RSA

목차

- RSA 개요
- RSA 예
- 보안강도
- RSA 알고리즘
  - 키 생성
  - 암호화
  - 복호화
- RSA: 공격자, 인수분해 문제
- RSA: 소수 선택
- Square-and-multiply algorithm
- RSA 복호화에서 평문이 복원되는 이유

RSA

- 공개키 암호 기술 개념
  - 1976년 Whitfield **Diffie**, Martin **Hellman**이 최초 소개
- RSA
  - 1977년 Ron **R**ivest, Adi **S**hamir, Len **A**dleman에 의해 제안된 공개키 암호 기술

RSA 예

Reference: (Forouzan, 2008)

Alice

- Alice가 Bob에게 평문 $T = 5$를 전송하려 함
- Bob의 공개키$(e, n) = (13,77)$로 평문 암호화

$$C = T^e \; mod \; n = \; 5^{13} \; mod \; 77 = 26$$

- Bob에게 암호문 $C = 26$ 전송

Bob

- 두 소수 $p = 7, \; q = 11$ 선택
- $n = p \times q = 7 \times 11 = 77$ 계산
- $\phi(n) = (p-1) \; \times (q-1) \; = 60$ 계산
- 다음 조건을 만족하는 $e = 13$ 선택

$$\gcd(\phi(n), e) = 1, \qquad 1 < e < \phi(n)$$

- 다음 수식을 만족하는 $d = 37$ 계산

$$e \times d \; mod \; \phi(n) = 13 \times d \; mod \; 60 = 1$$

- 공개키 $(e, n) = (13,77)$ → 공개
- 개인키 $(d, n) = (37,77)$ → 안전하게 보관

- Alice가 보낸 암호문 $C = 26$ 수신
- 개인키$(d, n) = (37,77)$로 암호문 복호화

$$T = C^d \; mod \; n = \; 26^{37} \; mod \; 77 = 5$$

- Alice가 보낸 평문 5 수신

RSA 예

Reference: (Paar & Pelzl, 2010)

Alice

- Alice가 Bob에게 평문 $T = 4$를 전송하려 함
- Bob의 공개키 $(3, 33)$으로 평문 암호화

$$C = T^e \; mod \; n = \; 4^3 \; mod \; 33 = 31$$

- Bob에게 암호문 $C = 31$ 전송

Bob

- 두 소수 $p = 3,\; q = 11$ 선택
- $n = p \times q = 3 \times 11 = 33$ 계산
- $\phi(n) = (p-1) \times (q-1) = 20$ 계산
- 다음 조건을 만족하는 $e = 3$ 선택

$$\gcd(\phi(n), e) = 1, \qquad 1 < e < \phi(n)$$

- 다음 수식을 만족하는 $d = 7$ 계산

$$e \times d \; mod \; \phi(n) = 3 \times d \; mod \; 20 = 1$$

- 공개키 $(e, n) = (3,33)$ → 공개
- 개인키 $(d, n) = (7,33)$ → 안전하게 보관

- Alice가 보낸 암호문 $C = 31$ 수신
- 개인키 $(d, n) = (7,33)$으로 암호문 복호화

$$T = C^d \; mod \; n = \; 31^7 \; mod \; 33 = 4$$

- Alice가 보낸 평문 4 수신

보안 강도

Reference: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

보안강도 (비트)	안전 사용 기간	대칭키 알고리즘	공개키 알고리즘			해쉬 함수	
			이산대수	인수분해	타원곡선	전자서명, 충돌저항 필요 응용	메시지인증, 키유도함수, 난수생성
≤ 80	안전하지 않음	2TDEA	L=1024, N=160	RSA-1024	160	SHA-1	
112	2030년까지	3TDEA	L=2048, N=224	RSA-2048	224	SHA-224	
128	2031년 이후 계속	AES-128	L=3072, N=256	RSA-3072	256	SHA-256	SHA-1
192		AES-192	L=7680, N=384	RSA-7680	384	SHA-384	SHA-224
256		AES-256	L=15360, N=512	RSA-15360	512	SHA-512	SHA-256

- $\log_{10}(2^{1024}) \approx 309$
- 77을 소인수분해해 보시오
- 300자리 이상 십진수(예: https://en.wikipedia.org/wiki/RSA_numbers)를 소인수분해해 보시오

- L → 공개키 크기
- N → 개인키 크기
- 이산대수 → DH, DSA 등
- 인수분해 → RSA
- 타원곡선 → ECDH, ECDSA 등

RSA 알고리즘: 키 생성, 암호화, 복호화

Reference: (Stallings, 2014; 김명환, 2019)

Key generation (키 생성)

두 소수 p, q 선택 $(p \neq q)$

$n = p \times q$ 계산

$\phi(n) = (p-1) \times (q-1)$ 계산

e 선택 $(\gcd(\phi(n), e) = 1,\ 1 < e < \phi(n))$

d 계산 $((e \times d)\ mod\ \phi(n) = 1)$

공개키 (e, n)

개인키 (d, n)

Encryption (암호화)

평문 T에 대해 $(T < n)$, 암호문 C 생성

$$C = T^e\ mod\ n$$

Decryption (복호화)

암호문 C에 대해, 평문 T 복원

$$T = C^d\ mod\ n$$

- n으로부터 p, q를 빠른 시간에 알아내지 못하도록 p, q는 충분히 큰 소수여야 함
- **소수 선택 절차**

① 임의의 홀수 n 랜덤 선택
② 임의 정수 $a < n$ 랜덤 선택
③ a를 인자로 하여 확률적소수테스트(예: Miller-Rabin) 수행. 실패 시 ①로 이동
④ 테스트 성공 횟수가 충분하지 않으면 ②로 이동
⑤ n을 소수로 채택

- 소수 발견 전까지 얼마나 많은 후보 수들을 검사해야 하는가 → 소수정리에 따르면 $n = 2^{200}$인 경우 소수 찾기 전에 $(\log_e 2^{200})/2 \approx 70$회 시도 필요 (짝수는 즉시 거부)
- 소수정리(참고: 김명환, 2019, p71) → 임의의 양수 n에 대해, n 이하 소수의 개수 $\pi(n)$은 다음과 같다(아래 식에서 e는 자연로그의 밑으로 그 값은 약 2.718)

$$\pi(n) \sim \frac{n}{\log_e n}$$

- ***e*, *d* 선택 절차**

① $1 < e < \phi(n)$를 만족하는 임의의 수 e 랜덤 선택
② $\gcd(\phi(n), e)$를 구하는 확장유클리드알고리즘 수행
③ $\gcd(\phi(n), e) \neq 1$ 이면 ①로 이동
④ 확장유클리드알고리즘 결과로부터 법 $\phi(n)$에 대한 e의 역원 d 결정

- $\phi(n)$과 **서로소**인 수 e 발견 전까지 얼마나 많은 후보 수들을 검사해야 하는가 → 두 임의 정수가 서로소일 확률은 대략 0.6 (참고: https://en.wikipedia.org/wiki/Coprime_integers)

- 모듈러 지수승의 효율적 계산 → **square-and-multiply 알고리즘** → 이 알고리즘에도 불구하고 모듈러 지수승 계산에는 많은 계산량 필요 → 짧은 공개 지수 사용, 중국인의 나머지 정리 기반 빠른 복호화 등 가속화 기술

- 복호화에서 $\boldsymbol{C^d\ mod\ n = (T^e)^d\ mod\ n = T^{ed}\ mod\ n = T}$로 평문이 복원되는 이유?

RSA 알고리즘: 공격자, 인수분해 문제

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

Key generation (키 생성)
두 소수 $p,\ q$ 선택 $(p \neq q)$
$n = p \times q$ 계산
$\phi(n) = (p-1)\ \times (q-1)$ 계산
e 선택 $(\gcd(\phi(n), e) = 1,\ 1 < e < \phi(n))$
d 계산 $((e \times d)\ mod\ \phi(n)\ = 1)$
공개키 (e, n)
개인키 (d, n)

Encryption (암호화)
평문 T에 대해 $(T < n)$, 암호문 C 생성
$C = T^e\ mod\ n$

Decryption (복호화)
암호문 C에 대해, 평문 T 복원
$T = C^d\ mod\ n$

RSA 공격자
- 공개키 (e, n)
- $n = p \times q$로 인수분해할 수 있다면
- $(p-1) \times (q-1) = \phi(n)$ 계산한 후
- $ed \equiv 1\ (mod\ \phi(n))$을 만족하는 개인키 d를 얻을 수 있음

정수 인수분해 문제
- 두 큰 소수의 곱은 쉽게 계산됨
- 그러나 그 곱을 인수분해하는 것은 어려움

- $\log_{10}(2^{1024}) \approx 309$
- 77을 소인수분해해 보시오
- 300자리 이상 십진수(예: https://en.wikipedia.org/wiki/RSA_numbers)를 소인수분해해 보시오

RSA 알고리즘: 소수 선택

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

- RSA에서의 소수 p, q는 충분히 큰 소수여야 함 (n으로부터 p, q를 빠른 시간에 알아내지 못하도록)

소수 선택 절차

① 임의의 홀수 n 랜덤 선택
② 임의 정수 $a < n$ 랜덤 선택
③ a를 인자로 하여 확률적소수테스트(예: Miller-Rabin) 수행. 실패 시 ①로 이동
④ 테스트 성공 횟수가 충분하지 않으면 ②로 이동
⑤ n 을 소수로 채택

소수 발견 전까지 얼마나 많은 후보 수들을 검사해야 하는가

- 소수정리(참고: 김명환, 2019, p71) → 임의의 양수 x 에 대해, x 이하 소수의 개수 $\pi(x)$는 다음과 같다

$$\boldsymbol{\pi(x) \sim \frac{x}{\log_e x}}$$

- 임의의 정수 x가 소수일 확률은 다음과 같다(아래 식에서 e는 자연로그의 밑으로 그 값은 약 2.718)

$$\boldsymbol{P(x\text{가 소수}) \approx \frac{1}{\log_e x}}$$

- RSA가 1024비트 모듈로 n에 대해 동작하려면 소수 p, q는 512비트여야 하며 p, q의 후보로 선택된 임의의 홀수 x가 소수일 확률은 다음과 같으므로 177개 임의의 수를 검사하면 소수 발견 가능

$$\boldsymbol{P(\text{홀수 } x\text{가 소수}) \approx \frac{2}{\log_e 2^{512}} \approx \frac{1}{177}}$$

Square-and-multiply algorithm

Reference: (Stallings, 2014)

모듈러 지수승 계산

- RSA encryption, decryption에서의 모듈러 지수승 계산이 필요
- 지수승 계산 과정에서 큰 수가 생성됨

$$2108670300248232970141261 9^{497289573267425094678260 87} \ mod\ 6728035403036696970031072 1 = 123456789$$

- 아이디어 ① → 모듈러 연산의 다음 성질 이용하여 중간계산 결과의 크기를 줄임

$(a \times b)\ mod\ n = ((a\ mod\ n) \times (b\ mod\ n))\ mod\ n$
$654^2\ mod\ 13 = (654 \times 654)\ mod\ 13 = ((654\ mod\ 13) \times (654\ mod\ 13))\ mod\ 13 = (4 \times 4)\ mod\ 13 = 3$

- 아이디어 ② → a^{1024} 계산 시 1023회 곱셈 대신 a^2 계산 후 $a^2 \times a^2 = a^4$을 계산하고 이후 $a^4 \times a^4 = a^8$을 계산하는 방식으로 진행하면 10회 곱셈으로 a^{1024} 계산 가능

a^{45}의 효율적 계산

k	$45 = 101101_{(2)}$	k	a^k	v	b_i 값에 따른 처리
b_i	1 0 1 1 0 1	$k = 0$		$v = 1$	
1	0 1 1 0 1	$1 = 2 \times (0) + 1$	$a^1 = (a^0 \times a^0) \times a^1$	$v = (v \times v) \times a$	$b_i = 1$ → 제곱, 곱셈
1 0	1 1 0 1	$2 = 2 \times (1) + 0$	$a^2 = (a^1 \times a^1) \times a^0$	$v = (v \times v)$	$b_i = 0$ → 제곱
1 0 1	1 0 1	$5 = 2 \times (2) + 1$	$a^5 = (a^2 \times a^2) \times a^1$	$v = (v \times v) \times a$	$b_i = 1$ → 제곱, 곱셈
1 0 1 1	0 1	$11 = 2 \times (5) + 1$	$a^{11} = (a^5 \times a^5) \times a^1$	$v = (v \times v) \times a$	$b_i = 1$ → 제곱, 곱셈
1 0 1 1 0	1	$22 = 2 \times (11) + 0$	$a^{22} = (a^{11} \times a^{11}) \times a^0$	$v = (v \times v)$	$b_i = 0$ → 제곱
1 0 1 1 0 1		$45 = 2 \times (22) + 1$	$a^{45} = (a^{22} \times a^{22}) \times a^1$	$v = (v \times v) \times a$	$b_i = 1$ → 제곱, 곱셈

Square-and-multiply algorithm

Reference: (Stallings, 2014)

```
def square_and_multiply( a, k, n ):
    bits='{:b}'.format(k)
    v = 1
    for b in bits:
        v = (v * v) % n
        if b == '1' : v = (v * a) % n
    return v

c, d, n = 26, 37, 77
# c = 21086703002482329701412619
# d = 49728957326742509467826087
# n = 67280354030366969700310721
print( square_and_multiply(c, d, n) )
```

$a^k \ mod \ n$ 계산

Square

Multiply

$26^{37} \ mod \ 77 = 5$
$21086703002482329701412619^{49728957326742509467826087} \ mod \ 67280354030366969700310721 = 123456789$

49728957326742509467826087 (십진수)
1010010010001010000100001011011010111100110100010110100111001010101101011101111010011 (이진수)

RSA 알고리즘: 복호화에서 평문이 복원되는 이유

Reference: (Paar & Pelzl, 2010)

Key generation (키 생성)
두 소수 $p,\ q$ 선택 $(p \neq q)$
$n = p \times q$ 계산
$\phi(n) = (p-1) \times (q-1)$ 계산
e 선택 $(\gcd(\phi(n), e) = 1,\ 1 < e < \phi(n))$
d 계산 $((e \times d)\ mod\ \phi(n)\ = 1)$
공개키 (e, n)
개인키 (d, n)

Encryption (암호화)
평문 T에 대해 $(T < n)$, 암호문 C 생성
$C = T^e\ mod\ n$

Decryption (복호화)
암호문 C에 대해, 평문 T 복원
$T = C^d\ mod\ n$

복호화에서 $C^d\ mod\ n = T$로 평문이 복원되는 이유?

- $ed \equiv 1\ (mod\ \phi(n))$으로부터 $ed = 1 + k \times \phi(n)$이 성립
- $C^d\ mod\ n = (T^e)^d\ mod\ n = T^{ed}\ mod\ n = T^{1+k\times\phi(n)}\ mod\ n = T(T^{\phi(n)})^k\ mod\ n$
- 즉 $C^d\ mod\ n = T(T^{\phi(n)})^k\ mod\ n = T$임을 보여야 함

① $\gcd(T, n) = 1$인 경우

- 오일러 정리에 따라 $T^{\phi(n)} \equiv 1\ (mod\ n)$이므로
- $T(T^{\phi(n)})^k\ mod\ n = T \times (1)^k\ mod = T$

② $\gcd(T, n) = \gcd(T, p \times q) \neq 1$인 경우 아래 두 ⓐ, ⓑ로 나뉨

- 소수 $p,\ q$에 대해, T와 $p \times q$가 서로소가 아니라면 T는 p 혹은 q의 배수임

ⓐ T가 소수 p의 배수이고 소수 q의 배수가 아닌 경우 $(T = r \times p)$

- $\gcd(T, q) = 1$ 이므로 오일러 정리에 따라 $T^{\phi(q)} \equiv 1\ (mod\ q)$이므로
- $(T^{\phi(n)})^k\ mod\ q = (T^{(p-1)(q-1)})^k\ mod\ q = (T^{\phi(q)})^{k(p-1)}\ mod\ q = 1$
- 즉 $(T^{\phi(n)})^k\ mod\ q = 1$ 이므로
- $(T^{\phi(n)})^k = 1 + uq$
- 양변에 T를 곱하면 $T(T^{\phi(n)})^k = T + Tuq = T + (rp)uq = T + (ru)n$이 되어 $T(T^{\phi(n)})^k\ mod\ n = T$

ⓑ T가 소수 q의 배수이고 소수 p의 배수가 아닌 경우 $(T = s \times q)$

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