차례

4.1 개요

■ 대칭키 암호 vs 공개키 암호

- 대칭키 암호: 암호화와 복호화에 같은 키 사용
  - 키를 안전하게 공유하는 것이 어려움
- 공개키 암호: 서로 다른 키 쌍(공개키, 개인키) 사용
  - 공개키는 모두에게 알려져 있어도 되고, 개인키는 본인만 보관
  - 대칭키 배포 문제를 해결 할 수 있음

■ **공개키 암호의 특징**

- 단방향 함수 기반
  - 어떤 입력 -> 출력은 쉽다
  - 출력 -> 입력은 사실상 불가능(컴퓨터로 풀기 어려움)
  - Ex) 소인수 분해
    - 큰 수 $N = p \times q$ 는 쉽게 계산됨
    - 하지만 N으로 부터 p, q를 찾는 것은 어려움
  - Trapdoor 함수(덫문 함수)
    - 단방향 함수 이지만, “특정한 비밀 정보(개인키)”

4.1 개요

- 공개키 암호의 응용
  - 암호화
    - 발신자가 수신자의 공개키로 암호화 -> 수신자만 개인키로 복호화 가능
    - 예시: 인터넷 뱅킹, 메신저 보안
  - 디지털 서명
    - 발신자가 자신의 개인키로 메시지를 암호화 -> 누구나 발신자의 공개키로 검증
    - 종이 서명과 달리 위조 복제가 어려움
  - 키 교환
    - 대칭키 암호를 안전하게 사용할 수 있도록, 공개키 암호를 이용해 세션키 교환

- **예시**
  - Alice -> Bob 메시지 전송
    - Alice : bob의 공개키로 메시지를 암호화
    - Bob : 자신의 개인키로 복호화
  - 디지털 서명
    - Alice : 자신의 개인키로 서명
    - Bob: Alice의 공개키로 서명 검증

4.1 개요

- 여러 가지 공개키 암호체계
  - 배낭암호체계(knapsack cryptosystem)
  - 표준 공개키 암호인 RSA
  - 디피-헬먼 키 교환 알고리즘
  - 타원곡선 암호체계(ECC)

- 양자 컴퓨팅이 공개키 암호에 미치는 영향

4.2 배낭암호

- 배낭(= 부분합 문제) : “정해진 물건(가중치들) 중 몇 개를 골라 정확히 목표합을 만들 수 있나?”를 맞히는 퍼즐.

- 슈퍼증가 수열을 쓰면 이 퍼즐은 그리디로 순식간에 풀린다.

- **Merkle–Hellman(MH)**은 **쉬운 퍼즐(슈퍼증가)**을 비밀키로 갖고, 바깥엔 “어려워 보이는 퍼즐”만 보이게 변환한다. 하지만 이 변환이 충분히 복잡하지 않아 **샤미르(Shamir)·LLL 격자 공격**으로 무너졌다.

4.2 배낭암호

- 디피(Diffie)와 헬먼(Hellman)
  - 1976년 “공개키 암호(public-key cryptography)” 개념을 처음 세상에 발표
  - 키 교환 알고리즘(key exchange algorithm)만을 제시

4.2 배낭암호

■ 머클-헬먼

- 배낭암호체계(Merkle-Hellman knapsack cryptosystem)를 제안
- 머클-헬먼 배낭암호체계는 NP-완비로 알려진 문제를 기반으로 사용
  - 어려운 문제를 쓰면 안전하다

1. 비밀키로 쉬운 퍼즐(슈퍼증가 수열 K)을 고른다
2. K를 “겉보기엔 일반 배낭” 처럼 보이게 모듈러 연산 + 곱셈으로 섞어 공개키 B를 만든다
3. 송신자는 공개키 B만 보고 메시지를 부분합으로 만들어 암호문 C를 보낸다
4. 수신자는 비밀키로 C를 역변환해 다시 “쉬운 퍼즐(슈퍼증가)”로 바꾼 뒤 그리디로 해를 복원한다.

- 그리디(Greedy, 탐욕법) : 순간마다 가장 큰 것을 고르는 방법

4.2 배낭암호

■ 머클-헬먼

- 배낭암호체계(Merkle-Hellman knapsack cryptosystem)를 제안
- 머클-헬먼 배낭암호체계는 NP-완비로 알려진 문제를 기반으로 사용
  - 어려운 문제를 쓰면 안전하다
- 배낭문제는 다음과 같이 표현할 수 있음
  - 예) n개의 배낭에 담을 수 있는 무게가 다음과 같고 목표로 하는 총합이 S

$$W = (W_0,\ W_1,\ \ldots,\ W_{n-1})$$

  - $ai \in \{0, 1\}$인 $(a_0,\ a_1,\ \ldots,\ a_{n-1})$을 찾아서 S가 다음과 같이 되게 하기

$$S = a_0 W_0 + a_1 W_1 + \ldots + a_{n-1} W_{n-1}$$

  - 배낭 무게 W는 다음과 같다고 가정

$$W = (85, 13, 9, 7, 47, 27, 99, 86)$$

  - 희망하는 무게의 총합인 S=172라면 85+13+47+27=172

$$a = (a_0,\ a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7) = (11001100)$$

4.2 배낭암호

■ 슈퍼증가(superincreasing) 배낭문제

- 무게가 가장 작은 것에서 가장 큰 것으로 배열할 때 각 무게는 이전의 모든 무게를 더한 것보다 크다는 점만 제외하면 일반적인 배낭문제와 유사

- 예) 슈퍼증가 배낭문제(총합이 S=309라고 가정)

$$K=(3, 6, 11, 25, 46, 95, 200, 411) \qquad (4.1)$$

  - a_i를 구하기 위해 가장 큰 무게부터 가장 작은 무게를 순차적으로 찾아감

$S < 411$이기 때문에, $a_7=0$

$S > 200$이므로, $a_6=1$

$S=S-200=109$

$S > 95$이므로, $a_5=1$이고 $S=109-95=14$라고 계산

$a=10100110$

$3+11+95+200=309$이므로 해답이 맞는지 쉽게 검증할 수 있음

4.2 배낭암호

- 배낭암호체계를 구성하기 위해 사용하는 절차
  1. 슈퍼증가 배낭을 생성한다.
  2. 슈퍼증가 배낭을 일반 배낭으로 변환한다.
  3. 공개키는 일반 배낭이 된다
  4. 개인키는 변환 값이 있는 슈퍼증가 배낭키가 된다.

4.2 배낭암호

■ 키 생성 과정을 보여주는 자세한 예

❶ 다음과 같은 슈퍼증가 배낭을 선택한다.

K=(2, 3, 7, 14, 30, 57, 120, 251)

❷ 슈퍼증가 배낭을 일반 배낭과 같은 배낭으로 변환시키기 위해 승수 m 과 계수 n을 선정(m과 n은 서로소 공약수가 1)

$$\begin{aligned}
2m &= 2\cdot 41 = 82 \pmod{491}\\
3m &= 3\cdot 41 = 123 \pmod{491}\\
7m &= 7\cdot 41 = 287 \pmod{491}\\
14m &= 14\cdot 41 = 83 \pmod{491}\\
30m &= 30\cdot 41 = 248 \pmod{491}\\
57m &= 57\cdot 41 = 373 \pmod{491}\\
120m &= 120\cdot 41 = 10 \pmod{491}\\
251m &= 251\cdot 41 = 471 \pmod{491}
\end{aligned}$$

그 결과로 나타나는 배낭은 (82, 123, 287, 83, 248, 373, 10, 471)

4.2 배낭암호

❸ 공개키는 일반 배낭으로 보이는 배낭이다.

공개키: (82, 123, 287, 83, 248, 373, 10, 471)

❹ 개인키는 $m^{-1} \bmod n$인 전환 계수의 곱셈역을 포함하는 슈퍼증가 배낭

개인키: (2, 3, 7, 14, 30, 57, 120, 251) 그리고 41^{-1}(mod 491) =12

- 다음을 가정
  - 밥의 공개키와 개인키 쌍이 각각 위에서 제시한 ❸과 ❹로 주어짐
  - 앨리스는 밥에게 보낼 이진수 메시지 M=10010110을 암호화함
  - 앨리스는 총합이 암호문이 되는 공개키 배낭 요소를 선택하기 위해 메시지의 1비트 사용
  - 앨리스는 다음과 같이 계산: C = 82+83+373+10 = 548
  - 이 암호문을 복호화하기 위해 밥은 공개키를 사용

$$m^{-1} \cdot C \pmod{n} = 12 \cdot 548 \pmod{491} = 193$$

  - 밥은 193에 대한 슈퍼증가 개인키 배낭을 품
  - 밥은 개인키를 가지고 있기 때문에 이진수로 M=10010110 혹은 10진수로 M=150인 메시지를 찾는 것은 쉬움
  - 이 예시에서는 다음을 알고 있음: 548=82+83+373+10

4.2 배낭암호

- 다음과 같은 결론에 이름

$$
\begin{aligned}
548m^{-1} &= 82m^{-1}+83m^{-1}+373m^{-1}+10m^{-1} \\
&= (2m)m^{-1}+(14m)m^{-1}+(57m)m^{-1}+(120m)m^{-1} \\
&= 2(mm^{-1})+14(mm^{-1})+57(mm^{-1})+120(mm^{-1}) \\
&= 2+14+57+120 \\
&= 193 \pmod{491}
\end{aligned}
$$

  - 이 예시는 m^{-1}을 곱함으로써 공개키 배낭 영역에 있는 암호문을 개인키 슈퍼증가 영역으로 변환시킨다는 것을 보여줌

- 암호문 값 C로 총합이 되는 공개키 요소의 부분 집합을 찾을 수 있다면 공격자 트루디는 개인키 없이 메시지를 풀 수 있음
- 트루디는 다음과 같은 배낭의 부분 집합을 찾아야만 함

$$K=(82, 123, 287, 83, 248, 373, 10, 471)$$

4.2 배낭암호

- Shamir 계열 공격:
  - 공개키를 “어떤 숫자(곱셈의 역원 또는 비슷한 배수)”로 곱해 보면서, **원래의 슈퍼증가(복호가 쉬운 형태)**가 드러나는지를 찾아냅니다. (즉, 겉보기 BB를 비밀 변환의 역으로 되돌리려고 함)
- LLL / 격자 공격:
  - 부분합 문제(정답을 표현한 0/1 벡터)는 격자(수학적 격자구조) 안에서 특별히 짧은 벡터로 나타날 수 있음
  - LLL이라는 알고리즘이 그런 ‘짧은 벡터’를 골라냅니다. 이 ‘짧은 벡터’가 곧 해(=원문 비트열)를 가리킬 수 있습니다.

4.3 RSA

■ RSA

- Diffie-Hellman의 불편한점
  - 공유 비밀 생성 방식일 뿐, 암호문 구조가 없음.
    - DH는 “키 교환 프로토콜 " 이라서 결과물은 단지 공유된 수
    - 메시지를 암호화 하려면 K를 대칭키로 다시 써야함
  - 직접 암호화 규격 부재
    - RSA처럼 직접 암호화 수식이 없음
    - 따라서 메시지를 M 자체로 변환하는 과정이 표준화되지 못했음.
  - 인증 부족
    - 순수 DH는 키 교환만 하므로 “누구와 키를 공유했는지“ 증명할 방법이 없음
    - 중간자 공격(MiTM)에 취약 -> 단독 메시지 암호화에 바로 쓰기 곤란

4.3 RSA

■ RSA

- 리베스트(Rivest), 샤미르(Shamir), 애들맨(Adleman)의 이름을 따서 지음
- DH의 공개키 개념을 실제로 메시지 암호화 전자서명에 적용할 수 있도록 한 첫 실용적 공개키 암호 시스템.
- 핵심
  - 곱셈은 쉽지만 소인수 분해는 어렵다는 정수론적 비대칭성
  - 오일러 정리와 모듈서 산술 구조를 기반으로 암호화 복호화가 수학적으로 일관되게 동작
- RSA는 인수분해를 통해 해독할 수 있음
- RSA의 공개키와 개인키 쌍을 생성하기 위해 두 개의 큰 소수 p와 q를 선택하고 그 둘의 곱 $N=pq$를 만들어냄
- 그다음에 $(p-1)(q-1)$에 상대적으로 소수인 e를 선택
- 마지막으로 e 모듈로 $(p-1)(q-1)$의 역곱수를 찾아내고 이 역을 d로 나타냄
- N을 얻게 되는데, 이는 $ed=1(\mathrm{mod}\ (p-1)(q-1))$을 만족시키는 e와 d뿐만 아니라 두 소수 p와 q의 결과물

4.3 RSA

■ RSA 키 생성 과정

1. 두 개의 큰 소수 p, q 선택
2. $N = p \times q$ 계산 -> 모듈러 기반
3. 오일러 피 함수 $\varphi(N) = (p-1)(q-1)$
4. 공개 지수 e 선택 $(1 < e < \phi(N), \gcd(e, \phi(N)) = 1$
5. 개인 지수 d 계산

$$e \quad \cdot d \equiv 1 \left(mod\ \varphi(N)\right)$$

6. 공개키: (*N*, *e*)　　　　　　개인키: *d*

■ RSA 키 쌍은 다음과 같이 구성

- 공개키: (*N*, *e*)　　　　　　　　　　개인키: *d*

(숫자 *N*은 계수, *e*는 암호화 지수, *d*는 복호화 지수)

4.3 RSA

■ RSA 키 생성 과정(예)

1. 두 개의 큰 소수 p, q 선택 / $p = 11, q = 17$
2. $N = p \times q$ 계산 -> 모듈러 기반 / $N = 11 \cdot 17 = 187$
3. 오일러 피 함수 $\varphi(N) = (p-1)(q-1) = (11 \; - 1)(17 - 1) = 160$
4. 공개 지수 e 선택 $(1 < e < \phi(N), \gcd\big(e, \phi(N)\big) = 1$
   선택 : $e = 7$ (확인 gcd(7,160) = 1)
   * gcd(Great Common Divisor) : 최대 공약수
5. 개인 지수 d 계산

   $$e \; \cdot d \equiv 1 \; \big(mod \; \varphi(N)\big) \; 7 \cdot d \equiv 1 \; (mod \; 160)$$
6. 공개키: (N, e) 개인키: d

4.3 RSA

■ RSA 키 생성 과정(예)

개인 지수 d 계산

$$e \cdot d \equiv 1 \left(mod\ \varphi(N)\right)\ 7 \cdot d \equiv 1\ (mod\ 160)$$

1. 유클리드 알고리즘(나눗셈)
   1. $160 = 7 \cdot 22 + 6$
   2. $7 = 6 \cdot 1 + 1$
   3. $6 = 1 \cdot 6 + 0$
2. 역으로 대입하여 1을 7과 160의 선형결합으로 표현
   1. $1 = 7 - 6 \cdot 1$
   2. $6 = 160 - 7 \cdot 22$, 따라서
   3. $1 = 7 - (160 - 7 \cdot 22) = 7 - 160 + 7 \cdot 22 = 7 \cdot 23 - 160 \cdot 1$
      즉, $1 = 23 \cdot 7 - 1 \cdot 160$ 이므로 $23 \cdot 7 \equiv 1(mod\ 160)$. 따라서

$$d = 23$$

공개키: (180, 7)　　　　개인키: 23

4.3 RSA

- RSA에서는 모듈로 지수를 통해 암호화와 복호화가 이루어짐
- RSA로 암호화하기 위해 평문 메시지 M을 숫자로 대하고 이를 권한 e, 모듈 N으로 올림

$$C = M^e \bmod N$$

- C 를 복호화하기 위해서는 복호화 지수 d를 사용한 모듈로 거듭제곱법으로 다음과 같이 해결

$$M = C^d \bmod N$$

- RSA가 정말로 작동하는가? $C = M^e \bmod N$이 주어지면 다음과 같음

$$M = C^d \bmod N = M^{ed} \bmod N \qquad (4.2)$$

  - 이를 위해서는 정수론으로부터 다음과 같은 표준 결과가 필요
  - **오일러 정리**: x가 n에 서로소이면, $x^{ø(n)} = 1 (\bmod\ n)$
  - e와 d : $ed = 1 (\bmod\ (p-1)(q-1))$
  - $N = pq$인데 이는 다음을 의미: $ø(N) = (p-1)(q-1)$
  - 두 인수는 다음을 의미: $ed - 1 = kø(N)$
- RSA가 작동한다는 것을 증명
  - $C^d = M^{ed} = M^{(ed-1)+1} = M \cdot M^{ed-1} = M \cdot M^{kø(N)} = M \cdot 1^k = M (\bmod) N \qquad (4.3)$

4.3 RSA

■ RSA 예시

- 예) 앨리스의 키 쌍을 생성하기 위해 두 개의 '큰' 소수 p=11, q=3을 선택
  - 계수 $N=pq=33$, $(p-1)(q-1)=20$
  - $(p-1)(q-1)$과 서로소가 되는 암호화 지수 e=3을 선택
  - 이에 대응하는 복호화 지수를 계산하는데 $ed=3\cdot 7=1(\mathrm{mod}\ 20)$이기 때문에 d=7

    앨리스의 공개키: $(N,e)=(33,3)$

    앨리스의 개인키: d=7
- 밥이 앨리스에게 메시지 M을 보내는데 메시지 M=15인 숫자라고 가정
  - 밥은 앨리스의 공개키 $(N,e)=(33,3)$을 보고 다음과 같이 암호문을 계산할 것임

    $$C=M^e(\mathrm{mod}\ N)=15^3=3375=9(\mathrm{mod}\ 33)$$

  - 이것을 앨리스에게 보내면 앨리스는 암호문 C=9를 복호화하기 위해 다음과 같이 개인키 d=7을 이용

    $$M=C^d(\mathrm{mod}\ N)=9^7=4{,}782{,}969=15(\mathrm{mod}\ 33)$$

  - 앨리스는 암호문 C=9로부터 원래 메시지가 M=15라는 것을 알아낼 수 있음

4.3 RSA – 제곱의 반복

■ 5^{20}을 계산하는 예

- 간단하게 5를 20번 곱해서 그 결과를 모듈로 35로 줄이면 다음과 같음

$$5^{20} = 95{,}367{,}431{,}640{,}625 = 25 \pmod{35} \qquad (4.4)$$

- RSA 암호화 $C=M^e \pmod{N}$ 또는 복호화 $M=C^d \pmod{N}$을 계산한다고 가정
- 안전하게 RSA를 실행시키면, 계수 N은 적어도 1025비트

- 지수 20은 이진수로 10100
- 지수 10100은 고차 비트에서 시작해 한 번에 한 비트씩 다음과 같이 만들어질 수 있음

$$(0, 1, 10, 101, 1010, 10100) = (0, 1, 2, 5, 10, 20)$$

- 지수 20은 다음과 같은 단계로 구성

$$\begin{aligned} 1 &= 0 \cdot 2 + ① \\ 2 &= 1 \cdot 2 + ⓪ \\ 5 &= 2 \cdot 2 + ① \\ 10 &= 5 \cdot 2 + ⓪ \\ 20 &= 10 \cdot 2 + ⓪ \end{aligned}$$

4.3 RSA – 제곱의 반복

- 제곱의 반복을 이용해 5^{20}을 계산하기 위해 이 알고리즘을 지수에 적용

$$
\begin{aligned}
5^{1} &= 1^{2} \cdot 5^{①} = 1^{2} \cdot 5 = 5 = 5 \pmod{35} \\
5^{2} &= 5^{2} \cdot 5^{⓪} = 5^{2} \cdot 1 = 25 = 25 \pmod{35} \\
5^{5} &= 25^{2} \cdot 5^{①} = 25^{2} \cdot 5 = 3125 = 10 \pmod{35} \\
5^{10} &= 10^{2} \cdot 5^{⓪} = 10^{2} \cdot 1 = 100 = 30 \pmod{35} \\
5^{20} &= 30^{2} \cdot 5^{⓪} = 30^{2} \cdot 1 = 900 = 25 \pmod{35}
\end{aligned}
$$

4.3 **RSA** – RSA 속도 증가

- RSA의 속도를 증가시키는 방법은 모든 사용자가 같은 암호화 지수 *e*를 사용하는 것
- 공통 암호화 지수를 위한 적정한 선택은 e=3
- 일반적으로 많은 암호화 연산은 중앙 서버(송신자)에서 이루어지고 복호화는 많은 클라이언트(수신자)에게 효율적으로 배분되어야 함

- e=3으로 세제곱근 공격이 가능
- 평문 M이 $M < N^{1/3}$을 만족한다면, $C = M^e = M^3$

- 많은 사용자가 e=3을 가지고 있다면 또 다른 종류의 세제곱근 공격이 존재
- 같은 메시지 M이 암호문 C_0, C_1, C_2를 만들어내고 세 명의 다른 사용자의 공개키로 암호화되어 있다면 중국인의 나머지 정리(chinese remainder theorem)를 사용하여 메시지 M을 복구할 수 있음

4.3 RSA – RSA 속도 증가

- 또 다른 대중적인 암호화 지수는 $e=2^{16}+1$
- 암호화 지수가 $e=3$일 때보다 $e=2^{16}+1$일 때 장점은 중국인의 나머지 정리 공격이 성공하기 위해 같은 암호화 메시지를 $2^{16}+1$ 사용자에게 보내야 한다는 것

4.3 RSA – 실습

■ RSA 기본 개념

- RSA는 대표적 **공개키 암호화** 알고리즘
  - 공개키(public key): 누구나 알 수 있음 -> 메시지를 암호화
  - 개인키(private key): 소유자만 알고 있음 -> 암호문을 복호화
- 핵심 원리: 소인수분해의 어려움
  - 큰 수 $n = p \times q$를 소인수분해하는 것은 매우 어려움
  - 이를 이용해 보안성을 확보

■ RSA 키 생성 절차

1. 소수 선택	p, q : 서로 다른 두 소수
2. 모듈러스 계산	$n = p \times q$
3. 오일러 피함수	$\varphi(n) = (p-1)(q-1)$
4. 공개 지수 선택	$1 < e < \varphi(n), \gcd\big(e, \varphi(n)\big) = 1$
5. 개인 지수 선택	$e \cdot d \equiv 1\ (mod\ \varphi(n))$, $d = e^{-1}\ (mod\ \varphi(n))$
6. 키 완성	공개키 (n, e) 개인키 (n, d)

4.3 RSA – 실습

■ RSA 기본 개념

- RSA는 대표적 **공개키 암호화** 알고리즘
  - 공개키(public key): 누구나 알 수 있음 -> 메시지를 암호화
  - 개인키(private key): 소유자만 알고 있음 -> 암호문을 복호화
- 핵심 원리: 소인수분해의 어려움
  - 큰 수 $n = p \times q$를 소인수분해하는 것은 매우 어려움
  - 이를 이용해 보안성을 확보

■ RSA 키 생성 절차

1. 소수 선택	p, q : 서로 다른 두 소수
2. 모듈러스 계산	$n = p \times q$
3. 오일러 피함수	$\varphi(n) = (p-1)(q-1)$
4. 공개 지수 선택	$1 < e < \varphi(n), \gcd\big(e, \varphi(n)\big) = 1$
5. 개인 지수 선택	$e \cdot d \equiv 1 \ (mod\ \varphi(n))$, $d = e^{-1}\ (mod\ \varphi(n))$
6. 키 완성	공개키 (n, e) 개인키 (n, d)

4.3 RSA – 실습

■ RSA 암 복호화

- 암호화
  - 평문 $m(0 \le m < n)$

$$c \equiv m^e \ (mod\ n)$$

- 복호화
  - 암호문 c

$$m \equiv c^d (mod\ n)$$

■ 확장 유클리드 알고리즘(EGCD)

1. 베주 항등식

$$g = \gcd(a, b) = ax + by$$

2. 모듈러 역원과의 관계
   * RSA에서 d는 e의 역원

$$e \cdot d \equiv 1 \ (mod\ \varphi(n))$$

4.3 RSA – 실습

■ 확장 유클리드 알고리즘(EGCD) 부연 설명

- 풀고자 하는 문제

$$\gcd(a, b) = a \cdot x + b \cdot y$$

*gcd 값 g, 계수(x, y)를구하는 게 목표

- 종료조건

만약 $b = 0$

$$\gcd(a, 0) = a$$

이때 식은

$$a \cdot 1 + 0 \cdot 0 = a$$

따라서 답은

$$(g, x, y) = (a, 1, 0)$$

```
if b == 0:
    return (a, 1, 0)
```

4.3 RSA – 실습

■ 확장 유클리드 알고리즘(EGCD) 부연 설명

- 문제 쪼개기
  - Gcd 성질

$$\gcd(a, b) = \gcd(b, a\ mod\ b)$$

    작은 문제 $egcd(b, a\%b)$를 먼저 풀자
    결과가 (g, x_1, y_1)라면

$$g = b \cdot x_1 + (a\ mod\ b) \cdot y_1$$

- 역추적
  - $a\ mod\ b = a\ - [a\ /\ \mathrm{c}\ ] \cdot b$
  - 따라서,

$$g = b \cdot x_1 + (a\ - (a//b) \cdot b) \cdot y_1$$
$$g = a \cdot y_1 + b \cdot (x_1 - (a//b) \cdot y_1$$

  - 즉, 새로운 해는

$$x = y_1, y = x_1 - (a//b) \cdot y_1$$

4.3 RSA – 실습

■ 예시 계산

1. $p = 17, q = 11$

$$n = 17 \times 11 = 187, \varphi(n) = 160$$

2. $e = 7$(선택) (gcd(7,160) = 1)

3. $EGCD$ 로 d 계산

$$7d \equiv 1\ (mod\ 160)$$

   $EGCD$ 과정

$$160 = 7 \times 22 + 6$$
$$7 = 6 \times 1 + 1$$

   역추적:

$$1 = 7 - 6 \times 1 = 7 - (160 - 7 \times 22)$$
$$= 7 \times 23 - 160 \times 1 \rightarrow d = 23$$

4. 키완성
   * 공개키 (187,7), 개인키(187,23)

5. 메시지 암 복호화
   * 평문 $m = 8$
   * 암호화 $c = 88^7 mod\ 187 = 11$
   * 복호화 $m = 11^{23} mod\ 187 = 88$

4.3 RSA – 실습

■ 실습

- 제공된 코드의 주석을 참고하여 RSA 암 복호화 코드를 완성하시오.