2024/07/25, 2024 보안 기초 세미나

암호학과 네트워크 보안

- 10장 비대칭 키 암호 -

김 혜 정(hyejeong@pel.sejong.ac.kr)

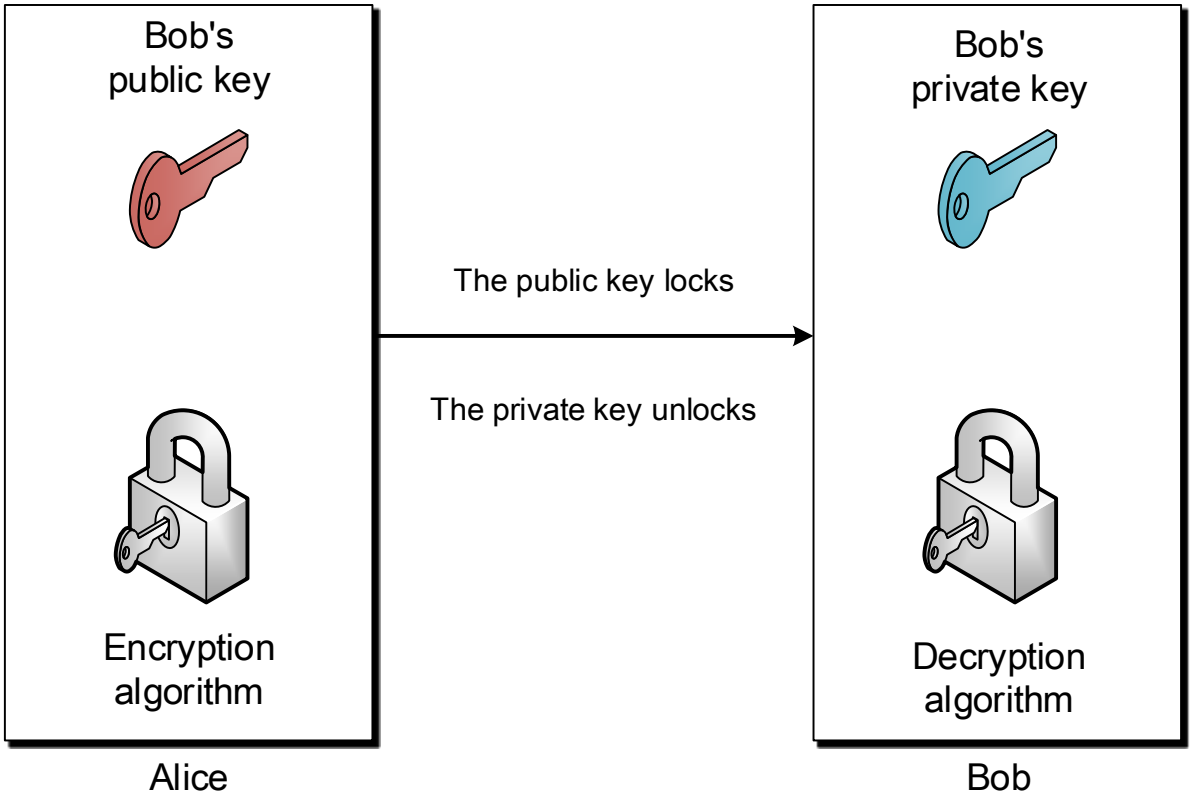
세종대학교 프로토콜공학연구실

목 차

목 차

비대칭 키

- 개요
  - 정의
    - 암복호화 시 두 개의 서로 다른 키를 사용하는 암호 방식으로 공개 키라고도 함



비대칭 키

- 대칭 키와 비대칭 키
  - 차이점
    - 비밀 개수
      - 대칭 키: $n(n-1)/2$
        - e.g., a, b, c 3명이 사용하는 경우, 대칭 키는 a-b, b-c, c-a 간 각각 다른 암호 키를 사용함(즉, ${}_nC_2 = n(n-1)/2$)
      - 비대칭 키: $2n$
        - e.g., a, b, c 3명이 사용하는 경우, 비대칭 키는 (공개 키, 개인 키)× n
    - 암호 방식
      - 대칭 키: 기호(문자나 비트)를 대체하거나 치환하는 것에 중점
        - e.g.,64비트의 평문을 대치, 전치를 시키는 DES(Data Encryption Standard)
      - 비대칭 키: 숫자를 다른 숫자로 변경하는 것에 중점
        - e.g., $n = p \times q$와 같이 하나의 복잡한 합성수나 소수를 쓰는 RSA(Rivest-Shamir-Adleman)

비대칭 키

- 대칭키와 비대칭키
  - 활용 예시 (1/2)
    - 대칭키 암호화 방식
      - 실시간 데이터 처리
        - 처리 속도가 빠르며 대량의 데이터를 암호화할 때 유리하여 대량의 파일이나 데이터 베이스 처리할 때 유리함
      - 컴퓨팅 자원 환경
        - 연산량이 다소 적어 모바일 기기, IoT 기기 등 제한된 컴퓨팅 자원을 활용하는 시스템에 적합함
    - 비대칭키 암호화 방식
      - 전자서명
        - 송신자의 개인키로 서명하면 송신자의 공개키로 검증할 수 있어 데이터 무결성 검증과 출처가 인증 가능함
      - 키 교환
        - SSL/TLS, IPsec 등 프로토콜에서 대칭키를 교환하는 데에 쓰임

비대칭 키

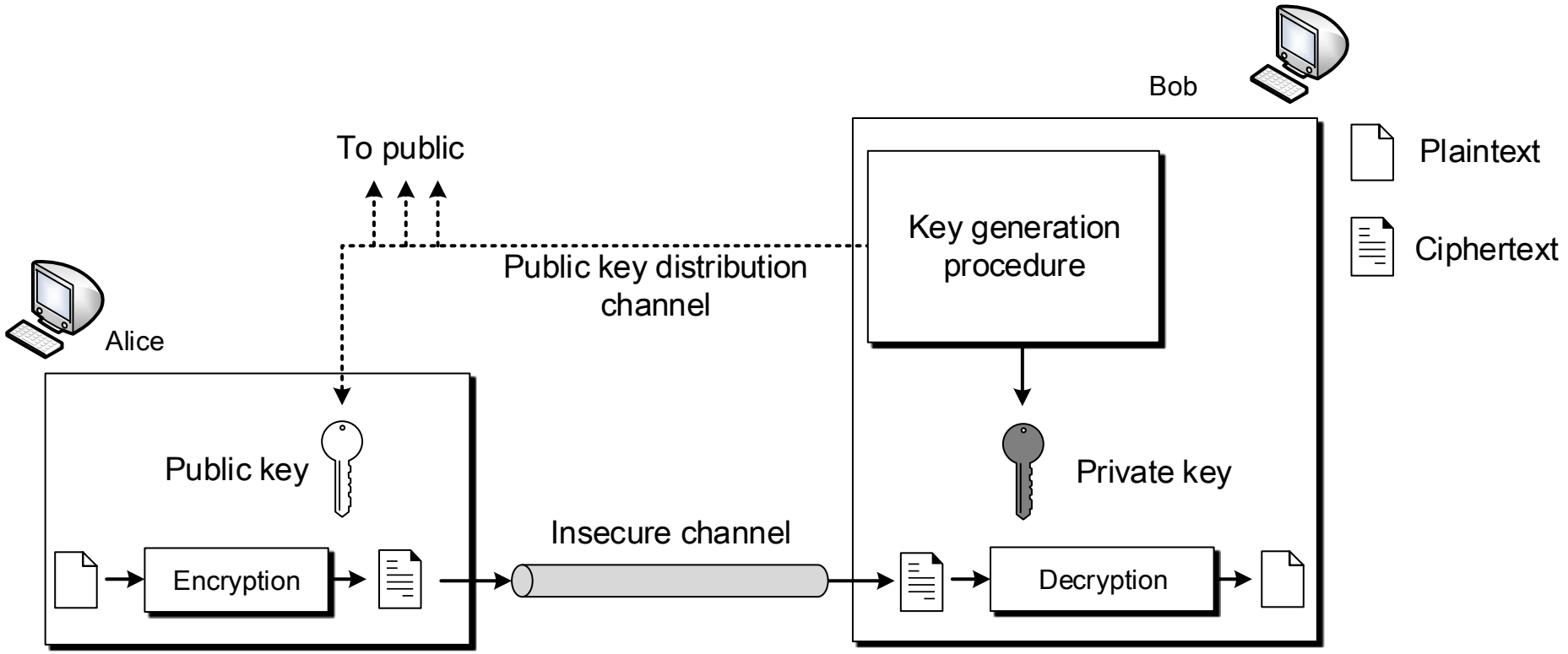
- 대칭키와 비대칭키
  - 활용 예시 (2/2)
    - 비대칭키
      - 공개키로 암호화하는 경우
        - 수신자의 공개키로 암호화함
        - 장점
          - 수신자만이 자신의 개인키로 복호화 할 수 있어 데이터 기밀성이 보장됨
        - 단점
          - 공개키를 사전에 알아야 하므로 데이터 무결성 검증과 출처 인증이 어려움
      - 개인키로 암호화하는 경우
        - 송신자의 개인키로 암호화함
        - 장점
          - 데이터 무결성 검증과 출처 인증이 가능
        - 단점
          - 송신자의 공개키를 알고 있다면 누구나 복호화 할 수 있어 비밀성 보장이 어려움

비대칭 키

- 대칭 키와 비대칭 키
  - 하이브리드 암호 시스템
    1. 비대칭 키 암호화
       - 데이터 전송을 시작하기 전에 비대칭 암호화를 통해 비밀 키 (대칭 키)를 수신자의 공개 키로 암호화함
    2. 대칭 키 암호화
       - 송신자는 대칭 키를 사용하여 실제 데이터를 암호화함
       - 암호화된 데이터와 대칭 키(비대칭 암호화로 암호화된)를 함께 수신자에게 전송함
    3. 비대칭 키 암호화
       - 수신자는 자신의 개인 키를 사용하여 대칭 키를 복호화함
       - 수신자는 복호화된 대칭 키를 사용하여 데이터를 복호화함
  - e.g., PGP (Pretty Good Privacy)

비대칭 키

- 비대칭 키의 특징 (1/2)
  - 수신자(Bob)가 책임을 지고 키 생성과 키 배분을 해야 함
    - 키 배분 채널은 무결성과 인증을 제공해야 함



비대칭 키

- 비대칭 키의 특징 (2/2)
  - 양방향 통신에서 동일한 키 쌍을 사용할 수 없음
    - 송신자(Alice)가 메시지 m을 수신자(Bob)에게 보내고, Bob이 메시지 m에 대한 응답을 하기 위해선 Alice가 자신의 키 쌍을 가지고 있어야 함
  - 대칭 키의 키 교환 문제를 해결함
    - 송신자는 공개 키(또는 개인 키)를 알고, 수신자는 개인 키(또는 공개 키)를 알면 되기 때문에 두 사람이 동일한 키를 공유하지 않음

비대칭 키

- 트랩도어 일방향 함수(TOWF, Trapdoor OWF)
  - 아래 성질 1, 2, 3을 만족하는 일방향 함수의 일종임
  - 일방향 함수(OWF, One-Way Function)
    - 성질 1: 주어진 x가 있다면 $y = f(x)$를 계산하는 것은 쉬움
    - 성질 2: 주어진 y가 있다면 $x = f^{-1}(x)$를 계산하는 것은 어려움
  - 트랩도어(Trapdoor)
    - “다락방 문”을 뜻하는 말로 시스템 보안이 제거된 비밀통로를 의미함
    - 성질 3: y와 비밀(트랩도어)가 주어지면 x를 쉽게 구할 수 있음

비대칭 키

- 트랩도어 일방향 함수(TOWF, Trapdoor OWF)
  - 예제 10.1

n이 매우 큰 수라면, $n = p \times q$는 일방향 함수이다

- 일방향 함수 $y = f(x)$에 적용해보자
- 소인수분해 문제에 기반하여 x는 두 개의 소수로 이루어진 순서쌍 (p, q)
- y는 n임

  - 예제 10.2

n이 매우 큰 수라면, 함수 $y = x^k \ mod \ n$은 하나의 트랩도어 일방향 함수이다

- 주어진 x, k, n이 있다면 y를 구하는 것은 쉬우나 y, k, n이 주어졌을 때는 x를 구하는 것이 어려움 (이산 로그 문제)
- $k' = 1 mod \ \varphi(n)$이 되는 k'를 알고 있다면 $x = y^{k'} \ mod \ n$을 이용하여 x를 구할 수 있음

목 차

- 비대칭 키
- 배낭 암호
- RSA
- Rabin
- ElGamal
- ECC

배낭 암호

- 개요
  - 배경
    - Merkle과 Hellman이 만든 최초 공개 키 암호시스템으로, 오늘날의 표준에는 안전하지 않지만 추후 공개 키 암호에 토대가 됨
  - 정의
    - 배낭 속에 있는 정해진 수들의 집합들의 원소를 알고 원소들의 합을 구하는 것은 쉽지만 합을 알고 각 원소들을 알기는 어렵다는 문제를 기반으로 한 암호

배낭 암호

- 동작
  - 키 구성
    - 2개의 k순서짝 $(k \in Z)$
      - $a = [a_1, a_2 \cdots a_k]$: 사전에 정의된 집합
      - $x = [x_1, x_2 \cdots x_k]$:0과 1로만 이루어진 집합으로, a가 배낭에 들어갈지를 결정
  - 함수
    - 배낭 속 원소들의 합 표현
      - $s = knapsackSum(a, x) = x_1 a_1 + x_2 a_2 + \cdots + x_k a_k$
      - $x = inv\_knapsackSum(s, a)$
    - a와 x를 알고 s를 구하는 것은 쉬우나 s와 a를 알고 x를 구하는 것은 어려움
      - 함수 $knapsackSum(a, x)$는 일방향함수임

배낭 암호

- 동작
  - 초증가 순서짝(Superincreaing tuple)
    - 집합 a가 있을 때, $a_i \geq a_1 + a_2 + \cdots + a_{i-1}$일 경우를 뜻함

knapsackSum($x[1 \cdots k]$, $a[1 \cdots k]$){

```
    s← 0
    for(i=1 to k){

        s ← s+a_i × x_i
    }
    return s
}
```

inv_knapsackSum(s, $a[1 \cdots k]$){

```
    for(i=k down to 1){

        if (s ≥ a_i){

            x_i ← 1      // x_i = 1이면 배낭 속에 존재한다는 의미
            s ← s−a_i //존재하므로 삭제
        }
        else x_i ← 0
    }
    return x[1 ⋯ k]
}
```

배낭 암호

- 동작
  - 초증가 순서짝(Superincreaing tuple)
    - 예제 10.3

$a = [17, 25, 46, 94, 201, 400]$이고 $s = 272$일 때, $knapsackSum$과 $inv_knapsackSum$를 계산하라.

i	a_i	s	$s \geq a_i$	x_i	$s \leftarrow s - a_i$
6	400	272	X	$x_6 = 0$	272
5	201	272	O	$x_5 = 1$	71
4	94	71	X	$x_4 = 0$	71
3	46	71	O	$x_3 = 1$	25
2	25	25	O	$x_2 = 1$	0
1	17	0	X	$x_1 = 0$	0

- 이 경우, $x = [011010]$이며, 배낭 속에는 $25, 46, 201$이 들어가있음을 의미함

배낭 암호

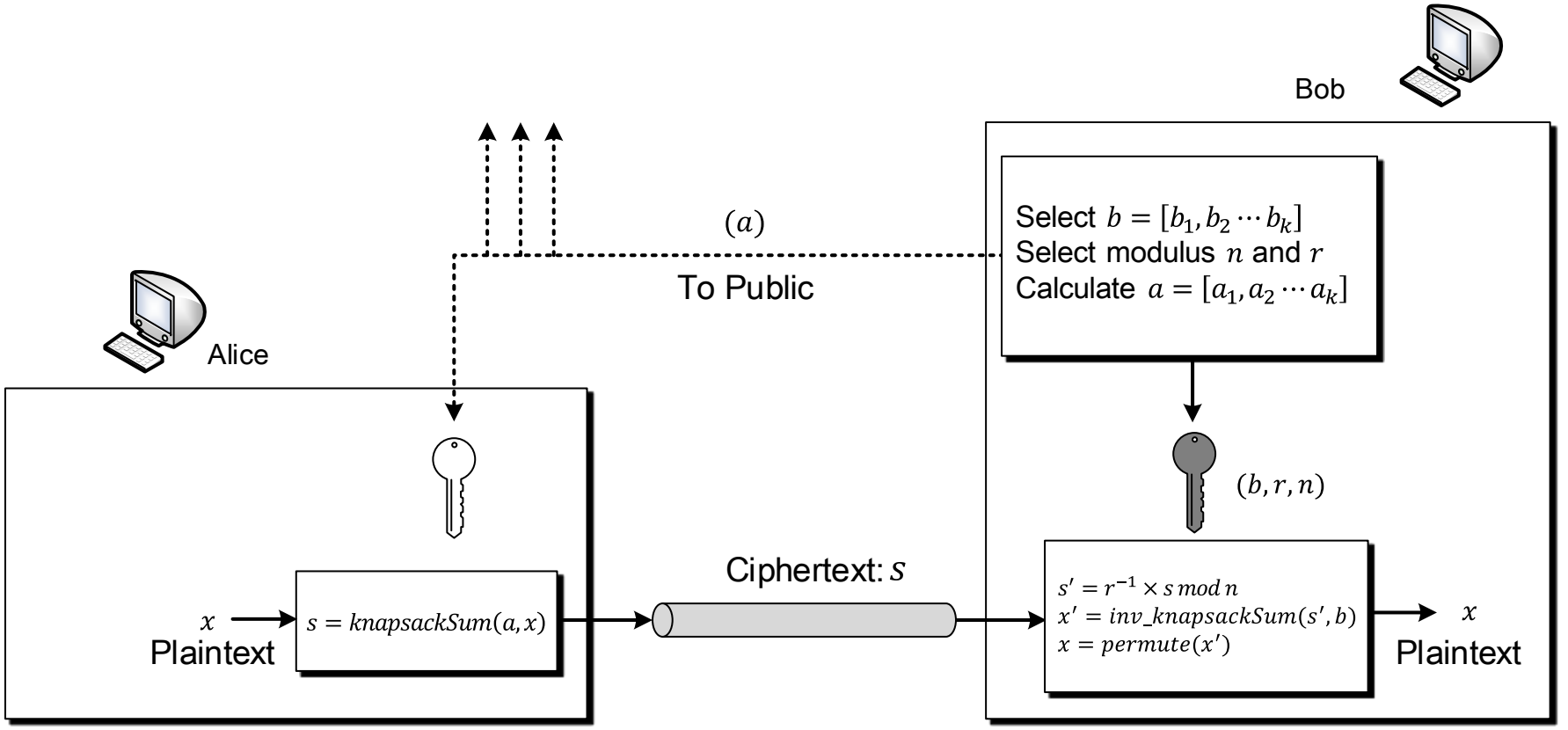
- 동작
  - 키 생성
    1. 초증가 k순서짝 $b = [b_1, b_2 \cdots b_k]$ 를 만듦
    2. $n > b_1 + b_2 + \cdots + b_k$인 모듈로 n 선정함
    3. n하고 서로소이면서 $1 \leq r \leq n-1$인 임의의 정수 r선택함
    4. $t_i = r \times b_i \, mod \, n$을 만족하는 임시 k순서짝 $t = [t_1, t_2 \cdots t_k]$ 생성함
    5. k개의 원소들과 매칭되는 새로운 순서짝 $a = permute(t)$를 구함
    6. k순서짝 a를 공개 키로 사용하고 n, r, b는 개인 키로 사용함

배낭 암호

- 동작
  - 암호화
    1. 메시지를 k순서짝 $x = [x_1, x_2 \cdots x_k]$로 변환함
       - x_i는 0 또는 1임
       - 순서짝 x는 평문임
    2. $knapsackSum$ 함수를 사용하여 s를 계산함
       - 원소 합 s는 암호문임
  - 복호화
    1. $s' = r^{-1} \times s \, mod \, n$ 계산함
    2. $inv_knapsackSum$을 사용하여 x'를 구함
    3. x'를 치환하여 x를 구함

배낭 암호

- 동작
  - Bob이 가지고 있는 트랩도어는 $s' = r^{-1} \times s \, mod \, n$와 초증가 순서짝 b임



배낭 암호

- 동작
  - 예제 10.4 (1/2)

절차를 보여주기 위한 간단한 예를 들어보자.

- 키 생성
  - Bob이 초증가 순서짝 $b = [7, 11, 19, 39, 79, 157, 313]$을 만듦
  - Bob이 모듈로로 $n = 900, r = 37$을 선택하고 치환표로는 $[4\ 2\ 5\ 3\ 1\ 7\ 6]$을 선택함
  - Bob은 순서짝 $t = [259, 407, 703, 543, 223, 409, 781]$을 계산함
    - $t_i = r \times b_i \, mod \, n$
  - Bob이 치환표를 가지고 $a = permute(t)$를 수행함
    - $t = [543, 407, 223, 703, 259, 781, 409]$
  - Bob이 a를 공개하고 n, r, b는 비밀로 함

배낭 암호

- 동작
  - 예제 10.4 (2/2)

절차를 보여주기 위한 간단한 예를 들어보자

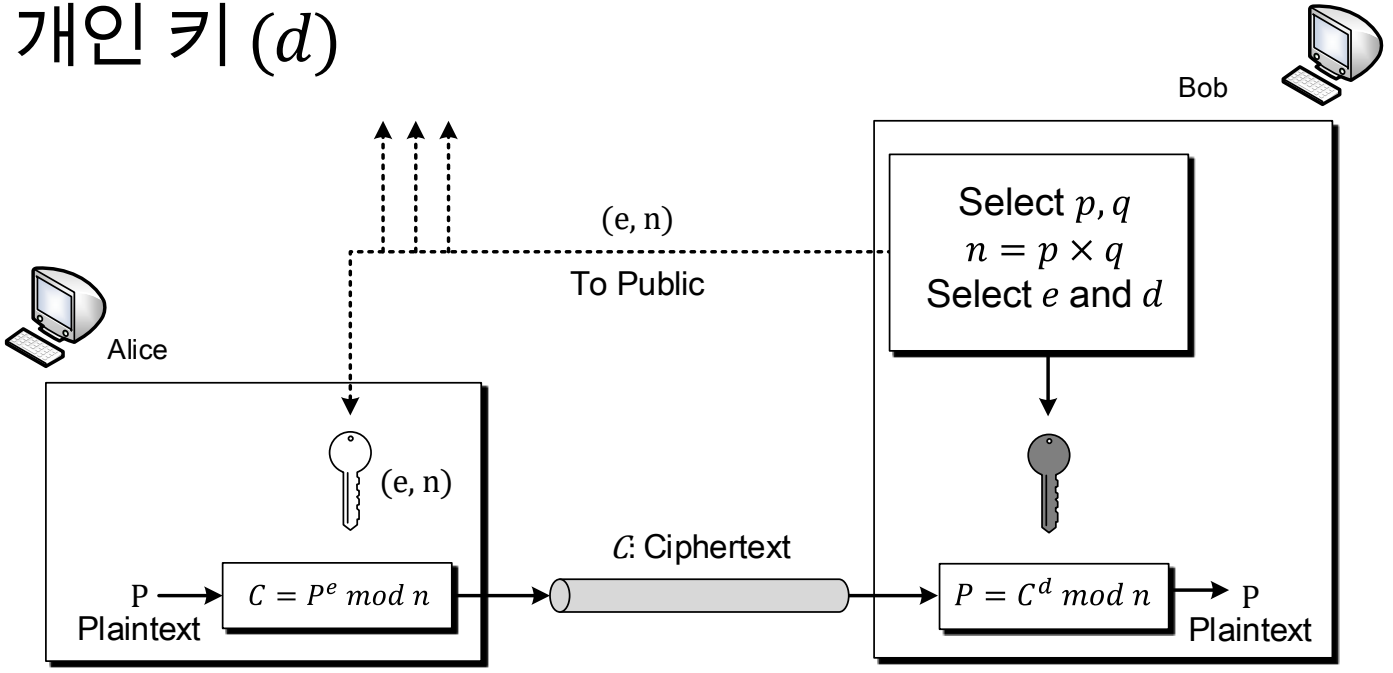
- Alice가 단 하나의 문자 g를 Bob에게 보낼 것임
  - 7비트 ASCII 코드를 이용해 g를 부호화함
    - g= $(1100111)_2$
  - 순서짝 $x = [1, 1, 0, 0, 1, 1, 1]$ 생성함
    - x가 평문임
  - S를 계산함
    - $543 \times 1 + 407 \times 1 + 223 \times 0 + 703 \times 0 + 259 \times 1 + 781 \times 1 + 409 \times 1 = 2399$
- Bob이 s를 사용하여 x를 복호화함
  - $s' = r^{-1} \times s \, mod \, n = (37^{-1} \times 2399) \bmod 900 = 527$
  - $x = inv\_knapsackSum(s', b) = [1, 1, 0, 1, 0, 1, 1]$ 를 계산함
  - $x = permute(x') = [1, 1, 0, 0, 1, 1, 1]$을 계산하여 x를 얻음
  - x를 ASCII 코드로 변환하면 g가 나옴

목 차

- 비대칭 키
- 배낭 암호
- RSA
- Rabin
- ElGamal
- ECC

RSA

- 개요
  - 정의
    - 3명의 개발자(Rivest, Shamir, Adleman)의 성을 따온 이름으로 두 개의 큰 소수로 생성된 키 쌍을 사용한 공개 키 암호 시스템
  - 키 구성
    - 공개 키 (e, n), 개인 키 (d)

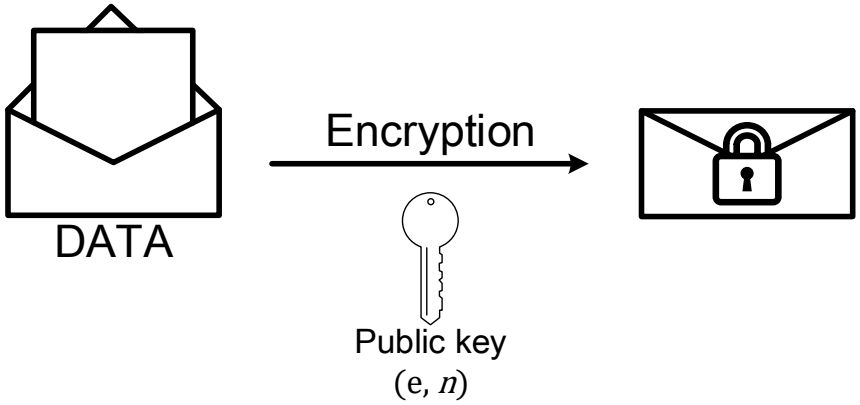


RSA

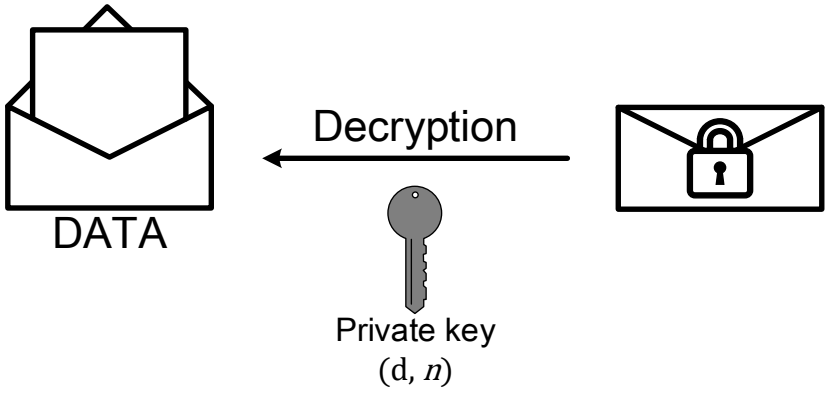
- 동작
  - 키 생성
    - 곱셈군 $G =< Z_{\varphi(n)}{}^{*}, \times >$ 사용
      - 합성수 $n = p \times q$
      - 이때, $\varphi(n), p, q$는 비밀임
    - 과정
      1. 1024bit 이상인 p와 q를 선택하여 n 생성
      2. $\varphi(n)$과 서로소인 정수 e 선택 $(1 < e < \varphi(\mathrm{n}))$
      3. $ed \bmod \varphi(\mathrm{n}) = 1 \left(1 < d < \varphi(\mathrm{n})\right)$
         - e의 곱셈의 역원인 d

RSA

- 동작
  - 암호화
    - $\mathrm{C} = \mathrm{P}^{e} \; mod \; n$
      - 메시지 P가 n보다 큰 경우 작은 블록으로 나누어 줌



  - 복호화
    - $\mathrm{P} = \mathrm{C}^{d} \; mod \; n$
      - 메시지 P가 나누어진 경우 해당 변환방식이 공유되어야 함



RSA

- 동작
  - 암·복호화 역관계 증명
    - 오일러 정리 활용
      - 만약 $n = p \times q,\ a < n$ 이고 k는 정수라면, $a^{k \times \varphi(n)+1} \equiv a\ (mod\ n)$이 성립된다.
        - $ed\ mod\ \varphi(n) = 1$
        - 즉, $ed = k \times \varphi(\mathrm{n}) + 1$ 으로 표현 가능
    - Bob이 복호화한 평문을 P_1이라고 하고, 이것이 P와 같음
      - $\mathrm{P}_1 = \mathrm{C}^d\ mod\ n = (\mathrm{P}^e\ mod\ n)^{\ d}\ mod\ n = \ \mathrm{P}^{ed}\ mod\ n$
      - $\mathrm{P}_1 = \mathrm{P}^{ed}\ mod\ n \rightarrow \mathrm{P}_1 = \mathrm{P}^{k \times \varphi(n)+1}\ mod\ n = \mathrm{P}\ mod\ n$

RSA

- 동작
  - 예제 10.5

주어진 수를 통해 평문과 복호화를 거친 암호문이 같은지 살펴보자.

- Bob이 $p = 7, q = 11$을 선택했다고 가정하자
  $n = 7 \times 11 = 77, \varphi(77) = 6 \times 10 = 60$
- ${Z_{60}}^*$에서 e와 d를 선정할 수 있다. e를 13으로 선택했다면 d는 37이 됨
  - $ed \; mod \; \varphi(n) = 13d \; mod \; 60 = 1$
  - 유클리드 확장 알고리즘을 사용해서 $1 = -23 \cdot 13 + 5 \cdot 60 \therefore -23$
  - 양수로 전환하면 $60 - 23 = 37 \;\; \therefore d = 37$

- Alice가 평문 5를 Bob에게 보낸다고 가정하자
- $\mathrm{P}: 5, \quad \mathrm{C}: 5^{13} = 26 \; mod \; 77$
- Bob은 자신의 개인 키(d)를 사용하여 복호화함
- $\mathrm{C}: \; 26 \quad P: \; 26^{37} = 5 \; mod \; 77$

RSA

- 동작
  - 예제 10.7

Jennifer는 자신의 키 쌍을 만들기 위해 $p = 397, q = 401$을 선택하고 $e = 343$과 $d = 12007$을 선택했다. 만약에 Ted 가 e와 n을 알면 메시지를 Jennifer 에게 보낼 수 있다는 것을 보이시오.

- Jennifer에게 보내고자 하는 메시지가 "NO"라고 가정하자
- Ted는 각 문자를 숫자로 된 코드(00부터 25까지)로 바꿈
  N→13, O→14
- 바꾼 문자코드를 이어붙어 4자리 숫자를 만듦
  NO→1314
- e와 n을 사용하여 암호화함

$$\text{C: } 1314^{343} = 33677 \; mod \; 159197$$

- Jennifer는 33677을 받게 될 것이고 이를 개인 키 d를 사용하여 복호화함

$$\text{P: } 33677^{12007} = 1314 \; mod \; 159197$$

- Jennifer는 이를 다시 문자로 변환하여 "NO"를 얻음

RSA

- 응용
  - 데이터 암호화
    - 전자메일
      - e.g., PGP(Pretty Good Privacy)
    - 파일 암호화
      - 파일을 암호화하여 저장하거나 전송할 때 사용됨
  - 디지털 서명
    - 소프트웨어 배포
      - 소프트웨어에 디지털 서명을 추가하여 배포 시 사용
  - 인증
    - 웹 사이트 인증
      - SSL/TLS 프로토콜의 일부로 사용되어 웹 사이트의 인증을 제공함
    - 사용자 인증
      - 사용자 비밀번호를 암호화하거나 두 단계 인증 시 사용함

RSA

- 공격
  - RSA에 대한 치명적인 공격은 아직 알려진 것이 없으나, 문제가 있는 평문이나 약한 매개변수 선정 혹은 적합하지 않은 암호적용 등에 대한 공격 방법이 존재함
  - 종류
    - 소인수분해 공격
    - 선택 암호문 공격
    - 암호화 지수에 대한 공격
    - 평문 공격
    - 일반 모듈로 공격
    - 타이밍 공격

RSA

- 공격
  - 소인수분해 공격
    - RSA의 안전성은 소인수분해가 어렵다는 것에 근거하기 때문에 n값은 공개하더라도 p와 q는 비밀로 유지함
    - 공격자가 p와 q를 알아냈을 경우, $\varphi(n)$를 계산할 수 있게 되고 e가 공개되어 있기 때문에 d까지 계산할 수 있음
    - 이를 고려하여 n을 300자리 이상의 십진수(대략 1024비트)를 사용해야하며, 최근에는 4096비트까지 사용할 것을 권장

RSA

- 공격
  - 선택 암호문 공격
    - RSA의 곱셈성질을 이용한 공격으로 Bob이 Alice가 보낸 암호문이 아닌 공격자 Eve가 보낸 암호문을 복호화하는 공격
      - Eve가 Alice가 보낸 C를 가로챈 다음, 아래와 같은 절차를 거침
        1. Eve는 $Z_n{}^*$에 속하는 임의의 정수 X 선택함
        2. Eve는 $Y = \mathrm{C} \times X^e \ mod\ n$를 계산함
        3. Eve는 Y를 Bob에게 보내어 복호화를 부탁하여 $Z = Y^d \ mod\ n$를 얻음
        4. Eve는 다음과 같이 해서 쉽게 P를 구함
           - $Z = Y^d \ mod\ n = (\mathrm{C} \times X^e)^d \ mod\ n = \left(\mathrm{C}^d \times X^{ed}\right) mod\ n$
             $= \left(\mathrm{C}^d \times X\right) mod\ n = (\mathrm{P} \times X) mod\ n$
           - $Z = (\mathrm{P} \times X) mod\ n \rightarrow \mathrm{P} = (Z \times X^{-1}) mod\ n$

RSA

- 공격
  - 암호화 지수에 대한 공격 (1/5)
    - 암호화 지수 e가 작을 때 발생하는 취약점을 노린 공격
      - e.g., $e = 3$
        - 과거에 많이 사용되었지만, 패딩이 불완전할 경우 보안 취약점이 있을 수 있음
    - Coppersmith 정리 공격
      - 모듈로가 n이고 차수가 e인 다항식 $f(x)$에서 만약 한 개의 근이 $n^{1/e}$보다 작으면 복잡도가 $\log n$인 알고리즘을 사용하여 근을 구할 수 있다
      - 즉, 작은 e와 그보다 더 작은 P일 경우를 말함
        - $\mathrm{C} = \mathrm{P}^e \ mod \ n$에서 P^e 가 n보다 작으면 모듈로 연산이 필요 없음
        - e.g., $e = 3, n = 1000, \mathrm{P} = 5$
          - $\mathrm{C} = 5^3 \ mod \ 1000 = 125 \ mod \ 1000$

RSA

- 공격
  - 암호화 지수에 대한 공격 (2/5)
    - 브로드캐스트 공격 (1/2)
      - 다수의 수신자가 같은 메시지를 서로 다른 공개 키로 암호화하는데 이때, 동일한 작은 지수 e를 사용할 경우에 가능한 공격
        - e.g., Alice가 한 메시지 P를 3명의 수신자에게 보내는데 $e = 3$을 사용하고, 모듈로 값은 각각 n_1, n_2, n_3를 사용한다고 가정함
          - $\mathrm{C}_1 = \mathrm{P}^3 \, mod \, n_1 \quad \mathrm{C}_2 = \mathrm{P}^3 \, mod \, n_2 \quad \mathrm{C}_3 = \mathrm{P}^3 \, mod \, n_3$
          - CRT를 사용하여 P^3을 모듈로 $n_1 \times n_2 \times n_3$에 대해 재구성함
          - 재구성된 P^3은 $n_1 \times n_2 \times n_3$보다 작거나 같으므로, $\mathrm{C} = \mathrm{P}^3$이 모듈로 상의 항등이 아니라 일반 산술적인 의미에서 항등이 될 수 있음

RSA

- 공격
  - 암호화 지수에 대한 공격 (3/5)
    - 브로드캐스트 공격 (2/2)
      - 예제 10.8
        - Prob)
          - $e = 3, \mathrm{P} = 42$
          - $n_1 = 61 \times 53 = 3233$
          - $n_2 = 47 \times 59 = 2773$
          - $n_3 = 43 \times 71 = 3053$
        - Sol)
          - 공격자가 $\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3$와 각 n_i를 알고 있기 때문에 다음을 만족하는 P^3을 찾으면 됨
            - $\mathrm{P}^3 \equiv C_1 \,(mod\, n_1), \mathrm{P}^3 \equiv C_2 \,(mod\, n_2), \mathrm{P}^3 \equiv C_3 \,(mod\, n_3)$
          - $\mathrm{C}_1 = 42^3 \, mod \, 3233 = 74088 \, mod \, 3233 = 1392$
          - $\mathrm{C}_2 = 42^3 \, mod \, 2773 = 74088 \, mod \, 2773 = 1712$
          - $\mathrm{C}_3 = 42^3 \, mod \, 3053 = 74088 \, mod \, 3053 = 1986$

RSA

- 공격
  - 암호화 지수에 대한 공격(4/5)
    - 관련된 메시지 공격(Franklin-Reiter Related Message Attack)
      - Franklin-Reiter가 발견한 방법으로, RSA 암호화된 메시지들을 선형 관계를 이용해 특정 값을 빠르게 계산하는 방법
        - 암호화 메시지에 패딩을 추가하는 과정, 공통적인 암호 키를 사용할 때 메시지 간 선형 관계가 생성될 수 있음
        - RFID 태그, 스마트 카드 등 동일한 암호화를 사용하는 경우 이 공격에 대한 가능성을 고려하여 구현되어야 함
        - e.g., 암호화 지수 $e = 3$, 공격자가 $c1$과 $c2$를 가로챘다고 가정
          - $c1 \equiv m1^3 mod\, N$
          - $c2 \equiv (m1 + p)^3 mod\, N$ (이때 p는 공격자가 알고 있는 임의의 패딩)
          - 공통근 m을 구하기 위해서는 $\gcd(m1^3 - c1, (1 + p)^3 - c2)$ 을 구해야 함

출처: Yacobi, Oded, and Yacov Yacobi, "A new related message attack on RSA.“, *Public Key Cryptography-PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Proceedings 8.*, 2005.

RSA

- RSA-공격
  - 암호화 지수에 대한 공격 (5/5)
    - 짧은 패드 공격(Coppersmith's Short Pad Attack)
      - 메시지에 패딩을 추가할 때, 랜덤 비트를 추가하는 방법을 적용하거나 짧은 패딩을 적용할 경우 발생하는 공격
        - e.g., 사용자가 메시지 m을 암호화한다고 가정
          - 메시지 m에 짧은 패딩 p를 추가하여 암호화할 메시지 c를 생성
            - $c = (m + p)^e mod\, n$
          - 공격자가 암호문 c와 공개키 e를 알고 있다면, 가능한 모든 메시지 m'을 시도하면서 $c = (m + p)^e mod\, n$를 만족시키는 m'을 찾음

RSA

- 공격
  - 평문 공격 (1/2)
    - 짧은 메시지 공격
      - 공격자가 이미 많은 평문을 알고 있을 때, 가로챈 암호문과 동일한 암호문이 나올 때까지 자신이 가지고 있는 평문을 다 암호화하는 공격
        - e.g., Alice가 4자리 숫자를 Bob에게 보낸다면, Eve는 0000부터 9999까지 다 암호화해보는 것을 의미함
    - 감추어지지 않는 공격
      - 암호화를 했음에도 도로 자기 자신이 되어 감추어지지 않는다는 메시지에 대한 공격
        - e.g., 암호화된 메시지가 원래의 평문과 유사한 구조를 가지는 경우

RSA

- 공격
  - 평문 공격 (2/2)
    - 순환 공격
      - 암호문이 평문의 치환이라는 것과 암호문을 연속해서 암호화하면 궁극적으로 평문에 도달하게 된다는 사실에 기초한 공격
        - 똑같은 암호문이 나올 때까지 반복함
        - e.g., 가로챈 암호문: $C,\ P = C_{k-1}$

$$C_1 = C^e \ mod\ n$$
$$C_2 = {C_1}^e \ mod\ n$$
$$\vdots$$
$${C_k} = {C_{k-1}}^e \ mod\ n \rightarrow C_k = C\text{라면, 여기서 멈춤}$$

RSA

- 공격
  - 일반 모듈로 공격
    - 여러 개의 암호문이 동일한 모듈로를 사용할 경우, 공격자는 이 암호문들 간의 관계를 활용하여 원래의 평문을 복구하려는 공격
      - e.g., 동일한 모듈로와 평문, 작은 지수 e_i를 사용할 경우
        - $\mathrm{C}_1 = \mathrm{M}^{e_1} \; mod \; n \qquad \mathrm{C}_2 = \mathrm{M}^{e_2} \; mod \; n$
        - ${\mathrm{C}_1}^{e_2} \equiv \mathrm{M}^{e_1 \cdot e_2} \; mod \; n \qquad {\mathrm{C}_2}^{e_1} = \mathrm{M}^{e_1 \cdot e_2} \; mod \; n$
        - $\frac{{\mathrm{C}_1}^{e_2}}{{\mathrm{C}_2}^{e_1}} \equiv \mathrm{M}^{e_1 \cdot e_2 - e_2 \cdot e_1} \; mod \; n$
        - $\frac{{\mathrm{C}_1}^{e_2}}{{\mathrm{C}_2}^{e_1}} \equiv \mathrm{M}^{0} \; mod \; n$
        - $\frac{{\mathrm{C}_1}^{e_2}}{{\mathrm{C}_2}^{e_1}} \equiv 1 \; mod \; n$
        - 이후, 공통근을 찾음

RSA

- 공격
  - 타이밍 공격 (1/2)
    - 암호화 시스템에서 특정 연산을 수행할 때 걸리는 시간을 변화를 관찰하고 분석하며, 시스템의 비밀정보를 추출하거나 암호화된 데이터를 해독하는 공격
      - e.g., 공격자 Eve가 암호문 C_1부터 C_m까지 가로챘다고 가정하자
        - Eve는 Bob이 각 암호문을 복호화하는 과정을 지켜보면서 해독 시간을 기록 (T_1부터 T_m)
        - Eve는 하드웨어에서 곱셈을 하는 데 걸리는 시간을 알고있고 이를 기록 (t_1부터 t_m)
          - 곱셈연산은 $Result = Result \times C_i \; mod \; n$을 수행하는 데 걸리는 시간

RSA

- 공격
  - 타이밍 공격 (2/2)
    - 코드 설명
      - 초기화 단계
      - 비트 추정 단계
        - 비밀 키 비트 크기 k만큼 반복문을 돌림
        - $[D_1 \cdots D_m] \leftarrow [T_1 \cdots T_m] - [t_1 \cdots t_m]$
          - 새로 계산된 시간에서 곱셈 시간을 빼서 d_j에 예상되는 시간 차이를 구함
        - var←variance($[D_1 \cdots D_m]$)−($[T_1 \cdots T_m]$)
          - 시간 차이의 분산을 계산
        - if(var>0) $d_j \leftarrow 1$
          - 분산 값이 양수이면, d_j를 1로 설정
        - $[T_1 \cdots T_m] \leftarrow [T_1 \cdots T_m] - d_j \times [t_1 \cdots t_m]$
          - d_j에 해당하는 시간 차이 조정

RSA_Timing_Attack($[T_1 \cdots T_m]$){

$d_0 \leftarrow 1$
Calculate$[t_1 \cdots t_m]$
$[T_1 \cdots T_m] \leftarrow [T_1 \cdots T_m] - [t_1 \cdots t_m]$
for(j from 1 to k-1)
{
Recalculate$[T_1 \cdots T_m]$
$[D_1 \cdots D_m] \leftarrow [T_1 \cdots T_m] - [t_1 \cdots t_m]$
var←variance($[D_1 \cdots D_m]$)−($[T_1 \cdots T_m]$)
if(var>0) $d_j \leftarrow 1$
$[T_1 \cdots T_m] \leftarrow [T_1 \cdots T_m] - d_j \times [t_1 \cdots t_m]$
}
}

RSA

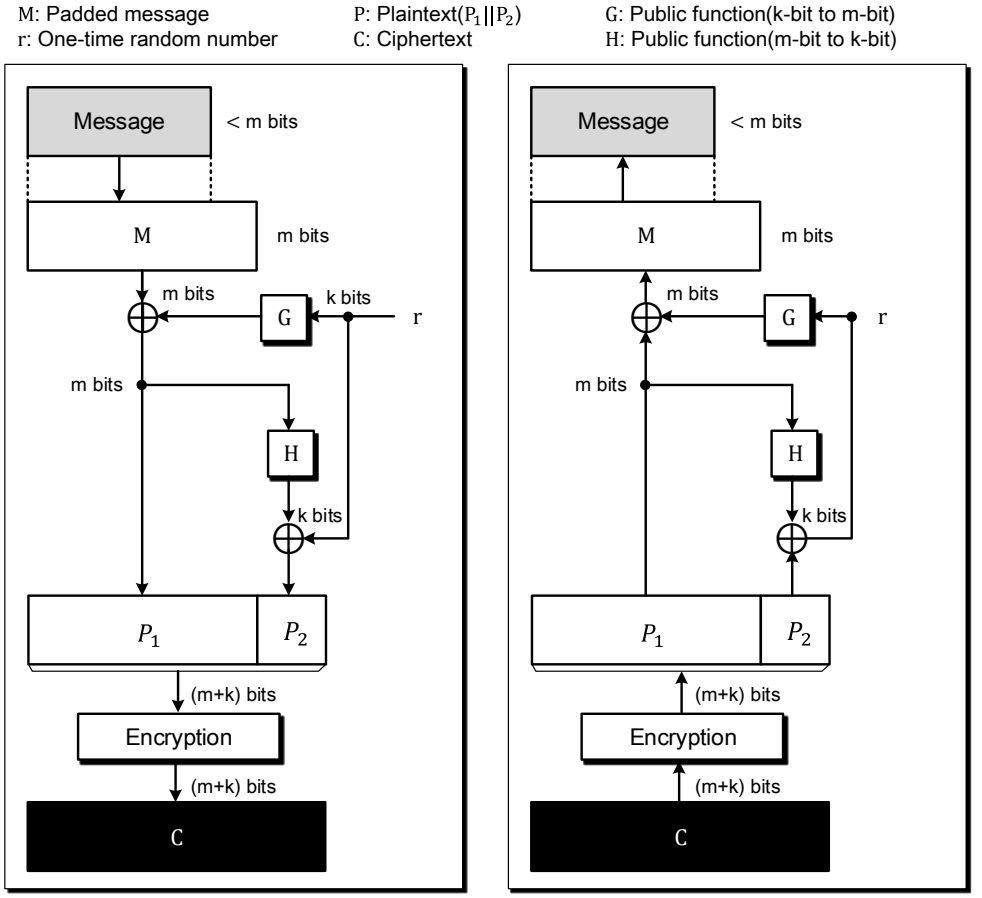
- 공격
  - 타이밍 공격 방지
    - 지수 계산을 할 때 무작위적으로 지연을 추가
      - 알고리즘의 실행 시간이 입력 데이터나 비밀 키의 비트에 따라 달라지도록 않게 함
      - e.g., 복호화 연산 중 특정 조건이 충족될 때, 1ms에서 5ms 사이의 무작위 지연을 추가
    - 암호문을 복호화하기 전에 난수로 곱하는 블라인딩 기법 추가
      1. 비밀로 된 난수 r을 1과 n-1 사이에서 선택
      2. $\mathrm{C}_1 = \mathrm{C}^e \times r \; mod \; n$을 계산
      3. $\mathrm{P}_1 = {\mathrm{C}_1}^d \; mod \; n$을 계산
      4. $\mathrm{P} = \mathrm{P}_1 \times r^{-1} \; mod \; n$을 계산

RSA

- 공격
  - 공격 방지를 위한 권장사항
    - n의 비트 수는 적어도 2048비트. 즉, n은 2^{2048} 근방에 있는 수로 선정
    - p와 q는 서로 너무 가까이 있는 수가 되어서는 안 됨
    - $p-1$과 $q-1$은 각각 적어도 하나의 큰 소인수를 가져야 함
    - p/q는 작은 분자나 분모를 갖는 유리수와 가까이 있어서는 안 됨
    - 모듈로 n을 공동으로 사용해서는 안 됨
    - e는 $2^{16}+1$이거나 이 값에 가까운 정수이어야 함
    - 만약 d가 침해당했다면 n, p, q, e와 d 모두 바꾸어야 함
    - 메시지는 OAEP를 사용해서 패딩 되어야 함

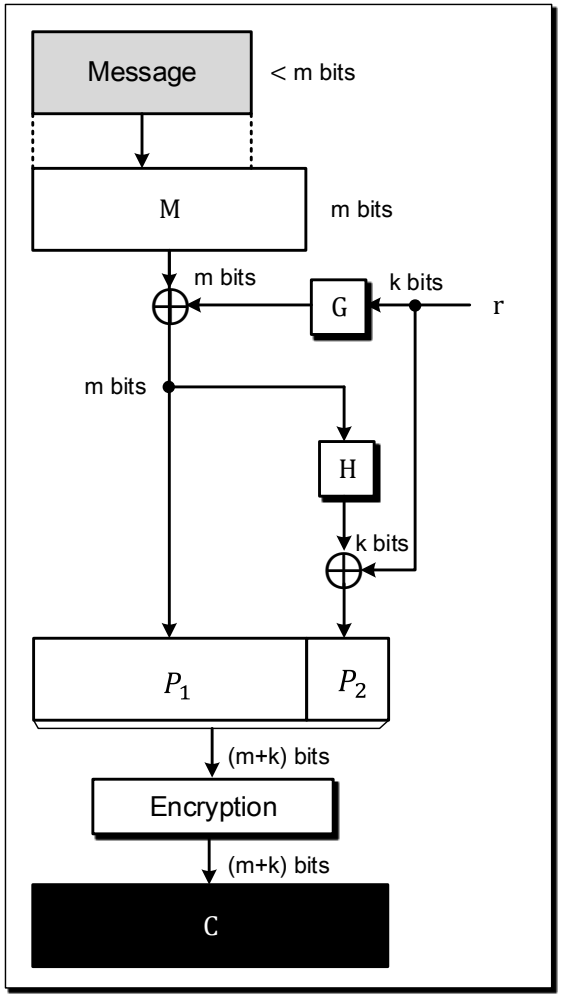
RSA

- 공격
  - 최적 비대칭 암호 패딩(OAEP, Optimal Asymmetric Encryption Padding)
    - 암호 시스템에서 사용하는 패딩 방식으로, 메시지를 암호화하기 전에 추가적인 정보를 넣는 방법



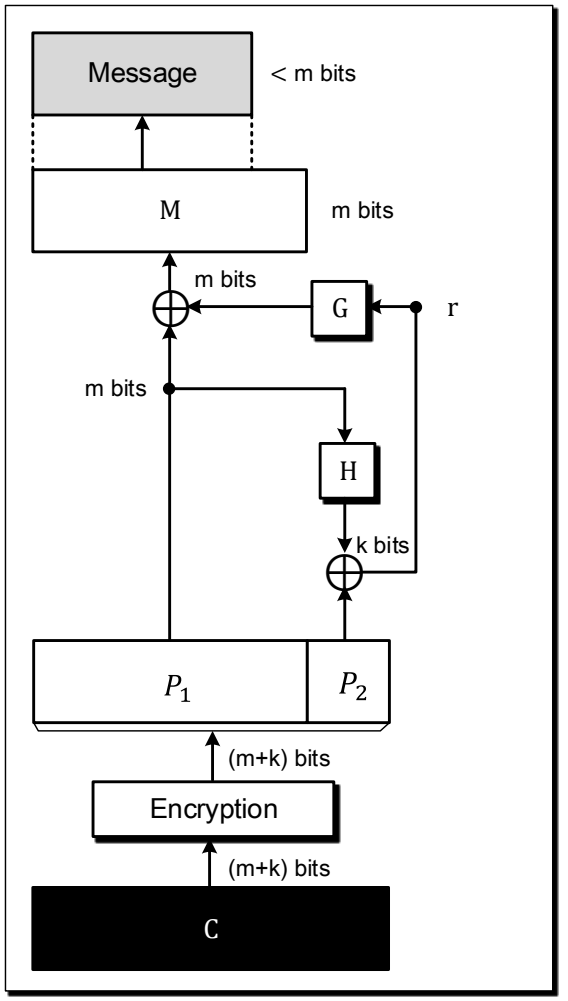
RSA

- 공격
  - 최적 비대칭 암호 패딩(OAEP, Optimal Asymmetric Encryption Padding)
    - 암호화 절차
      - $\mathrm{P} = \mathrm{P}_1 || \mathrm{P}_2$
      1. m비트 메시지가 되도록 패딩을 붙인 것을 M이라고 부름
      2. k비트를 갖는 난수 r을 선택함
      3. r비트 정수 입력으로 m비트 정수를 출력하는 공개된 일방향 함수 $\mathrm{G}(r)$ 사용
         - 평문의 첫 번째 부분인 $\mathrm{P}_1 = \mathrm{M} \oplus \mathrm{G(r)}$ 생성
      4. m비트를 입력하여 k비트를 내보내는 공개된 함수 $H(P_1)$ 를 사용
         - 평문의 두 번째 부분인 $\mathrm{P}_2 = \mathrm{H}(\mathrm{P}_1) \oplus r$ 생성
      5. Alice는 $\mathrm{C} = \mathrm{P}^e = (\mathrm{P}_1 || \mathrm{P}_2)^e$을 전송



RSA

- 공격
  - 최적 비대칭 암호 패딩(OAEP, Optimal Asymmetric Encryption Padding)
    - 복호화 절차
      1. $\mathrm{P} = \mathrm{C}^{d} = (\mathrm{P}_1 || \mathrm{P}_2)$
      2. $\mathrm{H}(\mathrm{P}_1) \oplus \mathrm{P}_2 = \mathrm{H}(\mathrm{P}_1) \oplus \mathrm{H}(\mathrm{P}_1) \oplus r = r$을 이용하여 r을 재생성
      3. 패딩된 메시지 값을 재생성하기 위해 $\mathrm{G}(r) \oplus \mathrm{P} = \mathrm{G}(r) \oplus \mathrm{G}(r) \oplus \mathrm{M} = \mathrm{M}$을 이용
      4. M으로 부터 패딩을 제거하고 원래의 메시지를 구함



목 차

- 비대칭 키
- 배낭 암호
- RSA
- Rabin
- ElGamal
- ECC

Rabin

- 개요
  - 정의
  - RSA 변형으로 평문의 제곱을 암호문으로 생성하는 공개키 암호 시스템

  - 키 구성
    - 공개 키 (n), 개인 키(p, q)
    - RSA의 $e = 2, d = \frac{1}{2}$로 고정임

Rabin

- 동작
  - 키 생성
    - $k \in Z, 4k+3$인 서로 다른 두 소수 p와 q를 선택
      - $4k+1$을 가질 수 있지만, 이 경우엔 복호화 과정이 어려워짐
    - $n = p \times q$

Rabin_key_Generation{

Choose two large primes p and q in the form 4k+3 and $p \neq q$.
$n \leftarrow p \times q$
Public_key $\leftarrow n$
Private_key $\leftarrow (q, p)$
return Public_key and Private_key

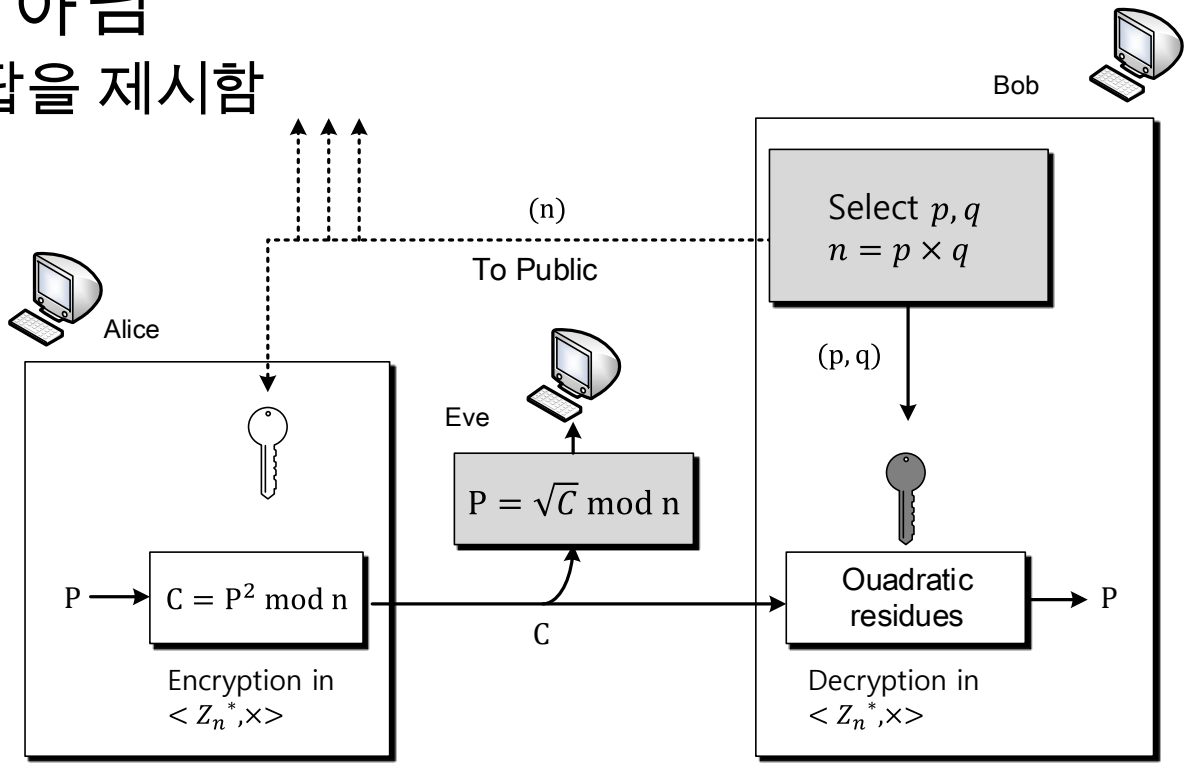
}

Rabin

- 동작
  - 암호화
    - $\mathrm{C} = \mathrm{P}^2 \ mod \ n$
  - 복호화
    1. $P_p = \sqrt{C} \ mod \ p, P_q = \sqrt{C} \ mod \ q$
       - $p \cdot y_p + q \cdot y_q = 1$을 만족하는 y_p와 y_q 구하기
       - $a_1 \equiv \mathrm{C}^{\frac{p+1}{4}} \ mod \ p, \quad a_2 \equiv -\mathrm{C}^{\frac{p+1}{4}} \ mod \ p$
       - $b_1 \equiv \mathrm{C}^{\frac{q+1}{4}} \ mod \ q, \quad b_2 \equiv -\mathrm{C}^{\frac{q+1}{4}} \ mod \ q$
    2. CRT를 이용
       - $\mathrm{P}_1 = \mathrm{CRT}(a_1, b_1, p, q)$, $\mathrm{P}_2 = \mathrm{CRT}(a_1, b_2, p, q)$, $\mathrm{P}_3 = \mathrm{CRT}(a_2, b_1, p, q)$, $\mathrm{P}_4 = \mathrm{CRT}(a_2, b_2, p, q)$
    3. $\{\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, \mathrm{P}_4\}$ 중 하나가 평문

Rabin

- 동작
  - 복호화
    - 2차 합동에 근을 가지고 있음
      - P는 $Z_n{}^*$의 값이기 때문에 C는 $Z_n{}^*$안에 근을 반드시 가지고 있음
    - 결정적 알고리즘이 아님
      - 복호화에선 4개의 답을 제시함



Rabin

- 동작
  - 예제 10.9 (1/2)

Rabin 알고리즘을 설명하기 위해 간단한 예를 들어보기로 하자.

- Bob은 $p = 23, q = 7$을 선택함 (두 수는 모두 모듈로 4로 3과 합동임)
  공개 키(161), 개인 키$(23, 7)$
- Alice가 Bob에게 평문 $\mathrm{P} = 24$를 보냄(161과 24는 서로소로 $24 \in {Z_{161}}^{*}$)

$$\mathrm{C} = 24^2 = 93 \; mod \; 161$$

- Alice는 93을 Bob에게 보냄
- Bob은 93을 수신하고 CRT를 계산함
  - $a_1 = +\left(93^{\frac{23+1}{4}}\right) mod \; 23 = 1 \; mod \; 23$
  - $a_2 = -\left(93^{\frac{23+1}{4}}\right) mod \; 23 = 22 \; mod \; 23$
  - $b_1 = +\left(93^{\frac{7+1}{4}}\right) mod \; 7 = 4 \; mod \; 23$
  - $b_2 = -\left(93^{\frac{7+1}{4}}\right) mod \; 7 = 3 \; mod \; 23$

Rabin

- 동작
  - 예제 10.9 (2/2)

Rabin 알고리즘을 설명하기 위해 간단한 예를 들어보기로 하자.

- Bob은 4개의 가능한 답$\{P_1, P_2, P_3, P_4\}$을 계산
  - $P_1 = \text{CRT}(a_1, b_1, p, q)$, $P_2 = \text{CRT}(a_1, b_2, p, q)$, $P_3 = \text{CRT}(a_2, b_1, p, q)$, $P_4 = \text{CRT}(a_2, b_2, p, q)$
  - $\{P_1, P_2, P_3, P_4\} = \{116, 24, 137, 45\}$

- 25%의 확률로 Alice가 보낸 평문 24를 고를 수 있음
  - 패딩의 유무, 중복 암호문의 유무 등의 메타데이터를 통해 선택함

- 4개의 답은 제곱하여 모듈로 값을 구해보면 전부 암호문 93이 나옴
  - $116^2 = 93 \bmod 161$
  - $24^2 = 93 \bmod 161$
  - $137^2 = 93 \bmod 161$
  - $45^2 = 93 \bmod 161$

Rabin

- RSA와 비교
  - 효율성
    - 암호화에서 Rabin은 제곱 모듈로를 계산하기 때문에 3차 이상을 계산하는 RSA에 비해 효율적임
    - 암호화와 복호화가 상대적으로 간단하지만 복호화 과정에서 제곱근 문제로 인해 다소 복잡할 수도 있음
  - 수학적 기반
    - 두 암호 시스템 모두 소인수분해의 어려움에 기반해 보안성을 제공함
    - 두 암호 시스템 모두 큰 키 길이를 요구함

	RSA	Rabin
키 생성	공개 키: (n, e), 개인 키: d	공개 키: n, 비밀 키: (p, q)
암호화	$C = P^e \bmod n$	$C = P^2 \bmod n$
복호화	$P = C^d \bmod n$	$P = \sqrt{C} \bmod p$ or $\sqrt{C} \bmod q$

목 차

- 비대칭 키
- 배낭 암호
- RSA
- Rabin
- ElGamal
- ECC

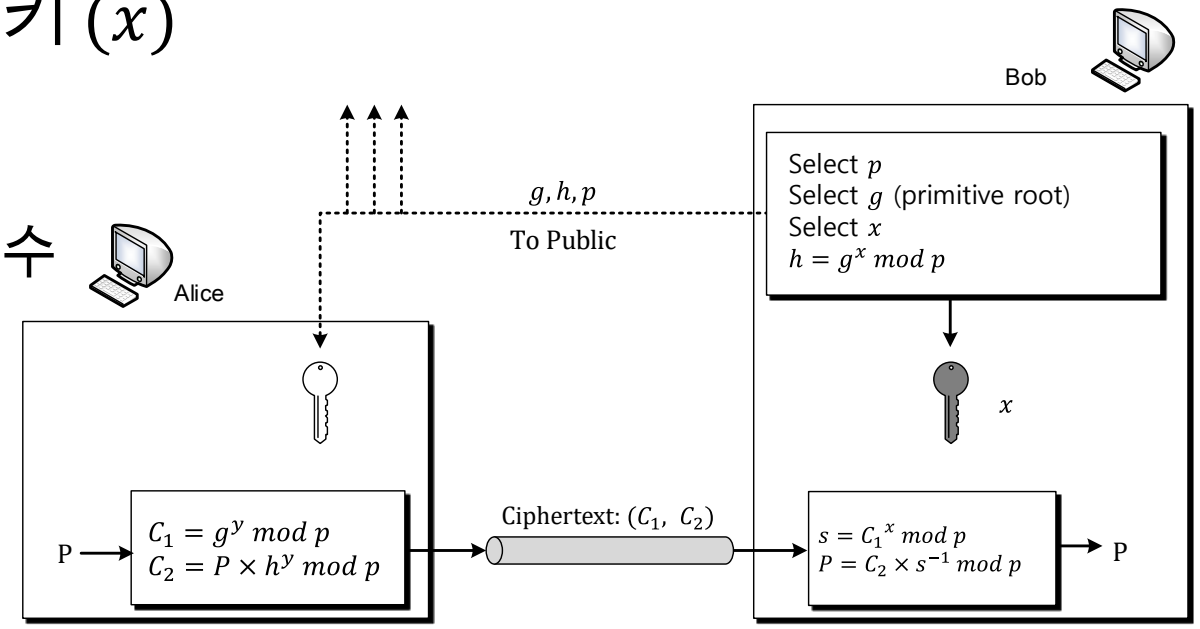
ElGamal

- 개요
  - 정의
    - 이산 로그 문제에 기반하고 난수를 사용하여 매번 다른 암호문을 생성하는 공개 키 암호 시스템

이산로그 문제

소수 p와 이의 원시근 정수 g, 그리고 g의 거듭제곱 $y \equiv g^x mod\ p$에서 y가 주어졌을 때, $x = log_g y$를 찾는 문제

  - 키 구성
    - 공개 키(p, g, h), 개인 키 (x)
      - g: p의 원시 근
      - h: $g^x\ mod\ p$
      - x: $1 \leq x \leq p-2$의 정수



ElGamal

- 동작
  - 키 생성
    1. 큰 소수 p를 선택
    2. p의 원시 근 g를 선택
    3. 1이상 $p-2$이하의 정수 x 선택
       - $p-1$은 p의 약수와 관련된 성질을 가져, 모듈로 연산에 영향을 미침
       - 페르마 소정리에 의해, $h = g^{p-1} \, mod \, p = 1$이 되어 공개키가 1이 됨
    4. $h = g^x \, mod \, p$ 계산

ElGamal

- 동작
  - 암호화
    1. 1이상 $p-2$ 이하 정수 y 선택
    2. $C_1 = g^y \ mod \ p$ 계산
    3. $C_2 = P \times h^y \ mod \ p$
    4. 암호문 (C_1, C_2) 생성

ElGamal_Encryption(g, h, p, P){ //P는 평문

Select a random integer y in the group G =< $Z_p{}^*$,×>
$C_1 \leftarrow g^y \ mod \ p$
$C_2 \leftarrow P \times h^y \ mod \ p$
return C_1 and C_2

}

ElGamal

- 동작
  - 복호화
    1. $s = \mathrm{C}_1{}^{x} \; mod \; p$ 계산
    2. s의 모듈러 역원 계산
       - $s \times s^{-1} \; mod \; p$
    3. $\mathrm{P} = \mathrm{C}_2 \times s^{-1} \; mod \; p$로 복호화

ElGamal_Decryption($\mathbf{s}, \boldsymbol{p}, \mathbf{C_1}, \mathbf{C_2}$){

$\mathrm{P} \leftarrow \mathrm{C}_2 \times s^{-1} \; mod \; p$
return P

}

  - 증명
    - $s = \mathrm{C}_1{}^{x} \; mod \; p = g^{xy} \; mod \; p$
    - $\mathrm{C}_2 = \mathrm{P} \times h^{y} \; mod \; p = \mathrm{P} \times g^{xy} \; mod \; p$
    - $\mathrm{P} = \mathrm{P} \times g^{xy} \times (g^{xy})^{-1} \; mod \; p$

ElGamal

- 동작
  - 예제 10.10 (1/2)

Bob은 $p = 11, g = 2(2 \in Z_{11}{}^{*})$를 선택하고 $x = 3, y = 4$를 선택했다. 평문 $\mathrm{P} = 7$에 대한 암호문을 계산하라.

- 암호화
  - $\mathrm{C}_1 = g^y \bmod p \quad \mathrm{C}_2 = \mathrm{P} \times h^y \bmod p \quad h = g^x \bmod p$
  - 공개 키$(p, g, h) = (11, 2, 2^3)$, 개인 키(3)
  - $\mathrm{C}_1 = 2^4 \bmod 11 = 16 \bmod 11 = 5 \bmod 11$
  - $\mathrm{C}_2 = 7 \times 8^4 \bmod 11 = 7 \times 4096 \bmod 11 = 6 \bmod 11$
  - $\therefore$암호문$(5, 6)$

ElGamal

- 동작
  - 예제 10.10 (2/2)

Bob은 $p = 11, g = 2(2 \in Z_{11}{}^{*})$를 선택하고 $x = 3, y = 4$를 선택했다. 평문 $\mathrm{P} = 7$에 대한 암호문을 계산하라.

- 복호화
  - $\mathrm{P} = \mathrm{C}_2 \times s^{-1} \, mod \, p \qquad s = \mathrm{C}_1{}^{x} \, mod \, p$
  - $\mathrm{P} = \mathrm{C}_2 \times s^{-1} \, mod \, 11 = 6 \times (5^3)^{-1} \, mod \, 11 = 6 \times 3 \, mod \, 11 = 7 \, mod \, 11$
  - ∴평문 7
- 페르마 소정리에 의해 아래와 같이 표현 가능
  - $\mathrm{P} = \mathrm{C}_2 \times (C_1{}^{x})^{-1} \, mod \, p \qquad \mathrm{P} = \mathrm{C}_2 \times \mathrm{C}_1{}^{p-1-x} \, mod \, p$
  - $\mathrm{C}_2 \times \mathrm{C}_1{}^{p-1-x} \, mod \, p = 6 \times 5^{11-1-3} = 6 \times 5^7 \, mod \, 11 = 7$

ElGamal

- 공격
  - 작은 모듈로 공격
    - 만약 p가 작아 $p-1$이 작은 소수들의 곱으로 표현될 수 있는 경우, 이산로그 문제가 쉽게 풀림
      - p는 적어도 300자리 이상의 10진수로 설정해야 함
      - e.g., $p-1=q_1q_2\cdots q_n$, g가 작은 소수 부분군의 원시근이라면 g의 거듭제곱이 작은 소수 부분 군 내의 모든 요소를 생성할 수 있음
  - 알려진 평문 공격
    - 공격자가 암호문과 대응하는 평문의 쌍을 알고 있는 경우, y에 임의의 정수 k를 넣어 해독함
      - y는 암호화할 때마다 매번 다르게 사용해야 함
      - $\mathrm{C}_1 = g^k \ mod\ p \quad \mathrm{C}_2 = \mathrm{P} \times h^k \ mod$

목 차

- 비대칭 키
- 배낭 암호
- RSA
- Rabin
- ElGamal
- ECC

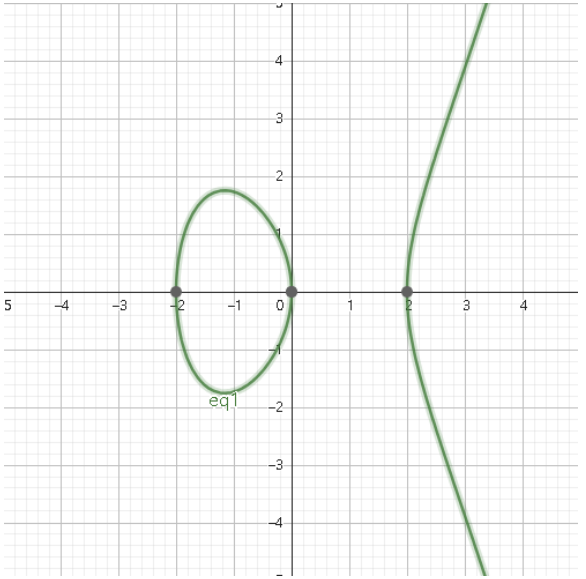
ECC (1/23)

- 개요
  - 정의
    - 타원 곡선 암호화(ECC, Eliptic Curve Cryptography)라고 하며 타원 곡선의 수학적 구조를 이용해 보안성을 높이고 키 크기를 줄이는 공개 키 암호 시스템
  - 타원 곡선 방정식
    - 일반적 타원 곡선식: $y^2 + b_1xy + b_2y = x^3 + a_1x^2 + a_2x + a_3$
    - 정칙 타원 곡선: $y^2 = x^3 + ax + b$
      - $4a^3 + 27b^2 \neq 0$을 만족해야 함
      - 즉, 곡선 미분이 0이 되는 점이 없도록 하는 곡선을 사용함

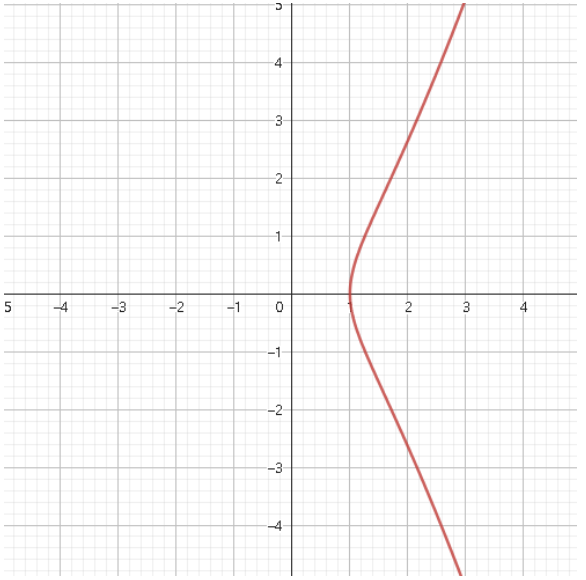
ECC (2/23)

- 공격
  - 예제 10.13

두 타원 곡선 방정식의 특이점을 살펴보자



- $y^2 = x^3 - 4x$
- $a = -4,\ b = 0$
- $4a^3 + 27b^2 = 4 \times (-4)^3 = -256 \neq 0$
- 해당 방정식은 특이점이 없는 정칙 타원 곡선임



- $y^2 = x^3 - 1$
- $a = 0,\ b = -1$
- $4a^3 + 27b^2 = 27 \times (-1)^3 = -27 \neq 0$
- 해당 방정식은 특이점이 없는 정칙 타원 곡선임

ECC (3/23)

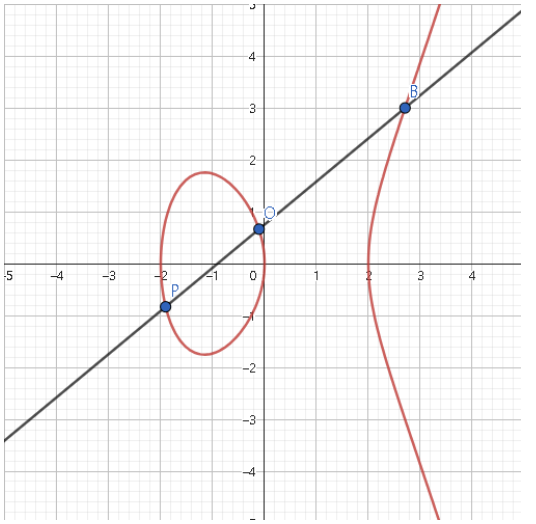
- 아벨 군
  - 타원 곡선의 점들로 아벨 군을 정의할 수 있음

$$G =< E(K), + >$$

  - 집합$(E(K))$
    - 곡선 상의 한 점을 좌표를 나타내는 쌍 $P = (x_1, y_1)$
    - $E(K) = \{(x_1, y_1) \in K^2 | y^2 = x^3 + ax + b\} \cup \{O\}$
      - O은 무한원점을 의미
  - 연산$(+)$
    - 실수 상 정의하는 덧셈과는 다르며, 곡선 상의 두 점을 더했을 때 곡선 상의 다른 한 점을 찾는 규칙을 의미함
      - $R = P + Q$, 여기서 $P = (x_1, y_1)$, $Q = (x_2, y_2)$이고 $R = (x_3, y_3)$

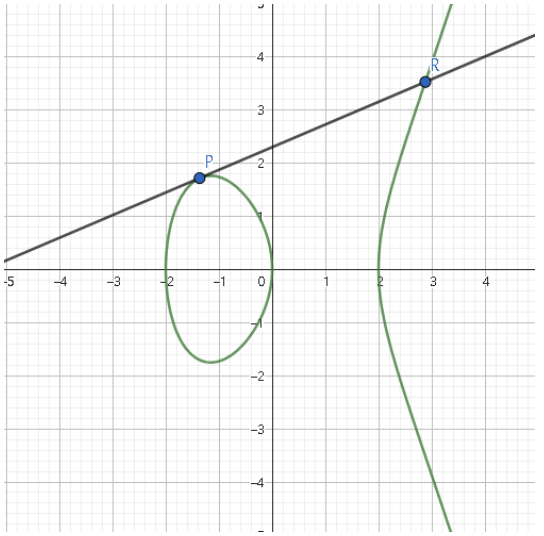
ECC (4/23)

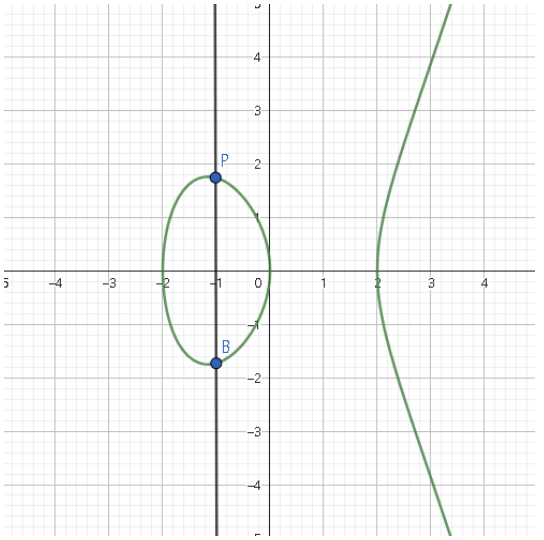
- 아벨 군
  - 연산(+) (1/2)
    - 다른 점들: $P + Q = -R$
      - 두 점 P와 Q가 다르면 P와 Q를 지나는 직선을 그려 타원 곡선과 만나는 세 번째 점 R을 찾음
      - R의 y좌표를 반전 시킨 점이 $P + Q$
      - 기울기 구하기
        - $x_3 = \lambda^2 - x_1 - x_2$
        - $y_3 = \lambda(x_1 - x_3) - y_1$
        - $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$



ECC (5/23)

- 아벨 군
  - 연산(+) (2/2)
    - 자기 자신과의 덧셈: $2\mathrm{P} = -\mathrm{R}$
      - 점 P에서 접선이 타원 곡선과 만나는 점 R를 찾음
      - 기울기 구하기
        - $x_3 = \lambda^2 - x_1 - x_2$
        - $y_3 = \lambda(x_1 - x_3) - y_1$
        - $\lambda = \frac{3{x_1}^2 + a}{2y_1}$
    - 무한 원점: $\mathrm{P} + O = \mathrm{P}$
      - 무한 원점 O를 덧셈의 항등원 역할을 함





ECC (6/23)

- 아벨 군
  - 연산의 성질
    - 닫힘
      - 앞에서 정의한 덧셈 연산을 이용하면 두 점을 합친 것이 다시 곡선상에 있다는 점을 증명할 수 있음
    - 결합법칙
      - $(\mathrm{P}+\mathrm{Q})+\mathrm{R}=\mathrm{P}+(\mathrm{Q}+\mathrm{R})$
    - 교환 법칙
      - $\mathrm{P}+\mathrm{Q}=\mathrm{Q}+\mathrm{P}$
    - 항등원의 존재성
      - 해당 덧셈 연산에 대한 항등원은 무한 영점 O임
    - 역원의 존재성
      - 곡선상의 점에 대한 역원은 x축에 대한 대칭 점임
        - $\mathrm{P}=(\mathrm{x}_1,\mathrm{y}_1)$과 $\mathrm{Q}=(\mathrm{x}_1,-\mathrm{y}_1)$는 서로 역원 관계이며 $\mathrm{P}+\mathrm{Q}=O$

ECC (7/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 앞선 아벨 군 같은 경우, 연산이 실수 위의 계산이었다면 점 좌표상의 연산은 $p > 3$에 대한 체 $\mathrm{GF}(p)$상의 연산임
    - 타원 곡선 군 $\mathrm{E}_p(a, b)$ 로 표현
  - 집합
    - $\mathrm{E}\big(\mathrm{GF}(p)\big) = \{(x, y) \in \mathrm{GF}(p)^2 | y^2 = x^3 + ax + b \,(mod\, p)\} \cup \{O\}$
      - (a, b)는 $\mathrm{GF}(p)$의 원소
      - O은 무한원점 또는 점 연산의 항등원을 의미함

ellipticCurve_points(*p*, *a*, *b*){ //p는 암호문

$x \leftarrow 0$
while($x < p$){

$w \leftarrow (x^3 + ax + b)\ mod\ p$ //w is y^3
if(w is perfect square in Z_p) output$(x, \sqrt{w})$ $(x, -\sqrt{w})$
$x \leftarrow x + 1$
}
}

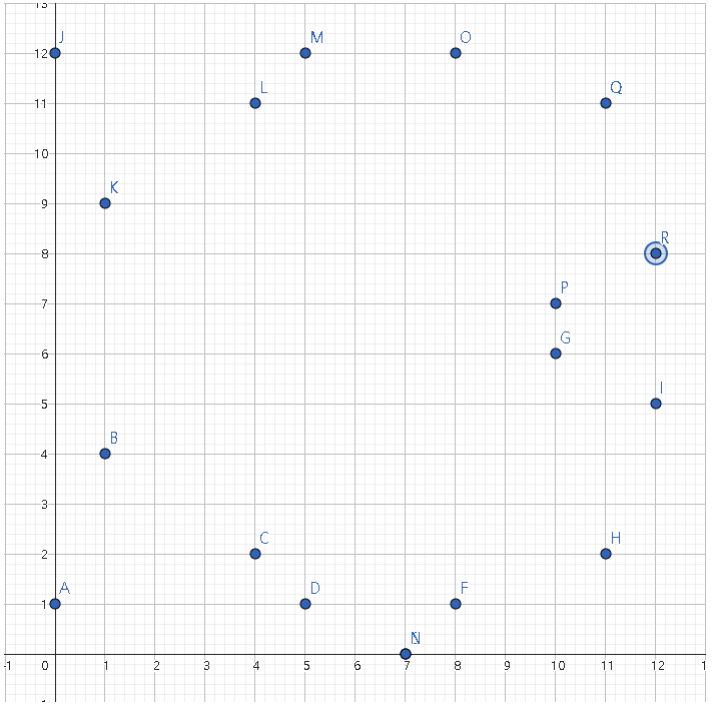
ECC (8/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 예제 10.14 (1/2)

타원 곡선 $\mathrm{E}_{13}(1,1)$을 정의하시오.

- 방정식은 $y^2 = x^3 + x + 1$이고 계산은 모듈로 13으로 연산함
- 곡선 상의 점들은 그림과 같이 표현 가능함

$(0,1)$	$(0,12)$
$(1,4)$	$(1,9)$
$(4,2)$	$(4,11)$
$(5,1)$	$(5,12)$
$(7,0)$	$(7,0)$
$(8,1)$	$(8,12)$
$(10,6)$	$(10,7)$
$(11,2)$	$(11,11)$
$(12,5)$	$(12,8)$



ECC (9/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 예제 10.14 (2/2)

타원 곡선 $\mathrm{E}_{13}(1,1)$을 정의하시오.

- 특징
  - 어떤 y^2값은 모듈로 13 상에서 제곱근을 갖지 못함
    - y가 0부터 12까지의 값일 때 y^2를 계산하면 $\mathrm{GF}(13)$에서 가능한 제곱 값은 $\{0, 1, 3, 4, 9, 10, 12\}$임
    - $x^3 + x + 1 (\mathrm{mod}\ 13)$ 값을 계산하여 제곱 값이 가능한지 확인함
      - 제곱근이 없는 값은 $\{2, 5, 7, 8, 9\}$임
    - 모든 점이 위에 값이 곡선 위에 있는 것은 아님을 의미함
  - 곡선에 정의된 모든 점을 역원을 가짐
    - e.g., $(7, 0)$은 자기 자신이 역원임
  - 역원들은 그래프에서 같은 수직선쌍에 나타남
    - e.g., 4가 y라면 9가 $-y$

ECC (10/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 연산
    - 두 점을 더하기
      - 덧셈과 곱셈에 대한 역원을 사용함
      - e.g., $\mathrm{P} + \mathrm{Q} = -\mathrm{R}$
        1. 기울기를 계산함
           - $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$
        2. 기울기 역수 계산함
           - $\lambda^{-1} \equiv \lambda^{p-2} \, mod \, p$
        3. 새로운 좌표 계산함
           - $x_3 = \lambda^2 - x_1 - x_2$
           - $y_3 = \lambda(x_1 - x_3) - y_1$

ECC (11/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 예제 10.15

$P = (4,2), Q = (10,6)$일 때, $\mathrm{R} = \mathrm{P} + \mathrm{Q}$를 구하라 (예제 10.14 사용)

- $\mathrm{P}=(x_1, y_1),\ \mathrm{Q}=(x_2, y_2),\ \mathrm{R}=(x_3, y_3)$

- $\lambda = \frac{6-2}{10-4} (\mathrm{mod} 13) = \frac{4}{6}$
- $\frac{4}{6} \equiv 4 \cdot 11 \equiv 44 \equiv 5 (mod\ 13)$
- $\therefore \lambda = 5$

- $x_3 = \lambda^2 - x_1 - x_2 = 5^2 - 4 - 10 = 12 - 4 - 10 = 11 (mod\ 13)$
- $y_3 = \lambda(x_1 - x_3) - y_1 = 5(4 - 11) - 2 = -37 \equiv -37 + 52 = 15 \equiv 2\ (mod\ 13)$

- $\mathrm{R}=(x_3, y_3)=(11, 2)$

ECC (12/23)

- $\mathrm{GF}(p)$상의 타원 곡선
  - 연산
    - 한 점에 상수를 곱하기
      - 어떤 수에 k번 곱한다는 것은 그 수를 k번 더하는 산술연산과 마찬가지로 자기자신에게 k번 더한다는 의미임
      - e.g., $\mathrm{E}_{13}(1,1)$에서 점 $\mathrm{P} = (8,1)$에 3을 곱함
        - $\mathrm{P} + \mathrm{P} = 2\mathrm{P} = (1,11)$계산
          - $\lambda = \frac{3{x_1}^2 + a}{2y_1} = \frac{3 \cdot 8^2 + 1}{2 \cdot 1} = \frac{193}{2} \equiv \frac{9}{2} (mod\ 13)$
          - $\lambda = 9 \cdot 7 = 63 \equiv 11\ mod\ 13$ ($\because 2$의 역원 7)
          - $x_3 = \lambda^2 - 2x_1 = 11^2 - 2 \cdot 8 = 121 - 16 = 105 \equiv 1\ (mod\ 13)$
          - $y_3 = \lambda(x_1 - x_3) - y_1 = 11(8-1) - 1 = 11 \cdot 7 - 1 = 76 \equiv 11\ (mod\ 13)$
        - $2\mathrm{P} + \mathrm{P}$ 계산
          - $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{10}{-7} \left(mod\ 13\right)$
          - $\lambda = 10 \cdot 11 = 110 \equiv 6\ mod\ 13$ ($\because -7 \equiv 6, 6$의 역원 11)
          - 이하 동일
          - $3\mathrm{P} = (1,2)$

ECC (13/23)

- $\mathrm{GF}(2^n)$상의 타원 곡선
  - $\mathrm{GF}(p)$와 다르게 모듈로 연산이 아닌 다항식으로 정의됨
    - 일반적으로 방정식 $y^2 + xy = x^3 + ax^2 + b$ 로 표현됨
  - 집합
    - $\mathrm{E}\big(\mathrm{GF}(2^n)\big) =$
      $\{(x, y) \in \mathrm{GF}(2^n) \times \mathrm{GF}(2^n) | y^2 + xy = x^3 + ax^2 + b\} \cup \{O\}$
      - a와 b는 $\mathrm{GF}(2^n)$의 원소임
      - O은 무한원점 또는 영점, 덧셈의 항등원을 의미함
      - y^2: 이진 체에서의 제곱 연산은 다항식의 비트 연산으로 처리됨
      - $x^3 + ax^2 + b$: 이항식의 계산은 비트연산과 다항식 연산을 통해 수행됨

ECC (14/23)

- $\mathrm{GF}(2^n)$상의 타원 곡선
  - 연산
    - 역원
      - $\mathrm{P} = (x, y)$라면 $-\mathrm{P} = (x, x + y)$임
    - 곡선 상의 점을 찾기 위해선 다항식 생성원을 사용함
      - e.g., 기약 다항식 $f(x) = x^3 + x + 1$을 사용하여 원소가 g^i인 $\mathrm{GF}(2^8)$
        - $\{0, g, g^2, g^3, g^4, g^5, g^6\}$로 표현
        - $g^3 + g + 1 = 0$으로 사용되거나 $g^3 = g + 1$로 사용할 수 있음
          - e.g., $g^5 = g^3 \cdot g^2 = (g + 1) \cdot g^2 = g^3 + g^2 = g^2 + g + 1$

0	000	$g^3 = g + 1$	011
1	001	$g^4 = g^2 + g$	110
g	010	$g^5 = g^2 + g + 1$	111
g^2	100	$g^6 = g^2 + 1$	101

(0,1)	(0,1)
$(g^2, 1)$	(g^2, g^6)
(g^3, g^2)	(g^3, g^5)
$(g^5, 1)$	(g^5, g^4)
(g^6, g)	(g^6, g^5)

ECC (15/23)

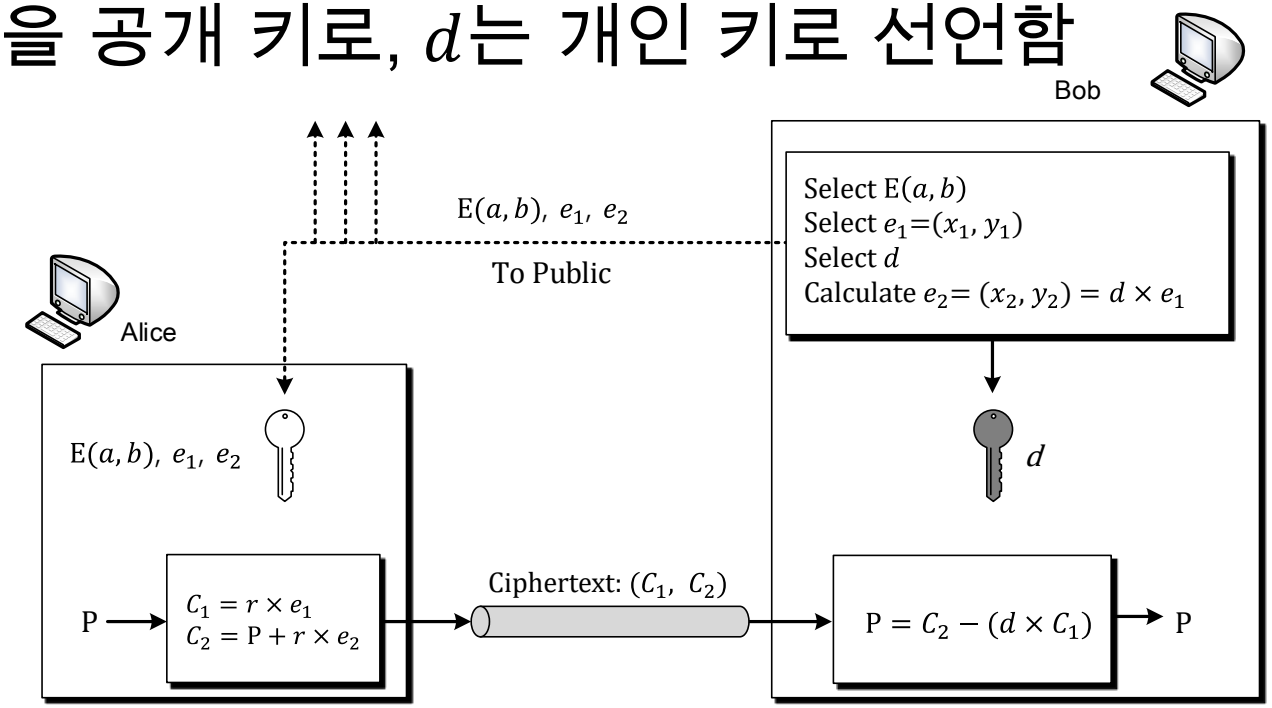
- $\mathrm{GF}(2^n)$상의 타원 곡선
  - 연산
    - 두 점을 더하기
      - $\mathrm{GF}(p)$의 규칙과 다르게 적용됨

1. $\mathrm{P} = (\mathrm{x}_1, \mathrm{y}_1)$, $\mathrm{Q} = (\mathrm{x}_2, \mathrm{y}_2)$이고 $\pm\mathrm{P} \neq \mathrm{Q}$인 경우, $\mathrm{P} + \mathrm{Q} = \mathrm{R} = (\mathrm{x}_3, \mathrm{y}_3)$
   - $\lambda = \frac{\mathrm{y}_1 + \mathrm{y}_2}{\mathrm{x}_1 + \mathrm{x}_2}$
   - $\mathrm{x}_3 = \lambda^2 + \lambda + \mathrm{x}_1 + \mathrm{x}_2 + a$
   - $\mathrm{y}_3 = \lambda(\mathrm{x}_1 + \mathrm{x}_3) + \mathrm{y}_1$
   - e.g., $\mathrm{P} = (0, 1)$이고 $\mathrm{Q} = (g^2, 1)$인 경우, $\lambda = 0$ ($\because$ XOR), $\mathrm{R} = (g^5, g^4)$

2. $\mathrm{P} = (\mathrm{x}_1, \mathrm{y}_1)$, $\mathrm{Q} = (\mathrm{x}_2, \mathrm{y}_2)$이고 $\mathrm{P} = \mathrm{Q}$인 경우, $\mathrm{P} + \mathrm{P} = \mathrm{R}$
   - $\lambda = \mathrm{x}_1 + \frac{y_1}{\mathrm{x}_1}$
   - $\mathrm{x}_3 = \lambda^2 + \lambda + a$
   - $\mathrm{y}_3 = {\mathrm{x}_1}^2 + (\lambda + 1)\,\mathrm{x}_3$
   - e.g., $\mathrm{P} = (g^2, 1)$인 경우, $\lambda = g^2 + \frac{1}{g^2} = g^2 + g^5 = g + 1$ ($\because x^3 = x + 1$)

ECC (16/23)

- ElGamal에 타원 곡선 적용
  - 키 생성
    - $\mathrm{GF}(2^n)$나 $\mathrm{GF}(p)$상의 타원 곡선 하나와 $\mathrm{E}(a, b)$ 선택함
    - 곡선상의 한점 $e_1=(x_1, y_1)$ 와 d 선택함
    - $e_2=d \cdot (x_1, y_1)$를 계산함 (일반 산술 연산이 아닌 GF상의 연산)
    - $\mathrm{E}(a, b)$, e_1, e_2을 공개 키로, d는 개인 키로 선언함



ECC (17/23)

- ElGamal에 타원 곡선 적용
  - 암호화
    - 평문에 해당되는 곡선 상의 한 점 P를 선택함
    - Alice가 암호문으로 사용할 두 점을 계산함
      - $C_1 = r \times e_1,\ \ C_2 = \mathrm{P} + r \times e_2$
      - (C_1, C_2)이 곡선 상 점일 수 있도록 Alice는 알고리즘을 사용해서 기호들과의 일대일 대응을 찾아야 함
  - 복호화
    - Bob은 수신받은 암호문을 해독함
      - $\mathrm{P} = \mathrm{C}_2 - (d \times \mathrm{C}_1)$ (여기서 마이너스 기호는 역원을 더한다는 의미임)
  - 증명
    - $\mathrm{P} + r \times e_2 - (d \times r \times e_1) = \mathrm{P} + (r \times d \times e_1) - (d \times r \times e_1)$
      $= \mathrm{P} + O = \mathrm{P}$

ECC (18/23)

- ElGamal에 타원 곡선 적용
  - 예제 10.19 (1/2)

$GF(p)$상의 타원 곡선을 사용한암호에 대한 예제를 살펴보자.

- 키 생성
  - Bob이 $E_{67}(2,3)$을 선택함
    - $y^2 = x^3 + 2x + 3 \ (mod\ 67)$
  - Bob이 $e_1 = (2, 22), d = 4$를 선택함
  - Bob이 $e_2 = (13, 45)$를 선택함. 이때 $e_2 = d \times e_1$임
  - ∴공개 키 (E, e_1, e_2)

- 암호화
  - Alice는 평문 $P = (24, 26)$을 보내려고 하고, $r = 2$를 선택함
  - $C_1 = r \times e_1 = (35, 1)$ $C_2 = P + r \times e_2 = (21, 44)$

ECC (19/23)

- ElGamal에 타원 곡선 적용
  - 예제 10.19 (2/2)

$\mathrm{GF}(p)$상의 타원 곡선을 사용한암호에 대한 예제를 살펴보자.

- 복호화
  - Bob이 전달받은 C_1과 C_2를 구함
  - $\mathrm{P} = \mathrm{C}_2 - (d \times \mathrm{C}_1)$
  - Bob이$(d \times \mathrm{C}_1) = (23, 25)$의 역원인 $(23, 42)$를 구함
  - $\therefore (35, 1) + (23, 42) = \mathrm{P}$

ECC (20/23)

- ElGamal에 타원 곡선 적용
  - 기존 ElGamal vs 타원 곡선을 적용한 ElGamal
    - 같은 키 길이에 비해 시뮬레이션이 더 높은 보안성을 가짐
    - 타원 곡선 연산은 모듈러 곱셈보다 연산 비용이 낮아 전력 소모가 적은 편임

	기존 ElGamal	시뮬레이션
사용되는 군	$G =< Z_p{}^*, \times >$	$\mathrm{E}_p(a,b)$
지수 계산	$h = g^x \bmod p$	점 P
곱셈와 덧셈	$s = g^{xy} \bmod p$	$\mathrm{P}+\mathrm{Q}$
역원	$s^{-1} \bmod p$	$-\mathrm{P}$
연산의 효율성	지수 계산	점 곱셈
연산의 대체	곱셈	덧셈
개인 키	정수	정수

ECC (21/23)

- 안전성
  - 타원 곡선 이산 로그 문제
    - 곡선 E와 타원 곡선 상의 두 점 P와 Q에 대하여, 정수 k를 찾아서 $\mathrm{Q} = \mathrm{kP}$를 만족시키는 k를 구하는 문제
      - P는 타원 곡선 상의 생성점이며, k는 이산 로그 문제의 해
    - r를 알아내기 위해선 r를 알 수 있는 대표적인 알고리즘 Pollard's rho알고리즘이 있음
      - $\mathrm{GF}(2^n)$의 n이나 $\mathrm{GF}(p)$의 p가 크다면 효과가 없음
    - 전통적인 이산로그 문제보다 해결이 어려움
  - 키 길이
    - $\mathrm{GF}(2^n)$상의 ECC의 160비트는 RSA의 1024비트 크기의 모듈로가 제공하는 안전성과 동일함

ECC(22/23)

- 비대칭 키 암호 비교
  - 안전성
    - 다른 암호 시스템보다 키 길이 대비 안전성이 강함

Date	Security Strength	Symmedtric Algorithms	Factoring Modulus	Discrete Logarithm		Elliptic Curve
				Key	Group	
Legacy	80	2TDEA	1024	160	1024	160
2019-2030	112	3TDEA AES-128	2048	224	2048	224
2019-2030 &beyond	128	AES-128	3072	256	3072	256
2019-2030 &beyond	192	AES-192	7680	384	7680	384
2019-2030 &beyond	256	AES-256	15360	512	15360	512

<출처>Elaine Barker, “Recommendation for Key Management”, Special Publication 800-57 Part 1 Rev.5, NIST, 05/2020.

ECC(23/23)

- 비대칭 키 암호 비교
  - 효율성
    - 다양한 암호 방식 설계가 용이해 SW 및 HW로 구현하기 쉬움
      - RSA는 인프라가 다소 구현된 환경에 적용된다면 ECC는 소형 모바일 환경에도 적용가능함

	ECC	ElGamal	Rabin	RSA
키 생성	타원 곡선의 원소를 기반으로 생성	큰 소수와 원시근을 기반으로 생성	큰 소수의 제곱을 기반으로 생성	두 큰 소수의 곱을 기반으로 생성
키 길이	일반적으로 RSA보다 짧음 (같은 보안 수준에서)	RSA와 비슷한 키 길이	RSA와 비슷한 키 길이	상대적으로 긴 키가 필요
수학적 기반	이산 로그 문제와 타원 곡선 문제 기반	이산 로그 문제 기반	소인수분해 문제에 기반	소인수분해 문제에 기반

Thanks!

김 혜 정(hyejeong@pel.sejong.ac.kr)