암호화 알고리즘의 성능 분석 보고서

암호화 방식에 따른 암호화 속도 차이

I.서론

1. 연구의 필요성 또는 동기

현재 중요한 정보를 암호화 하거나 서로를 확인하기 위한 방법으로 소인수분해를 사용하는 RSA 암호화 알고리즘이 대표적으로 사용되고 있다. 최근에는 타원곡선을 사용하는 새로운 암호화 알고리즘인 ECC 알고리즘이 공개되었는데 두 알고리즘의 암호화 속도에 차이가 있다. 우리는 어떤 요소로 인하여 속도 차이가 나는지 궁금해져, 이를 분석하기로 했다.

2. 연구 문제

각 알고리즘의 속도에 영향을 미치는 요소는 무엇인가?

II. 이론적 배경

1. RSA 암호화는 현재 실생활에서 가장 많이 사용되고 있는 대표적인

양방향 데이터 암호화 기법이자 공개키 암호화 방식이다.

*양방향 암호화: 암호화된 암호문을 복호화할 수 있는, 즉 사람이 알기 쉽게 처리할 수 있는 알고리즘을 의미한다.

*공개키 암호화 방식: 암호학적으로 연관된 두 개의키를 만들어서 하나는 자신이 보관하고 다른 하나는 상대방에게 공개하는 것을 말한다. 다른 사용자와키를 공유하지 않더라도 암호를 통한 통신을 할 수 있다는 장점이 있다.

* 비밀키 = 비공개키 = 개인키

*RSA 알고리즘이 암호화되는 간략한 과정

1. 임의의 두 소수 p, q 를 생성하고 그 둘을 곱해 n 을 구한다. (n = p * q)

2. e (공개키) 구하기: Φ(n) = (p - 1) * (q - 1) 식을 이용하여 Φ(n)을 구한 후,

e 는 1 < e < Φ(n)로써 1 과 Φ(n) 사이에 있고 Φ(n)와 서로소인 e 를 구한다.

3. d (개인키) 구하기: (e * d) mod Φ(n) = 1 즉, e*d 를 Φ(n)으로 나누었을 때 나머지가 1 인 d 를 구한다.

4. 암호화할 평문을 숫자로 변환한 것을 M 이라고 하고 암호화된 암호문을 C 라고 할 때

C = (M^e) mod (n) 을 이용해 암호화한다.

* Φ(n) (오일러 피 함수): 1 부터 n 까지의 정수 가운데 n 과 서로소인 원소의 개수를 나타낸다.

* a mod b (모듈러 연산) : a 를 b 로 나누었을 때 나머지를 나타낸다.

RSA 암호화 과정의 예시

1. 임의의 두 소수 p=31 q=47 를 정하고 n = pq 를 이용하여 n 을 구한다 (n = p * q = 1457)

2. **Φ(n) =(p-1) * (q-1) 공식으로 Φ(n) 을 구한다. Φ(n) = (31-1)(47-1)=1380**

3. 공개키 e 는 1< e < **Φ(n) = 1380** 이고 **Φ(n)**와 서로소임을 이용하여 e 를 구한다. (e = 19)

4. (e * d) mod **Φ(n) = 1 을 이용하여 비밀키 d 를 구한다.** (19 * d) mod **Φ(n) = 1, d=73**

만약, m = 100 를 암호화하기 위해서는 다음과 같이 계산한다. 암호화된 암호문을 c 라고 할 때,

c = (m^e) mod (n) 을 이용하여 c 를 구한다. c = (100 ^ 19) = 280 mod 1457

복호화 시 m = (c ^ d) mod n 를 이용하여 평문 m 을 구할 수 있다. m = (280 ^ 72) mod 1457

이때 공격자는 암호문 c 를 알아도 비밀키 d 를 알아낼 수 없기에 복호화가 불가능하다.

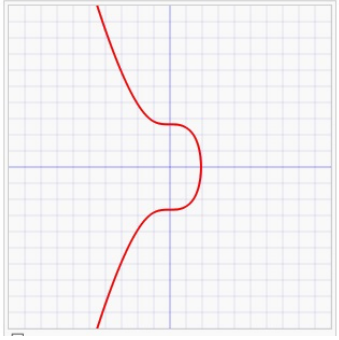
---

2. ECC 암호화란 타원곡선 이론에 기반한 공개키 암호 방식으로, 타원곡선 위의 임의의 점 P 에 정수 k 를 곱하는 점 스칼라 곱셈 Q=kP 로 정의된다.

이때 타원곡선 위의 점들, 즉 좌푯값들의 계산이므로 비밀키 데이터는 모두 숫자여야 한다.

타원곡선 위의 점 P 와 점 Q 를 지나는 직선이 타원과 만나는 교점 R 을 x 축으로 대칭시킨 점을 P+Q=R 로 정의한다.

사용되는 타원곡선은 대부분 미리 정의되어 있는데 비트코인에서 사용되는 secp256k1 curve 가 대표적인 예시이다.



secp256k1curve (Y^2 = X^3 + 7)

*ECC 알고리즘이 암호화되는 간략한 과정

-P : 유한체의 표수로, 쉽게말해 유한체의 크기를 설정하는 값이다. 반드시 소수여야 한다.

-G: 생성자, 타원곡선 위의 임의의 점

-x: 개인키, P 보다 작은 소수로, 난수 생성기로 생성

-Q: 공개키, 개인키로부터 연산

점 P(x,y)가 타원곡선 위에 위치할 때 두 점 P, Q 는 임의의 정수 x 에 대하여 Q=xG 라는 방정식이 성립한다.

이때, Q 와 G 값만을 가지고 x 값을 알아내기 힘든 속성을 이용해 x 를 개인키로 Q 를 공개키로 활용한다.

* 유한체란 유한개의 원소를 가지는 집합이다. 집합 내에서 이루어지는 연산의 결과가 다시 그 집합 내의 원소로 나타난다.

ECC 암호화 과정의 예시

예를 들어 y 가 P 가 11 인 유한체 위에 존재할 때

G(α) = α = (x_1, y_1) = 생성자

$$E: y^2 = x^3 + x + 6 \; over \; Z_{11}$$

$$G(\alpha) = (2, 7)$$

$$2\alpha = (x_2, y_2)$$

2α 를 구한다면

$$\lambda = \frac{3x_1^{\,2} + a}{2y_1} = \frac{3(2)^2 + 1}{2 \times 7} = \frac{13}{14} = 2 \times 3^{-1} = 2 \times 4 = 8 \mod 11$$
$$x_2 = \lambda^2 - 2x_1 = (8)^2 - 2 \times (2) = 5 \mod 11$$
$$y_2 = (x_1 - x_2)\lambda - y_1 = (2-5) \times 8 - 7 = 2 \mod 11$$

2α=(5,2)

$$\alpha = (2,7)$$
$$pk = 7 \ (Private \ Key)$$
$$k = 3 \ (Random \ value)$$
$$x = (10,9) \ (x \ is \ plaintext)$$

위와 같이 유한체 위 타원곡선이 존재하고 공개키 **α(2, 7)**인 상태에서

어떤 사람 A 가 B 에게 x(10, 9)라는 평문을 보내고 싶어한다면

A 가 선택한 비밀키 pK=7 이라 가정하고 A 가 선택한 난수 k=3 이라 가정한다.

ECC 에서 암호화 공식은 다음과 같다. β = 공개키

- $\beta = pk \cdot \alpha$
- $y_1 = k \cdot \alpha$
- $y_2 = x + k \cdot \beta$

(y_1, y_2) 를 B 에게 보내면 B 는 아래와 같은 복호화 과정을 거친다.

$$x = y_2 - (pk * y_1)$$

$$x = y_2 - 7y_1$$

이때 공격자는 암호화 된 (y_1, y_2)를 알아도 비밀키 pk 를 알 수 없기에 복호화가 불가능하다.

Ⅲ. 연구 방법

1. 각 알고리즘의 암호화 과정을 분석하고 세분화한다. 1)

2. 암호화 코드를 세분화한 것에 따라 시간을 재는 코드를 삽입한다.

3. 각 코드를 10 만 번 반복 실행시켜 걸린 시간의 평균값을 얻는다. 2)

4. 위의 실험에서 얻은 결과를 바탕으로 알고리즘의 어떤 요소가 암호화의 성능에 어느 정도의 영향을 미치고 있는지 확인한다.

* 이 실험에서 사용하는 두 알고리즘의 보안 수준은 같게 설정한다. 즉 공개, 비공개 키의 bit 를 같게 설정한다.

1) 이 실험에서는 키 생성을 위한 소수 준비, 공개키와 비밀키 생성, 암호화 총 3 개 과정으로 구분했다.

2) 반복 실행 횟수를 10 만 번으로 정한 근거는 횟수를 5 만번부터 10 만 번 까지 점차 올렸을때

결괏값의 변동이 거의 없기 때문에, 10 만 번 이상은 무의미한 것으로 판단하였다.

Ⅳ. 본론

1. RSA 데이터 분석

RSA 알고리즘은키 생성을 위한 소수 준비 과정에서 0.000196 초를, 공개, 비밀키 생성 과정에서 0.220435 초, 암호화 과정에서 0.000012 초로 총 0.220644 초가 소요되었다.

소수 준비 과정 0.09%, 공개, 비밀키 생성 과정 99.91%, 암호화 과정 0.01%

2. ECC 데이터 분석

ECC 알고리즘은 생성을 위한 소수 준비과정이 반올림 후 0 초, 공개키 비밀키 생성 과정에서 0.00198 초, 암호화 과정에서 0.00001 초로 총 0.00199 초가 소요되었다.

소수 준비 과정 0.00%, 공개, 비밀키 생성 과정 99.5%, 암호화 과정 0.5%

V. 결론

두 알고리즘 모두 총 암호화 시간 중에서 공개, 비밀키 생성 과정이 가장 큰 비율을 차지하였다.

이 과정에 비하면 나머지 소수 준비 과정과 암호화 과정은 무시할 수 있을 정도이다.

RSA 암호화의 경우 매우 큰 소수 두 개의 소인수분해를 이용하는 데 반해 ECC 알고리즘은 미리 정의된 타원곡선을 이용하기 때문에 비교적 암호화에 사용되는 공개키와 비밀키 크기가 작고 이는 암호화 시간단축으로 이어진다. 따라서 ECC, RSA 알고리즘의 속도 차이는 암호화 방식에 따른 키의 크기의 차이 때문이다.

시간이 짧게 걸리므로 암호 해독에 필요한 시간도 짧아져 결국 보안성이 떨어지는 것 아닌가 하는 반론이 제기될 수 있으나 ECC 알고리즘은 여러 타원 곡선 방정식을 사용하여 공격자의 공격을 어렵게 한다. RSA 암호화의 경우 두 소수를 알아내면 암호가 풀리지만, ECC 암호의 경우 생성자, 유한체의 표수, 타원 곡선 방정식을 알아야 하므로 경우의 수가 기하급수적으로 늘어나 짧은 키로도 RSA 암호화와 동일한 수준의 보안 수준을 제공한다.

VI.참고문헌

-ECC-

dev_mac-_-, "기초암호학(4) - ECC(타원곡선 암호화 알고리즘), 티스토리, 2019.04.13, https://developer-mac.tistory.com/83

최준백 • 신경욱, 2021, 타원곡선 기반 공개키 암호 시스템 구현을 위한 scalable ECC 프로세서, 〈한국정보통신학회논문지〉

김현수 • 박석천, 2011, 음성 데이터 보안을 위한 효율적인 ECC 암호 알고리즘 설계 및 구현, 〈한국정보통신학회논문지〉

〈정보통신기술용어해설〉, 2023, http://www.ktword.co.kr/test/view/view.php?m_temp1=752

-RSA-

x