힐, 2019
- 이민섭, 정수론과 암호론, 교우사, 2008