15장. 네트워크 보안

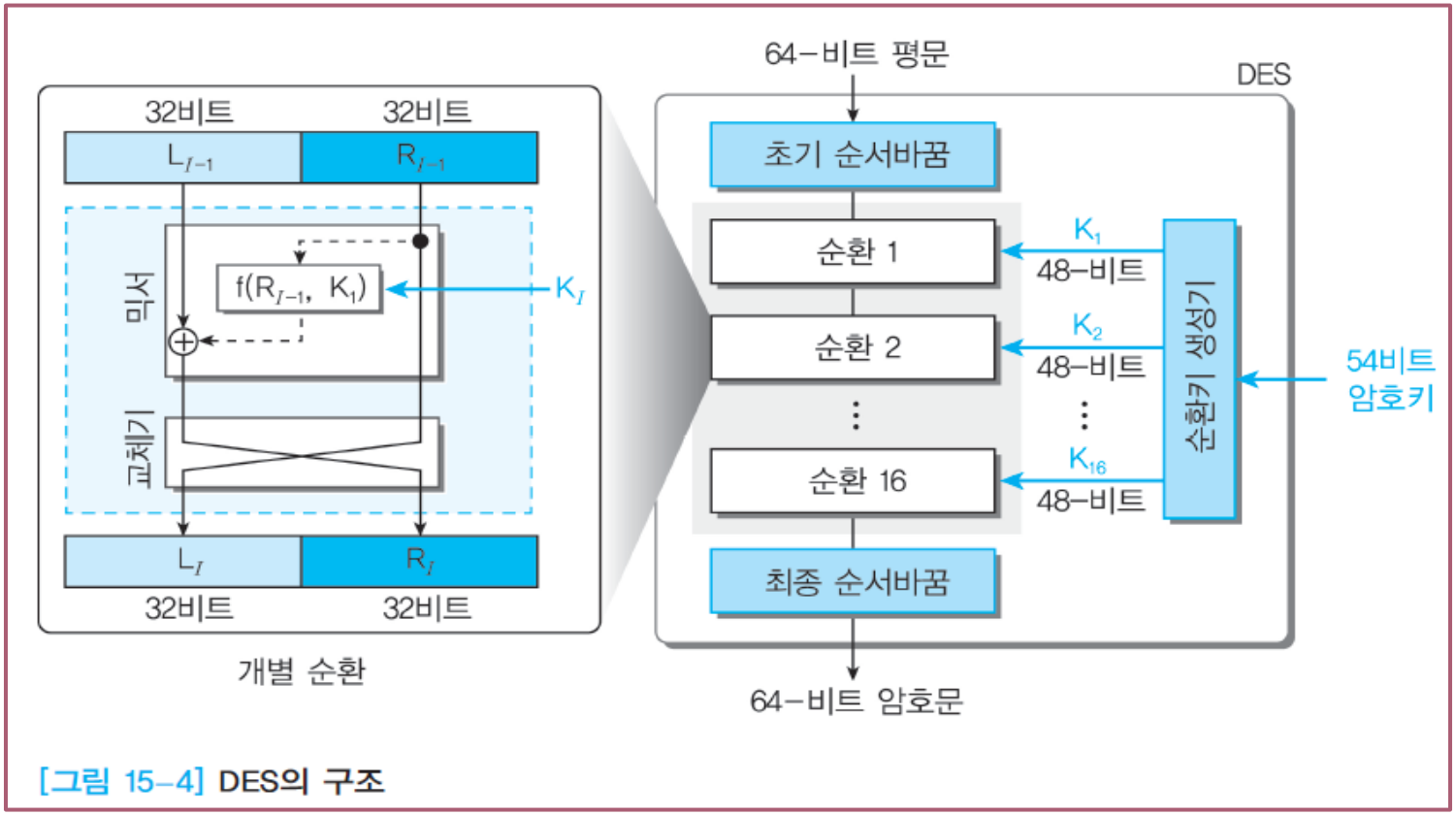
15-2 표준 암호 알고리즘

DES 암호화 (1)

- 암호화/복호화의 단위로 여러 비트들을 묶은 형태의 블록(block)을 사용하는 것이 일반적
- DES 알고리즘→ 복잡한 블록 암호문을 생성
  - DES 알고리즘은 시저 암호문과 유사하지만 규칙이 복잡함
  - IBM에서 설계, 비군사적 사용에 대한 암호화 표준으로 1976년 11월 미국 연방표준으로 승인
  - 1977년 1월, FIPS(Federal Information Processing Standards) PUB 46 공식 표준으로 발표
  - 이후 FIPS-46-1(1988년), FIPS-46-2(1993년), FIPS-46-3(1999년), 3-DEStriple DES로 수정, 발표
  - 2002년 5월, 후속 버전인 AES 표준에게 그 자리를 넘겨줌
  - DES는 대칭키 암호화, 즉 비밀키를 사용하는 방법
  - DES 표준(FIPS-46-3)은 2005년 5월 19일 공식적으로 폐지
  - 3-DES가 NIST(National Institute of Standards and Technology)에 의해 2030년까지 표준으로 승인
- 암호문 생성 측에서 DES의 구조와 구성 요소 간 동작의 연계성 → [그림 15-4]

DES 암호화 (2)

- 암호문 생성 측에서 DES의 구조와 구성 요소 간 동작의 연계성



[그림 15-4] DES의 구조

DES 암호화 (3)

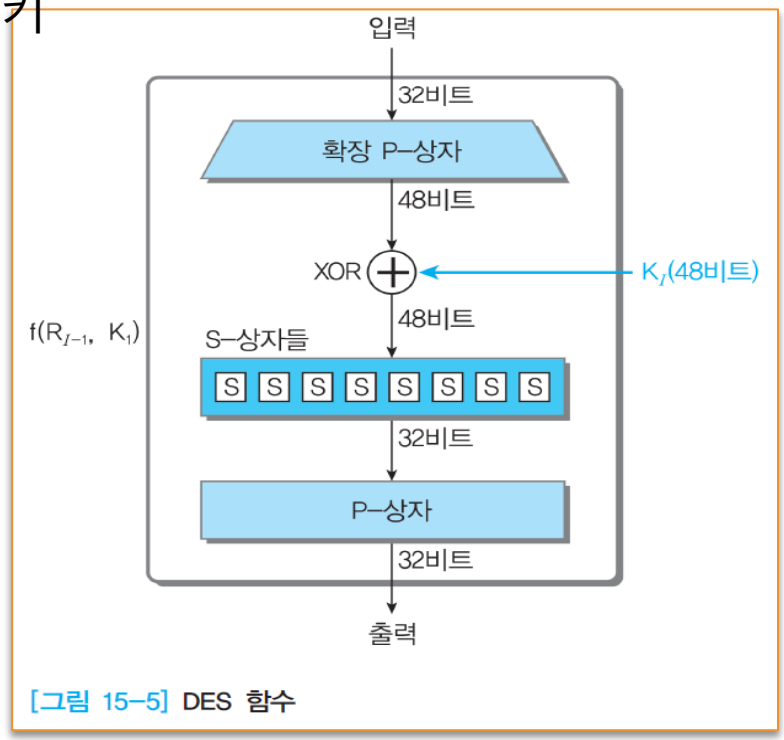
- 암호화를 수행하는 쪽
  - DES는 64-비트 plain text을 사용하여 64-비트 cipher text 생성
- 복호화를 수행하는 쪽
  - 64-비트 비트 cipher text를 64-비트 plain text로 바꿈
- 암호화 과정이나 복호화 과정 모두에서 동일하게 56-비트 암호키 사용
- initial permutation 과정
  - 64-비트 입력을 미리 정한 규칙에 따라 순서를 바꿈
- final permutation 과정
  - 초기 순서 바꿈의 역순으로 순서를 바꿈

DES 알고리즘의 동작 (1)

- 순환과정
  - 16번의 순환과정(round)을 거침
  - 각 순환과정에서는 swapper를 이용하여 역변환
  - 전 단계의 L_{I-1}, R_{I-1} 값을 받아서 R_{I-1}을 DES 함수를 이용하여 연산을 수행한 결과로 나온 출력값과 아무런 연산을 수행하지 않은 원래 L_{I-1} 값과의 X-OR 연산을 수행함
  - 그 결과를 교체기를 통해 오른쪽 32비트와 왼쪽 32비트를 서로 교체하여 최종적으로 L_I, R_I 를 생성한 다음, 그 다음 단계로 넘어감

DES 알고리즘의 동작 (2)

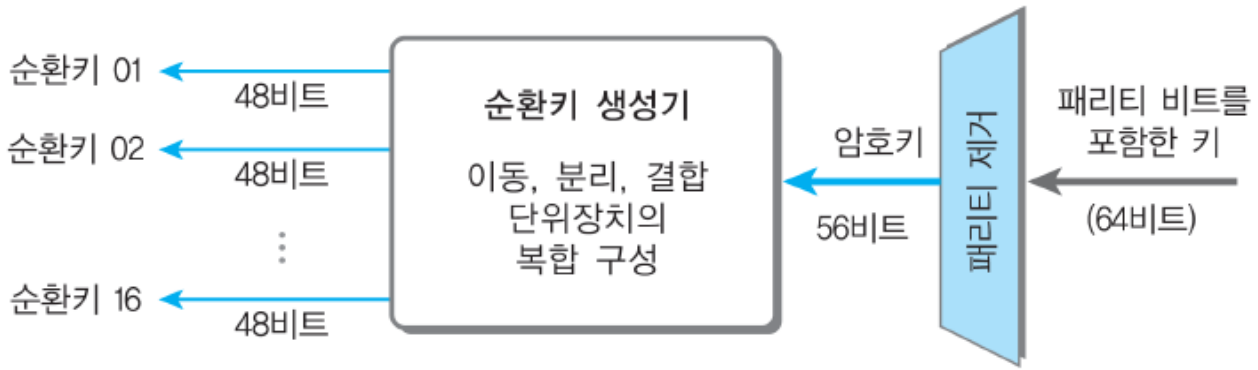
- DES 함수 실행 → DES 알고리즘의 핵심
  - DES 함수 → 48-비트키를 오른쪽 32-비트 (R_{I-1})에 적용하여 32-비트의 결과 생성
  - 확장 P-상자, X-OR 요소, S-상자 그룹, P-상자의 4개의 섹션으로 구성
  - (R_{I-1}) : 32-비트 입력값, K_I : 48-비트키
  - 확장 순서 바꿈(permutation)과정을 수행한 후, 확장된 오른쪽 섹션과 순환키의 X-OR 연산을 취함
  - 다음으로 S-상자를 이용하여 섞음 → 이때 DES는 8개의 S-상자 이용
  - 각각의 S-상자는 6비트를 입력으로 받아 4비트의 출력 생성
  - 최종적으로 32비트 입력과 32비트 출력으로 순서바꿈 수행



[그림 15–5] DES 함수

DES 알고리즘의 동작 (3)

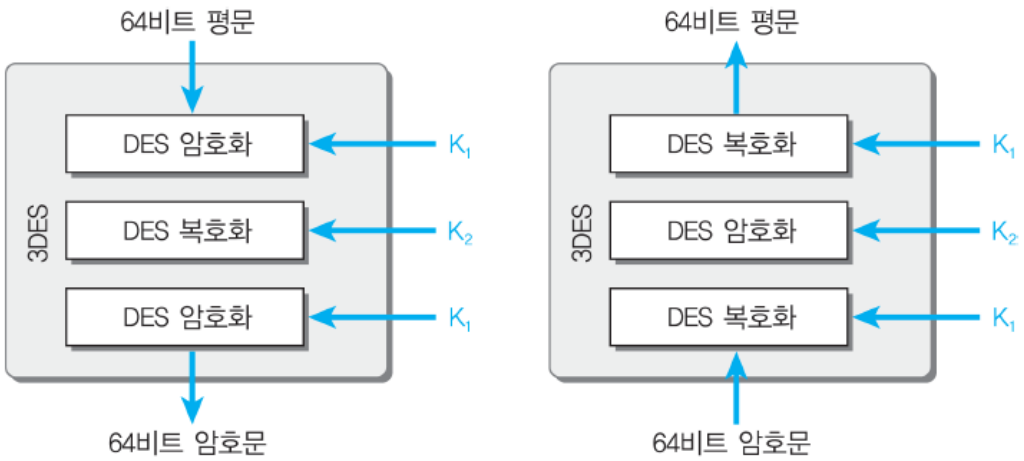
- DES 알고리즘의 키 생성
  - 순환키 생성기는 56-비트 암호키로부터 48-비트키를 생성
  - 정상적인 암호키는 64-비트키 → 이 중 8개는 잉여 비트인 패리티 비트임
  - 패리티 비트 → 키 생성 (key generation) 과정 전에 drop시키고 56-비트 키를 사용
  - 키 생성 과정



[그림 15-6] 키 생성 과정

3DES 기법

- 3DES 기법 -> 'Triple Data Encryption Algorithm(TDEA 또는 Triple DEA)'
- DES 암호 방식을 세 번 사용하는 방법
- 56비트키를 사용 → 이는 키의 길이가 비교적 짧아서 외부 공격에 취약하다는 단점
- 새로운 암호 알고리즘을 설계하는 대신 기존의 방식을 반복적으로 세 번 사용
  → 간단하게 암호키의 길이를 증가시키는 효과



[그림 15-7] 3DES의 구조

AES 암호화 (1)

- AES(Advanced Encryption Standard)에 대한 연구계획 발표 (1997년)
  - DES를 대체할 표준
  - 요구조건
    - ① DES와의 연속성을 갖도록 비밀키 대칭 블록 암호화 방식을 사용
    - ② 3DES보다 강력하고 빠르며
    - ③ 20~30년 정도의 수명을 갖도록 하는 것
  - 15개의 후보 알고리즘 중 요구조건을 만족하고 다양한 환경에서의 정상적인 동작과 구현의 용이성 등을 고려
    - 벨기에의 암호학자인 조앤 대먼(Joan Daemen)과 빈센트 리즈멘(Vicent Rijmen)박사가 만든 라인달(Rijndael)이 최종 선정 (2000년 10월)

AES 암호화 (2)

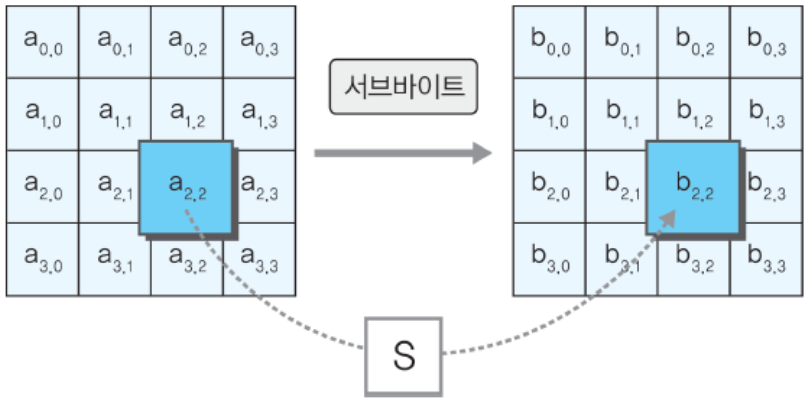
- AES의 동작과 DES와의 차이
  - DES가 각 순환 과정에서 F 순환을 반복한다면, AES는 IDEA처럼 순환과정을 반복함
  - IDEA란?
    - 1990년대초 유럽에서 발표된 암호 알고리즘인 PES/IPES(Proposed Encryption standard/Improved PES1)의 또 다른 이름
  - IDEA는 128비트키를 이용하며 64비트 데이터를 암호화
  - 암호화/복호화 과정에서 8번의 순환과정을 사용하고, 각 순환과정은 6개의 16비트 서브키를 사용하고 있음

AES 암호화 (3)

- AES의 암호화 단계
  - ❶ AES의 데이터의 단위가 128비트 → 다시 4개의 32비트 데이터로 나누어짐
    - 순환키는 키 크기에 따라 가변적
    - 예) 128비트 키인 경우 암호화/복호화 과정에서 순환과정을 9번 반복하고, 192 비트 키라면 순환과정을 11번 반복함
  - ❷ 각 순환과정에서는 먼저 데이터 비트마다 S-상자를 이용한 치환을 한 번 수행하고, 가상 순서바꿈 연산을 해서 그룹에 있는 비트를 다른 그룹으로 섞음
  - ❸ 각 그룹은 행렬 형태로 곱해지고 그 결과가 순환의 서브키에 추가

AES 암호화 (4)

- AES 암호화 단계 (계속)
  - 서브바이트(SubBytes) 단계
    - 순환과정 중 첫번째 단계를 나타낸 것으로 S-상자를 이용하여 각 $a_{i,\ j}$ 를 $b_{i,\ j} = S(a_{i,\ j})$ 로 치환하는 과정



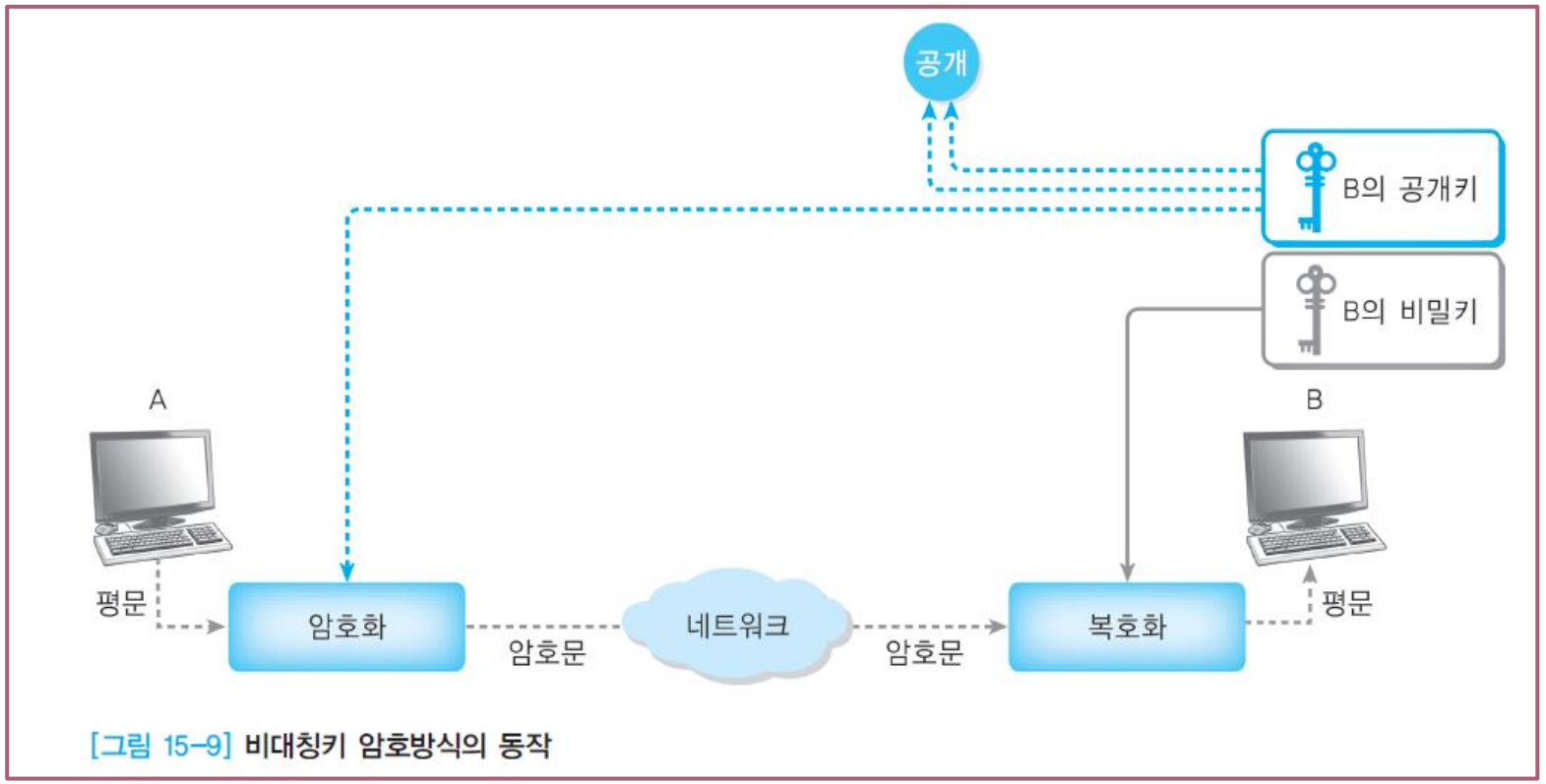
[그림 15-8] AES의 서브바이트 단계

비대칭 암호화 방식과 RSA 기법 (1)

- 대칭키 암호방식과 비대칭키 암호방식의 차이
  - 어떻게 비밀을 유지하는가
    - 대칭키 암호방식
      - 전송 측과 수신 측 모두 비밀 키를 공유하고, 상호 간에 반드시 비밀을 유지해야 함
    - 비대칭키 암호방식
      - 비밀 유지의 책임은 각자에게 있으며, 전송 측과 수신 측은 각각 자신의 비밀키를 만들어서 보관함
- n명의 집단이 있다고 하면, 대칭키 암호방식에서는 개의 공유 비밀키가 $\frac{n(n-1)}{2}$ 필요한 반면, 비대칭키 암호방식에서는 각자 n개의 비밀키만 필요

비대칭 암호화 방식과 RSA 기법 (2)

- 비대칭키 암호방식의 동작



[그림 15-9] 비대칭키 암호방식의 동작

비대칭 암호화 방식과 RSA 기법 (3)

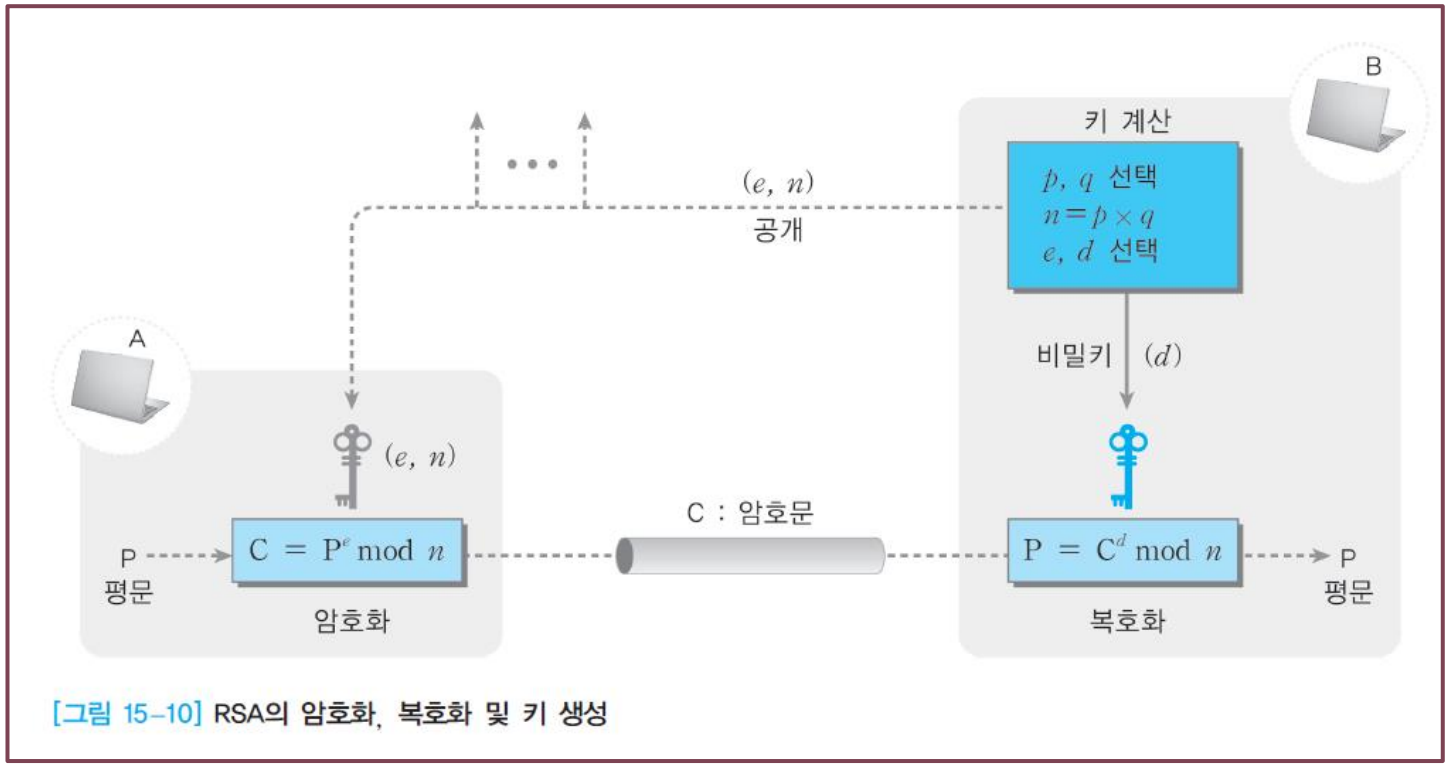
- 비대칭키 암호방식의 동작 (계속)
  - 비대칭키 암호문에서는 개인키(private key)와 공개키(public key)와 같은 2개의 키를 각각 분리해서 사용
  - A는 B의 공개키를 사용하여 암호화를 수행한 다음, 암호화된 메시지를 전송함
  - 암호화된 메시지는 B가 갖고 있는 개인키가 있어야만 복호화가 가능
  - 개인키가 있는 B만 암호문을 평문으로 바꿀 수 있게 됨

비대칭 암호화 방식의 동작

- 비대칭키 암호방식의 실제적인 동작
  - 대칭키 암호방식은 치환과 순서바꿈 기법을 사용
  - 비대칭키 암호방식은 매우 큰 수에 대한 수학적인 문제, 즉 한쪽으로는 계산을 손쉽게 할 수 있지만, 반대로는 계산하기가 매우 어려운 문제를 이용
  - 예) 두 수의 곱은 쉽게 구할 수 있지만, 역으로 곱하기 전의 처음 두 수를 구하는 것은 쉬운 일이 아님 → 두 수를 곱한 계산 결과가 수백자리 수라면, 원래의 두 수를 구하는 것은 더욱 어렵게 됨

RSA 암호화 (1)

- RSA 암호화



[그림 15-10] RSA의 암호화, 복호화 및 키 생성

RSA 암호화 (2)

- RSA 암호화 동작
  - RSA ->Rivest, Shamir, Adleman
  - RSA는 e와 d로 표시되는 두 개의 지수(exponent)를 사용
  - e는 공개되고 d는 비밀을 유지
  - P는 평문이고 C는 암호문을 나타낸다고 했을 때, A는 $C = P^e \bmod n$ 이라는 수식을 사용하여 평문인 P로부터 암호문 C를 생성하여 전송함
  - B가 A로부터 C라는 메시지를 받으면, B는 $P = C^d \bmod n$ 라는 수식을 이용하여 복호화를 수행 → 다시 원래의 평문으로 변환
    - n: 매우 큰 숫자로 (키 생성 과정에서 만들어짐)

암호방식의 비교

- 대칭키, 비대칭키 암호방식의 장,단점
  - 대칭키 암호방식
    - 전송 측과 수신 측 모두 동일한 비밀키를 사용
    - 알고리즘으로 구현할 경우 상대적으로 복잡하지 않음
    - 암호화/복호화 시간도 단축
  - 비대칭키 암호방식
    - 사용자들이 각기 자신의 개인키를 관리하고 비밀 유지의 책임도 각자에게 있어 키 관리 측면에서는 유리
    - 구현된 비대칭 알고리즘은 상대적으로 복잡
    - 암호화/복호화 시간이 오래 걸림
  - 대칭키 암호방식
    - 일반적으로 긴 메시지를 전송하는 데 적합
  - 비대칭키 암호방식
    - 비교적 짧은 메시지를 전송하는 데 적합

RSA 기법 – 예제 (1)

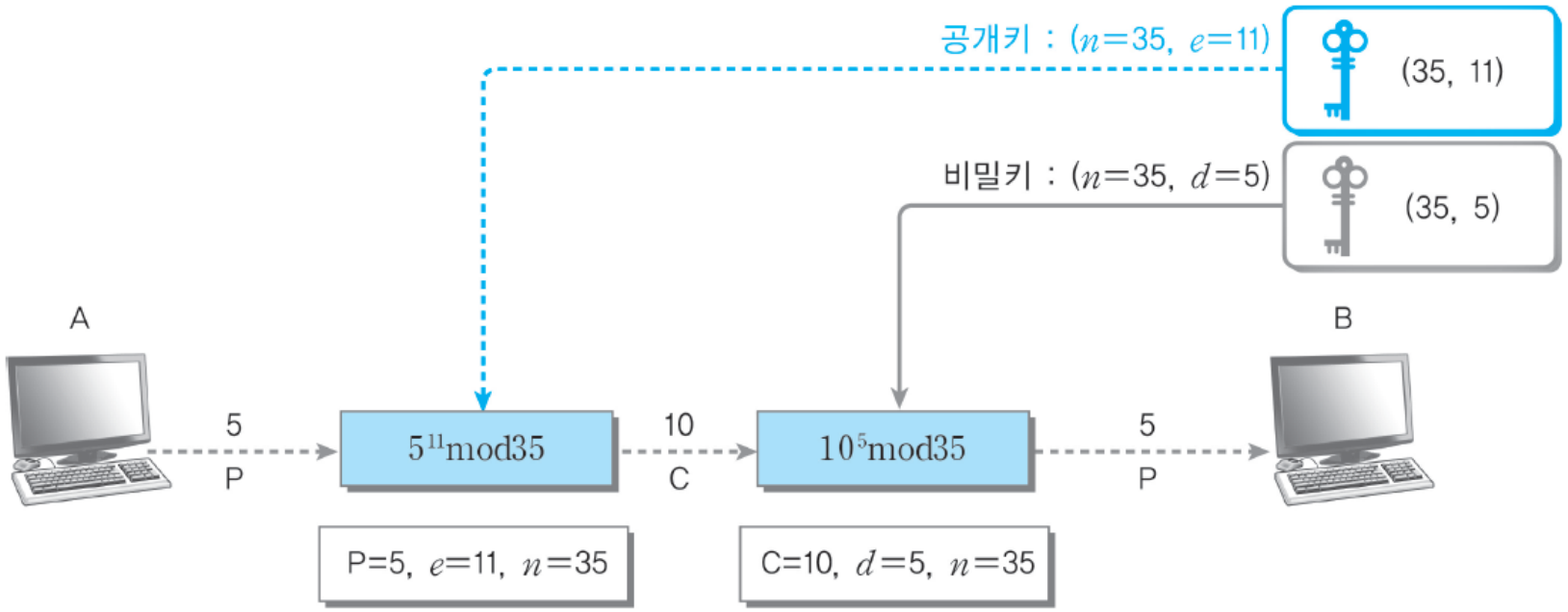
예제 15-3

A와 B가 RSA 암호방식을 이용하여 메시지를 전송하는 경우를 생각해보자. B는 공개키로 $e = 11$, $n = 35$를 사용하고, 비밀키 $d = 5$를 갖고 있다. A가 평문 $P = 5$를 B로 전송하는 과정을 설명하라.

풀이

[그림 15-11]과 같이, 먼저 A는 $C = P^e \bmod n$이라는 수식을 사용하여 평문인 P로부터 암호문 C를 만든다. 즉 $C = P^e \bmod n = 5^{11} \bmod 35 = 10$이 되어 암호문인 10을 전송한다. 이제 B가 A로부터 $C = 10$이라는 메시지를 받으면, 이것을 사용하여 복호화함으로써 다시 원래의 평문을 만들어낸다. 즉 $P = C^d \bmod n = 10^5 \bmod 35 = 5$가 되어 원래의 메시지가 만들어진다.

RSA 기법 – 예제 (2)



[그림 15-11] RSA 암호방식의 예