Attacks on RSA

Yael Zayats
yael.zayats@gmail.com

Euler Circle

July 11, 2023

What is Cryptography? Why do we need it?

Cryptography: method of protecting information and communications through the use of codes

History of RSA

Ronald Rivest

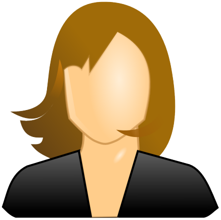
Adi Shamir

Leonard Adleman

What is it used for?

- Digital Signatures
- Transactions
- Communication
- Remote Access
- File Encryption
- IoT Security

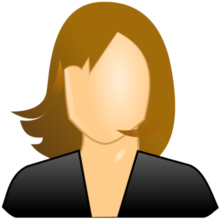
How does RSA work?



Alice

Bob

How does RSA work?



Alice

Bob

Eve

How does RSA work?

Public Key: (e, N)

Private Key: d

How does RSA work?

Choose 2 prime numbers, p and q

Public Key:

- $N = p \cdot q$
- calculate $\varphi(n) = (p-1)\cdot(q-1)$
- choose e where $1 < e < \varphi(n)$ and e is coprime to $\varphi(n)$

Private Key:

- $d = e^{-1} mod \varphi(n)$

How do you Encrypt/Decrypt?

To encrypt a message M, where $M < N$, into a cipher text C , the user will use the equations:

Encrypt:
$C = M^e \cdot mod(N)$

Decrypt:
$M = C^d \cdot mod(N)$

List of Attacks

- Searching the Message Space
- Guessing d
- Cycle Attack
- Wiener's Attack
- Common Modulus
- Faulty Communication
- Coppersmith Theorem
- Hastad's Broadcast
- Coppersmith's Short Pad
- Partial Key Exposure
- Blinding
- Timing
- Bleichenbacher's Attack on PKCS
- Random Faults
- Fermat's Factorization
- Pollard's $p-1$ Algorithm
- Number Field Sieve
- Shor's Algorithm
- Quantum Computing

List of Attacks

- Searching the Message Space
- Guessing d
- Cycle Attack
- Wiener's Attack
- Common Modulus
- Faulty Communication
- Coppersmith Theorem
- Hastad's Broadcast
- Coppersmith's Short Pad
- Partial Key Exposure
- Blinding
- Timing
- Bleichenbacher's Attack on PKCS
- Random Faults
- Fermat's Factorization
- Pollard's $p - 1$ Algorithm
- Number Field Sieve
- Shor's Algorithm
- Quantum Computing

Types of Attacks

Guess & Check

Faults & Errors

Factorization

Guess & Check Methods

Brute forcing through, generally trying to guess part of the key(s).

Cycle Attack

Encrypt the ciphertext over and over until the plaintext appears. This number of "cycles" will decrypt any ciphertext.

Essentially, you calculate $M^e(\mathrm{mod}(N)), M^{e^2}(\mathrm{mod}(N))\ldots$ and so on until, for some k, $M^{e^k}(\mathrm{mod}(N)) = M$.

Cycle Attack

Encrypt the ciphertext over and over until the plaintext appears. This number of "cycles" will decrypt any ciphertext.

Essentially, you calculate $M^e(\mathrm{mod}(N)), M^{e^2}(\mathrm{mod}(N))\ldots$ and so on until, for some k, $M^{e^k}(\mathrm{mod}(N)) = M$.

The issue? this attack takes an absurdly long time.

Cycle Attack

There are very few values of e with a short cycle length $\varphi(n)$.

On average, the biggest prime factor of $p - 1$ will have a size close to 30% of $p - 1$, aka 150 bits for a 1024 RSA modulus.

If r is the prime factor, this implies that:

Cycle Attack

There are very few values of e with a short cycle length $\varphi(n)$.

On average, the biggest prime factor of $p-1$ will have a size close to 30% of $p-1$, aka 150 bits for a 1024 RSA modulus.

If r is the prime factor, this implies that:

- If the attacker chooses a e with an order multiple of r, then the cycle length will be at least as big as r, hence way too long for the attack to be feasible

Cycle Attack

There are very few values of e with a short cycle length $\varphi(n)$.

On average, the biggest prime factor of $p-1$ will have a size close to 30% of $p-1$, aka 150 bits for a 1024 RSA modulus.

If r is the prime factor, this implies that:

- If the attacker chooses a e with an order multiple of r, then the cycle length will be at least as big as r, hence way too long for the attack to be feasible
- The chances of a random e having an order which is not a multiple of r, are at most $1/r$, aka way too small for the attacker hitting one out of pure luck.

Faults in Encryption & Human Errors

Humans and our code aren't perfect! We make a LOT of mistakes.

Timing Attack

Cryptographic operations take a varying amount of time to complete, depending on the keys.

Timing Attack

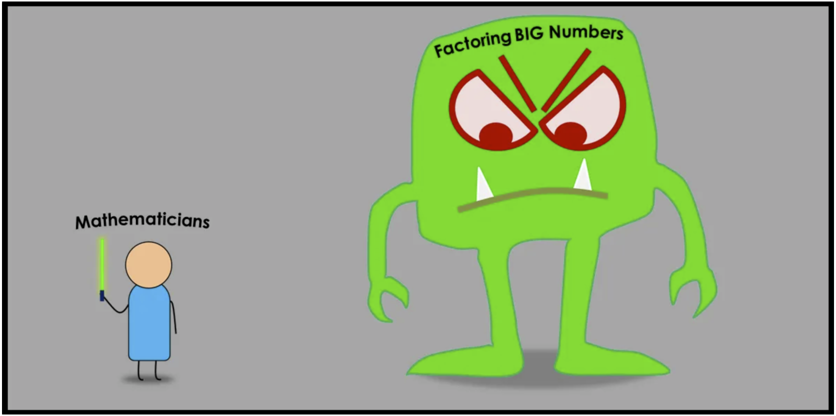
Cryptographic operations take a varying amount of time to complete, depending on the keys.

This is computationally practical as the sample size required is proportional to the number of bits in the private key, and the number of bits is finite.

Factorization

Sort of guess & check, but trying to find p and q instead of the keys.

Factorization

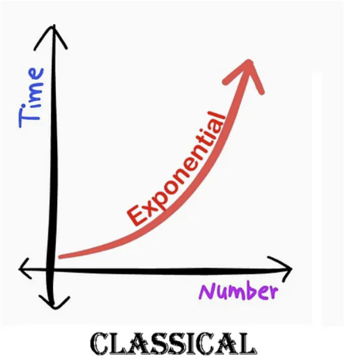


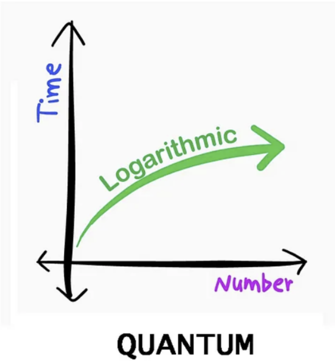
Shor's Algorithm

Choose a number x to factorize

1. choose a number k between 1 and x
2. find $\gcd(x, k)$
   2.1 if gcd is not 1, then the GCD is a factor
   2.2 if gcd is 1, define $q = 1$
   2.3 find $(q \cdot k) \mod (x)$
      2.3.1 if the remainder is 1, set $r = 1$
      2.3.2 if the remainder is not 1, set q to the remainder and do the calculation again. r is the number of steps needed for the remainder to become 1
   2.4 if r is odd, go back and choose a different k. if even, move on
   2.5 define p as the remainder in the $(r/2)$th transformation
      2.5.1 if $p + 1 = x$, choose a new k
      2.5.2 if not, move on
3. the factors or x are $\gcd(p + 1, x)$ and $\gcd(p - 1, x)$

Shor's Algorithm





Quantum Computers & Shor's Algorithm

Quantum computers are getting faster and stronger:

- QCs have allegedly factorized 8, 10, 16, 19, 22, and 48 bit numbers

Quantum Computers & Shor's Algorithm

Quantum computers are getting faster and stronger:

- QCs have allegedly factorized 8, 10, 16, 19, 22, and 48 bit numbers
  - why allegedly?

Quantum Computers & Shor's Algorithm

Quantum computers are getting faster and stronger:

- QCs have allegedly factorized 8, 10, 16, 19, 22, and 48 bit numbers
  - why allegedly?
- No one has actually been able to use Shor's Algorithm

Thank You

Thank you for your attention!

Questions?