RSA Encryption Algorithm: An In-Depth Review

By Subhajit Das and Daniel Hoogasian

Description

- RSA is a cryptography algorithm created in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT
- Widely used in e-commerce
- Utilizes prime numbers and number theory concepts discovered in 15th and 16th centuries
- Uses large prime numbers called “public” and “private” keys, shared between parties

Math and Mechanics: Key Generation

- First, two prime numbers (*p* and *q*) are generated with a primality test (i.e., an algorithm that finds primes), say 7 and 19
  - In practice, they are very large (often 155 digits long)
- Calculate $p*q$, to create the modulus, 7*19=133 (we will call this *n*)
- Calculate the total amount of numbers lower than *n*, that share no factors with *n* besides 1; $\phi(n) = (p-1)(q-1)$
  - Called Euler's totient function
  - Numbers are said to be "relatively prime" to *n*
- So, $\phi(p) = 6$ and $\phi(q) = 18$, then multiply them 6*18 to get $\phi(n) = 108$

Math and Mechanics: Key Generation (cont'd)

- Choose an integer *e,* such that $1 < e < \phi(n)$ and is relatively prime to $\phi(n)$, say 29
  - (n, e) is the public key
- Choose an integer *d,* such that $1 < d < \phi(n)$ and is relatively prime to $\phi(n)$, where $(e*d) \% \phi(n) \equiv 1$
  - Use what is called Euler's extended formula to get 41
  - (n, d) is the private key

Math and Mechanics: Encryption

- Message, *m*, needs to be in numeric form (e.g., ASCII, UTF-8, etc.)
  - Example; ‘RSA’ in ASCII = 82 83 65
- Compute ciphertext, $c = m^e \% n$
  - For first letter ‘R’, 82^{29} % 133 = 17
  - Second letter ‘S’, 83^{29} % 133 = 125
  - Third letter ‘A’, 65^{29} % 133 = 88
  - Ciphertext is 17 125 88

Math and Mechanics: Decryption

- Compute deciphered message, $m = c^d \% n$
  - Ciphertext: 17 125 88
  - For the first ciphertext character encoding, $17^{41} \% 133 = 82$, which maps to ‘R’
  - Second, $125^{41} \% 133 = 83$, which maps to ‘S’
  - Third, $88^{41} \% 133 = 65$, which maps to ‘A’
  - Deciphered text: RSA

Principle Mathematics: Two Main Concepts

1.)

- Prime generation is easy (i.e., choosing p and q)
- Multiplication is easy (i.e., solving for the product, $n = p*q$)
- Finding the prime factors of n is hard
  - Finding the prime factors of a 1024-bit number would take one year on a $10M supercomputer [1]

2.)

- Modular exponentiation is easy, if we know n, m and e (i.e., $c = m^e \% n$)
- Modular root extraction, the reverse of modular exponentiation, is easy if we know the prime factors (solving for m in, $c = m^e \% n$, knowing p and q)
- If we don't know the prime factors, modular root extraction is hard (i.e., recovering m, only knowing n, e, and c)[1]

Sample Code Results

- With our example:

```
Public key:  (133, 31)
Private key:  (7, 19, 7)
Encrypted message:  =SA
Decrypted message:  RSA
```

- With randomly generated slightly larger primes:

```
Public key:  (33491, 3311)
Private key:  (107, 313, 13055)
Encrypted message:  穎怌
Decrypted message:  RSA
```

- Unfortunately using traditionally sized primes isn’t feasible on a regular machine…

```
1024 bit prime p is:
 926673964095313927666976136945924085492161375755945319591764915030157359560015
9261885478806988821113574324734432755622991661477031251148879523249156984803652
4014375447752054042996001021998276712860654706844513292783412949692920318994337
988982003724584944484940327315968458514691736127132770493009577997639721
```

Two Historic Theorems and One Historic Function

- Fermat's Little Theorem
  - If p is a prime that is relatively prime to integer a, then $a^{p-1} \equiv 1 \% p$
  - What RSA based on
  - Great influence on algorithmic number theory and is the basis of some of the most well-known algorithms for primality testing [2]
- Euler's Theorem
  - States that if gcd(a,n) = 1, then $a^{\phi(n)} \equiv 1 \% n$ [3]
  - An extension of Fermat's Little Theorem, for when numbers aren't prime
- Euler's Phi Function (AKA Euler's totient function)
  - If the prime factorization of n is given as $n = p_1^{e1} * \ldots * p_n^{en}$, then $\phi(n) = n * (1 - 1/p_1) * \ldots (1 - 1/p_n)$ [4]
  - Many applications in cryptography and computer security

Issues of RSA Algorithm

- A computer can quickly compute the greatest common divisor of two numbers using the Euclidean algorithm, so an attacker can run this algorithm to find prime numbers
- If *p* and *q* are too close to each other, private key will be smaller, then an attacker can efficiently determine the private key
- MIT mathematician Peter Shor developed a theoretical algorithm for quantum computers that factors numbers exponentially faster than current algorithms do

Conclusion

- There is a growing body of evidence that RSA is no longer the best choice for modern asymmetric applications
- Elliptic curve algorithms as an alternative to RSA
- Larger RSA keys with more carefully chosen modulus operators as an immediate solution to bolster RSA algorithm