
3. 공개키 암호 시스템

담당교수: 차 영욱

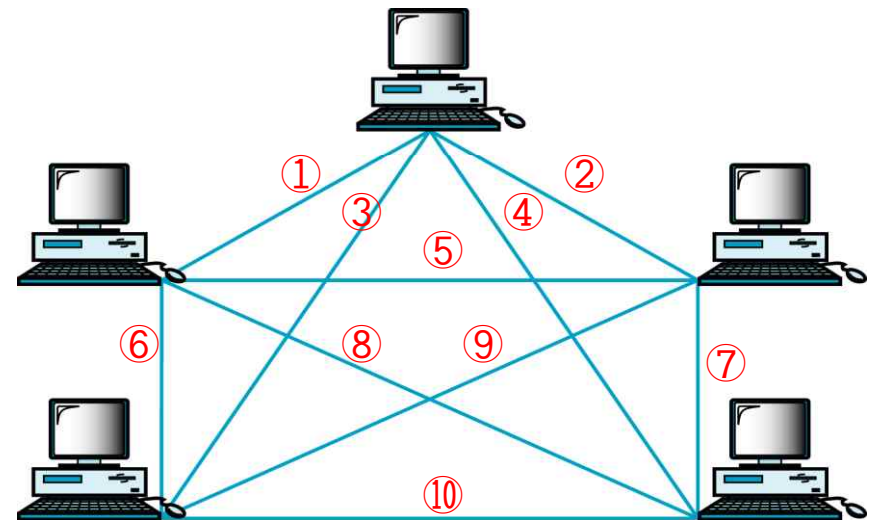
ywcha@andong.ac.kr

목 차

- 공개키 암호 시스템의 개요
- 공개키 암호 시스템의 아이디어
- 기본 정리
- RSA 공개키 암호 시스템
- Diffie-Hellman 키 교환
- 대칭키 암호 시스템과 공개키 암호 시스템 비교

대칭키 암호 시스템

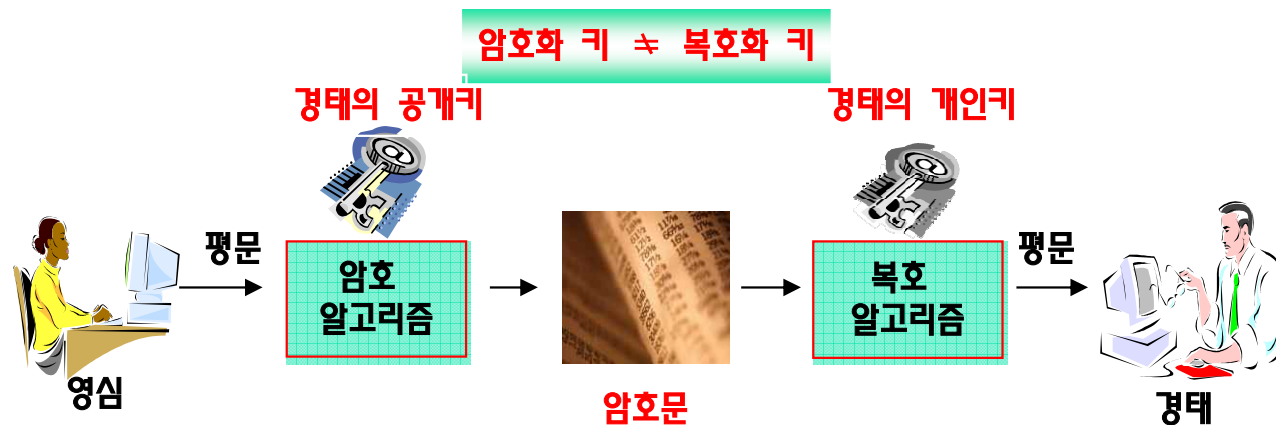
- 암호화와 복호화에 동일한 키 사용
- **암호키 분배**: 송신자와 수신자는 안전한 채널을 통해서 암호 키가 먼저 교환되어야 함
- **암호키 관리**: N 명의 상호 통신을 위하여 요구되는 암호 키의 개수 $(N*(N-1)/2)$ 가 많이 요구됨
 - $N=5 \Rightarrow 5*(5-1)/2=10$ 개
 - $N=10 \Rightarrow 10*(10-1)/2=45$ 개
 - $N=20 \Rightarrow 20*(20-1)/2=190$ 개



공개키 암호 시스템 개요

□ 암호화 및 복호화

- 영심이는 네트워크에 공개되어 있는 형태의 공개키로 문서를 암호화
- 경태는 비밀리에 보관하고 있는 개인키로 암호화된 문서를 복호화



□ N 명의 암호 통신을 위하여 요구되는 암호 키의 개수는 2N

- $N=5 \Rightarrow 10$ 개
- $N=10 \Rightarrow 20$ 개

공개키 기법과 암호 시스템



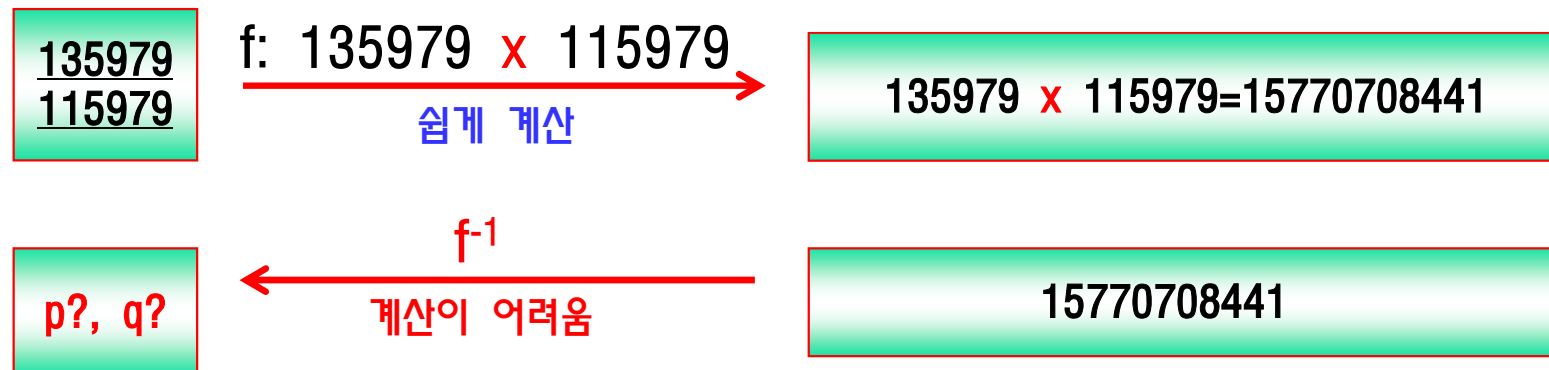
공개키 암호 시스템의 아이디어[1/2]

□ 일방향 함수

- 입력 x 에 대하여 출력 $y=f(x)$ 의 계산은 용이,
- 출력 y 에 대하여 역으로 입력 $x=f^{-1}(y)$ 를 구하는 것이 불가능한 함수

□ 일방향 함수의 예

- 큰 소수 p 와 q 에 대하여 두 수의 곱인 $y = p \times q$ 의 계산은 쉬움
- 출력 y 가 주어지더라도 두 수 p 와 q 의 분해는 매우 어려운 작업



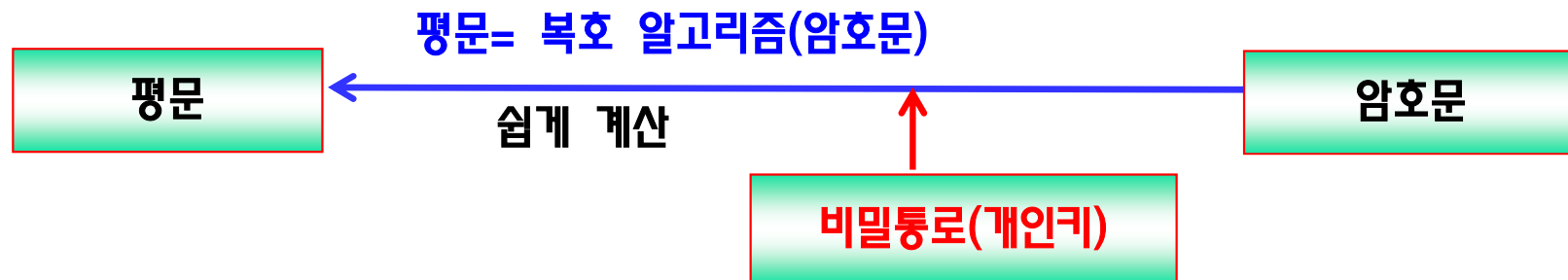
공개키 암호 시스템의 아이디어[2/2]

□ 비밀통로 일방향 함수

- 추가적인 정보를 이용하면 쉽게 역 연산이 가능

□ 비밀통로 일방향 함수의 적용 예

- 암호문 = 암호 알고리즘(평문) ➡ 암호 알고리즘은 일방향 함수
- 평문 = 복호 알고리즘(암호문) ➡ 개인키는 암호문에서 평문을 유도하는
비밀통로 일방향 함수 역할





기본 정리-소수 구하기

- ❑ 소수(Prime Number): 1과 그 수 자신으로만 나누어 떨어지는 수(예, 2, 3, 5, 7, 11, 13, ...)
- ❑ 그리스의 수학자 에라토스테네스(BC 276년~194년)의 체를 이용한 소수 구하기
 - 1을 제거
 - 2를 남기고 2의 배수를 모두 제거
 - 3을 남기고 3의 배수를 모두 제거
 - 5를 남기고 5의 배수를 모두 제거
 - 이와 같이 계속하여 체로 걸러진 수들이 소수 임




1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

기본 정리-모듈러 연산과 최대공약수(1/2)

□ 법 연산: $a \bmod n = r$

- 양의 정수 a 를 n 으로 나누었을 때의 나머지 값 r
- $25 \bmod 3 = 1$  25를 3으로 나눈 나머지 값 1
- $33 \bmod 9 = 6$  33을 9로 나눈 나머지 값 6

□ 정수들의 집합, $Z = \{\dots, -2, -1, 0, 1, 2, \dots\}$

- 임의의 두 정수 $a, b \in Z$ 에 대해 a 를 $b \geq 1$ 로 나눈 나머지가 r , 몫이 q 이면 $a = b * q + r, 0 \leq r < b$
- $r = 0$ 이면 b 는 a 의 약수(divisor) 또는 인수(factor)이며, $b|a$ 로 표시
- 임의의 정수 $a, b, c \in Z$ 에 대해 $c|a, c|b$ 이면 c 를 a 와 b 의 공약수(common divisor)
- $\gcd(a, b)$  a, b 의 공약수 중에서 가장 큰 양의 정수
- $\gcd(a, b) = 1$ 이면 정수 a 와 b 는 서로 소(relatively prime)의 관계
 - 11과 15의 최대공약수, $11 = 1 * 11, 15 = 1 * 15$  1  11과 15는 서로 소
- 임의의 정수 $p \geq 2$ 에 대한 약수가 1과 p 밖에 없다면 p 는 소수

기본 정리-모듈러 연산과 최대공약수(2/2)

□ 유클리드 알고리즘: 두 정수 a , b 의 최대 공약수를 구하는 알고리즘

- 12와 15의 최대공약수, $12 = 3 * 4$, $15 = 3 * 5$ ➡ 3
- $a=252$ 와 $b=198$ 의 최대 공약수(18) 구하기

① $252 \bmod 198 \rightarrow 54$;

② $198 \bmod 54 \rightarrow 36$;

③ $54 \bmod 36 \rightarrow 18$;

④ $36 \bmod 18 \rightarrow 0$;

① $s \leftarrow a$; $t \leftarrow b$;

② $\text{while}(t > 0)$

③ $\{r \leftarrow s \bmod t; s \leftarrow t; t \leftarrow r;\}$

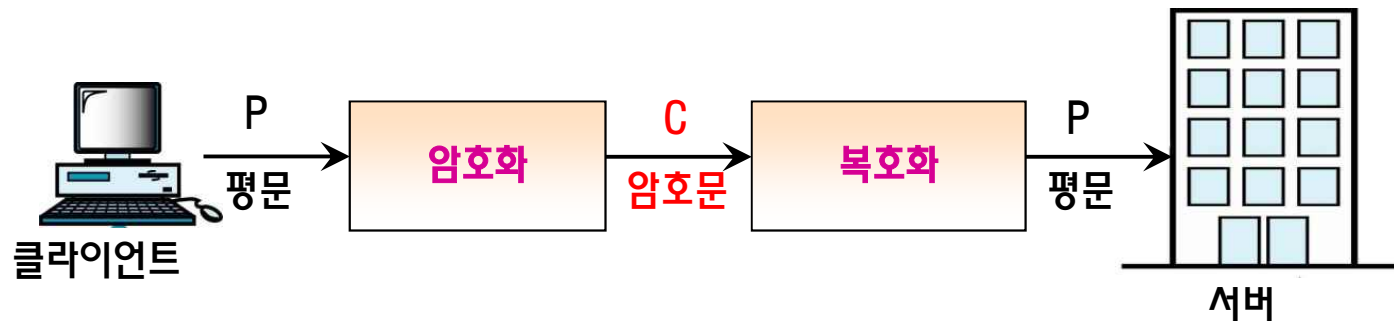
④ $\text{return}(s)$;

RSA 공개키 암호 시스템

□ 1978년 MIT의 Rivest, Shamir, 그리고 Adleman에 의해 개발된 최초의 공개키 암호 시스템

□ RSA 공개키 암호 시스템 기반의 통신

- 두 소수의 곱: N
- 서버의 공개키: K_p
- 서버의 개인키: K_s
- 암호화 알고리즘: 평문(P)을 공개키로 곱성한 후 N 으로 나눈 나머지 값
 - $C = P^{K_p} \bmod N$
- 복호화 알고리즘: 암호문(C)을 개인키로 곱성한 후 N 으로 나눈 나머지 값
 - $P = C^{K_s} \bmod N$



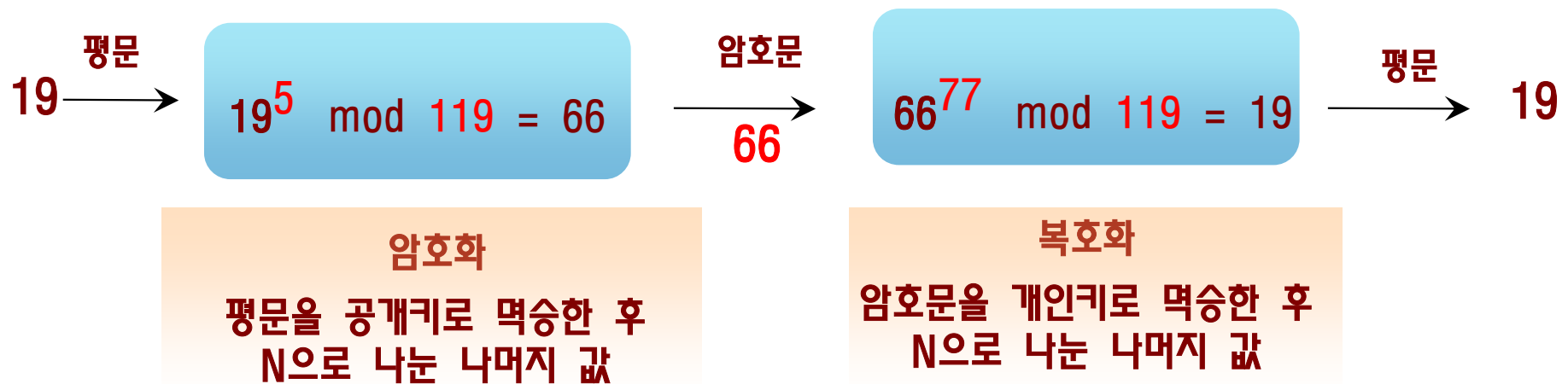
RSA 암호 시스템의 공개키 및 비밀키 생성

- 비밀로 유지하는 값: 비밀키, 두 개의 소수 p , q
- 공개되는 값: 공개키, N

- ① 두 개의 소수 $p = 7$, $q = 17$ 선택
- ② $N = p * q = 7 * 17 = 119$
- ③ $\phi(n) = (p-1) * (q-1) = 96$
- ④ $\phi(n)$ 에 서로 소인 공개키 K_p 를 선택, $K_p = 5$
- ⑤ $(K_p * K_s) \bmod \phi(n) = 1$ 인 K_s 선택
 $5 * 77 = 385 = 4 * 96 + 1$ 이므로 $K_s = 77$

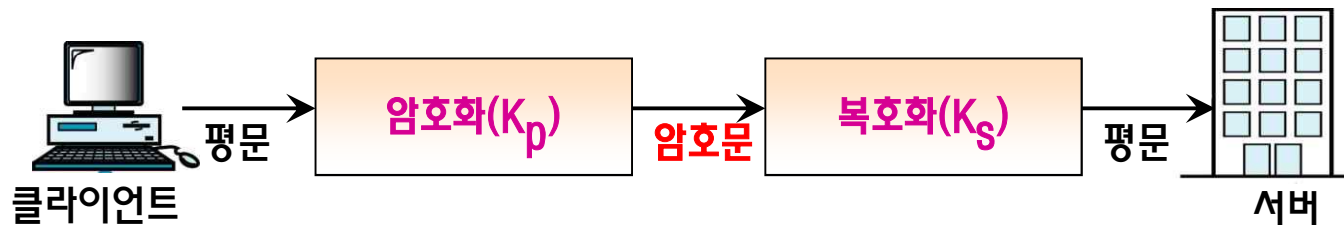
RSA 암호 시스템의 실행 예

- 공개키: 5
- 비밀키: 77
- 두 개의 소수 p 와 q 의 곱, N : 119



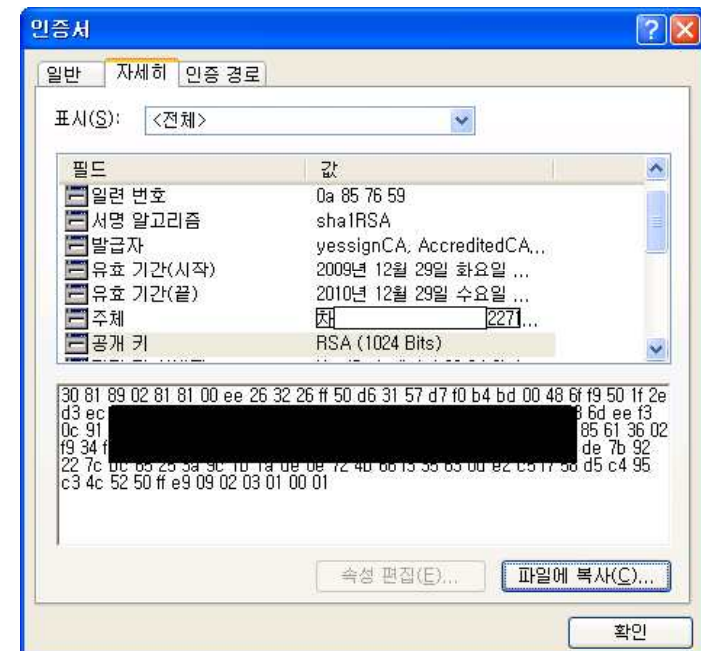
RSA의 안전성91/2]

- ❑ 큰 두 소수의 곱인 N 을 소인수 분해하는 문제의 어려움에 기반
 - $60=2*2*3*5$ 과 같은 작은 수의 소인수 분해는 간단
 - $2^{113}-1 = 3391*23279*1868569*1066818132868207$ 과 같은 큰 수의 소인수 분해는 상당히 많은 시간 요구
- ❑ 서버
 - 두 소수 p, q 로 부터 N , 공개키(K_p), 개인키(K_s)의 계산은 간단
- ❑ 공격자
 - N 과 서버의 공개키 정보를 획득하더라도 $\rightarrow N$ 에서 p 와 q 를 분해하기 어려움
 \rightarrow 서버의 개인키(K_s)를 생성하는 것이 상당히 어려움



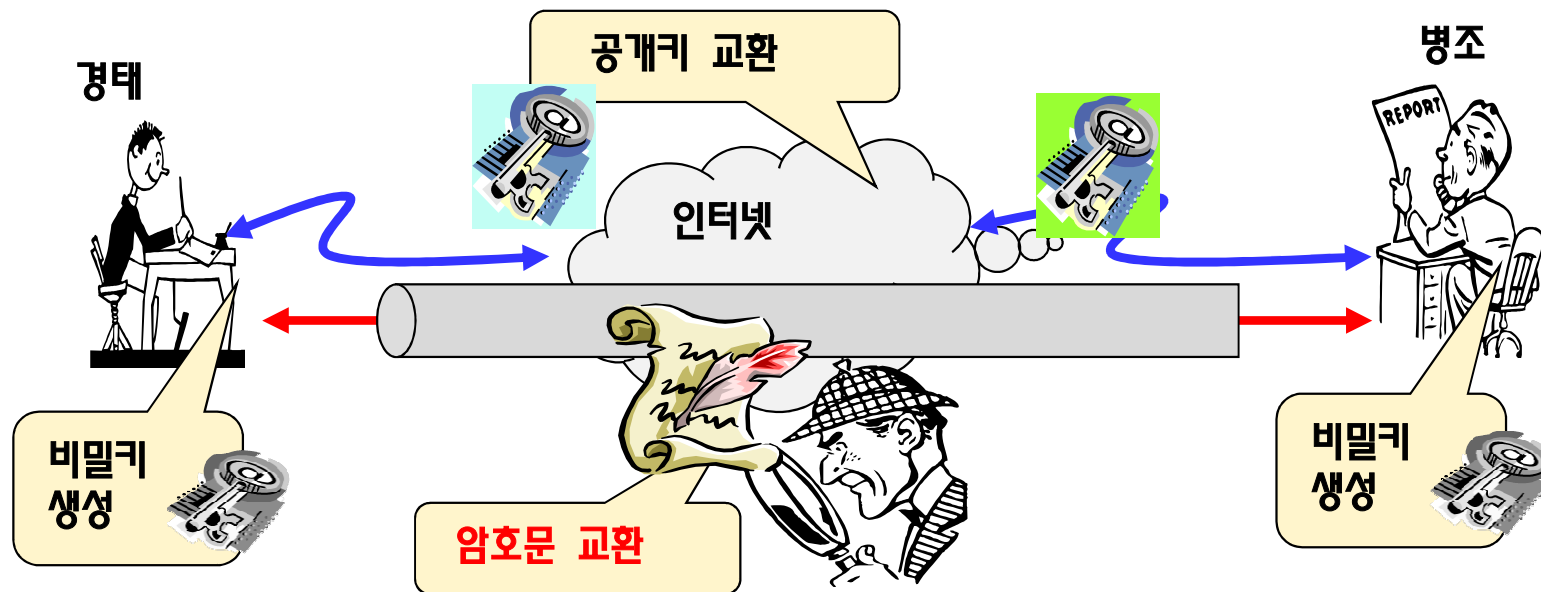
RSA의 안전성(2/2)

- ❑ 1999년에 네트워크에 연결된 292개 컴퓨터들의 협동 작업으로 512비트의 N 값에 대한 소인수분해에 5.2 개월 소요
 - 참고: 50 비트의 2진수 최대값은 약 1000조(2^{50})
- ❑ RSA 암호 시스템의 안전성을 보장하기 위하여 적어도 1024비트 길이의 N 값 사용
- ❑ RSA의 1024 비트 공개키



Diffie-Hellman의 키 교환

- ❑ 1976년에 최초로 발표된 공개키 암호 기법
- ❑ 공개키를 교환하여 상호간에 사용할 비밀키 생성
- ❑ 비밀키는 암호문의 생성 및 평문의 복구를 위한 암호 및 복호 키로 사용



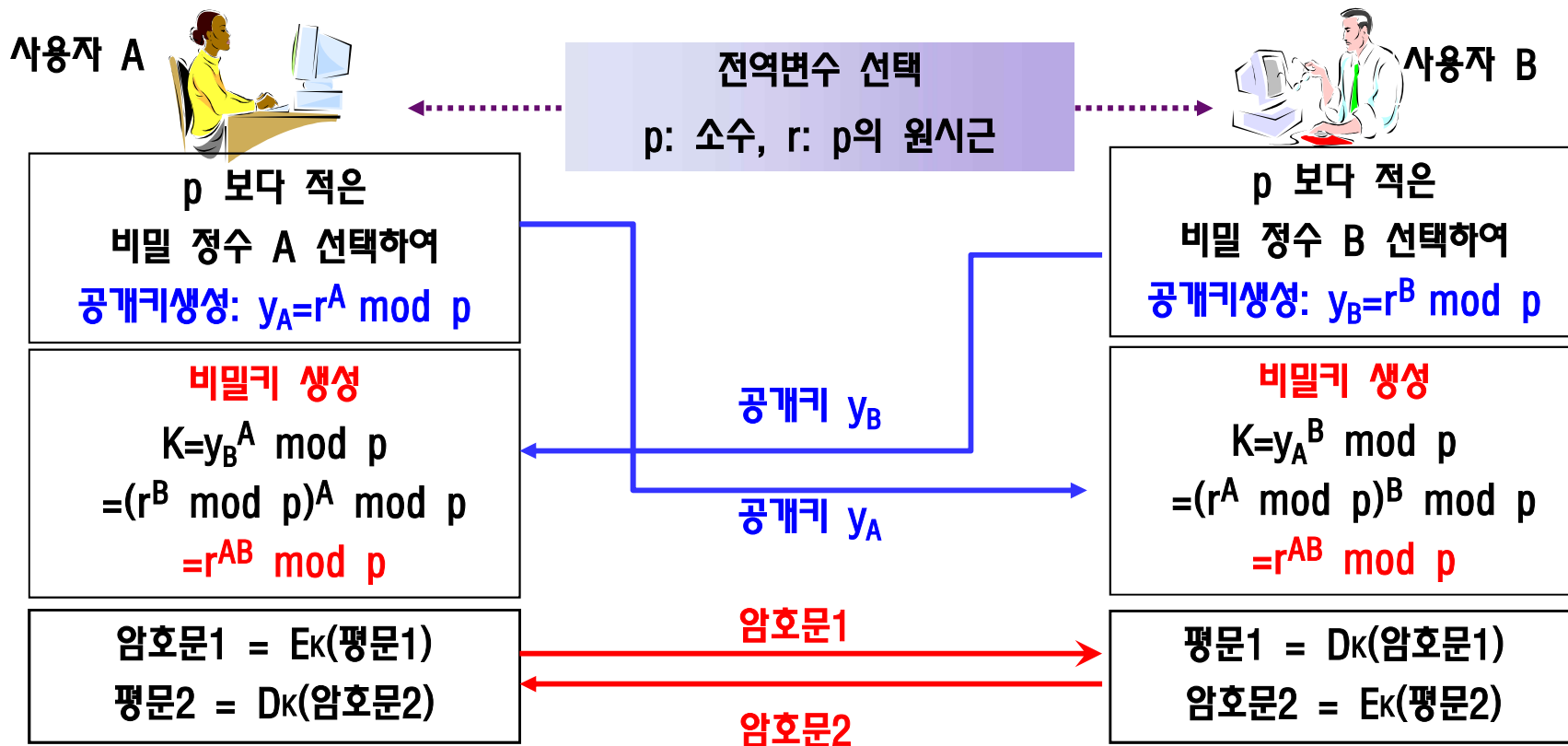
기본정리-원시근(1/2)

- 법 n 에 대한 유한한 정수들의 집합 $Z_n = \{0, 1, 2, \dots, n-1\}$ 에서
 - b 의 곱셈상의 역원(multiplicative inverse)이 존재한다면 $b * b^{-1} \bmod n = 1$ 을 만족하여야 함.
 - 즉, $b * b^{-1} \equiv 1 \pmod{n}$
- 법 n 과 b 가 서로 소(relatively prime)의 관계에 있을 경우에만 b 의 곱셈상의 역원이 존재
 - Z_{15} 의 원소: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$
 - 15에 서로 소인 수: 1, 2, 7, 11, 13, 14
 - $1^{-1} = 1, 2^{-1} = 8, 7^{-1} = 13, 11^{-1} = 11, 13^{-1} = 7, 14^{-1} = 14$
 - $2 * 2^{-1} = 2 * 8 = 16 \equiv 1 \pmod{15}$
 - $14 * 14^{-1} = 14 * 14 = 196 \equiv 1 \pmod{15}$
- 유한체(finite field) $Z_{p^*} = \{0, 1, 2, 3, \dots, p-1\}$
 - 체에 있는 0을 제외한 모든 수들은 곱셈 상의 역원을 갖는다.
 - 체의 모든 원소들이 소수 p 와 서로 소

기본정리-원시근(2/2)

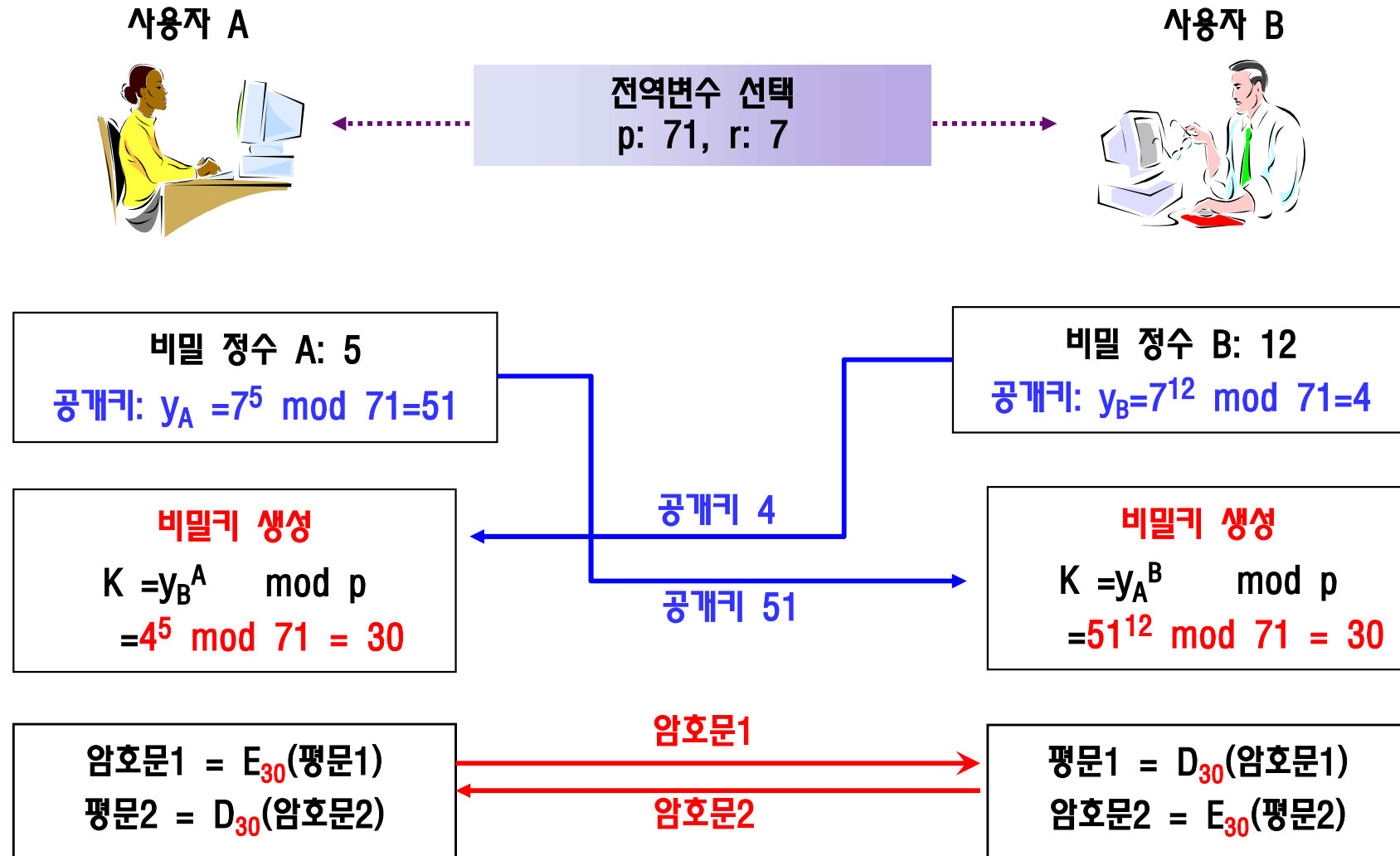
- 생성원 g 를 임의의 $a(0 \leq a \leq p-2)$ 로 멍승하여 modular 연산을 하면 모든 원소들이 만들어짐
- 생성원 g : 법(modulo) p 에 대한 원시근(primitive root)
- 소수 7에 대한 유한체 $Z_{7^*} : \{1,2,3,4,5,6\}$
 - 곱셈상의 역원 $\Rightarrow 1^{-1} = 1, 2^{-1} = 4, 3^{-1} = 5, 4^{-1} = 2, 5^{-1} = 3, 6^{-1} = 6$
 - $3^0 \bmod 7=1, 3^2 \bmod 7=2, 3^1 \bmod 7=3,$
 - $3^4 \bmod 7=4, 3^5 \bmod 7=5, 3^3 \bmod 7=6$
 - 모든 원소는 3에 대한 임의의 $a(0 \leq a \leq 5)$ 멍승으로 생성, \rightarrow 3은 법 7 상의 원시근
- 소수 13에 대한 유한체 $Z_{13^*} : \{1,2,3,4,5,6,7,8,9,10,11,12\}$
 - 모든 원소들은 2를 임의의 a 멍승하여 만들어짐, $0 \leq a \leq 11$
 - $2^0 \bmod 13=1, 2^1 \bmod 13=2, 2^4 \bmod 13=3, 2^2 \bmod 13=4,$
 - $2^9 \bmod 13=5, 2^5 \bmod 13=6, 2^{11} \bmod 13=7, 2^3 \bmod 13=8,$
 - $2^8 \bmod 13=9, 2^{10} \bmod 13=10, 2^7 \bmod 13=11, 2^6 \bmod 13=12$
 - 모든 원소는 2에 대한 임의의 $a(0 \leq a \leq 11)$ 멍승으로 생성, \rightarrow 2는 법 13 상의 원시근

Diffie-Hellman의 키 교환 절차(1/2)



- 안전성: 공개키 y_A, y_B 로 부터 공격자가 비밀 정수(A, B)를 구하는 것이 계산적으로 불가능하다는 이산대수문제(discrete logarithm problem)를 이용
 - 즉, $y_A = r^A \bmod p$ 를 계산해서 y_A 를 구하기는 쉬우나, 반대로 y_A 를 알고 있다고 해도 $A = \log_r y_A$ 를 통해 A 를 구하기 어려움

Diffie-Hellman의 키 교환 절차(2/2)



중간자 공격

□ Diffie-Hellman 키 교환의 중간자 공격에 대한 취약성

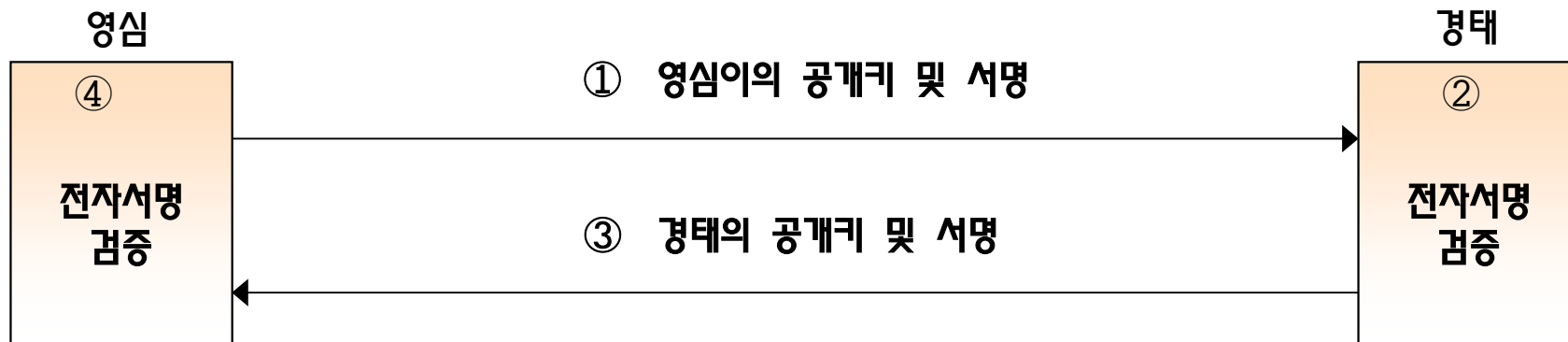


- 영심이 생성한 공개키(y_A)를 공격자는 y_{AA} 로 변조하여 경태에게 전달
- 경태는 영심과 사용할 비밀키를 공격자의 공개키인 y_{AA} 를 이용하여 생성
- 경태가 생성한 공개키(y_B)를 공격자는 y_{BB} 로 변조하여 영심에게 전달
- 영심은 경태와 사용할 비밀키를 공격자의 공개키인 y_{BB} 를 이용하여 생성
- 결과적으로 영심과 경태는 서로 다른 비밀키를 생성하며, 공격자를 경유한 통신이 수행됨.

□ 중간자 공격의 해결책 🖐️ 인증된 Diffie-Hellman 키 교환 사용

인증된 Diffie-Hellman 키 교환

- ① 영심이는 공개키를 생성하여 전자 서명한 후 경태에게 전송
- ② 경태는 공개된 검증 알고리즘으로 수신한 서명이 영심이에 의하여 생성된 것인지 검증
- ③ 경태도 공개키를 생성하여 전자 서명한 후 영심에게 전송
- ④ 영심이는 공개된 검증 알고리즘으로 수신한 서명이 경태에 의하여 생성된 것인지 검증



공개키와 대칭키 암호 시스템의 비교

□ 대칭키 암호 시스템

- 기본 연산을 주로 이용하므로 계산 시간이 빠름
 - 고속의 처리를 요구하는 IP 보안 프로토콜(IPSec)에 사용
- 암호키 관리: 네트워크에서 소요되는 전체 키의 개수가 많음
- 암호키 분배: 사전에 비밀키가 안전하게 분배되어 있어야 함

□ 공개키 암호 시스템

- 대칭키 암호 시스템의 단점인 암호 키의 관리와 분배 문제 해결
- 큰 정수와 연관된 곱셈 연산이 포함되어 계산 시간이 많이 소요
 - 전자상거래 및 인터넷 뱅킹과 같은 응용에 사용

요점 정리[1/2]

□ 공개키 암호 시스템

- 송신자는 수신자의 공개키로 문서를 암호화
- 수신자는 비밀리에 보관하고 있는 개인키로 암호화된 문서를 복호화
- N 명의 암호 통신을 위하여 요구되는 **암호 키(공개키+개인키)의 개수는 $2N$**

□ 일방향 함수

- 입력 x 에 대하여 출력 $y=f(x)$ 의 계산은 용이,
- 출력 y 에 대하여 역으로 입력 $x=f^{-1}(y)$ 를 구하는 것이 불가능한 함수

□ 비밀통로 일방향 함수

- 추가적인 정보를 이용하면 쉽게 일방향 함수의 역 연산이 가능
- 암호문 = 암호 알고리즘(평문) ➡ 암호 알고리즘은 일방향 함수
- 평문 = 복호 알고리즘(암호문) ➡ 개인키는 암호문에서 평문을 유도하는 비밀통로 일방향 함수 역할

□ RSA 암호 시스템

- 1978년 MIT에서 발표된 최초의 공개키 암호 시스템
- 비밀로 유지하는 값: 비밀키, 두 개의 소수 p, q
- 공개되는 값: 공개키, 두 소수의 곱(N)

요점 정리[2/2]

❑ Diffie-Hellman 키 교환

- 1976년에 최초로 발표된 공개키 암호 기법
- 공개키를 교환하여 상호간에 사용할 비밀키 생성
- 비밀키는 암호문의 생성 및 평문의 복구를 위한 암호 및 복호 키로 사용
- 중간자 공격의 취약성 ➡ **인증된 Diffie-Hellman 키 교환 사용**
 - 공격자에 의한 공개키의 변조를 방지하기 위하여 전자 서명된 공개키를 상호 교환

❑ 암호 시스템의 비교

- 대칭키 암호 시스템: **암호화 키 = 복호화 키**
 - 고속의 처리를 요구하는 IP 보안 프로토콜(IPSec)에 사용
 - 네트워크에서 소요되는 전체 키의 개수가 많음($N*(N-1)/2$)
 - 사전에 비밀키가 안전하게 분배되어 있어야 함
- 공개키 암호 시스템: **암호화 키 \neq 복호화 키**
 - 대칭키 암호 시스템의 단점인 암호 키의 관리와 분배 문제 해결
 - 큰 정수와 연관된 곱셈 연산이 포함되어 계산 시간이 많이 소요(전자상거래 및 인터넷 뱅킹과 같은 응용에 사용)