RSA (ENCRYPTION)

RSA LÀ GÌ ?

- **RSA (Rivest–Shamir–Adleman)** là một trong những thuật toán mã hóa công khai đầu tiên và được dùng rất rộng rãi để đảm bảo tính bảo mật trong việc truyền dẫn dữ liệu. Và đây là một ứng dụng nổi tiếng nhất của **Hàm Phi Euler.**

Hàm phi (ϕ) Euler của một số nguyên dương n được định nghĩa là số lượng các số nguyên dương m không vượt quá n sao cho $\gcd(m, n) = 1$

TƯ TƯỞNG

Tư tưởng cơ bản của RSA là tìm ba số nguyên cực kỳ lớn và sao cho mô-đun lũy thừa của mọi số nguyên $\boldsymbol{m}$ $(\boldsymbol{0 \leq m < n})$ thỏa mãn

$$(\boldsymbol{m^e})^{\boldsymbol{d}} \equiv \boldsymbol{m} \ (\mathbf{mod}\ \boldsymbol{n})$$

Bạn có biết được $\boldsymbol{e}$ **và** $\boldsymbol{n}$ **(công khai)** thì vẫn rất khó để tìm được $\boldsymbol{d}$ **(bí mật).**

TẠO KHOÁ

Bạn có biết được ***e*** **và** ***n*** **(công khai)** thì vẫn rất khó để tìm được ***d*** **(bí mật).**

- RSA có hai loại khóa: ***khóa công khai*** **(public key)** và ***khóa bí mật*** **(secret key).**

TẠO KHOÁ (tt)

1. Chọn hai **số nguyên tố** khác nhau và p và q

Trên thực tế, hai số p và q là hai số rất rất lớn và chúng được chọn một cách ngẫu nhiên và độc lập. Có ý kiến bảo là độ dài của hai số này nên chênh lệch với nhau ít nhất là vài ký tự để việc truy lại hai số khó hơn

2. Tính $n = q \cdot p$

Theo những lời khuyên hiện tại thì n nên có đâu đó là khoảng **2048 bit**, tức là nếu ta biểu diễn n dưới dạng thập phân thì đâu đó hơn **600 chữ số.**

3. Tính $\phi(n) = (p-1) \cdot (q-1)$

TẠO KHOÁ (tt)

4. Chọn một số nguyên $\boldsymbol{e}$ $(\boldsymbol{1 < e < \phi(n)})$ và nó là số nguyên tố cùng nhau với $\boldsymbol{\phi(n)}$ (hay nói cách khác là $\boldsymbol{gcd(e, \phi(n)) = 1}$).

Thực tế, ta được khuyên rằng nên chọn một giá trị thuộc một tập các số nguyên tốt dễ nhận biết, và thông thường thì sẽ là 65537. Việc chọn cụ thể một số e như này sẽ không làm giảm độ bảo mật của thuật toán RSA. Mặt khác, giá trị 65537 trong dạng biểu diễn nhị phân chỉ có đúng 2 bit bật, nên nó có thể làm tăng hiệu suất của một số thuật toán lũy thừa nhanh nếu mình lấy lũy thừa của một số với e (tại sao lại lũy thừa cho thì đó là một phần của thuật toán mình sẽ bàn ở phần **Mã hóa**)

5. Tính $\boldsymbol{d}$ sao cho $\boldsymbol{d \cdot e \equiv 1 \left(mod\ \phi(n)\right)}$ (hay nói cách khác d là **Inverse Modulo** của e với modulo $\boldsymbol{\phi(n)}$)

Dùng thuật toán Euclid mở rộng để giải

TẠO KHOÁ (tt)

- Cặp giá trị (e, n) là **khóa công khai.**
- Cặp giá trị (d, n) là **khóa bí mật.**

Theo lý thuyết, mọi người đều có thể biết khóa công khai và dùng nó để mã hóa thông điệp. Tuy nhiên thông điệp đã được mã hóa bởi khóa công khai chỉ có thể được giải mã bởi khóa bí mật.

MÃ HOÁ & GIẢI MÃ

Mã Hoá

Giả sử bên **B muốn chuyển cho bên A một đoạn văn bản M.**

1. Chuyển M thành một số nguyên không âm $\boldsymbol{m < n}$ bằng một hàm có thể đảo ngược và hàm này phải được biết bởi hai bên.
2. Sau khi có được m, n và e, bên B sẽ tính và gửi cho bên A bản mã hóa của m được tính theo công thức

$$\boldsymbol{Encrypt(m) = m^e \bmod n = c}$$

Giải mã

Sau khi A nhận được đoạn mã hóa từ B, A có thể sử dụng d để tìm được m từ c bằng công thức

$$\boldsymbol{Decrypt(c) = c^d \bmod n = m}$$

Biết được m, A lại tiếp tục chuyển đổi từ m sang M theo hàm chuyển đổi đã được thỏa thuận từ trước.

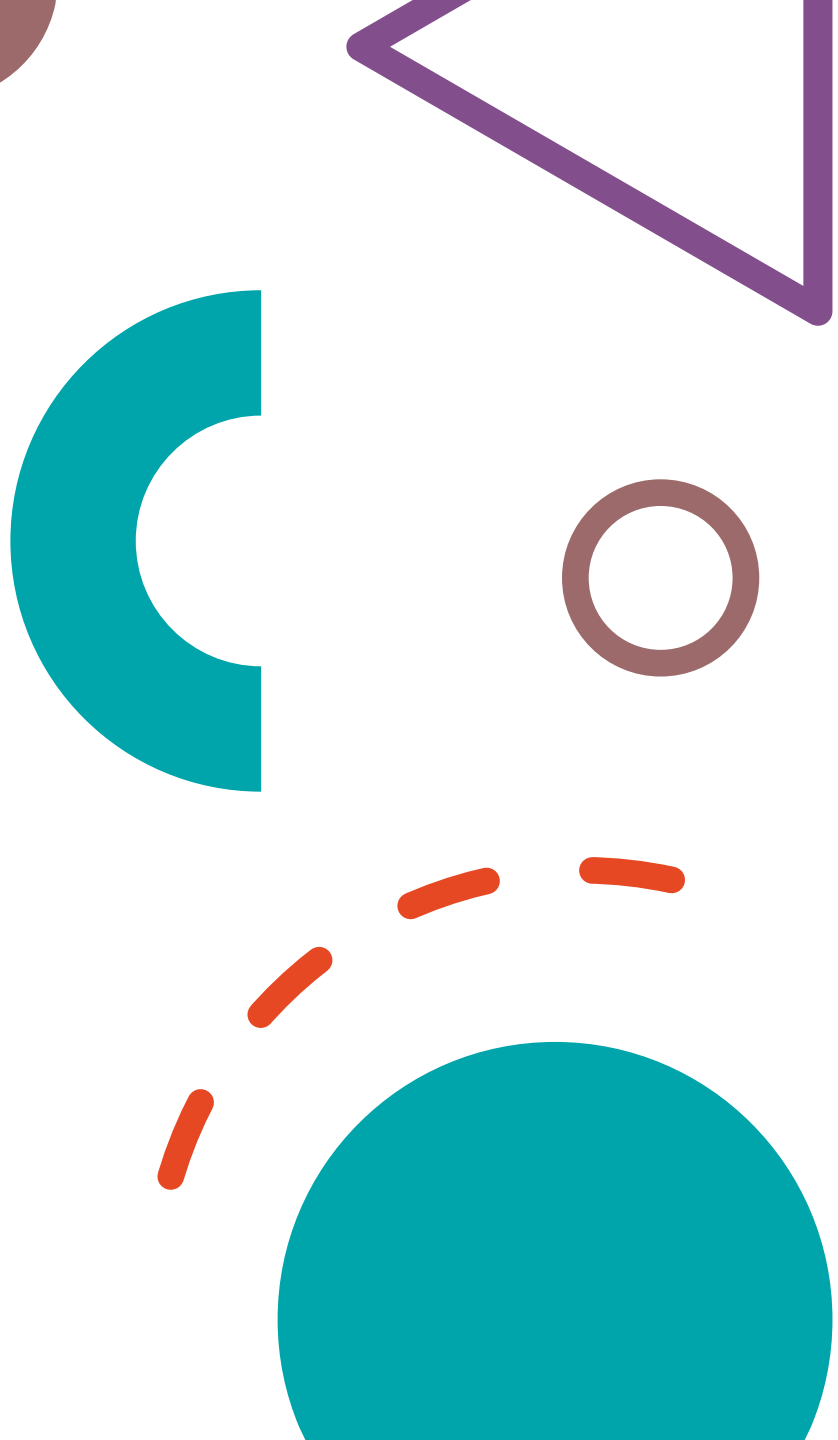
CÀI ĐẶT

Mã hóa RSA

```
function encrypt (pub, m) {
        // padding
        EB = padding(m)
        // encrypt
        tmp = EB^pub.E % pub.N
        // padding output if needed
        c = padding_left_if_needed(tmp)
        return c
}
```

Giải mã RSA

```
function decrypt (pri, c) {
        // decypt block
        tmp = c^pri.D % priv.N
        // unwrap padding
        m = unwrap_padding(tmp)
        return m
}
```



TÍNH THỰC TIỄN

ĐỘ AN TOÀN CỦA RSA

Giả sử bạn bị tin tặc tấn công và bị đánh cắp thông tin về bản tin đã được mã hóa của mình. Như vậy, mình chỉ bị rò rỉ giá trị $c = m^e \bmod n$ và có thể là cả hai giá trị n và e. Tuy nhiên, với việc chỉ có thông tin về văn bản đã được mã hóa mà không có thông tin gì về khóa bí mật, thì liệu tin tặc đó có thể truy ngược về bản tin gốc ban đầu được hay không?

Câu trả lời là được, nhưng cực kỳ khó. Tại sao?

Vì khi bị mất c, và cả n và e thì với những giá trị này, giả sử rằng ta dùng những giá trị và thủ thuật padding đủ tốt, thì tin tặc đó sẽ gần như không thể truy ngược về lại hai số nguyên tố p và q của ta từ n vì cơ bản là nó quá lớn để có thể brute force, từ đó gần như không thể truy được giá trị $\phi(n)$ và dẫn tới việc không thể tính được giá trị d.

Vì thế, cho dù trên thực tế là tên tin tặc hoàn toàn có thể truy ngược về lại bản tin ban đầu nhưng để làm được việc đó là cực kỳ khó với khả năng của máy tính hiện tại. Cho nên đến thời điểm hiện tại, RSA vẫn còn được cho là một trong những thuật toán mã hóa bất đối xứng an toàn và được xử dụng rộng rãi.

ỨNG DỤNG

CHỮ KÝ ĐIỆN TỬ

Bên cạnh việc ứng dụng vào làm mã hóa thuần túy thì RSA cũng có thể dùng để làm chữ ký điện tử. Giải sử thầy Tùng muốn gửi một văn bản mà thầy viết cho thầy Bảo, mà thầy Bảo muốn xác nhận văn bản đó thực sự là của thầy Tùng, thì ta sẽ áp dụng *chữ ký điện tử*.

1. Quá trình tạo key diễn ra hoàn toàn y hệt như RSA mã hóa. Thầy Tùng tạo ra hai khóa như thường lệ và thầy sẽ công bố public key của thầy.
2. Khi thầy Tùng gửi văn bản cho thầy Bảo, thầy Bảo không tin, thầy Bảo challenge thầyTùng, yêu cầu thầy Tùng phải **mã hóa** văn bản thầy Tùng đưa bằng **private key** của thầy Tùng mà chỉ có mình thầy Tùng biết được. Giá trị mã hóa này được gọi là một *chữ ký* và nó được đính kèm văn bản thầy Tùng đưa cho thầy Bảo.
3. Thầy Bảo sau khi nhận được văn bản kèm chữ ký, thầy Bảo sẽ dùng **public key** mà thầy Tùng đã công bố và **giải mã** chữ ký đó. Sau khi giải mã xong, nếu thông điệp giải mã mà thầy Bảo nhận được giống với văn bản mà thầy Tùng gửi thì văn bản đó đã không bị chỉnh sửa và chữ ký hợp lệ.

ĐỌC THÊM

http://people.csail.mit.edu/rivest/Rsapaper.pdf

TÀI LIỆU THAM KHẢO

- *http://people.csail.mit.edu/rivest/Rsapaper.pdf*
- *https://simple.wikipedia.org/wiki/RSA_algorithm*
- *https://en.wikipedia.org/wiki/RSA_(cryptosystem)*
- *https://www.youtube.com/watch?v=4zahvcJ9glG*
- *https://www.youtube.com/watch?v=oOcTVTpUsPQ*
- *https://www.youtube.com/watch?v=9sY57iwNDJw*
- *https://eli.thegreenplace.net/2019/rsa-theory-and-implementation/*

THANK YOU !