

# NGHIÊN CỨU KIẾN TRÚC VÀ XÂY DỰNG HỆ THỐNG CHỨNG THỰC TẬP TRUNG

## Mục lục

|  |           |
|--|-----------|
| <b>Chương 1 Mở đầu.....</b>                            | <b>1</b>  |
| 1.1 Giới thiệu tổng quan.....                          | 1         |
| 1.2 Tình hình triển khai.....                          | 3         |
| 1.2.1 Thế giới.....                                    | 3         |
| 1.2.2 Việt Nam.....                                    | 5         |
| 1.3 Nhu cầu thực tế.....                               | 7         |
| 1.4 Mục tiêu của đề tài.....                           | 8         |
| 1.5 Nội dung của luận văn.....                         | 9         |
| <b>Chương 2 Chữ ký số.....</b>                         | <b>11</b> |
| 2.1 Giới thiệu.....                                    | 11        |
| 2.1.1 Nhu cầu thực tế.....                             | 11        |
| 2.1.2 Khái niệm.....                                   | 13        |
| 2.1.3 Các dịch vụ bảo mật.....                         | 14        |
| 2.1.4 Nguyên lý hoạt động của chữ ký số.....           | 15        |
| 2.2 Thuật toán hàm băm mật mã.....                     | 17        |
| 2.2.1 Giới thiệu.....                                  | 17        |
| 2.2.2 Một số hàm băm mật mã thông dụng.....            | 18        |
| 2.2.3 Kết quả thử nghiệm và nhận xét.....              | 24        |
| 2.3 Thuật toán chữ ký số.....                          | 29        |
| 2.3.1 Giới thiệu.....                                  | 29        |
| 2.3.2 Một số thuật toán chữ ký số thông dụng.....      | 29        |
| 2.3.3 Kết quả thử nghiệm và nhận xét.....              | 34        |
| 2.4 Kết luận.....                                      | 39        |
| <b>Chương 3 Tổ chức chứng nhận khóa công khai.....</b> | <b>41</b> |
| 3.1 Giới thiệu.....                                    | 41        |
| 3.2 Chứng nhận số.....                                 | 42        |
| 3.2.1 Các loại chứng nhận.....                         | 42        |
| 3.2.2 Chu kỳ sống của chứng nhận số.....               | 46        |
| 3.3 Các chức năng chính.....                           | 47        |
| 3.3.1 Khởi tạo.....                                    | 47        |
| 3.3.2 Yêu cầu chứng nhận.....                          | 47        |
| 3.3.3 Tạo lại chứng nhận.....                          | 49        |
| 3.3.4 Hủy bỏ chứng nhận.....                           | 49        |
| 3.3.5 Lưu trữ và phục hồi khóa.....                    | 50        |
| 3.4 Kết luận.....                                      | 51        |

|  |           |
|--|-----------|
| <b>Chương 4 Hạ tầng khóa công khai.....</b>                                | <b>52</b> |
| 4.1 Giới thiệu.....  | 52        |
| 4.1.1 Khái niệm.....   | 52        |
| 4.1.2 Vai trò và chức năng.....  | 53        |
| 4.1.3 Các thành phần của một hạ tầng khóa công khai.....                   | 55        |
| 4.2 Các kiến trúc PKI.....   | 59        |
| 4.2.1 Kiến trúc CA đơn.....  | 61        |
| 4.2.2 Kiến trúc danh sách tín nhiệm.....                                   | 63        |
| 4.2.3 Kiến trúc phân cấp.....  | 65        |
| 4.2.4 Kiến trúc lưới.....  | 68        |
| 4.2.5 Kiến trúc lai.....   | 70        |
| 4.2.6 Nhận xét.....  | 76        |
| 4.3 Kết luận.....  | 77        |
| <b>Chương 5 Phân tích một số nguy cơ tổn thương trong hệ mã RSA.....</b>   | <b>79</b> |
| 5.1 Tổng quan về hệ mã RSA.....  | 79        |
| 5.1.1 Giới thiệu.....  | 79        |
| 5.1.2 Thuật toán.....  | 80        |
| 5.1.3 Các ứng dụng quan trọng.....   | 81        |
| 5.2 Nguy cơ tổn thương của hệ mã trước các tấn công và cách khắc phục..... | 82        |
| 5.2.1 Tổn thương do các tấn công phân tích ra thừa số nguyên tố.....       | 83        |
| 5.2.2 Tổn thương do bản thân hệ mã.....                                    | 87        |
| 5.2.3 Tổn thương do lạm dụng hệ mã.....                                    | 88        |
| 5.2.4 Tổn thương do sử dụng số mũ bí mật nhỏ.....                          | 90        |
| 5.2.5 Tổn thương do sử dụng số mũ công khai nhỏ.....                       | 90        |
| 5.2.6 Tổn thương do khai thác thời gian thực thi.....                      | 94        |
| 5.3 Kết luận.....  | 94        |
| <b>Chương 6 Một số bài toán quan trọng trong hệ mã RSA.....</b>            | <b>97</b> |
| 6.1 Nhu cầu.....   | 97        |
| 6.2 Bài toán tính toán nhanh trên số lớn.....                              | 97        |
| 6.3 Bài toán phát sinh số ngẫu nhiên.....                                  | 100       |
| 6.4 Bài toán kiểm tra tính nguyên tố của một số nguyên.....                | 101       |
| 6.4.1 Giới thiệu.....  | 101       |
| 6.4.2 Một số thuật toán kiểm tra tính nguyên tố theo xác suất.....         | 102       |
| 6.4.3 Nhận xét.....  | 104       |
| 6.5 Bài toán phát sinh số nguyên tố.....                                   | 106       |
| 6.5.1 Giới thiệu.....  | 106       |
| 6.5.2 Phát sinh số khả nguyên tố.....                                      | 107       |
| 6.5.3 Phát sinh số nguyên tố.....  | 111       |
| 6.5.4 Nhận xét.....  | 112       |
| 6.6 Kết luận.....  | 112       |

|   |            |
|---|------------|
| <b>Chương 7 Xây dựng bộ thư viện “SmartRSA”, cài đặt hiệu quả hệ mã RSA.....</b>                                      | <b>113</b> |
| 7.1 Giới thiệu.....   | 113        |
| 7.2 Các thuật toán và chức năng được cung cấp trong thư viện.....   | 113        |
| 7.3 Một số đặc tính của bộ thư viện.....  | 114        |
| 7.4 Kết quả thử nghiệm và nhận xét.....   | 114        |
| 7.4.1 Các thuật toán tính nhanh lũy thừa modulo.....  | 115        |
| 7.4.2 Các thuật toán kiểm tra tính nguyên tố theo xác suất.....   | 116        |
| 7.4.3 Các thuật toán phát sinh số nguyên tố.....  | 120        |
| 7.5 Kết luận.....   | 124        |
| <b>Chương 8 Cải tiến và triển khai hệ thống chứng thực khóa công khai sử dụng gói phần mềm mã nguồn mở EJBCA.....</b> | <b>125</b> |
| 8.1 Gói phần mềm mã nguồn mở EJBCA.....   | 125        |
| 8.1.1 Giới thiệu.....   | 125        |
| 8.1.2 Kiến trúc.....  | 127        |
| 8.1.3 Chức năng.....  | 128        |
| 8.1.4 So sánh với các gói phần mềm khác.....  | 128        |
| 8.1.5 Lý do chọn gói phần mềm mã nguồn mở EJBCA.....  | 129        |
| 8.2 Cải tiến gói phần mềm mã nguồn mở EJBCA.....  | 130        |
| 8.2.1 Nhu cầu.....  | 130        |
| 8.2.2 Cải tiến bộ sinh khóa RSA của EJBCA.....  | 131        |
| 8.2.3 Nhận xét.....   | 133        |
| 8.3 Triển khai hệ thống.....  | 133        |
| 8.3.1 Mục tiêu.....   | 133        |
| 8.3.2 Mô hình triển khai.....   | 133        |
| 8.3.3 Kết quả triển khai và thử nghiệm.....   | 137        |
| 8.4 Kết luận.....   | 143        |
| <b>Chương 9 Kết luận.....</b>   | <b>144</b> |
| 9.1 Một số kết quả đạt được.....  | 144        |
| 9.2 Hướng phát triển.....   | 145        |
| <b>Tài liệu tham khảo.....</b>  | <b>146</b> |
| <b>Phụ lục A     Tên phân biệt theo chuẩn X.500.....</b>  | <b>151</b> |
| <b>Phụ lục B     Triển khai EJBCA trên môi trường Windows và Linux.....</b>   | <b>152</b> |

## **Danh sách hình**

|   |    |
|---|----|
| Hình 1.1. Thời điểm ban hành các luật liên quan PKI của các quốc gia trên thế giới..... | 6  |
| Hình 2.1. Kiến trúc bảo mật trong TCP/IP.....   | 12 |
| Hình 2.2. Băm dữ liệu.....  | 15 |
| Hình 2.3. Ký nhận một thông điệp rút gọn.....   | 16 |
| Hình 2.4. Kiểm định chữ ký điện tử.....   | 17 |
| Hình 2.5. Tỷ lệ thời gian băm giữa SHA-1 và MD5.....                                    | 25 |
| Hình 2.6. Tỷ lệ thời gian băm giữa SHA-2 và SHA-1.....                                  | 26 |
| Hình 2.7. Tỷ lệ thời gian băm giữa Whirlpool và SHA-512.....                            | 27 |
| Hình 2.8. Tỷ lệ thời gian băm giữa SHA-1 và RIPEMD-160, SHA-256 và RIPEMD-256.....      | 28 |
| Hình 2.9. Thời gian tạo khóa của RSA và DSA.....  | 35 |
| Hình 2.10. Thời gian tạo chữ ký của RSA và DSA.....                                     | 36 |
| Hình 2.11. Thời gian xác nhận chữ ký của RSA và DSA.....                                | 36 |
| Hình 3.1. Phiên bản 3 của chứng nhận X.509.....   | 43 |
| Hình 3.2. Phiên bản 2 của cấu trúc chứng nhận thuộc tính.....                           | 45 |
| Hình 3.3. Chu kỳ sống của chứng nhận.....   | 46 |
| Hình 3.4. Mẫu yêu cầu chứng nhận theo chuẩn PKCS #10.....                               | 47 |
| Hình 3.5. Định dạng thông điệp yêu cầu chứng nhận theo RFC 2511.....                    | 48 |
| Hình 3.6. Phiên bản 2 của định dạng danh sách chứng nhận bị hủy.....                    | 50 |
| Hình 4.1. Các thành phần của một hạ tầng khóa công khai.....                            | 55 |
| Hình 4.2. Mô hình chuỗi tín nhiệm.....  | 59 |
| Hình 4.3. Kiến trúc CA đơn.....   | 61 |
| Hình 4.4. Đường dẫn chứng nhận trong kiến trúc CA đơn.....                              | 62 |
| Hình 4.5. Kiến trúc danh sách tín nhiệm.....  | 63 |
| Hình 4.6. Đường dẫn chứng nhận trong kiến trúc danh sách tín nhiệm.....                 | 64 |
| Hình 4.7. Kiến trúc PKI phân cấp.....   | 65 |
| Hình 4.8. Đường dẫn chứng nhận trong kiến trúc PKI phân cấp.....                        | 66 |
| Hình 4.9. Mở rộng kiến trúc PKI phân cấp.....   | 67 |
| Hình 4.10. Kiến trúc lưới.....  | 68 |

|   |     |
|---|-----|
| Hình 4.11. Đường dẫn chứng nhận trong kiến trúc lưới.....   | 69  |
| Hình 4.12. Các PKI được triển khai ở các tổ chức khác nhau.....   | 71  |
| Hình 4.13. Kiến trúc danh sách tín nhiệm mở rộng.....   | 72  |
| Hình 4.14. Kiến trúc chứng nhận chéo.....   | 73  |
| Hình 4.15. Kiến trúc CA cầu nối.....  | 74  |
| Hình 4.16. Kiến trúc Gateway CA.....  | 75  |
| Hình 7.1. Thời gian thực hiện của các thuật toán tính lũy thừa modulo.....                                | 116 |
| Hình 7.2. Thời gian kiểm tra tính nguyên tố với $p_{k,t} \leq \frac{1}{2}^{80}$ khi thử nghiệm trên ..... | 117 |
| hợp số ngẫu nhiên.....  |     |
| Hình 7.3. Thời gian kiểm tra tính nguyên tố với $p_{k,t} \leq \frac{1}{2}^{80}$ khi thử nghiệm trên ..... | 118 |
| số nguyên tố ngẫu nhiên.....  |     |
| Hình 7.4. Tỷ lệ thời gian kiểm tra giữa thuật toán Miller-Rabin cải tiến.....                             |     |
| và thuật toán Miller-Rabin gốc với $p_{k,t} \leq \frac{1}{2}^{80}$ .....                                  | 120 |
| Hình 7.5. Thời gian phát sinh số khả nguyên tố ngẫu nhiên.....  | 121 |
| Hình 7.6. Tỷ lệ thời gian phát sinh số nguyên tố của thuật toán Gordon.....                               |     |
| và thuật toán tìm kiếm ngẫu nhiên.....  | 122 |
| Hình 7.7. Tỷ lệ thời gian phát sinh số nguyên tố giữa thuật toán Maurer.....                              |     |
| và thuật toán tìm kiếm ngẫu nhiên.....  | 124 |
| Hình 8.1. Kiến trúc EJBCA.....  | 127 |
| Hình 8.2. Hàm phát sinh khóa RSA của EJBCA.....   | 132 |
| Hình 8.3. Mô hình triển khai hệ thống chứng thực tại Khoa CNTT, .....                                     |     |
| trường ĐH KHTN, Tp.HCM.....   | 134 |
| Hình 8.4. Giao diện trang web của CA KCNTT (CA gốc).....  | 137 |
| Hình 8.5. Giao diện trang web của CA BMCNPM.....  | 137 |
| Hình 8.6. Chứng nhận cho người dùng tên “SuperAdmin (BMTHCS)”.....  | 138 |
| Hình 8.7. Giao diện quản trị toàn quyền của người dùng “SuperAdmin (BMTHCS)”.....                         | 138 |
| Hình 8.8. Giao diện quản trị CA của người dùng “CAAdmin (BMTHCS)”.....                                    | 139 |
| Hình 8.9. Giao diện quản trị RA của người dùng “RAAdmin (BMTHCS)”.....                                    | 139 |
| Hình 8.10. Giao diện giám sát của người dùng “Supervisor (BMTHCS)”.....                                   | 140 |

|   |     |
|---|-----|
| Hình 8.11. Nội dung chứng nhận của người dùng.....                            | 140 |
| Hình 8.12. Lỗi không thể đọc được thư đã ký và mã hóa.....                    | 141 |
| Hình 8.13. Nội dung thư được ký và nội dung thư được ký đồng thời mã hóa..... | 141 |
| Hình 8.14. Ký và xác nhận chữ ký trong tài liệu điện tử PDF.....              | 142 |
| Hình 8.15. Tài liệu điện tử PDF được ký đã bị thay đổi.....                   | 142 |
| Hình 9.1. Ví dụ về tên phân biệt theo chuẩn X.500.....                        | 151 |

## Danh sách bảng

|  |  |
|--|--|
| Bảng 2.1. Đặc điểm của các thuật toán băm SHA.....                               | 21   |
| Bảng 2.2. Thời gian băm của MD5 và SHA-1.....                                    | 25   |
| Bảng 2.3. Thời gian băm của SHA-1 và SHA-224/256/384/512.....                    | 26   |
| Bảng 2.4. Thời gian băm của SHA-512 và Whirlpool.....                            | 27   |
| Bảng 2.5. Thời gian băm của SHA-1 và RIPEMD-160, SHA-256 và RIPEMD-256.....      | 28   |
| Bảng 2.6. So sánh thời gian tạo khóa, tạo chữ ký và xác nhận chữ ký.....         | 35   |
| Bảng 2.7. So sánh kích thước khóa RSA và ECC với cùng độ an toàn.....            | 37   |
| Bảng 2.8. So sánh tốc độ tạo và xác nhận chữ ký của RSA và ECDSA.....            | 38   |
| Bảng 2.9. Khuyến cáo trong sử dụng các thuật toán hàm băm mật mã SHA.....        | 39   |
| Bảng 4.1. So sánh các kiến trúc PKI đơn giản.....                                | 76   |
| Bảng 4.2. So sánh các kiến trúc PKI lai.....                                     | 77   |
| Bảng 5.1. Thời gian phân tích ra thừa số nguyên tố của một số lớn.....           | 82   |
| Bảng 7.1. Thời gian thực hiện của các thuật toán tính lũy thừa modulo.....       | 115  |
| Bảng 7.2. Thời gian kiểm tra tính nguyên tố với $p$ trên hợp số ngẫu nhiên.....  | $\frac{1}{k,t} \leq \frac{80}{2}$ khi thử nghiệm .....     |
|  | 117  |
| Bảng 7.3. Thời gian kiểm tra tính nguyên tố với $p$ số nguyên tố ngẫu nhiên..... | $\frac{1}{k,t} \leq \frac{80}{2}$ khi thử nghiệm trên..... |
|  | 118  |
| Bảng 7.4. Thời gian kiểm tra của các thuật toán Miller-Rabin với $p$             | $\frac{1}{k,t} \leq \frac{80}{2}$ ..... 119                |
| Bảng 7.5. Thời gian phát sinh số khả nguyên tố ngẫu nhiên.....                   | 121  |
| Bảng 7.6. Thời gian phát sinh số khả nguyên mạnh bằng thuật toán Gordon.....     | 122  |
| Bảng 7.7. Thời gian phát sinh số nguyên tố bằng thuật toán Maurer.....           | 123  |
| Bảng 8.1. So sánh các đặc điểm của EJBCA và OpenCA.....                          | 129  |
| Bảng 8.2. Tên của các CA theo chuẩn X.500.....                                   | 135  |

## **Danh sách thuật toán**

|  |     |
|--|-----|
| Thuật toán 6.1. Tính lũy thừa modulo bằng thuật toán nhị phân.....                     | 98  |
| Thuật toán 6.2. Tính lũy thừa modulo bằng thuật toán sử dụng định lý số dư Trung Hoa.. | 100 |
| Thuật toán 6.3. Kiểm tra tính nguyên tố theo xác suất Fermat.....                      | 102 |
| Thuật toán 6.4. Kiểm tra tính nguyên tố theo xác suất Solovay-Strassen.....            | 103 |
| Thuật toán 6.5. Kiểm tra tính nguyên tố theo xác suất Miller Rabin.....                | 104 |
| Thuật toán 6.6. Phát sinh số khả nguyên tố kiểu tìm kiếm ngẫu nhiên.....               | 107 |
| Thuật toán 6.7. Phát sinh số khả nguyên tố kiểu tìm kiếm tăng.....                     | 107 |
| Thuật toán 6.8. Phát sinh số khả nguyên tố kiểu tìm kiếm tăng cải tiến.....            | 108 |
| Thuật toán 6.9. Phát sinh số khả nguyên tố mạnh đơn giản.....                          | 109 |
| Thuật toán 6.10. Phát sinh số khả nguyên tố mạnh Williams/Chim.....                    | 109 |
| Thuật toán 6.11. Phát sinh số khả nguyên tố mạnh Gordon.....                           | 110 |
| Thuật toán 6.12. Phát sinh số nguyên tố Maurer.....                                    | 111 |
| Thuật toán 8.1. Phát sinh cặp khóa RSA trong Bouncy Castle.....                        | 132 |

### **Danh sách các ký hiệu, các từ viết tắt**

| Viết tắt | Ý nghĩa   |
|----------|---|
| CA       | Certificate Authority: Tổ chức chứng nhận                     |
| CRL      | Certificate Revocation List: Danh sách hủy bỏ chứng nhận      |
| DN       | Distinguished Name: Tên phân biệt                             |
| PKCS     | Public Key Cryptography Standard: Chuẩn mã hóa khóa công khai |
| PKI      | Public Key Infrastructure: Hạ tầng khóa công khai             |
| SHA      | Secure Hash Algorithm: Thuật toán băm an toàn                 |
| SHS      | Secure Hash Standard: Chuẩn băm an toàn                       |

## Chương 1

### Mở đầu

- Nội dung của chương này trình bày tổng quan về hạ tầng khóa công khai (PKI), giới thiệu khái quát về tình hình triển khai và nhu cầu sử dụng PKI trong thực tế, đồng thời nêu lên mục tiêu, nội dung và ý nghĩa của đề tài.

#### 1.1 Giới thiệu tổng quan

Ngày nay, với sự phát triển của hạ tầng truyền thông công nghệ thông tin, việc giao tiếp qua mạng Internet đang trở thành một nhu cầu cần thiết. Hầu hết mọi hoạt động như giao tiếp, giải trí, kinh doanh, ... đều chuyển từ cách thức truyền thống sang môi trường điện tử. Môi trường làm việc này mang đến nhiều cơ hội nhưng cũng nảy sinh rất nhiều vấn đề về an toàn thông tin nghiêm trọng. Hầu hết các thông tin kinh doanh nhạy cảm và quan trọng đều được lưu trữ và trao đổi dưới hình thức điện tử như mã số tài khoản, thông tin mật, ... Nhưng với các thủ đoạn tinh vi, nguy cơ những thông tin này bị đánh cắp qua mạng thật sự là vấn đề đáng quan tâm.

Truyền thông trên Internet chủ yếu sử dụng giao thức TCP/IP. Giao thức này cho phép thông tin được gửi từ máy tính này tới máy tính khác thông qua một loạt các máy trung gian hoặc các mạng riêng biệt nên đã tạo cơ hội cho những kẻ trộm công nghệ cao có thể thực hiện các hành động phi pháp do các thông tin này có thể bị nghe trộm, giả mạo, mạo danh, ... Biện pháp bảo mật hiện nay như dùng mật khẩu đều không đảm bảo vì có thể bị nghe trộm hoặc bị dò ra nhanh chóng do người sử dụng thường chọn mật khẩu ngắn, dễ nhớ, dùng chung và ít khi thay đổi mật khẩu. Mặt khác, do các thông tin điện tử này không được xác thực trong quá trình trao đổi nên khi bị sao chép hay sửa đổi sẽ không thể phát hiện được.

Chữ ký số ra đời đã giải quyết được vấn đề đó. Chữ ký số dựa trên kỹ thuật mã hóa bất đối xứng, trong đó mỗi người có một cặp khóa, một khóa bí mật và một khóa

công khai. Khóa công khai được công bố rộng rãi còn khóa bí mật được giữ kín và không thể tìm được khóa bí mật nếu chỉ biết khóa công khai. Để trao đổi thông tin bí mật, người gửi sử dụng khóa công khai của người nhận để mã hóa thông điệp cần gửi, sau đó người nhận sẽ sử dụng khóa bí mật tương ứng của mình giải mã thông điệp nhận được. Để đảm bảo tính toàn vẹn, chống bị giả mạo hoặc thay đổi nội dung trong quá trình gửi, người gửi sử dụng khóa bí mật của mình để “ký” vào thông điệp cần gửi, sau đó người nhận sẽ sử dụng khóa công khai của người gửi để xác nhận chữ ký trên thông điệp nhận được.

Tuy nhiên, do khóa công khai được trao đổi thoải mái giữa các đối tác nên khi nhận được một khóa công khai do một người khác gửi đến, người nhận thường băn khoăn không biết đây có phải là khóa công khai của chính người mà mình muốn trao đổi hay không. Sự chứng nhận khóa công khai này được thực hiện bởi một tổ chức trung gian thứ ba đáng tin cậy và được gọi là Tổ chức chứng nhận (Certification Authority

– CA). Tổ chức này sẽ cấp cho mỗi người sử dụng một chứng nhận số để xác nhận danh tính người sử dụng và khóa công khai của người này. Chứng nhận số chứa thông tin cá nhân và khóa công khai của người được cấp kèm với chữ ký xác nhận của tổ chức cấp chứng nhận. Với chứng nhận số, người sử dụng có thể mã hóa thông tin một cách hiệu quả, chống giả mạo (cho phép người nhận kiểm tra xem có bị thay đổi không) và xác thực danh tính người gửi. Ngoài ra, chứng nhận số còn là bằng chứng ngăn chặn người gửi chối cãi nguồn gốc tài liệu mình đã gửi.

Cơ cấu tổ chức gồm con người, phần cứng và phần mềm, những chính sách, tiến trình và dịch vụ bảo mật, những giao thức hỗ trợ việc sử dụng mã hóa khóa công khai để phát sinh, quản lý, lưu trữ, phát hành và thu hồi các chứng nhận khóa công khai tạo thành một hạ tầng khóa công khai (Public Key Infrastructure – PKI). Nhu cầu thiết lập hạ tầng có từ cuối những năm 1990, khi các tổ chức công nghiệp và các chính phủ xây dựng các tiêu chuẩn chung dựa trên phương pháp mã hóa để hỗ trợ hạ tầng bảo mật trên mạng Internet. Mục tiêu được đặt ra tại thời điểm đó là xây dựng một bộ tiêu chuẩn bảo mật tổng hợp cùng các công cụ cho phép người sử dụng cũng như các

tổ chức (doanh nghiệp hoặc phi lợi nhuận) có thể tạo lập, lưu trữ và trao đổi các thông tin một cách an toàn trong phạm vi cá nhân và công cộng.

## 1.2 Tình hình triển khai

### 1.2.1 Thế giới

Rất nhiều quốc gia trong khu vực cũng như trên toàn thế giới đã và đang đẩy mạnh ứng dụng công nghệ thông tin nhằm phục vụ cho các hoạt động kinh doanh và đời sống xã hội bằng việc ban hành các bộ luật liên quan đến thương mại điện tử, chữ ký điện tử. Dưới đây là thời điểm ban hành các bộ luật của một số quốc gia trên thế giới [15, tr.35-37]:

#### **Luật thương mại quốc tế của Ủy ban Liên hợp quốc**

- **UNCITRAL:** Luật mẫu về Chữ ký điện tử (2001), có ảnh hưởng lớn đến các bộ luật của các quốc gia trên thế giới.

#### **Châu Mỹ**

- **Canada:** luật Thương mại điện tử thống nhất (1999).
- **Mexico:** luật Thương mại điện tử (2000).
- **Mỹ:** luật Giao dịch điện tử thống nhất (1999), luật Chữ ký điện tử trong thương mại quốc gia và quốc tế (2000).

#### **Châu Âu**

- **Khối EU:** Hướng dẫn số 1999/93/EC của Quốc hội châu Âu (13/12/1999) về khung pháp lý của chữ ký điện tử, Quyết định 2003/511/EC sử dụng 3 thỏa thuận tại hội thảo CEN làm tiêu chuẩn kỹ thuật.
- **Anh, Scotland và Wales:** luật Thông tin điện tử (2000), Chữ ký điện tử (2002).
- **Áo:** luật Chữ ký điện tử (2000).
- **Cộng hòa Czech:** luật Chữ ký điện tử (2000).
- **Cộng hòa Litva:** luật Chữ ký điện tử (2002).
- **Đức:** luật Chữ ký điện tử (2001, chỉnh sửa vào năm 2005).
- **Ireland:** luật Thương mại điện tử (2000).
- **Liên bang Nga:** luật Liên bang về chữ ký số điện tử (10/01/2002)
- **Nauy:** luật Chữ ký điện tử (2001).
- **Rumani:** luật Chữ ký điện tử (2001).
- **Tây Ban Nha:** luật Chữ ký điện tử (2003).

- **Thụy Điển:** luật Chữ ký điện tử (2000).
- **Thụy Sĩ:** luật Liên bang về chứng thực liên quan đến chữ ký điện tử (2003).

### **Châu Đại Dương**

- **New Zealand:** luật Giao dịch điện tử (2002).
- **Úc:** luật Giao dịch điện tử (1999).

### **Châu Á**

- **Ấn Độ:** luật Công nghệ thông tin (6/2000).
- **Đài Loan:** luật Chữ ký điện tử (4/2002).
- **Hàn Quốc:** luật Chữ ký điện tử (2/1999).
- **Hong Kong:** quy định Giao dịch điện tử (2000, chỉnh sửa vào năm 2004).
- **Nhật Bản:** luật Chữ ký số (4/2001).
- **Singapore:** luật Giao dịch điện tử (1998).
- **Thái Lan:** luật Giao dịch điện tử (2001).
- **Trung Quốc:** luật Chữ ký số (8/2004).

### **Châu Phi**

- **Nam Phi:** luật Giao dịch và Thông tin điện tử (2003).

Bên cạnh việc ban hành các bộ luật, các nước cũng đã triển khai thành công các hạ tầng PKI cho toàn quốc gia chứ không phải đơn lẻ cho từng tổ chức. Các số liệu sau đây được ghi nhận cho đến tháng 5/2005 tại một số quốc gia Châu Á [35, tr.4-10]:

- **Hàn Quốc:** có 2 mô hình song song, PKI công cộng (NPKI) và PKI chính phủ (GPKI). NPKI phục vụ cho các doanh nghiệp, lĩnh vực tài chính ngân hàng, công dân và có 6 CA được thừa nhận đã phát hành khoảng 11 triệu chứng nhận. GPKI phục vụ cho khôi chính phủ và còn cung cấp dịch vụ cho các đơn vị hành chính khác. Các PKI được áp dụng cho nhiều lĩnh vực thương mại điện tử như ngân hàng, mua bán trực tuyến, đấu giá điện tử, bảo mật email, ...
- **Trung Quốc:** gồm hai hệ thống PKI chính phủ và PKI công cộng. Theo mô hình này, hệ thống PKI chính phủ chỉ phục vụ giao dịch nội bộ của chính phủ còn hệ thống PKI công cộng chỉ cung cấp dịch vụ cho các đối tượng là công chúng. Tính đến tháng 5/2006, Trung Quốc đã có 77 CA và đã phát hành khoảng 5 triệu chứng nhận được ứng dụng cho mua hàng, thuế, tài chính, ...
- **Nhật Bản:** gồm hai hệ thống PKI chính phủ và PKI công cộng. Các dịch vụ chứng nhận công cộng sử dụng thẻ thông minh cho các cá nhân do PKI công

cộng bắt đầu vào tháng 4/2004. Các lĩnh vực ứng dụng của PKI là các dịch vụ mua bán điện tử, hồ sơ điện tử và chính phủ điện tử.

- **Singapore:** Các lĩnh vực ứng dụng của PKI có thể được phân loại như lĩnh vực chính phủ, hậu cần và tập đoàn như thẻ dịch vụ công cộng cho người dân, thương mại điện tử chính phủ cho việc thu mua hàng hóa, hệ thống hồ sơ điện tử, hệ thống email và ứng dụng bảo mật, ...
- **Đài Loan:** đến tháng 9/2004, có khoảng 1,2 triệu chứng nhận được phát hành. Lĩnh vực áp dụng là chính phủ, tài chính, doanh nghiệp như trao đổi công văn điện tử, mua bán hàng hóa điện tử, bảo mật web, email, thẻ tín dụng, ...
- **Thái Lan:** Lĩnh vực ứng dụng chính phủ và tài chính như ThaiDigital ID trong chính phủ và chi trả điện tử trong ngân hàng trong lĩnh vực tài chính.
- **Ấn Độ:** Hiện có 4 CA được cho phép và hơn 18.000 chứng nhận được phát hành. Lĩnh vực ứng dụng là chính phủ, ngân hàng, chăm sóc sức khỏe như thẻ chứng minh, ngân hàng điện tử, mua bán trực tuyến, hệ thống quản lý sức khỏe, đơn thuốc điện tử, thông tin y khoa điện tử.

### **1.2.2 Việt Nam**

Ở Việt Nam, các bộ luật cũng như nghị định được ban hành khá trễ so với các quốc gia trong khu vực và trên thế giới:

- Luật Giao dịch điện tử ban hành ngày 29/11/2005 (số 51/2005/QH11), có hiệu lực từ ngày 1/3/2006.
- Luật Công nghệ thông tin ban hành ngày 26/6/2006 (số 67/2006/QH11), có hiệu lực từ ngày 1/1/2007.
- Nghị định số 26/2007/NĐ-CP quy định chi tiết thi hành luật Giao dịch điện tử về Chữ ký số và Dịch vụ chứng thực chữ ký số.
- Nghị định số 27/2007/NĐ-CP về Giao dịch điện tử trong hoạt động tài chính.
- Nghị định số 35/2007/NĐ-CP về Giao dịch điện tử trong hoạt động ngân hàng.
- Nghị định số 63/2007/NĐ-CP quy định xử phạt vi phạm hành chính trong lĩnh vực công nghệ thông tin.
- Nghị định số 64/2007/NĐ-CP về ứng dụng công nghệ thông tin trong hoạt động của cơ quan nhà nước.

|   |           |          |           |          |             |              |             |          |
|---|-----------|----------|-----------|----------|-------------|--------------|-------------|----------|
|   |           |          |           |          |             |              |             |          |
| Luật Giao dịch điện tử,<br>Thương mại điện tử |           | Wales    |           |          |             |              |             |          |
|   | USA       | Mexico   | Scotland  |          | New Zealand | South Africa |             | Viet Nam |
|   | Canada    | India    | Ireland   |          |             |              |             |          |
|   | Australia | England  | Hong Kong |          |             |              |             |          |
| Luật Chữ ký điện tử,<br>Chữ ký số             |           |          |           | Thailand |             |              |             |          |
|   | USA       | Sweden   | Norway    | Romania  | Taiwan      | Wales        |             |          |
|   | Austria   | CH Czech | Germany   | Japan    | Russia      | Scotland     | Switzerland |          |
|   | Singapore | Korea    |           |          | England     | Latvia       | Spain       | China    |
|   | 1998      | 1999     | 2000      | 2001     | 2002        | 2003         | 2004        | 2006     |

**Hình 1.1. Thời điểm ban hành các luật liên quan PKI của các quốc gia trên thế giới**

Tuy đã có hành lang pháp lý về giao dịch điện tử nhưng đến nay chỉ có một số tổ chức, doanh nghiệp tự triển khai hệ thống CA nội bộ dùng riêng như Ngân hàng Nhà nước, Vietcombank, ACB, công ty VDC, VASC, ... Sự chậm trễ này một phần là do trình độ khoa học kỹ thuật trong lĩnh vực công nghệ thông tin nước ta còn non kém, phần khác là do sự thiếu quyết tâm và trì trệ trong công tác nghiên cứu và triển khai.

Tại hội thảo “Triển khai ứng dụng chữ ký số và thúc đẩy ứng dụng CNTT-TT trong doanh nghiệp” tại Quảng Ninh diễn ra vào ngày 8/8/2007, ông Đào Đình Khả (Phó Trưởng phòng Phát triển Nguồn lực Thông tin, Cục Ứng dụng CNTT) cho biết trung tâm chứng thực số quốc gia (Root CA) dự kiến sẽ đi vào hoạt động trong tháng 9 hoặc đầu tháng 10/2007, có nhiệm vụ cấp phép cung cấp dịch vụ chứng thực chữ ký số công cộng và cấp phát chứng nhận số cho các tổ chức đăng ký cung cấp dịch vụ chứng thực chữ ký số công cộng nhưng mãi đến 16/5/2008 (tức hơn nửa năm sau), Bộ Thông tin và Truyền thông (TT&TT) mới tổ chức lễ tạo bộ khóa bí mật và khoá công khai của Root CA dùng để cấp phép cho các tổ chức và doanh nghiệp cung cấp dịch vụ chữ ký số và cho biết dự kiến khoảng 2 tuần nữa sẽ chính thức ra mắt Root CA. Tuy nhiên, đến thời điểm này hệ thống vẫn chưa được đi vào hoạt động.

### 1.3 Nhu cầu thực tế

Sự chậm trễ trong việc triển khai hạ tầng PKI, đẩy mạnh ứng dụng CNTT, đặc biệt là chữ ký số trong các giao dịch điện tử không chỉ ảnh hưởng đến các cá nhân, tổ chức có liên quan mà còn làm cho Việt Nam ngày càng tụt hậu về mặt công nghệ và thiệt hại về kinh tế. Tại hội thảo “Triển khai ứng dụng chữ ký số và thúc đẩy ứng dụng CNTT-TT trong doanh nghiệp” tại Quảng Ninh diễn ra vào ngày 8/8/2007, ông Vũ Đức Nam (Thứ trưởng Bộ TT&TT), ông Hoàng Văn Dũng (Phó Chủ tịch Phòng Thương mại và Công nghiệp Việt Nam – VCCI) và ông Hoàng Quốc Lập (Cục trưởng Cục ứng dụng CNTT, Bộ TT&TT) đã đưa ra một số ví dụ điển hình như sau:

- Một công ty do người Việt lập ra tại Singapore chuyên để mua hàng thông qua các giao dịch điện tử rồi bán ngược lại ở Việt Nam, để ăn chênh lệch khoảng 10 – 20% tổng giá trị hàng hoá. Mức chênh lệch lợi nhuận khá cao này được thực hiện đơn giản bởi giao dịch điện tử và đặc biệt là chữ ký số, một điều quá bình thường ở một đất nước như Singapore nhưng xa lạ ở Việt Nam.
- Nokia định thực hiện một hợp đồng trị giá gần 1 triệu đô-la Mỹ, để gia công phần mềm cho điện thoại di động ở Việt Nam. Nhưng sau đó công ty này bỏ cuộc bởi họ cho rằng không thể chỉ vì Việt Nam mà phải lập một nhóm chuyên gia chuyên xử lý các văn bản, fax, ... trong khi giao dịch điện tử thông qua hình thức chữ ký số là điều bình thường ở rất nhiều nước trên thế giới.
- Việc giải phóng 1 container ở Việt Nam hiện mất khoảng 7 ngày, trong khi đó tốc độ trung bình của thế giới hiện nay là một container được giải phóng ngay trong ngày. Điều này được thực hiện nhờ vào việc ứng dụng CNTT và chữ ký số nhằm giải phóng hàng nhanh tại các cảng biển cũng như các cửa khẩu.

Sau khi ban hành các bộ luật và nghị định làm hành lang pháp lý trong lĩnh vực thương mại điện tử, tuy chưa xây dựng hạ tầng quốc gia nhưng Việt Nam đã thu hút khá nhiều đầu tư quan trọng:

- Ngày 24/4/2008, công ty Điện toán và Truyền số liệu VDC, NCS Solutions và Global Sign đã phối hợp cùng tổ chức “Hội thảo về chứng nhận số và hệ

thống chứng thực điện tử” nhằm hợp tác triển khai đề án chứng thực điện tử VDC để

xây dựng một tổ chức chứng thực gốc tại Việt Nam, cung cấp chứng nhận số cá nhân cho người dùng, cho máy chủ, mã và phần mềm. Thời điểm cung cấp dịch vụ chứng thực điện tử của tổ chức này sẽ bắt đầu ngay sau khi Trung tâm Chứng thực số gốc Quốc gia (Root CA) chính thức đi vào hoạt động.

- Ngày 3/6/2008, Bộ TT&TT và Cisco đã ký Biên bản ghi nhớ tăng cường hợp tác trong lĩnh vực công nghệ mạng và truyền thông, góp phần thúc đẩy quá trình xây dựng Chính phủ điện tử tại Việt Nam.
- Ngày 17/06/2008, eBay, công ty nổi tiếng trong lĩnh vực mua bán qua mạng của Mỹ đã hợp tác với [www.chodientu.vn](http://www.chodientu.vn)<sup>1</sup> (thuộc công ty PeaceSoft) nhằm đẩy mạnh giao dịch tại thị trường nội địa Việt Nam.

Một khó khăn chủ yếu trong việc triển khai hạ tầng khóa công khai ở Việt Nam là lựa chọn mô hình PKI nào phù hợp với nước ta để triển khai. Hiện có nhiều mô hình khác nhau được áp dụng trên thế giới, nhưng chỉ có hai mô hình điển hình được hầu hết các quốc gia áp dụng là mô hình tập trung PKI phân cấp (Root CA) và mô hình CA cầu nối (Bridge CA). Mỗi mô hình đều có những điểm mạnh, điểm yếu riêng và theo phân tích thì mô hình PKI phân cấp phù hợp với Việt Nam nhất và hiện đang được tập trung nghiên cứu và triển khai (chi tiết được trình bày ở Chương 4).

## 1.4 Mục tiêu của đề tài

Chứng nhận khóa công khai và hạ tầng khóa công khai có phạm vi nghiên cứu rất rộng. Do đó, mục tiêu của đề tài nhằm tập trung nghiên cứu một số vấn đề sau:

- Phân tích và thử nghiệm các mô hình chữ ký số và các thuật toán liên quan nhằm chọn mô hình phù hợp để tập trung nghiên cứu.
- Tìm hiểu chức năng và các vấn đề liên quan đến chứng nhận khóa công khai.
- Nghiên cứu và phân tích các kiến trúc hạ tầng khóa công khai, từ đó đánh giá và chọn lựa kiến trúc phù hợp có thể triển khai trong thực tế.

---

<sup>1</sup> Trang web [www.chodientu.vn](http://www.chodientu.vn) hiện có 120.000 thành viên với khoảng 600.000 lượt người truy cập mỗi ngày.

- Nghiên cứu và phân tích các nguy cơ tổn thương trên hệ mã khóa công khai RSA, hạt nhân của PKI, đồng thời đưa ra giải pháp để chống lại các nguy cơ tổn thương này.
- Nghiên cứu và giải quyết một số bài toán quan trọng nhằm cài đặt hệ mã RSA an toàn và hiệu quả.
- Trên cơ sở phân tích đó, xây dựng bộ thư viện mã hóa nhằm cài đặt hiệu quả hệ mã RSA, đồng thời tiến hành thử nghiệm các thuật toán được cài đặt để đánh giá tính hiệu quả của nó.
- Tìm hiểu và chọn một gói phần mềm mã nguồn mở phổ biến hiện nay có thể triển khai một hệ thống PKI/CA hoàn chỉnh để phân tích và cải tiến nhằm triển khai hiệu quả trong thực tế đồng thời có thể kiểm soát được quá trình phát triển cũng như độ an toàn của hệ thống này.

## 1.5 Nội dung của luận văn

Chương 2, Chương 3 và Chương 4 trình bày các nghiên cứu, phân tích và đánh giá về kiến trúc hạ tầng khóa công khai và các thành phần quan trọng liên quan. Chương 5 và Chương 6 sẽ tập trung nghiên cứu và phân tích các nguy cơ tổn thương trên hệ mã RSA và một số bài toán quan trọng khi cài đặt hệ mã này. Trên cơ sở phân tích đó, Chương 7 sẽ giới thiệu bộ thư viện “SmartRSA” được xây dựng cùng các thử nghiệm nhằm đánh giá tính hiệu quả khi cài đặt hệ mã RSA. Chương 8 sẽ giới thiệu gói phần mềm mã nguồn mở “EJBCA” có khả năng triển khai một hệ thống PKI/CA hoàn chỉnh được chọn để tìm hiểu, phân tích, cải tiến và triển khai thử nghiệm trong thực tế. Một số kết luận và hướng phát triển của đề tài được trình bày ở Chương 9.

Nội dung các chương cụ thể như sau:

- **Chương 1** trình bày tổng quan về hạ tầng khóa công khai, đồng thời giới thiệu mục tiêu và nội dung của luận văn.
- **Chương 2** trình bày khái niệm, nhu cầu và nguyên lý hoạt động của chữ ký số đồng thời khảo sát hai thuật toán quan trọng trong chữ ký số là thuật toán hàm băm mật mã (MD5, SHA) và thuật toán chữ ký số (RSA, Elgamal, DSA,

ECDSA). Các thử nghiệm cũng được lần lượt tiến hành nhằm so sánh tính hiệu quả của các thuật toán này, từ đó chọn mô hình chữ ký số phù hợp để tập trung nghiên cứu.

- **Chương 3** trình bày tổng quan về tổ chức chứng nhận khóa công khai (CA) và các chứng nhận khóa công khai, đồng thời giới thiệu các chức năng quan trọng của tổ chức này.
- **Chương 4** trình bày khái niệm, vai trò và chức năng của hạ tầng khóa công khai, đồng thời tập trung nghiên cứu và phân tích các kiến trúc hạ tầng khóa công khai hiện có, từ đó đánh giá và chọn lựa kiến trúc phù hợp có thể triển khai trong thực tế.
- **Chương 5** trình bày và phân tích các nguy cơ tấn công gây tổn thương trên hệ mã RSA, từ đó đưa ra các giải pháp nhằm cài đặt hệ mã một cách an toàn.
- **Chương 6** trình bày các nghiên cứu về một số bài toán quan trọng cần giải quyết kết hợp với những cơ sở phân tích ở chương trên nhằm xây dựng hệ mã RSA an toàn, hiệu quả.
- **Chương 7** giới thiệu bộ thư viện mã hóa “SmartRSA” được xây dựng nhằm cài đặt hiệu quả hệ mã RSA trên cơ sở nghiên cứu và phân tích về các nguy cơ tổn thương hệ mã ở Chương 5 và các bài toán quan trọng trong việc cài đặt hệ mã hiệu quả ở Chương 6. Các thử nghiệm nhằm kiểm tra tính hiệu quả cũng được trình bày ở chương này.
- **Chương 8** giới thiệu gói phần mềm mã nguồn mở EJBCA, gói phần mềm cho phép triển khai một hệ thống PKI/CA hoàn chỉnh, đầy đủ chức năng. Nhằm tận dụng các tính chất ưu việt của gói phần mềm này cũng như có thể kiểm soát được quá trình phát triển và độ an toàn của hệ thống, đề tài đã tiến hành phân tích, cải tiến và triển khai thử nghiệm một hệ thống PKI/CA phân cấp đơn giản có thể sử dụng ngay trong thực tế.
- **Chương 9** trình bày một số kết luận và hướng phát triển của đề tài.

## Chương 2

# Chữ ký số

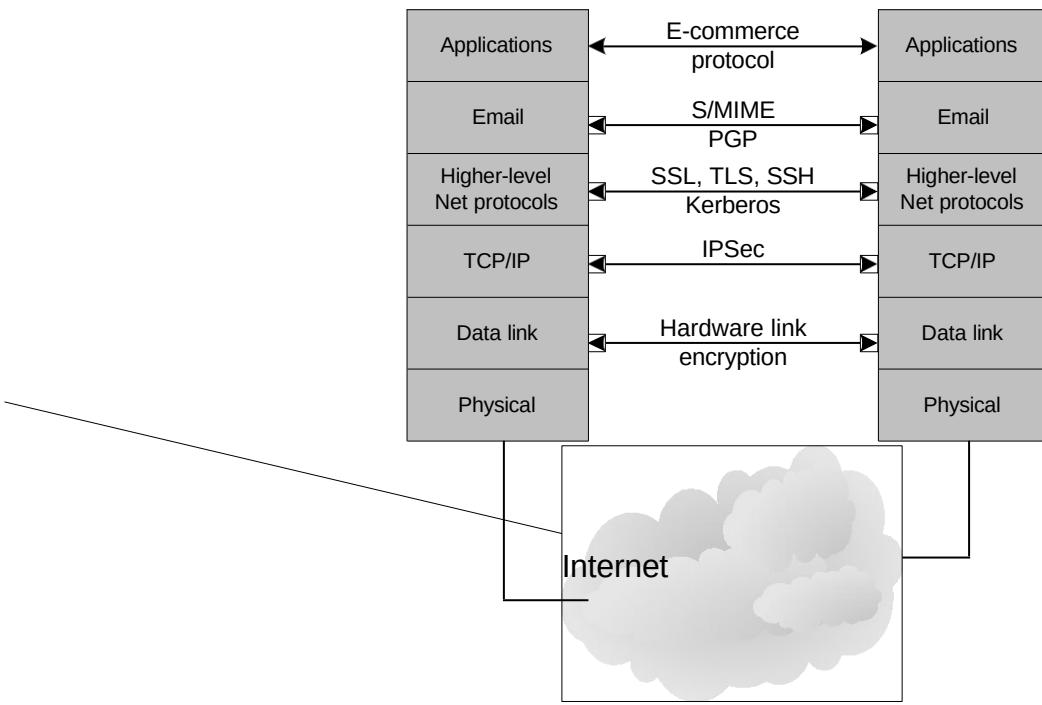
☐ *Nội dung của chương này trình bày khái niệm, nhu cầu và nguyên lý hoạt động của chữ ký số đồng thời khảo sát hai thuật toán quan trọng trong chữ ký số là thuật toán hàm băm mật mã (MD5, SHA) và thuật toán chữ ký số (RSA, Elgamal, DSA, ECDSA). Các thử nghiệm cũng lần lượt được tiến hành ở mục 2.2.3 và mục 2.3.3 nhằm so sánh tính hiệu quả của các thuật toán này, từ đó chọn mô hình chữ ký số phù hợp để tập trung nghiên cứu.*

### 2.1 Giới thiệu

#### 2.1.1 Nhu cầu thực tế

Ngày nay, với sự phát triển các phương tiện thông tin liên lạc hiện đại đặc biệt là mạng Internet đã giúp việc trao đổi thông tin giữa người và người hầu như không còn bị giới hạn bởi không gian và thời gian nữa. Tuy nhiên, do hạ tầng truyền thông hiện nay vẫn chưa thật sự hoàn thiện nên đã tạo cơ hội cho một số kẻ xấu lợi dụng những lỗ hổng trên mạng để tìm cách đánh cắp, phá hủy hoặc thay đổi thông tin trên mạng vì mục đích nào đó. Chính vì lý do đó, ngành an ninh trên mạng ra đời và đã đưa ra nhiều giải pháp với nhiều mức độ và phạm vi áp dụng khác nhau để bảo vệ người dùng trên mạng.

Hình 2.1 dưới đây cho thấy mức độ bảo mật cao nhất được thực hiện ở tầng ứng dụng sử dụng trong giao thức thương mại điện tử. Thương mại điện tử đang là xu hướng phát triển tất yếu của một xã hội thông tin hiện đại và cao hơn nữa là mô hình “Chính phủ điện tử” mà nhiều nước đang dần tiến đến. Chính vì vậy, việc tìm ra các giải pháp an toàn cho bảo mật thông tin trong việc trao đổi thông tin càng trở nên cần thiết. Thương mại điện tử hay Chính phủ điện tử thực ra là việc thay đổi toàn bộ các công việc trên giấy tờ, hợp đồng, công văn, ... trên giấy trước đây bằng các công việc thực hiện trên văn bản điện tử và trao đổi qua mạng máy tính.



**Hình 2.1. Kiến trúc bảo mật trong TCP/IP**

Tuy nhiên, các tài liệu điện tử này không được xác thực trong quá trình trao đổi thông tin nên khi bị sao chép hay sửa đổi ta cũng không thể phát hiện được dẫn đến những hậu quả nghiêm trọng (nhất là trong các lĩnh vực kinh doanh, thương mại, pháp luật, chính phủ, ...). Nguyên nhân chủ yếu là do các ký tự số hóa trong tài liệu điện tử không phải là duy nhất, vì vậy nó dễ dàng bị sao chép và sửa đổi mà không thể phát hiện được. Mặt khác, các biện pháp bảo mật hiện nay như dùng mật khẩu đều không đảm bảo vì có thể bị nghe trộm hoặc bị dò ra nhanh chóng do người sử dụng thường chọn mật khẩu ngắn, dễ nhớ, dùng chung và ít khi thay đổi mật khẩu.

Trong các công việc truyền thống sử dụng văn bản giấy như lĩnh vực kinh doanh, để đảm bảo pháp nhân cho một quyết định hay công văn, người ta thường sử dụng các chữ ký tay hay con dấu vào cuối văn bản đó. Hành động này có một số mục đích như *tạo dấu hiệu* (xác thực người ký), *sự chứng nhận và thừa nhận* (bằng chứng về những gì người ký đã làm, đã xem, đã thừa nhận đối với một tài liệu vào thời điểm ký), *tính an toàn* (rõ ràng, sự quyết định dứt khoát trong giao dịch) và là một *nghị thức* (cho biết người ký đã tạo trách nhiệm ràng buộc về mặt pháp lý, bắt buộc người ký phải cân nhắc trước tài liệu được ký).

Tương tự, để đáp ứng được các cuộc kiểm tra về mặt pháp lý và công nghệ, chữ ký trên các tài liệu điện tử được tạo ra để người nhận có thể xác thực đến một đối tác đáng tin cậy thứ ba (như tòa án, quan tòa, trọng tài, ...) để đảm bảo nội dung của tài liệu là nguyên gốc của người gửi đồng thời bảo đảm rằng người gửi không thể phủ nhận đã ký trên tài liệu này. Chữ ký điện tử nói chung và chữ ký số nói riêng đã ra đời để giải quyết vấn đề đó.

### **2.1.2 Khái niệm**

Vào năm 1889, tòa án tối cao bang New Hampshire (Hoa Kỳ) đã phê chuẩn tính hiệu lực của chữ ký điện tử. Tuy nhiên, chỉ với những phát triển của khoa học kỹ thuật gần đây thì chữ ký điện tử mới đi vào cuộc sống một cách rộng rãi.

Theo luật Giao dịch Điện tử Thông nhất UETA của NCCUSL năm 1999 [68], chữ ký điện tử được định nghĩa như sau:

*“Chữ ký điện tử là thông tin (âm thanh, ký hiệu, tiến trình điện tử, ...) đi kèm theo dữ liệu (văn bản, hình ảnh, video, ...) nhằm mục đích xác định người chủ của dữ liệu đó”.*

hoặc theo điều 21 trong luật Giao dịch điện tử của Quốc hội nước Cộng hòa Xã hội Chủ nghĩa Việt Nam [3] định nghĩa:

*“Chữ ký điện tử được tạo lập dưới dạng từ, chữ, số, ký hiệu, âm thanh hoặc các hình thức khác bằng phương tiện điện tử, gắn liền hoặc kết hợp một cách lôgic với thông điệp dữ liệu, có khả năng xác nhận người ký thông điệp dữ liệu và xác nhận sự chấp thuận của người đó đối với nội dung thông điệp dữ liệu được ký”.*

Như vậy, theo các định nghĩa này, chữ ký điện tử chỉ đến bất kỳ phương pháp nào (không nhất thiết là mật mã) để xác định người chủ của văn bản điện tử. Hiện nay, chữ ký điện tử có thể bao hàm các cam kết gửi bằng email, nhập các số định dạng cá nhân (Personal Identification Number – PIN) vào các máy ATM, ký bằng bút điện tử với thiết bị màn hình cảm ứng tại các quầy tính tiền, chấp nhận các điều khoản người dùng (End User Licence Agreement – EULA) khi cài đặt phần mềm máy tính, ký các hợp đồng điện tử trực tuyến, ...

**Chữ ký số (Digital Signature)** chỉ là tập con của chữ ký điện tử. Chữ ký số là chữ ký điện tử dựa trên kỹ thuật mã hóa với khóa công khai, trong đó, mỗi người có một cặp khóa (một khóa bí mật và một khóa công khai). Khóa bí mật không bao giờ được công bố, trong khi đó, khóa công khai được tự do sử dụng. Để trao đổi thông điệp bí mật, người gửi sử dụng khóa công khai của người nhận để mã hóa thông điệp gửi, sau đó, người nhận sẽ sử dụng khóa bí mật tương ứng của mình để giải mã thông điệp. Chữ ký số lại hoạt động theo hướng ngược lại. Người gửi sẽ sử dụng khóa bí mật của mình để mã hóa thông điệp gửi, sau đó người nhận sẽ sử dụng khóa công khai của người gửi để giải mã thông điệp. Như vậy, chỉ có khóa công khai tương ứng với khóa bí mật đã dùng mới giải mã được thông điệp, điều đó đã đảm bảo cho thông điệp gửi không bị giả mạo, không bị thay đổi trong quá trình gửi.

### 2.1.3 Các dịch vụ bảo mật

Khi nghiên cứu về chữ ký số, ta cần quan tâm đến bốn dịch vụ bảo mật cơ bản sau:

- **Dịch vụ cẩn mật** hạn chế việc truy cập đến các dữ liệu có tính nhạy cảm, ngăn ngừa việc để lộ thông tin đến các cá nhân bất hợp pháp và chỉ một số cá nhân có quyền mới truy xuất được dữ liệu này.
- **Dịch vụ về tính toàn vẹn** chỉ ra các thay đổi không được xác thực hay các thay đổi ngẫu nhiên trên dữ liệu (bao gồm việc thêm, xóa, sửa đổi trên dữ liệu) giúp người nhận dữ liệu xác minh rằng dữ liệu đã không được thay đổi. Để đảm bảo tính toàn vẹn của dữ liệu, một hệ thống phải có khả năng phát hiện ra những sự thay đổi không được xác thực trên dữ liệu.
- **Dịch vụ nhận diện và xác thực** thiết lập tính hợp lệ của một cuộc truyền nhận dữ liệu hay một thông điệp với người tạo ra nó và giúp người nhận có thể xác định được nguồn gốc của dữ liệu.
- **Dịch vụ không thể chối từ** ngăn chặn một cá nhân từ chối các hành vi đã thực hiện trước đây và bảo đảm rằng người nhận dữ liệu được bảo đảm sự nhận biết của người gửi.

### 2.1.4 Nguyên lý hoạt động của chữ ký số

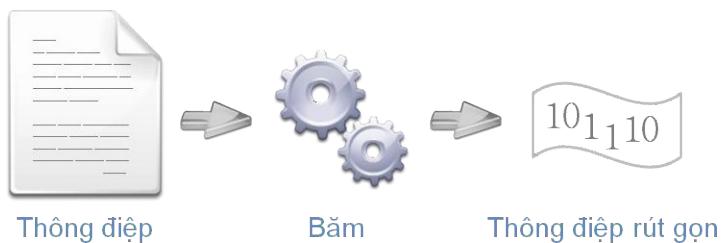
Chữ ký số là cách dùng mật mã để tạo dữ liệu nhằm xác thực được người gửi và bảo đảm dữ liệu đó không bị thay đổi. Chữ ký số được thực hiện nhờ vào kỹ thuật mã hóa với sự phối hợp với một cặp khóa (một khóa bí mật – private key và một khóa công khai – public key). Người gửi sẽ sử dụng khóa bí mật để mã hóa dữ liệu gửi, người nhận sẽ sử dụng khóa công khai của người gửi để giải mã dữ liệu nhận. Cơ sở cho kỹ thuật này chính là kỹ thuật mã hóa khóa công khai (Public Key Cryptography – PKC).

Chữ ký số tạo một ràng buộc giữa người dùng khóa bí mật với tài liệu được ký bởi anh ta. Đồng thời, chữ ký số cũng bảo đảm tính toàn vẹn thông tin và tính không thể chối từ của người gửi từ khi người gửi ký nhận trên thông tin đến khi thông tin đó được gửi đến nơi người nhận.

Hai thuật toán chính được sử dụng trong chữ ký số là thuật toán hàm băm mật mã và thuật toán chữ ký số (tạo khóa, ký và xác nhận chữ ký).

#### 2.1.4.1 Chức năng của hàm băm trong chữ ký số

Hàm băm được sử dụng ở đây là hàm băm một chiều. Chức năng chính của hàm băm trong kỹ thuật chữ ký điện tử là xử lý một thông điệp có chiều dài bất kỳ để cho ra kết quả là một thông điệp có chiều dài cố định được gọi thông điệp rút gọn (message digest) hay giá trị băm (hash value). Chiều dài của thông điệp rút gọn tùy thuộc vào hàm băm nào đã được sử dụng. Điều này giúp tăng tốc cho thuật toán chữ ký số (sẽ trình bày ở phần ...) và khả năng an toàn cao (do khó có thể tìm ra 2 thông điệp có thông điệp rút gọn giống nhau).

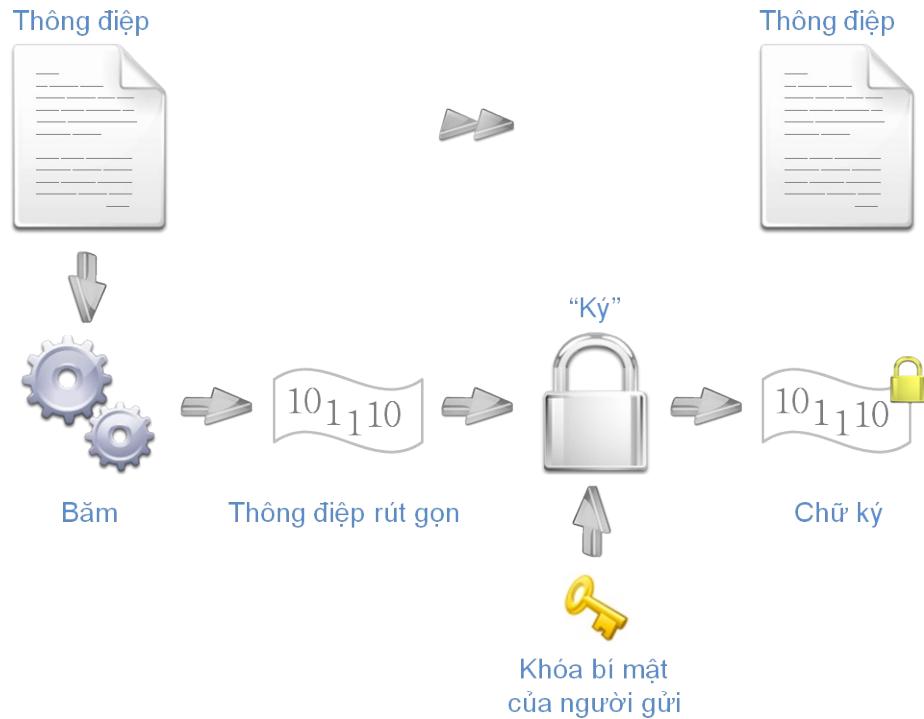


**Hình 2.2. Băm dữ liệu**

Chi tiết về hàm băm sẽ được trình bày ở mục 2.2.

#### 2.1.4.2 Chức năng của thuật toán “ký” trong chữ ký số

Thuật toán ký có chức năng “ký” vào thông điệp rút gọn được tạo ra từ hàm băm như trên.

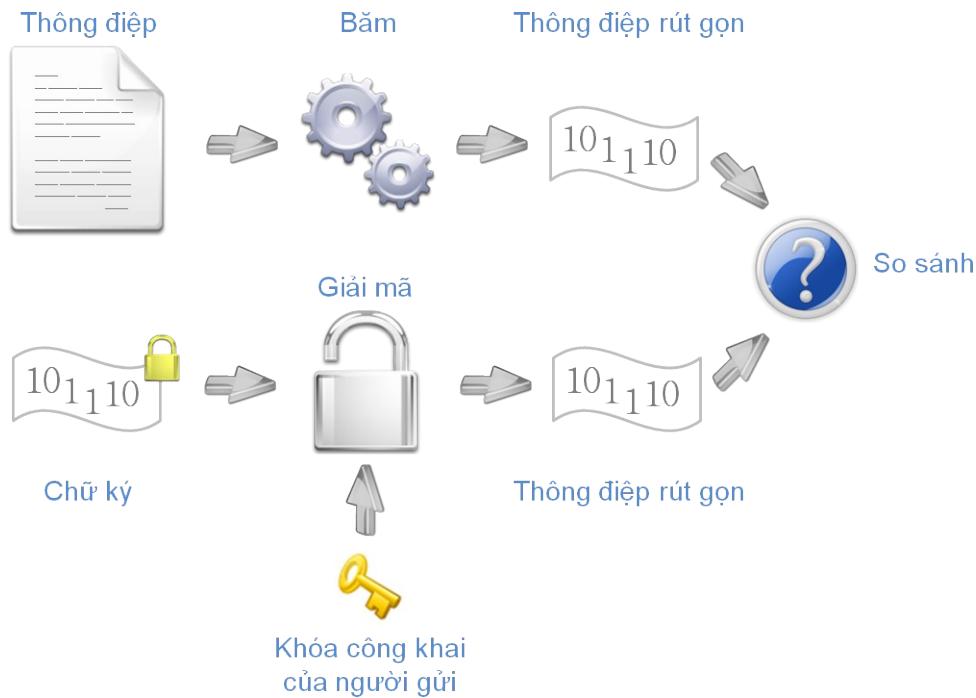


**Hình 2.3. Ký nhận một thông điệp rút gọn**

Việc ký này thực ra là việc mã hóa thông điệp rút gọn bằng mật mã (hay khóa). Tùy thuộc vào mục đích của ứng dụng mà ta sử dụng khóa công khai hay khóa bí mật để ký. Trong trường hợp chữ ký số thì người ta sử dụng khóa bí mật để ký. Người ký sẽ gửi thông điệp và chữ ký số vừa tạo ra đến người nhận.

#### 2.1.4.3 Chức năng của thuật toán “xác nhận chữ ký” trong chữ ký số

Khi nhận được thông điệp và chữ ký số từ người gửi, người nhận sử dụng cùng một thuật toán băm với người gửi trên thông điệp nhận được sẽ thu được một thông điệp rút gọn tương ứng. Sau đó người nhận dùng khóa công khai (public key) của người gửi để giải mã chữ ký số nhận được và so sánh kết quả với thông điệp tóm tắt nhận được ở trên. Nếu giống nhau thì chấp nhận. Nếu khác nhau thì chứng tỏ thông điệp nhận được không phải do người gửi ký nhận hoặc thông điệp đã bị thay đổi.



**Hình 2.4. Kiểm định chữ ký điện tử**

Chi tiết của thuật toán ký và xác nhận chữ ký sẽ được trình bày ở mục 2.3.

## 2.2 Thuật toán hàm băm mật mã

### 2.2.1 Giới thiệu

Trên thực tế, các thông điệp sử dụng chữ ký số có độ dài bất kỳ, thậm chí lên đến vài MegaByte. Trong khi đó, các thuật toán chữ ký số (sẽ được trình bày ở mục 2.3) lại được áp dụng trên các thông điệp có độ dài cố định và tương đối ngắn. Để giải quyết vấn đề này, ta có thể chia nhỏ thông điệp thành từng phần nhỏ có độ dài thích hợp rồi ký trên từng phần đó. Tuy nhiên, giải pháp này không khả thi vì một số lý do sau:

- Nếu thông điệp quá dài thì số lượng chữ ký sẽ rất nhiều dẫn đến lượng thông tin gửi đi tăng lên đáng kể.
- Hầu hết các phương pháp chữ ký số có độ an toàn cao đều đòi hỏi chi phí tính toán cao, tốc độ xử lý chậm. Nếu bắt buộc phải áp dụng thuật toán tạo chữ ký số nhiều lần thì sẽ tốn nhiều thời gian để ký thông điệp gửi.
- Không đảm bảo được tính toàn vẹn của thông tin ban đầu khi phải chia nhỏ thông tin gửi. Vì từng phần thông điệp được ký có thể dễ dàng bị thay đổi

thứ tự hoặc bỏ bớt, nhưng vẫn không làm mất đi tính hợp lệ của chữ ký trên các phần văn bản được ký.

Trên thực tế, các vấn đề trên được giải quyết bằng cách sử dụng hàm băm mật mã (Cryptographic Hash Function). Một thông điệp có độ dài bất kỳ, sau khi qua hàm băm  $\mathcal{H}$  sẽ nhận được một thông điệp rút gọn (message digest) có độ dài xác định. Độ an toàn của hàm băm  $\mathcal{H}$  được đo bằng khả năng xảy ra “đụng độ” (collision) khi tính toán cặp thông điệp  $x, x'$  sao cho  $x \neq x'$  và  $\mathcal{H}(x) = \mathcal{H}(x')$ . Khả năng đụng độ càng ít thì độ an toàn của hàm băm càng cao. Ngoài ra, hàm băm  $\mathcal{H}$  còn phải là hàm một chiều, nghĩa là nếu biết được một thông điệp rút gọn  $z$  bất kì thì không thể xác định ngược lại một thông điệp  $x$  sao cho  $\mathcal{H}(x) = z$ .

Có rất nhiều thuật toán băm đã được công bố nhưng hai thuật toán băm được sử dụng phổ biến trong chữ ký số từ thập niên 1990 đến nay là MD5 và SHA-1 (chuẩn SHS).

## 2.2.2 Một số hàm băm mật mã thông dụng

### 2.2.2.1 Thuật toán hàm băm MD5

MD5 (Message-Digest algorithm 5) là một hàm băm mật mã được sử dụng phổ biến, được thiết kế bởi Giáo sư Ronald L. Rivest tại trường MIT vào năm 1991 để thay thế cho hàm băm trước đó là MD4 (1990). Là một chuẩn Internet (RFC 1321 [69]), MD5 đã được dùng trong nhiều ứng dụng bảo mật và cũng được dùng phổ biến để kiểm tra tính toàn vẹn của tập tin. Cũng như các hàm băm khác như MD4 và SHS (Secure Hash Standard), MD5 là phương pháp có ưu điểm tốc độ xử lý rất nhanh, thích hợp với các thông điệp dài cho và cho ra giá trị băm dài 128 bit.

Trong MD5, thông điệp ban đầu  $x$  sẽ được mở rộng thành dãy bit  $X$  có độ dài là bội của 512. Dãy bit  $X$  gồm các thành phần được sắp thứ tự như sau: Dãy bit  $x$  ban đầu, một bit 1, dãy  $d$  bit 0 ( $d$  được tính sao cho dãy  $X$  cuối cùng là bội của 512), dãy 64 bit l biểu diễn chiều dài của thông điệp. Đơn vị xử lý trong MD5 là các từ 32-bit, nên dãy bit  $X$  ở trên sẽ được biểu diễn thành dãy các từ  $X_i$  32-bit sau:

$X = X_0 X_1 X_2 \dots X_{N-1}$ , với  $N$  là bội của 16. Nội dung chi tiết thuật toán hàm băm MD5 xin tham khảo tại [1, tr.116-117].

Phương pháp MD5 có những ưu điểm sau so với phương pháp MD4:

- Thay vì có 3 chu kỳ biến đổi như trong MD4, MD5 bổ sung thêm chu kỳ thứ 4 để tăng mức độ an toàn.
- Trong mỗi thao tác của từng chu kỳ, MD5 sử dụng hằng số *ti* phân biệt, trong khi MD4 sử dụng hằng số chung cho mọi thao tác trong cùng chu kỳ biến đổi.
- Hàm  $G$  ở chu kỳ 2 của MD4:  $G(X, Y, Z) = X \wedge Z \vee X \wedge Y \vee Y \wedge Z$  được thay thế bằng  $G(X, Y, Z) = X \wedge Z \vee Y \wedge \neg Z$  để giảm tính đối xứng.
- Mỗi bước biến đổi trong từng chu kỳ chịu ảnh hưởng kết quả của bước biến đổi trước, vì vậy làm tăng nhanh tốc độ của hiệu ứng lan truyền (avalanche).
- Các hệ số dịch chuyển xoay vòng trong mỗi chu kỳ được tối ưu hóa nhằm tăng tốc độ hiệu ứng lan truyền. Ngoài ra, mỗi chu kỳ sử dụng 4 hệ số dịch chuyển khác nhau.

Lý do MD5 được thiết kế thay thế cho MD4 là vì các phân tích chỉ ra rằng phương pháp MD4 có vẻ không an toàn. Den Boer và Bosselaers đã chỉ ra các điểm yếu trong MD4 trong một bài báo được đăng vào năm 1991 [9] và một tấn công xung đột đầu tiên được tìm thấy bởi Han Dobbertin vào năm 1996 [20].

Tuy nhiên, các nỗ lực tấn công, phân tích của các nhà nghiên cứu cho thấy MD5 cũng không còn an toàn và cần được thay thế bằng một thuật toán băm khác như các công bố của Den Boer và Bosselaers năm 1993 [10]; của Hans Dobbertin năm 1996 [21]; của nhóm tác giả Xiaoyun Wang, Dengguo Feng, Xuejia Lai, và Hongbo ngày 19/8/2004 [56],[57]; của Arjen Lenstra, Xiaoyun Wang, và Benne de Weger ngày 1/3/2005 [36]; và của Vlastimil Klima [32], [33], ...

### **2.2.2.2 Chuẩn băm an toàn SHS**

SHS (Secure Hash Standard) là chuẩn gồm tập hợp các thuật toán băm mật mã an toàn (Secure Hash Algorithm – SHA) như SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 do NIST<sup>2</sup> và NSA<sup>3</sup> xây dựng.

<sup>2</sup> Viện Tiêu chuẩn và Công nghệ quốc gia Hoa Kỳ (National Institute of Standards and Technology – NIST).

<sup>3</sup> Cục an ninh quốc gia Mỹ (National Security Agency – NSA ).

Phiên bản đầu tiên của thuật toán này được công bố trên Federal Register (tập san chính thức của chính phủ Hoa Kỳ) vào ngày 31/1/1992 và sau đó chính thức trở thành phương pháp chuẩn từ ngày 13/5/1993 trong FIPS<sup>4</sup> 180 [66]. Phiên bản này hiện được xem là SHA-0. Nó đã bị NSA hủy bỏ sau khi công bố một thời gian ngắn và được thay thế bởi phiên bản sửa lại được công bố vào năm 1995 trong FIPS 180-1

[66] và thường được xem là SHA-1.

SHA-1 khác với SHA-0 duy nhất một phép xoay bit đơn trong thông điệp của hàm nén của nó. NSA thực hiện việc này để sửa một sai lầm trong thuật toán ban đầu đã làm giảm sự an toàn mật mã của nó. Tuy nhiên, NSA đã không đưa ra bất kỳ lời giải thích hay nhận biết sai lầm này. Các điểm yếu trong cả SHA-0 và SHA-1 sau đó đã được phát hiện. Qua đó, SHA-1 dường như tốt hơn trước các tấn công và cho thấy sự khẳng định của NSA rằng sự thay đổi đã làm tăng sự an toàn. SHA-1 được coi là thuật giải thay thế MD5 và được sử dụng rộng rãi trong nhiều ứng dụng và giao thức an ninh khác nhau, bao gồm TLS và SSL, PGP, SSH, S/MIME và IPSec.

Bốn thuật toán SHA-224, SHA-256, SHA-384 và SHA-512 gọi chung là SHA-2. Ba thuật toán SHA-256, SHA-384 và SHA-512 được công bố lần đầu năm 2001 trong bản thảo FIPS 180-2. Năm 2002, FIPS 180-2 [66], bao gồm cả SHA-1 được chấp nhận thành chuẩn chính thức. Năm 2004, FIPS 180-2 được bổ sung thêm một biến thể SHA-224, với mục đích tạo ra một biến thể có độ dài khóa trùng với 2 lần khóa của TripleDES (112 bit). Năm thuật toán SHA này được đặc tả trong bản thảo FIPS 180-3 công bố vào 8/6/2007 [66].

Phương pháp SHA-1 (cũng như SHA-0) được xây dựng trên cùng cơ sở với phương pháp MD4 và MD5. Tuy nhiên, phương pháp SHA-1 sử dụng trên hệ thống Big-endian<sup>5</sup> thay vì Little-endian<sup>6</sup> như phương pháp MD4 và MD5. Ngoài ra, hàm băm SHA-1 tạo ra thông điệp rút gọn kết quả có độ dài 160 bit nên thường được sử dụng

<sup>4</sup> Chuẩn xử lý thông tin liên bang (Federal Information Processing Standard – FIPS).

<sup>5</sup> Trong hệ thống Big-endian, các byte có địa chỉ thấp là các byte nhiều ý nghĩa trong word.

<sup>6</sup> Trong hệ thống Little-endian, các byte có địa chỉ thấp là các byte ít ý nghĩa trong word.

kết hợp với thuật toán chữ ký số DSA (sẽ được trình bày ở mục 2.3.2.3). Nội dung chi tiết thuật toán hàm băm SHA-1 xin tham khảo tại [1, tr.118-119].

Phương pháp SHA-1 giống với MD5 (cải tiến từ MD4) nhưng thông điệp tóm tắt được tạo ra có độ dài 160 bit. Dưới đây là một số điểm so sánh giữa MD5 và SHA-1:

- Giống như MD5, SHA-1 cũng thêm chu kỳ thứ 4 để tăng mức độ an toàn cho thuật toán. Tuy nhiên, chu kỳ 4 của SHA-1 sử dụng lại hàm  $f$  của chu kỳ thứ 2.
- Trong SHA-1, 20 bước biến đổi trong cùng một chu kỳ sử dụng cùng một hàng số  $K_t$ . Trong khi đó, mỗi bước biến đổi trong cùng một chu kỳ của MD5 sử dụng các hằng số khác nhau.
- So với MD4, hàm  $G$  trong MD5 được thay thế thành hàm mới để làm giảm tính đối xứng. Trong khi SHA-1, hàm  $G$  trong SHA-1 vẫn giữ lại hàm  $G$  của MD4.
- Cả MD5 và SHA-1, mỗi bước biến đổi trong từng chu kỳ chịu ảnh hưởng kết quả của biến đổi trước, vì vậy làm tăng nhanh tốc độ của hiệu ứng lan truyền.

Về mặt giải thuật toán, các biến thể của SHA-2 không khác nhau mặc dù chúng sử dụng giá trị biến và hằng số cũng như độ dài từ, ... khác nhau. Dưới đây là bảng liệt kê đặc điểm của các thuật toán băm SHA.

**Bảng 2.1. Đặc điểm của các thuật toán băm SHA**

| Thuật toán  | Kết quả | Trạng thái | Khối | Thông điệp tối đa | Tù | Số chu kỳ | Các thao tác  | Độ đập            | Độ an toàn <sup>7</sup> |
|-------------|---------|------------|------|-------------------|----|-----------|---|-------------------|-------------------------|
| SHA-0       | 160     | 160        | 512  | $2^{64} - 1$      | 32 | 80        | $+, \text{and}, \text{or}, \text{xor}, \text{rotl}$             | Có                | 80                      |
| SHA-1       | 160     | 160        | 512  | $2^{64} - 1$      | 32 | 80        | $+, \text{and}, \text{or}, \text{xor}, \text{rotl}$             | $2^{63}$ thao tác | 80                      |
| SHA-256/224 | 256/224 | 256        | 512  | $2^{64} - 1$      | 32 | 64        | $+, \text{and}, \text{or}, \text{xor}, \text{shr}, \text{rotr}$ | Chưa              | 112/128                 |
| SHA-512/384 | 512/384 | 512        | 1024 | $2^{128} - 1$     | 64 | 80        | $+, \text{and}, \text{or}, \text{xor}, \text{shr}, \text{rotr}$ | Chưa              | 192/256                 |

<sup>7</sup> "Độ an toàn" là việc sử dụng phương pháp tấn công vào thông điệp rút gọn kích thước  $n$ , đòi hỏi xử lý xấp xỉ  $2^n$

Từ khi SHA-0 ra đời, rất nhiều kết quả nghiên cứu được công bố cho thấy thuật toán này cần phải được thay thế như của Florent Chabaud và Antonie Joux tại CRYPTO 98 [18]; của Biham và Chen năm 2004 [8]; của Joux, Carribault, Lemuet và Jalby ngày 12/8/2004 [30]; của Wang, Feng, Lai và Yu vào ngày 12/8/2004 tại CRYPTO 2004 [30]; và của Xiaoyun Wang, Yiqun Lisa Yin, và Hongbo Yu tháng 2/2005 [58].

Với các kết quả nghiên cứu được công bố đối với SHA-0, một số chuyên gia đề nghị rằng kế hoạch sử dụng SHA-1 trong các hệ thống mã hóa mới nên xem xét lại. Sau khi những kết quả của CRYPTO 2004 được công bố, NIST thông báo rằng họ dự định thôi không dùng SHA-1 sau 2010 với việc ủng hộ các biến thể SHA-2. Một số tấn công trên SHA-1 có thể kể đến như của Rijmen và Oswald năm 2005 [47]; của Xiaoyun Wang, Yiqun Lisa Yin và Hongbo Yu tháng 2/2005 [59], của Xiaoyun Wang, Andrew Yao and Frances Yao ngày 17/8/2005 tại CRYPTO 2005 [16].

Đối với các biến thể SHA-2, tuy Gilbert và Handschuh [24] đã nghiên cứu và không tìm ra điểm yếu của các biến thể SHA-2 nhưng trên thực tế chúng vẫn chưa được kiểm chứng kỹ như SHA-1. Mặc dù chưa có tấn công nào được ghi nhận trên các biến thể SHA-2, nhưng do về mặt thuật toán, SHA-2 không khác biệt mấy so với SHA-1 nên nhiều nhà khoa học đã bắt đầu phát triển một thuật giải khác tốt hơn SHA. Một cuộc thi tìm SHA-3 được thông báo một cách trang trọng trên Federal Register vào ngày 2/11/2007 với nội dung “*NIST bắt đầu nỗ lực để phát triển một hoặc nhiều thuật toán băm mới thông qua một cuộc thi công khai, giống như quy trình phát triển chuẩn mã hóa tiên tiến AES<sup>8</sup>*”. Theo kế hoạch, ngày 31/10/2008 sẽ tiến hành xem xét và dự định thời điểm công bố người thắng cuộc và chuẩn mới vào năm 2012 [64].

### **2.2.2.3 Một số hàm băm khác**

Ngoài MD5 và SHA, còn một số hàm băm khác như RIPEMD-128/160/256/320, Tiger và Whirlpool.

<sup>8</sup> Ngày 2/1/1997, NIST đã công bố một cuộc thi công khai nhằm tìm một thuật toán mã hóa quy ước có độ an toàn cao hơn DES, được gọi là Chuẩn mã hóa nâng cao AES (Advanced Encryption Standard).

## Hàm băm mật mã RIPEMD

- RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) là hàm băm mật mã cho thông điệp tóm tắt có độ lớn 160 bit, được phát triển bởi Hans Dobbertin, Antoon Bosselaers và Bart Preneel tại nhóm nghiên cứu COSIC tại đại học Leuven (Bỉ), và được công bố lần đầu tiên năm 1996. Nó là phiên bản cải tiến của RIPEMD, dựa trên các nguyên lý thiết kế được sử dụng trong MD4 và tương tự cách thực hiện của hàm băm phổ biến hơn là SHA-1.
- Ngoài RIPEMD-160 còn các phiên bản 128, 256 và 320 bit được gọi là RIPEMD-128, RIPEMD-256 và RIPEMD-320. Phiên bản RIPEMD-128 nhằm thay thế phiên bản RIPEMD gốc (cũng 128 bit) do có một số vấn đề về sự an toàn. Phiên bản RIPEMD-256 và RIPEMD-320 chỉ giảm bớt cơ hội xảy ra đụng độ mà không có các độ an toàn cao hơn so với RIPEMD-128 và RIPEMD-160 theo thứ tự đó.
- RIPEMD-160 được thiết kế trong cộng đồng học thuật mở, trái ngược với các nhóm các thuật toán được thiết kế bởi NSA như SHA. Mặc khác, RIPEMD-160 ít được sử dụng thường xuyên hơn SHA-1 do nó ít được khảo sát kỹ lưỡng hơn SHA-1.

## Hàm băm mật mã Tiger

- Tiger là một hàm băm mật mã được thiết kế bởi Ross Anderson và Eli Biham vào năm 1995 cho sự hiệu quả trên nền 64 bit. Độ lớn của giá trị băm Tiger là 192 bit. Phiên bản rút ngắn (Tiger/128 và Tiger/160) có thể được sử dụng cho tính tương thích với các giao thức cần một kích thước băm riêng biệt.
- Tiger thường được sử dụng ở dạng cây băm Merkle, được nhắc đến như là TTH (Tiger Tree Hash). TTH được sử dụng bởi nhiều khách hàng trên các mạng chia sẻ tập tin Direct Connect và Gnutella. Tiger được xem xét trong chuẩn OpenPGP, nhưng sau đó không được quan tâm do thuật toán RIPEMD-160 được ủng hộ hơn.
- Không giống MD5 hay SHA-0/1, không có tấn công nào được biết trên phiên bản 24 chu kỳ đầy đủ của Tiger. Trong khi MD5 xử lý các trạng thái của nó

với 64 thao tác 32 bit đơn giản mỗi khối 512 bit và SHA-1 là 80, Tiger cập nhật trạng thái của nó với tổng cộng 144 thao tác như thế trên khối 512 bit, hơn nữa được làm cho kiên cố hơn bởi bảng dò S-box.

### Hàm băm mật mã Whirlpool

- Whirlpool (hay WHIRLPOOL) là một hàm băm mật mã được thiết kế bởi Vincent Rijmen (đồng sáng lập của thuật toán AES) và Paulo S. L. M. Barreto [6]. Whirlpool được đề nghị bởi dự án NESSIE và được ISO<sup>9</sup> và IEC<sup>10</sup> chấp nhận như một phần liên kết của chuẩn quốc tế 10118-3 ISO/IEC. Các tác giả đã tuyên bố rằng “WHIRLPOOL không được và sẽ không bao giờ được cấp bằng sáng chế. Nó được sử dụng miễn phí cho bất kỳ trường hợp nào và được thực thi trong các lĩnh vực công khai”.
- Whirlpool là một kiến trúc Miyaguchi-Preneel dựa trên AES được thay đổi về căn bản. Cho trước một thông điệp ngắn hơn  $2^{256}$  bit, nó trả về một thông điệp tóm tắt 512 bit.
- Thuật toán được đặt tên sau Whirlpool Galaxy diễn ra ở Canes Venatici. Thuật toán Whirlpool đã trải qua hai lần chỉnh sửa kể từ đặc tả gốc năm 2000.

#### 2.2.3 Kết quả thử nghiệm và nhận xét

Tất cả thử nghiệm trong đề tài này được thực hiện trên môi trường như sau:

- Hệ điều hành: Windows Vista™ Home Premium (32 bit).
- Bộ xử lý: Intel® Core™ 2 Duo, CPU T9300 2.50GHz, 3.5 GB RAM.
- Ngôn ngữ lập trình: Java (JDK 1.6)

Để so sánh tốc độ của SHA-1 và MD5, Thử nghiệm 2.1 sau đã được tiến hành.

**Thử nghiệm 2.1:** Kích thước đầu vào lần lượt là 0.1 MB, 0.2 MB, ..., 0.5 MB, 1.0 MB, 1.5 MB, ..., 5.0 MB (chính là kích thước phổ biến của các văn bản hiện nay). Ứng với mỗi kích thước, chương trình tự động phát sinh ngẫu nhiên đầu vào và

<sup>9</sup> Tổ chức Tiêu chuẩn Quốc tế (International Organization for Standardization – ISO).

<sup>10</sup> Ủy ban Kỹ thuật Điện Quốc tế (International Electrotechnical Commission – IEC).

lần lượt tiến hành tính thời gian của 2 thuật toán MD5 và SHA-1. Thử nghiệm được lặp lại 50.000 lần.

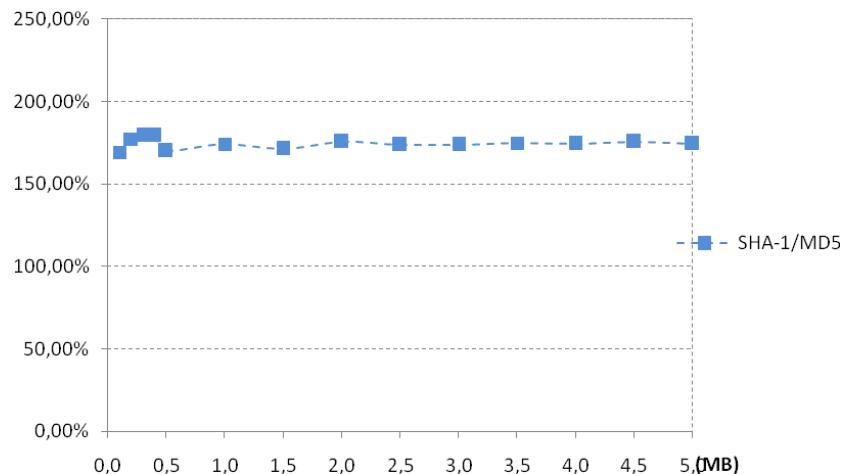
Kết quả nhận được như sau:

**Bảng 2.2. Thời gian băm của MD5 và SHA-1**

| Đầu vào<br>(MB) | Thời gian (giây)   |                      | (2)<br>(1) |
|-----------------|--------------------|----------------------|------------|
|                 | MD5 <sup>(1)</sup> | SHA-1 <sup>(2)</sup> |            |
| 0,1             | 0,0009             | 0,0016               | 169,11%    |
| 0,2             | 0,0018             | 0,0032               | 177,37%    |
| 0,3             | 0,0027             | 0,0049               | 179,37%    |
| 0,4             | 0,0036             | 0,0065               | 179,44%    |
| 0,5             | 0,0047             | 0,0080               | 170,17%    |
| 1,0             | 0,0089             | 0,0155               | 174,03%    |
| 1,5             | 0,0136             | 0,0233               | 171,47%    |

| Đầu vào<br>(MB) | Thời gian (giây)   |                      | (2)<br>(1) |
|-----------------|--------------------|----------------------|------------|
|                 | MD5 <sup>(1)</sup> | SHA-1 <sup>(2)</sup> |            |
| 2,0             | 0,0177             | 0,0311               | 175,63%    |
| 2,5             | 0,0222             | 0,0386               | 173,69%    |
| 3,0             | 0,0267             | 0,0464               | 173,72%    |
| 3,5             | 0,0309             | 0,0540               | 174,70%    |
| 4,0             | 0,0354             | 0,0618               | 174,37%    |
| 4,5             | 0,0394             | 0,0693               | 175,78%    |
| 5,0             | 0,0439             | 0,0766               | 174,50%    |



**Hình 2.5. Tỷ lệ thời gian băm giữa SHA-1 và MD5**

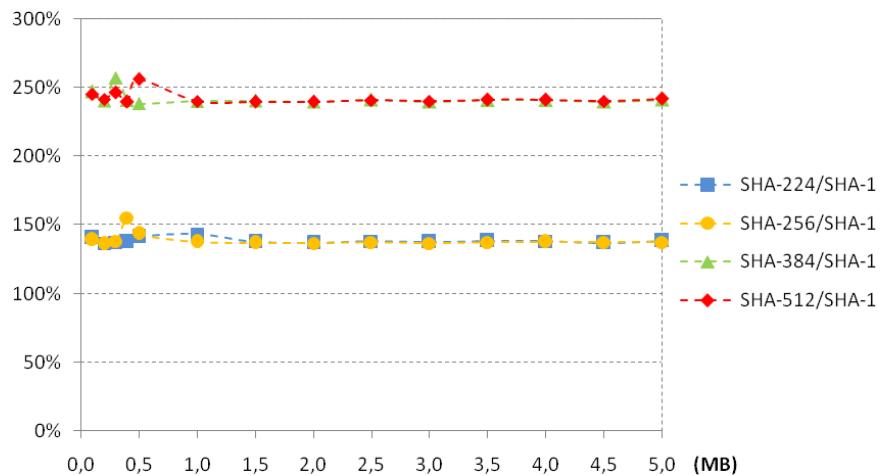
Kết quả của Thử nghiệm 2.1 cho thấy tốc độ của SHA-1 chỉ chậm hơn MD5 trung bình 75% khi kích thước đầu vào tăng dần. Tuy nhiên, với độ an toàn cao hơn và kích thước thông điệp tóm tắt lớn hơn MD5 (160 bit so với 128 bit) thì tốc độ này hoàn toàn hợp lý và chấp nhận được.

Các biến thể của SHA-1 là SHA-2 (gồm 4 thuật toán băm SHA-224, SHA-256, SHA-384 và SHA-512) cũng đang được sử dụng để mang lại mức độ an toàn cao hơn rất nhiều. Để so sánh tốc độ của các thuật toán này, Thử nghiệm 2.2 sau đây đã được tiến hành và ghi nhận.

**Thử nghiệm 2.2:** Quy trình thực hiện giống Thử nghiệm 2.1 nhưng sử dụng 5 thuật toán băm là SHA-1, SHA-224, SHA-256, SHA-384 và SHA-512. Kết quả nhận được như sau:

Bảng 2.3. Thời gian băm của SHA-1 và SHA-224/256/384/512

| Đầu vào<br>(MB) | Thời gian (giây)     |                        |                        |                        |                        | Tỷ lệ (%)  |            |            |            |
|-----------------|----------------------|------------------------|------------------------|------------------------|------------------------|------------|------------|------------|------------|
|                 | SHA-1 <sup>(1)</sup> | SHA-224 <sup>(2)</sup> | SHA-256 <sup>(3)</sup> | SHA-384 <sup>(4)</sup> | SHA-512 <sup>(5)</sup> | (2)<br>(1) | (3)<br>(1) | (4)<br>(1) | (5)<br>(1) |
| 0,1             | 0,0016               | 0,0022                 | 0,0022                 | 0,0039                 | 0,0039                 | 140,58%    | 139,13%    | 247,12%    | 244,92%    |
| 0,2             | 0,0032               | 0,0044                 | 0,0044                 | 0,0078                 | 0,0078                 | 136,63%    | 136,35%    | 240,17%    | 241,21%    |
| 0,3             | 0,0049               | 0,0067                 | 0,0067                 | 0,0125                 | 0,0120                 | 137,27%    | 138,14%    | 257,12%    | 246,20%    |
| 0,4             | 0,0065               | 0,0089                 | 0,0101                 | 0,0156                 | 0,0156                 | 137,64%    | 155,22%    | 240,50%    | 239,46%    |
| 0,5             | 0,0080               | 0,0113                 | 0,0115                 | 0,0191                 | 0,0205                 | 141,54%    | 143,74%    | 238,24%    | 255,88%    |
| 1,0             | 0,0155               | 0,0223                 | 0,0214                 | 0,0373                 | 0,0373                 | 143,25%    | 137,81%    | 240,03%    | 239,93%    |
| 1,5             | 0,0233               | 0,0321                 | 0,0319                 | 0,0559                 | 0,0557                 | 137,76%    | 136,79%    | 240,10%    | 239,36%    |
| 2,0             | 0,0311               | 0,0425                 | 0,0425                 | 0,0746                 | 0,0745                 | 136,78%    | 136,73%    | 239,70%    | 239,55%    |
| 2,5             | 0,0386               | 0,0534                 | 0,0530                 | 0,0930                 | 0,0929                 | 138,21%    | 137,39%    | 240,90%    | 240,62%    |
| 3,0             | 0,0464               | 0,0640                 | 0,0635                 | 0,1113                 | 0,1115                 | 137,71%    | 136,69%    | 239,55%    | 240,01%    |
| 3,5             | 0,0540               | 0,0747                 | 0,0741                 | 0,1299                 | 0,1301                 | 138,38%    | 137,33%    | 240,76%    | 241,15%    |
| 4,0             | 0,0618               | 0,0853                 | 0,0851                 | 0,1488                 | 0,1490                 | 138,00%    | 137,62%    | 240,76%    | 241,14%    |
| 4,5             | 0,0693               | 0,0950                 | 0,0950                 | 0,1660                 | 0,1663                 | 137,12%    | 137,18%    | 239,64%    | 240,02%    |
| 5,0             | 0,0766               | 0,1060                 | 0,1052                 | 0,1846                 | 0,1852                 | 138,39%    | 137,35%    | 240,99%    | 241,81%    |
| Trung Bình      |                      |                        |                        |                        |                        |            |            |            |            |
|                 |                      |                        |                        |                        |                        | 137,80%    | 137,13%    | 241,83%    | 240,46%    |



Hình 2.6. Tỷ lệ thời gian băm giữa SHA-2 và SHA-1

Do thông điệp tóm tắt nhận được của các thuật toán SHA-2 lớn hơn SHA-1 nên tốc độ của SHA-2 chậm hơn nhưng không quá nhiều. Cụ thể, so với SHA-1, SHA-224/256 chỉ chậm hơn trung bình 38% còn SHA-384/512 chậm hơn trung bình 142%.

Các thuật toán băm khác như RIPEMD-128/160/256/320, Tiger/192 và Whirlpool/512 không được sử dụng rộng rãi do tính lịch sử cũng nhưng chưa được xem xét kỹ lưỡng. Thử nghiệm 2.3 sau đã được thực hiện để so sánh tốc độ của Whirlpool và thuật toán SHA cùng kích thước thông điệp tóm tắt là SHA-512.

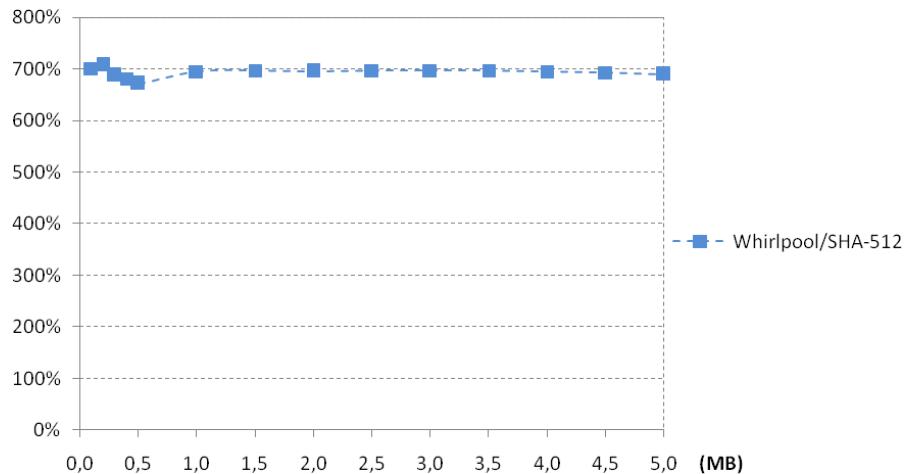
**Thử nghiệm 2.3:** Quy trình thực hiện giống Thử nghiệm 2.1 nhưng sử dụng 2 thuật toán băm là Whirlpool và SHA-512. Kết quả thử nghiệm như sau:

**Bảng 2.4. Thời gian băm của SHA-512 và Whirlpool**

| Đầu vào<br>(MB) | Thời gian (giây)           |                               | (2)<br>(1) |
|-----------------|----------------------------|-------------------------------|------------|
|                 | SHA-<br>512 <sup>(1)</sup> | Whirl-<br>pool <sup>(2)</sup> |            |
| 0,1             | 0,0039                     | 0,0273                        | 701,11%    |
| 0,2             | 0,0078                     | 0,0552                        | 707,55%    |
| 0,3             | 0,0120                     | 0,0823                        | 688,26%    |
| 0,4             | 0,0156                     | 0,1059                        | 680,84%    |
| 0,5             | 0,0205                     | 0,1377                        | 672,37%    |
| 1,0             | 0,0373                     | 0,2591                        | 695,05%    |
| 1,5             | 0,0557                     | 0,3888                        | 697,53%    |

| Đầu vào<br>(MB) | Thời gian (giây)           |                               | (2)<br>(1) |
|-----------------|----------------------------|-------------------------------|------------|
|                 | SHA-<br>512 <sup>(1)</sup> | Whirl-<br>pool <sup>(2)</sup> |            |
| 2,0             | 0,0745                     | 0,5190                        | 696,58%    |
| 2,5             | 0,0929                     | 0,6476                        | 697,23%    |
| 3,0             | 0,1115                     | 0,7775                        | 697,48%    |
| 3,5             | 0,1301                     | 0,9066                        | 696,78%    |
| 4,0             | 0,1490                     | 1,0356                        | 694,93%    |
| 4,5             | 0,1663                     | 1,1523                        | 692,89%    |
| 5,0             | 0,1852                     | 1,2785                        | 690,19%    |



**Hình 2.7. Tỷ lệ thời gian băm giữa Whirlpool và SHA-512**

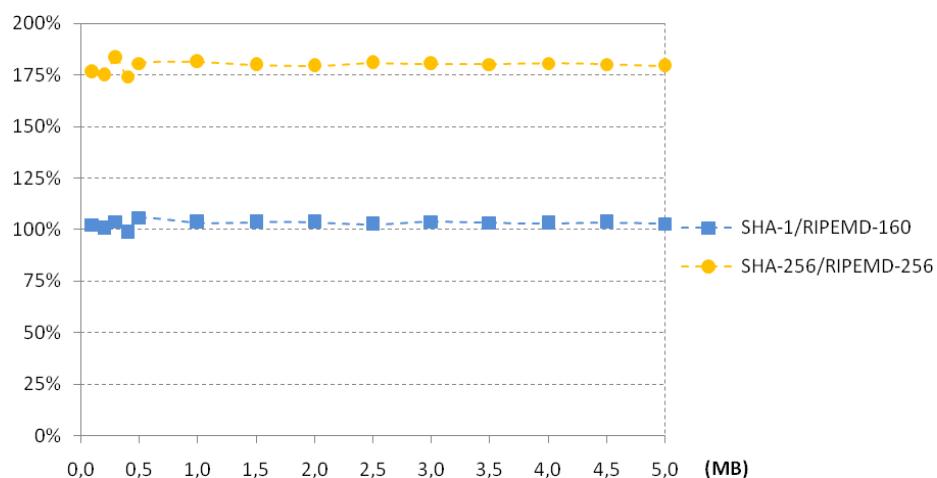
Kết quả Thử nghiệm 2.3 cho thấy tốc độ của Whirlpool rất thấp, chậm hơn trung bình 594% tốc độ của hàm băm cho cùng độ lớn thông điệp tóm tắt là SHA-512.

Để so sánh tốc độ của các thuật toán RIPEMD với các hàm băm SHA cho cùng kích thước thông điệp tóm tắt (cụ thể RIPEMD-160 so với SHA-1 và RIPEMD-256 so với SHA-256), Thử nghiệm 2.4 sau đã được tiến hành.

**Thử nghiệm 2.4:** Quy trình thực hiện giống Thử nghiệm 2.1 nhưng sử dụng 4 thuật toán băm là SHA-1, SHA-256, RIPEMD-160, RIPEMD-256. Kết quả như sau:

**Bảng 2.5. Thời gian băm của SHA-1 và RIPEMD-160, SHA-256 và RIPEMD-256**

| Đầu vào<br>(MB) | Thời gian (giây)              |                          |                               |                            | Tỷ lệ (%)  |            |
|-----------------|-------------------------------|--------------------------|-------------------------------|----------------------------|------------|------------|
|                 | RIPEMD-<br>160 <sup>(1)</sup> | SHA-<br>1 <sup>(2)</sup> | RIPEMD-<br>256 <sup>(3)</sup> | SHA-<br>256 <sup>(4)</sup> | (2)<br>(1) | (4)<br>(3) |
| 0,1             | 0,0016                        | 0,0016                   | 0,0013                        | 0,0022                     | 102,21%    | 176,44%    |
| 0,2             | 0,0032                        | 0,0032                   | 0,0025                        | 0,0044                     | 100,56%    | 174,93%    |
| 0,3             | 0,0047                        | 0,0049                   | 0,0037                        | 0,0067                     | 103,82%    | 183,27%    |
| 0,4             | 0,0066                        | 0,0065                   | 0,0058                        | 0,0101                     | 98,69%     | 174,16%    |
| 0,5             | 0,0076                        | 0,0080                   | 0,0064                        | 0,0115                     | 105,79%    | 180,55%    |
| 1,0             | 0,0150                        | 0,0155                   | 0,0118                        | 0,0214                     | 103,49%    | 181,30%    |
| 1,5             | 0,0225                        | 0,0233                   | 0,0177                        | 0,0319                     | 103,62%    | 179,95%    |
| 2,0             | 0,0301                        | 0,0311                   | 0,0237                        | 0,0425                     | 103,49%    | 179,47%    |
| 2,5             | 0,0377                        | 0,0386                   | 0,0293                        | 0,0530                     | 102,51%    | 180,80%    |
| 3,0             | 0,0447                        | 0,0464                   | 0,0352                        | 0,0635                     | 103,88%    | 180,37%    |
| 3,5             | 0,0523                        | 0,0540                   | 0,0411                        | 0,0741                     | 103,23%    | 180,24%    |
| 4,0             | 0,0599                        | 0,0618                   | 0,0471                        | 0,0851                     | 103,11%    | 180,60%    |
| 4,5             | 0,0670                        | 0,0693                   | 0,0528                        | 0,0950                     | 103,43%    | 180,12%    |
| 5,0             | 0,0744                        | 0,0766                   | 0,0586                        | 0,1052                     | 102,91%    | 179,51%    |
| Trung Bình      |                               |                          |                               | 102,91%                    | 179,41%    |            |



**Hình 2.8. Tỷ lệ thời gian băm  
giữa SHA-1 và RIPEMD-160, SHA-256 và RIPEMD-256**

Thử nghiệm 2.4 cho thấy, tốc độ của SHA-1 xấp xỉ RIPEMD-160 còn tốc độ của SHA-256 chậm hơn trung bình 80% tốc độ của RIPEMD-256 nhưng các thuật toán SHA lại mang đến độ an toàn cao hơn rất nhiều với cùng thông điệp tóm tắt.

## 2.3 Thuật toán chữ ký số

### 2.3.1 Giới thiệu

Chữ ký số giúp xác định được người tạo ra hay chịu trách nhiệm đối với một thông điệp được ký. Một phương pháp chữ ký số phải bao gồm ít nhất 3 thuật toán chính, đó là thuật toán dùng để *tạo khóa*, thuật toán dùng để *tạo ra chữ ký số* và thuật toán tương ứng để *xác nhận chữ ký số*.

### 2.3.2 Một số thuật toán chữ ký số thông dụng

#### 2.3.2.1 Thuật toán chữ ký số RSA

Phương pháp chữ ký số RSA được xây dựng dựa trên thuật toán mã hóa khóa công khai RSA (sẽ được trình bày chi tiết ở Chương 5).

Để tạo một cặp khóa, RSA thực hiện các bước sau:

- Chọn 2 số nguyên tố lớn ngẫu nhiên  $p, q$ . Nhằm có sự an toàn tối đa nên chọn  $p$  và  $q$  có độ dài bằng nhau.
- Tính  $n = pq$  và  $\varphi = (p - 1)(q - 1)$ .
- Chọn ngẫu nhiên một số nguyên  $e$  ( $1 < e < \varphi$ ) sao cho  $gcd(e, \varphi) = 1$  với  $gcd$  là ước số chung lớn nhất.
- Tính:  $d = e^{-1} \pmod{\varphi}$ .

Kết quả là ta có được cặp khóa: khóa công khai  $(n, e)$  và khóa bí mật  $(n, d)$ .

Hai người sẽ sử dụng chung một hàm băm  $\mathcal{H}$  an toàn trước hiện tượng xung đột. Để ký một thông điệp  $m$ , người ký thực hiện các bước sau:

- Dùng hàm băm  $\mathcal{H}$  để băm thông điệp  $m$ :  $\overline{m} = \mathcal{H}(m)$ .
- Sử dụng khóa bí mật  $(n, d)$  để tính:  $s = \overline{m}^d \pmod{n}$ .

Chữ ký của  $m$  là  $s$  và được gửi kèm với thông điệp  $m$  đến người nhận.

Để xác nhận chữ ký, người nhận thực hiện các bước sau:

- Sử dụng khóa công khai  $(n, e)$  của người ký để giải mã chữ ký:  

$$\overline{s} = s^e \pmod{n}$$
- Sử dụng cùng hàm băm  $\mathcal{H}$  với người ký để băm thông điệp  $m$ :  $\overline{m}' = \mathcal{H}(m)$ .
- Chấp nhận chữ ký nếu  $\overline{s} = \overline{m}'$ . Ngược lại từ chối chữ ký.

### 2.3.2.2 Thuật toán chữ ký số ElGamal

Thuật toán chữ ký số ElGamal được Taher ElGamal giới thiệu vào năm 1984 [22], dựa trên tính khó giải của bài toán logarit rời rạc trên trường hữu hạn. Thuật toán chữ ký số ElGamal ít khi được sử dụng trong thực tế. Một biến thể của nó được phát triển bởi NSA là DSA được sử dụng rộng rãi hơn (sẽ được trình bày ở phần sau). Khác với thuật toán chữ ký số RSA có thể áp dụng trong bài toán mã hóa công khai và bài toán chữ ký số, thuật toán ElGamal được xây dựng chỉ nhằm giải quyết bài toán chữ ký số. Thuật toán chữ ký số ElGamal cho phép người kiểm tra có thể xác nhận tính xác thực của thông điệp  $m$  được người ký gửi đến trên một kênh truyền không an toàn.

Các tham số hệ thống sau đây được chọn và chia sẻ giữa những người sử dụng.

- $\mathcal{H}$  là hàm băm an toàn trước hiện tượng xung đột.
- $p$  là số nguyên tố lớn ngẫu nhiên sao cho việc tính loragit rời rạc *modulo p* khó khăn.
- $g$  là phần tử sinh (*modulo p*) ngẫu nhiên.

Quy trình tạo khóa cho mỗi người như sau:

- Chọn ngẫu nhiên một khóa bí mật  $x$  với  $1 < x < p - 1$ .
- Tính  $y = g^x \text{ mod } p$ .
- Khóa công khai là  $(p, g, y)$ , khóa bí mật là  $x$ .

Để ký một thông điệp  $m$ , người ký thực hiện các bước sau:

- Chọn một số ngẫu nhiên  $k$  sao cho  $0 < k < p - 1$  và  $\gcd(k, p - 1) = 1$ .
- Tính  $r \equiv g^k \pmod{p}$ .
- Tính  $s \equiv (\mathcal{H}(m) - xr)k^{-1} \pmod{p - 1}$ .
- Nếu  $s = 0$  tính lại từ đầu.

Cặp  $(r, s)$  là chữ ký số của  $m$ . Người ký lặp lại các bước trên cho mỗi lần ký.

Người xác nhận chấp nhận chữ ký nếu tất cả điều kiện sau thỏa mãn, ngược lại từ chối chữ ký:

- $0 < r < p$  và  $0 < s < p - 1$ .
- $g^{\mathcal{H}(m)} \equiv y^r r^s \pmod{p}$ .

Tính đúng đắn của giải thuật được chứng minh như sau:

- Việc tạo chữ ký dẫn đến:  $\mathcal{H}(m) \equiv xr + sk \pmod{p-1}$ .
- Do đó định lý Fermat nhỏ dẫn đến:

$$g^{\mathcal{H}(m)} \equiv g^{xr} g^{ks} \equiv g^x r^k s \equiv y^r r^s \pmod{p}.$$

Tổ chức thứ ba có thể giả mạo chữ ký bằng cách tìm khóa bí mật  $x$  của người ký hay tìm sự xung đột trong hàm băm  $\mathcal{H}(m) \equiv \mathcal{H}(M) \pmod{p-1}$ . Tuy nhiên, cả hai vấn đề này được xem là khó giải quyết.

Người ký phải chú ý chọn giá trị  $k$  ngẫu nhiên đồng dạng cho mỗi chữ ký và chắc chắn rằng không để lộ  $k$  hoặc thậm chí một phần thông tin về  $k$ . Nếu không thì kẻ tấn công có thể loại trừ các khóa bí mật  $x$  với khó khăn được giảm bớt đủ để cho một tấn công thực tế có thể thực hiện được. Cụ thể là nếu hai thông điệp được gửi sử dụng cùng giá trị  $k$  hoặc cùng một khóa, kẻ tấn công có thể tính  $x$  một cách trực tiếp.

### 2.3.2.3 Thuật toán chữ ký số DSA

Thuật toán chữ ký số DSA (Digital Signature Algorithm), là sự cải tiến của phương pháp ElGamal, được đề nghị bởi NIST vào tháng 8/1991 để sử dụng trong chuẩn chữ ký số DSS (Digital Signature Standard), được chỉ ra trong FIPS 186 [67], được chấp nhận năm 1993. Một sửa đổi nhỏ được đưa ra ngày năm 1996 trong FIPS 186-1 [67], chuẩn được mở rộng hơn năm 2000, được xem như xem như FIPS 186-2 [67].

Việc tạo khóa gồm hai bước. Bước thứ nhất là lựa chọn các tham số cho thuật toán được chia sẻ giữa các người sử dụng khác nhau trong cùng hệ thống:

- Chọn một hàm băm mã hóa  $\mathcal{H}$ . Trong DSS chuẩn  $\mathcal{H}$  luôn là SHA-1, nhưng các hàm băm tốt hơn trong nhóm SHA cũng đang được sử dụng. Đôi khi đầu ra của một thuật toán băm mới hơn bị rút ngắn kích thước so với các thuật toán băm mới cũ để tương tích với cặp khóa hiện có.
- Chọn kích thước khóa  $L$ . Đây là thước đo chính quyết định sức mạnh mã hóa của khóa. DSS chuẩn ràng buộc  $L$  là bội số của 64 và  $512 \leq L \leq 1024$ . Sau đó, FIPS 186-2 xác định  $L$  luôn là 1024. Không lâu sau, NIST 800-57 đề nghị độ dài khóa là 2048 (hoặc 3072) để thời gian an toàn đến năm 2010 (hoặc

2030), sử dụng tương ứng với các giá trị băm và  $q$  dài hơn. Bản thảo FIPS 186-3 [67] cũng tính đến các hàm băm sau này và các khóa dài hơn.

- Chọn một số nguyên tố  $q$  cùng số bit với đầu ra của  $\mathcal{H}$ .
- Chọn một số nguyên tố  $p$  độ dài  $L$  bit sao cho  $p-1$  là bội của  $q$ . Tức là  $p = qz - 1$  với số nguyên  $z$  nào đó.
- Chọn  $g = \boxed{?}^{(p-1)/q} \text{ mod } p$  với  $\boxed{?}$  bất kỳ ( $1 < \boxed{?} < p-1$ ), và chọn lại nếu kết quả là 1. Hầu hết cách chọn  $h$  đều nhận được  $g$  có thể sử dụng, thông thường chọn  $\boxed{?} = 2$ .

Các tham số thuật toán  $(p, q, g)$  có thể chia sẻ giữa những người khác nhau trong hệ thống. Bước thứ hai tính các khóa bí mật và công khai của từng người:

- Chọn  $x$  ngẫu nhiên sao cho  $0 < x < q$ .
- Tính  $y = g^x \text{ mod } p$ .
- Khóa công khai là  $p, q, g, y$ , khóa bí mật là  $x$ .

Phiên bản FIPS 186-3 sắp tới sử dụng SHA-224/256/384/512 là các hàm băm, kích thước của  $q$  là 224 (hoặc 256 bit), và  $L$  bằng 2048 (hoặc 3072).

Để ký một thông điệp  $m$ , người ký thực hiện các bước sau:

- Phát sinh một số ngẫu nhiên  $k$  ( $0 < k < q$ ) cho mỗi thông điệp.
- Tính  $r = (g^k \text{ mod } p) \text{ mod } q$ .
- Tính  $s = k^{-1}(\mathcal{H}(m) + xr) \text{ mod } q$ .
- Tính toán lại chữ ký trong trường hợp không chắc chắn  $r = 0$  hoặc  $s = 0$ .
- Chữ ký là  $(r, s)$ .

Để xác nhận chữ ký, người nhận thực hiện các bước sau:

- Loại bỏ chữ ký nếu  $0 < r < q$  hoặc  $0 < s < q$  không thỏa mãn.
- Tính  $w = s^{-1} \text{ mod } q$ .
- Tính  $u_1 = (\mathcal{H}(m) \times w) \text{ mod } q$ .
- Tính  $u_2 = (r \times w) \text{ mod } q$ .
- Tính  $v = g^{u_1} \times y^{u_2} \text{ mod } p \text{ mod } q$ .
- Chữ ký có hiệu lực nếu  $v = r$ .

Tính đúng đắn của giải thuật được chứng minh như sau:

- Đầu tiên, nếu  $g = \mathbb{F}^{(p-1)/q} \text{ mod } p$  suy ra  $gq \equiv \mathbb{F}^{p-1} \equiv 1 \text{ (mod } p)$  theo định lý Fermat nhỏ. Bởi vì  $g > 1$  và  $q$  là số nguyên tố nên  $g$  có bậc  $q$ .
- Người ký tính  $s = k^{-1}(\mathcal{H}(m) + xr) \text{ mod } q$ .
- Như vậy  $k \equiv \mathcal{H}(m)s^{-1} + xrs^{-1} \equiv \mathcal{H}(m)w + xrw \text{ (mod } q)$ .
- Bởi vì  $g$  có bậc  $q$  nên ta có:  

$$g^k \equiv g^{\mathcal{H}(m)w} g^{xrw} \equiv g^{\mathcal{H}(m)w} y^{rw} \equiv g^{u_1} y^{u_2} \text{ (mod } p).$$
- Cuối cùng, tính đúng đắn của DSA suy ra từ:  

$$r = g^k \text{ mod } p \text{ mod } q = g^{u_1} y^{u_2} \text{ mod } p \text{ mod } q = v.$$

Độ an toàn của thuật toán ElGamal phụ thuộc vào độ phức tạp của việc tìm lời giải cho bài toán logarit rời rạc nên cần phải sử dụng số nguyên tố  $p$  đủ lớn (tối thiểu là 512 bit). Nếu số nguyên tố  $p$  dài 512 bit thì chữ ký điện tử tạo ra có độ dài là 1024 bit và không phù hợp với các ứng dụng sử dụng thẻ thông minh vốn có nhu cầu sử dụng chữ ký ngắn. Phương pháp DSA đã giải quyết vấn đề này bằng cách sử dụng chữ ký 320 bit cho văn bản 160 bit với các phép tính được thực hiện trên tập con có  $2^{160}$  phần tử với  $p$  là số nguyên tố 512 bit.

#### 2.3.2.4 Thuật toán chữ ký số ECDSA

ECDSA (Elliptic Curve DSA) là một biến thể của thuật toán chữ ký số DSA, được thực hiện trên đường cong elliptic. Cũng như mã hóa đường cong elliptic nói chung, kích thước theo bit của khóa công khai cần thiết cho ECDSA là khoảng hai lần kích thước của độ an toàn (theo bit). Khi so sánh độ an toàn của 80 bit, nghĩa là người tấn công cần khoảng  $2^{80}$  thao tác tạo chữ ký để tìm khóa bí mật, kích thước của khóa công khai DSA ít nhất là 1024 bit, nhưng ngược lại kích thước của khóa công khai ECDSA sẽ là 160 bit. Mặt khác, kích thước chữ ký của DSA và ECDSA là nhau và bằng  $4t$  bit trong đó  $t$  là độ an toàn theo bit, nghĩa là khoảng 320 bit cho độ an toàn của 80 bit.

Để thực hiện tạo và xác nhận chữ ký điện tử bằng ECDSA, cần thống nhất các tham số được sử dụng sau:

- Đường cong ellipse  $E$ .
- Điểm  $P \in E$ . Điểm  $P$  có bậc  $n$  ( $n \times P = 0$ ).

- Chọn một số nguyên bất kỳ  $d, d \in 2, n - 2$ . Đây chính là khóa bí mật.
- Tính giá trị của điểm  $Q = d \times P, Q \in E$ .  $Q$  chính là khóa công khai.

Các tham số  $E, P, Q$  được công khai, tham số  $d$  được giữ bí mật và chỉ sử dụng trong quá trình tạo khóa.

Để ký một thông điệp  $m$ , người ký thực hiện các bước sau:

- Tạo thông điệp rút gọn của thông điệp  $m$  bằng hàm băm  $\mathcal{H}$ , sau đó, chuyển thành một số nguyên  $e$ .
- Chọn một số nguyên ngẫu nhiên  $k \in 2, n - 2$ . Đây là giá trị bí mật khác nhau cho mỗi lần tạo chữ ký.
- Tính giá trị của điểm  $x, y = k \times P$  và biểu diễn  $x$  dưới dạng số nguyên  $z$ .
- Tính giá trị  $r = z \bmod n$ .
- Tính giá trị  $s = k^{-1}(e + dr) \bmod n$ .

Cặp số nguyên  $(r, s)$  chính là chữ ký của thông điệp  $m$  và được gởi kèm với thông điệp  $m$  đến người nhận.

Gọi  $m'$ ,  $r'$ ,  $s'$  là các phiên bản nhận được của  $m, r, s$ .

- Tạo thông điệp rút gọn của  $m'$  bằng hàm băm  $\mathcal{H}$ . Hàm băm  $\mathcal{H}$  phải là hàm băm được sử dụng trong quá trình tạo chữ ký. Biểu diễn thông điệp rút gọn thu được dưới dạng một số nguyên  $e$ .
- $c = s' - 1 \bmod n; u_1 = e \times c \bmod n; u_2 = r \times c \bmod n$
- Tính giá trị  $c = s' - 1 \bmod n; u_1 = e \times c \bmod n; u_2 = r \times c \bmod n$ .
- Tính giá trị điểm  $x, y = u_1 \times P + u_2 \times Q$ , biểu diễn  $x$  dưới dạng số nguyên  $z$ .
- Tính  $v = z \bmod n$ .
- Nếu  $v = r'$ , chữ ký điện tử được xác nhận và người nhận có thể đảm bảo văn bản được gởi là xác thực.
- Nếu  $v \neq r'$ , văn bản đã bị sửa đổi hoặc văn bản không được ký bằng đúng chữ ký của người gởi. Thông tin văn bản được xem là không hợp lệ.

### 2.3.3 Kết quả thử nghiệm và nhận xét

#### 2.3.3.1 So sánh RSA và DSA

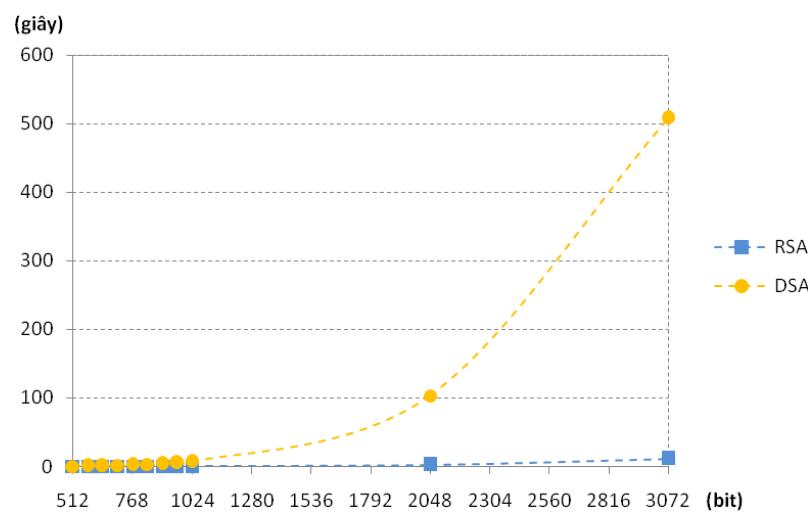
Để so sánh tốc độ của hai thuật toán chữ ký số RSA và DSA, Thử nghiệm 2.5 dưới đây đã được tiến hành và ghi nhận.

**Thử nghiệm 2.5:** DSS chuẩn ràng buộc độ dài khóa  $L$  là bội số của 64 và  $512 \leq L \leq 1024$  và để an toàn lâu dài độ dài khóa  $L$  được đề nghị là 2048 hoặc 3072. Do đó độ dài khóa được thử nghiệm cho cả RSA và DSA là 576, 640, 704, 768, 832, 896, 960, 1024, 2048, 3072 (bit). Ứng với mỗi độ dài khóa, lần lượt cho cả RSA và DSA phát sinh khóa, ký văn bản ngẫu nhiên (kích thước 2 MB) và kiểm tra chữ ký tạo được. Để thuận tiện so sánh, hàm băm mật mã SHA-1 được chọn để sử dụng cho cả RSA và DSA. Thử nghiệm được lặp lại 50.000 lần. Kết quả nhận được như sau:

**Bảng 2.6. So sánh thời gian tạo khóa, tạo chữ ký và xác nhận chữ ký của RSA**

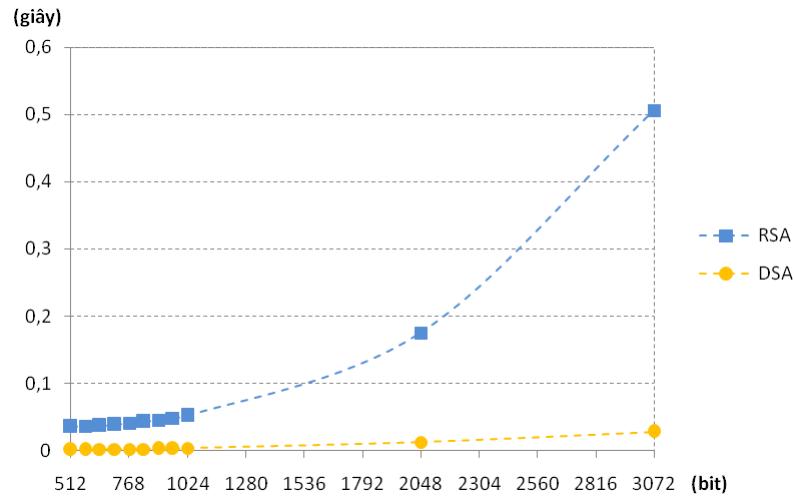
với DSA

| Kích<br>thước<br>(bit) | Tạo khóa (giây) |          |             | Tạo chữ ký (giây) |        |             | Xác nhận chữ ký (giây) |        |             |
|------------------------|-----------------|----------|-------------|-------------------|--------|-------------|------------------------|--------|-------------|
|                        | RSA             | DSA      | DSA/<br>RSA | RSA               | DSA    | RSA/<br>DSA | RSA                    | DSA    | RSA/<br>DSA |
| 512                    | 0,0408          | 0,5676   | 13,93       | 0,0351            | 0,0011 | 32,60       | 0,0320                 | 0,0017 | 19,32       |
| 576                    | 0,0568          | 0,8030   | 14,14       | 0,0361            | 0,0013 | 27,24       | 0,0321                 | 0,0022 | 14,60       |
| 640                    | 0,0757          | 1,2464   | 16,47       | 0,0371            | 0,0015 | 24,53       | 0,0319                 | 0,0025 | 12,57       |
| 704                    | 0,0994          | 1,7948   | 18,06       | 0,0387            | 0,0019 | 20,25       | 0,0320                 | 0,0031 | 10,16       |
| 768                    | 0,1278          | 2,3668   | 18,52       | 0,0408            | 0,0016 | 25,29       | 0,0321                 | 0,0040 | 7,94        |
| 832                    | 0,1609          | 3,0526   | 18,97       | 0,0428            | 0,0021 | 20,31       | 0,0322                 | 0,0044 | 7,34        |
| 896                    | 0,2026          | 4,2369   | 20,92       | 0,0454            | 0,0027 | 16,58       | 0,0321                 | 0,0050 | 6,36        |
| 960                    | 0,2446          | 5,4622   | 22,33       | 0,0480            | 0,0026 | 18,45       | 0,0321                 | 0,0061 | 5,29        |
| 1024                   | 0,2734          | 7,1210   | 26,05       | 0,0515            | 0,0035 | 14,86       | 0,0318                 | 0,0068 | 4,69        |
| 2048                   | 2,4876          | 103,1124 | 41,45       | 0,1749            | 0,0124 | 14,16       | 0,0325                 | 0,0240 | 1,35        |
| 3072                   | 11,1882         | 508,2395 | 45,43       | 0,5056            | 0,0278 | 18,19       | 0,0341                 | 0,0539 | 0,63        |



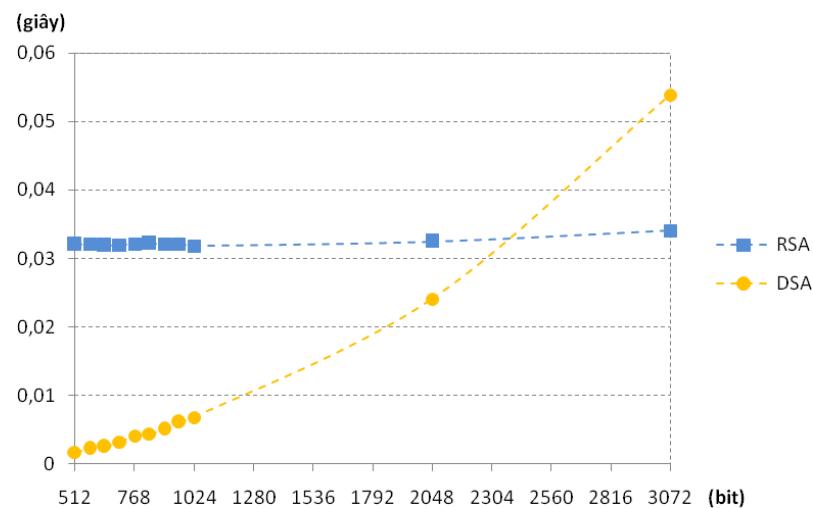
**Hình 2.9. Thời gian tạo khóa của RSA và DSA**

Kết quả Thử nghiệm 2.5 cho thấy tốc độ tạo khóa của RSA nhanh hơn rất nhiều so với DSA và khi kích thước khóa tăng lên thì tỷ lệ này ngày càng gia tăng. Hơn nữa, khi tăng kích thước  $L$  của DSA và tương ứng với các hàm băm SHA có đầu ra lớn hơn thì DSA sẽ còn chậm hơn rất nhiều.



**Hình 2.10. Thời gian tạo chữ ký của RSA và DSA**

Kết quả Thử nghiệm 2.5 cho thấy tốc độ tạo chữ ký của RSA chậm hơn DSA nhưng tỷ lệ này có xu hướng giảm khi kích thước khóa tăng lên. Nguyên nhân là do khi số mũ khóa công khai  $e$  cố định thì số mũ khóa bí mật  $d$  sẽ tăng khi kích thước  $n$  tăng. Mặt khác, phép tính chiếm thời gian nhiều nhất của quy trình ký chính là phép lũy thừa *modulo* nên khi số mũ tăng thì thời gian thực hiện cũng sẽ tăng. Tuy nhiên, kích thước khóa được sử dụng phổ biến hiện nay là 1024 và 2048 nên thời gian ký lúc này sẽ không còn là vấn đề đáng lo ngại do toàn bộ quy trình chỉ mất ít hơn 0,2 giây.



**Hình 2.11. Thời gian xác nhận chữ ký của RSA và DSA**

Kết quả Thử nghiệm 2.5 cũng cho thấy tốc độ xác nhận chữ ký của RSA không thay đổi đáng kể khi kích thước khóa tăng do số mũ công khai  $e$  được sử dụng luôn là một số nhỏ (giá trị phổ biến hiện nay là 65537) và tốc độ thực hiện phép lũy thừa

*modulo* (phép toán chính trong quy trình xác nhận chữ ký) sẽ tăng không nhiều. Ngược lại, tốc độ xác nhận chữ ký của DSA mặc dù thấp hơn RSA nhưng sẽ ngày càng tăng khi kích thước khóa tăng lên. Nguyên nhân là do quy trình xác nhận chữ ký của DSA gồm rất nhiều phép tính tốn chi phí cao (phép lũy thừa *modulo* và phép nhân) nên khi kích thước khóa tăng dần thì điều này sẽ trở thành gánh nặng. Mặc khác, nếu kích thước  $L$  được chọn lớn hơn thì tốc độ xác nhận chữ ký sẽ chậm hơn nữa.

Với các thử nghiệm trên ta dễ dàng nhận thấy RSA tốt hơn DSA về mọi mặt, đặc biệt là tốc độ phát sinh khóa của RSA nhanh hơn DSA rất nhiều. Ngoài ra, tốc độ ký và xác nhận chữ ký đầu có chậm hơn DSA nhưng thời gian này là không đáng kể.

### 2.3.3.2 So sánh RSA với ECDSA

Đối với phương pháp chữ ký số ECDSA, ưu điểm của hệ mã khóa công khai ECC được sử dụng trong ECDSA là khả năng bảo mật cao với kích thước khóa nhỏ dựa vào mức độ khó giải quyết của vấn đề ECDLP. Đây chính là một tính chất rất hữu ích đối với xu hướng ngày nay là tìm ra phương pháp tăng độ an toàn của mã hóa khóa công khai với kích thước khóa được rút gọn. Kích thước khóa nhỏ hơn giúp thu gọn được kích thước của chứng nhận giao dịch trên mạng và giảm kích thước tham số của hệ thống mã hóa. Bảng sau thể hiện kết quả so sánh kích thước khóa công khai của phương pháp RSA và ECC với cùng độ an toàn<sup>11</sup>.

**Bảng 2.7. So sánh kích thước khóa RSA và ECC với cùng độ an toàn**

| Độ an toàn | Kích thước khóa công khai (bit) <sup>12</sup> | Tỷ lệ kích thước khóa RSA : ECC |
|------------|---|---------------------------------|
| RSA        | ECC   |                                 |
| 80         | 1024  | 192                             |
| 112        | 2048  | 224                             |
| 128        | 3072  | 256                             |
| 192        | 7680  | 384                             |
| 256        | 15360   | 521                             |

<sup>11</sup> Nguồn: Certicom Corp. <http://www.certicom.com>

<sup>12</sup> Kích thước khóa được NIST đề nghị.

Do có kích thước khóa nhỏ và khả năng phát sinh khóa rất nhanh nên ECC rất được quan tâm để áp dụng cho các ứng dụng trên môi trường giới hạn về thông lượng truyền dữ liệu, giới hạn về khả năng tính toán, khả năng lưu trữ. ECC thích hợp với các thiết bị di động kỹ thuật số như Pocket PC, PDA, điện thoại di động và thẻ thông minh. Bảng sau cho thấy tốc độ tạo chữ ký và xác nhận chữ ký của ECDSA vượt trội so với RSA với cùng độ an toàn.

**Bảng 2.8. So sánh tốc độ tạo và xác nhận chữ ký của RSA và ECDSA với cùng độ an toàn<sup>13</sup>**

| Độ lớn<br>khóa<br>RSA | Độ lớn<br>khóa<br>ECC | Tạo chữ ký                     |                                  |            | Xác nhận chữ ký                |                                  |            |
|-----------------------|-----------------------|--------------------------------|----------------------------------|------------|--------------------------------|----------------------------------|------------|
|                       |                       | RSA <sup>(1)</sup><br>(1 phút) | ECDSA <sup>(2)</sup><br>(1 phút) | (2)<br>(1) | RSA <sup>(3)</sup><br>(1 phút) | ECDSA <sup>(4)</sup><br>(1 phút) | (4)<br>(3) |
| 2048                  | 224                   | 2940                           | 105840                           | 36         | 26880                          | 47520                            | 1,77       |
| 3072                  | 256                   | 480                            | 54000                            | 112,5      | 11280                          | 22800                            | 2,02       |
| 7680                  | 384                   | 60                             | 30960                            | 516        | 2160                           | 11040                            | 5,11       |
| 15360                 | 521                   | 60                             | 14400                            | 240        | 480                            | 5280                             | 11         |

Tuy nhiên, ECC vẫn có một số hạn chế nhất định. Hạn chế lớn nhất hiện nay là việc chọn sử dụng các tham số đường cong và điểm quy ước chung như thế nào để thật sự đạt được độ an toàn cần thiết. Hầu hết các đường cong được đưa ra đều thất bại khi áp dụng vào thực tiễn. Do đó hiện nay số lượng đường cong thật sự được sử dụng không được phong phú. NIST đề xuất một số đường cong elliptic đã được kiểm định là an toàn để đưa vào sử dụng thực tế trong tài liệu FIPS 186-2 [67]. Ngoài ra, đối với các tham số mang giá trị nhỏ, độ an toàn của ECC không bằng RSA (khi  $e = 3$ ).

ECC vẫn còn non trẻ và cần được kiểm định trong tương lai để có thể ứng dụng rộng rãi trong thực tế. Hiện tại, trong đa số các ứng dụng thực tế, RSA vẫn là lựa chọn tốt nhất do RSA đã chứng minh được tính ổn định trong một khoảng thời gian khá dài.

---

<sup>13</sup> Thử nghiệm trên môi trường Windows XP, bộ xử lý Pentium 4 3.00 GHz, bộ nhớ 512 MB.

## 2.4 Kết luận

Như đã trình bày ở phần trên, hai nhóm thuật toán quan trọng nhất được sử dụng trong chữ ký số đó là thuật toán hàm băm mật mã và thuật toán chữ ký số.

Có rất nhiều hàm băm mật mã được đề xuất nhưng chỉ có hai thuật toán băm được sử dụng phổ biến trong chữ ký số từ thập niên 1990 đến nay là MD5 và SHA-1 (thuộc chuẩn SHS). Thực tế cho thấy, thuật toán băm MD5 không còn an toàn và đã được thay thế bởi SHA-1. Kết quả thực nghiệm cũng cho thấy thấy tốc độ của SHA-1 chỉ chậm hơn MD5 trung bình 75% khi kích thước đầu vào tăng dần nhưng lại mang đến độ an toàn cao hơn. Ngoài ra, các biến thể của SHA-1 là SHA-2 (gồm 4 thuật toán băm SHA-224, SHA-256, SHA-384 và SHA-512) cũng đang được sử dụng để mang lại mức độ an toàn cao hơn rất nhiều và thực nghiệm cũng cho thấy tốc độ của SHA-224/256 chỉ chậm hơn SHA-1 trung bình 38% còn tốc độ của SHA-384/512 chậm SHA-1 trung bình 142% nhưng lại mang đến độ bảo mật cao hơn tương ứng là 1.4, 1.6, 2.4, 3.2 lần.

Ngày 4/4/2006, trong báo cáo về tình trạng các chuẩn mã hóa NIST của Bill Burr (giám đốc nhóm công nghệ bảo mật của NIST) [14] đã có những phân tích và khuyến cáo trong việc sử dụng các hàm băm SHA như sau:

**Bảng 2.9. Khuyến cáo trong sử dụng các thuật toán hàm băm mật mã SHA**

| Thuật toán | Sử dụng thông thường |          | Sử dụng tối mật |           |
|------------|----------------------|----------|-----------------|-----------|
|            | Đến 2010             | Sau 2010 | Tối mật         | Tuyệt mật |
| SHA-1      | ✓                    |          |                 |           |
| SHA-224    | ✓                    | ✓        |                 |           |
| SHA-256    | ✓                    | ✓        | ✓               |           |
| SHA-384    | ✓                    | ✓        | ✓               | ✓         |
| SHA-512    | ✓                    | ✓        |                 |           |

Như vậy, thuật toán băm SHA-1 chỉ nên được sử dụng đến 2010 còn sau đó phải chuyển sang sử dụng các hàm băm SHA-2. Đối với các sử dụng tối mật thì chỉ có SHA-256 và SHA-384 được sử dụng, đặc biệt là SHA-384 được sử dụng trong các ứng dụng tuyệt mật.

Đối với chữ ký số, ba thuật toán chữ ký số phổ biến hiện nay RSA, DSA và ECDSA. Thuật toán chữ ký số RSA sử dụng thuật toán khóa công khai RSA, tận dụng tính khóa giải của bài toán phân tích ra thừa số nguyên tố của một số lớn, đã được đưa ra từ cuối thập niên 70, thuật toán chữ ký số DSA sử dụng thuật toán khóa công khai dựa trên tính khó giải của bài toán logarit rời rạc trên trường hữu hạn và thuật toán chữ ký số ECDSA sử dụng thuật toán khóa công khai dựa trên bài toán logarit rời rạc trên trường số của đường cong elliptic, mới được đưa ra từ năm 1985. Thực nghiệm cho thấy DSA tạo khóa chậm hơn RSA rất nhiều nhưng lại thực hiện tạo chữ ký và xác nhận chữ ký nhanh hơn. Tuy nhiên, độ chậm này không đáng kể và khi kích thước khóa tăng lên thì các tỷ lệ này giảm dần theo chiều hướng có lợi cho RSA. Đối với ECDSA, thuật toán khóa công khai được sử dụng là ECC cho mức độ an toàn tương đương RSA với kích thước khóa nhỏ hơn. Do đó tốc độ xử lý của ECC nhanh hơn RSA. Thực nghiệm cũng cho thấy ECDSA thực hiện tạo chữ ký và xác nhận chữ ký nhanh hơn RSA nhưng tạo khóa chậm hơn RSA. Vì vậy, ECDSA phù hợp với các thiết bị có tốc độ xử lý và kích thước bộ nhớ hạn chế. Tuy nhiên, ECDSA vẫn có một số hạn chế nhất định. Hạn chế lớn nhất hiện nay là việc chọn sử dụng các tham số đường cong và điểm quy ước chung như thế nào để thật sự đạt được độ an toàn cần thiết. Hầu hết các đường cong được đưa ra đều thất bại khi áp dụng vào thực tiễn. Do đó hiện nay số lượng đường cong thật sự được sử dụng không được phong phú. NIST đề xuất một số đường cong elliptic đã được kiểm định là an toàn để đưa vào sử dụng thực tế trong tài liệu FIPS 186-2 [67]. Ngoài ra, đối với các tham số mang giá trị nhỏ, độ an toàn của ECC không bằng RSA (khi  $e = 3$ ).

ECC vẫn còn non trẻ và cần được kiểm định trong tương lai để có thể ứng dụng rộng rãi trong thực tế. Hiện tại, trong đa số các ứng dụng thực tế, RSA vẫn là lựa chọn tốt nhất do RSA dễ hiểu, dễ triển khai và đã được chứng minh được tính ổn định trong một khoảng thời gian khá dài. Hơn nữa, RSA đã hoàn toàn miễn phí kể từ năm 2000.

Do đó, trong phạm vi đề tài, thuật toán mã hóa khóa công khai RSA sử dụng trong chữ ký số được chọn để nghiên cứu, phân tích và cải tiến. Nội dung vấn đề này sẽ lần lượt được trình bày ở Chương 5 và Chương 6.

## Chương 3

# Tổ chức chứng nhận khóa công khai

- Nội dung của chương này trình bày tổng quan về tổ chức chứng nhận khóa công khai (CA) và các chứng nhận khóa công khai, đồng thời giới thiệu các chức năng quan trọng của tổ chức này.

### 3.1 Giới thiệu

Như đã trình bày ở Chương 2, nếu hai người muốn trao đổi thông điệp mã hóa, mỗi người phải được trang bị những công cụ để giải mã thông điệp nhận được và mã hóa thông điệp gửi đi, việc này phụ thuộc vào kỹ thuật mã hóa mà họ sử dụng. Nếu sử dụng hệ thống mã hóa khóa bí mật, việc trao đổi chỉ có thể thực hiện giữa một nhóm ít người vì chỉ cần một người lộ khóa sẽ ảnh hưởng đến tất cả những người liên quan.

Hệ thống mã hóa khóa công khai ra đời giải quyết được vấn đề gút mắc trong trao đổi khóa của mã hóa đối xứng, đem đến một phương pháp mã hóa mới an toàn hơn. Nếu A muốn gửi thông tin bí mật cho B thì A chỉ cần biết khóa công khai của B. Tuy nhiên, bất kỳ người M nào cũng có khả năng đưa cho A một khóa công khai khác và giả mạo đó là khóa của B. Bằng cách này M có thể đọc được mọi thông tin mà A gửi cho B. Vấn đề đặt ra là làm sao A biết khóa công khai nhận được chính là khóa của B, người mà mình muốn truyền thông tin bí mật.

Vấn đề này được giải quyết bằng cách có một tổ chức thứ ba được mọi người tin cậy đứng ra chứng nhận khóa công khai cho mỗi người bằng các phát hành một giấy chứng nhận khóa công khai (Public Key Certificate), một loại văn bản điện tử kết hợp một chữ ký số để ràng buộc khóa công khai và danh tính người chủ khóa. Tổ chức này được gọi là nhà cung cấp chứng nhận số, gọi tắt là CA (Certificate Authority). Theo tổ chức IETF<sup>14</sup>, CA là một tổ chức được tín nhiệm bởi một hoặc nhiều người để tạo và phát hành các chứng nhận khóa công khai [65].

---

<sup>14</sup> IETF (Internet Engineering Task Force) là tổ chức gồm các con người có trách nhiệm trong việc tạo lập, chuẩn hóa và phát triển các giao thức/chức năng nhằm làm cho mạng Internet trở nên hữu dụng.

## 3.2 Chứng nhận số

### 3.2.1 Các loại chứng nhận

Để khóa công khai của mình được chứng nhận, bên đối tác phải tạo ra một cặp khóa bất đối xứng và gửi cặp khóa này cho tổ chức CA. Bên đối tác phải gửi kèm các thông tin về bản thân như tên hoặc địa chỉ. Khi tổ chức CA đã kiểm tra tính xác thực các thông tin của bên đối tác, CA sẽ phát hành một giấy chứng nhận khóa công khai cho bên đối tác. Giấy chứng nhận là một tập tin nhị phân có thể dễ dàng chuyển đổi qua mạng máy tính.

Tổ chức CA áp dụng chữ ký điện tử của mình cho giấy chứng nhận khóa công khai mà CA đó phát hành. Một tổ chức CA chứng nhận khóa công khai bằng cách ký nhận chúng. Nếu phía đối tác bên kia tin tưởng vào tổ chức CA thì họ có thể tin vào chữ ký của CA đó.

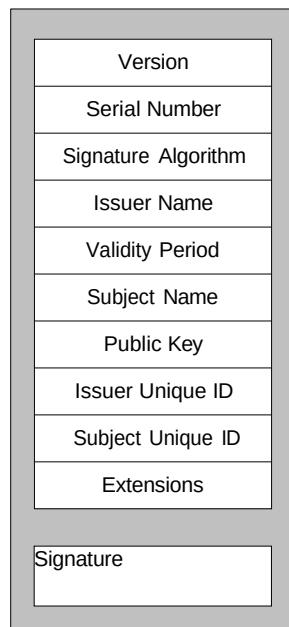
Một số loại giấy chứng nhận khóa công khai có thể được phát hành như chứng nhận X.509, chứng nhận chất lượng và chứng nhận thuộc tính.

#### 3.2.1.1 Chứng nhận X.509

Chứng nhận X.509 là chứng nhận khóa công khai phổ biến nhất. Hiệp hội Viễn thông quốc tế (International Telecommunications Union – ITU) đã chỉ định chuẩn X.509 vào năm 1988 [5]. Đây là định dạng phiên bản 1 của chuẩn X.509. Vào năm 1993, phiên bản 2 của chuẩn X.509 được phát hành với 2 trường tên nhận dạng duy nhất được bổ sung. Phiên bản 3 của chuẩn X.509 được bổ sung thêm trường mở rộng đã phát hành vào năm 1997.

Một chứng nhận khóa công khai kết buộc một khóa công khai với sự nhận diện của một người (hoặc một thiết bị). Khóa công khai và tên thực thể sở hữu khóa này là hai mục quan trọng trong một chứng nhận. Hầu hết các trường khác trong chứng nhận

X.509 phiên bản 3 đều đã được chứng tỏ là có ích. Sau đây là thông tin về các trường trong chứng nhận X.509 phiên bản 3 [5]:



**Hình 3.1. Phiên bản 3 của chứng nhận X.509**

- **Version:** Chỉ định phiên bản của chứng nhận X.509.
- **Serial Number:** Số loạt phát hành được gán bởi CA. Mỗi CA nên gán một mã số loạt duy nhất cho mỗi giấy chứng nhận mà nó phát hành.
- **Signature Algorithm:** Thuật toán chữ ký chỉ rõ thuật toán mã hóa được CA sử dụng để ký giấy chứng nhận. Trong chứng nhận X.509 thường là sự kết hợp giữa thuật toán băm (chẳng hạn như MD5 hoặc SHA-1) và thuật toán khóa công khai (chẳng hạn như RSA).
- **Issuer Name:** Tên tổ chức CA phát hành giấy chứng nhận, đây là một tên phân biệt theo chuẩn X.500 (xem Phụ lục A). Hai CA không được sử dụng cùng một tên phát hành.
- **Validity Period:** Trường này bao gồm 2 giá trị chỉ định khoảng thời gian mà giấy chứng nhận có hiệu lực. Hai phần của trường này là not-before và not-after. Not-before chỉ định thời gian mà chứng nhận này bắt đầu có hiệu lực, Not-after chỉ định thời gian mà chứng nhận hết hiệu lực. Các giá trị thời gian này được đo theo chuẩn thời gian Quốc tế, chính xác đến từng giây.
- **Subject Name:** là một X.500 DN, xác định đối tượng sở hữu giấy chứng nhận mà cũng là sở hữu của khóa công khai. Một CA không thể phát hành 2 giấy chứng nhận có cùng một Subject Name.
- **Public key:** Xác định thuật toán của khóa công khai (như RSA) và chứa khóa công khai được định dạng tùy vào kiểu của nó.

- **Issuer Unique ID** và **Subject Unique ID**: Hai trường này được giới thiệu trong X.509 phiên bản 2, được dùng để xác định hai tổ chức CA hoặc hai chủ thể khi chúng có cùng DN. RFC 2459 đề nghị không nên sử dụng 2 trường này.
- **Extensions**: Chứa các thông tin bổ sung cần thiết mà người thao tác CA muốn đặt vào chứng nhận. Trường này được giới thiệu trong X.509 phiên bản 3.
- **Signature**: Đây là chữ ký điện tử được tổ chức CA áp dụng. Tổ chức CA sử dụng khóa bí mật có kiểu quy định trong trường thuật toán chữ ký. Chữ ký bao gồm tất cả các phần khác trong giấy chứng nhận. Do đó, tổ chức CA chứng nhận cho tất cả các thông tin khác trong giấy chứng nhận chứ không chỉ cho tên chủ thể và khóa công khai.

Những phần mở rộng của tên tập tin phổ biến cho chứng nhận X.509 bao gồm:

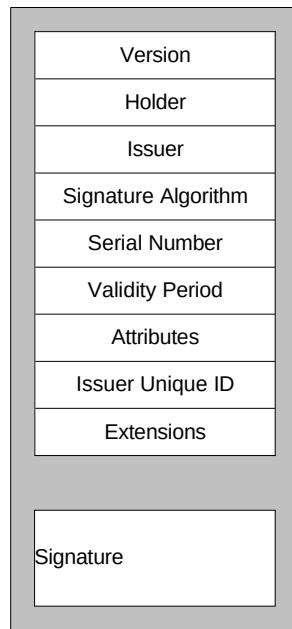
- **.cer**: chứng nhận được mã hóa theo luật mã hóa tiêu chuẩn (Canonical Encoding Rules – CER).
- **.der**: chứng nhận được mã hóa theo luật mã hóa phân biệt (Distinguished Encoding Rules – DER).
- **.pem** (Privacy-Enhanced Electronic Mail): định dạng mã hóa được sử dụng để lưu trữ các chứng nhận và khóa. Một tập tin được định dạng với chuẩn này có thể chứa các khóa bí mật (RSA và DSA), khóa công khai (RSA và DSA) và các chứng nhận X509. Định dạng này lưu trữ dữ liệu ở định dạng DER được mã hóa cơ sở 64, nằm giữa "-----BEGIN CERTIFICATE-----" và "-----END CERTIFICATE-----", phù hợp cho việc trao đổi ở dạng văn bản giữa các hệ thống.
- **.p7b, p7c**: PKCS #7 là một định dạng mã hóa cho việc lưu trữ một chứng nhận số và chuỗi chứng nhận của nó dưới dạng các ký tự ASCII. Định dạng này được sử dụng bởi CA để trả về các chứng nhận được phát hành cùng với chuỗi chứng nhận. Định dạng này có thể được sử dụng như đầu vào cho yêu cầu gia hạn chứng nhận đến một CA.
- **.pfx, .p12**: PKCS #12 là một định dạng mã hóa cho việc lưu trữ một chứng nhận số và kết hợp với khóa bí mật dưới dạng các ký tự ASCII. Định dạng này luôn luôn được trả về bởi CA khi CA phát sinh các khóa và phát hành chứng nhận đồng thời.

### **3.2.1.2 Chứng nhận chất lượng**

Đặc điểm chính của các giấy chứng nhận chất lượng là chúng quan tâm quan tới đối tượng mà chúng được phát hành đến. Thực thể cuối sở hữu giấy chứng nhận X.509 hoặc RFC 2459 có thể là một người hoặc một máy. Tuy nhiên, các giấy chứng nhận chất lượng chỉ có thể được phát hành cho con người.

Giấy chứng nhận chất lượng RFC 3039 cung cấp các yêu cầu chi tiết dựa trên nội dung của nhiều trường trong chứng nhận X.509. Các trường tên nhà xuất bản, tên chủ thể, phần mở rộng đều được cung cấp các yêu cầu nội dung cụ thể. Tên nhà xuất bản của giấy chứng nhận chất lượng phải xác định được tổ chức chịu trách nhiệm phát hành giấy chứng nhận đó. Tên chủ thể của giấy chứng nhận chất lượng phải xác định một con người thật.

### **3.2.1.3 Chứng nhận thuộc tính**



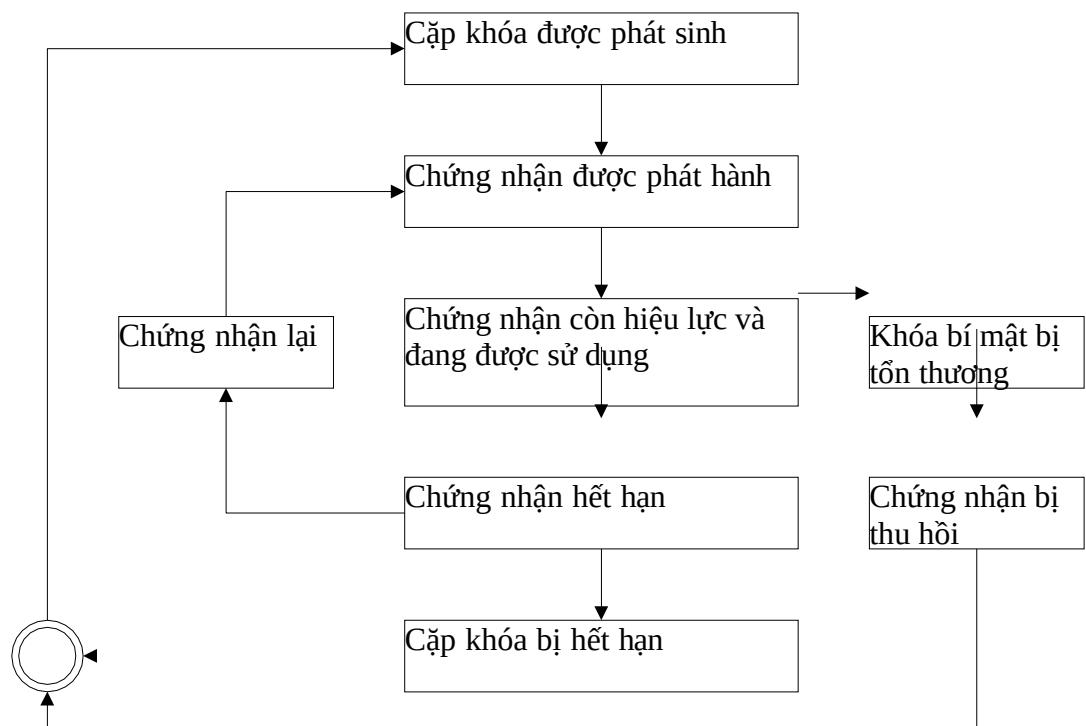
**Hình 3.2. Phiên bản 2 của cấu trúc chứng nhận thuộc tính**

Các giấy chứng nhận thuộc tính (Attribute Certificate – AC [5]) là các giấy chứng nhận điện tử không chứa khóa công khai. Thay vì thao tác chứng nhận khóa công khai, AC chỉ thao tác chứng nhận một tập hợp các thuộc tính. Các thuộc tính trong một AC được dùng để chuyển các thông tin giấy phép liên quan đến người giữ giấy chứng nhận. Các chứng nhận thuộc tính phân quyền cho người giữ chúng.

Hệ thống phát hành, sử dụng và hủy AC là hạ tầng quản lý đặc quyền (Privilege Management Infrastructure – PMI). Trong PMI, tổ chức chứng nhận thuộc tính (Attribute Authority – AA) phát hành AC. Một AA có thể không giống như một CA.

Động cơ chính cho việc sử dụng AC là để cấp phép. Vì một người dùng có thể chỉ giữ một vai trò nào đó trong tổ chức trong một thời gian ngắn, nên khác với giấy chứng nhận khóa công khai, AC chỉ có giá trị trong một vài ngày hoặc ngắn hơn.

### 3.2.2 Chu kỳ sống của chứng nhận số



**Hình 3.3. Chu kỳ sống của chứng nhận**

Trước khi phát hành chứng nhận, cặp khóa bí mật/ công khai sẽ được phát sinh. Trong khi chứng nhận có hiệu lực và được sử dụng, chứng nhận có thể hết hạn hoặc khóa bí mật của người sử dụng bị tổn thương (bị mất hoặc lộ khóa). Trong trường hợp chứng nhận hết hạn, cặp khóa cũng sẽ không còn hiệu lực hoặc người sử dụng có thể yêu cầu gia hạn chứng nhận cho họ. Trong trường hợp khóa bí mật bị tổn thương, chứng nhận sẽ được thu hồi để phát hành chứng nhận cho cặp khóa khác.

### 3.3 Các chức năng chính

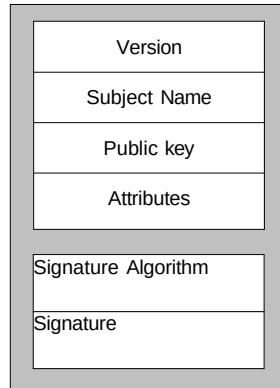
#### 3.3.1 Khởi tạo

Trước khi yêu cầu một chứng nhận, đối tác phải tìm hiểu về CA mà mình muốn tham gia. Đối tác phải có địa chỉ của tổ chức CA và kho lưu trữ nếu chúng tồn tại. Đối tác cũng cần phải có giấy chứng nhận của tổ chức CA và cuối cùng cần phải có cách tạo ra cặp khóa bất đối xứng và lựa chọn các thuộc tính cho tên phân biệt (DN).

#### 3.3.2 Yêu cầu chứng nhận

Đối tác có thể yêu cầu một chứng nhận từ CA thông qua nhiều kỹ thuật. Trong trường hợp phát sinh lại, đối tác không cần yêu cầu, tổ chức CA sẽ tạo ra một giấy chứng nhận thay cho đối tác. Kỹ thuật này yêu cầu tổ chức CA cũng phải phát sinh cặp khóa bất đối xứng để có được khóa công khai được kèm theo trong chứng nhận. Hầu hết các CA sử dụng một trong hai phương thức tiêu chuẩn của yêu cầu chứng nhận: PKCS #10 và CRMF [5].

##### 3.3.2.1 Yêu cầu chứng nhận theo chuẩn PKCS #10

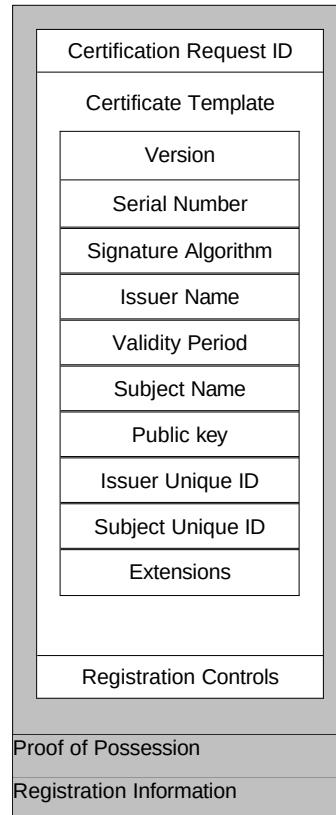


**Hình 3.4. Mẫu yêu cầu chứng nhận theo chuẩn PKCS #10**

- **Version:** phiên bản của định dạng yêu cầu chứng nhận.
- **Subject Name:** là một X.500 DN, xác định thực thể cuối yêu cầu giấy chứng nhận, người sở hữu khóa công khai.
- **Public Key:** chỉ ra thuật toán của khóa công khai, chứa khóa công khai có định dạng tùy thuộc vào loại của nó.
- **Attributes:** bao gồm các thông tin bổ sung dùng để xác định thực thể cuối.

- **Signature Algorithm:** chỉ ra thuật toán mã hóa được dùng bởi thực thể cuối để ký yêu cầu chứng nhận.
- **Signature:** chữ ký điện tử được áp dụng bởi thực thể cuối yêu cầu chứng nhận.

### 3.3.2.2 Yêu cầu chứng nhận theo chuẩn của CRMF



**Hình 3.5. Định dạng thông điệp yêu cầu chứng nhận theo RFC 2511**

- **Request ID:** số được sử dụng bởi đối tác và tổ chức CA để liên kết yêu cầu với trả lời chứa chứng nhận được yêu cầu.
- **Certificate Template:** trong yêu cầu PKCS #10, đối tác chỉ có thể chỉ định tên và thông tin khóa công khai bao gồm trong giấy chứng nhận. Trong CRMF, đối tác có thể bao gồm bất cứ trường nào của chứng nhận X.509 như là một mẫu chứng nhận trong yêu cầu của họ.
- **Controls:** cung cấp cách thức mà đối tác gửi các chi tiết giám sát liên quan tới yêu cầu của họ tới tổ chức CA. Trường này có thể được dùng tương tự như trường thuộc tính trong PKCS #10.

- **Proof of Possession:** CRMF hỗ trợ 4 phương thức để đối tác chứng minh rằng họ sở hữu khóa bí mật tương ứng với khóa công khai trong yêu cầu. Mỗi phương thức được sử dụng tùy thuộc vào mục đích sử dụng khóa.
- **Registration Information:** là trường tùy chọn chứa các dữ liệu liên quan đến yêu cầu chứng nhận được định dạng trước hoặc được thay thế.

### 3.3.3 Tạo lại chứng nhận

Đối tác có thể muốn tạo mới lại chứng nhận của mình vì nhiều lý do: giấy chứng nhận hết hạn, thêm thông tin mới vào chứng nhận, xác nhận lại khóa công khai hiện có, hoặc xác nhận khóa mới. Khi tổ chức CA đáp ứng yêu cầu tạo mới lại này, nó sẽ phát hành cho đối tác một giấy chứng nhận mới và có thể xuất bản giấy chứng nhận mới này vào kho lưu trữ.

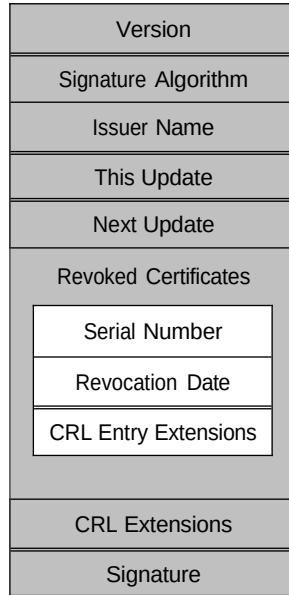
Yêu cầu tạo lại thì đơn giản hơn rất nhiều so với yêu cầu chứng nhận nguyên thủy. Khi CA nhận yêu cầu chứng nhận, nó phải xác minh sự tồn tại của đối tác. Nhưng khi đối tác gửi yêu cầu tạo lại, họ có thể bao gồm giấy chứng nhận hiện có và chữ ký sử dụng khóa bí mật tương ứng với chứng nhận đó. Điều đó có thể xem như sự chứng nhận tồn tại của đối tác. Do đó, việc tạo lại chứng nhận thì dễ cho CA đáp ứng hơn.

### 3.3.4 Hủy bỏ chứng nhận

Tất cả chứng nhận đều có thời hạn sử dụng của nó và chúng cuối cùng sẽ bị hết hạn. Tuy nhiên, cần phải hủy bỏ một chứng nhận trước khi nó bị hết hạn. Lý do chung nhất để hủy một chứng nhận là do sự nhận diện được xác nhận bởi CA đã thay đổi.

Danh sách hủy bỏ chứng nhận (Certificate Revocation List – CRL) [5] là cách đầu tiên và thông dụng nhất để phổ biến thông tin hủy bỏ. CRL chứa thông tin thời gian nhằm xác định thời điểm tổ chức CA phát hành nó. CA ký CRL với cùng khóa bí mật được dùng để ký các chứng nhận. Các CRL thường được chứa trong cùng kho với các chứng nhận nhằm dễ dàng cho việc rút trích.

Các CA phát hành các CRL theo định kì, thường là hàng giờ hoặc hàng ngày.



**Hình 3.6. Phiên bản 2 của định dạng danh sách chứng nhận bị hủy**

- **Version:** phiên bản định dạng CRL
  - **Signature Algorithm:** xác định thuật toán mã hóa được dùng để ký CRL.
  - **Issuer Name:** một X.500 DN, xác định tên tổ chức ký CRL.
  - **This-Update:** thời điểm CRL được tạo ra.
  - **Next-Update:** thời điểm CA tạo ra CRL kế tiếp.
  - **Revoked Certificates:** danh sách các chứng nhận bị hủy bỏ. Mỗi chứng nhận bị hủy có một mục CRL, chứa các thông tin sau:
    - **Serial Number:** mã số chứng nhận
    - **Revocation Date:** ngày hủy bỏ
    - **CRL Entry Extension:** các thông tin bổ sung
  - **CRL Extensions:** các thông tin bổ sung hỗ trợ cho việc dùng và quản lý các CRL.
  - **Signature:** chữ ký của tổ chức phát hành CRL.

### 3.3.5 Lưu trữ và phục hồi khóa

Lưu trữ khóa là một dịch vụ được cung cấp bởi nhiều tổ chức CA. Thông qua việc lưu trữ khóa mã hóa bí mật, khách hàng có thể tránh được trường hợp không giải mã được dữ liệu khi bị mất khóa. Để lưu trữ khóa, khách hàng phải gửi khóa bí mật tới nơi lưu trữ. Bởi vì các yêu cầu lưu trữ hay khôi phục khóa đều phải được xác minh

nên các người sử dụng không thể thao tác trực tiếp đến nơi lưu trữ mà phải thông qua CA phát hành chứng nhận đó.

Khả năng làm mất hoặc sai các khoá bí mật của người dùng là rất lớn, do đó ta cần phải có một cơ chế lưu trữ dự phòng và khôi phục khoá bí mật. Hãy tưởng tượng một người sử dụng mã hoá toàn bộ các văn bản mà họ tổn công sức tạo ra trong nhiều năm với một mã khóa duy nhất và sau đó đánh mất nó. Trên thực tế, việc khôi phục tài liệu là không thể nếu không có khoá bí mật.

### **3.4 Kết luận**

Như vậy, tổ chức chứng nhận khóa công khai (CA) là một tổ chức thứ ba đáng tin cậy có nhiệm vụ phát hành, quản lý và hủy bỏ các chứng nhận nhằm giúp cho người sử dụng có thể giao tiếp với nhau bảo đảm an toàn, tin cậy. Khi người sử dụng tin tưởng vào một CA và có thể kiểm tra chữ ký số của CA đó thì họ cũng có thể tin tưởng vào khóa công khai và thực thể được ghi trong chứng nhận.

Nếu CA bị xâm nhập thì an toàn của hệ thống sẽ bị phá vỡ. Nếu kẻ tấn công có thể can thiệp vào hệ thống để tạo ra một chứng nhận giả trong đó gắn khóa công cộng của kẻ tấn công với định danh của người dùng khác thì mọi giao dịch của người khác với người này có thể bị kẻ tấn công can thiệp. Vì vậy, việc đảm bảo độ chính xác của thông tin trong chứng nhận là rất quan trọng nhưng lại khó thực hiện, đặc biệt khi phần lớn các giao dịch sẽ được thông qua môi trường điện tử.

Hơn nữa, khi được ứng dụng trên quy mô lớn, một CA sẽ không thể nào đảm nhiệm hết mọi công việc. Lúc này CA cần chia sẻ công việc với các CA hoặc các tổ chức đặc thù khác (như tổ chức đăng ký chứng nhận), từ đó hình thành kiến trúc hạ tầng khóa công khai (Public Key Infrastructure – PKI) mà trong đó CA là thành phần trung tâm. Vấn đề này sẽ được trình bày chi tiết ở Chương 4.

## Chương 4

# Hạ tầng khóa công khai

☐ Nội dung của chương này trình bày khái niệm, vai trò và chức năng của hạ tầng khóa công khai, đồng thời tập trung nghiên cứu và phân tích các kiến trúc hạ tầng khóa công khai hiện có, từ đó đánh giá và chọn lựa kiến trúc phù hợp có thể triển khai trong thực tế.

### 4.1 Giới thiệu

#### 4.1.1 Khái niệm

Trong mật mã học, hạ tầng khóa công khai (Public Key Infrastructure – PKI), là hệ thống vừa mang tính tiêu chuẩn, vừa mang tính công nghệ cho phép người sử dụng trong một mạng công cộng không bảo mật (như Internet), có thể trao đổi thông tin một cách an toàn thông qua việc sử dụng một cặp khóa bí mật và công khai được chứng nhận bởi một nhà cung cấp chứng nhận số CA được tín nhiệm.

Theo X.509 PKIX<sup>15</sup> định nghĩa, một PKI là một tập các phần cứng, phần mềm, con người và các thủ tục cần thiết để tạo, lưu trữ, phân phối, thu hồi khóa/ chứng nhận dựa trên mã hóa khóa công khai. Nhu cầu sử dụng hạ tầng này có từ cuối những năm 1990, khi mà các tổ chức công nghiệp và các chính phủ xây dựng các tiêu chuẩn chung dựa trên phương pháp mã hoá để hỗ trợ một hạ tầng bảo mật trên mạng Internet. Mục tiêu được đặt ra tại thời điểm đó là xây dựng một bộ tiêu chuẩn bảo mật tổng hợp cùng các công cụ và lý thuyết cho phép người sử dụng cũng như các tổ chức (doanh nghiệp hoặc phi lợi nhuận) có thể tạo lập, lưu trữ và trao đổi các thông tin một cách an toàn trong phạm vi cá nhân và công cộng.

---

<sup>15</sup> PKIX là viết tắt của một trong các nhóm đang làm việc trong lĩnh vực bảo mật của IETF là Hạ tầng khóa công khai PKI (Public-Key Infrastructure), X.509 và được thành lập vào mùa thu năm 1995.

Một lựa chọn khác để phân phối chứng nhận khóa công khai giữa một hệ thống mà không cần đến tổ chức thứ ba đó là hướng tiếp cận trong hệ thống PGP (Pretty Good Privacy) của NAI (Network Associates, Inc). Mỗi thành viên tham gia vào hệ thống này có thể đóng vai trò của CA để tạo và ký vào chứng nhận khóa công khai của một thành viên khác mà họ biết, do đó không cần phải phát triển hạ tầng trung tâm.

Mô hình này chỉ hoạt động rất tốt cho một nhóm nhỏ gồm những người có những mối quan hệ trước đó với người khác, nhưng nó không mở rộng tốt cho những nhóm lớn hoặc ở những môi trường cần đòi hỏi sự quản lý chặt chẽ (chẳng hạn mức độ xác thực được đòi hỏi trước khi chứng nhận được phát hành). Vì vậy, đề tài chỉ tập trung nghiên cứu các kiến trúc PKI sử dụng CA trong việc quản lý chứng nhận.

#### **4.1.2 Vai trò và chức năng**

Chức năng chính của một PKI cho phép những người tham gia xác thực lẫn nhau và sử dụng thông tin từ các chứng nhận khóa công khai để mã hóa và giải mã thông tin. Đặc biệt, nó cho phép các giao dịch điện tử được diễn ra đảm bảo tính cẩn mật, tính toàn vẹn, tính xác thực và tính không thể phủ nhận mà không cần phải trao đổi các thông tin mật từ trước [60, tr.9-10]:

- **Tính cẩn mật (Confidentiality)** nghĩa là bảo đảm tính bí mật của dữ liệu. Tính bí mật này được cung cấp bởi các cơ chế mã hóa mật mã học, bằng cách sử dụng cả mã hóa khoá công khai lẫn mã hóa khóa bí mật. Do mã hóa khóa công khai không hiệu quả bằng mã hóa bí mật trong việc mã hóa dữ liệu lớn, nó thường được sử dụng để mã hóa những đối tượng dữ liệu tương đối nhỏ như các khóa bí mật được sử dụng trong các hệ thống mã hóa bất đối xứng.
- **Tính toàn vẹn (Integrity)** nghĩa là đảm bảo dữ liệu không thể bị mất mát hoặc chỉnh sửa và các giao tác không thể bị thay đổi. Tính toàn vẹn có thể được cung cấp bên trong PKI bằng cách sử dụng cả mã hóa công khai và mã hóa bí mật. Mã hóa khóa công khai đặc biệt được sử dụng chung với một thuật toán băm như SHA-1 hay MD5 để cung cấp tính toàn vẹn. Một PKI được thiết kế tốt sẽ sử dụng các giao thức đòi hỏi sử dụng các thuật toán đó để cung cấp cơ chế toàn vẹn hiệu quả.

- **Tính xác thực (Authentication)** nghĩa là danh tính của thực thể được xác minh. Tính xác thực trong môi trường thương mại điện tử được thực hiện rất tốt bằng các hệ thống mã hóa khóa công khai, dựa trên mỗi quan hệ toán học giữa khóa công khai và khóa bí mật. Thông điệp được ký bởi một thực thể có thể được kiểm tra bởi bất kỳ thực thể nào quan tâm. Các thực thể này có thể an tâm rằng chỉ có chủ của khóa bí mật mới có thể tạo ra thông điệp này, bởi vì chỉ có người đó mới có khóa bí mật.
- **Tính không thể chối từ (Non-Repudiation)** nghĩa là đảm bảo dữ liệu không thể bị không thừa nhận hoặc giao tác bị từ chối. Đây là một dịch vụ bảo mật then chốt của bất kỳ ứng dụng thương mại nào trong đó việc trao đổi giá trị hay các quy định pháp luật được thỏa hiệp. Tính không thể chối từ được cung cấp thông qua mã hóa khóa công khai bằng chữ ký số. Khi dữ liệu được ký theo cách mật mã học sử dụng khóa bí mật của cặp khóa, bất kỳ ai có thể truy cập khóa công khai của cặp khóa này đều có thể xác định rằng chỉ có chủ của cặp khóa mới có thể ký vào dữ liệu.

PKI không chỉ phục vụ cho các chức năng thương mại nói riêng, nó còn cung cấp một nền tảng cho các dịch vụ bảo mật khác. PKI là nền tảng cho các ứng dụng và các thành phần bảo mật mạng khác được xây dựng trên nó. Các hệ thống thường xuyên đòi hỏi các cơ chế bảo mật dựa trên PKI có thể kể ra như thư điện tử, các ứng dụng thẻ thông minh, giao dịch điện tử (ví dụ thẻ ghi nợ và tín dụng), ngân hàng điện tử, và các hệ thống bưu điện điện tử.

Mục tiêu chính của PKI là cung cấp và xác thực mối liên hệ giữa khóa và danh tính người dùng. Nhờ vậy người dùng có thể sử dụng trong một số ứng dụng như:

- Mã hoá email hoặc xác thực người gửi email (OpenPGP hay S/MIME).
- Ký và xác thực văn bản.
- Xác thực người dùng ứng dụng (đăng nhập bằng thẻ thông minh – smartcard, nhận thực người dùng trong SSL, bầu cử).
- Các giao thức truyền thông an toàn dùng kỹ thuật bootstrapping (IKE, SSL): trao đổi khóa bằng khóa bất đối xứng, còn mã hóa bằng khóa đối xứng.

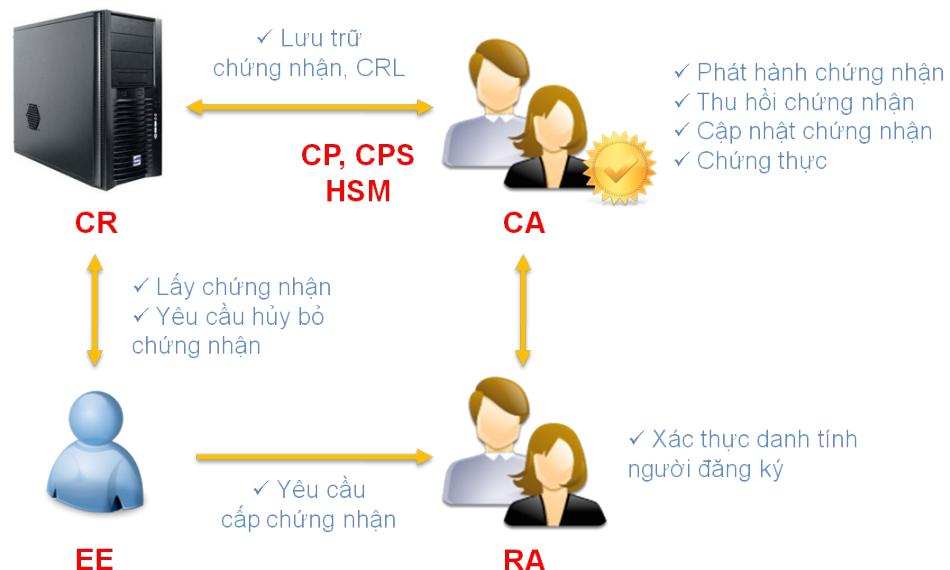
Ngoài ra, việc sử dụng PKI và mã hóa công khai trong thương mại điện tử giúp các tổ chức giảm chi phí xử lý giao tác, giảm rủi ro và giảm độ phức tạp của các hệ thống bảo mật với các phương pháp đối xứng.

#### 4.1.3 Các thành phần của một hạ tầng khóa công khai

PKI là một cơ cấu tổ chức gồm con người, tiến trình, chính sách, giao thức, phần cứng và phần mềm dùng để phát sinh, quản lý, lưu trữ, triển khai và thu hồi các chứng nhận khóa công khai [60, tr.10-15].

Về cơ bản, PKI gồm các thành phần như sau:

- Thực thể cuối (End Entity – EE).
- Tổ chức chứng nhận (Certificate Authority – CA).
- Chính sách chứng nhận (Certificate Policy – CP).
- Tuyên bố trong sử dụng chứng nhận (Certificate Practices Statement – CPS).
- Các module bảo mật phần cứng (Hardware Security Module – HSM).
- Chứng nhận khóa công khai (Public Key Certificate).
- Tổ chức đăng ký chứng nhận (Registration Authority – RA).
- Kho lưu trữ chứng nhận (Certificate Repository – CR).



Hình 4.1. Các thành phần của một hạ tầng khóa công khai

#### **4.1.3.1 Thực thể cuối**

Thực thể cuối không chỉ là người sử dụng mà còn bao gồm những thứ vô tri vô giác như máy tính, những đối tượng cần chứng nhận số để nhận biết chúng vì một số lý do nào đó. Thực thể cuối thông thường phải có khả năng phát sinh cặp khóa công khai/ bí mật và một số phương tiện cho việc lưu trữ và sử dụng khóa bí mật một cách an toàn. Theo định nghĩa này, một thực thể cuối không phải là một CA.

#### **4.1.3.2 Tổ chức chứng nhận**

Tổ chức chứng nhận (CA) là một thực thể quan trọng duy nhất trong PKI và được người sử dụng tín nhiệm. Tổ chức này có nhiệm vụ phát hành, quản lý và hủy bỏ các chứng nhận. Tổ chức này gồm tập hợp các con người và các hệ thống máy tính có độ an toàn cao (ví dụ sử dụng tường lửa trong hệ thống mạng, ...) để chống lại các nguy hiểm bên ngoài và khả năng quản lý tốt để chống lại các nguy hiểm bên trong. Chi tiết về tổ chức này đã được trình bày ở Chương 2.

CA làm việc trong ngữ cảnh của một chính sách làm việc tổng thể, gọi là một “chính sách chứng nhận” (Certificate Policy – CP) và các chức năng hoạt động theo một “tuyên bố trong sử dụng chứng nhận” (Certificate Practices Statement – CPS).

#### **4.1.3.3 Chính sách chứng nhận**

Chính sách chứng nhận (CP) cung cấp những nguyên tắc hướng dẫn tổng thể để một tổ chức có thể biết được ai được làm gì hay bằng cách nào vào được hệ thống và dữ liệu. Một CP cũng cần phải chỉ rõ cách thức để kiểm soát và quản lý. Hơn nữa, CP định rõ một tập những luật lệ cho thấy tính khả thi của một chứng nhận khóa công khai đối với một cộng đồng riêng biệt hoặc một lớp các ứng dụng với những yêu cầu an ninh chung. Ví dụ, một CP riêng biệt có thể cho biết tính khả thi của một loại chứng nhận khóa công khai đối với việc xác thực một giao dịch trao đổi điện tử trong kinh doanh hàng hóa hoặc giá trị tiền tệ, ...

#### **4.1.3.4 Tuyên bố trong sử dụng chứng nhận**

Tuyên bố trong sử dụng chứng nhận (CPS) rất giống với chính sách chứng nhận, ngoại trừ nó tập trung vào vấn đề bảo mật của CA trong suốt các hoạt động và quản

lý chứng nhận được phát hành bởi CA. CPS thể hiện chi tiết mọi quy trình bên trong chu kỳ số của chứng nhận khóa công khai bao gồm sự phát sinh, phát hành, quản lý, lưu trữ, triển khai và hủy bỏ. Có thể xem CPS như một thỏa thuận giữa người dùng chứng nhận và công ty chịu trách nhiệm cho việc phát hành CA. Không giống như chính sách chứng nhận, CPS luôn có sẵn ở công cộng để một người dùng nào đó có chứng nhận luôn có thể truy cập vào CPS. Trong mỗi chứng nhận mà CA phát hành, sẽ có một liên kết để chỉ ra vị trí nơi CPS được công bố.

#### **4.1.3.5 Các môđun bảo mật phần cứng**

Môđun bảo mật phần cứng (HSM) là một thành phần chính khác của một CA. Một CA phải mang đến sự tin nhiệm không chỉ đối với khách hàng của nó mà còn đối với những người tin cậy vào những chứng nhận đã được phát hành. Do sự tin nhiệm đó phải được xác nhận nhờ vào sự bảo mật và sự toàn vẹn của khóa bí mật được sử dụng để ký chứng nhận khóa công khai của người đăng ký, khóa bí mật đó cần phải được bảo vệ tốt nhất có trong các thiết bị máy tính chuyên dụng được biết đến như là HSM. Sự thực thi và sử dụng một HSM đủ tiêu chuẩn mang tính quyết định đối với bất kỳ CA và PKI mà nó hỗ trợ.

#### **4.1.3.6 Tổ chức đăng ký chứng nhận**

Tổ chức đăng ký chứng nhận (RA) là thành phần tùy chọn nhưng thường có trong PKI [5]. RA được thiết kế để chia sẻ bớt công việc mà CA thường phải đảm trách và không thể thực hiện bất kỳ một dịch vụ nào mà tổ chức CA của nó không thực hiện được. Quan trọng nhất là RA được ủy quyền và có quyền thực hiện các công việc mà CA cho phép vì lợi ích của CA. Một RA chỉ nên phục vụ cho một CA, trong khi đó một CA có thể được hỗ trợ bởi nhiều RA. Thông qua việc chia sẻ bớt nhiệm vụ cho các RA, một CA sẽ có thể đáp ứng nhanh các yêu cầu của thực thể cuối.

Mục đích chính của một RA là xác minh danh tính của thực thể cuối và quyết định xem thực thể này có được cấp chứng nhận khóa công khai hay không. RA phải tuân theo các chính sách và các thủ tục được định nghĩa trong CP và CPS. Chức năng đặc trưng của RA là thẩm tra yêu cầu cấp chứng nhận của thực thể cuối bằng cách kiểm tra tên, ngày hiệu lực, các ràng buộc thích hợp, khóa công khai, sự gia hạn chứng

nhận và các thông tin liên quan. RA còn có thể có trách nhiệm về việc thực hiện các kiểm tra chi tiết thực thể cuối đơn giản như việc chắc chắn rằng tên của thực thể cuối là duy nhất trong phạm vi của PKI.

#### **4.1.3.7 Chứng nhận khóa công khai**

Mục đích chính của CA là hỗ trợ phát sinh, quản lý, lưu trữ, triển khai và thu hồi chứng nhận khóa công khai. Một chứng nhận khóa công khai thể hiện hay chứng nhận sự ràng buộc của danh tính và khóa công khai của thực thể cuối. Nghĩa là nó chứa đủ thông tin cho những thực thể khác có thể xác nhận hoặc kiểm tra danh tính của chủ nhận chứng nhận đó. Định dạng được sử dụng rộng rãi nhất của chứng nhận số dựa trên chuẩn IETF X.509 (đã được trình bày ở Chương 2). Lưu ý rằng không có định nghĩa duy nhất nào của chứng nhận khóa công khai trong chuẩn IETF do mỗi tổ chức triển khai PKI sẽ có ý kiến riêng về dữ liệu mở rộng và đặc biệt nào mà một chứng nhận X.509 nên có. Các tổ chức nên đánh giá các nhu cầu công việc liên quan đến cấu trúc của chứng nhận khóa công khai mà họ muốn phát hành.

#### **4.1.3.8 Kho lưu trữ chứng nhận**

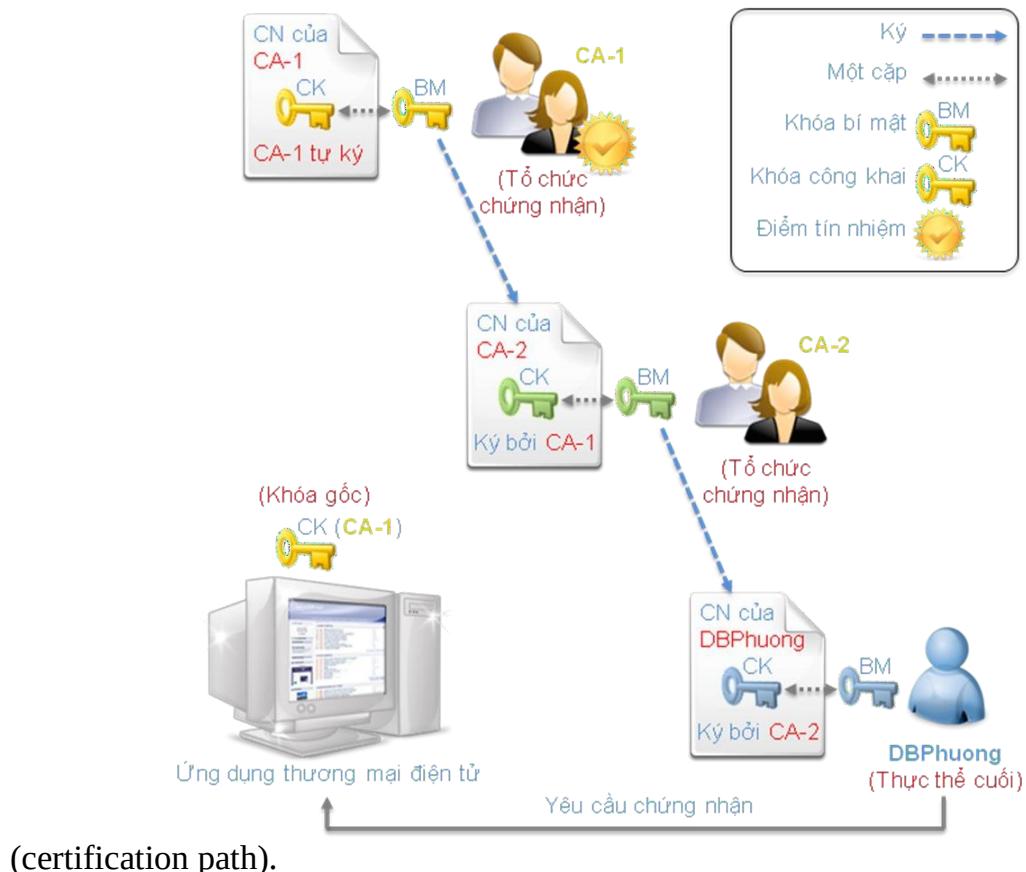
Kho lưu trữ chứng nhận (CR) là nơi chứa điện tử để chứa các thông tin và trạng thái của các chứng nhận được phát hành bởi CA và cũng có thể chứa cả danh sách các chứng nhận bị hủy (CRL). Kho chứa còn lưu trữ các chứng nhận chéo của CA này được phát hành bởi CA khác, chứng nhận chéo của CA khác phát hành bởi CA này. Kho chứa còn có nhiệm vụ chứa những biểu mẫu điện tử và các công cụ cho phép tải về, công bố CP và CPS, cập nhật thông tin, hỏi và đáp (Q & A), ...

Kho lưu trữ chứng nhận phải là một hệ thống tín nhiệm và an toàn. Trên lý thuyết có thể truy cập bằng cách sử dụng HTTP, FTP, thư mục X.500, LDAP (Lightweight Directory Access Protocol) hoặc thậm chí bằng thư điện tử nhưng hầu như được truy cập thông qua HTTP hoặc LDAP. LDAP là giao thức tìm thông tin trên máy chủ, nó là một giao thức client/ server dùng để truy cập dịch vụ thư mục X500. LDAP chạy trên TCP/IP hoặc những dịch vụ hướng kết nối khác. LDAP được định nghĩa trong RFC 2251 [70]. Hiện nay, để xây dựng các hệ thống lớn, LDAP chính là giải pháp để tích hợp dữ liệu để từ đó có thể dùng chung giữa các hệ thống khác nhau.

## 4.2 Các kiến trúc PKI

Ngày nay, PKI được triển khai bởi nhiều tổ chức như là công cụ để bảo vệ những tài nguyên tập thể nhạy cảm. Tuy nhiên, với những nhu cầu, quy trình và sự phức tạp khác nhau trong mỗi công việc, chỉ một mô hình được chuẩn hóa cho PKI hoàn toàn không linh hoạt. Vì lý do đó, có nhiều kiến trúc PKI khác nhau mà mỗi tổ chức có thể triển khai để phù hợp nhất với nhu cầu của họ. Tuy vậy, cho dù kiến trúc PKI nào được triển khai, một thứ quan trọng trong cốt lõi của mỗi kiến trúc đó chính là sự tín nhiệm.

CA giúp thiết lập cho việc nhận dạng của các thực thể giao tiếp với nhau được đúng đắn. CA không chỉ chứng nhận cho người sử dụng, mà còn cho những CA khác bằng cách phát hành chứng nhận số đến chúng. Những CA đã được chứng nhận lần lượt có thể chứng nhận cho những CA khác và chuỗi mốc xích này sẽ tiếp tục cho đến khi có thể chứng nhận cho khóa công khai của thực thể cuối. Chuỗi mốc xích này được gọi là “chuỗi tín nhiệm” (chain of trust) hay “đường dẫn chứng nhận”



Hình 4.2. Mô hình chuỗi tín nhiệm

Ngược lại, khi thực thể cuối xác nhận chính mình cho một ứng dụng điện tử (như thương mại điện tử hay chính phủ điện tử), phần mềm mã hóa của ứng dụng sẽ kiểm tra chữ ký trong chứng nhận của thực thể cuối bằng việc sử dụng khóa công khai của CA tạo ra chứng nhận đó. Nếu khóa của CA này không phải là khóa “gốc” (là khóa của CA gốc được mọi người tin cậy) thì chứng nhận chứa nó cũng sẽ được xác thực với khóa công khai của CA ký chứng nhận đó, và cứ như vậy đến khi chứng nhận trong chuỗi tín nhiệm có thể được kiểm tra với một khóa gốc được tin cậy. Chuỗi được xác nhận đó lúc này hàm ý tính chất xác thực của tất cả chứng nhận, bao gồm cả chứng nhận của người dùng cuối.

Dưới đây là một số kiến trúc PKI phổ biến có thể được sử dụng để thiết lập chuỗi tín nhiệm như vậy và mỗi kiến trúc đều có những lý lẽ tán thành và phản đối khi được triển khai thực tế. Sự khác biệt giữa chúng dựa trên số lượng CA, sự sắp xếp và mối quan hệ giữa chúng [19, tr.39-57], [27].

#### **Kiến trúc PKI đơn giản:**

- Kiến trúc CA đơn (Single CA).
- Kiến trúc danh sách tín nhiệm cơ bản (Basic Trust-List).

#### **Kiến trúc PKI trong tổ chức:**

- Kiến trúc phân cấp (Hierarchical).
- Kiến trúc lưới (Mesh).

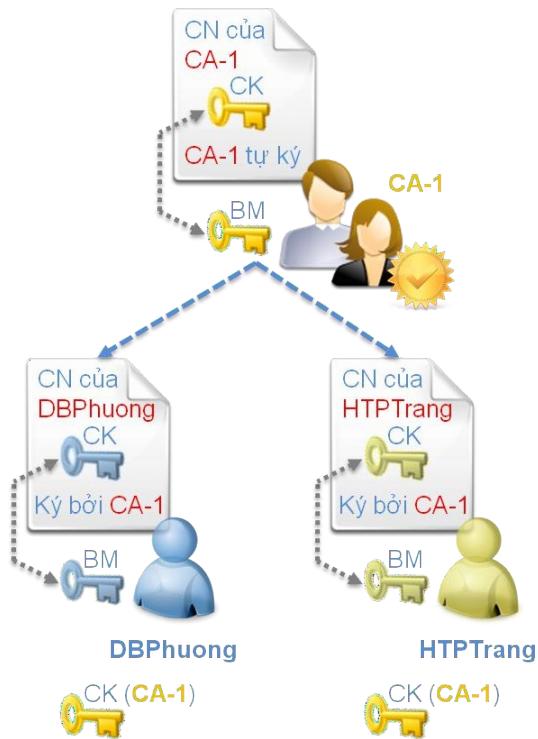
#### **Kiến trúc lai:**

- Kiến trúc danh sách tín nhiệm mở rộng (Extended Trust-List).
- Kiến trúc chứng nhận chéo (Cross-certified)
- Kiến trúc CA cầu nối (Bridge CA).
- Kiến trúc Gateway CA.

## 4.2.1 Kiến trúc CA đơn

### 4.2.1.1 Khái niệm

Kiến trúc CA đơn là kiểu kiến trúc PKI cơ bản nhất. Trong kiểu kiến trúc này, chỉ có một CA phát hành và phân phối các chứng nhận hay danh sách các chứng nhận bị hủy (CRL) đến các thực thể cuối. Tất cả thực thể đó tín nhiệm CA này và chỉ sử dụng các chứng nhận được phát hành bởi CA đó. Không có mối quan hệ tín nhiệm giữa các CA trong kiến trúc này bởi vì chỉ tồn tại duy nhất một CA. Mọi thực thể trong kiến trúc này giao tiếp với nhau trong một môi trường tin cậy nhờ sử dụng cùng một điểm tín nhiệm (trust point) chung chính là CA đó. Hình sau mô tả một kiến trúc CA đơn.



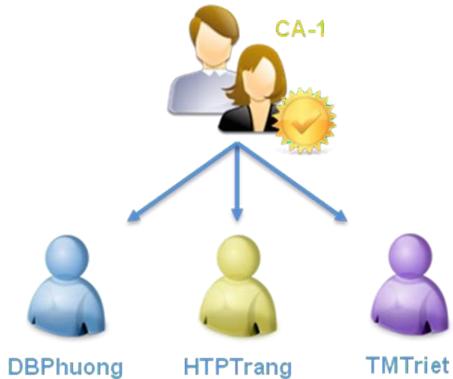
**Hình 4.3. Kiến trúc CA đơn**

Hình trên cho thấy DBPhương và HTPTrang là hai thực thể tín nhiệm CA-1. Vì thế, cả hai có thể kiểm tra và xác nhận các chứng nhận của nhau trước khi giao tiếp.

### 4.2.1.2 Đường dẫn chứng nhận

Trong kiến trúc này, sự xây dựng đường dẫn chứng nhận cực kỳ đơn giản. Có thể nói rằng không có sự xây dựng đường dẫn nào trong kiến trúc CA đơn do kiến trúc này chỉ bao gồm duy nhất một CA hay điểm tín nhiệm và vì vậy một chứng nhận đơn thuần.

hiện toàn bộ đường dẫn. Hình sau là một ví dụ thể hiện các đường dẫn chứng nhận trong kiến trúc CA đơn.



**Hình 4.4. Đường dẫn chứng nhận trong kiến trúc CA đơn**

Bằng cách sử dụng ký hiệu [Tên CA  $\equiv$  Tên thực thể cuối] cho biết CA đó phát hành chứng nhận cho thực thể cuối, đường dẫn chứng nhận của các thực thể cuối được mô tả như sau:

- [CA-1  $\equiv$  DBPhuong]
- [CA-1  $\equiv$  HTPTrang]
- [CA-1  $\equiv$  TMTriet]

Dễ thấy chứng nhận được phát hành bởi CA-1 cho thực thể cuối là đường dẫn chứng nhận hoàn chỉnh và chỉ gồm có một chứng nhận.

#### 4.2.1.3 Nhận xét

Triển khai một kiến trúc CA đơn hoàn toàn đơn giản bởi vì chỉ cần phải thiết lập duy nhất một CA. Tuy nhiên, ưu điểm đó cũng chính là khuyết điểm của kiến trúc. Do chỉ có một CA duy nhất nắm giữ các thông tin quan trọng của mọi thực thể cuối, nếu khóa bí mật của CA bị tổn thương thì mọi chứng nhận được phát hành bởi CA này sẽ trở nên vô hiệu, và kết quả là hệ thống PKI sụp đổ hoàn toàn. Vì vậy, trong trường hợp khóa công khai của CA bị tổn thương, CA nên lập tức thông báo tình trạng này đến mọi thực thể. Hơn nữa, nếu khóa bí mật của CA bị tổn thương, CA cần phải được tái thiết lập. Để tái thiết lập một CA, mọi chứng nhận được phát hành bởi CA phải được thu hồi và phải được phát hành lại đồng thời thông tin về CA mới sau đó phải

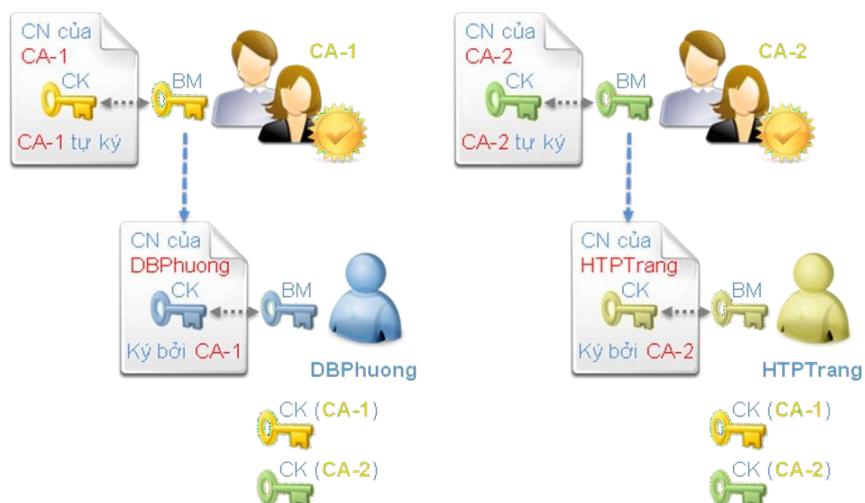
được chuyển đến mọi thực thể cuối. Vì tính chất quan trọng đó, CA phải có các thủ tục cho việc bảo vệ khóa bí mật của nó và cũng nên có một cơ chế an toàn cho việc xác nhận trực tuyến của các chứng nhận được phát hành bởi các thực thể khác nhau. Kiến trúc CA không thể mở rộng quy mô do nó không cho phép bất kỳ CA mới nào thêm vào PKI. Vì vậy, nó chỉ phù hợp cho một tổ chức nhỏ với số lượng người giới hạn và khi kích thước của tổ chức tăng lên làm cho kiến trúc này trở nên bị quá tải.

## 4.2.2 Kiến trúc danh sách tín nhiệm

### 4.2.2.1 Khái niệm

Kiến trúc CA đơn chỉ thích hợp cho tổ chức có số lượng người hạn chế. Trong trường hợp số lượng người trong tổ chức tăng lên, nhu cầu sử dụng thêm các CA khác là cần thiết. Vấn đề đặt ra là làm sao những thực thể cuối có chứng nhận được phát hành bởi các CA khác nhau nhưng có thể giao tiếp với nhau.

Một cải tiến của kiến trúc CA đơn là kiến trúc danh sách tín nhiệm cơ bản trong đó các dịch vụ PKI được cung cấp bởi nhiều CA. Trong kiến trúc này, các thực thể cuối phải duy trì một danh sách các CA mà họ tin cậy và chỉ sử dụng các chứng nhận và CRL được phát hành bởi các CA trong danh sách các CA được tín nhiệm của nó.

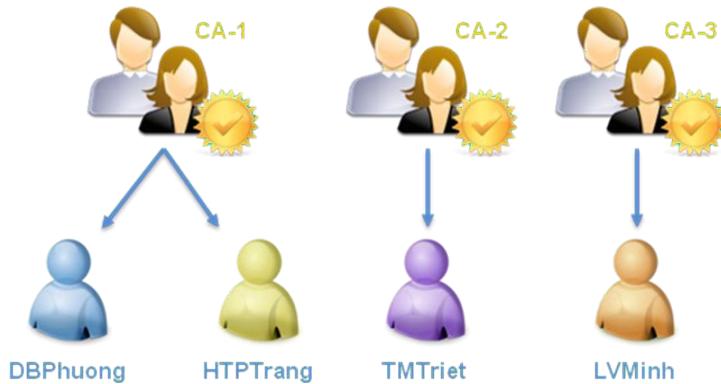


**Hình 4.5. Kiến trúc danh sách tín nhiệm**

Ở ví dụ trên, danh sách tín nhiệm của DBPhuong và HTPTrang đều là {CA-1, CA-2}

#### 4.2.2.2 Đường dẫn chứng nhận

Tuy có nhiều CA trong kiến trúc nhưng những CA này không thiết lập một mối quan hệ tín nhiệm giữa chúng. Do đó, đường dẫn chứng nhận trong kiến trúc danh sách tín nhiệm cũng chỉ chứa duy nhất một chứng nhận đơn.



**Hình 4.6. Đường dẫn chứng nhận trong kiến trúc danh sách tín nhiệm**

Trong ví dụ trên, đường dẫn chứng nhận của các thực thể cuối được mô tả như sau:

- [CA-1 → DBPhuong]
- [CA-1 → HTPTrang]
- [CA-2 → TMTriet]
- [CA-3 → LVMinh]

#### 4.2.2.3 Nhận xét

Ngược lại với kiến trúc CA đơn trong đó CA mới không thể được thêm vào PKI, trong kiến trúc này ta có thể thêm CA mới bằng cách thay đổi trong danh sách tín nhiệm. Mặc dù kiến trúc này có một ưu điểm dễ thấy trong việc đơn giản trong thiết kế nhưng đôi khi nó có thể trở nên hoàn toàn phức tạp. Với sự tăng thêm của số lượng CA được tin cậy bởi một thực thể, số lượng thực thể trong danh sách tín nhiệm của tăng lên. Ngoài ra, thông tin quan trọng về các CA được tin cậy cũng được duy trì bởi mọi thực thể. Sự cập nhật thông tin này có thể cho thấy một nhiệm vụ vất vả cho các thực thể khi số lượng CA tăng lên.

Mô hình này được thực thi trong các trình duyệt web Netscape và Microsoft, cho phép người sử dụng linh động trong việc thêm hay gỡ bỏ CA từ danh sách tín

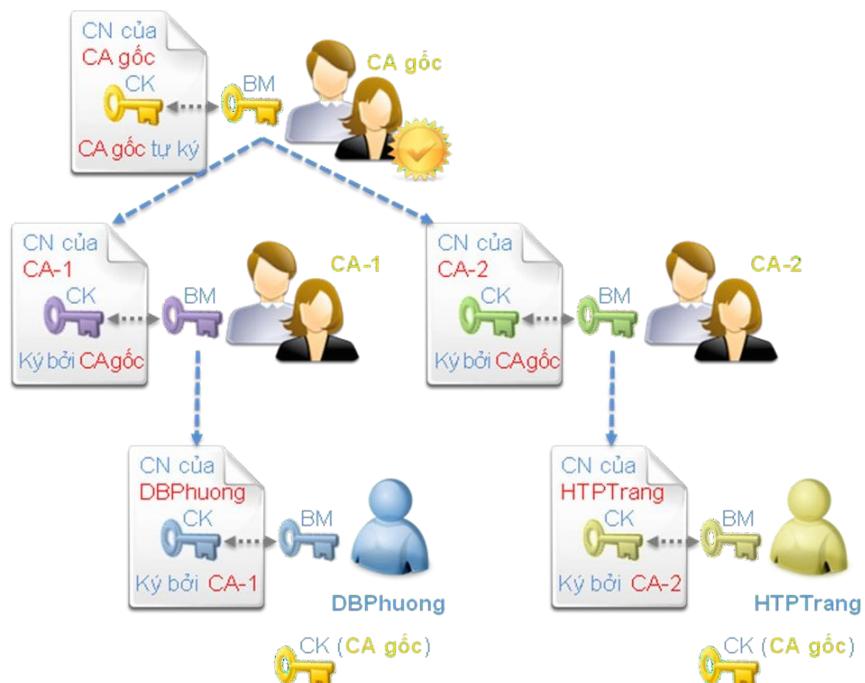
nhiệm của mình. Tuy nhiên, một tổ chức hiềm độc có khả năng thêm một chứng nhận CA không có thật hay giả mạo vào danh sách và từ đó các chứng nhận người dùng không có thật được xác nhận bởi trình duyệt.

#### 4.2.3 Kiến trúc phân cấp

#### **4.2.3.1 Khái niệm**

Khi số lượng thực thể cuối trong tổ chức tăng lên, việc quản lý các chứng nhận và sự xác thực bắt đầu phức tạp và đòi hỏi nhiều thời gian cho một CA đơn. Đây thực sự trở thành gánh nặng và vì thế nhu cầu chia sẻ công việc cho các CA khác trở nên cần thiết.

Kiến trúc PKI phân cấp là kiến trúc PKI phổ biến nhất thường được triển khai trong những tổ chức có quy mô lớn. Trong kiến trúc này, các dịch vụ PKI được cung cấp bởi nhiều CA. Không giống như kiến trúc danh sách tín nhiệm, mọi CA trong kiến trúc PKI phân cấp chia sẻ mối quan hệ tín nhiệm giữa chúng. Các CA trong kiến trúc này được kết nối thông qua mối quan hệ phụ thuộc cấp trên.



**Hình 4.7. Kiến trúc PKI phân cấp**

Sự phân cấp CA là một cấu trúc giống cây lộn ngược có gốc ở trên đỉnh (top-down), được gọi là CA gốc (root CA), và từ đó phát triển ra các nhánh hay nút. Những nút

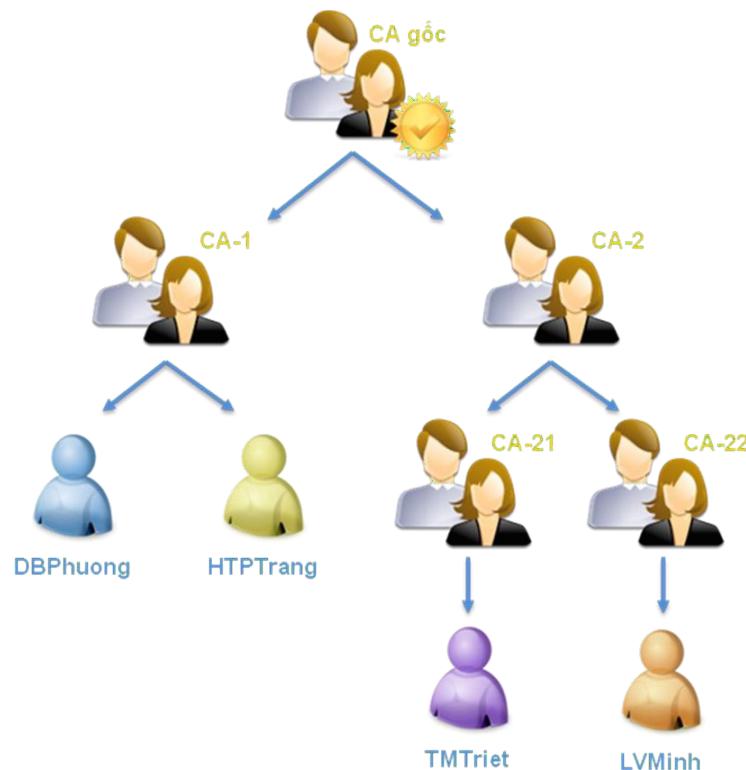
là

các CA cấp dưới của CA gốc. Các CA cấp dưới này cũng giống bất kỳ CA khác và thực hiện các chức năng của một CA. Chúng cũng có thể ủy thác trách nhiệm của việc phát hành chứng nhận cho các CA cấp dưới hơn của nó. Bất cứ lúc nào CA gốc bổ nhiệm một CA cấp dưới, CA gốc sẽ phát hành một chứng nhận đến CA cấp dưới đó nhằm cho biết loại công việc nào cấp dưới có thể thực hiện.

CA gốc luôn luôn phát hành chứng nhận đến các CA cấp dưới chứ không cho thực thể cuối. Tuy nhiên, các CA cấp dưới có thể phát hành các chứng nhận cho cả thực thể cuối và CA cấp dưới hơn của nó. Trong PKI phân cấp, CA cấp dưới không phát hành chứng nhận cho CA cấp trên của nó hoặc CA gốc. Trừ trường hợp CA gốc, tất cả các CA khác có một CA cấp trên duy nhất phát hành chứng nhận cho nó. CA gốc tự phát ký chứng nhận cho mình (self-signed) và được mọi thực thể cuối tin nhiệm.

#### **4.2.3.2 Đường dẫn chứng nhận**

Đường dẫn chứng nhận trong kiến trúc PKI phân cấp khá ngắn và duy nhất, bắt đầu từ gốc cho đến chứng nhận của thực thể cuối.



**Hình 4.8. Đường dẫn chứng nhận trong kiến trúc PKI phân cấp**

Đường dẫn chứng nhận cho DBPhuong được mô tả như sau:

- [Root CA  $\equiv$  CA-1] : [CA-1  $\equiv$  DBPhuong]

Đường dẫn chứng nhận cho HTPTrang được mô tả như sau:

- [Root CA  $\equiv$  CA-1] : [CA-1  $\equiv$  HTPTrang]

Đường dẫn chứng nhận cho TMTriet được mô tả như sau:

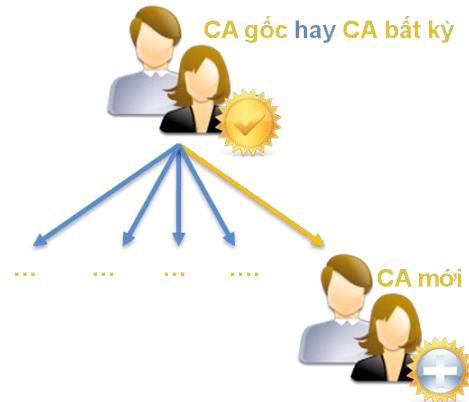
- [Root CA  $\equiv$  CA-2] : [CA-2  $\equiv$  CA-21] : [CA-21  $\equiv$  TMTriet]

Đường dẫn chứng nhận cho LVMinh được mô tả như sau:

- [Root CA  $\equiv$  CA-2] : [CA-2  $\equiv$  CA-22] : [CA-22  $\equiv$  LVMinh]

#### 4.2.3.3 Nhận xét

PKI phân cấp hoàn toàn có thể mở rộng vì vậy chúng dễ dàng thỏa mãn nhu cầu phát triển của tổ chức. Để thêm vào một thực thể mới trong hệ thống PKI, CA gốc hay CA bất kỳ đơn giản thiết lập một mối quan hệ tin cậy với thực thể CA cấp dưới đó bằng cách phát hành một chứng nhận đến CA mới này.



**Hình 4.9. Mở rộng kiến trúc PKI phân cấp**

Hơn nữa, do chứng nhận chỉ được phát hành theo một hướng nên kiến trúc PKI phân cấp hoàn toàn dễ triển khai. Đường dẫn cho một thực thể cuối đến gốc hay CA phát hành có thể được xác định dễ dàng và nhanh chóng.

Tuy nhiên, kiến trúc PKI phân cấp gặp một trở ngại chính, đó là chỉ có một điểm tín nhiệm duy nhất (chính là CA gốc) điều khiển toàn bộ kiến trúc PKI phân cấp. Nếu sự cố thường nằm ở CA cấp dưới vẫn có thể giải quyết được bằng cách các CA cấp trên

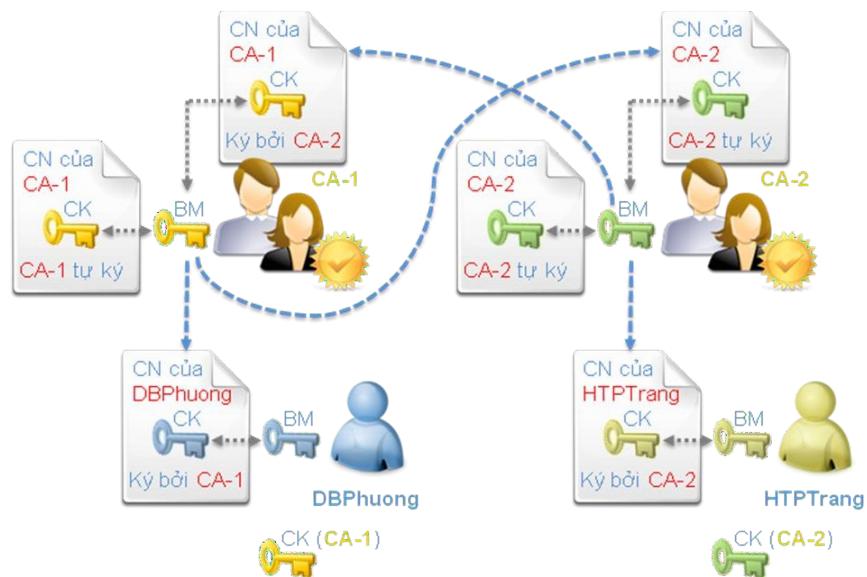
thu hồi các chứng nhận của chúng và thiết lập lại. Trong trường hợp CA gốc bị tổn thương, toàn bộ sự tín nhiệm trên kiến trúc PKI sẽ sụp đổ.

Hơn nữa, việc chuyển từ một tập CA cô lập và trong PKI phân cấp có thể không thực hiện được bởi vì lúc đó mọi thực thể phải điều chỉnh lại điểm tin cậy của mình. Một vấn đề khác kiến trúc PKI phân cấp không phù hợp trong các mối quan hệ ngang hàng. Ví dụ, khi hai tổ chức muốn hoạt động trong cùng một kiến trúc thì ai sẽ quản lý CA gốc. Để khắc phục sự điều này, kiến trúc lưới được sử dụng.

#### 4.2.4 Kiến trúc lưới

##### 4.2.4.1 Khái niệm

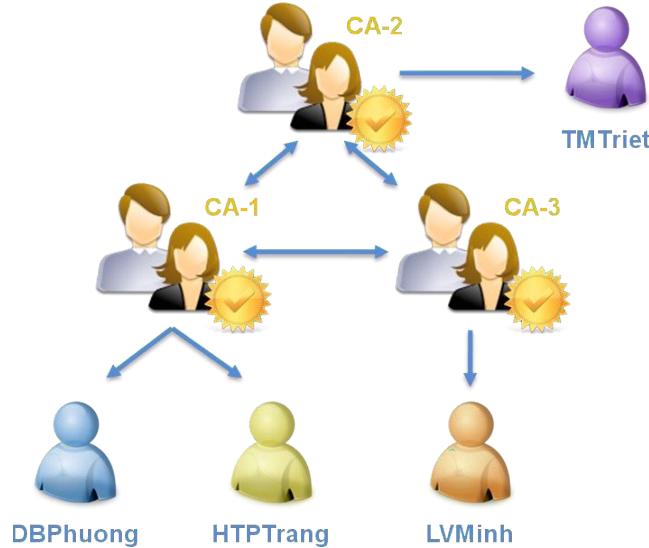
Trong kiến trúc PKI lưới, các CA có mối quan hệ ngang hàng (peer-to-peer) và không có CA đơn lẻ nào trong toàn bộ kiến trúc PKI. Mọi CA trong PKI lưới có thể là điểm tín nhiệm và thực thể cuối tín nhiệm CA phát hành chứng nhận cho họ. Trong kiến trúc này, mọi CA chứng nhận chéo cho nhau (cross-certified). Sự chứng nhận chéo là quy trình kết nối hai CA nhằm thiết lập một mối quan hệ tin cậy hai chiều. Hai CA sẽ chứng nhận chéo bất cứ khi nào các thực thể tương ứng của chúng cần giao tiếp một cách an toàn.



**Hình 4.10. Kiến trúc lưới**

#### 4.2.4.2 Đường dẫn chứng nhận

Trong kiến trúc này, đường dẫn chứng nhận được bắt đầu tại điểm tín nhiệm và di chuyển về hướng CA phát hành chứng nhận cho thực thể cuối đang cần xây dựng đường dẫn chứng nhận.



**Hình 4.11. Đường dẫn chứng nhận trong kiến trúc lưới**

Một CA có thể có nhiều chứng nhận chéo vì vậy việc xây dựng đường dẫn thêm phức tạp do có nhiều lựa chọn. Một trong số lựa chọn dẫn đến một đường dẫn hợp lệ trong khi các lựa chọn khác dẫn đến ngõ cụt hoặc rơi vào vòng lặp chứng nhận vô tận. Độ dài tối đa của đường dẫn chứng nhận trong PKI lưới là số lượng CA có trong PKI.

Ngoài ra, kiến trúc lưới liên quan đến sự xây dựng của các đường dẫn chứng nhận khác nhau bởi các người sử dụng khác nhau. Do điểm tín nhiệm luôn là CA phát hành chứng nhận cho thực thể cuối, nên khi DBPhuong xây dựng một đường dẫn chứng nhận cho TMTriet, điểm khởi đầu là CA đã phát hành chứng nhận cho DBPhuong (là CA-1) và điểm cuối là chứng nhận của TMTriet. Tương tự, khi LVMinh xây dựng một đường dẫn chứng nhận cho TMTriet, điểm khởi đầu là nhà phát hành chứng nhận cho LVMinh (là CA-3) và điểm cuối là chứng nhận của TMTriet. Các đường dẫn chứng nhận được dựng lên bởi DBPhuong cho TMTriet:

- [CA-1  $\equiv$  CA-2] : [CA-2  $\equiv$  TMTriet]
- [CA-1  $\equiv$  CA-3] : [CA-3  $\equiv$  CA-2] : [CA-2  $\equiv$  TMTriet]

Các đường dẫn chứng nhận được dựng lên bởi LVMinh cho TMTriet:

- [CA-3 CA-2] : [CA-2 TMTriet]
- [CA-3 CA-1] : [CA-1 CA-2] : [CA-2 TMTriet]

Trong cả hai trường hợp, đường dẫn chứng nhận sẽ không như nhau trừ khi cả DBPhuong và LVMinh đều cùng có chung nhà phát hành CA hay điểm tín nhiệm giống như DBPhuong với HTPTrang (chung một điểm tín nhiệm là CA-1)

Mặc khác, do kiến trúc lưới chứa nhiều mối quan hệ hai chiều giữa các CA, thường có nhiều hơn một đường dẫn chứng nhận giữa thực thể bất kỳ và một điểm tín nhiệm. Trong kiến trúc phân cấp, xây dựng một đường dẫn chứng nhận từ chứng nhận người dùng đến điểm tín nhiệm cao nhất là tất định trong khi ở kiến trúc lưới là bất định. Sự tìm ra đường dẫn khó hơn trong trường hợp kiến trúc ngang hàng. Độ dài đường dẫn có thể dài hơn trong kiến trúc PKI phân cấp.

#### **4.2.4.3 Nhận xét**

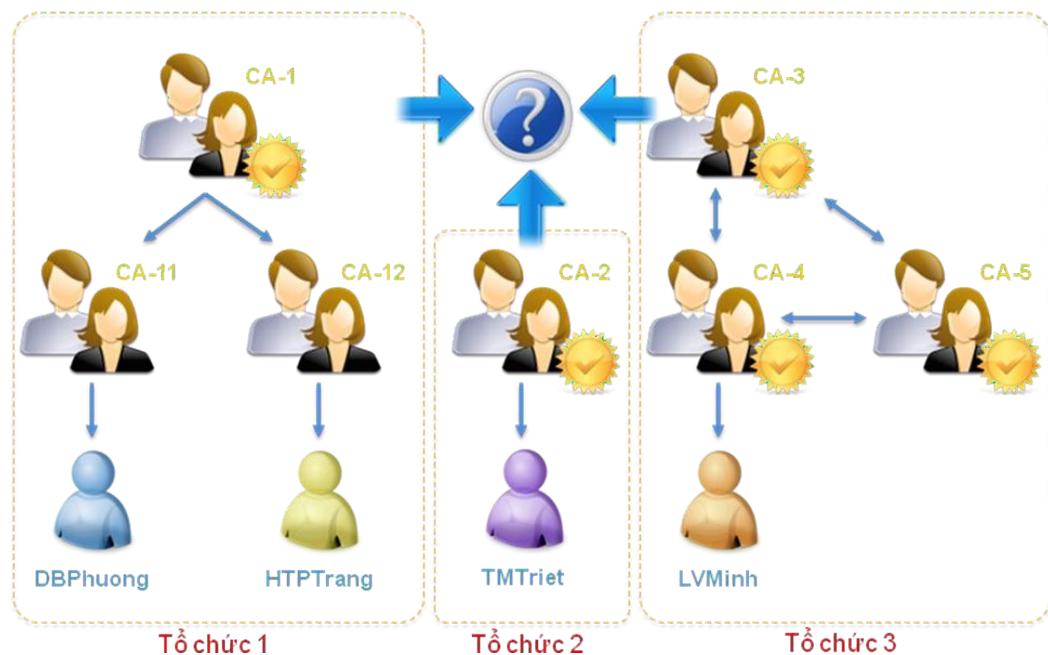
Có nhiều điểm tin cậy trong kiến trúc PKI lưới, và do đó sự tổn thương của một CA đơn lẻ không thể làm sụp đổ toàn bộ PKI mà chỉ ảnh hưởng đến các thực thể liên kết với CA bị tổn thương đó. Lúc này, chứng nhận của CA bị tổn thương sẽ bị thu hồi bởi các CA đã phát hành chứng nhận đến CA đó.

CA mới dễ dàng được thêm vào kiến trúc PKI bằng cách phát hành chứng nhận đến ít nhất một CA khác trong lưới. Kiến trúc lưới có một điểm đặc biệt là mỗi CA phải kết nối với các CA khác, tạo thành một đồ thị đầy đủ. Điều đó có nghĩa là nếu có  $n$  CA, số lượng liên kết cần thiết sẽ là  $n \times (n - 1)$ . Do đó, khi số lượng CA tăng lên, số lượng chứng nhận chéo cũng như số lượng chuỗi chứng nhận trở nên vô cùng lớn.

#### **4.2.5 Kiến trúc lai**

Các kiến trúc PKI kể trên trong chừng mực nào đó đã thỏa mãn các nhu cầu của một tổ chức hay một nhóm người sử dụng. Tuy nhiên, khi các tổ chức muốn tương tác với nhau thì việc triển khai kiến trúc PKI trở nên phức tạp do các tổ chức này không phải lúc nào cũng sử dụng các kiến trúc PKI giống nhau. Ví dụ, một tổ chức triển khai kiến trúc CA đơn, trong khi tổ chức khác lại triển khai kiến trúc phân cấp hay lưới.

Trong tình huống như vậy, PKI cần cung cấp một giải pháp tối ưu cho phép các tổ chức có thể tương tác với nhau trong một môi trường tin cậy. Trong trường hợp này, kiến trúc “lai” sẽ rất hữu dụng trong việc cho phép quá trình tương tác giữa các tổ chức thành công.



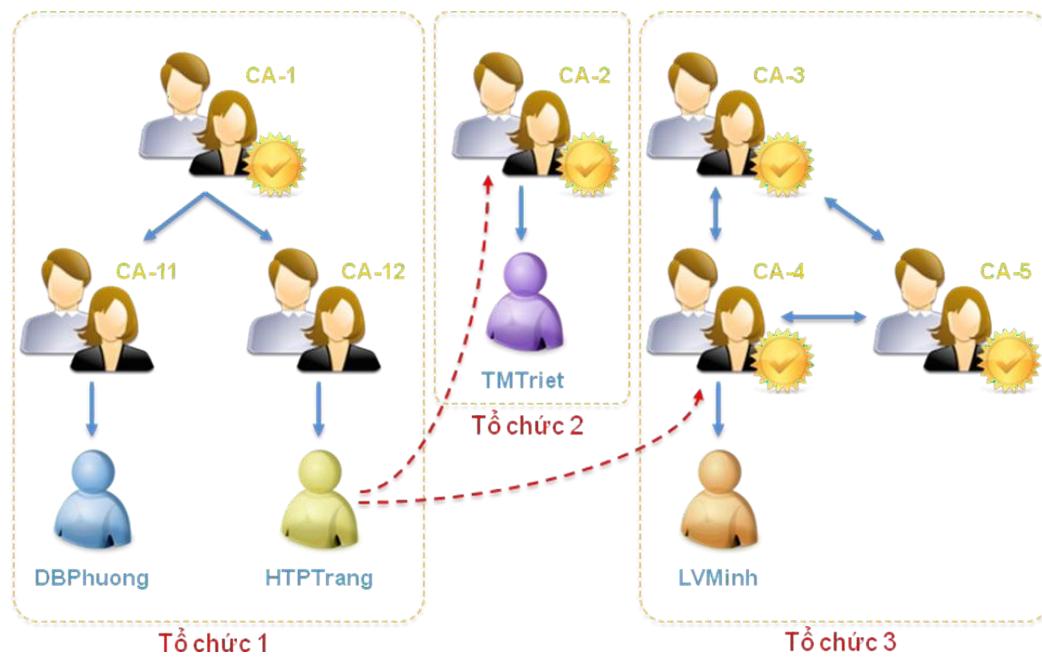
**Hình 4.12. Các PKI được triển khai ở các tổ chức khác nhau**

Có ba loại kiến trúc PKI lai, đó là:

- **Kiến trúc danh sách tín nhiệm mở rộng (Extended Trust List):** dạng mở rộng kiến trúc danh sách tín nhiệm để hỗ trợ đường dẫn tín nhiệm có độ dài nhiều hơn một chứng nhận.
- **Kiến trúc PKI chứng nhận chéo (Cross-certified PKI):** các PKI thiết lập mối quan hệ ngang hàng để cho phép giao tiếp an toàn.
- **Kiến trúc CA cầu nối (Bridge CA):** hỗ trợ cho các kiến trúc PKI phức tạp.

#### 4.2.5.1 Kiến trúc danh sách tín nhiệm mở rộng

Giống như kiến trúc danh sách tín nhiệm cơ bản, ở kiến trúc này tất cả các thực thể cuối sử dụng PKI lưu trữ một danh sách mở rộng của tất cả các điểm tín nhiệm. Mỗi điểm tín nhiệm liên quan đến một PKI của mỗi tổ chức mà thực thể cuối tin cậy. PKI đó có thể là một CA đơn, PKI phân cấp hay PKI lưới. Nếu là kiến trúc phân cấp, điểm tín nhiệm là CA gốc còn nếu là kiến trúc lưới, điểm tín nhiệm là CA bất kỳ.



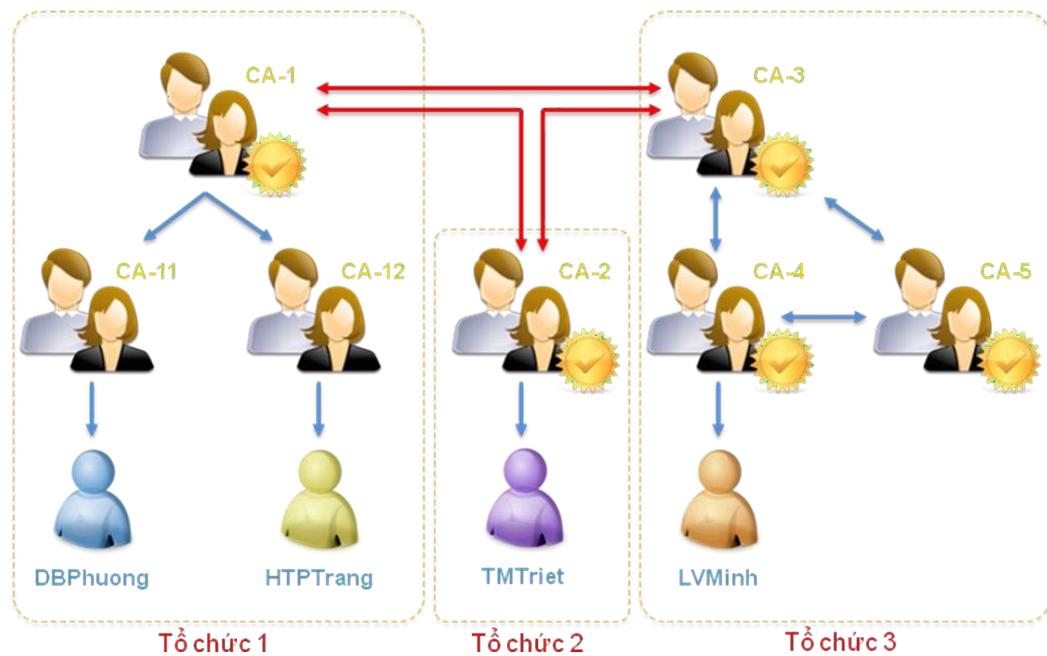
Hình 4.13. Kiến trúc danh sách tín nhiệm mở rộng

Danh sách tín nhiệm của DBPhuong, HTPTrang, TMTriet là {CA-1, CA-2, CA-3} hoặc {CA-1, CA-2, CA-4} hoặc {CA-1, CA-2, CA-5}.

Danh sách tín nhiệm của LVMinh là {CA-1, CA-2, CA-4}.

#### 4.2.5.2 Kiến trúc chứng thực chéo

Trong kiến trúc chứng nhận chéo, CA gốc của một cơ sở hạ tầng của tổ chức nắm giữ mối quan hệ ngang hàng với những gốc CA của các tổ chức khác. tức là các CA gốc của mỗi nhóm sẽ cấp chứng nhận cho nhau (cross-certification). Kiến trúc này tốt cho một nhóm nhỏ các PKI của tổ chức muốn thiết lập mối quan hệ tín nhiệm.



**Hình 4.14. Kiến trúc chứng nhận chéo**

Đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho HTPTrang:

- [CA-1 ≡ CA-12] : [CA-12 ≡ HTPTrang]

Đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho TMTriet:

- [CA-1 ≡ CA-2] : [CA-2 ≡ TMTriet]

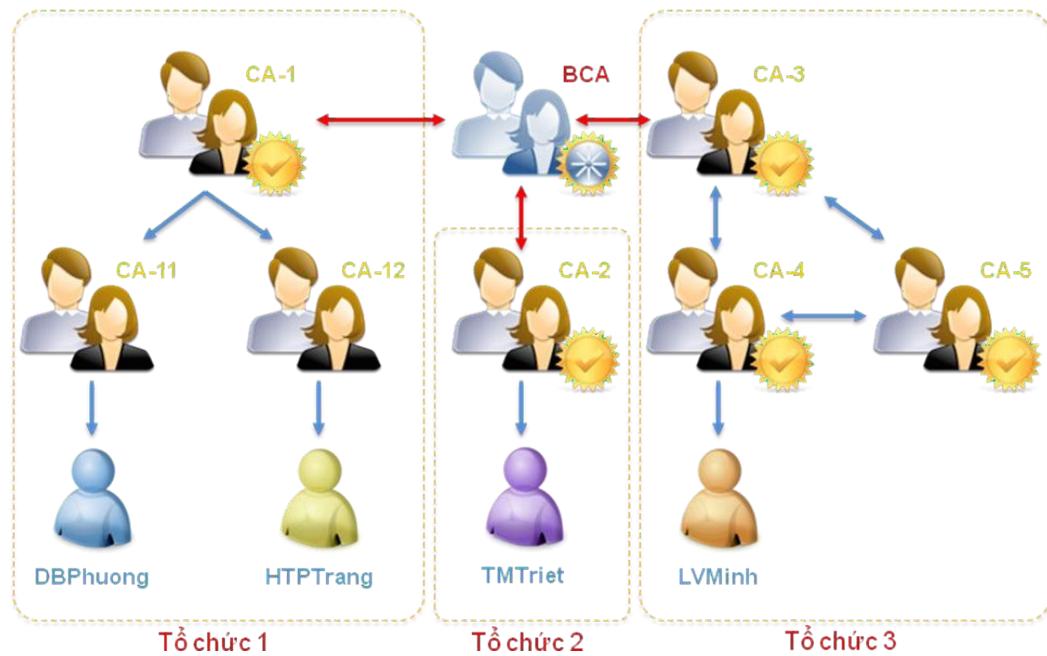
Các đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho LVMinh:

- [CA-1 ≡ CA-3] : [CA-3 ≡ CA-4] : [CA-4 ≡ LVMinh]
- [CA-1 ≡ CA-3] : [CA-3 ≡ CA-5] : [CA-5 ≡ CA-4] : [CA-4 ≡ LVMinh]

#### 4.2.5.3 Kiến trúc CA cầu nối

Kiến trúc CA cầu nối là kiến trúc phù hợp nhất để liên kết các PKI có kiến trúc khác nhau. Không giống kiến trúc chứng thực chéo, nơi nào tồn tại mối quan hệ ngang hàng giữa các CA gốc trong mỗi cơ sở hạ tầng của tổ chức, một thực thể mới gọi là CA cầu nối (Bridge CA – BCA) lưu giữ quan hệ ngang hàng của giữa các CA này.

Sự thiết lập của một mối quan hệ tín nhiệm trong kiến trúc này phụ thuộc vào loại kiến trúc PKI mà sự tín nhiệm được thiết lập. Đối với kiến trúc PKI phân cấp, sự tín nhiệm được thiết lập với CA gốc, đối với kiến trúc PKI lưới, mỗi quan hệ tín nhiệm được thiết lập với bất kỳ CA trong PKI lưới đó. Các mối quan hệ giữa một CA chính với CA cầu nối là ngang hàng. Cách này làm giảm đáng kể số lượng chứng nhận chéo.



**Hình 4.15. Kiến trúc CA cầu nối**

Đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho HTPTrang:

- [CA-1  $\equiv$  CA-12] : [CA-12  $\equiv$  HTPTrang]

Đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho TMTriet:

- [CA-1  $\equiv$  BCA] : [BCA  $\equiv$  CA-2] : [CA-2  $\equiv$  TMTriet]

Các đường dẫn chứng nhận sau được dựng lên bởi DBPhuong cho LVMinh:

- [CA-1  $\equiv$  BCA] : [BCA  $\equiv$  CA-3] : [CA-3  $\equiv$  CA-4] : [CA-4  $\equiv$  LVMinh]
- [CA-1  $\equiv$  BCA] : [BCA  $\equiv$  CA-3] : [CA-3  $\equiv$  CA-5] : [CA-5  $\equiv$  CA-4] : [CA-4  $\equiv$  LVMinh]

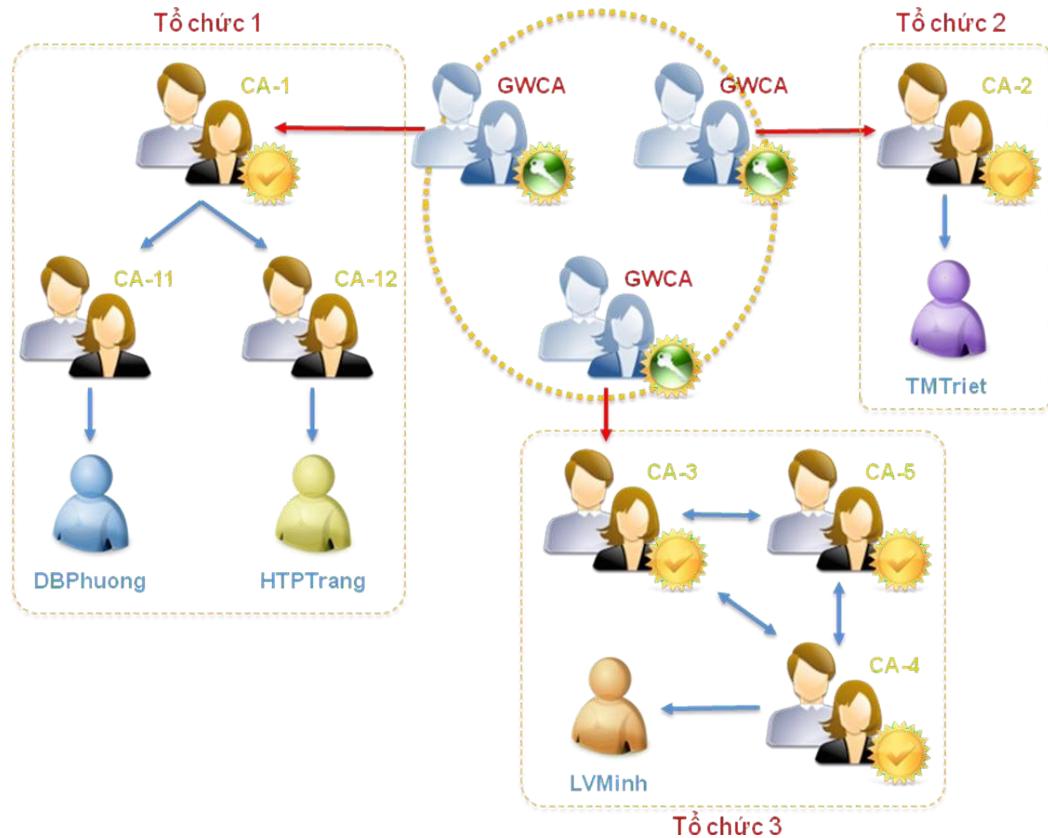
Kiến trúc này dễ thêm một CA hoặc toàn bộ PKI vào kiến trúc và sự thay đổi này “trong suốt” với người dùng và không có thay đổi nào trong các điểm tín nhiệm xảy ra.

#### 4.2.5.4 Kiến trúc Gateway CA

Vào năm 2005, Zheng Guo, Tohru Okuyama và Marion R. Finley. Jr đề xuất kiến trúc mới có tên Gateway CA (GWCA) cho phép các PKI của các tổ chức khác nhau có thể cùng hoạt động [27]. Ý tưởng chính của kiến trúc này là các GWCA được kết nối với nhau trên một cấu hình vòng, các CA cấp dưới hay trung gian có thể được kết nối trong cấu hình phân cấp hoặc cầu nối.

Kiến trúc này có các đặc điểm sau:

- Các GWCA nằm trên một vòng như một CA gốc được chia thành nhiều CA giống nhau, mỗi CA tương ứng với một CA gốc của các CA cấp dưới của nó.
- GWCA là điểm tín nhiệm duy nhất của các thực thể cuối và thực thể cuối chỉ cần tín nhiệm bất kỳ GWCA nào thì nó cũng sẽ tín nhiệm toàn bộ hệ thống.



**Hình 4.16. Kiến trúc Gateway CA**

Do các GWCA thực ra giống nhau và xem như một CA gốc ảo (virtual Root CA) được chia ra thành nhiều phần nên sẽ có cùng cặp khóa. Vì vậy sẽ không an toàn nếu một GWCA bị lộ khóa bí mật. Giải pháp được chia sẻ bí mật được Shamir đề nghị

[52] là các GWCA sẽ có cùng khóa công khai còn khóa bí mật sẽ được chia thành  $n$  mảnh, mỗi GWCA sẽ giữ một mảnh với đặc điểm sau:

- Khi cần khóa bí mật, hệ thống sẽ tập hợp  $k$  mảnh trong  $n$  mảnh.
- Nếu chỉ có  $k-1$  mảnh, không thể tạo thành khóa bí mật.

Ưu điểm của giải pháp này là nếu như một GWCA bị lộ phần khóa bí mật cũng không ảnh hưởng đến hệ thống do không thể tìm được khóa bí mật với chỉ một mảnh khóa bí mật. Kỹ thuật chia sẻ khóa này thường được ứng dụng trong mạng ad hoc [51].

#### 4.2.6 Nhận xét

Bảng sau cho thấy các ưu điểm và khuyết điểm của các kiến trúc PKI được triển khai trong phạm vi một tổ chức:

**Bảng 4.1. So sánh các kiến trúc PKI đơn giản**

| Kiến trúc           | Ưu điểm   | Khuyết điểm  |
|---------------------|---|--|
| CA đơn              | <ul style="list-style-type: none"> <li>Chi phí rẻ, đơn giản và dễ triển khai.</li> <li>Đường dẫn chứng nhận đơn giản.</li> </ul>  | <ul style="list-style-type: none"> <li>Do chỉ có một CA nên nếu CA này bị tổn thương thì toàn bộ hệ thống sẽ sụp đổ.</li> <li>Không thể mở rộng quy mô, chỉ phù hợp cho tổ chức nhỏ.</li> </ul>  |
| Danh sách tín nhiệm | <ul style="list-style-type: none"> <li>Đơn giản, dễ triển khai, rất linh hoạt.</li> <li>Đường dẫn chứng nhận đơn giản.</li> </ul>   | <ul style="list-style-type: none"> <li>Khó quản lý.</li> <li>Thực thể cuối phải lưu trữ rất nhiều thông tin của các tổ chức tín nhiệm và phải cập nhật thông tin thường xuyên.</li> </ul>  |
| PKI phân cấp        | <ul style="list-style-type: none"> <li>Sự quản lý có cấu trúc.</li> <li>Triển khai trong tổ chức lớn.</li> <li>Đường dẫn chứng nhận khá ngắn, dễ xây dựng</li> <li>Tiềm năng lớn, dễ mở rộng</li> </ul> | <ul style="list-style-type: none"> <li>Chỉ có một điểm tín nhiệm là CA gốc nên nếu bị tổn thương thì toàn bộ hệ thống sẽ sụp đổ.</li> <li>Không thích hợp trong môi trường có mối quan hệ ngang hàng vì lúc này nảy sinh vấn đề ai sẽ quản lý CA gốc.</li> </ul> |
| Lưới                | <ul style="list-style-type: none"> <li>Triển khai trong tổ chức lớn.</li> <li>Sự tổn thương một CA đơn lẻ sẽ không ảnh hưởng đến toàn bộ hệ thống.</li> </ul>   | <ul style="list-style-type: none"> <li>Đường dẫn chứng nhận phức tạp và khó tìm.</li> <li>Số lượng chứng nhận chéo quá nhiều.</li> </ul>   |

Dễ dàng nhận thấy kiến trúc *PKI phân cấp* rất thích hợp khi triển khai trong các tổ chức có sự quản lý chặt chẽ. Kiến trúc này có sự quản lý chặt chẽ theo cấu trúc phân cấp, phụ thuộc cấp trên và xây dựng đường dẫn chứng nhận cũng khá đơn giản. Ngoài ra, trong tổ chức có số lượng thực thể cuối ít, kiến trúc CA đơn cũng được sử dụng rất hiệu quả.

Để các PKI của các tổ chức khác nhau có thể cùng hoạt động ta có thể sử dụng các kiến trúc lai. Dưới đây là bảng so sánh các ưu và khuyết điểm của các kiến trúc lai này.

**Bảng 4.2. So sánh các kiến trúc PKI lai**

| Kiến trúc                   | Ưu điểm  | Khuyết điểm  |
|-----------------------------|--|--|
| Danh sách tín nhiệm mở rộng | <ul style="list-style-type: none"> <li>Cho phép các hệ thống PKI có kiến trúc khác cùng hoạt động.</li> </ul>  | <ul style="list-style-type: none"> <li>Khó quản lý</li> <li>Thực thể cuối phải lưu trữ rất nhiều thông tin của các tổ chức tín nhiệm và phải cập nhật thông tin thường xuyên.</li> </ul> |
| Chứng nhận chéo             | <ul style="list-style-type: none"> <li>Cho phép các hệ thống PKI có kiến trúc khác cùng hoạt động.</li> </ul>  | <ul style="list-style-type: none"> <li>Số lượng chứng nhận chéo tăng cao khi mở rộng kiến trúc</li> </ul>  |
| CA cầu nối                  | <ul style="list-style-type: none"> <li>Cho phép các hệ thống PKI có kiến trúc khác cùng hoạt động.</li> <li>Số lượng chứng nhận chéo ít.</li> <li>Dễ dàng mở rộng, thích nghi với các PKI hiện có và trong suốt với người dùng.</li> </ul> | <ul style="list-style-type: none"> <li>Có thể khó khăn trong việc cho phép các kiến trúc khác nhau có thể cùng hoạt động nếu không được tổ chức và quy định tốt.</li> </ul>              |
| GatewayCA                   | <ul style="list-style-type: none"> <li>Cho phép các hệ thống PKI có kiến trúc khác cùng hoạt động.</li> </ul>  | <ul style="list-style-type: none"> <li>Chưa được đặc tả hoàn chỉnh, và kiểm chứng trong thực tế.</li> </ul>  |

Có thể nhận thấy kiến trúc *CA cầu nối* là kiến trúc phù hợp nhất để liên kết các PKI có kiến trúc khác nhau. Kiến trúc này giảm đáng kể số lượng chứng nhận chéo so với kiến trúc chứng nhận chéo. Ngoài ra việc mở rộng quy mô trong kiến trúc này rất đơn giản và không ảnh hưởng đến các hoạt động trước đó của thực thể cuối.

### 4.3 Kết luận

PKI cho phép những người tham gia xác thực lẫn nhau và sử dụng thông tin từ các chứng nhận khóa công khai để mã hóa và giải mã thông tin. Đặc biệt, nó cho phép các giao dịch điện tử được diễn ra đảm bảo tính cẩn mật, toàn vẹn, xác thực và không thể phủ nhận mà không cần phải trao đổi các thông tin mật từ trước.

Như đã phân tích ở trên, *kiến trúc PKI phân cấp* rất thích hợp khi triển khai trong các tổ chức có sự quản lý chặt chẽ và *kiến trúc CA cầu nối* là kiến trúc phù hợp nhất để liên kết các PKI được thực thi với những kiến trúc khác nhau. Thật vậy, đây là hai kiến trúc điển hình được hầu hết các quốc gia trên thế giới áp dụng.

Mỗi kiến trúc đều có những điểm mạnh, điểm yếu riêng. Kiến trúc CA cầu nối có lợi điểm là các đơn vị có thể triển khai ngay, tự do phát triển các CA. Tuy nhiên, đầu tư xây dựng hệ thống CA cầu nối sẽ rất phức tạp và tốn kém. Ví dụ, Trung Quốc đã áp dụng kiến trúc này 3 năm nay vẫn còn nhiều vấn đề chưa giải quyết nổi hay Mỹ đã thực hiện 7 năm rồi nhưng vẫn chưa thực sự khả quan. Ngược lại, mô hình PKI phân cấp theo kiểu Hàn Quốc có lợi thế là triển khai đơn giản, dễ quản lý và tiện cho người dùng, mỗi người chỉ cần một bộ khóa cho mọi dịch vụ chứng thực. Theo số liệu đến tháng 5/2006, Trung Quốc có đến 77 CA nhưng chỉ xếp hạng chính phủ điện tử là 57/191 nước trong khi Hàn Quốc chỉ có 6 CA nhưng xếp hạng rất cao là 5/191.

Tại hội thảo lần thứ nhất về chuyên đề “Ứng dụng chữ ký số và dịch vụ chứng thực chữ ký số” do Bộ BCVT tổ chức ngày 6/4/2006 và tại hội thảo “Mô hình tổ chức và chính sách phát triển hệ thống chứng thực quốc gia” do Bộ BCVT tổ chức ngày 25/5/2006, hầu hết các chuyên gia tham dự đều nhất trí rằng với diện tích đất nước nhỏ như Việt Nam thì nên chọn mô hình PKI tập trung kiểu phân cấp hình cây, tức là một tổ chức CA chung của quốc gia (Root CA) và bên dưới là hệ thống các CA phụ. Mô hình này vừa đơn giản lại vừa dễ triển khai và quản lý với phù hợp với những nước mới bắt đầu. Hơn nữa, trong “Hội thảo chia sẻ kinh nghiệm về xây dựng hệ thống CA” được tổ chức vào giữa tháng 5/2006, các chuyên gia Hungary (quốc gia có các dịch vụ chứng thực phát triển rất mạnh) cũng đã khuyến nghị Việt Nam nên chọn mô hình PKI hình cây giống như mô hình của Hungary đang áp dụng. Ngoài ra, cũng có ý kiến cho rằng Việt Nam nên phát triển 2 hệ thống CA (giống Malaysia và một số quốc gia khác trong khu vực), 1 hệ thống CA công cộng cung cấp cho người dân và các tổ chức tư nhân, 1 CA cho các cơ quan nhà nước. Có 2 CA hoạt động song song cũng là giải pháp phòng trường hợp một CA bị sập hoặc bị phá vỡ.

Như vậy, hệ thống chứng thực trên mô hình PKI phân cấp cần được nghiên cứu để triển khai trong thực tế. Trước hết, một trong các vấn đề cần quan tâm hàng đầu là hệ mã khóa công khai (đặc biệt là RSA), hạt nhân của PKI phải thật sự mang đến sự an toàn. Chương 5 và Chương 6 sẽ tập trung nghiên cứu và phân tích về vấn đề này.

## Chương 5

# Phân tích một số nguy cơ tồn thương trong hệ mã RSA

☐ Nội dung của chương này tập trung phân tích các nguy cơ tấn công gây tổn thương trên hệ mã RSA, từ đó từ đó đưa ra các giải pháp nhằm cài đặt hệ mã an toàn.

## 5.1 Tổng quan về hệ mã RSA

### 5.1.1 Giới thiệu

Tin học ngày càng đóng vai trò quan trọng trong nhiều hoạt động của con người, đặc biệt là trong lĩnh vực bảo mật thông tin. Nhằm đảm bảo tính xác thực, tính tin cậy của thông tin được truyền trên mạng, các vấn đề bảo mật thông tin trong quá trình trao đổi và lưu trữ dữ liệu là hết sức quan trọng, đồng nghĩa với việc thông tin phải được bảo vệ bởi một hệ mã an toàn.

Các hệ mã đối xứng ban đầu đã làm tốt nhiệm vụ bảo mật thông tin. Các hệ mã này đều sử dụng chung một khóa bí mật trong cả hai quy trình mã hóa – giải mã và vì thế việc bảo mật thông tin đồng nghĩa với việc bảo mật khóa chung đó. Tuy nhiên, nếu trong hệ thống có nhiều nhóm người cần trao đổi thông tin mật với nhau thì số khóa chung cần giữ bí mật là rất lớn, khó có thể quản lý và trao đổi. Hơn nữa, nếu khóa này bị lộ, sẽ có rất nhiều người sử dụng chung khóa đó bị ảnh hưởng.

Các hệ mã bất đối xứng ra đời đã giải quyết được việc đó bằng cách sử dụng hai loại khóa trong cùng một cặp khóa, khóa bí mật và khóa công khai. Khóa công khai được công bố rộng rãi và được sử dụng để mã hóa thông tin còn khóa bí mật chỉ do một người nắm giữ và được sử dụng để giải mã thông tin đã được mã hóa bằng khóa công khai. Đặc điểm quan trọng là không thể tìm được khóa giải mã khi chỉ biết khóa lập mã trong thời gian chấp nhận được. Hơn nữa, nếu một người bị lộ khóa giải mã của mình cũng không ảnh hưởng đến các người khác.

Thuật toán mã hóa bất đối xứng phổ biến nhất hiện nay là RSA [48]. Thuật toán này do 3 nhà toán học Ron Rivest, Adi Shamir và Len Adleman, mô tả lần đầu tiên vào năm 1977 tại học viện Công nghệ Massachusetts (MIT) và được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm 1983. Thuật toán ban đầu không được sử dụng rộng rãi do khả năng tính toán chậm tại thời điểm đó nhưng hiện nay đã được ứng dụng trong rất nhiều lĩnh vực như chữ ký điện tử và chứng thực sử dụng trong các hệ thống ngân hàng, thương mại điện tử, chính phủ điện tử.

### 5.1.2 Thuật toán

Thuật toán RSA được mô tả như sau:

- (1) Chọn 2 số nguyên tố lớn phân biệt  $p$  và  $q$ .
- (2) Tính  $n = pq$  và  $\varphi(n) = (p - 1)(q - 1)$ .
- (3) Chọn ngẫu nhiên một số nguyên  $e$  ( $1 < e < \varphi$ ) sao cho  $\gcd(e, \varphi) = 1$ .
- (4) Tính  $d$  sao cho  $de \equiv 1 \pmod{\varphi(n)}$ .
- (5) Công bố  $(n, e)$  và giữ bí mật  $(p, q, d)$ .

Trong đó,  $n$  được gọi là *modulo*,  $e$  là số mũ công khai (hay còn gọi là số mũ mã hóa) và  $d$  là số mũ bí mật (hay còn gọi là số mũ giải mã). Bộ  $(n, e)$  gọi là khóa công khai còn  $(n, d)$  gọi là khóa bí mật. Cặp khóa này có tính chất đối xứng, tức là khóa này dùng để mã và khóa kia dùng để giải mã và ngược lại.

Hai quá trình mã hóa và giải mã tương tự nhau. Cụ thể như sau:

- **Mã hóa:** B muốn gửi thông điệp  $M$  cho A, B sẽ chuyển  $M$  thành  $m < n$  bằng hàm hai chiều nào đó đã thỏa thuận trước với A. B đã biết  $n$  và  $e$  (do A gửi) nên B sẽ tính  $c = m^e \pmod{n}$  rồi gửi  $c$  này cho A.
- **Giải mã:** Khi nhận được  $c$  từ B và do biết khóa bí mật  $d$ , A sẽ tính  $m = c^d \pmod{n}$ . Sử dụng hàm hai chiều đã thỏa thuận với B, A sẽ tìm được  $M$  từ  $m$  tính được ở trên.

Dễ nhận thấy, các phép toán chính được sử dụng trong hệ mã RSA là phép nhân và phép lũy thừa *modulo*. Chính vì vậy tốc độ mã hóa của RSA rất chậm, gấp khoảng 1000 lần so với tốc độ mã hóa của các hệ mã đối xứng.

### 5.1.3 Các ứng dụng quan trọng

Việc phát minh ra hệ mã RSA thực sự là một cuộc cách mạng trong công nghệ an toàn thông tin điện tử do nó đã loại bỏ rắc rối trong việc phân phối khóa dùng chung cho mã hóa và giải mã trong hệ mã đối xứng. Hệ mã RSA sử dụng cặp khóa: khóa công khai cùng để mã hóa còn khoá bí mật dùng để giải mã. Tính linh hoạt này mang đến cho RSA rất nhiều ứng dụng quan trọng, đặc biệt như sau:

- **Tạo vỏ bọc an toàn cho văn bản:** tốc độ mã hóa của RSA rất chậm nên RSA thường không được sử dụng để mã hóa văn bản lớn. Do đó, RSA thường được sử dụng kết hợp với các hệ mã đối xứng có tốc độ cao như DES, AES, IDEA,  
... để tạo vỏ bọc an toàn cho văn bản. Các hệ mã đối xứng sẽ mã hóa văn bản bằng khóa bí mật nào đó còn RSA sẽ mã hóa chìa khóa bí mật này. Như vậy, các hệ mã đối xứng đã khắc phục được tốc độ mã hóa chậm chạp của RSA còn RSA đã khắc phục được vấn đề khó khăn trong chuyển giao khoá cho người nhận của các hệ mã đối xứng.
- **Xác nhận chủ thể:** các khóa lập mã được công bố công khai nên không thể tránh được trường hợp một cá thể này mạo danh một cá thể khác để gửi thông điệp cho một cá thể thứ ba. Nói cách khác, làm sao có thể “ký tên” dưới các thông điệp điện tử để người nhận biết đích xác mình nhận thông điệp của ai và để người gửi không thoái thác trách nhiệm về thông điệp mà mình đã gửi đi. Tóm lại, RSA nói riêng và phương pháp mã hóa khóa công khai nói chung đã mang lại một công cụ hiệu quả để “ký văn bản điện tử”, trong đó việc ký văn bản đồng nghĩa với việc mã hóa bằng khóa bí mật của chính cá thể đó (đã trình bày ở Chương 2).

Ngoài ra, RSA còn được sử dụng trong các giao thức bảo mật như bảo mật dữ liệu IP (IPSEC/IKE), bảo mật truyền dữ liệu (TLS/SSL), bảo mật thư điện tử (PGP), bảo mật kết nối đầu cuối (SSH), bảo mật dịch vụ hội nghị (SILC), ...

## 5.2 Nguy cơ tồn thaqong của hệ mă traqorc các tǎn công và cách khắc phục

Hệ mă RSA đưốc xây dựng dựa trên cơ sở mă mū và tận dụng tính khó giải của bài toán phân tích một số lớn ra thừa số nguyên tố xem như không thể thực hiện đưốc (trong thời gian chấp nhận). Thời gian cần thiết để phân tích một số nguyên  $n$  ra thừa số nguyên tố bằng thuật toán nhanh nhất hiện nay trên máy tính có tốc độ  $10^5$  phép tính/giây cũng vô cùng lâu [1, tr.5-6]:

**Bảng 5.1. Thời gian phân tích ra thừa số nguyên tố của một số lớn**

| Số chữ số thập phân | Số phép tính bit    | Thời gian               |
|---------------------|---------------------|-------------------------|
| 50                  | $1,4 \cdot 10^{10}$ | 3,9 giờ                 |
| 75                  | $9,0 \cdot 10^{12}$ | 104 ngày                |
| 100                 | $2,3 \cdot 10^{15}$ | 74 năm                  |
| 200                 | $1,2 \cdot 10^{23}$ | $3,8 \cdot 10^9$ năm    |
| 300                 | $1,5 \cdot 10^{29}$ | $4,9 \cdot 10^{15}$ năm |
| 500                 | $1,3 \cdot 10^{39}$ | $4,2 \cdot 10^{23}$ năm |

Như vậy, khi ta chọn các chữ số  $p$  và  $q$  khoảng 100 chữ số thập phân, thì  $n$  sẽ có khoảng 200 chữ số thập phân. Để phân tích một số nguyên lớn như thế với các thuật toán nhanh nhất hiện nay và với những máy tính hiện đại nhất, ta mất hàng tỷ năm!

Để rút ngắn thời gian, người có thể huy động nhiều máy tính để phân tích một số  $n$  cho trước. Một ví dụ điển hình là số  $n$  dài 128 chữ số thập phân đã bị phân tích vào ngày 26/4/1994 bằng một cỗ găng tổng lực mang tính quốc tế (qua Internet) với việc sử dụng 1600 workstation, mainframe và supercomputer trong 8 tháng liên tục.

Do đó, thực tế cho thấy hệ răng thuật hệ mă khóa công khai RSA là rất an toàn vì không măy khi có điều kiện để huy động một lực lượng tính toán hung hậu như thế. Ngoài ra, do tính đơn giản trong thiết kế và triển khai, RSA đưốc sử dụng rộng rãi và có lẽ là đưốc dùng nhiều nhất trong số các thuật toán với khóa công khai. Cũng chính vì vậy, nó đã trải qua nhiều cuộc thử thách, xem xét, khảo sát kỹ lưỡng của cộng đồng và đã có đưốc nhiều bằng chứng kiểm nghiệm về tính an toàn của nó.

Tuy nhiên, với thời gian tồn tại hơn 30 năm trên vai trò một hệ mã công khai thông dụng nhất, RSA đã phải đổi mặt với các khảo sát kỹ lưỡng dưới các kiểu tấn công đủ loại của giới thám mã chuyên nghiệp và thực tế cho thấy RSA có thể bị bẻ nếu người ta không biết sử dụng nó một cách bài bản [11].

Phần dưới đây trình bày chi tiết một số tấn công phổ biến và cách khắc phục.

### **5.2.1 Tổn thương do các tấn công phân tích ra thừa số nguyên tố**

Mặc dù mã hóa công khai ra đời đã giải quyết được các hạn chế của mã hóa bí mật nhưng do việc phổ biến rộng rãi khóa công khai nên cũng không tránh khỏi việc bị người khác tìm cách phân tích nhằm kiểm soát được các thông tin mật. Hệ mã công khai phổ biến RSA chủ yếu khai thác bài toán phân tích số  $n$  ra thừa số nguyên tố và xem như việc giải bài toán này là không thể thực hiện được khi  $n$  lớn trong khoảng thời gian chấp nhận. Các thuật toán phân tích ra thừa số nguyên tố có thể được chia thành hai nhóm:

- **Nhóm các toán phân tích biệt (special purpose):** sử dụng hiệu các thuật toán này phụ thuộc vào các thừa số nguyên tố chưa biết, rất tốt khi các thừa số nguyên tố được chọn để lập hệ mã là nhỏ. Nhóm này bao gồm phương pháp chia thử, phương pháp  $p-1$  và "rational" của Pollard, phương pháp  $p+1$  của Williams và đặc biệt nhất trong nhóm này là phương pháp đường cong elliptic (Elliptic Curve Method – ECM) của Lenstra.
- **Nhóm các toán phân tích tổng quát (general purpose):** sử dụng hiệu của nhóm này phụ thuộc vào chính số cần phân tích. Phương pháp phân tích tốt nhất nay là phương pháp sàng trường số tổng quát (General Number Field Sieve – GNFS). Trước đó, phương pháp phân tích tổng quát được sử dụng rộng rãi nhất là phương pháp sàng toàn phương (Quadratic Sieve – QS) và các biến thể của nó.

#### **5.2.1.1 Các phương pháp phân tích đặc biệt**

Hệ mã RSA trong một số trường hợp cũng không quá khó để phân tích do việc phát sinh khóa rơi vào các trường hợp dễ phân tích và với sự hỗ trợ của các hệ thống máy

tính hiện đại. Trên thực tế, có khá nhiều phương pháp tân công phân tích hệ mã RSA được đề xuất tỏ ra hiệu quả khi các thành phần tạo mã rơi vào các trường hợp đặc biệt.

- ***Phương pháp chia thử (Trial division).*** Đây là thuật toán phân tích thành thừa số cổ điển, tự nhiên và dễ hiểu nhất, bao gồm việc kiểm tra mỗi số nguyên tố nhỏ hơn hay bằng căn bậc hai của số cần phân tích. Phương pháp này chỉ hiệu quả khi số cần phân tích có các thừa số nhỏ.
- ***Phương pháp phân tích của Fermat và R.Sherman Lehman (1974).*** Hai phương pháp này cố phân tích một số bằng cách biểu diễn chúng dưới dạng hiệu của hai số chính phương. Những phân tích này sẽ thành công khi khoảng cách giữa hai số nguyên tố tạo nên nó là rất nhỏ, hoặc khi tỷ lệ của chúng gần với tỷ lệ của hai số nguyên nhỏ.
- ***Phương pháp phân tích  $p - 1$  của John Pollard (1974) [43].*** Phương pháp này hiệu quả khi số  $n$  cần phân tích có các thừa số nguyên tố  $p$  có dạng  $p - 1$  là mịn, nghĩa là  $p - 1$  chỉ chứa các thừa số nhỏ. Phương pháp này có độ phức tạp  $O(p^{\frac{1}{2}})$  với  $p$  là thừa số nguyên tố lớn nhất của  $p - 1$ .
- ***Phương pháp “rho” của John Pollard (1975) [44].*** Dựa trên thuật toán tìm chu trình của Floyd và lý thuyết xác suất cho biết rằng nếu ta chọn ngẫu nhiên một số trong tập có  $n$  số thì gần như chắc chắn là không quá  $\frac{6}{5} \sqrt{n}$  lần chọn ta sẽ nhận được số mà ta nhận được ở những lần chọn trước đó. Phương pháp này hiệu quả khi số cần phân tích có các thừa số nhỏ. Phương pháp này có độ phức tạp  $O(\bar{p})$  với  $p$  là thừa số nguyên tố lớn nhất của  $n$ . Năm 1980, Richard P. Brent công bố một biến thể nhanh hơn của thuật toán này do sử dụng thuật toán khác thay thế thuật toán phát hiện chu trình của Floyd [13].
- ***Phương pháp phân tích  $p + 1$  của H. C. Williams (1982) [63].*** Phương pháp này hiệu quả khi số cần phân tích có các thừa số nguyên tố  $p$  có dạng  $p + 1$  là mịn, nghĩa là  $p + 1$  chỉ chứa các thừa số nhỏ. Phương pháp này có độ phức tạp  $O(p^{\frac{1}{2}})$  với  $p$  là thừa số nguyên tố lớn nhất của  $p + 1$ .
- ***Phương pháp đường cong Elliptic (ECM) của H.W Lenstra Jr. (1985) [37].*** Các phương pháp trên mất rất nhiều thời gian tăng theo cấp số mũ của chiều dài theo bit của  $p$ , các thừa số mà chúng tìm thấy rất chậm. Phương pháp này

cao cấp hơn chúng, độ phức tạp của phương pháp này là  $O(e^{2\sqrt{\ln p \ln \ln p}})$ .

Phương pháp này thường hiệu quả khi thừa số bé của  $n$  chỉ có khoảng từ 13 đến 47 chữ số còn thừa số lớn thì lại có thể lớn hơn rất nhiều. Thừa số lớn nhất (có 67 chữ số) được tìm thấy bằng ECM vào 24/8/2006 bởi B. Dodson.

Các phương pháp trên thường sử dụng trong tể để tìm các thừa số của các số

được phát sinh cách nhiên, có các thừa số nguyên tố được chọn mịn (có một ngẫu

các thừa số nguyên tố nhỏ). Trước các phương pháp phân tích trường hợp đặc biệt đó, hàng loạt các đề xuất liên quan đến số nguyên tố được chọn để lập mã có một số tính chất đặc biệt nhằm giảm thiểu cơ hội thành công của các phương pháp phân tích này. Những số nguyên tố có những tính chất đặc biệt đó được gọi là số nguyên tố mạnh (sẽ được trình bày ở Chương 6).

### 5.2.1.2 Phương pháp phân tích tổng quát

Các phương pháp phân tích đặc biệt ở trên không đủ nhanh để phân tích các *modulo* lớn sử dụng trong các hệ mã RSA. Hiện nay, các phương pháp phân tích tổng

quát như sàng toàn phương (Quadratic Sieve – QS) và sàng trường số tổng quát (General Number Field Sieve – GNFS) đã dần thay thế các phương pháp phân tích đặc biệt trên. Đây là các phương pháp phân tích tổng quát do sự hiệu quả của các phương pháp này chỉ phụ thuộc vào kích thước của số cần phân tích chứ không phụ thuộc các tính chất đặc biệt của nó.

- **Phương pháp sàng toàn phương (QS) của Carl Pomerance (1981)** [45].

Đây là phương pháp nhanh nhất được biết đến để phân tích các số nhỏ hơn 110 chữ số thập phân và được sử dụng rất rộng rãi. Phiên bản nhanh hơn của thuật toán này được gọi là the Multiple Polynomial Quadratic Sieve [53].

Biến thể nhanh

nhất của QS có thời gian thực hiện là  $e^{1+O(\frac{1}{\log n} \cdot \frac{1}{2(\log \log n)^2})}$ .

- **Sàng trường số tổng quát (GNFS)** [38]. Đây là thuật toán phân tích thành thừa số nhanh nhất được biết đến để phân tích các số lớn hơn 110 chữ số, chính là những số được dùng phổ biến trong RSA hiện nay. Nó không thực tế khi được đề xuất nhưng đã dần thay đổi qua hàng loạt các cải tiến trong



năm gần đây. Phiên bản đầu tiên được sử dụng để phân tích số Fermat thứ 9 là  $2^{512} + 1$ . Thuật toán GNFS nhanh hơn thuật toán QS rất nhiều, với độ phức tạp là  $e^{\frac{1}{1.923+O(\log n)}} \cdot \log n^{\frac{2}{3}} (\log \log n)^3$ .

Vào tháng 3 năm 1991, RSA Data Security, Inc. đã thiết lập cuộc thi phân tích RSA (RSA Factoring Challenge). Cuộc thi bao gồm danh sách các số “khó”<sup>16</sup>, mỗi số là tích của hai số nguyên tố có kích thước xấp xỉ nhau. Có 42 số trong cuộc thi, có độ dài từ 100 đến 500 chữ số, số này cách số kia 10 chữ số (có thêm một số 129 chữ số).

Hiện nay, RSA-100, RSA-110, RSA-120, và RSA-129 đã được phân tích và đều bằng QS. RSA-129 được phân tích vào ngày 2/4/1994, là số dài nhất được công bố sử dụng phương pháp QS cho đến khi NFS phân tích thành công RSA-130 vào ngày 10/4/1996. Tất cả các RSA cho đến bây giờ đều được phân tích bởi NFS.

Các cải tiến gần đây trong NFS làm cho NFS hiệu quả hơn MPQS trong việc phân tích các số lớn hơn kh oảng 115 chữ số, trong khi MPQS tốt hơn cho các số nguyên

nhỏ. Trong khi RSA-129 (129 chữ số) bị phân tích sử dụng một biến thể tháp của MPQS, còn một biến thể của NFS đã sử dụng gần đây để phân tích RSA-155. Như vậy, ước tính nếu NFS được sử dụng để phân tích RSA-129, nó sẽ chỉ cần

một phần tử thời gian mà MPQS đã mất. Có thể nói, NFS đã qua MPQS như là vượt

thuật toán phân tích sử dụng rộng rãi nhất

Vào ngày 9/5/2005, RSA-200 (200 chữ số thập phân tương ứng với 663bit) đã bị phân tích bằng phương pháp GNFS do F. Bahr, M. Boehm, J. Franke và T. Kleinjung thực hiện. Theo chuyên gia Arjen Lenstra của tổ chức EPFL Thụy Sĩ (Ecole Polytechnique Fédérale de Lausanne) thì khả năng phá khóa RSA-1024 sẽ chỉ đạt được trong 5 – 10 năm nữa, nhưng đã đến lúc tìm kiếm giải pháp bảo mật mạnh mẽ hơn.

Để tăng độ an toàn của hệ mã trước các phương pháp này, độ dài khóa (*modulo n*) được chọn ngày càng lớn hơn. Hiện nay, độ dài *modulo n* tối thiểu là 1024 bit.

<sup>16</sup> Số “khó” là số không có bất kỳ thừa số nguyên tố nào nhỏ và không thuộc dạng đặc biệt nào để bị phân tích dễ dàng.

## 5.2.2 Tổn thương do bản thân hệ mã

### 5.2.2.1 Các thông điệp không có tính che dấu

Một số thông điệp không có tính che dấu. Với *modulo n* bất kỳ, 3 thông điệp  $m = 0$ ,  $m = 1$  và  $m = n - 1$  là các thông điệp dạng đó vì  $m^e \bmod n = m$ . Ngoài ra, với *modulo n* cụ thể, sẽ có rất nhiều giá trị  $m$  như thế và cần được lưu ý.

Ví dụ:

Chọn  $p = 11$  và  $q = 3 \Rightarrow n = pq = 33$ ,  $\varphi = p - 1 - q - 1 = 10 \times 2 = 20$ .

Chọn  $e = 3$  (thỏa do  $\gcd(e, \varphi) = 1$  và  $\gcd(e, q - 1) = 1$ )

$ed \equiv 1 \pmod{\varphi} \Rightarrow d = 7$ .

Khóa công khai là  $(n, e) = (33, 3)$ .

Khóa bí mật là  $(n, d) = (33, 7)$ .

Với cái thông điệp  $m$ ,  $0 \leq m < n = 33$ , ta tính được các bản mã  $c$  như sau và nhận thấy có đến thêm 6 giá trị  $m$  không có khả năng che dấu thông tin.

|          |          |          |    |    |           |           |           |    |    |    |           |           |           |    |    |           |    |
|----------|----------|----------|----|----|-----------|-----------|-----------|----|----|----|-----------|-----------|-----------|----|----|-----------|----|
| <b>m</b> | <b>0</b> | <b>1</b> | 2  | 3  | 4         | 5         | 6         | 7  | 8  | 9  | <u>10</u> | <u>11</u> | <u>12</u> | 13 | 14 | 15        | 16 |
| <b>c</b> | <b>0</b> | <b>1</b> | 8  | 27 | 31        | 26        | 18        | 13 | 17 | 3  | <u>10</u> | <u>11</u> | <u>12</u> | 19 | 5  | 9         | 4  |
| <b>m</b> | 17       | 18       | 19 | 20 | <u>21</u> | <u>22</u> | <u>23</u> | 24 | 25 | 26 | 27        | 28        | 29        | 30 | 31 | <u>32</u> |    |
| <b>c</b> | 29       | 24       | 28 | 14 | <u>21</u> | <u>22</u> | <u>23</u> | 30 | 16 | 20 | 15        | 7         | 2         | 6  | 25 | <u>32</u> |    |

Các giá trị  $m = 0$ ,  $m = 1$  và  $m = n - 1$  sẽ luôn luông không có khả năng che dấu thông tin với mọi  $n$  dù lớn hay nhỏ. Nhưng trong thực tế, các giá trị lớn sẽ không gặp vấn đề này khi chúng ta sử dụng các giá trị lớn hơn của  $n$  (hơn vài trăm bit).

### 5.2.2.2 Tấn công lặp

Simmons và Norris sớm đề xuất một tấn công trên RSA gọi là siêu mã hóa superencryption) hay tấn công lặp (cycling), được dựa trên sự quan sát rằng qua nhiều lần mã hóa liên tục có thể tìm lại được thông điệp ban đầu [54]. Hệ thống RSA có thể

bị tổn thương khi sử dụng tấn công lặp liên tiếp khi đối thủ biết cặp khóa công cộng  $(n, e)$  và bản mã c thì có thể tính chuỗi các bản mã sau:

$$c_1 = c^e \pmod{n}$$

$$c_2 = c_1^e \pmod{n}$$

...

$$c_i = c_{i-1}^e \pmod{n}$$

Nếu có một phần tử  $c_j$  trong chuỗi  $c_1, c_2, \dots, c_i$  sao cho  $c_j = c$  thì khi đó sẽ tìm được thông điệp gốc  $m = c_{j-1}$  vì:

$$c_j = c_{j-1}^e \pmod{n}$$

$$c = m^e \pmod{n}$$

Ví dụ, giả sử biết  $(n, e) = (35, 17)$  và  $c = 3$ , ta sẽ tính:

$$c_1 = c^e \pmod{n} = 3^{17} \pmod{35} = 33$$

$$c_2 = c_1^e \pmod{n} = 33^{17} \pmod{35} = 3$$

Vì  $c_2 = c$  nên  $m = c_1 = 33$ .

Tấn công này đe dọa sự an toàn của RSA với điều kiện số lần mã hóa cần thiết là nhỏ. Tuy nhiên, đây không phải là một tấn công khả thi trong thực tế nếu các số nguyên tố được chọn là lớn.

### 5.2.3 Tổn thương do lạm dụng hệ mã

Đây là các tấn công cơ bản và cũ. Những tấn công này thực được do việc lạm dụng và dùng sai hệ mã RSA.

#### 5.2.3.1 Tấn công modulo chung

Để tránh phát sinh lại các *modulo n* khác nhau cho mỗi người sử dụng, người ta có thể sử dụng cùng *modulo n* cho mọi người. Lúc này, tổ chức chúng có thể nhân

cung cấp cho người sử dụng thứ  $i$  duy nhất  $e_i, d_i$ , từ đó người sử dụng sẽ có khóa

công khai là  $n, e_i$  và khóa bí mật  $n, d_i$ . Điều này rất lợ do không phải mất i

thời gian tính lại *modulo n* nhưng kết quả là hệ thống không an toàn.

Theo Boneh, với  $n, e$  là một khóa công khai RSA, cho trước một khóa bí mật  $d$ ,

người ta có khả năng phân tích *modulo n = pq*. Ngược lại, cho trước các thừa số của  $n$ , người ta có thể tìm lại được  $d$  [11].

Như vậy, B có thể sử dụng lũy thừa  $e_B, d_B$  của chính anh ta để phân tích *modulo n*.

Một khi  $n$  đã bị phân tích, B có thể tìm lại khóa bí mật  $d_A$  từ khóa công khai  $e_A$  được

của A hay một người nào khác sử dụng chung *modulo n* này.

### 5.2.3.2 Tấn công bản mã~được chọn lựa

Hệ mã RSA rất hay được sử dụng trong chữ ký số vì nó đơn giản trong thiết lập nhưng lại có độ an toàn cao. Tuy nhiên, việc ký bừa bãi những văn bản không rõ nguồn gốc có thể bị lợi dụng và gây ra những hậu quả nghiêm trọng [11].

Giả sử như B muốn A ký vào thông điệp  $m$  (có thể gây thiệt hại cho A), tức là B muốn có  $s = m^{d_A} \text{ mod } n$ . Tất nhiên A sẽ từ chối thực hiện công việc này. B có thể thực hiện các bước sau để có được chữ ký của A trên thông điệp  $m$ :

- Đầu tiên, B chọn một giá trị  $x$  tùy ý và tính  $y = x^{e_A} \text{ mod } n$ ,  $e_A$  là khóa công khai của A.
- Sau đó B tính  $m' = ym \text{ mod } n$  và đề nghị A ký vào thông điệp ngẫu nhiên  $m'$  này.
- A sẵn sàng ký vào thông điệp trông “vô nghĩa” này và đưa B chữ ký  $s' = m'^{d_A} \text{ mod } n$ .
- Bây giờ B tính giá trị  $s' x^{-1} \text{ mod } n$  sẽ thu được chữ ký  $s$  trên thông điệp  $m$ :

$$\begin{aligned}s' x^{-1} \text{ mod } n &= m'^{d_A} x^{-1} \text{ mod } n \\&= ym^{d_A} x^{-1} \text{ mod } n = x^{e_A} m^{d_A} x^{-1} \text{ mod } n \\&= x^{e_A d_A} x^{-1} m^{d_A} \text{ mod } n = m^{d_A} \text{ mod } n\end{aligned}$$

Kỹ thuật này gọi là ký mù (blinding).

Để tránh tấn công này, lưu ý không bao giờ được sử dụng RSA để ký một văn bản tùy ý đưa tới bởi một người lạ. Tuy nhiên, do mọi mô hình chữ ký số đều sử dụng hàm băm một chiều trên thông điệp  $m$  trước khi ký nên tấn công này không còn là vấn đề đáng lo ngại.

#### **5.2.4 Tổn thương do sử dụng số mũ bí mật nhỏ**

Wiener đề xuất một tấn công trên hệ mã RSA bằng một xấp xỉ phân số liên tục, sử dụng khóa công khai  $n, e$  để cung cấp các thông tin cần thiết nhằm tìm số mũ bí mật  $d$  [61]. Tấn công này hiệu quả, khả thi và được quan tâm chỉ khi số mũ bí mật  $d$  được chọn là bé so với *modulo n*, chính xác hơn chỉ nếu  $d < n^{1/4}$  (hay chiều dài của  $d$  ngắn hơn  $\frac{1}{4}$  chiều dài của  $n$  theo bit). Tuy nhiên, do số mũ  $e$  được chọn trước, điều đó không chắc chắn một  $d$  nhỏ được tạo ra. Nếu  $e$  đủ nhỏ thì  $d$  sẽ đủ lớn để chống lại tấn công này.

Do các hệ thống hiện nay đều sử dụng  $n$  có độ dài trên 1024, điều đó dẫn đến  $d$  phải có ít nhất 256 bit để tránh tấn công này. Điều này hoàn toàn làm được vì số mũ  $e$  thường được chọn là nhỏ và dẫn đến số mũ  $d$  có độ dài xấp xỉ  $n$ . Tuy nhiên, trên các hệ thống giới hạn khả năng xử lý và lưu trữ như thẻ thông minh thì kích thước khóa lớn trở thành một gánh nặng.

#### **5.2.5 Tổn thương do sử dụng số mũ công khai nhỏ**

Để giảm thời gian mã hóa và xác nhận chữ ký có thể sử dụng số mũ công khai  $e$  nhỏ,

Giá trị ưa thích của  $e$  được sử dụng trong quá khứ là  $e = 3$ . Tuy nhiên, điều này không nên được thực hiện do nó đã làm cho một số tấn công có thể thực hiện được.

Giá trị  $e$  được đề nghị sử dụng là  $e = 2^{16} + 1 = 65537$ . Cũng giống như giá trị  $e = 3$ , giá trị  $e$  được đề nghị này chỉ có 2 bit 1 nên thời gian mã hóa và xác nhận chữ ký cũng rất nhanh (cần 17 phép nhân, ít hơn hẳn so với khoảng 1000 phép nhân khi  $e$  được chọn ngẫu nhiên) nhưng lại mang đến độ an toàn cao hơn rất nhiều.

### 5.2.5.1 Tấn công loan truyền tin của Hastad

Giả sử một người muốn gửi cùng một bản mã của cùng một thông điệp  $m$  đến các tổ chức khác nhau, Hastad cho thấy điều này rất nguy hiểm khi kẻ tấn công có được các bản mã này thì có thể tìm được  $m$  [28].

Ví dụ, nếu ta muốn gửi một thông điệp  $m$  đến 3 tổ chức cùng sử dụng chung số mũ công khai  $e = 3$ , với các *modulo*  $n_1, n_2, n_3$  khác nhau, một người có thể dễ dàng tìm được  $m$  từ 3 bản mã như sau:

$$c_1 = m^3 \bmod n_1$$

$$c_2 = m^3 \bmod n_2$$

$$c_3 = m^3 \bmod n_3$$

Nhận xét, thông điệp  $m$  phải bé hơn các *modulo*, và do đó  $m^3$  sẽ bé hơn  $n_1n_2n_3$ . Sử dụng định lý số dư Trung Hoa, người ta có thể tính kết quả duy nhất:

$$c = m^3 \bmod n_1n_2n_3$$

Do đó, người ta có thể tính  $m$  bằng cách lấy căn bậc ba của  $c$ .

Tổng quát hơn, nếu các số mũ công khai đều bằng  $e$ , một người có thể tìm được  $m$  khi  $k \geq e$ . Do đó tấn công này chỉ có thể thực hiện được khi  $e$  nhỏ.

Giá trị đề nghị của  $e$  thường được sử dụng ngày nay là  $e = 2^{16} + 1$ . Một ưu điểm của giá trị  $e$  này là chỉ có 2 bit được bật, nghĩa là thuật toán bình phương và nhân đòi hỏi rất ít thao tác, do đó nó rất hiệu quả. Cụ thể, khi giá trị này được sử dụng, xác

nhận ký chỉ cần 17 phép nhân so với khoảng 1000 phép nhân khi giá trị ngẫu

nhiên  $e < (p-1)(q-1)$  được chọn. Ngoài ra, với việc chọn  $e$  như vậy, việc kiểm tra  $\gcd(e, p-1) = 1$  và  $\gcd(e, q-1) = 1$  dễ dàng hơn khi phát sinh và kiểm tra các số nguyên tố lặp mã.

Tấn công trên có thể khắc phục bằng một cách đơn giản bằng cách đếm thêm giá trị  $i$  trước khi gửi đến tổ chức thứ  $i$ . Lúc này kẻ tấn công có được bản mã của các thông điệp khác nhau nên không thể nào tìm được  $m$ .

Tuy nhiên, bằng một biến thể mạnh hơn của tấn công trên, Hastad cho thấy nếu các phần đệm tuyếntính được thêm vào vẫn không thể chống được tấn công này.

Tổng quát hơn, giả sử  $k$  bản rõ liên quanđược mã hóa bởi cùng số mũ  $e$ :

$$c_1 = a_1 m + b_1 \text{ mod } n_1$$

$$c_2 = a_2 m + b_2 \text{ mod } n_2$$

...

$$c_k = a_k m + b_k \text{ mod } n_k$$

trong đó  $a_i$  và  $b_i$ ,  $1 \leq i \leq k$  biết trước và  $k > \frac{e(e+1)}{2}$  và  $\min_i n_i > 2^{e^2}$ . Khi đó, kẻ

tấn công có thể tìm  $m$  với thời gian đa thức sử dụng kỹ thuật rút gọn giàn. Khảo sát này dựa trên Johan Håstad [28].

Để tránh gặp phải tấn công này, đệm vào các thông điệp với các chuỗi (giả) ngẫu nhiên trước khi mã hóa. Nếu các thông điệp liên quan nhau trong một cách biết trước, chúng không nên được mã hóa với các khóa RSA khác nhau.

#### **5.2.5.2 Tấn công thông điệp liên quan của Franklin-Reiter**

Giả sử B gửi cho A hai bản mã  $C_1, C_2$  của hai thông điệp  $M_1, M_2$  thuộc  $\mathbb{Z}_n^*$  với  $M_1 = f(M_2) \text{ mod } n$  với  $f$  là một đa thức tuyếntính dạng  $f(x) = ax + b$ , ( $b \neq 0$ ), Franklin và Reiter cho thấy, khi có được  $C_1$  và  $C_2$ , một người có thể tìm được  $M_1$  và  $M_2$  [11].

Tuy nhiên, tấn công này tốn thời gian tỷ lệ với bình phương của  $e$ . Do đó, nó có thể được áp dụng khi số mũ công khai  $e$  nhỏ được sử dụng.

#### **5.2.5.3 Tấn công phần đệm thêm ngắn của Coppersmith**

Tấn công của Franklin-Reiter có một điểm không tự nhiên, đó là tại sao B lại gửi cho A bản mã của các thông điệp có liên quan như vậy? Coppermsith đã làm mạnh hơn tấn công đó và chứng minh được kết quả quan trọng của việc đệm thêm [17].

Một thuật toán đệm thêm ngẫu nhiên đơn giản là đệm thêm vào thông điệp  $m$  bằng cách nối một vài bit ngẫu nhiên vào cuối thông điệp. Tấn công sau đây cho thấy sự

nguy hiểm của việc đệm thêm đơn giản như vậy. Giả sử B gửi một bản mã được đệm thêm của  $m$  cho A. Kẻ tấn công C chặn đứng bản mã và ngăn nó đến được người nhận. B nhận thấy A không phản hồi thông điệp của anh ta và quyết định gửi lại  $m$  cho A. B lại đệm thêm một cách ngẫu nhiên vào  $m$  và gửi đi kết quả được mã hóa. Lúc này C có hai bản mã tương ứng với hai mã hóa của cùng một thông điệp sử dụng hai phần đệm ngẫu nhiên khác nhau. Định lý sau đây cho thấy mặc dù C không biết các phần đệm được sử dụng, C cũng có thể tìm được thông điệp gốc  $m$ .

Cho  $(N, e)$  là khóa RSA công khai trong đó  $N$  dài  $n$  bit. Đặt  $m = \frac{n}{e^2}$ . Gọi  $M \in \mathbb{Z}^*$

là thông điệp có độ dài tối đa  $n - m$  bit. Định nghĩa  $M_1 = 2^m M + r_1$  và  $M_2 = 2^m M + r_2$ , trong đó  $r_1$  và  $r_2$  là các số nguyên khác nhau với  $0 \leq r_1, r_2 < 2^m$ .

Nếu có  $(N, e)$  và bản mã  $C_1, C_2$  của  $M_1, M_2$  (nhưng không có  $r_1, r_2$ ), ta có thể tìm được  $M$  một cách hiệu quả [59].

Khi  $e = 3$ , tấn công có thể được thực hiện miễn là chiều dài phần đệm ngắn hơn  $\frac{1}{9}$  chiều dài thông điệp. Đây là một kết quả quan trọng. Lưu ý rằng, với giá trị được đề nghị của  $e = 65537$ , tấn công là vô ích trước các kích thước *modulo* lớn được sử dụng hiện nay.

#### 5.2.5.4 Tấn công từ một phần khóa

Gọi  $(n, d)$  là khóa bí mật. Giả sử A có được một phần dãy bit của  $d$  này, khoảng  $\frac{1}{4}$  dãy bit, A có thể tái tạo phần còn lại của  $d$  nếu như khóa công khai tương ứng là nhỏ. Gần đây, Boneh, Durfee và Frankel cho thấy rằng một tấn công tương tự có thể thực hiện với số mũ  $e$  lớn miễn là  $e < n$  thì có thể tái tạo lại toàn bộ dãy bit  $d$  chỉ từ một phần dãy bit của nó nhưng những kỹ thuật này phức tạp hơn [12].

Điều này cho thấy việc bảo vệ khóa bí mật là vô cùng quan trọng vì chỉ cần một phần khóa bị lộ thì nguy cơ toàn bộ khóa bị lộ là điều hoàn toàn có thể. Hơn nữa, tấn công này cho thấy khi số mũ mã hóa  $e$  nhỏ, hệ thống RSA để lộ một nửa các bit ý nghĩa

nhất của khóa bí mật  $d$  tương ứng. Do đó, giá trị  $e = 65537$  được đề nghị sử dụng thay cho giá trị  $e = 3$ .

### 5.2.6 Tổn thương do khai thác thời gian thực thi

Thay vì tấn công vào cấu trúc bên dưới của RSA, tấn công sau tập trung vào sự thực thi của hệ mã. Khi hệ mã được triển khai trên thẻ thông minh, kẻ tấn công không thể xem xét được nội dung bên trong thẻ thông minh để lấy được khóa. Tuy nhiên, một tấn công thông minh của Kocher cho thấy bằng cách đo chính xác thời gian thẻ thông minh thực hiện thao tác giải mã (hay ký), kẻ tấn công có thể nhanh chóng tìm được số mũ giải mã bí mật  $d$  [34].

Có hai cách để chống lại tấn công này. Cách đơn giản nhất là trì hoãn một lượng thời gian thích hợp sao cho phép lũy thừa *modulo* luôn cần một lượng thời gian cố định. Cách thứ hai theo Rivest là sử dụng ký mù (blinding). Nghĩa là, trước khi giải mã  $m$ , thẻ thông minh chọn một số ngẫu nhiên  $r \in \mathbb{Z}^*$  và tính  $m' = m \cdot r^e \bmod n$ . Sau đó nó sử dụng  $d$  trên  $m'$  và nhận được  $c' = m' \cdot d \bmod n$ . Cuối cùng, thẻ thông minh tính  $c = c' \cdot r^{-1} \bmod n$ . Với cách này, thẻ thông minh áp dụng  $d$  với thông điệp  $m'$  mà kẻ tấn công không biết và vì thế không thể thực hiện tấn công này được.

## 5.3 Kết luận

Hệ mã khóa công khai RSA là một hệ mã rất dễ hiểu, dễ triển khai và có thể ứng dụng trong rất nhiều lĩnh vực bảo mật mang đến độ an toàn cao. Với thời gian tồn tại hơn 30 năm trên vai trò một hệ mã công khai thông dụng nhất, RSA đã phải đổi mới với các khảo sát kỹ lưỡng dưới các kiểu tấn công đủ loại của giới thám mã chuyên nghiệp và thực tế cho thấy RSA có thể bị bẻ nếu người ta không biết sử dụng nó một cách bài bản.

Để đạt được bảo mật tối đa, cần tuân thủ các đề nghị sau đây:

- Phát sinh khóa là một phần quan trọng nhất của RSA. Nó cũng là phần khó khăn nhất của RSA để thực thi chính xác. Các số nguyên tố được chọn để lập mã phải thật sự là số nguyên tố hoặc là số giả nguyên tố với xác suất sai thấp

nếu không RSA sẽ không thể hoạt động hoặc không còn an toàn. Tồn tại rất hiếm các hợp số làm cho RSA hoạt động nhưng kết quả cuối cùng là không an toàn. Nên kiểm tra các khóa sau khi tạo bằng cách thực hiện một số thao tác mã hóa và giải mã RSA.

- Có thể chọn các số nguyên tố ngẫu nhiên để lập mã nhưng điều này có thể tạo điều kiện cho một số tấn công phân tích ra thừa số nguyên tố đặc biệt như rho và  $p - 1$  của Pollard,  $p + 1$  của Williams. Để tránh các tấn công đặc biệt này, các số nguyên tố mạnh nên được chọn để lập mã. Các số nguyên tố ngẫu nhiên và thời gian phát sinh chỉ lâu hơn một chút so với phát sinh số nguyên tố ngẫu nhiên nhưng mang lại sự bảo vệ trước các tấn công phân tích đặc biệt.
- Độ dài *modulo n* tối thiểu là 1024 bit để tránh các tấn công phân tích ra thừa số nguyên tố và tấn công lặp của Simmons và Norris. Nếu hệ thống cần bảo mật trong một thời gian dài, nếu sử dụng khóa có kích thước 2048 bit hoặc lớn hơn.
- Không sử dụng *modulo n* chung cho các cặp khóa khác nhau vì chỉ với cặp khóa của mình, một người có thể phân tích được  $n = pq$  và từ đó tìm được khóa của người khác.
- Không bao giờ được sử dụng RSA để ký một văn bản tùy ý được đưa tới bởi một người lạ và chỉ nên ký lên giá trị băm của thông điệp đó.
- Không nên chọn số mũ giải mã  $d$  nhỏ (cụ thể  $d < n^{1/4}$ ) để tránh tấn công của Wiener. Độ dài của  $d$  nên xấp xỉ độ dài *modulo n* (theo bit). Tuy nhiên, điều này chỉ đáng ngại trên các hệ thống sử dụng  $n$  nhỏ do giới hạn lưu trữ, xử lý.
- Không nên chọn số mũ mã hóa  $e$  nhỏ. Giá trị thường sử dụng là  $e = 3$  tuy giúp mã hóa nhanh nhưng rất nguyên hiến trước các tấn công loan truyền tin của Hastad, tấn công thông điệp liên quan của Franklin-Reiter, tấn công phần đệm ngắn của Coppersmith và tấn công phục hồi từ một phần khóa của Boneh. Giá trị được đề nghị sử dụng là  $e = 65537$  vừa có thể mã hóa nhanh (vì cũng chỉ có 2 bit 1 trong dãy bit) vừa chống được các tấn công trên.

- Nên đêm thêm chuỗi bit ngẫu nhiên trước khi mã hóa, như vậy kẻ tấn công sẽ không thể biết được nội dung thực sự của thông điệp. Tuy nhiên phần đệm thêm này không được ngắn (dưới  $\frac{1}{e^2}$  chiều dài thông điệp) nếu không tấn công của Coppersmith có thể sẽ thành công trong trường hợp  $e$  nhỏ.
- Nên thêm vào một khoảng thời gian trì hoãn thích hợp cho phép lũy thừa modulo luôn cần một lượng thời gian cố định hoặc sử dụng ký mù (blinding) để chống lại tấn công đo thời gian thực thi.
- Luôn phải giữ kín khóa bí mật  $d$  và không được để lộ dù chỉ một phần (cụ thể là  $\frac{1}{4}$  chiều dài theo bit), nếu không người khác có phục hồi toàn bộ khóa bí mật bằng tấn công của Boneh.
- Phép tính chính trong hệ mã RSA là phép lũy thừa *modulo*, phép toán tốn rất nhiều chi phí. Vì vậy cần cài đặt các thực thi hiệu quả của tính toán này. Ngoài ra, giải mã và ký thường chậm hơn mã hóa và xác nhận chữ ký nên có thể sử dụng kỹ thuật tính toán theo định lý số dư Trung hoa để tăng tốc quá trình này.

Với những phân tích các tấn công gây tổn thương ở trên, Chương 6 sẽ tiếp tục nghiên cứu và giải quyết một số bài toán quan trọng khi cài đặt hệ mã RSA nhằm đạt độ an toàn và hiệu quả.

## Chương 6

# Một số bài toán quan trọng trong hệ mã RSA

☐ Nội dung của chương này tập trung nghiên cứu các bài toán quan trọng cần giải quyết kết hợp với những cơ sở phân tích ở Chương 5 nhằm xây dựng hệ mã RSA an toàn và hiệu quả. Kết quả thử nghiệm của các thuật toán ở chương này sẽ lần lượt được trình bày ở Chương 7.

## 6.1 Nhu cầu

Các phân tích chi tiết ở Chương 5 cho thấy hệ mã khóa công khai RSA chỉ thật sự an toàn khi người ta thực hiện việc lập mã một cách bài bản, bao gồm việc chọn các tham số cho hệ mã ( $p, q, e, d$ ) phù hợp, quy trình thực hiện ký và giải mã hợp lý, ... Ngoài ra, tính hiệu quả cũng cần được quan tâm do chi phí thực hiện các công việc như tạo khóa, ký/ giải mã và xác nhận chữ ký/ mã hóa là rất lớn. Vì vậy, chương này tập trung nghiên cứu cách giải quyết các bài toán quan trọng trong hệ mã RSA nhằm đạt độ an toàn và hiệu quả khi triển khai trong thực tế.

## 6.2 Bài toán tính toán nhanh trên số lớn

Khi nói đến độ dài khóa của một khóa RSA là nói đến *modulo n* theo bit. Độ dài khóa được đề nghị nhỏ nhất cho một RSA an toàn hiện nay là 1024 bit. Để thực hiện các phép tính trên các số lớn như vậy đòi hỏi phải sử dụng các thuật toán tính toán hiệu quả do các thao tác chính phải thực hiện trong RSA chính là các phép lũy thừa

*modulo*, phép tính có chi phí rất cao.

Một phương pháp hiệu quả rất thường được sử dụng để thực hiện phép tính lũy thừa *modulo* trên máy tính là thuật toán nhị phân [2, tr. 40].

Để tính  $y = x^e \bmod n$ , ta triển khai y dưới dạng cơ số 2:

$$e = e_0 2^0 + e_1 2^1 + e_2 2^2 + \cdots + e_I 2^I = \sum_{i=0}^I e_i 2^i$$

Như vậy,  $y = x^e \ mod \ n = x \ e^0 \ x^2 \ e^1 \ x^4 \ e^2 \ \dots \ x^{2^i} \ e_i$ .

Dễ dàng nhận thấy,  $e_i$  ( $i = 0, 1, \dots, I$  với  $I$  là chiều theo bit của biểu diễn nhị phân của  $e$ ) có giá trị 0 hoặc 1 nên khi  $e_i = 0$  thì  $x^{2^i} = 1$  không làm thay đổi kết quả tổng thể còn  $e_i = 1$  thì  $x^{2^i} = x^{2^i}$  ta sẽ nhận kết quả trước cho giá trị này.

Hơn nữa giá trị  $x^{2^i}$  có thể tính bằng  $x^{2^{i-1}}^2$ , tức là chỉ bình phương số trước đó là ta có ngay kết quả mà không cần phải tính toán lại.

#### ModPowBinary(x, e, n)

**Đầu vào:** số nguyên  $x > 0$ , số mũ nguyên  $e \geq 0$ , số nguyên  $n > 0$

**Đầu ra:** giá trị  $x^e \ mod \ n$

- (1)  $y = 1$
- (2) **Nếu**  $e = 0$  **thì** trả về  $y$
- (2)  $A \leftarrow x$
- (4) **Nếu**  $e_0 = 0$  **thì**  $y \leftarrow x$
- (5) **Với mỗi**  $i = 1 \rightarrow I$ 
  - (5.1)  $A \leftarrow A^2 \ mod \ n$
  - (5.2) **Nếu**  $e_i = 1$  **thì**  $y \leftarrow A * y \ mod \ n$
- (6) Trả về  $y$

#### Thuật toán 6.1. Tính lũy thừa modulo bằng thuật toán nhị phân

Thời gian thực hiện phép lũy thừa modulo  $c = m^e \ mod \ n$  tăng theo số lượng bit 1 trong số mũ  $e$ . Với việc mã hóa, lựa chọn thích hợp của  $e$  có thể làm giảm chi phí tính toán. Những giá trị phổ biến được chọn là 3, 17, 65537 là các số nguyên chỉ có duy nhất 2 bit được bật là  $3 = 11_2$ ,  $17 = 11_{16}$ ,  $65537 = 10001_{16}$ . Các số này cũng được biết đến như là các số nguyên tố Fermat<sup>17</sup>.

Tuy nhiên, các bit trong lũy thừa giải mã  $d$  không thuận lợi và do đó giải mã sử dụng phương pháp chuẩn của phép lũy thừa modulo như trên sẽ chậm hơn nhiều so với mã hóa. Lưu ý, không được chọn giá trị  $d$  nhỏ nhằm giảm thời gian giải mã bởi vì chúng không an toàn (đã được trình bày ở mục 5.2.5).

---

<sup>17</sup> Số nguyên tố Fermat là các số có dạng  $F_x = 2^{2^x} + 1$  với  $x \geq 0$ . Ví dụ,  $F(0) = 3$ ,  $F(2) = 17$ ,  $F(4) = 65537$ .

Một phương pháp hiệu quả hơn là sử dụng định lý số dư Trung Hoa (Chinese Remainder Theorem – CRT) để tính lũy thừa  $x^d \bmod pq$  với  $p, q$  là các số nguyên tố [2]. Điều này hoàn toàn thực hiện được bởi người giải mã do người này có khóa bí mật  $d$  và các số nguyên tố  $p, q$ .

Cho  $m_1, m_2, \dots, m_n$  là các số nguyên, đôi một nguyên tố cùng nhau.

Đặt  $m = m_1m_2 \dots m_n$ . Theo định lý số dư Trung Hoa, ta có kết quả sau:

$$f : \mathbb{Z}_m \rightarrow \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_n}$$

$$f(x) = (x \bmod m_1, x \bmod m_2, \dots, x \bmod m_n) \text{ là song ánh.}$$

Như vậy, để tính  $x^d \bmod m$ ,  $m = m_1m_2 \dots m_n$  với  $m_1, m_2, \dots, m_n$  là các số nguyên đôi một nguyên tố cùng nhau, ta có thể tính  $(x^d \bmod m_1, x^d \bmod m_2, \dots, x^d \bmod m_n)$ .

Cụ thể, để thực hiện phép tính  $x^d \bmod pq$ , ta thực hiện như sau:

Đặt  $d_1 = d \bmod (p-1)$  và  $d_2 = d \bmod (q-1)$

Tồn tại số nguyên  $n_1, n_2$  sao cho:

$$d = d_1 + n_1(p-1) = d_2 + n_2(q-1)$$

Vậy:

$$x^d \bmod p = x^{d_1} \bmod p \quad x^{p-1} \bmod p \quad n^1$$

Theo định lý Fermat nhỏ, ta có  $x^{p-1} \bmod p = 1$ , vậy:

$$x^d \bmod p = x^{d_1} \bmod p$$

Tương tự ta có:

$$x^d \bmod q = x^{d_2} \bmod q$$

Như vậy, tính  $x^d \bmod qp$  được đưa về tính:

$$x^{d_1} \bmod p = x_1^{d_1} \bmod p \text{ với } x_1 = x \bmod p$$

$$x^{d_2} \bmod q = x_2^{d_2} \bmod q \text{ với } x_2 = x \bmod q$$

Từ các lập luận trên, ta có thuật toán sau để tính nhanh  $m = c^d \bmod pq$  với  $d, p, q$  biết trước.

**ModPowCRT(x, d, p, q)**

Đầu vào: số nguyên  $x > 0$ , số mũ nguyên  $d \geq 0$ , hai số nguyên tố  $p$  và  $q$  với  $p > q$

Đầu ra: giá trị  $x^d \bmod pq$

- (1)  $dP \leftarrow d \bmod p - 1$ .
- (2)  $dQ \leftarrow d \bmod q - 1$ .
- (3)  $qInv \leftarrow q^{-1} \bmod p$ .
- (4)  $c_1 \leftarrow c \bmod p$ .
- (5)  $c_2 \leftarrow c \bmod q$ .
- (6)  $m_1 \leftarrow c_1^{dP} \bmod p$ .
- (7)  $m_2 \leftarrow c_2^{dQ} \bmod q$ .
- (8)  $\square \leftarrow qInv(m_1 - m_2) \bmod p$ .
- (9)  $m \leftarrow m_2 + \square q$ .
- (10) Trả về  $m$ .

**Thuật toán 6.2. Tính lũy thừa modulo bằng thuật toán sử dụng định lý số dư Trung Hoa**

Để không mất thời gian tính toán lại, bước (1), (2) và (3) có thể được tính trước và lưu trữ cùng với  $p$  và  $q$  như là khóa bí mật, nghĩa là khóa bí mật được đại diện như là một bộ năm ( $p, q, dP, dQ$  và  $qInv$ ), trong đó  $p$  và  $q$  là các thừa số nguyên tố của  $n$ ,

$dP$  và  $dQ$  được biết đến như là lũy thừa CRT và  $qInv$  là hệ số CRT. Mặc dù có nhiều bước hơn trong thủ tục nhưng lũy thừa *modulo* trong phương pháp này được thực hiện với các số mũ ngắn hơn và do đó trên tổng thể ít chi phí hơn. Thực nghiệm cho thấy phương pháp CRT trong giải mã nhanh hơn gần 9 lần trên tổng thể so với phương pháp nhị phân khi tính  $m = c^d \bmod n$  (sẽ được trình bày ở Chương 7).

Các kỹ thuật nhân nhanh như các phương pháp dựa trên biến đổi *Fourier* nhanh (Fast Fourier Transform – FFT) ít được sử dụng do phức tạp trong cài đặt và đôi khi chúng chậm hơn các thuật toán trên đối với một số kích thước khóa đặc trưng.

### 6.3 Bài toán phát sinh số ngẫu nhiên

Sự an toàn của bất kỳ thuật toán mã hóa nào cuối cùng cũng đều phụ thuộc trên các số ngẫu nhiên. Cần sử dụng các phương pháp tìm số ngẫu nhiên đủ mạnh (có chu kỳ dài, các số được lựa chọn không thể dự đoán được) để kẻ tấn công không thể lợi dụng để biết thêm thông tin về việc lựa chọn.

Phát sinh số ngẫu nhiên (Random Number Generator – RNG) là sự tính toán hoặc thiết bị vật lý được thiết kế để phát sinh một chuỗi các số hoặc ký hiệu “không có dạng” (không đoán trước được). Có hai loại số ngẫu nhiên:

- Số thật sự ngẫu nhiên (“true” random number) có được bằng cách đo các hiện tượng vật lý được mong đợi là ngẫu nhiên.
- Số giả ngẫu nhiên (pseudo-random number) có được bằng cách sử dụng các thuật toán tính toán để tạo một chuỗi dài các kết quả ngẫu nhiên, hoàn toàn quyết định bởi giá trị khởi tạo (seed). Giả ở đây nghĩa là không có thật.

Thông thường các số giả ngẫu nhiên thường được quan tâm sử dụng vì nó phụ thuộc vào các thuật toán phát sinh hơn là các thiết bị vật lý. Đầu ra ngẫu nhiên của bộ phát sinh số ngẫu nhiên (pseudo-random number Generator – PRNG) được kiên cố bởi hàm mã hóa (sử dụng mật mã hoặc hàm băm). Trong trường hợp này PRNG được gọi theo cách mã hóa là PRNG mạnh. PRNG được sử dụng để tạo các số ngẫu nhiên cho việc tạo lập hệ mã và phát sinh khóa trong RSA (cũng như trong các hệ mã khác). Đây là một lĩnh vực rất thú vị nhưng khó khăn do việc tạo PRNG an toàn là rất khó.

## 6.4 Bài toán kiểm tra tính nguyên tố của một số nguyên

### 6.4.1 Giới thiệu

Nếu thuật toán kiểm tra tính nguyên tố trả về kết quả sai thì hệ mã hóa RSA có thể không còn an toàn hoặc sẽ không hoạt động đúng. Có rất nhiều thuật toán được đề xuất với tốc độ và độ chính xác khác nhau nhằm xác định một số nguyên cho trước có phải là số nguyên tố hay không. Các thuật toán này được chia làm hai nhóm:

- **Nhóm các thuật toán kiểm tra tính nguyên tố theo xác suất:** nghĩa kết luận có thể sai nhưng với xác suất rất thấp. Các số nguyên vượt qua các kiểm tra bằng cách này được gọi là số nguyên tố xác suất (probable prime) hay số giả nguyên tố mạnh (strong-pseudo prime).
- **Nhóm các thuật toán kiểm tra tính nguyên tố “thật sự”:** nghĩa là có thể chứng minh được nó là số nguyên tố nên còn được gọi là thuật toán chứng minh tính nguyên tố. Các số nguyên vượt qua các kiểm tra bằng cách này được gọi là số nguyên tố (provable prime).

Một thuật toán nổi tiếng trong nhóm các thuật toán kiểm tra tính nguyên tố “thật sự” là thuật toán AKS. Thuật toán này có độ phức tạp đa thức, được phát triển bởi ba tác giả Manindra Agrawal, Neeraj Kayal và Nitin Saxena, tại Viện công nghệ Kanpur – Ấn Độ, vào tháng 8 năm 2002 [4]. Tuy nhiên, thuật toán chưa chứng tỏ được tính hiệu quả rõ rệt trong tính toán thực tiễn do bậc đa thức khá cao, khoảng  $O(\log^{12}n)$  nhưng lại có ý nghĩa lớn về mặt lý thuyết vì vấn đề này đã thu hút sự quan tâm nghiên cứu của nhiều người trong một thời gian dài. Trên thực tế, những thuật toán kiểm tra tính nguyên tố theo xác suất thường được sử dụng vì đơn giản và chi phí tính toán ít hơn rất nhiều so với các thuật toán kiểm tra tính nguyên tố thật sự. Mặt khác, các thuật toán này tuy có khả năng kết luận sai nhưng với xác suất cực kỳ thấp, có thể xem là không thể xảy ra. Vì vậy, trong đề tài này chỉ tập trung nghiên cứu và phân tích các thuật toán kiểm tra tính nguyên tố theo xác suất.

### 6.4.2 Một số thuật toán kiểm tra tính nguyên tố theo xác suất

#### 6.4.2.1 Thuật toán Fermat

Theo định lý Fermat, nếu  $n$  là số nguyên tố và  $a$  là số nguyên bất kỳ,  $1 \leq a \leq n - 1$  thì  $a^{n-1} \equiv 1 \pmod{n}$ . Do đó, cho trước một số nguyên  $n$ , nếu tìm được một giá trị  $a$  sao cho  $a^{n-1} \not\equiv 1 \pmod{n}$  thì  $n$  chẵn chẵn là hợp số. Chiều ngược lại của định lý không đúng, nghĩa là nếu một số nguyên  $n$  thỏa  $a^{n-1} \equiv 1 \pmod{n}$  thì chưa chắc  $n$  là số nguyên tố. Trong trường hợp này ta gọi  $n$  là số giả nguyên tố cơ sở  $a$ .

Thuật toán kiểm tra Fermat như sau [40, tr.136-137]:

**Fermat(n, t)**

**Đầu vào:** số nguyên lẻ  $n \geq 3$  và tham số an toàn  $t \geq 1^{18}$ .

**Đầu ra:**  $n$  là “số nguyên tố” hay “hợp số”

(1) **Với mỗi**  $i = 1 \rightarrow t$

(1.1) Chọn một số ngẫu nhiên  $a$ ,  $2 \leq a \leq n - 2$ .

(1.2)  $r \leftarrow a^{n-1} \pmod{n}$ .

(1.3) **Nếu**  $r \neq 1$  **thì** trả về “hợp số”.

(2) Trả về “số nguyên tố”.

#### Thuật toán 6.3. Kiểm tra tính nguyên tố theo xác suất Fermat

<sup>18</sup> Nếu xác suất để một hợp số vượt qua phép thử (được kết luận là số nguyên tố) là  $p \leq 1$  thì xác suất để nó vượt qua  $t$  phép thử là  $p^t$  (rất thấp khi  $t$  lớn).

Do không thể nào kiểm tra hết mọi giá trị  $a$  nên khi thuật toán này kết luận  $n$  là “số nguyên tố” thì không có bằng chứng nào chứng minh  $n$  thật sự là số nguyên tố. Một số  $n$  là số giả nguyên tố với nhiều cơ sở  $a$  thì khả năng nó là số nguyên tố là rất lớn. Tuy nhiên, tồn tại các số giả nguyên tố với mọi cơ sở  $a$ , đó là số *Carmichael*. Số *Carmichael* là hợp số nguyên  $n$  thỏa mãn  $a^{n-1} \equiv 1 \pmod{n}$  với mọi số nguyên dương  $a$  sao cho  $(n, a) = 1$ .

Như vậy, kiểm tra Fermat sẽ kết luận sai khi gặp các số *Carmichael*. Phần tiếp theo sẽ trình bày các thuật toán kiểm tra mạnh hơn thường được sử dụng.

#### 6.4.2.2 Thuật toán Solovay-Strassen

Robert Solovay và Volker Strassen đã phát triển thuật toán kiểm tra tính nguyên tố theo xác suất [55]. Thuật toán này sử dụng ký hiệu Jacobi phục vụ cho việc kiểm tra xem  $p$  có phải là số nguyên tố hay không.

Thuật toán kiểm tra Solovay-Strassen như sau [40, tr.137-138]:

##### Solovay-Strassen( $n, t$ )

**Đầu vào:** số nguyên lẻ  $n \geq 3$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:**  $n$  là “số nguyên tố” hay “hợp số”

(1) **Với mỗi**  $i = 1 \rightarrow t$

(1.1) Chọn một số ngẫu nhiên  $a$ ,  $2 \leq a \leq n - 2$ .

(1.2)  $r \leftarrow a^{n-1} \pmod{n}$ .

(1.3) **Nếu**  $r \neq 1$  **thì** trả về “hợp số”.

(2) Trả về “số nguyên tố”.

#### Thuật toán 6.4. Kiểm tra tính nguyên tố theo xác suất Solovay-Strassen

Nếu  $n$  là hợp số, xác suất của số ngẫu nhiên  $a$  cho biết  $n$  chẵn chẵn không là số nguyên tố không ít hơn 50%. Lập lại phép thử  $t$  lần với  $t$  giá trị ngẫu nhiên khác nhau

của  $a$ , xác suất để một hợp số  $n$  qua tất cả  $t$  phép thử không lớn hơn  $\frac{1}{2^t}$ .

Đây là thuật toán được sử dụng phổ biến đầu tiên cùng với sự phát triển của hệ mã khóa công khai, đặc biệt là hệ mã RSA. Tuy nhiên, hiện nay nó được thay thế bởi thuật toán Miller Rabin hiệu quả hơn và luôn cho kết quả chính xác.

#### 6.4.2.3 Thuật toán Miller Rabin

Thuật toán khá dễ hiểu và được nhiều người sử dụng được phát triển bởi Michael Rabin, dựa trên một phần ý tưởng của Gary Miller [41], [46]. Đây là thuật toán kiểm

tra tính xác suất được sử dụng hầu hết trong thực tế, cũng được biết đến như là kiểm tra số giả nguyên tố mạnh. Nó là phiên bản đơn giản hóa của thuật toán được đề nghị trong đề xuất của chuẩn chữ ký số DSS.

Kiểm tra Miller Rabin dựa trên định lý sau:

Cho  $n$  là một số nguyên tố lẻ, cho  $n - 1 = 2^s r$  trong đó  $r$  là số lẻ.

Cho  $a$  là số nguyên bất kỳ sao cho  $\gcd(a, n) = 1$ . Lúc này  $a^r \equiv 1 \pmod{n}$  hoặc  $a^{2^j r} \equiv 1 \pmod{n}$  với  $j$  nào đó,  $0 \leq j \leq s - 1$ .

Thuật toán kiểm tra Miller-Rabin như sau [40, tr.138-140]:

#### Miller-Rabin(n, t)

**Đầu vào:** số nguyên lẻ  $n \geq 3$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:**  $n$  là “số nguyên tố” hay “hợp số”

(1) Tính  $r$  lẻ và  $s$  với  $n - 1 = 2^s r$ .

(2) **Với mỗi**  $i = 1 \rightarrow t$

(2.1) Chọn một số ngẫu nhiên  $a$ ,  $2 \leq a \leq n - 2$ .

(2.2)  $y \leftarrow a^r \pmod{n}$ .

(2.3) **Nếu**  $y \neq 1$  và  $y \neq n - 1$  **thì**

$j \leftarrow 1$ .

**Trong khi**  $j \leq s - 1$  và  $y \neq n - 1$

$y \leftarrow y^2 \pmod{n}$ .

**Nếu**  $y = 1$  **thì** trả về “hợp số”.

$j \leftarrow j + 1$ .

**Nếu**  $y \neq n - 1$  **thì** trả về “hợp số”.

(3) Trả về “số nguyên tố”.

#### Thuật toán 6.5. Kiểm tra tính nguyên tố theo xác suất Miller Rabin

Xác suất của một hợp số  $n$  vượt qua phép thử giảm nhanh hơn qua các phép thử sau.

Ba phần tư các giá trị có thể của  $a$  cho biết  $n$  là hợp số. Điều đó có nghĩa là xác suất cho một hợp số  $n$  vượt qua  $t$  phép thử không nhiều hơn  $\frac{1}{4^t}$ . Số nguyên vượt qua

phép thử Miller-Rabin được gọi là số giả nguyên tố mạnh.

#### 6.4.3 Nhận xét

Cả hai thuật toán Miller-Rabin và Solovay-Strassen đều chính xác khi biến cỗ đầu vào là số nguyên tố hoặc hợp số. Tuy nhiên, không có lý do gì để sử dụng kiểm tra Solovay-Strassen (và kiểm tra Fermat) vì kiểm tra Miller-Rabin tốt hơn rất nhiều:

- Kiểm tra Solovay-Strassen tốn rất nhiều chi phí trong tính toán.
- Kiểm tra Solovay-Strassen khó thực thi do liên quan đến tính ký hiệu Jacobi.
- Xác suất sai của kiểm tra Solovay-Strassen được chặn trên bởi  $\frac{1}{2}^t$  trong khi xác suất sai của thuật toán Miller-Rabin được chặn trên bởi  $\frac{1}{4}^t$ .

Nếu một số nguyên lẻ  $k$ -bit có thể chia hết bởi số nguyên tố nhỏ, nó sẽ tốn ít chi phí hơn để phát hiện ra bằng cách sử dụng thuật toán chia thử hơn là sử dụng kiểm tra Miller-Rabin. Do xác suất một số nguyên ngẫu nhiên  $n$  có số chia nguyên tố nhỏ là rất cao, trước khi áp dụng kiểm tra Miller-Rabin,  $n$  nên được kiểm tra với các số chia nhỏ bé hơn biên  $B$  định trước. Việc này có thể thực hiện bằng cách chia  $n$  cho tất cả các số nguyên tố nhỏ hơn  $B$ , hoặc bằng cách tính ước số chung lớn nhất của  $n$  và tích của một vài số nguyên tố  $\leq B$ . Tỷ lệ của các số nguyên lẻ  $n$  không bị loại trừ bởi việc

chia thử này là  $\prod_{p \leq B} \frac{1 - \frac{1}{p}}{\ln p}$ , (theo định lý Mertens). Ví dụ, nếu

$B = 256$  thì chỉ có 20% số nguyên tố lẻ vượt qua bước chia thử, nghĩa là 80% bị loại bỏ trước khi kiểm tra chi phí cao Miller-Rabin được thực hiện.

Vấn đề là chọn biên  $B$  bao nhiêu là tối ưu. Trong thực nghiệm, người ta chọn  $B = E/D$  với  $E$  là thời gian thực hiện của một phép lũy thừa modulo  $k$ -bit đầy đủ và  $D$  là thời gian cần thiểu để loại trừ một số nguyên tố nhỏ là số chia của số nguyên  $k$ -bit. Các số nguyên lẻ bé hơn  $B$  có thể được tính toán trước và chưa sẵn trong một bảng. Nếu có ít bộ nhớ, giá trị  $B$  nhỏ hơn giá trị tối ưu có thể được sử dụng.

Gọi  $p_{k,t}$  là xác suất một hợp số  $k$ -bit vượt qua  $t$  phép thử. Kiểm tra Miller-Rabin cho

ta  $p_{k,t} \leq \frac{1}{4}^t$  với  $t$  phép thử. Tuy nhiên, các chứng minh theo xác suất cho thấy số

phép thử  $t$  cần thiết ít hơn rất nhiều nhưng vẫn đạt được cùng xác suất sai, cụ thể như sau [40, tr.146-148]:

$$(1) p_{k,1} < k^2 4^{2-k} \text{ với } k \geq 2.$$

$$(2) p_{k,t} < k^{3/2} 2^t t^{-1/2} 4^{2-tk} \text{ với } (t = 2, k \geq 88) \text{ hoặc } (3 \leq t \leq \frac{k}{2}, k \geq 21).$$

$$(3) p_{k,t} < \frac{7}{20} k 2^{-5t} + \frac{1}{7} k^{15/4} 2^{-k/2-2t} + 12k 2^{-k/4-3t} \text{ với } \frac{k}{9} \leq t \leq \frac{k}{4}, k \geq 21.$$

$$(4) p_{k,t} < \frac{1}{7} k^{15/4} 2^{-k/2-2t} \text{ với } t \geq \frac{k}{4}, k \geq 21.$$

Ví dụ, nếu  $k = 512$  và  $t = 6$ , theo (2), xác suất để một hợp số  $k$ -bit vượt qua  $t$  phép thử là  $p \leq \frac{1}{2^t}$  thay vì  $p \leq \frac{1}{2^k} = \frac{1}{2^{12}}$ . Như vậy, ta không cần thiết phải sử

dụng  $t = 44$  để có được xác suất sai là  $\frac{1}{2^4} = \frac{1}{2^{88}}$ . Điều này làm giảm chi phí tính toán rất nhiều nhưng vẫn đạt được độ chính xác rất cao.

Trong thực hành, người ta thường hài lòng với xác suất sai bé hơn hay bằng  $\frac{1}{2^{80}}$ .

Dựa vào kết quả trên, ta có công thức sau để xác định giá trị  $t$  sao cho hợp số nguyên  $k$ -bit vượt qua  $t$  phép thử Miller-Rabin với xác suất bé hơn  $\frac{1}{2^{80}}$  là:

$$t = \begin{cases} 50 & k \text{ } \exists i \text{ } k < 100 \\ 27 & k \text{ } \exists i \text{ } 100 \leq k < 256 \\ 8 & k \text{ } \exists i \text{ } 256 \leq k < 512 \\ 4 & k \text{ } \exists i \text{ } 512 \leq k < 1024 \\ 2 & k \text{ } \exists i \text{ } k \geq 1024 \end{cases}$$

Các thử nghiệm nhằm đánh giá tính hiệu quả của các thuật toán này sẽ được lần lượt trình bày ở Chương 7.

## 6.5 Bài toán phát sinh số nguyên tố

### 6.5.1 Giới thiệu

Số nguyên tố là một thành phần không thể thiếu trong hệ mã RSA và phát sinh số nguyên tố là vấn đề đặc biệt quan trọng. Cách sai lầm để phát sinh các số nguyên tố là chọn ngẫu nhiên một số nguyên dương (lẻ) và cố gắng phân tích nó bởi vì việc trả lời câu hỏi “*n có phải là số nguyên tố hay không?*” dễ hơn rất nhiều so với việc trả lời “*các thừa số nguyên tố của n là gì?*”.

Lưu ý, các số nguyên tố nhận được thông qua các thuật toán kiểm tra tính nguyên tố theo xác suất (thường là thuật toán Miller-Rabin) được gọi là số nguyên tố xác suất hay tạm gọi là số “khả nguyên tố” (probable prime) vì nó vẫn có khả năng không phải là số nguyên tố nhưng với xác suất rất thấp. Ngược lại, ta nhận được các số nguyên tố có thể chứng minh được tính nguyên tố của nó (provable prime) hay ngắn gọn là số nguyên tố. Phần sau đây sẽ trình bày và phân tích các thuật toán phát sinh cả hai loại số nguyên tố trên.

## 6.5.2 Phát sinh số khả nguyên tố

### 6.5.2.1 Một số thuật toán phát sinh số khả nguyên tố ngẫu nhiên

Theo định lý số nguyên tố, tỷ lệ của các số nguyên  $\leq x$  là số nguyên tố xấp xỉ  $\frac{1}{\ln x}$ . Do một nửa số nguyên  $\leq x$  là chẵn, tỷ lệ của các số nguyên lẻ  $\leq x$  là số nguyên tố xấp xỉ  $\frac{2}{\ln x}$ . Ví dụ, tỷ lệ của tất cả số lẻ  $\leq 2^{512}$  là số nguyên tố xấp xỉ  $\frac{2}{512 \ln 2} \approx \frac{1}{177}$ . Điều này gợi ý một chiến thuật hợp lý cho việc chọn một số nguyên tố ngẫu nhiên  $k$ -bit bằng cách lặp lại việc chọn số nguyên  $k$ -bit  $n$  đến khi nó là số nguyên tố do vượt qua kiểm tra Miller-Rabin với giá trị thích hợp của tham số an toàn  $t$  [40, tr.145-146]

#### Random-Search( $k, t$ )

**Đầu vào:** số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:** số khả nguyên tố  $k$ -bit

- (1) Chọn ngẫu nhiên một số nguyên lẻ  $k$ -bit  $n$ .
- (2) Nếu  $\text{Miller-Rabin}(n, t) \neq \text{"số nguyên tố"}$  thì trở về bước (1).
- (3) Xuất  $n$ .

#### Thuật toán 6.6. Phát sinh số khả nguyên tố kiểu tìm kiếm ngẫu nhiên

Thuật toán 6.6 có một biến thể tìm kiếm tăng (incremental search). Thuật toán này khác với thuật toán trên ở một điểm là khi số  $n$  không phải là số nguyên tố thì thuật toán sẽ kiểm tra số lẻ tiếp theo [40, tr.148].

#### Incremental-Search( $k, t$ )

**Đầu vào:** số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:** số khả nguyên tố  $k$ -bit

- (1) Chọn ngẫu nhiên một số nguyên lẻ  $k$ -bit  $n$ .
- (2) Trong khi  $\text{Miller-Rabin}(n, t) \neq \text{"số nguyên tố"}$

$$n \leftarrow n + 2$$

- (3) Xuất  $n$ .

#### Thuật toán 6.7. Phát sinh số khả nguyên tố kiểu tìm kiếm tăng

Thuật toán 6.7 được cải tiến bằng cách chọn giá trị bắt đầu  $n$  là số nguyên tố cùng nhau với các số nguyên tố nhỏ [31, tr.3-4]. Thông thường, ta định nghĩa  $\Pi = 2 \times 3 \times 5 \times 7 \times \dots \times 29$  và chọn ngẫu nhiên một số  $n$   $k$ -bit thỏa  $\gcd(q, \Pi) = 1$ . Nếu  $n$  không phải là số nguyên tố thì  $q = q + \Pi$ .

**Improvement-Incremental-Search( $k, t$ )**

Đầu vào: số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

Đầu ra: số khả nguyên tố  $k$ -bit

- (1) Chọn ngẫu nhiên một số nguyên lẻ  $k$ -bit  $n$ .
  - (2) Nếu  $\text{gcd}(n, \Pi) \neq 1$  thì quay lại bước (1).
  - (3) Trong khi  $\text{Miller-Rabin}(n, t) \neq \text{"số nguyên tố"}$
- $n \leftarrow n + \Pi$
- (4) Xuất  $n$ .

**Thuật toán 6.8. Phát sinh số khả nguyên tố kiểu kiểm tăng cải tiến****6.5.2.2 Một số thuật toán phát sinh số khả nguyên tố mạnh**

Trước các phương pháp phân tích trường hợp đặc biệt, hàng loạt các đề xuất liên quan đến số nguyên tố được chọn để lập mã có một số tính chất đặc biệt đã được đưa ra. Do những tính chất đặc biệt đó đó, cơ hội để các phương pháp phân tích như trình bày ở trên thành công là rất nhỏ. Những số nguyên tố có những tính chất đặc biệt đó được gọi là số “nguyên tố mạnh”. Lý do lịch sử cho sự cần thiết này là để bảo vệ

trước các thuật toán phân tích đặc biệt như thuật toán “rho” và  $p - 1$  của Pollard,  $p + 1$  của Williams (đã được trình bày ở Chương 5).

Một số nguyên tố  $p$  được xem là một số nguyên tố mạnh nếu nó thỏa mãn các điều kiện sau:

- $p$  là một số nguyên tố lớn.
- Thừa số nguyên tố lớn nhất của  $p - 1$ , gọi là  $p^-$ , lớn.  
Nghĩa là  $p = a^-p^- + 1$  với số nguyên  $a^-$  và số nguyên tố lớn  $p^-$ .
- Thừa số nguyên tố lớn nhất của  $p^- - 1$ , gọi là  $p^{--}$ , lớn.  
Nghĩa là  $p^- = a^{--}p^{--} + 1$  với số nguyên  $a^{--}$  và số nguyên tố lớn  $p^{--}$ .
- Thừa số nguyên tố lớn nhất của  $p + 1$ , gọi là  $p^+$ , lớn.  
Nghĩa là  $p = a^+p^+ - 1$  với số nguyên  $a^+$  và số nguyên tố lớn  $p^+$ .

“Lớn” ở đây tùy thuộc vào các phương pháp phân tích hiện tại. Thường thì kích thước của  $p$  trên 256 bit còn kích thước của các thừa số  $p^-, p^{--}, p^+$  trên 100 bit.

Đôi khi, một số nguyên tố gọi là mạnh nếu nó chỉ cần thỏa mãn chỉ một tập con của các điều kiện trên, ví dụ  $p^-$ -mạnh nếu  $p^-$  lớn,  $p^{--}$ -mạnh nếu  $p^{--}$  lớn,  $p^+$ -mạnh nếu  $p^+$  lớn,  $p^-, p^+$  -mạnh nếu cả  $p^-$  và  $p^+$  đều lớn,  $p^-, p^{--}, p^+$  -mạnh, hoặc gọn hơn là mạnh, nếu cả  $p^-, p^{--}$  và  $p^+$  đều lớn.

Williams và Schmid gọi  $p^-$ -siêu mạnh (hoặc  $p^-$ -siêu mạnh hoặc  $p^+$ -siêu mạnh) nếu  $a^- = 2$  (hoặc  $a^- = 2$  hoặc  $a^+ = 2$ ) [62].

Năm 1984, Hellman và Bach còn đề nghị thêm  $p^+ - 1$  chứa một thừa số nguyên tố lớn (gọi là  $p^+$ ). Tuy nhiên, các ông chưa đưa ra chứng minh nào cho đề xuất đó [29].

Năm 1978, các tác giả của hệ mã RSA đã đề xuất việc sử dụng số nguyên tố  $p^-$ -mạnh [50], và được tìm dễ dàng như sau:

#### Simple-StrongPrime( $k, t$ )

**Đầu vào:** số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:** số khả nguyên tố mạnh  $k$ -bit

(1) Tìm một số nguyên tố ngẫu nhiên lớn  $p^-$  bằng cách kiểm tra tính nguyên tố một số nguyên ngẫu nhiên lớn.

(2) Tính  $p^-$  là số nguyên tố nhỏ nhất có dạng:

$$p^- = a^- p^- + 1$$

với số nguyên với số nguyên  $a^-$  nào đó. Có thể tính bằng cách thử  $a^- = 2, 4, 6, \dots$  đến khi  $p^-$  là số nguyên tố. Sử dụng phép thử tính nguyên tố bằng xác suất như phép thử Rabin-Miller để kiểm tra tính nguyên tố của mỗi  $p^-$ .

(3) Tính  $p$  là số nguyên tố nhỏ nhất có dạng:

$$p = a^- p^- + 1$$

với số nguyên  $a^-$  nào đó giống như cách tìm  $a^-$  ở bước (2).

#### Thuật toán 6.9. Phát sinh số khả nguyên tố mạnh đơn giản

Thời gian cần thiết để tìm  $p$  khá lâu, gấp khoảng 3 lần thời gian cần thiết để tìm một số nguyên tố ngẫu nhiên có cùng kích thước (do kiểm tra tính nguyên tố 3 lần). Hơn nữa, số nguyên tố  $p$  nhận được ở thuật toán trên là chỉ là số  $p^-$ -mạnh.

Năm 1979, Williams và Schmid đề xuất thuật toán tìm số nguyên tố mạnh như sau [62]:

#### Williams-Schmid( $k, t$ )

**Đầu vào:** số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:** số khả nguyên tố mạnh  $k$ -bit

(1) Tìm  $p^-$  và  $p^+$  là các số nguyên tố ngẫu nhiên lớn.

(2) Tính  $r = -p^- - 1 \bmod p^+$ .

(3) Tìm  $a$  nhỏ nhất sao cho:

$$p^- = 2ap^-p^+ + 2rp^- + 1$$

$$\text{và } p = 4ap^-p^+ + 4rp^- + 3 = 2p^- + 1$$

là số nguyên tố.

#### Thuật toán 6.10. Phát sinh số khả nguyên tố mạnh Williams/Schmid

Dễ thấy rằng  $p^+$  chính là thừa số của  $p + 1$  và  $p$  là  $p^-$ –siêu mạnh do  $a^- = 2$ . Độ dài của  $p$  và  $p^-$  khoảng gấp đôi độ dài của  $p^{--}$  và  $p^+$  với quy trình trên. Quy trình này không hiệu quả bằng việc tìm một số nguyên tố ngẫu nhiên, do việc tìm  $a$  đồng thời làm cho  $p^-$  và  $p$  là số nguyên tố trong bước (3) phức tạp hơn rất nhiều so với việc tìm

$a^-$  để tạo  $p$  là số nguyên tố trong bước (3) của Thuật toán 6.9. Tuy nhiên đây cũng là một cách để tìm các số nguyên tố mạnh.

Năm 1984, J. Gordon đề xuất quy trình khác để tìm các số nguyên tố mạnh [26], [25]. Gordon cho rằng việc tìm số nguyên tố mạnh chỉ khó hơn một chút so với việc tìm số nguyên tố ngẫu nhiên có cùng kích thước. Thuật toán của ông hiệu quả hơn thuật toán của Williams/ Schmid nhiều bởi vì thuật toán không tạo ra số nguyên tố

$p^-$ –siêu mạnh (giá trị của  $a^-$  sẽ lớn hơn 2 với thuật toán của Gordon).

### Gordon( $k, t$ )

**Đầu vào:** số nguyên  $k > 0$  và tham số an toàn  $t \geq 1$ .

**Đầu ra:** số khả nguyên tố mạnh  $k$ -bit

(1) Tìm  $p^{--}$  và  $p^+$  là các số nguyên tố ngẫu nhiên bằng thuật toán tìm kiếm tăng.

(2) Tính  $p^-$  là số nguyên tố nhỏ nhất có dạng:

$$p^- = a^{--}p^{--} + 1$$

với số nguyên  $a^{--} = 2, 4, 6, \dots$  nào đó.

(3) Đặt  $p_0 = p^+ p^{--1} - p^- p^{+-1} \bmod p^- p^+$

(4) Tính  $p$  là số nguyên tố nhỏ nhất có dạng:

$$p = p_0 + ap^-p^+$$

với số nguyên  $a = 2, 4, 6, \dots$  nào đó.

### Thuật toán 6.11. Phát sinh số khả nguyên tố mạnh Gordon

Bằng cách điều chỉnh độ dài (theo bit) của các số nguyên tố  $p^{--}$  và  $p^+$  và các giá trị  $a^{--}$  và  $a$ , chúng ta sẽ có thể điều chỉnh được kích thước mong muốn của số nguyên tố  $p$ . Lưu ý rằng độ dài theo bit của  $p^-$  và  $p^+$  sẽ xấp xỉ một nửa của  $p$  trong khi độ dài theo bit của  $p^{--}$  sẽ ít hơn độ dài theo bit của  $p^-$  một chút.

Gordon chứng minh được thuật toán của ông chỉ chậm hơn 19% so với thuật toán tìm số nguyên tố ngẫu nhiên cùng kích thước. Tuy nhiên, Gordon mô tả thuật toán tìm số nguyên tố trong bước (1) là thuật toán tìm kiếm tăng (Thuật toán 6.7) nên khi sử dụng phiên bản cải tiến của thuật toán tìm kiếm tăng (Thuật toán 6.8) thì tốc độ tổng thể của thuật toán phát sinh số khả nguyên tố mạnh Gordon sẽ tăng lên đáng kể.

### 6.5.3 Phát sinh số nguyên tố

Các thuật toán được giới thiệu ở trên đều sử dụng các thuật toán kiểm tra tính nguyên tố theo xác suất, vì vậy số nguyên tố phát sinh được chỉ là một số nguyên tố xác suất (probable prime) hay số khả nguyên tố mạnh. Những số nguyên tố loại này vẫn có thể

là hợp số mặc dù với xác suất sai vô cùng thấp, chẳng hạn xác suất sai ít hơn  $\frac{1}{2^{80}}$ .

Vì thế, người ta mong muốn có thể phát sinh được các số nguyên tố thật sự hay nói cách khác là có thể chứng minh được tính nguyên tố của các số này (provable prime). Thuật toán sau đây của Maurer cho phép phát sinh một số nguyên tố có độ dài  $k$  bit xác định trước [39].

#### Maurer( $k$ )

**Dầu vào:** số nguyên  $k > 0$

**Dầu ra:** số nguyên tố  $k$ -bit

- (1) **Nếu**  $k \leq 20$  **thì** thực hiện lặp lại các bước sau:
  - (1.1) Chọn một số nguyên  $n$  lẻ ngẫu nhiên  $k$ -bit.
  - (1.2) Sử dụng chia thử bởi tất cả các số nguyên tố bé hơn  $\sqrt{n}$  để xem  $n$  có phải là số nguyên tố hay không.
  - (1.3) **Nếu**  $n$  là số nguyên tố **thì** trả về  $n$ .
- (2)  $c \leftarrow 0.1, m \leftarrow 20.$ 
  - (3)  $B \leftarrow c \cdot k^2$  ( $B$  là chặn trên của chia thử).
  - (4) **Nếu**  $k > 2m$  **thì** thực hiện lặp lại các bước sau: chọn một số ngẫu nhiên  $s \in \{0, 1\}$ , đặt  $r \leftarrow 2^{s-1}$ , **cho đến khi**  $k - rk > m$ . **Ngược lại** (nghĩa là  $k \leq 2m$ ),  $r \leftarrow 0.5$ .
  - (5)  $q \leftarrow \text{Maurer}_{k-1}(r \cdot k + 1)$ .
  - (6)  $I \leftarrow \frac{2^{k-1}}{2q}.$
  - (7)  $\text{success} \leftarrow 0$
  - (8) **Trong khi** ( $\text{success} = 0$ )
    - (8.1) Chọn một số nguyên  $s \in I + 1, 2I$  và đặt  $n = 2Rq + 1$ .
    - (8.2) Sử dụng chia thử để xem  $n$  có bị chia hết bởi bất kỳ số nguyên tố nào  $< B$  hay không. Nếu không thì thực hiện các bước sau:
      - (8.2.1) Chọn một số nguyên ngẫu nhiên  $a \in \{2, n - 2\}$ .
      - (8.2.2)  $b \leftarrow a^{n-1} \bmod n$ . (8.2.3)
    - Nếu**  $b = 1$  **thì**

$$b \leftarrow a^{2R} \bmod n \text{ và } d \leftarrow \gcd(b - 1, n).$$
**Nếu**  $d = 1$  **thì**  $\text{success} \leftarrow 1$ .
  - (9) Trả về  $n$ .

Trong bước (2), giá trị tối ưu của hằng số  $c = 0.1$  để tính biên chia thử  $B = c \cdot k^2$ . Như đã đề cập ở mục 6.4.3, giá trị này tùy thuộc vào sự thực thi số học của số nguyên dài và được chọn thông qua thực nghiệm. Ngoài ra, hằng số  $m = 20$  để chắc chắn rằng  $I$  dài ít nhất 20 bit và do đó  $R$  được chọn trong đoạn  $I + 1, 2I$  đủ lớn để

$n = 2Rq + 1$  chứa ít nhất một số nguyên tố  $R$  lớn.

Maurer nhận xét rằng số nguyên tố xác suất nhận được trong thuật toán tìm kiếm ngẫu nhiên (Thuật toán 6.6) với  $t = 1$  chỉ nhanh hơn một chút so với thuật toán của Maurer. Tuy nhiên, trong thực tế người ta thường sử dụng  $t \geq 1$  nên thời gian phát sinh số nguyên tố bằng thuật toán Maurer sẽ lâu hơn rất nhiều. Ngoài ra, thuật toán này đòi hỏi nhiều bộ nhớ để chạy do có sự đệ quy trong hàm.

#### 6.5.4 Nhận xét

Các số nguyên tố được ưa thích hơn số giả nguyên tố mạnh do các số này có xác suất sai bằng không. Tuy nhiên, xác suất sai của các số giả nguyên tố có khả năng giảm xuống mức thấp có thể chấp nhận được như đã trình bày ở trên và thời gian tìm số khả nguyên tố mạnh ít hơn rất nhiều so với thời gian tìm số nguyên tố nên trong thực tế các số khả nguyên tố mạnh hay số nguyên tố xác suất thường được sử dụng.

Các thử nghiệm nhằm đánh giá tính hiệu quả của các thuật toán này sẽ được lần lượt trình bày ở Chương 7.

### 6.6 Kết luận

RSA là hệ mã rất dễ hiểu và dễ triển khai nhưng để vận dụng nó đúng cách nhằm đạt độ an toàn và hiệu quả lại vô cùng khó khăn. Để giải quyết tốt các vấn đề này, người lập mã cần tuân thủ các đề nghị về tính an toàn được đưa ra ở Chương 5 và các phân tích về tính hiệu quả được trình bày ở chương này.

Hơn nữa, nhu cầu xây dựng một bộ thư viện mã hóa để hiện thực hóa các phân tích ở trên là cần thiết. Chương 7 sẽ giới thiệu bộ thư viện mã hóa được xây dựng nhằm triển khai hệ mã RSA an toàn và hiệu quả.

## Chương 7

# Xây dựng bộ thư viện “SmartRSA”, cài đặt hiệu quả hệ mã RSA

☐ Nội dung của chương này giới thiệu bộ thư viện mã hóa “SmartRSA” được xây dựng nhằm cài đặt hiệu quả hệ mã RSA trên cơ sở nghiên cứu và phân tích về các nguy cơ tổn thương hệ mã ở Chương 5 và các bài toán quan trọng trong việc thiết lập hệ mã hiệu quả ở Chương 6. Các thử nghiệm nhằm kiểm tra tính hiệu quả được trình bày ở mục 7.4.

### 7.1 Giới thiệu

“SmartRSA” là bộ thư viện được xây dựng bằng ngôn ngữ lập trình Java nhằm cung cấp các chức năng cần thiết hỗ trợ cho việc cài đặt hoàn chỉnh hệ mã RSA. Với sự kế thừa một số chức năng hiệu quả có sẵn trong Java như thư viện tính toán nhanh trên số lớn (gói `java.util.BigInteger`) và thư viện phát sinh số ngẫu nhiên mạnh (gói `java.security.SecureRandom`), bộ thư viện SmartRSA cho phép cài đặt hệ mã RSA đạt độ an toàn và hiệu quả như đã phân tích ở Chương 5 và Chương 6.

### 7.2 Các thuật toán và chức năng đợc cung cấp trong thư viện

SmartRSA cung cấp đầy đủ các chức năng để cài đặt một hệ mã RSA hoàn chỉnh kể cả chức năng ký và xác nhận chữ ký số RSA:

- Hàm băm: MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320, Tiger, Whirlpool.
- Tính toán nhanh lũy thừa *modulo*: Thuật toán nhị phân và Thuật toán sử dụng định lý số dư Trung Hoa (Chinese Remainder Theorem – CRT).
- Kiểm tra tính nguyên tố: Thuật toán chia thử (Trial Division), Fermat, Solovay-Strassen, Miller-Rabin, Miller-Rabin tối ưu (Optimal Miller-Rabin).

- Phát sinh số nguyên tố xác suất: Thuật toán tìm kiếm ngẫu nhiên (Random Search), Tìm kiếm tăng (Incremental Search), Tìm kiếm tăng cải tiến (Optimal Incremental Search), Tìm số nguyên tố mạnh (Gordon).
- Phát sinh số nguyên tố bằng thuật toán Maurer.
- Phát sinh cặp khóa mạnh cho hệ mã RSA (sử dụng số nguyên tố mạnh).
- Ký và xác nhận chữ ký số RSA.

### 7.3 Một số đặc tính của bộ thư viện

- **Tốc độ thực hiện tương đối nhanh:** do sử dụng các thuật toán đã được phân tích và cải tiến đáng kể.
- **Độ an toàn bảo mật cao:** thư viện cung cấp các thuật toán đã chứng minh tính ổn định trong thời gian dài, có độ an toàn cao.
- **Độc lập môi trường:** do được viết bằng ngôn ngữ Java nên bộ thư viện có thể được sử dụng trên các môi trường khác nhau như Windows, Linux, ...
- **Dễ mở rộng, bổ sung thuật toán và sử dụng để phát triển các ứng dụng khác:** thư viện được xây dựng theo kiến trúc hướng đối tượng, có sự nhất quán trong việc tổ chức các phương thức xử lý và thuộc dạng mã nguồn mở nên giúp cho người sử dụng dễ hiểu và dễ dàng trong việc sử dụng thư viện để xây dựng các tính năng bảo vệ thông tin trong các ứng dụng khác.
- **Tính khả chuyển:** do có nhiều điểm tương đồng giữa ngôn ngữ C/C# và Java, chúng ta có thể dễ dàng chuyển đổi thư viện này sang môi trường C/C#.

### 7.4 Kết quả thử nghiệm và nhận xét

Các thử nghiệm và đánh giá tính hiệu quả của các hàm băm mật mã và thuật toán chữ ký số RSA đã được trình bày ở Chương 2. Phần này sẽ tiến hành thử nghiệm và đánh giá hiệu quả của các thuật toán đã được trình bày ở Chương 6 bao gồm: các thuật toán tính nhanh lũy thừa *modulo*, các thuật toán kiểm tra tính nguyên tố theo xác suất và các thuật toán phát sinh số nguyên tố.

### 7.4.1 Các thuật toán tính nhanh lũy thừa modulo

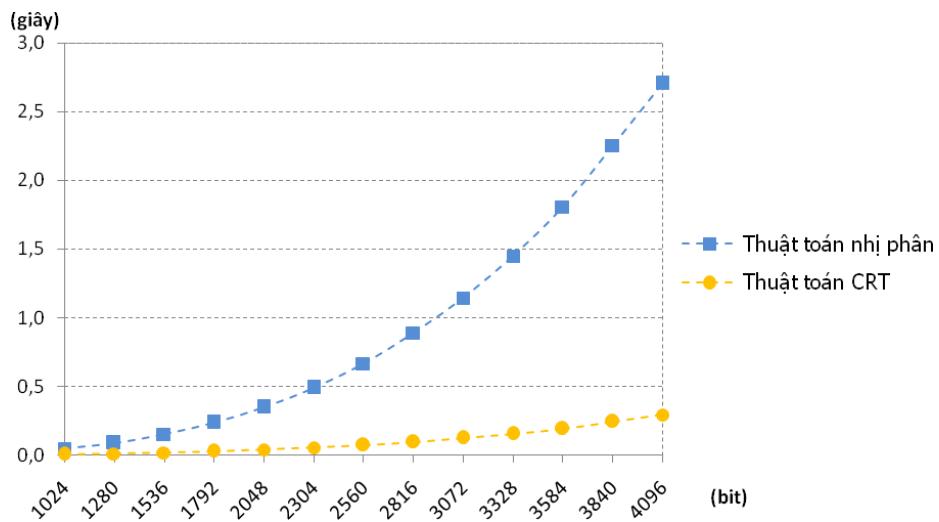
Mục 6.2 đã lần lượt giới thiệu 2 thuật toán tính nhanh lũy thừa *modulo*, đó là thuật toán nhị phân (Thuật toán 6.1) và thuật toán sử dụng định lý số dư Trung Hoa – CRT (Thuật toán 6.2). Thử nghiệm 7.1 sau đã được tiến hành để đánh giá hiệu quả của hai thuật toán này.

**Thử nghiệm 7.1:** Độ dài hai số nguyên tố  $p$  và  $q$  là  $512 + 128i$  (bit) với  $0 \leq i \leq 12$  ứng với độ dài của *modulo*  $n$  là  $1024 + 256i$ . Ứng với mỗi độ dài này, chương trình tự động phát sinh  $p$ ,  $q$  và tính  $n$ ,  $dP$ ,  $dQ$ ,  $qInv$ , số mũ khóa bí mật  $d$  (từ số mũ công khai cố định là  $e = 65537$ ). Sau đó chương trình phát sinh ngẫu nhiên thông điệp  $m < n$  cùng độ dài với  $n$  và tiến hành đo thời gian thực hiện phép tính  $m^d \bmod n$  bằng thuật toán nhị phân và thuật toán CRT. Thử nghiệm được lặp lại 50.000 lần.

**Bảng 7.1. Thời gian thực hiện của các thuật toán tính lũy thừa modulo**

| Độ dài<br>(bit) | Thời gian tính toán (giây)            |                                  | Tỷ lệ (%) |
|-----------------|---------------------------------------|----------------------------------|-----------|
|                 | Thuật toán<br>nhị phân <sup>(1)</sup> | Thuật toán<br>CRT <sup>(2)</sup> |           |
| 1024            | 0,0469                                | 0,0059                           | 800,84%   |
| 1280            | 0,0881                                | 0,0103                           | 858,58%   |
| 1536            | 0,1492                                | 0,0173                           | 864,29%   |
| 1792            | 0,2351                                | 0,0272                           | 865,64%   |
| 2048            | 0,3486                                | 0,0389                           | 895,59%   |
| 2304            | 0,4900                                | 0,0542                           | 903,74%   |
| 2560            | 0,6673                                | 0,0733                           | 910,30%   |
| 2816            | 0,8882                                | 0,0967                           | 918,08%   |
| 3072            | 1,1431                                | 0,1241                           | 921,34%   |
| 3328            | 1,4491                                | 0,1563                           | 927,37%   |
| 3584            | 1,8050                                | 0,1936                           | 932,28%   |
| 3840            | 2,2521                                | 0,2443                           | 921,73%   |
| 4096            | 2,7062                                | 0,2921                           | 926,45%   |
| Trung bình      |                                       |                                  | 895,86%   |

Kết quả Thử nghiệm 7.1 cho thấy thuật toán nhị phân chậm hơn rất nhiều so với thuật toán CRT (gấp trung bình 895,96%) khi số mũ lũy thừa là một số ngẫu nhiên lớn. Như vậy, thuật toán CRT nên được sử dụng để thực hiện công việc ký hay giải mã vì lúc này số mũ bí mật  $d$  là một số lớn đồng thời người thực hiện công việc này là chủ của khóa nên có trong tay  $p$  và  $q$ .



Hình 7.1. Thời gian thực hiện của các thuật toán tính lũy thừa modulo

#### 7.4.2 Các thuật toán kiểm tra tính nguyên tố theo xác suất

Mục 6.4 đã lần lượt giới thiệu 4 thuật toán kiểm tra số tính nguyên tố của một số nguyên dương, đó là thuật toán Fermat, thuật toán Solovay-Strassen, thuật toán Miller-Rabin và thuật toán AKS. Như đã phân tích, thuật toán Fermat yếu kém hơn so với các thuật toán khác còn thuật toán AKS lại rất phức tạp và chưa chứng tỏ được tính hiệu quả rõ rệt trong tính toán thực tiễn do bậc đa thức khá cao nên đề tài chỉ tiến hành thử nghiệm hai thuật toán phổ biến còn lại, đó là thuật toán Solovay-Strassen và thuật toán Miller-Rabin.

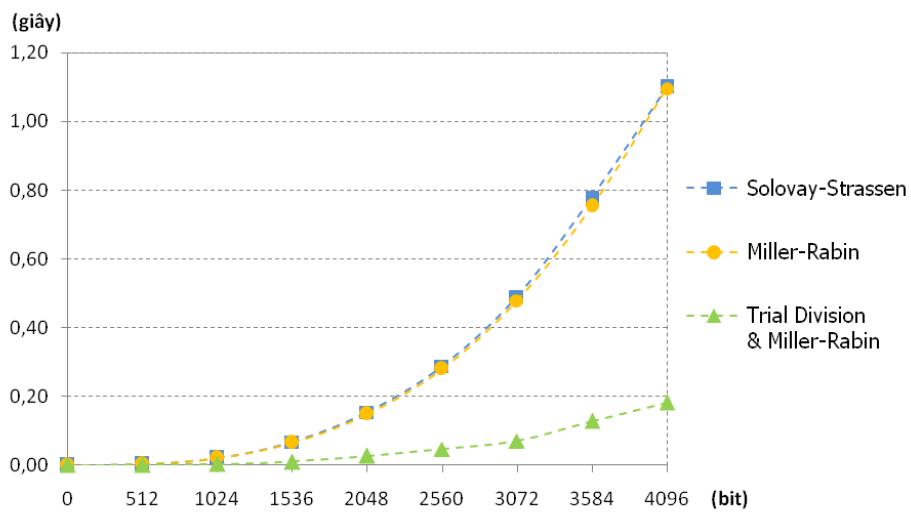
Để đánh giá hiệu quả trong kiểm tra tính nguyên tố của hai thuật toán này, đề tài tiến hành kiểm tra tính nguyên tố trên các hợp số được phát sinh ngẫu nhiên (Thử nghiệm 7.2) và trên các số nguyên tố được phát sinh ngẫu nhiên (Thử nghiệm 7.3).

**Thử nghiệm 7.2:** Độ dài số nguyên cần kiểm tra lần lượt là  $k = 512i$  (bit) với  $1 \leq i \leq 8$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh các hợp số ngẫu nhiên  $k$ -bit  $n$  và lần lượt cho kiểm tra tính nguyên tố với thuật toán Solovay-Strassen (Thuật toán 6.4), Miller-Rabin (Thuật toán 6.5) và chia thử (Trial Division) kết hợp

Miller-Rabin với xác suất kết luận sai  $p_{k,t} \leq \frac{1}{2}^{80}$ . Thử nghiệm được lặp lại 50.000 lần. Kết quả nhận được như sau:

**Bảng 7.2. Thời gian kiểm tra tính nguyên tố với  $p_{k,t} \leq \frac{1}{2}^{80}$  khi thử nghiệm trên hợp số ngẫu nhiên**

| Độ dài<br>(bit) | Thời gian kiểm tra (giây) |                                 |   | Tỷ lệ (%)<br>(2)<br>(1) |
|-----------------|---------------------------|---------------------------------|---|-------------------------|
|                 | Solovay<br>Strassen       | Miller-<br>Rabin <sup>(1)</sup> | Trial Division<br>& Miller-Rabin <sup>(2)</sup> |                         |
| 512             | 0,0030                    | 0,0030                          | 0,0005  | 16,32%                  |
| 1024            | 0,0205                    | 0,0205                          | 0,0032  | 15,39%                  |
| 1536            | 0,0663                    | 0,0658                          | 0,0093  | 14,08%                  |
| 2048            | 0,1530                    | 0,1486                          | 0,0262  | 17,62%                  |
| 2560            | 0,2872                    | 0,2807                          | 0,0460  | 16,40%                  |
| 3072            | 0,4896                    | 0,4781                          | 0,0693  | 14,50%                  |
| 3584            | 0,7801                    | 0,7548                          | 0,1277  | 16,92%                  |
| 4096            | 1,1002                    | 1,0963                          | 0,1814  | 16,55%                  |
| Trung bình      |                           |                                 |   | 15,97%                  |



**Hình 7.2. Thời gian kiểm tra tính nguyên tố với  $p_{k,t} \leq \frac{1}{2}^{80}$  khi thử nghiệm trên hợp số ngẫu nhiên**

Kết quả Thử nghiệm 7.2 cho thấy tốc độ kiểm tra của Miller-Rabin chỉ nhanh hơn Solovay-Strassen một chút khi các số được kiểm tra là các hợp số ngẫu nhiên. Mặc dù thuật toán Solovay-Strassen cần gấp đôi số lần thực hiện để đạt cùng xác suất sai,

cụ thể là  $t = 80$  so với  $t = 40$  của thuật toán Miller-Rabin để cho cùng  $p_{k,t} \leq \frac{1}{2}^{80}$

nhưng do các số được kiểm tra là hợp số nên cả hai thuật toán đều dừng lại ở một số bước xấp xỉ nhau. Hơn nữa, thuật toán Miller-Rabin có chi phí tính toán cao nên nếu trước đó ta sàng lọc bằng các phép chia thử tốn chi phí thấp thì thời gian kiểm tra

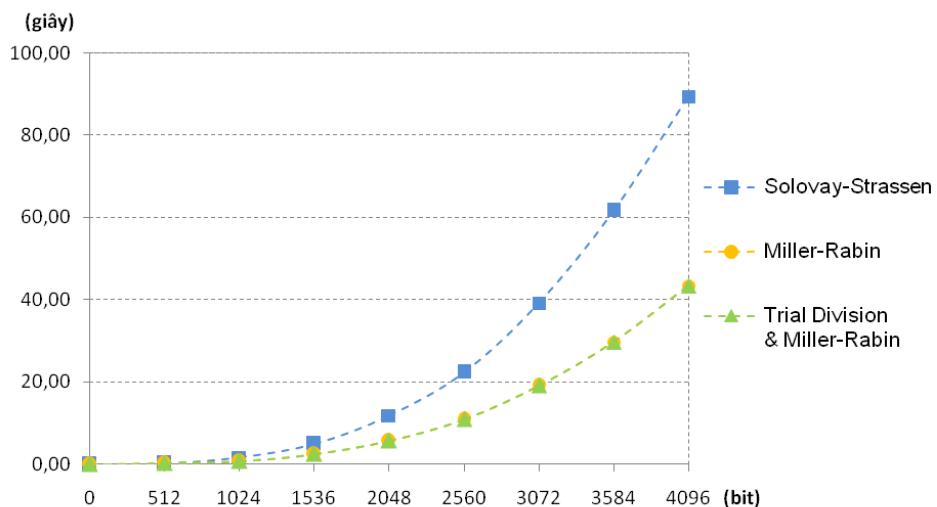
tổng thể sẽ giảm đi đáng kể (chỉ còn trung bình 20,39%) do phần lớn hợp số cần kiểm tra đều không vượt qua được phép chia thử.

**Thử nghiệm 7.3:** Độ dài số nguyên cần kiểm tra lần lượt là  $k = 512i$  (bit) với  $1 \leq i \leq 8$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh số nguyên tố ngẫu nhiên  $k$ -bit  $n$  và lần lượt cho kiểm tra tính nguyên tố với thuật toán Solovay-Strassen (Thuật toán 6.4), Miller-Rabin (Thuật toán 6.5) và Trial Division (chia thử) kết hợp

Miller-Rabin với xác suất kết luận sai  $p_{k,t} \leq \frac{1}{2}^{80}$ . Thử nghiệm được lặp lại 50.000 lần. Kết quả nhận được như sau:

**Bảng 7.3. Thời gian kiểm tra tính nguyên tố với  $p_{k,t} \leq \frac{1}{2}^{80}$  khi thử nghiệm trên số nguyên tố ngẫu nhiên**

| Độ dài<br>(bit) | Thời gian kiểm tra (giây) |                                 |   | Tỷ lệ (%)<br>(2)<br>(1) |
|-----------------|---------------------------|---------------------------------|---|-------------------------|
|                 | Solovay<br>Strassen       | Miller-<br>Rabin <sup>(1)</sup> | Trial Division<br>& Miller-Rabin <sup>(2)</sup> |                         |
| 512             | 0,2596                    | 0,1862                          | 0,1865  | 100,15%                 |
| 1024            | 1,6019                    | 0,7617                          | 0,7622  | 100,06%                 |
| 1536            | 5,0068                    | 2,4185                          | 2,4191  | 100,03%                 |
| 2048            | 11,7446                   | 5,7191                          | 5,7199  | 100,01%                 |
| 2560            | 22,3531                   | 10,9102                         | 10,9112   | 100,01%                 |
| 3072            | 39,1682                   | 19,1196                         | 19,1208   | 100,01%                 |
| 3584            | 61,6045                   | 29,7241                         | 29,7256   | 100,00%                 |
| 4096            | 89,2953                   | 43,3776                         | 43,3793   | 100,00%                 |
| Trung bình      |                           |                                 |   | 100,04%                 |



**Hình 7.3. Thời gian kiểm tra tính nguyên tố với  $p_{k,t} \leq \frac{1}{2}^{80}$  khi thử nghiệm trên số nguyên tố ngẫu nhiên**

Kết quả Thử nghiệm 7.3 cho thấy tốc độ kiểm tra Miller-Rabin tốt hơn nhiều so với kiểm tra Solovay-Strassen. Lý do là số được chọn để kiểm tra là số nguyên tố nên cả hai thuật toán đều phải thực hiện tất cả  $t$  lần thử và do kiểm tra Miller-Rabin chỉ thực hiện ít hơn một nửa và chi phí của Solovay-Strassen cao hơn (do phải tính ký hiệu Jacobi). Nếu sử dụng phép chia thử trước khi kiểm tra với Miller-Rabin thì thời gian tổng thể không chênh lệch nhiều, chỉ chậm hơn 0,04%, do các phép toán chia thử chỉ được thực hiện trên các số nguyên tố nhỏ nên cần rất ít chi phí.

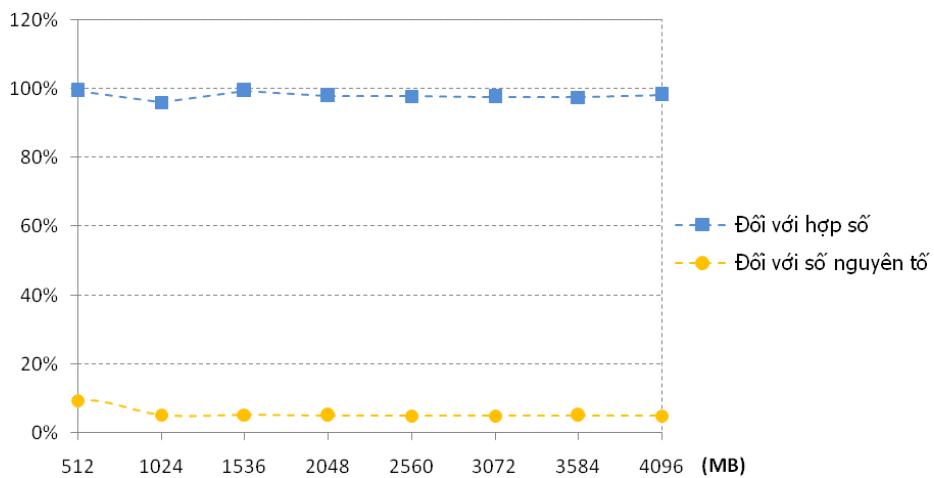
Như vậy, trong cả hai trường hợp số cần kiểm tra là hợp số hay số nguyên tố thì thuật toán kiểm tra Miller-Rabin đều cho hiệu quả vượt trội. Hơn nữa, với việc sử dụng phương pháp kiểm tra chi phí thấp là các phép chia thử trước khi sử dụng kiểm tra tố chí phí Miller-Rabin thì thời gian tổng thể đã cải thiện rất nhiều. Bên cạnh đó, với việc áp dụng các công thức xác suất (đã được trình bày ở mục 6.4.3), thuật toán Miller-Rabin có thể được tối ưu để đạt được cùng xác suất sai nhưng với số lần thử  $t$  rất ít. Thử nghiệm 7.4 sau được thực hiện nhằm chứng minh tính hiệu quả đó.

**Thử nghiệm 7.4:** Độ dài số nguyên cần kiểm tra lần lượt là  $k = 512i$  (bit) với  $1 \leq i \leq 8$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh hợp ngẫu nhiên  $k$ -bit  $n_1$  và số nguyên tố ngẫu nhiên  $k$ -bit  $n_2$  rồi lần lượt cho kiểm tra tính nguyên tố với thuật toán Miller-Rabin gốc và phiên bản tối ưu của nó với cùng xác suất kết luận

sai  $p_{k,t} \leq \frac{1}{2^{80}}$ . Thử nghiệm được lặp lại 50.000 lần. Kết quả nhận được như sau:

**Bảng 7.4. Thời gian kiểm tra của các thuật toán Miller-Rabin với  $p_{k,t} \leq \frac{1}{2^{80}}$**

| Độ dài<br>(bit) | Thời gian kiểm tra (giây) |                          |                         |                          | Tỷ lệ (%) |       |
|-----------------|---------------------------|--------------------------|-------------------------|--------------------------|-----------|-------|
|                 | Hợp số ngẫu nhiên         |                          | Số nguyên tố ngẫu nhiên |                          | (2)       | (4)   |
|                 | MR gốc <sup>(1)</sup>     | MR tối ưu <sup>(2)</sup> | MR gốc <sup>(3)</sup>   | MR tối ưu <sup>(4)</sup> | (1)       | (3)   |
| 512             | 0,0030                    | 0,0030                   | 0,1862                  | 0,0175                   | 99,37%    | 9,39% |
| 1024            | 0,0205                    | 0,0196                   | 0,7617                  | 0,0398                   | 95,94%    | 5,23% |
| 1536            | 0,0658                    | 0,0654                   | 2,4185                  | 0,1254                   | 99,41%    | 5,19% |
| 2048            | 0,1486                    | 0,1456                   | 5,7191                  | 0,2891                   | 98,00%    | 5,05% |
| 2560            | 0,2807                    | 0,2742                   | 10,9102                 | 0,5450                   | 97,70%    | 5,00% |
| 3072            | 0,4781                    | 0,4669                   | 19,1196                 | 0,9531                   | 97,65%    | 4,98% |
| 3584            | 0,7548                    | 0,7361                   | 29,7241                 | 1,4904                   | 97,53%    | 5,01% |
| 4096            | 1,0963                    | 1,0768                   | 43,3776                 | 2,1676                   | 98,21%    | 5,00% |
| Trung bình      |                           |                          |                         |                          | 97,98%    | 5,61% |



**Hình 7.4. Tỷ lệ thời gian kiểm tra giữa thuật toán Miller-Rabin cải tiến**

và thuật toán Miller-Rabin gốc với  $p_{k,t} \leq \frac{1}{2}^{80}$

Kết quả Thử nghiệm 7.4 cho thấy, khi áp dụng công thức tính số phép thử  $t$  tối ưu để kiểm tra mà vẫn đạt được xác suất sai  $p_{k,t} \leq \frac{1}{2}^{80}$  thì tốc độ đạt được tốt hơn rất nhiều, trung bình khoảng 5,61% khi thử nghiệm trên các số nguyên tố ngẫu nhiên và tốt hơn một chút, trung bình 97,98% khi thử nghiệm trên các hợp số ngẫu nhiên. Như vậy, thuật toán Miller-Rabin đã chứng tỏ được lý do tại sao là thuật toán kiểm tra tính nguyên tố theo xác suất được sử dụng hiệu quả nhất hiện nay. Với việc kết hợp với phương pháp chi phí thấp là chia thử (Trial Division) và tối ưu số lần lặp  $t$ , thuật toán kiểm tra Miller-Rabin đã mang lại hiệu quả thực hiện ấn tượng.

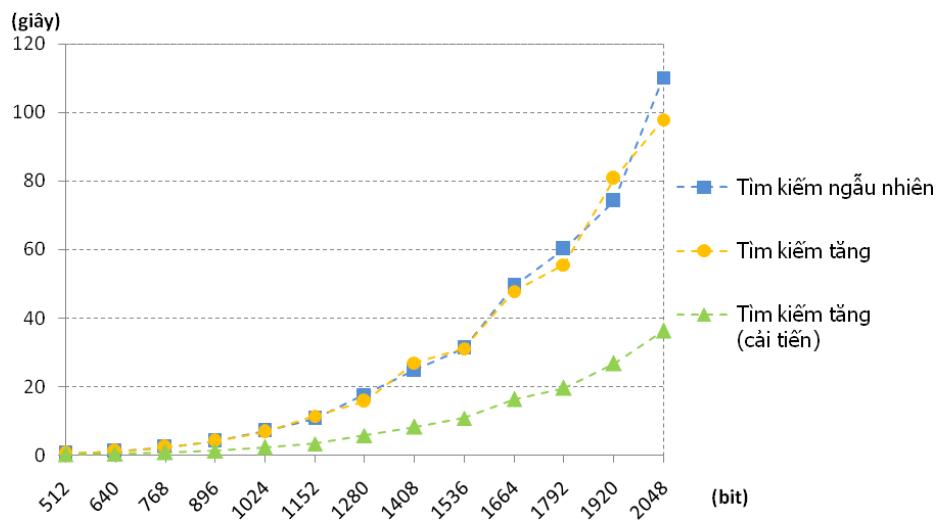
#### 7.4.3 Các thuật toán phát sinh số nguyên tố

Mục 6.5 đã lăn lướt giới thiệu các thuật toán phát sinh số khả nguyên tố và số nguyên tố. Để thử nghiệm tính hiệu quả của các thuật toán này, các thử nghiệm dưới đây đã được tiến hành và ghi nhận.

**Thử nghiệm 7.5:** Độ dài số nguyên cần phát sinh lần lượt là  $k = 512 + 128i$  (bit) với  $0 \leq i \leq 12$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh số nguyên tố ngẫu nhiên  $k$ -bit  $n$  bằng các thuật toán tìm kiếm ngẫu nhiên (Thuật toán 6.6), tìm kiếm tăng (Thuật toán 6.7) và tìm kiếm tăng cải tiến (Thuật toán 6.8). Thử nghiệm được lặp lại 50.000 lần. Kết quả nhận được như sau:

**Bảng 7.5. Thời gian phát sinh số khả nguyên tố ngẫu nhiên**

| Độ dài<br>(bit) | Thời gian phát sinh (giây) |                              |  | Tỷ lệ (%) |
|-----------------|----------------------------|------------------------------|--|-----------|
|                 | Tìm kiếm<br>ngẫu nhiên     | Tìm kiếm tăng <sup>(1)</sup> | Tìm kiếm tăng<br>(cải tiến) <sup>(2)</sup> |           |
| (1)             |                            |                              |  |           |
| 512             | 0,5517                     | 0,5369                       | 0,1855                                     | 34,56%    |
| 640             | 1,1683                     | 1,1265                       | 0,3989                                     | 35,41%    |
| 768             | 2,3451                     | 2,2969                       | 0,7658                                     | 33,34%    |
| 896             | 4,1388                     | 4,0435                       | 1,3090                                     | 32,37%    |
| 1024            | 7,0211                     | 6,7816                       | 2,2558                                     | 33,26%    |
| 1152            | 10,6597                    | 11,3729                      | 3,3687                                     | 29,62%    |
| 1280            | 17,6192                    | 15,7234                      | 5,7813                                     | 36,77%    |
| 1408            | 24,8770                    | 26,9937                      | 8,3361                                     | 30,88%    |
| 1536            | 31,2799                    | 31,0431                      | 10,8376                                    | 34,91%    |
| 1664            | 49,4430                    | 47,7730                      | 16,3048                                    | 34,13%    |
| 1792            | 60,1589                    | 55,4910                      | 19,5931                                    | 35,31%    |
| 1920            | 74,1441                    | 80,7833                      | 26,7150                                    | 33,07%    |
| 2048            | 110,1213                   | 97,8716                      | 36,3573                                    | 37,15%    |
| Trung bình      |                            |                              | 33,91%                                     |           |

**Hình 7.5. Thời gian phát sinh số khả nguyên tố ngẫu nhiên**

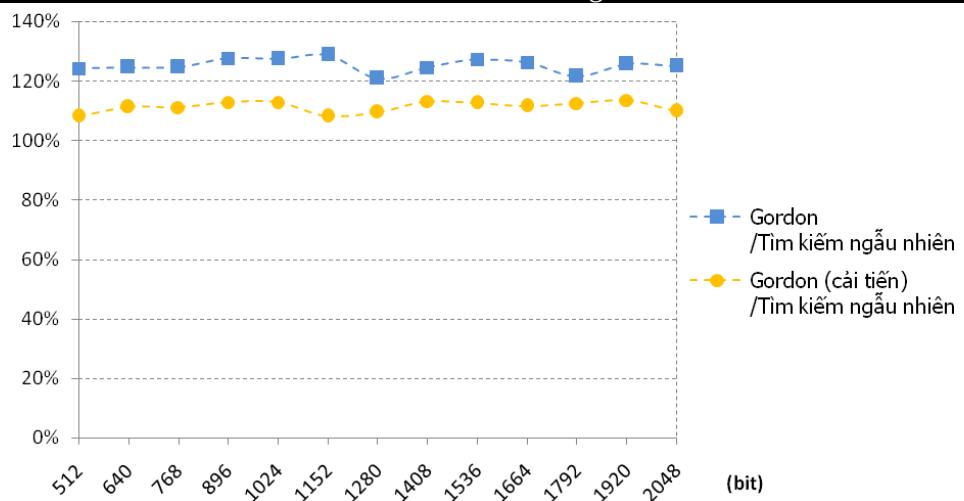
Kết quả Thử nghiệm 7.5 cho thấy hai thuật toán phát sinh số nguyên tố ngẫu nhiên là theo cách tìm kiếm ngẫu nhiên và tìm kiếm tăng xấp xỉ nhau. Tuy nhiên, biến thể cải tiến của thuật toán tìm kiếm tăng đã mang lại hiệu quả thực hiện rất ấn tượng, chỉ mất khoảng 33,91% thời gian thực hiện so với thuật toán gốc.

Để đánh giá hiệu quả của thuật toán phát sinh số nguyên tố mạnh của Gordon (phiên bản gốc và phiên bản cải tiến) với thuật toán phát sinh số nguyên tố ngẫu nhiên theo cách trên, Thử nghiệm 7.6 sau đã được tiến hành và ghi nhận.

**Thử nghiệm 7.6:** Độ dài số nguyên cần phát sinh lần lượt là  $k = 512 + 128i$  (bit) với  $0 \leq i \leq 12$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh số nguyên tố mạnh  $k$ -bit  $n$  bằng thuật toán Gordon (Thuật toán 6.11) và thuật toán Gordon cải tiến (sử dụng Thuật toán 6.8 trong việc tìm số nguyên tố ngẫu nhiên). Thử nghiệm được lặp lại 10.000 lần. Kết quả nhận được như sau:

**Bảng 7.6. Thời gian phát sinh số nguyên tố mạnh bằng thuật toán Gordon**

| Độ dài<br>(bit) | Thời gian phát sinh (giây)            |                                |                                     | Tỷ lệ (%)  |            |
|-----------------|---------------------------------------|--------------------------------|-------------------------------------|------------|------------|
|                 | Tìm kiếm<br>ngẫu nhiên <sup>(1)</sup> | Gordon<br>(gốc) <sup>(2)</sup> | Gordon<br>(cải tiến) <sup>(3)</sup> | (2)<br>(1) | (3)<br>(1) |
| 512             | 0,5517                                | 0,6849                         | 0,5967                              | 124,14%    | 108,16%    |
| 640             | 1,1683                                | 1,4559                         | 1,3002                              | 124,62%    | 111,29%    |
| 768             | 2,3451                                | 2,9255                         | 2,6059                              | 124,75%    | 111,12%    |
| 896             | 4,1388                                | 5,2824                         | 4,6659                              | 127,63%    | 112,74%    |
| 1024            | 7,0211                                | 8,9476                         | 7,9207                              | 127,44%    | 112,81%    |
| 1152            | 10,6597                               | 13,7220                        | 11,5368                             | 128,73%    | 108,23%    |
| 1280            | 17,6192                               | 21,3072                        | 19,2927                             | 120,93%    | 109,50%    |
| 1408            | 24,8770                               | 30,9903                        | 28,1361                             | 124,57%    | 113,10%    |
| 1536            | 31,2799                               | 39,7951                        | 35,2481                             | 127,22%    | 112,69%    |
| 1664            | 49,4430                               | 62,4022                        | 55,1786                             | 126,21%    | 111,60%    |
| 1792            | 60,1589                               | 73,0971                        | 67,6635                             | 121,51%    | 112,47%    |
| 1920            | 74,1441                               | 93,2537                        | 84,1233                             | 125,77%    | 113,46%    |
| 2048            | 110,1213                              | 137,5903                       | 120,9451                            | 124,94%    | 109,83%    |
| Trung Bình      |                                       |                                |                                     | 125,27%    | 111,31%    |



**Hình 7.6. Tỷ lệ thời gian phát sinh số nguyên tố của thuật toán Gordon và thuật toán tìm kiếm ngẫu nhiên**

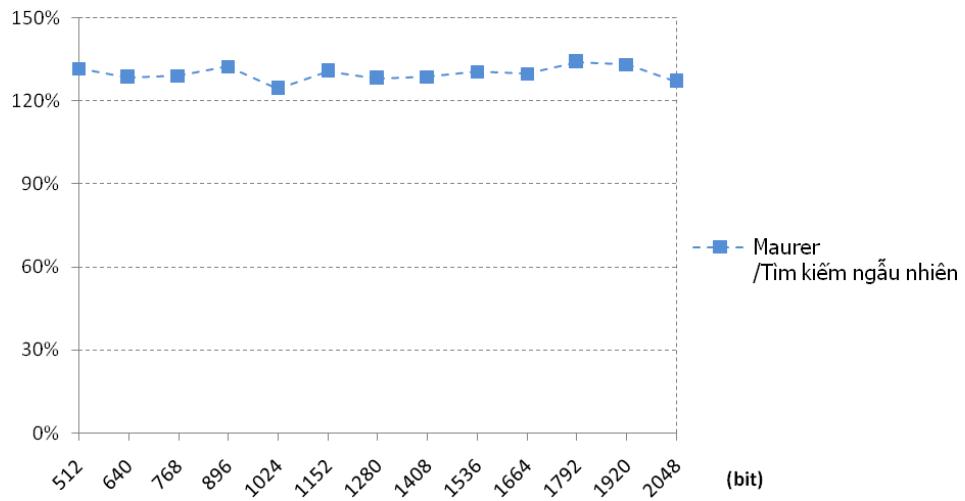
Trong công trình của mình, Gordon đã chứng minh trên lý thuyết thuật toán phát sinh số nguyên tố mạnh của ông chỉ chậm hơn thuật toán phát sinh số nguyên tố ngẫu nhiên theo cách tìm kiếm ngẫu nhiên trung bình 19% nhưng kết quả Thử nghiệm 7.6 cho thấy thực tế chậm hơn đến 25.27%. Nguyên nhân là do trong quá trình phát sinh ta phải điều chỉnh các kích thước tham số để đạt được độ dài số nguyên tố cuối cùng như mong đợi nên thời gian này đã tăng lên. Tuy nhiên, bằng cách sử dụng thuật toán tìm kiếm tăng cải tiến thay thế cho thuật toán tìm kiếm tăng theo mô tả gốc của Gordon, tốc độ của thuật toán Gordon đã cải thiện đáng kể. Thử nghiệm 7.6 cho thấy nó chỉ chậm hơn thuật toán tìm kiếm ngẫu nhiên trung bình 11,31%.

Để đánh giá tính hiệu quả của thuật toán Maurer (phát sinh số nguyên tố thực sự), Thử nghiệm 7.7 sau đã được tiến hành và ghi nhận.

**Thử nghiệm 7.7:** Độ dài số nguyên cần phát sinh lần lượt là  $k = 512 + 128i$  (bit) với  $0 \leq i \leq 12$ . Ứng với mỗi độ dài  $k$ , chương trình tự động phát sinh số nguyên tố  $k$ -bit  $n$  bằng thuật toán Maurer (Thuật toán 6.12). Thử nghiệm được lặp lại 10.000 lần. Kết quả thử nghiệm như sau:

**Bảng 7.7. Thời gian phát sinh số nguyên tố bằng thuật toán Maurer**

| Độ dài<br>(bit) | Thời gian phát sinh (giây)            |                       | Tỷ lệ (%)<br>(2)<br>(1) |
|-----------------|---------------------------------------|-----------------------|-------------------------|
|                 | Tìm kiếm<br>ngẫu nhiên <sup>(1)</sup> | Maurer <sup>(2)</sup> |                         |
| 512             | 0,5517                                | 0,7271                | 131,79%                 |
| 640             | 1,1683                                | 1,5013                | 128,50%                 |
| 768             | 2,3451                                | 3,0293                | 129,18%                 |
| 896             | 4,1388                                | 5,4785                | 132,37%                 |
| 1024            | 7,0211                                | 8,7353                | 124,42%                 |
| 1152            | 10,6597                               | 13,9478               | 130,85%                 |
| 1280            | 17,6192                               | 22,5780               | 128,14%                 |
| 1408            | 24,8770                               | 32,0296               | 128,75%                 |
| 1536            | 31,2799                               | 40,8527               | 130,60%                 |
| 1664            | 49,4430                               | 64,2260               | 129,90%                 |
| 1792            | 60,1589                               | 80,6918               | 134,13%                 |
| 1920            | 74,1441                               | 98,7540               | 133,19%                 |
| 2048            | 110,1213                              | 139,8584              | 127,00%                 |
| Trung Bình      |                                       |                       | 129,91%                 |



**Hình 7.7. Tỷ lệ thời gian phát sinh số nguyên tố  
giữa thuật toán Maurer và thuật toán tìm kiếm ngẫu nhiên**

Kết quả Thử nghiệm 7.7 cho thấy thuật toán Maurer chậm hơn rất nhiều so với thuật toán tìm kiếm ngẫu nhiên trung bình khoảng 29,91% và tất nhiên cũng chậm hơn thuật toán Gordon. Ngoài ra, do thuật toán Maurer sử dụng đệ quy nên nó cần nhiều bộ nhớ để thực hiện hơn. Điểm mạnh duy nhất của thuật toán này là nó tạo ra được số nguyên tố thật sự. Tuy nhiên, trong thực tế các số khả nguyên tố hay các số nguyên tố xác suất hay thường được sử dụng do nó mang đến độ an toàn cao hơn và thời gian phát sinh nhanh hơn.

## 7.5 Kết luận

Trên cơ sở các phân tích về nguy cơ tổn thương và cách khắc phục ở Chương 5, các nghiên cứu và giải quyết các bài toán về cài đặt hiệu quả ở Chương 6, đề tài đã xây dựng được một bộ thư viện hỗ trợ cài đặt hệ mã RSA đạt độ an toàn và hiệu quả cần thiết. Ngoài ra, các thử nghiệm ở cuối chương này cũng đã chứng minh tính hiệu quả của các thuật toán, phù hợp với các phân tích. Bộ thư viện đã đáp ứng được các yêu cầu về độ an toàn, hiệu quả cũng như tính khả chuyển, độc lập môi trường.

## Chương 8

# Cài tiến và triển khai hệ thống chứng thực khóa công khai sử dụng gói phần mềm mã nguồn mở EJBCA

☐ Nội dung của chương này giới thiệu gói phần mềm mã nguồn mở EJBCA, gói phần mềm cho phép triển khai một hệ thống PKI hoàn chỉnh và đầy đủ chức năng. Nhằm tận dụng các tính chất ưu việt của gói phần mềm này cũng như kiểm soát được quá trình phát triển và độ an toàn của hệ thống, để tài đã tiến hành phân tích, cải tiến và triển khai thử nghiệm một hệ thống chứng thực tập trung theo kiến trúc PKI phân cấp đơn giản, có thể sử dụng ngay trong thực tế.

## 8.1 Gói phần mềm mã nguồn mở EJBCA

### 8.1.1 Giới thiệu

Kiến trúc Enterprise Java Beans (EJB) là một đặc tả được công ty Sun Microsystems phát triển. EJB mô tả một kiến trúc dựa trên thành phần cho việc phát triển và triển khai các ứng dụng phân tán, cho phép các ứng dụng doanh nghiệp có thể mở rộng, an toàn và có thể giao tác.

EJB là các thành phần thực thi bên trong “khung chứa EJB” (EJB container), dưới sự giám sát của một máy chủ ứng dụng (như JBOSS<sup>19</sup>). Máy chủ ứng dụng và khung chứa EJB cung cấp các dịch vụ hệ thống cho EJB như tính bền vững dữ liệu, giao tác, bảo mật và quản lý tài nguyên. EJB là phần cốt lõi của ứng dụng J2EE<sup>20</sup>. Khung chứa EJB duy trì các kết nối dữ liệu dùng chung cũng như các thực thể EJB dùng chung được cung cấp cho người dùng khi cần.

<sup>19</sup> Máy chủ ứng dụng J2EE được sử dụng rộng rãi nhất hiện nay.

<sup>20</sup> J2EE (Java 2 Enterprise Edition) là một nền lập trình, một phần của nền Java, để phát triển và chạy các ứng dụng Java phân tán đa kiến trúc, phần lớn dựa trên môđun các thành phần phần mềm chạy trên một máy chủ ứng dụng.

EJBCA là một CA đầy đủ chức năng được xây dựng trên Java. Do được dựa trên công nghệ J2EE, EJBCA tạo thành một CA mạnh, hiệu suất cao và dựa trên thành phần. Với sự mềm dẻo và độc lập môi trường nền, EJBCA có thể được sử dụng độc lập hoặc được tích hợp trong các ứng dụng J2EE. EJBCA là một sản phẩm của công ty PrimeKey, một trong các công ty PKI mã nguồn mở đứng đầu trên thế giới, được thành lập năm 2002 tại Stockholm, Thụy Điển. PrimeKey cung cấp các sản phẩm PKI và thẻ thông minh, các giải pháp liên quan và các dịch vụ chuyên nghiệp.

EJBCA trải qua các giao đoạn phát triển như sau:

- Phiên bản 1.x bắt đầu như một bản beta trên SourceForge vào tháng 11/2001. Ý tưởng của EJBCA là thực thi một CA bên trong một máy chủ ứng dụng J2EE. Phiên bản 1.0-1.4 cung cấp các hỗ trợ đối với Jboss, WebLogic, CRL, LDAP, MySQL, PostgreSQL, Oracle.
- Phiên bản 2.x lấy kinh nghiệm từ phiên bản 1.x và được bắt đầu từ tháng 3/2003. Phiên bản này cung cấp các hỗ trợ đối với thẻ từ, PIN/PUK, phục hồi khóa, trạng thái chứng nhận, OCSP, SCEP, các tính năng đặc biệt cho AD và Outlook, OpenLDAP.
- Phiên bản 3.x bắt đầu từ tháng 6/2004, cung cấp các hỗ trợ đối với CA ảo, kiểm tra JUnit, hỗ trợ HSM (nCipher, Luna/Eracom/SafeNet), ngôn ngữ (Tây Ban Nha, Pháp, Ý, Trung Quốc, Thụy Điển, Đức), OCSP Responder bên ngoài, Infomix, OpenVPN, RA API ngoài, CMP, XKMSv2, các dịch vụ theo dõi, ECDSA, các mở rộng chứng nhận tùy thích, DN và altName OIDs.

EJBCA là phần mềm mở nguồn mở, hỗ trợ rất nhiều chức năng. Tính đến 6/10/2008, phiên bản 3.x đã có hơn 47.600 lượt tải về<sup>21</sup>. EJBCA thực sự đã trở thành một sản phẩm toàn diện cho các giải pháp PKI/CA thay thế cho mọi ứng sản phẩm khác.

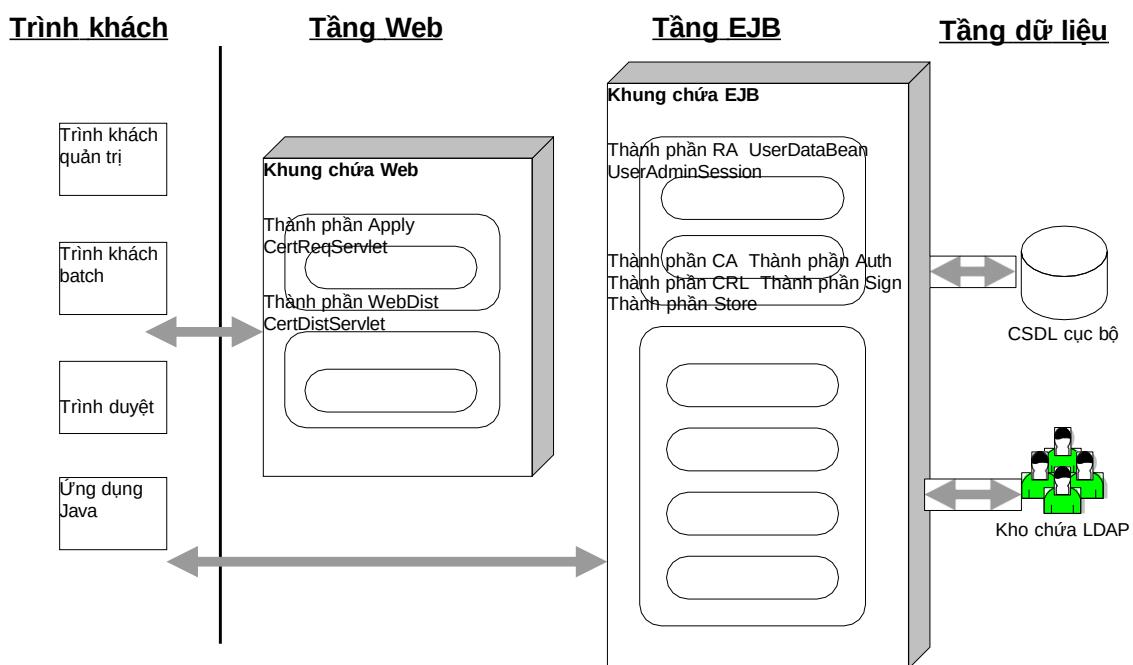
---

<sup>21</sup> Nguồn <http://sourceforge.net>

### 8.1.2 Kiến trúc

Kiến trúc của EJBCA bao gồm các thành phần sau:

- **Tầng dữ liệu (Data Tier):** Tầng dữ liệu lưu trữ các chứng nhận, CRL cũng như các thực thể cuối. EJBCA sử dụng một cơ sở dữ liệu mặc định để lưu trữ các thực thể cuối. Các chứng nhận được lưu trữ trong một kho chứa LDAP (Lightweight Directory Access Protocol).



**Hình 8.1. Kiến trúc EJBCA**

- **Thành phần CA:** Thành phần có chức năng tạo các CA gốc, CA con, chứng nhận, CRL và giao tiếp với kho chứa LDAP để lưu trữ thông tin chứng nhận.
- **Thành phần RA:** Thành phần có chức năng tạo, xóa và hủy bỏ người dùng. Nó giao tiếp với cơ sở dữ liệu cục bộ để chứa thông tin người dùng.
- **Tầng Web:** Đây là giao diện (điển hình là giao diện người – máy bằng đồ họa) để trình khách tương tác với hệ thống EJBCA, đồng thời quy định các cấp độ và phạm vi truy cập thông tin khác nhau cho thực thể cuối.
- **Trình khách:** Trình khách là thực thể cuối hay người sử dụng như trình khách thư điện tử, máy chủ web, trình duyệt web hay cổng VPN. Các thực thể cuối không được phép phát hành chứng nhận đến các thực thể khác, nói cách khác chúng là các nút lá trong PKI.

### **8.1.3 Chức năng**

EJBCA là một tổ chức chứng nhận rất phổ biến hiện đang được sử dụng, một trong những CA được ưa thích hiện nay. Các đặc trưng cơ bản của CA này bao gồm sự lựa chọn của thuật toán ta cần như tùy chọn chọn giữa các thuật toán SHA1 hay SHA- 256 với RSA và với các kích thước khóa khác nhau như 1024, 2048 và 4096.

EJBCA cung cấp một số tính năng nổi bật về lựa chọn ngôn ngữ trong quá trình cấu hình hệ thống. Ngoài ra ta cũng có thể chọn loại publisher chúng ta muốn như LDAP, thư mục động (AD – Active Directory) hay một kết nối publisher tự làm.

Sự phát hành của chứng nhận luôn thuộc chuẩn X509. Cũng có một tùy chọn được cung cấp để chọn loại khóa ký – soft hay hard. Việc ký chứng nhận có thể là tự ký (self-signed), CA bên ngoài (external CA) hay CA quản trị (admin CA).

CA gốc có khóa RSA độ dài mặc định là 2048 bit và có hiệu lực 10 năm. Việc đăng ký chứng nhận trong EJBCA cung cấp cho người sử dụng nhiều lựa chọn như người sử dụng có thể chọn nhà cung cấp dịch vụ mã hóa (Cryptographic Service Provider – CSP<sup>22</sup>) mà họ thích và có thể chọn kích thước khóa khác nhau được cung cấp như 512, 1024 và 2048. Nó cũng cung cấp cho người sử dụng những tùy chọn của việc thêm chứng nhận vào thẻ nhận dạng điện tử (Electronic Identity Card).

### **8.1.4 So sánh với các gói phần mềm khác**

Ngoài EJBCA còn có các sản phẩm khác có thể triển khai hệ thống PKI hoàn chỉnh như OpenCA và Windows 2003 Server CA. Do Windows 2003 Server CA không phải là sản phẩm mã nguồn mở, không thể tự do phát triển cũng như kiểm soát được quá trình phát triển và độ an toàn nên không được quan tâm tìm hiểu.

EJBCA và OpenCA đều là các dự án PKI mã nguồn mở mạnh và hiện cũng có nhiều phát triển đang được thực hiện trên cả hai phần mềm này.

---

<sup>22</sup> Nhà cung cấp dịch vụ mã hóa (Cryptographic Service Provider – CSP) là một thư viện phần mềm cài đặt các giao tiếp chương trình ứng dụng mã hóa (Cryptographic Application Programming Interface – CAPI).

Dưới đây là bảng so sánh một số đặc điểm giữa hai gói phần mềm này [23, tr.12].

**Bảng 8.1. So sánh các đặc điểm của EJBCA và OpenCA**

| Đặc điểm                            | EJBCA  | OpenCA  |
|-------------------------------------|--|---|
| Độ khó khi cấu hình                 | Rất phức tạp   | Phức tạp  |
| Tính cẩn mật                        | Có (sử dụng mã hóa)  | Có (sử dụng mã hóa)                                     |
| Tính toàn vẹn                       | Có (sử dụng mã hóa)  | Có (sử dụng mã hóa)                                     |
| Tính xác thực                       | Có (sử dụng chữ ký số)   | Có (sử dụng chữ ký số)                                  |
| Tính không thể chối từ              | Có   | Có  |
| Khả năng chọn thuật toán để sử dụng | Có   | Có  |
| OCSP <sup>23</sup>                  | Có   | Không   |
| Khả năng chọn CSP                   | Có   | Không   |
| Cập nhật CRL                        | Tự động  | Bằng tay  |
| Hỗ trợ thẻ thông minh               | Có   | Không   |
| Chi phí                             | Miễn phí   | Miễn phí  |
| Các mở rộng                         | Có   | Có  |
| Môi trường nền                      | Java J2EE (độc lập nền)  | Perl CGI trên Unix                                      |
| Cơ sở dữ liệu                       | Hypersonic, PostgreSQL, MySQL, MS SQL, Oracle, Sybase, Informix, DB2 | MySQL   |
| Hỗ trợ LDAP                         | Có   | Có  |
| Môđun                               | EJB  | Perl  |
| Dựa trên thành phần                 | Có   | Có  |
| Khả năng mở rộng                    | Được thiết kế và có thể mở rộng                                      | Mở rộng khó với độ phức tạp tăng rất nhiều              |
| Thành phần độc lập                  | PKI có thể được quản trị hoàn toàn thông qua dòng lệnh               | Chỉ có một cách quản trị PKI là thông qua giao diện web |
| Các trình duyệt được hỗ trợ         | Nhiều  | Nhiều   |

### 8.1.5 Lý do chọn gói phần mềm mã nguồn mở EJBCA

Nhằm kiểm soát được quá trình phát triển, độ an toàn và có thể tiếp tục phát triển hệ thống, đề tài đã chọn phần mềm mã nguồn mở để tập trung nghiên cứu thay vì phần mềm đóng như hệ thống CA của Windows Server 2003/2008.

---

<sup>23</sup> Giao thức trạng thái chứng nhận trực tuyến (Online Certificate Status Protocol – OCSP) là một chuẩn Internet được sử dụng để lấy trạng thái thu hồi của chứng nhận số X.509.

Hai gói phần mềm mã nguồn mở phổ biến hiện nay là EJBCA và OpenCA đều có khả năng triển khai hệ thống PKI hoàn chỉnh, phục vụ cho các đối tượng sử dụng khác nhau bao gồm cá nhân và doanh nghiệp. Các tiêu chí quyết định của một hệ thống PKI là phải đáng tin cậy, an toàn, linh hoạt và có hiệu quả kinh tế. Như đã so sánh ở mục 8.1.4, OpenCA chỉ đảm bảo tính tin cậy và an toàn còn EJBCA đảm bảo tất cả các tiêu chí trên.

EJBCA là một CA và là một hệ thống quản lý PKI hoàn chỉnh, là một giải pháp PKI rất mạnh, độc lập môi trường, hiệu suất cao, có thể mở rộng và dựa trên thành phần. Ngoài ra, EJBCA rất linh hoạt trong việc cung cấp các cách thức hoạt động tùy chọn như một CA độc lập hoặc được tích hợp hoàn toàn trong ứng dụng thương mại bất kỳ. Hơn nữa, tuy việc cấu hình hệ thống EJBCA phức tạp hơn OpenCA rất nhiều nhưng hệ thống EJBCA khi đã đi vào hoạt động lại mang đến rất nhiều tiện lợi và đơn giản cho người sử dụng trong việc phát sinh và quản lý chứng nhận. Ngoài ra, khác với OpenCA, việc cập nhật CRL trong EJBCA hoàn toàn tự động.

Ngoài ra, EJBCA được phát triển và cung cấp bởi PrimeKey, một công ty PKI mã nguồn mở đứng đầu trên thế giới nên với việc sử dụng EJBCA ta có thể thừa hưởng từ năng lực phát triển của công ty và hoàn toàn yên tâm về tính an toàn luôn có trong mã nguồn.

## **8.2 Cải tiến gói phần mềm mã nguồn mở EJBCA**

### **8.2.1 Nhu cầu**

Như đã giới thiệu và phân tích ở phần trên, EJBCA là một gói phần nổi tiếng, hỗ trợ đầy đủ chức năng để triển khai một hệ thống PKI đáng tin cậy, an toàn, linh hoạt và dễ mở rộng. Tuy nhiên, để có thể kiểm soát được quá trình phát triển cũng như độ an toàn của hệ thống khi được đưa vào sử dụng trong thực tế, gói phần mềm này cần phải được khảo sát, phân tích cũng như cải tiến nếu có thể để phù hợp với nhu cầu của tổ chức đồng thời đạt được độ an toàn và hiệu quả cần thiết.

Phản tiếp theo sẽ trình bày các phân tích nhằm cải tiến độ an toàn của EJBCA, đặc biệt trong chữ ký số với hệ mã khóa công khai RSA.

### 8.2.2 Cải tiến bộ sinh khóa RSA của EJBCA

EJBCA sử dụng gói thư viện mã hóa mã nguồn mở Bouncy Castle (gọi tắt là BC) trong mọi quy trình mã hóa và giao thức của mình nhằm mang lại tính cẩn mật, toàn vẹn, xác thực và không thể chối từ. Gói thư viện mã hóa này là một thực thi Java của các thuật toán mã hóa, được phát triển bởi công ty Legion of the Bouncy Castle. Gói thư viện này được tổ chức nhằm cung cấp một giao tiếp lập trình ứng dụng (Application Program Interface – API) “gọn nhẹ” (light-weight) phù hợp cho việc sử dụng trong bất kỳ môi trường nào (bao gồm cả phiên bản J2EE mới nhất) với hạ tầng bổ sung để các thuật toán phù hợp với cơ cấu mở rộng mã hóa Java (Java Cryptography Extension – JCE).

API mã hóa của Bouncy Castle cho Java bao gồm các phần sau:

- Một API mã hóa gọn nhẹ cho Java và C#.
- Một provider cho JCE và kiến trúc mã hóa Java (Java Cryptography Architecture – JCA).
- Một thư viện cho việc đọc và ghi các đối tượng ASN.1 được mã hóa.
- Một API TLS<sup>24</sup> phía trình khách “gọn nhẹ”.
- Các bộ phát sinh cho chứng nhận X.509 phiên bản 1 và 3, CRL phiên bản 2 và các tập tin PKCS #12.
- Các bộ phát sinh cho chứng nhận thuộc tính X.509 phiên bản 2.
- Các bộ phát sinh/xử lý cho S/MIME và CMS (PKCS #7/RFC 3852).
- Các bộ phát sinh/xử lý cho OCSP (RFC 2560).
- Các bộ phát sinh/xử lý cho TSP (RFC 3161).
- Các bộ phát sinh/xử lý cho OpenPGP (RFC 2440).
- Một phiên bản jar được ký phù hợp cho JDK 1.4-1.6 và Sun JCE.

API nhỏ gọn làm việc với mọi thứ từ J2ME đến JDK 1.6 và cũng có một API trong C# cung cấp hầu hết những chức năng tương tự như trên.

Như đã trình bày ở Chương 2, đề tài này quan tâm đến hệ mã khóa công khai RSA và các ứng dụng của nó trong mã hóa và chữ ký số nên các hàm liên quan đến hệ mã

<sup>24</sup> TLS (Transport Layer Security) là giao thức mật mã cung cấp các giao tiếp an toàn trên Internet như cho trình duyệt web, thư điện tử, gửi tin nhắn tức thời, trao đổi dữ liệu, ... Tiền thân của TLS chính là giao thức SSL (Secure Sockets Layer).

RSA được đặc biệt chú ý. Hàm sinh khóa genKeys của EJBCA trong lớp KeyTool thuộc gói org.ejbca.util như sau:

```
public static KeyPair genKeys(String keySpec, String keyAlg)
    throws NoSuchAlgorithmException, NoSuchProviderException,
    InvalidAlgorithmParameterException {
    ...
    KeyPairGenerator keygen = KeyPairGenerator.getInstance(keyAlg, "BC");
    ...
    // RSA keys
    int keysize =
        Integer.parseInt(keySpec);
    keygen.initialize(keysize);
    ...
    KeyPair keys = keygen.generateKeyPair();
    ...
    return keys;
```

### Hình 8.2. Hàm phát sinh khóa RSA của EJBCA

Ta thấy biến keygen có kiểu KeyPairGenerator (thuộc gói java.security) sẽ nhận thực thể của provider BC nếu được. Nếu gói thư viện BC này chưa được cài đặt, nó sẽ lấy thực thể mặc định của Java. Đây là thao tác kiểm tra trong trường hợp người sử dụng quên cài đặt gói thư viện BC này.

Lệnh keygen.generateKeyPair nhằm phát sinh cặp khóa. Khi thuật toán được chọn là RSA, hàm RSAKeyPairGenerator của BC (lớp RSAKeyPairGenerator thuộc gói org.bouncycastle.crypto.generators) sẽ được thực hiện. Thuật toán phát sinh cặp khóa RSA được hàm này sử dụng như sau:

#### **RSAKeyPairGenerator(e, strength)**

**Đầu vào:** số nguyên  $e$  là số mũ công khai,  $strength$  là độ dài khóa.

**Đầu ra:** cặp khóa công khai  $n, e$  và bí mật  $n, d$ .

- (1)  $pBitLength \leftarrow (strength + 1)/2$ .
- (2)  $qBitLength \leftarrow strength - pBitLength$ .
  - (3) Chọn một số nguyên ngẫu nhiên  $p$ , độ dài  $pBitLength$ .
  - (4) Nếu  $p$  không là số nguyên tố hoặc  $gcd(e, p) \neq 1$  thì trở lại bước (3).
  - (5) Chọn một số nguyên ngẫu nhiên  $q$ , độ dài  $qBitLength$ .
- (6) Nếu  $q$  không là số nguyên tố hoặc  $gcd(e, q) \neq 1$  hoặc độ dài của  $p \times q$  khác  $strength$  thì quay lại bước (5).
- (7)  $n \leftarrow p \times q$ .
- (8)  $p\#i \leftarrow p - 1 \times (q - 1)$ .
- (9)  $d \leftarrow e^{-1} \bmod p\#i$ .
- (10) Trả về  $(n, e)$  và  $(n, d)$ .

### Thuật toán 8.1. Phát sinh cặp khóa RSA trong Bouncy Castle

Theo các phân tích ở Chương 5, các số nguyên tố  $p$  và  $q$  được sinh ra ở bước (3) và (5) trong Thuật toán 8.1 trên nên là các số nguyên tố mạnh (strong prime) thay vì các số nguyên tố ngẫu nhiên nhằm tránh các phương pháp tấn công phân tích đặc biệt. Vì vậy, phần sinh khóa của RSA sẽ được thay bằng phần sinh khóa của thư viện mã hóa SmartRSA được xây dựng và giới thiệu ở Chương 7.

### 8.2.3 Nhận xét

Bouncy Castle là gói phần mềm mã hóa mã nguồn mở cung cấp các chức năng nhằm mang lại tính cẩn mật, toàn vẹn, xác thực và không thể chối từ cho bất kỳ hệ thống nào sử dụng nó, điển hình là hệ thống EJBCA. Với việc cài tiến hàm sinh khóa RSA của gói phần mềm mã hóa này bằng hàm sinh khóa mạnh của bộ thư viện SmartRSA đã được trình bày ở Chương 7, hệ thống EJBCA sẽ mang lại độ an toàn và hiệu quả cao hơn khi đi vào sử dụng trong thực tế.

## 8.3 Triển khai hệ thống

### 8.3.1 Mục tiêu

Nhằm hiện thực hóa các kết quả đã nghiên cứu, chúng tôi sẽ tiến hành triển khai thử nghiệm một hệ thống PKI chứng thực tập trung theo kiến trúc PKI phân cấp (kiến trúc đã được chọn lựa sau khi được phân tích ở Chương 4) sử dụng gói phần mềm mã nguồn mở EJBCA sau khi được cài tiến ở phần trên. Mục tiêu được đặt ra là hệ thống sau khi được triển khai có thể sử dụng ngay trong thực tế, cung cấp các chứng nhận cho người dùng để chứng thực người sử dụng web, ký và mã hóa thư điện tử, ký văn bản, ... Ngoài ra, hệ thống được triển khai thử nghiệm cần đảm bảo các tiêu chí khác như tính tổng quát, có thể áp dụng cho hệ thống khác và dễ dàng được mở rộng.

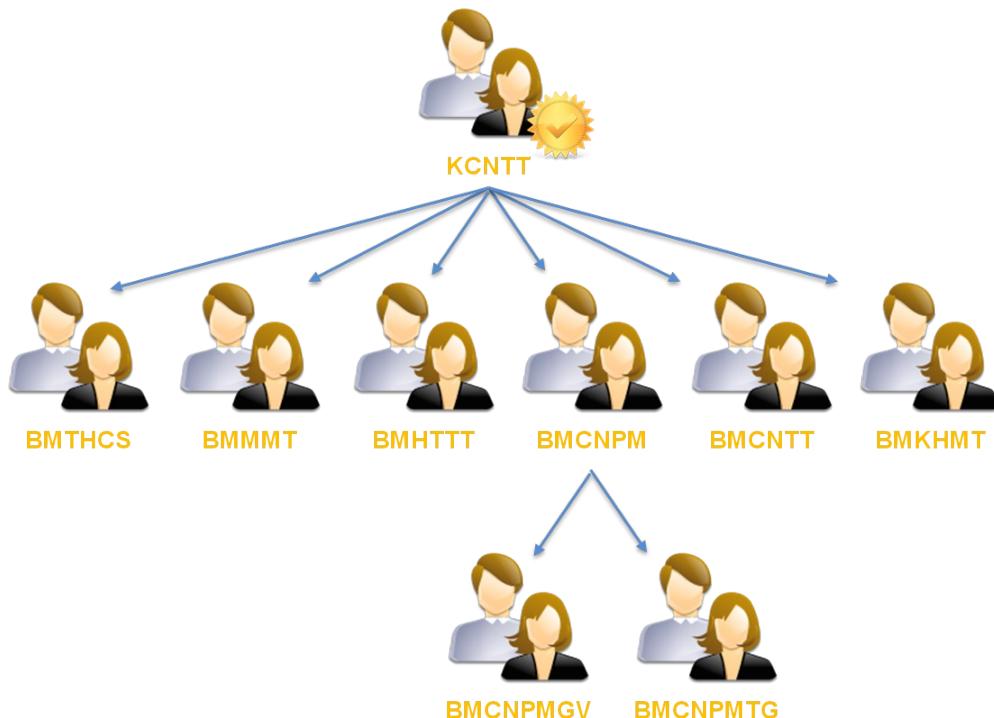
### 8.3.2 Mô hình triển khai

CA gốc (tên **KCNTT**) do Khoa CNTT quản lý. Tại mỗi bộ môn sẽ có một CA con của CA gốc này để cấp chứng nhận cho các giảng viên và trợ giảng trong bộ môn.

- Bộ môn Tin học Cơ sở: CA con “**BMTHCS**”.
- Bộ môn Mạng Máy tính: CA con “**BMMMT**”.

- Bộ môn Hệ thống Thông tin: CA con “**BMHTTT**”.
- Bộ môn Công nghệ Phần mềm: CA con “**BMCNPM**”.
- Bộ môn Công nghệ Tri thức: CA con “**BMCNTT**”.
- Bộ môn Khoa học Máy tính: CA con “**BMKHMT**”.

Nhằm tổng quát hóa hệ thống, CA con và “**BMCNPM**” sẽ có hai CA con khác là CA con “**BMCNPMSGV**” để cấp chứng nhận cho các giảng viên CA con “**BMCNPMTG**” để cấp chứng nhận cho các trợ giảng trong bộ môn Công nghệ Phần mềm.



**Hình 8.3. Mô hình triển khai hệ thống chứng thực tại Khoa CNTT,  
trường ĐH KHTN, Tp.HCM**

Mỗi CA (gốc và con) sẽ có các quản trị viên được chia làm 4 nhóm:

- **Quản trị viên CA (CA Administrator)**
  - ✓ Quản lý hiện trạng chứng nhận (Certificate Profile).
  - ✓ Quản lý hiện trạng thực thể cuối (End-Entity Profile).
  - ✓ Cấu hình log.
  - ✓ Tạo quản trị RA.

- **Quản trị viên RA (RA Administrator)**
  - ✓ Xem, thêm, xóa, sửa thực thể cuối.
  - ✓ Hủy chứng nhận của thực thể cuối.
- **Giám sát viên (Supervisor)**
  - ✓ Xem thông tin các thực thể cuối được tạo.
  - ✓ Tìm kiếm log và xem ai đã làm gì.
- **Siêu quản trị viên (Super Administrator)**
  - ✓ Có tất cả quyền.
  - ✓ Cấu hình hệ thống.
  - ✓ Quản lý các CA.
  - ✓ Quản lý các publisher (LDAP, AD hoặc tùy chọn).
  - ✓ Tạo quản trị CA, RA.

Nhằm đảm bảo tính an toàn hệ thống, chỉ có siêu quản trị viên mới có thể truy cập trực tiếp trên máy chủ của hệ thống còn các quản trị viên khác chỉ được phép truy cập vào hệ thống một cách gián tiếp thông qua máy tính khác.

Tên các CA nói riêng cũng như tên thực thể được phát hành chứng nhận đều được đặt theo chuẩn tên phân biệt X.500 (Phụ lục A), cụ thể như sau:

**Bảng 8.2. Tên của các CA theo chuẩn X.500**

| CA        | Tên   |
|-----------|---|
| KCNTT     | CN=KCNTT, OU=Khoa CNTT, O=Truong DH KHTN, C=VN  |
| BMTHCS    | CN=BMTHCS, OU=Bo mon THCS, O=Khoa CNTT, C=VN    |
| BMMMT     | CN=BMMMT, OU=Bo mon MMT, O=Khoa CNTT, C=VN      |
| BMHTTT    | CN=BMHTTT, OU=Bo mon BMHTTT, O=Khoa CNTT, C=VN  |
| BMCNPM    | CN=BMCNPM, OU=Bo mon CNPM, O=Khoa CNTT, C=VN    |
| BMCNPMSGV | CN=BMCNPMSGV, OU=Bo mon CNPM, O=Khoa CNTT, C=VN |
| BMCNPMTG  | CN=BMCNPMTG, OU=Bo mon CNPM, O=Khoa CNTT, C=VN  |
| BMCNTT    | CN=BMCNTT, OU=Bo mon CNTT, O=Khoa CNTT, C=VN    |
| BMKHMT    | CN=BMKHMT, OU=Bo mon KHMT, O=Khoa CNTT, C=VN    |

Ưu điểm của EJBCA là độc lập với môi trường nên có thể triển khai trên các hệ điều hành khác nhau như Windows và Linux. Hơn nữa, EJBCA có khả năng kết nối với các hệ quản trị cơ sở dữ liệu sau:

- Hypersonic (hsqldb) (mặc định trong JBoss)
- PostgreSQL 7.2 và 8.x (<http://www.postgresql.org/>)
- MySQL 4.x và 5.x (<http://www.mysql.com/>)
- Oracle 8i, 9i và 10g (<http://www.oracle.com/>)
- Sybase
- MS-SQL2000 và 2003
- Informix 9.2
- DB2

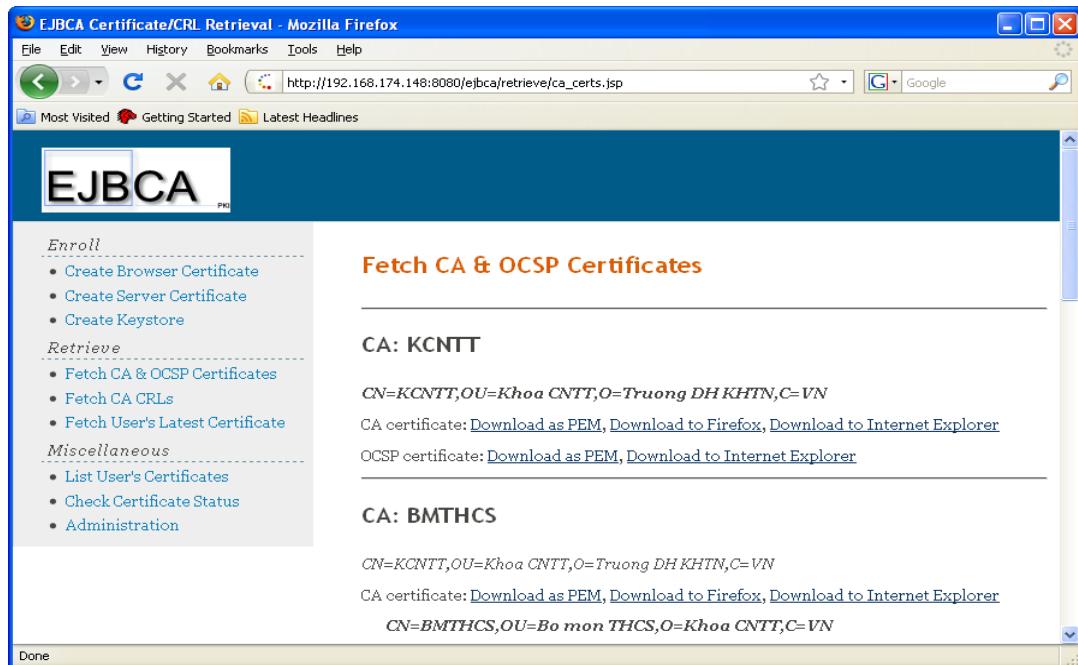
Hệ quản trị cơ sở dữ liệu mặc định trong EJBCA là Hypersonic (hsqldb) có sẵn trong JBoss không nên được sử dụng do có một số khuyết điểm như sau:

- Cơ sở dữ liệu Hypersonic nằm trực tiếp trong bộ nhớ, nghĩa là càng sử dụng nhiều càng tốn bộ nhớ. Khi một số lượng lớn chứng nhận được phát hành, điều thành trở thành trở ngại cho hệ thống. Do đó hệ quản trị cơ sở dữ liệu này không thích hợp cho các hệ thống lớn.
- Hypersonic không hỗ trợ đầy đủ SQL, đặc biệt trong các câu lệnh ALTER. Điều này làm cho việc nâng cấp hệ thống trở nên khó khăn vì khi một phiên bản mới của EJBCA được phát hành, ta không khôn thể tạo các tập lệnh (script) để cập nhật cơ sở dữ liệu nếu một số bảng đã thay đổi.

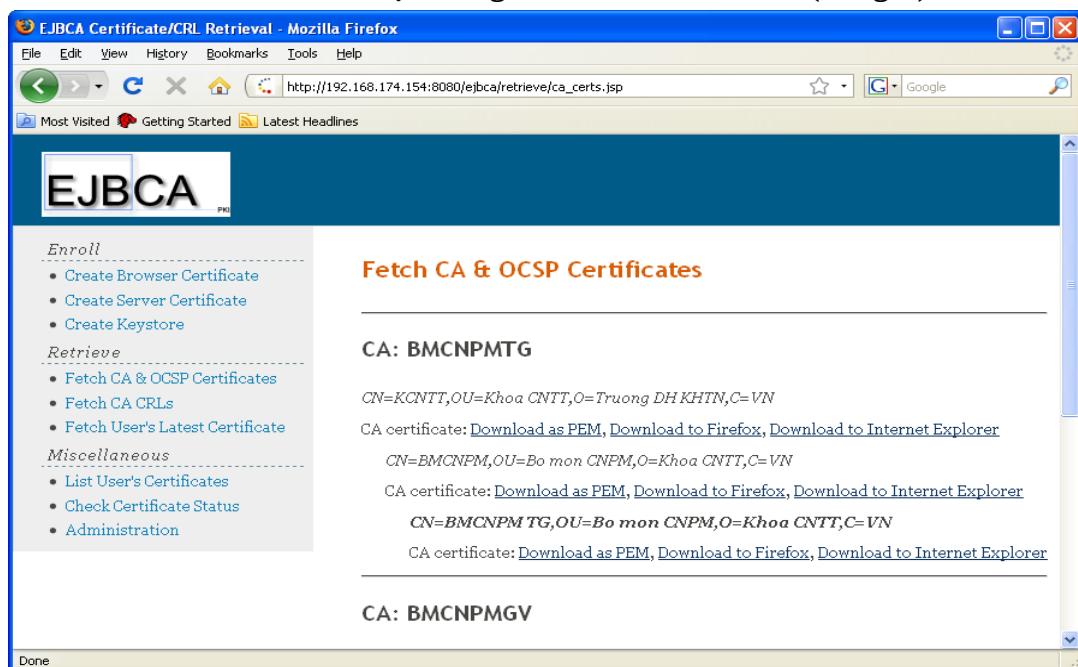
Vì lý do đó, chúng tôi chọn hệ quản trị cơ sở dữ liệu MySQL (cũng là một gói phần mềm mã nguồn mở) để triển khai. Hệ thống được chúng tôi triển khai trên cả hai môi trường Windows và Linux, sử dụng cơ sở dữ liệu MySQL (cách triển khai được trình bày ở Phụ lục B). Hệ thống sau khi triển khai đã có thể đi vào hoạt động, cung cấp các tính năng cho người sử dụng (yêu cầu cấp chứng nhận, chứng thực người sử dụng web, ký và mã hóa thư điện tử, ký văn bản, ...) và quản trị viên (cấu hình CA, thêm CA con, thêm RA, ...).

### 8.3.3 Kết quả triển khai và thử nghiệm

Hệ thống sau khi triển khai đã đạt được các mục tiêu đề ra. Hình 8.4 dưới đây cho thấy CA gốc KCNTT và các CA con trực tiếp của nó được thể hiện khi truy cập vào trang web của CA gốc và Hình 8.5 cho thấy CA con BMCNPMTG và các CA con trực tiếp của nó được thể hiện khi truy cập vào trang web của CA BMCNPMTG.



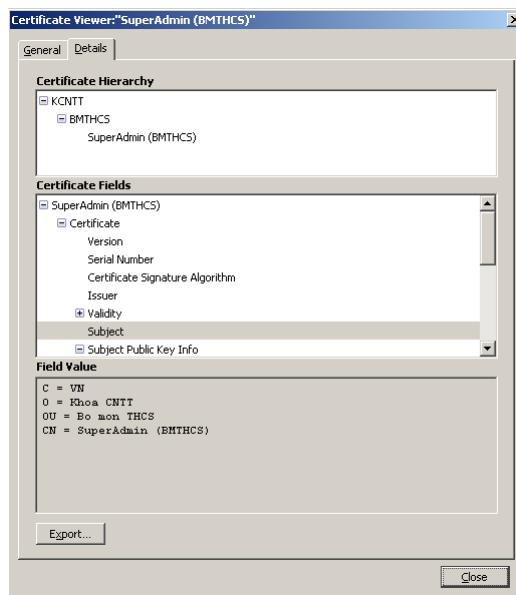
**Hình 8.4. Giao diện trang web của CA KCNTT (CA gốc)**



**Hình 8.5. Giao diện trang web của CA BMCNPMTG**

Hệ thống có thể cấp chứng nhận cho thực thể cuối (người sử dụng) và cho họ có thể ứng dụng vào một số lĩnh vực chính như:

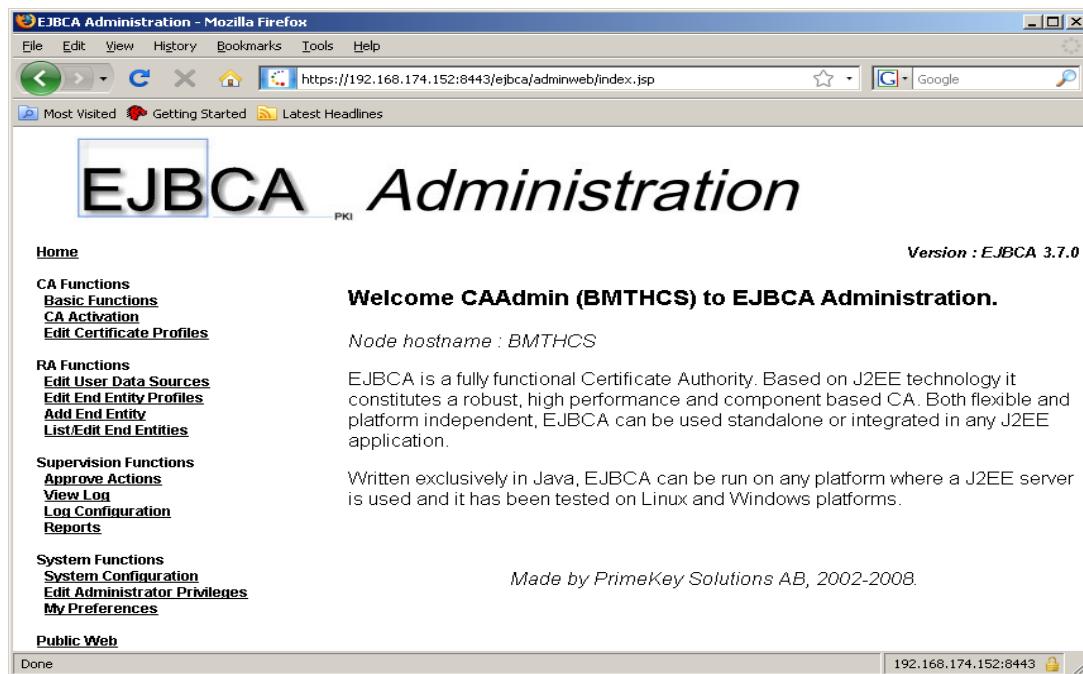
- **Chứng thực người sử dụng web:** cấp chứng nhận cho quản trị CA, RA, giám sát viên và siêu quản trị viên để có thể đăng nhập vào hệ thống quản trị CA trên trình duyệt web như Internet Explorer và Fire Fox. Hình 8.6 cho thấy một chứng nhận được cấp cho người sử dụng với tên “SuperAdmin (BMTHCS)” để người này có thể đăng nhập vào **CA BMTHCS** với tất cả các quyền.



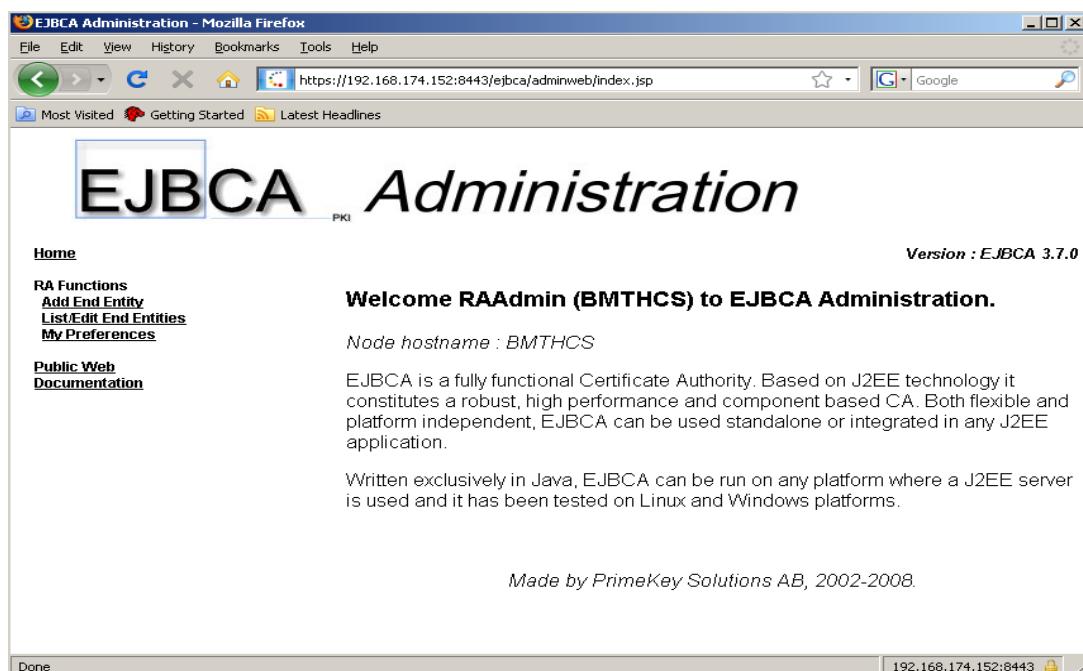
**Hình 8.6. Chứng nhận cho người dùng tên “SuperAdmin (BMTHCS)”**

**Hình 8.7. Giao diện quản trị toàn quyền của người dùng “SuperAdmin (BMTHCS)”**

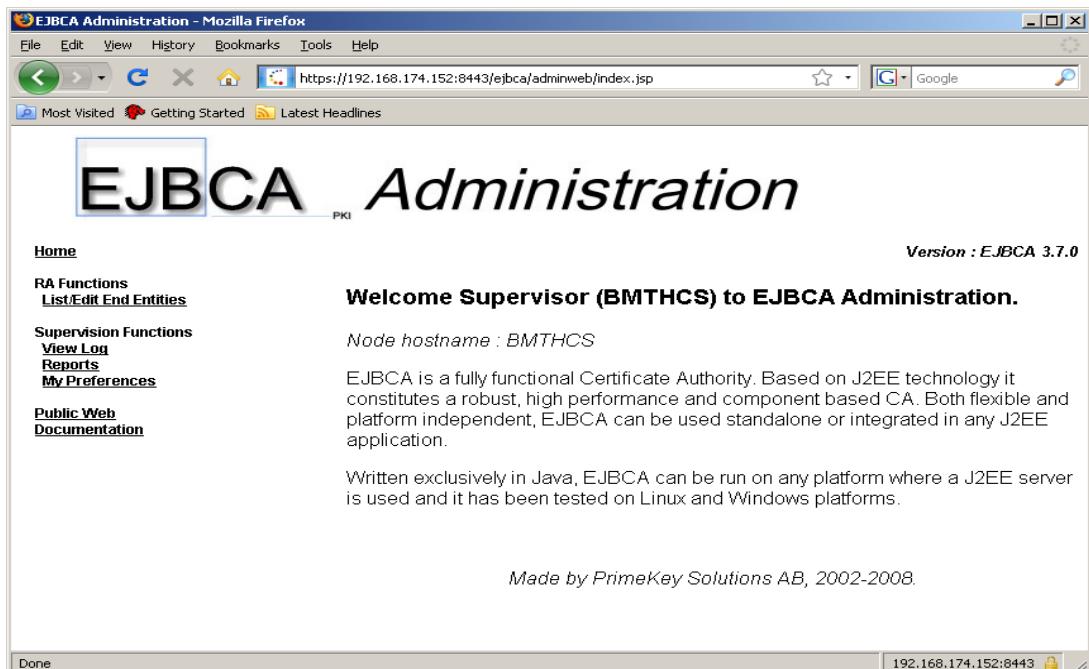
Hình 8.7 ở trên cho thấy người dùng có thể sử dụng chứng nhận của mình (được cấp phép với quyền siêu quản trị) để đăng nhập vào hệ thống CA của bộ môn THCS. Các Hình 8.8, Hình 8.9 và Hình 8.10 bên dưới cho thấy các người dùng khác có thể đăng nhập vào hệ thống với quyền quản trị lần lượt là quản trị CA, RA và giám sát viên.



**Hình 8.8. Giao diện quản trị CA của người dùng “CAAdmin (BMTHCS)”**

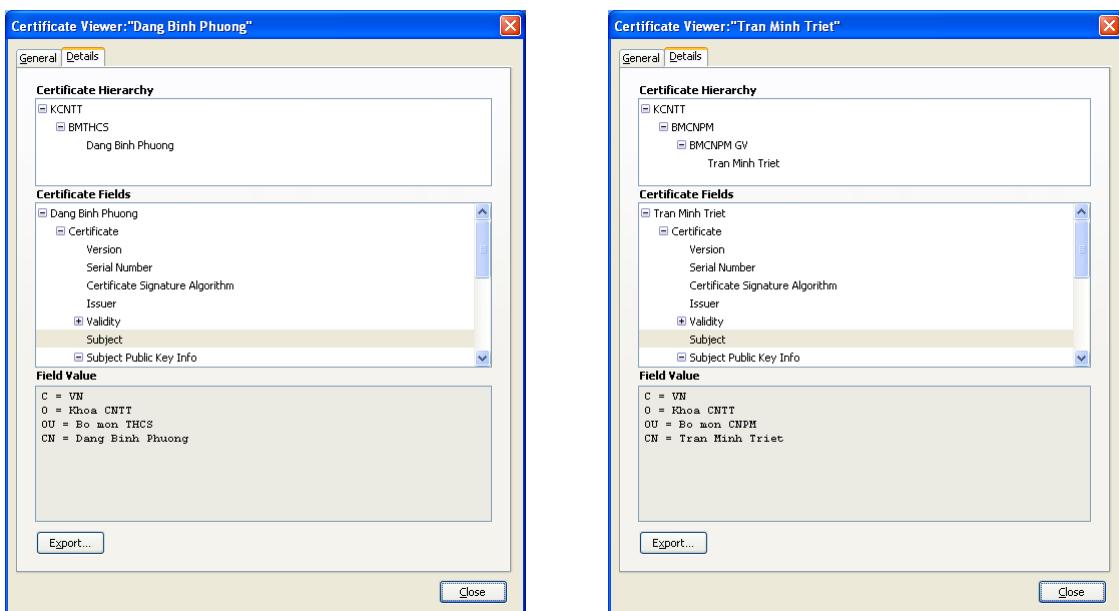


**Hình 8.9. Giao diện quản trị RA của người dùng “RAAdmin (BMTHCS)”**



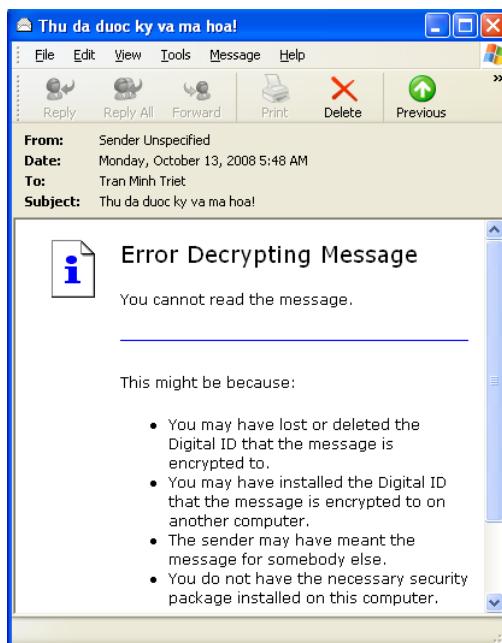
**Hình 8.10. Giao diện giám sát của người dùng “Supervisor (BMTHCS)”**

- **Ký và mã hóa thư điện tử:** hệ thống có thể cấp chứng nhận cho người dùng để ký/ xác nhận chữ ký, mã hóa/ giải mã thư điện tử trong ứng dụng thư điện tử như Outlook Express. Hình 8.11 cho thấy hai chi tiết chứng nhận của người dùng tên “Dang Binh Phuong” (được CA BMTHCS cấp) và chứng nhận của người dùng tên “Tran Minh Triet” được (CA BMCNPMGV cấp).

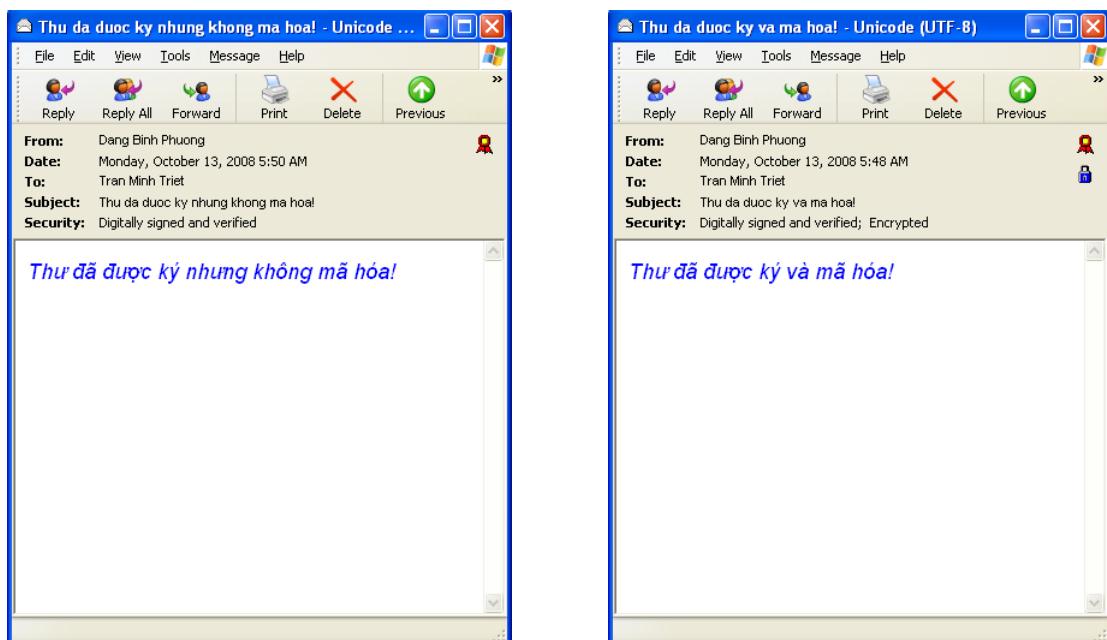


**Hình 8.11. Nội dung chứng nhận của người dùng**

Khi người dùng “Dang Binh Phuong” gửi thư điện tử cho người dùng “Tran Minh Triet” đồng thời ký hay mã hóa thư điện tử, Hình 8.12 cho thấy người nhận “Tran Minh Triet” sẽ không thể xác thực người gửi nếu không có chứng nhận của người gửi hoặc không phải đúng người mà người gửi muốn gửi (không có khóa khớp với khóa mà người gửi sử dụng). Hình 8.13 cho thấy khi có đầy đủ các chứng nhận cần thiết, người nhận sẽ có thể đọc được nội dung thư được ký hoặc mã hóa hay cả hai.

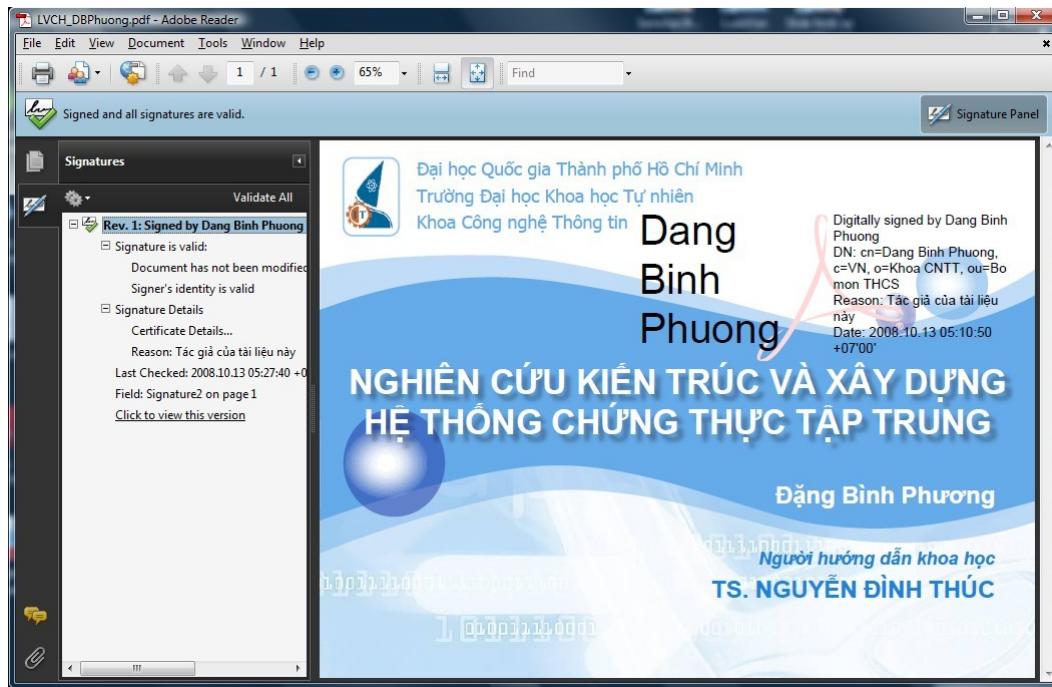


Hình 8.12. Lỗi không thể đọc được thư đã ký và mã hóa

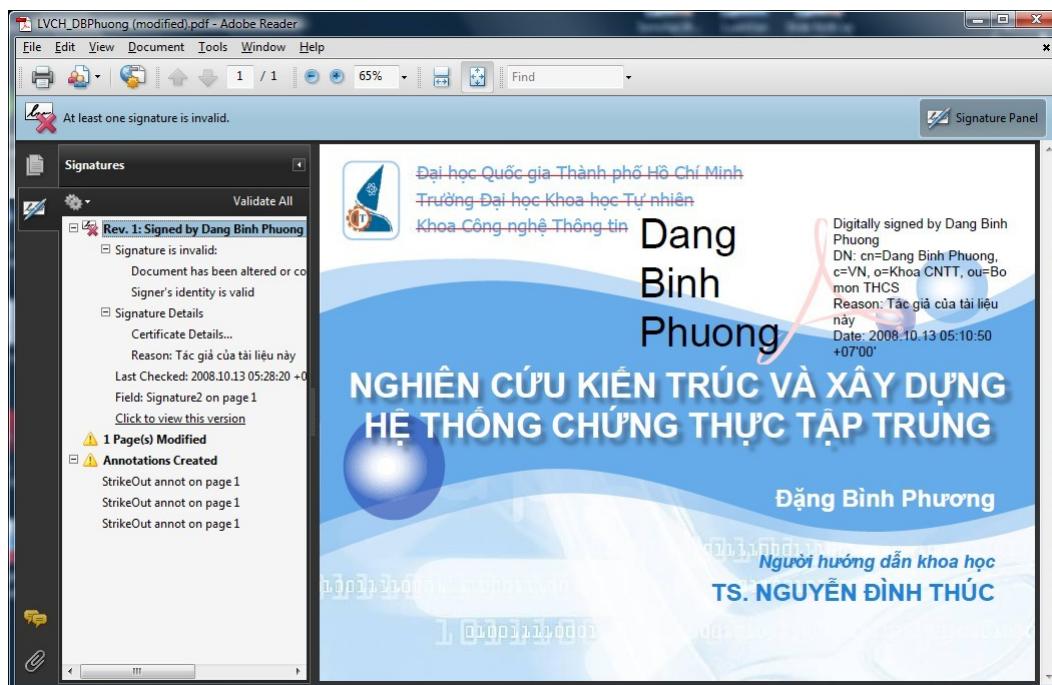


Hình 8.13. Nội dung thư được ký và nội dung thư được ký đồng thời mã hóa

- Ký và xác nhận tài liệu điện tử:** chứng nhận được cấp cho người dùng như trên cũng có thể được sử dụng để ký văn bản PDF. Hình 8.14 cho thấy người dùng đã ký vào tài liệu của mình (góc phải trên) đồng thời ứng dụng đọc văn bản PDF cũng xác định tài liệu này hợp lệ và chưa bị sửa đổi (góc trái).



Hình 8.14. Ký và xác nhận chữ ký trong tài liệu điện tử PDF



Hình 8.15. Tài liệu điện tử PDF được ký đã bị thay đổi

Hình 8.15 cho thấy ứng dụng đọc văn bản PDF đã nhận ra văn bản đã có thay đổi (gạch bỏ các thông tin ở góc trái trên). Ngoài ra ứng dụng cũng cung cấp chức năng phục hồi lại nguyên bản của tài liệu trước khi được ký.

## 8.4 Kết luận

EJBCA là một gói phần mềm mã nguồn mở nổi tiếng, có thể triển khai một hệ thống PKI hoàn chỉnh, đầy đủ chức năng. Nhằm tận dụng những đặc tính ưu việt của gói phần mềm này đồng thời có thể quản lý được quá trình phát triển cũng như độ an toàn của hệ thống, đề tài đã tiến hành tìm hiểu và phân tích.

Sau khi phân tích gói phần mềm này, chúng tôi nhận thấy khóa cho hệ mã RSA được phát sinh là rất yếu (trên cơ sở phân tích ở Chương 5) nên đã thay bằng bộ thư viện mã hóa SmartRSA mà chúng tôi đã xây dựng. Ngoài ra, chúng tôi đã triển khai thử nghiệm một hệ thống chứng thực tập trung theo kiến trúc PKI phân cấp đơn giản có thể sử dụng ngay trong thực tế. Hệ thống được triển khai này mang lại đầy đủ các tính chất cần thiết nhằm thiết lập một môi trường an toàn, tin cậy trong giao tiếp như tính cẩn mật, tính toàn vẹn, tính xác thực và tính không thể chối từ. Hơn nữa, hệ thống còn có khả năng mở rộng, tích hợp với các hệ thống khác một cách dễ dàng.

Ngoài ra, chúng tôi đã thử nghiệm hệ thống bằng cách cấp chứng nhận cho các thực thể cuối (người sử dụng) và ứng dụng vào một số lĩnh vực phổ biến như chứng thực người sử dụng web (đăng nhập vào hệ thống quản trị); ký/mã hóa và xác nhận/giải mã thư điện tử; ký và xác nhận tài liệu điện tử.

Hệ thống được triển khai có thể sử dụng ngay trong thực tế đồng thời có tính tổng quát cao (nhiều loại CA: CA gốc, CA con cấp một và CA con cấp khác nhau) nên có thể ứng dụng trong bất kỳ tổ chức nào có mô hình phân cấp tương tự. Ngoài ra hệ thống cũng rất dễ mở rộng bằng cách triển khai thêm các CA con và yêu cầu các CA sẵn có ký chứng nhận cho chúng.

## Chương 9

### Kết luận

#### 9.1 Một số kết quả đạt được

Dựa trên các nghiên cứu, phân tích hạ tầng khóa công khai (PKI) bao gồm các thuật toán băm và thuật toán chữ ký số (Chương 2), tổ chức chứng nhận khóa công khai (Chương 3) cũng như các kiến trúc hạ tầng khóa công khai (Chương 4), đề tài đã xác định thuật toán hàm băm SHA, chữ ký số RSA và hệ thống chứng thực tập trung theo kiến trúc PKI phân cấp là những vấn đề cần tập trung nghiên cứu.

Hạt nhân quan trọng của hạ tầng khóa công khai (PKI) chính là hệ thống mã hóa khóa công khai RSA. Do đó, đề tài đã tập trung nghiên cứu, phân tích các nguy cơ tấn công gây tổn thương trên hệ mã này, từ đó đưa ra được các giải pháp nhằm cài đặt hệ mã một cách an toàn (Chương 5). Ngoài ra đề tài cũng nghiên cứu một số bài toán quan trọng cần giải quyết kết hợp với những cơ sở phân tích ở trên nhằm xây dựng hệ mã RSA an toàn và hiệu quả (Chương 6).

Trên các kết quả nghiên cứu và phân tích đó, đề tài đã xây dựng được một bộ thư viện mã hóa “SmartRSA” nhằm hỗ trợ việc cài đặt hệ mã RSA an toàn, hiệu quả. Các thử nghiệm cũng được lần lượt tiến hành cho thấy các cơ sở phân tích đó là hợp lý và phần nào cho thấy tính hiệu quả của bộ thư viện này (Chương 7).

Cuối cùng, nhằm mục đích tận dụng tính ổn định của các gói phần mềm mã nguồn mở có sẵn, đồng thời có thể kiểm soát được quá trình phát triển và độ an toàn của hệ thống, đề tài đã chọn gói phần mềm mã nguồn mở “EJBCA” để phân tích. EJBCA là gói phần mềm mã nguồn mở nổi tiếng, cho phép triển khai một hệ thống PKI hoàn chỉnh, đầy đủ chức năng. Tuy nhiên, thư viện phát sinh khóa cho hệ mã RSA của EJBCA là không an toàn (dựa trên các phân tích ở Chương 5) nên đã được thay bằng thư viện SmartRSA. Từ đó đề tài đã triển khai thử nghiệm được một hệ thống chứng

thực tập trung theo kiến trúc PKI phân cấp đơn giản có khả năng sử dụng ngay trong thực tế.

Hệ thống chứng thực này đang được triển khai nhằm phục vụ cho luồng công việc (workflow) quản lý hồ sơ công văn tại Khoa Công nghệ Thông tin, trường Đại học Khoa học Tự nhiên, Thành phố Hồ Chí Minh. Hệ thống có thể cấp chứng nhận cho các giảng viên và trợ giảng trong Khoa để có thể xác thực với trình duyệt web, ký và mã hóa thư điện tử, ký văn bản PDF, ...

## **9.2 Hợp đồng phát triển**

- Tìm hiểu thêm về các tấn công hiện đại, có nguy cơ gây tổn thương hệ mã khóa công khai RSA, hạt nhân của PKI, và tìm cách khắc phục nhằm đảm bảo tính an toàn của hệ thống.
- Tiếp tục nghiên cứu các giải pháp nhằm cài đặt hệ mã RSA an toàn và hiệu quả hơn bao gồm các thuật toán tính toán nhanh, thuật toán phát sinh số ngẫu nhiên mạnh, thuật toán kiểm tra tính nguyên tố và thuật toán phát sinh số nguyên tố (xác suất).
- Tiếp tục nghiên cứu và phân tích gói phần mềm EJBCA nhằm phát hiện các điểm yếu khác để khắc phục. Ngoài ra, cần triển khai toàn diện hệ thống PKI/CA này để đưa vào hoạt động thực tế trên quy mô lớn, trong thời gian dài để có thể đánh giá chính xác tính hiệu quả của hệ thống.

## Tài liệu tham khảo

### Tiếng Việt

- [1] Phạm Huy Điện, Hà Huy Khoái (2003), *Mã hóa thông tin – Cơ sở toán học và ứng dụng*, NXB ĐHQG Hà Nội, Hà Nội.
- [2] Bùi Doãn Khanh, Nguyễn Đình Thúc (2004), *Giáo trình mã hóa thông tin – Lý thuyết và ứng dụng*, NXB LĐXH.
- [3] Luật Giao dịch điện tử của Quốc hội nước CHXHCN Việt Nam, Số 51/2005/QH11, 29/11/2005.

### Tiếng Anh

- [4] M. Agrawal, N. Kayal, N. Saxena (2002), “PRIMES is in P”, *Indian Institute of Technology Kanpur*, India.
- [5] M. Atreya, B. Hammond, S. Paine, P. Starrett, S. Wu (2002), “Digital Signatures, RSA”.
- [6] P.S.L.M. Barreto, V. Rijmen (2003), "The WHIRLPOOL Hashing Function".
- [7] S.M. Bellovin, M. Merritt (1992), “Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks”, *Proceedings of the I.E.E.E. Symposium on Research in Security and Privacy*, Oakland.
- [8] E. Biham, R. Chen (2004), “Near-Collisions of SHA-0”, *Cryptology ePrint Archive*.
- [9] B.D. Boer, A. Bosselaers (1991), “An Attack on the Last Two Rounds of MD4”, *Crypto 1991*, pp. 194–203.
- [10] B.D. Boer, A. Bosselaers (2003), “Collisions for the Compression Function of MD5”, 2003, pp. 293–304.
- [11] D. Boneh (1999), “Twenty Years of Attacks on the RSA Cryptosystem”, *Notices of the ACM*.
- [12] D. Boneh, G. Durfee, Y. Frankel (1998), “An Attack on RSA given a fraction of the private key bits”, *AsiaCrypt 1998*, volume 1514 of Lecture Notes in Computer Sciences, Springer-Verlag, 1514, pp. 25–34.

- [13] R. P. Brent (1980), “An Improved Monte Carlo Factorization Algorithm”, *BIT 20*, pp.176-184.
- [14] B. Burr (2006), “NIST Cryptographic Standards Status Report”, *Security Technology Group, NIST*.
- [15] J. Chang (2006), “PKI Legislation and Policy”, *Taiwan International PKI Training Program*, pp.53-92.
- [16] M. Cochran (2008), “Notes on the Wang et al.  $2^{63}$  SHA-1 Differential Path”.
- [17] D. Coppersmith (1997), “Small solutions to polynomial equations, and low exponent RSA vulnerabilities”, *Journal of Cryptology*, vol. 10, pp.233-260.
- [18] F. Chabaud, A. Joux (1998), “Differential Collisions in SHA-0”, *CRYPTO 1998*, pp. 56-71.
- [19] S. Choudhury, K. Bhatnagar, W. Haque (2002), “PKI implementation and design”, M&T Books, New York.
- [20] H. Dobbertin (1996), “Cryptanalysis of MD4”, *Fast Software Encryption 1996*, pp. 53–69.
- [21] H. Dobbertin (1996), “Cryptanalysis of MD5 compress”, *Announcement on Internet*.
- [22] T. ElGamal (1985), “A public key cryptosystem and a signature scheme based on discrete logarithms”, *IEEE Trans inf Theo*, 31, pp. 469-472.
- [23] A.I. Ghori, A. Parveen (2006), “PKI Administration Using EJBCA and OpenCA”, *George Mason University*.
- [24] H. Gilbert, H. Handschuh (2003), “Security Analysis of SHA-256 and Sisters”, *Selected Areas in Cryptography 2003*, pp. 175–193.
- [25] J. Gordon (1985), “Strong Primes are Easy to Find”, *Proceedings of EUROCRYPT 84*, pp. 216-223, Paris.
- [26] J. Gordon (1984), “Strong RSA keys”, *Electronics Letters*, 20(12), pp. 514-516.
- [27] Z. Guo, T. Okuyama, M.R. Finley. Jr (2005), “A New Trust Model for PKI Interoperability”.
- [28] J. Hastad (1998), “Solving simultaneous modular equations of low degree”, *SIAM J. of Computing*, 17, pp. 336-341.

- [29] M.E. Hellman, C.E. Bach (1986), “Method and apparatus for use in public-key data encryption system”.
- [30] A. Joux, S. Carribault, Lemuet, Jalby (2004), “Collision for the full SHA-0 algorithm”, *CRYPTO 2004*.
- [31] M. Joye, P. Paillier, S. Vaudenay (2000), “Efficient Generation of Prime Numbers”, *Lecture Notes in Computer Science*, pp. 340–354.
- [32] V. Klima (2005), “Finding MD5 Collisions – a Toy For a Notebook”, *Cryptology ePrint Archive Report 2005/075*.
- [33] V. Klima (2006), “Tunnels in Hash Functions: MD5 Collisions Within a Minute”, *Cryptology ePrint Archive Report 2006/105*.
- [34] P. Kocher (1996), “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, *Lecture Notes in Computer Science, CRYPTO 1996*, vol. 1109, pp. 104-113.
- [35] Jae-IL. Lee (2005), “PKI Deployment in Asia”, *2005 PKI Conference*, Tunisia.
- [36] A. Lenstra, X. Wang, B.D. Weger (2005), “Colliding X.509 Certificates”, *Cryptology ePrint Archive Report 2005/067*.
- [37] H.W. Lenstra Jr (1996), “Elliptic Curves and Number-Theoretic Algorithms”, *Report 86–19, Mathematisch Instituut, Universiteit van Amsterdam*.
- [38] A.K. Lenstra, H.W. Lenstra (1993), Jr., eds., Development of the Number Field Sieve, *Lecture Notes in Mathematics*, vol. 1554.
- [39] U.M. Maurer, “Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters”, *Institute for Theoretical Computer Science*, Switzerland.
- [40] A. Menezes, P. van Oorschot, S. Vanstone (1997), *Handbook of Applied Cryptography*, CRC Press.
- [41] G.L. Miller (1976), “Riemann’s Hypothesis and Tests for Primality”, *Journal of Computer Systems Science*, 13(3), pp. 300–317.
- [42] S. Pohlig and M. Hellman (1978), “An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance”, *IEEE Transactions on Information Theory* 24, pp. 106–110.

- [43] J.M. Pollard (1974), “Theorems of Factorization and Primality Testing”, *Proceedings of the Cambridge Philosophical Society* 76, pp. 521–528.
- [44] J.M. Pollard (1975), “A Monte Carlo method for factorization”, *BIT Numerical Mathematics*, 15(3), pp. 331-334.
- [45] C. Pomerance (1984), “The quadratic sieve factoring algorithm”, *Lecture Notes in Computer Science, Advances in Cryptology*, vol. 209, pp. 169-182.
- [46] M.O. Rabin (1980), “Probabilistic Algorithm for Testing Primality”, *Journal of Number Theory*, 12(1), pp. 128– 138.
- [47] V. Rijmen, E. Oswald (2005), “Updated on SHA-1, IAIK”, *Graz University of Technology*, Denmark.
- [48] R.L. Rivest, A. Shamir, L.M. Adleman (1978), “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, *Communications of the ACM*, 21(2), pp.120–126.
- [49] R.L. Rivest, R.D. Silverman (1999), “Are „Strong“ Primes Needed for RSA”.
- [50] R.L. Rivest, A. Shamir, L.M. Adleman (1978), “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM*, 21(2), pp. 120-126.
- [51] Y. Seung, R. Kravets (2002), “Practical PKI for Ad Hoc Wireless Networks”, *Departmnet of Computer Science*, University of Illinois, Urbana-Champain Technical Report.
- [52] A. Shamir (1979), “How to Share a Secret”, *Communication of the ACM*, 22(11).
- [53] R.D. Silverman (1987), “The Multiple Polynomial Quadratic Sieve,” *Mathematics of Computation*, 48(177), pp. 329–339.
- [54] G.J. Simmons, M.J. Norris (1977), “Preliminary comments on the MIT public-key cryptosystem”, *Cryptologia*, 1(4), pp. 406-414.
- [55] R. Solovay and V. Strassen (1977), “A Fast Monte–Carlo Test for Primality”, *SIAM Journal on Computing*, vol.6, pp. 84–85.
- [56] X. Wang, D. Feng, X. Lai, H. Yu (2004), “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD”, *Cryptology ePrint Archive Report 2004/199*.

- [57] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu (2005), “Cryptanalysis of the Hash Functions MD4 and RIPEMD”, *Eurocrypt 2005*, pp. 1–18.
- [58] X. Wang, H. Yu, Y.L. Yin (2005), “Efficient Collision Search Attacks on SHA-0”, *CRYPTO 2005*.
- [59] X. Wang, Y.L. Yin, H. Yu (2005), “Finding Collisions in the Full SHA-1”, *CRYPTO 2005*.
- [60] J. Weise (2001), “Public Key Infrastructre Overview”, *SunPS<sup>SM</sup> Global Security Practice*, Sun BluePrints™ OnLine.
- [61] M. Wiener (1990), “Cryptanalysis of short RSA secret exponent”, *IEEE Transactions of Information Theory*, 36, pp. 553-558.
- [62] H. C. Williams, B. Schmid (1979), “Some remarks concerning the MIT public-key cryptosystem”, *BIT*, vol. 19 pp.525-538.
- [63] H.C. Williams (1982), “A p+1 method of factoring”, *Math. Comp.* 39, pp. 225-234.
- [64] Federal Register (2007), *Notices*, 72 (212).
- [65] Internet X.509 Public Key Infrastructure PKIX Roadmap, March 10, 2000.
- [66] FIPS 180 (13/5/1993)/ FIPS 180-1 (1995)/ FIPS 180-2 (1/8/2002)/ FIPS 180-3 (draft, 8/6/2007), Announcing the Secure Hash Standard (SHS).
- [67] FIPS 186 (19/5/1994)/ 186-1 (15/12/1998)/ 186-2 (27/12000)/ 186-3 (draft, 3/2006), Announcing the Digital Signature Standard (DSS).
- [68] NCCUSL (1999), *Uniform Electronic Transactions Act*.
- [69] RFC 1321, The MD5 Message-Digest Algorithm.
- [70] RCF 2251, The Lightweight Directory Access Protocol version 3.

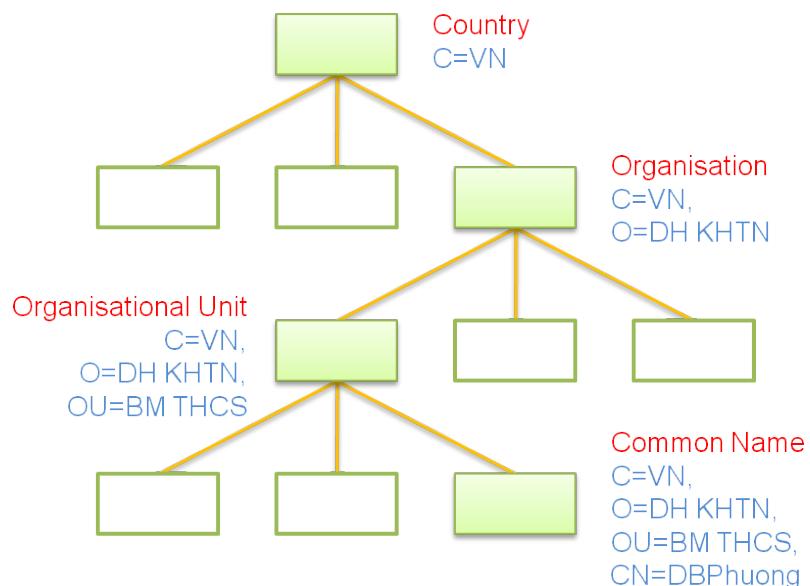
## Phụ lục A

### Tên phân biệt theo chuẩn X.500

X.500 giới thiệu cách đặt tên riêng biệt (Distinguished Name – DN), bảo đảm rằng mỗi đối tượng sẽ có một tên khác nhau.

Các thành phần:

- Quốc gia (Country – C)
- Tỉnh/Thành phố (State or Province – SP)
- Địa phương (Locality – L)
- Tổ chức (Organization – O)
- Bộ phận (Organizational Unit – OU)
- Tên chung (Common Name – CN)



**Hình 9.1. Ví dụ về tên phân biệt theo chuẩn X.500**

## Phụ lục B

### **Triển khai EJBCA trên môi trường Windows và Linux**

Các bước triển khai EJBCA trên môi trường Windows XP/2003/Vista, sử dụng hệ quản trị cơ sở dữ liệu MySQL như sau:

#### **Bước 1: Tạo thư mục c:\pki.**

Tất cả mọi thứ sau này sẽ được cài đặt trong thư mục này.

#### **Bước 2: Cài đặt Java**

- Tải phiên bản JDK về cài đặt, phiên bản mới nhất là jdk 1.6 update 6 tại địa chỉ: <http://www.sun.com/>
- Gói tải về có tên: jdk-6u6-windows-i586-p.exe
- Cài đặt vào thư mục c:\pki\java
- Tạo biến môi trường: JAVA\_HOME = c:\pki\java\jdk1.6.0\_06
- Thêm vào biến môi trường Path: c:\pki\java\jdk1.6.0\_06\bin
- Kiểm tra cài đặt thành công bằng cách mở cmd gõ lệnh: java –version

#### **Bước 3: Thay thế JCE Policy**

- Tải Unlimited Strength Jurisdiction Policy Files for JDK tại địa chỉ: <http://www.sun.com/>
- Giải nén và chép đè vào thư mục \$JAVA\_HOME \jre\lib\security và thư mục runtime của java \jre\lib\security.

#### **Bước 4: Cài đặt Ant**

- Tải apache-ant-1.7.0-bin.zip tại địa chỉ: <http://www.apache.com/>
- Giải nén vào thư mục C:\PKI
- Tạo biến môi trường: ANT\_HOME = c:\pki\apache-ant-1.7.0
- Tạo biến môi trường: ANT\_OPTS = -Xmx512m (điều chỉnh bộ nhớ tối đa để build ứng dụng)
- Thêm vào biến môi trường Path: c:\pki\apache-ant-1.7.0\bin
- Kiểm tra cài đặt thành công bằng cách mở cmd gõ lệnh: ant –version

#### **Bước 5: Cài đặt JBoss**

- Tải jboss-4.2.2.GA.zip tại trang địa chỉ: <http://sourceforge.net>
- Giải nén vào thư mục c:\pki
- Tạo biến môi trường: JBOSS\_HOME = c:\pki\jboss-4.2.2.GA

- Thêm vào biến môi trường Path: c:\pki\jboss-4.2.2.GA\bin
- Vào cmd gõ lệnh run.bat) để khởi động JBoss. Nhấn Ctrl+C để dừng JBoss.

#### Bước 6: Thiết lập JBoss chạy như một Windows Service (tùy chọn)

- Tải Java Service Wrapper tại địa chỉ: <http://sourceforge.net>
- Giải nén vào thư mục c:\java-wrapper
- Chép và đổi tên các tập tin sau sang thư mục \$JBOSS\_HOME\bin:
  - c:\java-wrapper\bin\Wrapper.exe → Wrapper.exe
  - c:\java-wrapper\src\bin\App.bat.in → JBoss.bat
  - c:\java-wrapper\src\bin\InstallApp-NT.bat.in → InstallApp-NT.bat
  - c:\java-wrapper\src\bin\UninstallApp-NT.bat.in → UninstallApp-NT.bat
- Chép hai tập tin sau sang thư mục \$JBOSS\_HOME\lib:
  - c:\java-wrapper\lib\Wrapper.DLL
  - c:\java-wrapper\lib\Wrapper.jar
- Wrapper cần một tập tin cấu hình:
  - Tạo thư mục \$JBOSS\_HOME\conf (JBoss mặc định không có thư mục này)
  - Chép tập tin c:\java-wrapper\src\conf\wrapper.conf.in sang thư mục \$JBOSS\_HOME\conf và đổi tên thành wrapper.conf
  - Tạo thư mục \$JBOSS\_HOME\logs (tập tin wrapper.conf sẽ tạo một tập tin wrapper.log trong thư mục logs này nhưng JBoss mặc định không có thư mục này).
- Thêm vào tập tin wrapper.conf các dòng sau :
  - Mục # Java Classpath
 

```
wrapper.java.classpath.2=%JAVA_HOME%/lib/tools.jar
wrapper.java.classpath.3=./run.jar
```
  - Mục # Java Additional Parameters
 

```
wrapper.java.additional.1=-Dprogram.name=run.bat
```
  - Mục # Application parameters
 

```
wrapper.app.parameter.1=org.jboss.Main
```
  - Mục # Log file to use for wrapper output logging.
 

```
wrapper.logfile=%JBOSS_HOME%/server/default/log/wrapper.log
```
  - Mục # Wrapper Windows Properties
 

```
<a href="mailto:wrapper.console.title=@app.long.name">wrapper.console.title=@app.long.name@[/code]</a>][code]
```
  - Mục # Name of the service
 

```
wrapper.ntservice.name=JBoss
```
  - Mục # Display name of the service

- wrapper.ntservice.displayname=JBoss Application Server
  - o Mục # Description of the service
    - wrapper.ntservice.description=JBoss Application Server
- Khởi động thử JBoss bằng cách chạy tập tin \$JBOSS\_HOME\bin\JBoss.bat. Nếu không có lỗi xảy ra, chạy tập tin \$JBOSS\_HOME\bin\InstallApp-NT.bat để thiết lập JBoss Webserver thành một dịch vụ của Windows.
- Để có thể Start/Restart/Stop JBoss, vào Start > Administrative Tools > Services > JBoss Application Server.

### Bước 7: Cài đặt MySQL

- Tải mysql-5.0.51b-win32.exe tại địa chỉ: <http://mysql.com/>
- Cài đặt MySQL 5.0 vào thư mục C:\PKI (hoặc mặc định)
- Cấu hình MySQL Server và thiết lập MySQL thành một dịch vụ của Windows.
  - o Configure the MySQL Server now
  - o Next
  - o Chọn Standard Configuration
  - o Chọn Install As a Windows Service
  - o Chọn MySQL5 (hoặc để mặc định là MySQL)
  - o Chọn Include Bin Directory in Windows Path
  - o Nhấn Next
  - o Modify Security Settings: 123456 / 123456 (mật khẩu của root)
  - o Next
  - o Execute

### Bước 8: Cài đặt MySQL Connector/J 5.1 (JDBC Driver)

- Tải phiên bản MySQL Connector cho Java 5.1 tại địa chỉ: <http://mysql.com/>
- Tập tin tải được có tên: mysql-connector-java-5.1.6.zip
- Giải nén và chép tập tin mysql-connector-java-5.1.6-bin.jar vào thư mục \$JBOSS\_HOME\server\default\lib\

### Bước 9: Cài đặt EJBCA

- Tải EJBCA phiên bản mới nhất trên ở địa chỉ: <http://ejbca.com/>
- Phiên bản mới nhất tải được có tên: ejbca\_3\_7\_0.zip
- Giải nén tập tin này vào thư mục c:\pki
- Tạo biến môi trường: EJBCA\_HOME = c:\pki\ejbca\_3\_7\_0
- Thêm vào biến môi trường Path: c:\pki\ejbca\_3\_7\_0\bin
- Vào thư mục c:\pki\ejbca\_3\_7\_0\conf, chép và đổi tên tập tin:
  - o web.properties.sample thành web.properties

- o ejbca.properties.sample thành ejbca.properties
- Các thông tin CA mặc định được lưu trong tập tin ejbca.properties:
  - o ca.name=**AdminCA1**
  - o ca.dn=CN=**AdminCA1**
  - o ca.keyspec=2048
  - o ca.keytype=RSA
  - o ca.signaturealgorithm=SHA1WithRSA
  - o ca.validity=3650
- Trong thư mục c:\pki, tạo thư mục ejbca-custom

### Bước 10: Thiết lập CSDL MySQL

- Tạo cơ sở dữ liệu tên ejbca
  - o mysqladmin –h localhost –P 3306 –u root –p create ejbca
  - o 123456
- Tạo user ejbca với password ejbca (cấp đầy đủ quyền)
  - o mysql –u root –p
  - o 123456
  - o grant all on ejbca.\* to ejbca@'localhost' identified by 'ejbca';
- Vào thư mục c:\pki\ejbca\_3\_7\_0\conf, chép và đổi tên tập tin database.properties.sample thành database.properties
- Chỉnh sửa file database.properties: bỏ các chú thích phần MySQL:
  - o database.name=mysql
  - o datasource.mapping=mySQL
  - o database.url=jdbc:mysql://127.0.0.1:3306/ejbca?characterEncoding=UTF-8
  - o database.driver=com.mysql.jdbc.Driver
  - o database.username=ejbca
  - o database.password=ejbca

### Bước 11: Triển khai hệ thống

Mở cmd, vào thư mục C:\PKI\ejbca\_3\_7\_0 thực hiện các công việc sau:

- ant bootstrap (biên dịch mã nguồn ejbca và triển khai vào server ứng dụng)
- Khởi động JBoss
- ant install (cài đặt ejbca)
- Tắt JBoss
- ant deploy (chép các tập tin cấu hình và ssl vào JBoss và triển khai lại ejbca)
- Khởi động lại JBoss

## Bước 12: Đăng nhập vào hệ thống

- Bật trình duyệt (browser) gõ <http://localhost:8080/ejbca> để vào CA.
- Để vào được trang quản trị (administration), cần import chứng nhận từ tập tin \$EJBCA\_HOME\p12\superadmin.p12 vào trình duyệt. Cách thêm vào như sau:
  - Với trình duyệt Mozilla Firefox 3.0
  - Tool > Options...
  - Chọn tab Advanced
  - Chọn View Certificates
  - Trong tab Your Certificates, chọn Import...
  - Chọn file superadmin.p12 trong thư mục \$EJBCA\_HOME\p12
  - Nhập mật khẩu mới, ví dụ 123456
  - Nhập tiếp mật khẩu: 123456

Triển khai trên môi trường Linux được thực hiện tương tự như trên, tuy nhiên sẽ không có **bước tùy chọn 6** (Thiết lập JBoss chạy như một Windows Service).

Các CA con được triển khai tương tự, sau đó gửi yêu cầu đến CA cấp trên để chứng nhận cho nó.