Họ và tên : Đinh Quang Hưng

Mã sinh viên: B22DCCN407

Thực hành An toàn bảo mật hệ thống thông tin

Chủ đề: Tìm hiểu Rabin và ElGamel

Tổ 1 - Nhóm 10

Giảng viên hướng dẫn: ThS Quản Trọng Thế

1. **Thuật toán Rabin**
   1. ***Tìm hiểu về thuật toán Rabin***

Thuật toán Rabin là một hệ mã hóa bất đối xứng, được Michael O. Rabin giới thiệu vào năm 1979, dựa trên bài toán phân tích số lớn (factoring problem). Nó hoạt động bằng cách sử dụng phép bình phương modulo, tương tự RSA nhưng có một số đặc điểm khác biệt.

Độ an toàn của Rabin tương đương với khó khăn của việc phân tích n thành p và q, một bài toán được coi là khó về mặt tính toán nếu n đủ lớn. Tuy nhiên, việc giải mã tạo ra bốn nghiệm, đòi hỏi cơ chế dư thừa để chọn nghiệm đúng, làm tăng độ phức tạp so với RSA. Rabin nhanh hơn RSA trong mã hóa, nhưng ít được sử dụng trong mã hóa trực tiếp do phức tạp trong giải mã.

Thuật toán Rabin phổ biến trong các ứng dụng chữ ký số, nơi tính chất bình phương modulo giúp xác minh tính toàn vẹn và nguồn gốc thông điệp. Một hạn chế là Rabin dễ bị tấn công nếu không có padding, ví dụ: tấn công chosen-ciphertext có thể khai thác bốn nghiệm để suy ra thông điệp. Thuật toán này yêu cầu chọn số nguyên tố an toàn và quản lý khóa cẩn thận để đảm bảo tính bảo mật.

* 1. ***Cách hoạt động***

Thuật toán Rabin sử dụng phép bình phương module và hoạt động của bao gồm 3 giai đoạn chính là : sinh khóa, mã hóa và giải mã

* **Sinh khóa**:
  + Chọn hai số nguyên tố lớn p và q, thường có kích thước 512-bit trở lên để đảm bảo an toàn. Để đơn giản hóa tính toán, p và q thường được chọn sao cho (tức là số nguyên tố dạng ).
  + Tính . Số n là khóa công khai, được chia sẻ công khai, trong khi cặp (p, q) là khóa bí mật, chỉ người nhận giữ.
  + Độ an toàn của thuật toán dựa trên việc phân tích n thành p và q là cực kỳ khó nếu p và q đủ lớn, tương đương với bài toán phân tích số lớn – một vấn đề chưa có thuật toán hiệu quả trên máy tính cổ điển.
* **Mã hóa:**
  + Với thông điệp m (m < n), người gửi tính bản mã. Phép bình phương modulo này rất nhanh, giúp Rabin hiệu quả hơn RSA trong giai đoạn mã hóa.
  + Để đảm bảo an toàn và tránh giả mạo, thông điệp m thường được thêm padding trước khi mã hóa. Padding giúp thông điệp có cấu trúc đặc biệt, hỗ trợ việc chọn nghiệm đúng khi giải mã.
  + Bản mã c được gửi đến người nhận qua kênh không an toàn.
* **Giải mã:**
  + Người nhận, với khóa bí mật (p, q), cần tìm căn bậc hai của c modulo n, tức là tìm m sao cho .
  + Do , việc này tương đương với tìm căn bậc hai của c modulo p và modulo q, rồi kết hợp các nghiệm bằng định lý số dư Trung Hoa.
  + Cụ thể, tính:
    - và , sử dụng thuật toán căn bậc hai modulo số nguyên tố
    - Kết hợp và để tìm bốn nghiệm ±m1, ±m2 modulo n (do phép bình phương tạo ra bốn căn bậc hai).
  + Bốn nghiệm này là vấn đề chính của Rabin: chỉ một nghiệm là thông điệp gốc m. Để chọn đúng, thông điệp gốc phải có cấu trúc đặc biệt (từ padding) hoặc thêm thông tin dư thừa (như một chuỗi bit cố định).
  + Nếu không có padding, kẻ tấn công có thể khai thác bốn nghiệm để thực hiện tấn công chosen-ciphertext, làm giảm an toàn.

1. **Thuật toán ElGamal**
   1. ***Tìm hiểu về thuật toán Rabin***

Thuật toán ElGamal, được Taher ElGamal phát triển vào năm 1985, là một hệ mã hóa bất đối xứng dựa trên bài toán logarit rời rạc (discrete logarithm problem) trong nhóm cyclic. Nó được thiết kế để cung cấp tính bảo mật cao cho mã hóa và chữ ký số, trở thành nền tảng cho nhiều giao thức bảo mật như Diffie-Hellman và DSA (Digital Signature Algorithm).

ElGamal hoạt động trong nhóm Zp\*, với p là số nguyên tố lớn, và sử dụng các phép lũy thừa modulo để mã hóa và giải mã. Độ an toàn của nó dựa trên việc tính logarit rời rạc (tìm a từ ) là rất khó nếu p đủ lớn, ngay cả với máy tính hiện đại.

Thuật toán này linh hoạt, cho phép sử dụng trong cả mã hóa thông điệp và tạo chữ ký số, nhưng nhược điểm là bản mã lớn gấp đôi thông điệp, khiến nó kém hiệu quả hơn RSA trong một số trường hợp. ElGamal được ứng dụng rộng rãi trong các hệ thống như GPG (GNU Privacy Guard), giao thức SSL/TLS, và blockchain.

So với RSA, ElGamal có cơ sở lý thuyết mạnh hơn vì bài toán logarit rời rạc chưa được chứng minh tương đương với một bài toán dễ hơn. Tuy nhiên, nó yêu cầu chọn số nguyên tố p lớn và số ngẫu nhiên k an toàn để tránh các cuộc tấn công. ElGamal cũng dễ bị tấn công nếu khóa tạm thời k bị lặp lại, do đó cần cơ chế sinh số ngẫu nhiên mạnh. Trong thực tế, ElGamal thường được dùng cho chữ ký số hơn mã hóa trực tiếp do kích thước bản mã và tốc độ xử lý.

* 1. ***Cách hoạt động***

Thuật toán ElGamal hoạt động qua ba giai đoạn chính: sinh khóa, mã hóa, và giải mã, dựa trên các phép tính lũy thừa modulo trong nhóm cyclic Zp\*. Dưới đây là chi tiết từng bước:

1. **Sinh khóa**:

* Chọn một số nguyên tố lớn p (thường 1024-bit hoặc hơn) để tạo nhóm Zp\*. Số p phải đủ lớn để bài toán logarit rời rạc trở nên bất khả thi.
* Chọn một số sinh g của Zp\*, tức là g khi lũy thừa modulo p sẽ tạo ra tất cả các phần tử từ 1 đến p-1.
* Chọn khóa bí mật a ngẫu nhiên, 1 < a < p-1, và tính
* Khóa công khai là (p, g, h), được chia sẻ công khai. Khóa bí mật là a, chỉ người nhận giữ.
* Độ an toàn dựa trên khó khăn của việc tính a từ, gọi là bài toán logarit rời rạc.

1. **Mã hóa:**
   * Với thông điệp m (0 < m < p), người gửi chọn một số ngẫu nhiên k (1 < k < p-1), gọi là khóa tạm thời (ephemeral key). Số k phải được chọn mới cho mỗi lần mã hóa để tránh lặp lại.
   * Tính y1 = g^k mod p, đại diện cho phần thông tin về khóa tạm thời.
   * Tính , mã hóa thông điệp bằng khóa công khai h.
   * Bản mã là cặp (y1, y2), được gửi đến người nhận qua kênh không an toàn.
   * Kích thước bản mã gấp đôi thông điệp, do đó ElGamal tiêu tốn nhiều băng thông hơn RSA hoặc Rabin.
2. **Giải mã:**
   * Người nhận dùng khóa bí mật a để tính
   * Tính nghịch đảo modulo của s, tức là , sử dụng thuật toán Euclid mở rộng.
   * Tính thông điệp gốc m = y2 \* s^(-1) mod p. Vì
   * Quá trình giải mã yêu cầu tính lũy thừa và nghịch đảo modulo, đòi hỏi hiệu suất tính toán cao.
3. **Mô hình, triển khai và mô phỏng chữ ký số Rabin/ElGamal**
   1. ***Mô hình chữ ký số***

Chữ ký số là cơ chế sử dụng mã hóa bất đối xứng để đảm bảo tính **toàn vẹn, xác thực nguồn,** và **không thể chối bỏ** của thông điệp. Mô hình chung bao gồm:

* ***Người gửi (Sender)****:* Sinh cặp khóa (khóa công khai, khóa bí mật), ký thông điệp bằng khóa bí mật, gửi thông điệp và chữ ký đến người nhận.
* ***Người nhận (Recipient)***: Dùng khóa công khai để kiểm tra chữ ký, xác minh thông điệp không bị thay đổi và đúng từ Sender.
* ***Quy trình:***
  1. Sender băm thông điệp (dùng hàm băm như SHA-256) để tạo giá trị băm H(m).
  2. Sender ký *H(m)* bằng khóa bí mật, tạo chữ ký s.
  3. Sender gửi thông điệp m và chữ ký s (hoặc chỉ gửi s nếu m được biết trước).
  4. Recipient băm lại m để được *H(m)* , dùng khóa công khai để kiểm tra s, xác minh H(m) khớp với giá trị từ s.
* ***Mục tiêu***: Đảm bảo thông điệp không bị sửa đổi (toàn vẹn) và đúng từ Sender (xác thực).
* ***Ứng dụng***: Chữ ký số Rabin/ElGamal được dùng trong email bảo mật, blockchain, và giao thức mạng.
* ***Yêu cầu***: Hàm băm an toàn (SHA-256), khóa đủ lớn, và kênh truyền (SSH) bảo mật.
* ***Kịch bản mô phỏng***: Hai máy Ubuntu 16.04 LTS (Sender và Recipient) giao tiếp qua SSH, Sender ký thông điệp và gửi, Recipient kiểm tra.
  1. ***Mô hình Rabin***
* **Sinh khóa**: Chọn p, q (số nguyên tố lớn, 512-bit, p ≡ q ≡ 3 mod 4), tính *n = p \* q*. Khóa công khai: n; khóa bí mật: (p, q).
* **Ký**: Băm thông điệp m bằng SHA-256 để được H(m), tính chữ ký (dùng khóa bí mật).
* **Kiểm tra**: Recipient băm m để được H(m), kiểm tra *H(m) ≡ s² mod n* (dùng khóa công khai).
* **Lưu ý**: Cần padding H(m) để H(m) là số chính phương modulo n, tránh giả mạo.
  1. ***Mô hình ElGamal***
* **Sinh khóa**: Chọn p (số nguyên tố lớn, 1024-bit), g (số sinh), a (khóa bí mật), tính . Khóa công khai: (p, g, h); khóa bí mật: a.
* **Ký**: Băm m bằng SHA-256 để được H(m), chọn k ngẫu nhiên (k và p-1 nguyên tố cùng nhau), tính , . Chữ ký là (r, s).
* **Kiểm tra**: Recipient băm m để được H(m), kiểm tra .
* **Lưu ý**: k phải ngẫu nhiên, an toàn, và không lặp lại.
  1. ***Chạy mô phỏng chữ ký số Rabin/ElGamal***

**Kịch bản mô phỏng** trên hai máy Ubuntu 16.04 LTS:

* **Môi trường**: Cả Sender và Recipient cài Ubuntu 16.04 LTS (ISO ~1.6 GB)
* **Thiết lập SSH**:
  + Cài OpenSSH: sudo apt install openssh-server -y.
  + Sender tạo khóa SSH: ssh-keygen -t rsa, gửi khóa công khai: ssh-copy-id recipient@<IP\_Recipient>.
  + Kiểm tra: ssh recipient@<IP\_Recipient> và scp test.txt recipient@<IP\_Recipient>:~/receive/.
* **Quy trình:**
  + **Sender sinh khóa**: Chạy script để tạo khóa Rabin/ElGamal, lưu khóa công khai vào pubkey\_\*.txt.
  + **Sender ký thông điệp**: Băm thông điệp (ví dụ: “Hello World”), ký, lưu thông điệp và chữ ký vào message\_\*.txt.
  + **Sender gửi file**: Dùng scp pubkey\_\*.txt message\_\*.txt recipient@<IP\_Recipient>:~/receive/.
  + **Recipient kiểm tra**: Chạy script đọc khóa công khai, thông điệp, chữ ký; băm lại m; kiểm tra chữ ký.
  + **Kết quả**: In “Chữ ký hợp lệ, thông điệp toàn vẹn” hoặc “Chữ ký không hợp lệ”.

**Chạy trên ubuntu**:

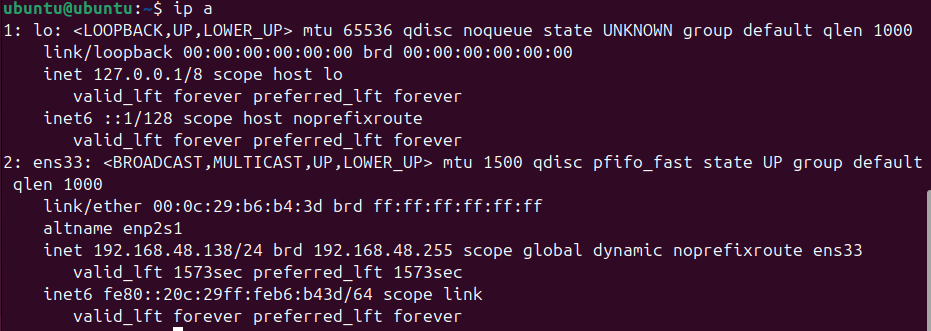
Thực hiện cài đặt OpenSSH và OpenSSL trên cả hai máy:





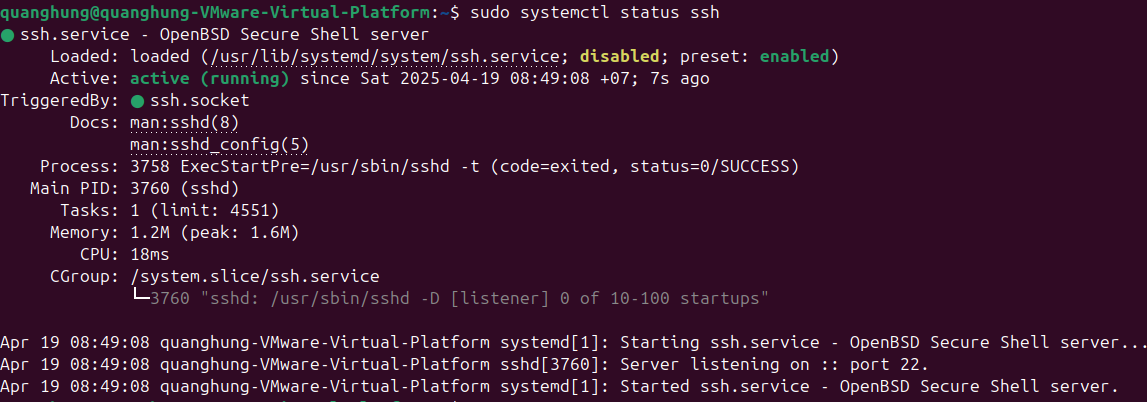
Thực hiện lấy địa chỉ IP máy Recipient và kết nối ssh sender.

+ Máy recipient:



+ Máy sender:

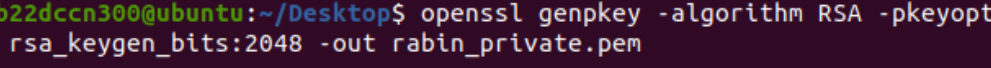
* + Kiểm tra trạng thái dịch vụ SSH





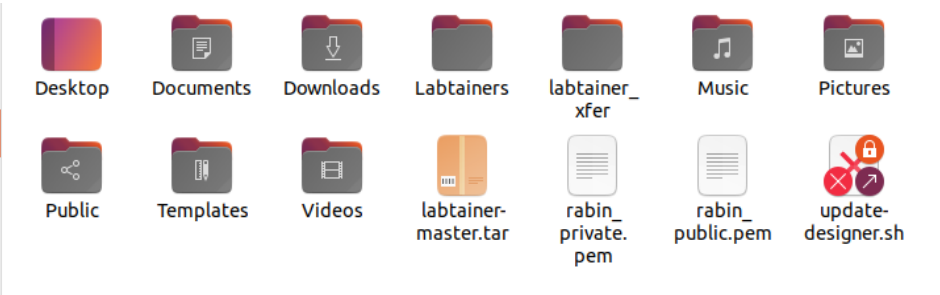
Thuật toán Rabin

- Tạo cặp khóa Rabin (private key và public key) trên Sender:

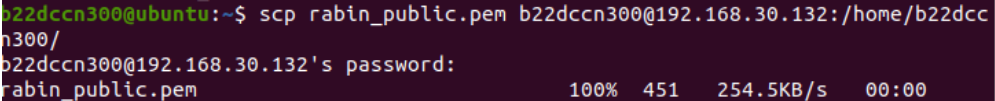




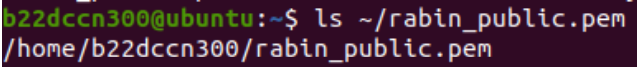
- Khi đó xuất hiện hai file khóa rabin\_private.pem và rabin\_public trên trang home:



- Thực hiện trên máy Sender gửi file sang máy Recipient với lệnh:

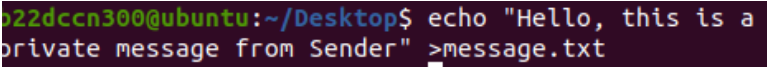


- Kiểm tra trên máy Recipient xem đã nhận được file chưa:

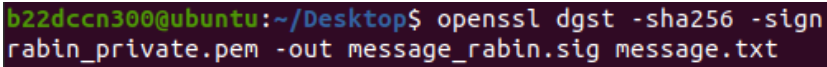


- Trên máy Sender:

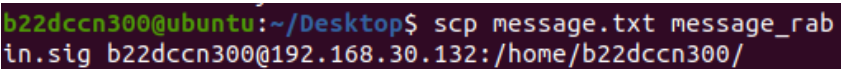
+ Tạo file thông điệp:



+Ký số bằng private key Rabin:

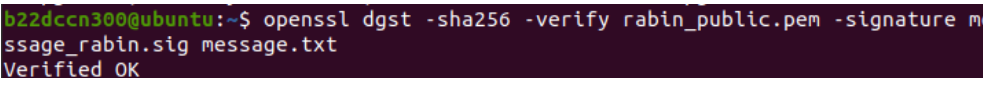


+Thực hiện gửi thông điệp và chữ ký cho Recipient:



- Trên máy Recipient:

+ Kiểm tra chữ ký Rabin, nếu kết quả trả về Verified OK → Thông điệp toàn vẹn và đúng nguồn gốc.



3. Thuật toán ElGamal

- OpenSSL không hỗ trợ trực tiếp ElGamal, do đó ta cần thực hiện mô phỏng bằng cách sử dụng tham số DH.

- Thực hiện tạo cặp khóa ElGamal trên máy Sender:

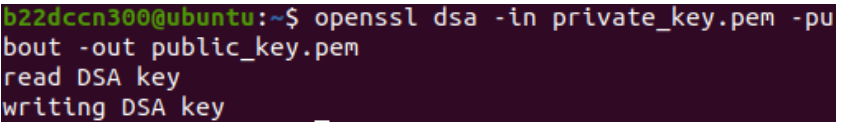
+ Tạo tham số DH: tương đương tạo p và alpha



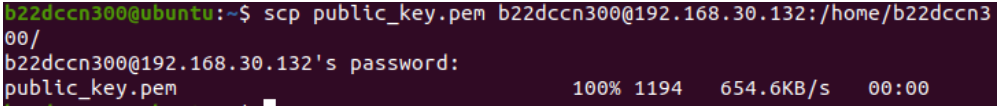
+ Tạo private key: tương đương tạo a



+ Trích xuất public key: 𝑏𝑒𝑡𝑎 = 𝑎𝑙𝑝ℎ𝑎 𝑎 𝑚𝑜𝑑 𝑝

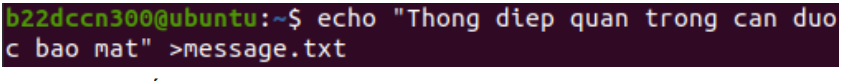


- Sender thực hiện gửi public key cho Recipient qua SSH:

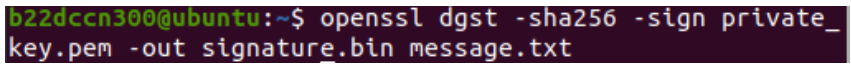


- Trên máy Sender:

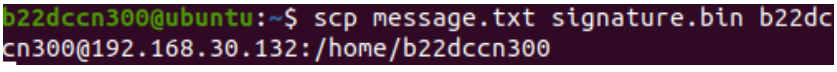
+ Tạo thông điệp:



+ Tạo chữ ký số:



+ Gửi thông điệp và chữ ký cho Recipient:



- Trên máy Recipient:

Kiểm tra chữ ký ElGamal, nếu kết quả trả về Verified OK → Thông điệp toàn vẹn và đúng nguồn gốc

