Chương trình KC-01:
Nghiên cứu khoa học phát triển công nghệ thông tin và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

Báo cáo kết quả nghiên cứu

PHẦN MỀM SINH VÀ KIỂM TRA CHỮ KÝ SỐ

Quyển 7A: "Một hệ chữ ký số có sử dụng RSA"

HÀ NỘI-2004

Báo cáo kết quả nghiên cứu

PHẦN MỀM SINH VÀ KIỂM TRA CHỮ KÝ SỐ

Quyển 7A: "Một hệ chữ ký số có sử dụng RSA"

Chủ trì nhóm thực hiện:
TS. Trần Duy Lai

Mục lục

Chương I
Chữ ký số dựa trên mật mã hiện đại

1-Giới thiệu

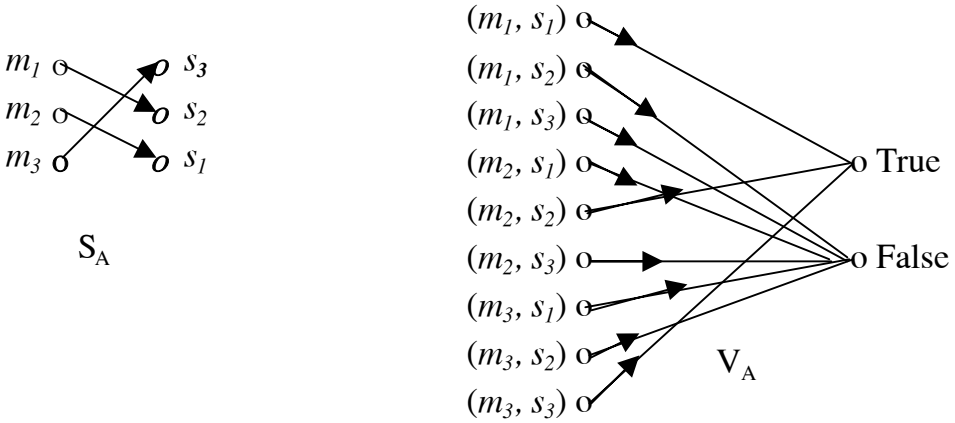
Một yếu tố gốc trong mật mã (*cryptographic primitive*) là nền tảng trong xác thực, chứng thực, và chống chối bỏ đó là chữ ký số. Mục đích của một chữ ký số là để cung cấp phương tiện cho một thực thể để gắn kết định danh của nó với một thông tin. Quá trình ký gây ra sự biến đổi thông điệp và một số thông tin bí mật được giữ bởi thực thể thành một cái được gọi là chữ ký. Mô tả chung của nó như sau.

Thuật ngữ và các ký hiệu

- M là tập các thông điệp mà có thể được ký.
- S là một tập các phần tử gọi là các chữ ký, có thể là các chuỗi nhị phân với độ dài xác định.
- S_A là một phép ánh xạ từ tập thông điệp M tới tập chữ ký S, và được gọi là phép ánh xạ ký (*signing transformation*) cho thực thể A (Alice). Phép ánh xạ S_A được giữ bí mật bởi A, và sẽ được sử dụng để tạo các chữ ký số cho các thông điệp từ tập M.
- V_A là một phép ánh xạ từ tập M x S tới tập {true, false}. V_A được gọi là phép ánh xạ kiểm tra (*verification transformation*) các chữ ký của A, đã được công bố công khai, và được sử dụng bởi các thực thể khác để kiểm tra các chữ ký đã tạo bởi A.

Định nghĩa Các phép ánh xạ S_A và V_A cung cấp một lược đồ chữ ký số (*digital signature scheme*) cho thực thể A. Đôi khi thuật ngữ kỹ thuật chữ ký số (*digital signature mechanism*) được sử dụng.

Ví dụ (*digital signature scheme*) M = $\{m_1, m_2, m_3\}$ và S = $\{s_1, s_2, s_3\}$. Hình ở bên trái dưới đây biểu diễn một hàm ký S_A từ tập M và hình ở bên phải biểu diễn hàm kiểm tra V_A tương ứng.



Hàm ký và kiểm tra của lược đồ chữ ký số.

Thủ tục ký

Thực thể A (*signer*) tạo một chữ ký cho một thông điệp $m \in M$ bằng cách thực hiện như sau:

1. Tính $s = S_A(m)$.
2. Chuyển giao cặp (m, s). s được gọi là chữ ký của thông điệp m.

Thủ tục kiểm tra

Để kiểm tra rằng một chữ ký s trên một thông điệp m đã được tạo bởi A, một thực thể B (*verifier*) thực hiện các bước sau:

1. Lấy hàm kiểm tra ký V_A của A.
2. Tính $u = V_A(m, s)$.
3. Chấp chữ ký đã được tạo bởi A nếu $u = true$, và bác bỏ chữ ký nếu $u = false$.

Nhận xét (*concise representation*) Các phép ánh xạ S_A và V_A thường được đặc trưng một cách gọn hơn bởi một khoá; tức là, có một lớp các thuật toán ký và kiểm tra ký được công bố công khai, và từng thuật toán đó được định danh bởi một khoá. Do vậy, thuật toán ký S_A của A được xác định bởi một khoá k_A và yêu cầu A phải giữ bí mật khoá k_A. Tương tự, thuật kiểm tra ký V_A của A được xác định bởi khoá l_A được đưa ra công khai.

Nhận xét (*handwritten signatures*, các chữ ký viết tay) Các chữ ký viết tay được coi như một lớp đặc biệt của các chữ ký số. Trường hợp này, tập các chữ ký S chỉ bao gồm một phần tử đó là chữ ký viết tay của A, gọi là s_A. Hàm kiểm tra chữ ký đơn giản kiểm tra xem chữ ký trên thông điệp được ký một cách có chủ ý bởi A là s_A.

Một đặc trưng không mong muốn, đó là chữ ký viết tay không phụ thuộc vào thông điệp (*message-dependent*). Do đó, cần có các bắt buộc thêm được áp đặt lên các kỹ thuật chữ ký số như thảo luận ở dưới đây.

Các tính chất yêu cầu đối với các hàm ký và kiểm tra ký

Có một vài tính chất mà các ánh xạ ký và kiểm tra ký phải thoả mãn.

(a) s là một chữ đúng của A trên thông điệp m nếu và chỉ nếu $V_A(m, s) = true$.

(b) Nó là không thể tính toán được đối với thực thể khác A để tìm một $s \in S$ mà $V_A(m, s) = true$, với $m \in M$.

Hình trên thể hiện tính chất (a). Có một đường mũi tên trong biểu đồ cho V_A từ (m_j, s_j) đến *true* tương ứng với một đường mũi tên từ m_j tới s_j trong biểu đồ S_A. Tính chất (b) đảm bảo tính an toàn cho phương pháp - chữ ký ràng buộc A duy nhất với thông điệp đã được ký.

Chưa có phương pháp nào chính thức chứng minh được rằng các lược đồ chữ ký số thoả mãn tồn tại tính chất (b) (mặc dù sự tồn tại được tin là đúng); tuy nhiên, cũng có một vài ứng cử viên rất tốt. Mục sau giới thiệu một lớp đặc biệt gồm các chữ ký số nảy sinh từ các kỹ thuật mã hoá khoá công khai. Sự mô tả về chữ ký số trong mục này là rất tổng quát, nó có thể được mở rộng chi tiết hơn nữa, như giới thiệu trong ở phía sau.

2- Chữ ký số từ hệ mã có thể đảo ngược

Trong mục này quan tâm đến một lớp các lược đồ chữ ký số mà được dựa trên hệ thống mã hoá khoá công khai có dạng đặc biệt.

Giả sử E_e là một ánh xạ mã hoá khoá công khai với không gian thông điệp M và không gian bản mã C. Hơn nữa, giả sử rằng M = C. Nếu D_d là ánh xạ giải mã tương ứng với E_e thì khi đó cả E_e và D_d đều là các phép hoán vị:

$$D_d(E_e(m)) = E_e(D_d(m)) = m, \text{ với } \forall\ m \in M.$$

Một lược đồ mã hoá khoá công khai theo kiểu này được gọi là reversible (có một lớp rộng hơn các chữ ký số mà có thể được coi là sự phát triển từ các thuật toán mật mã không thuận nghịch, như ở các mục 3.2.4 và 3.2.5). Chú ý rằng điều kiện M=C là cần thiết để cho đẳng thức là đúng với $\forall$ m $\in$ M; mặt khác, $D_d(m)$ sẽ không có nghĩa với $m \notin C$.

Cấu trúc cho một lược đồ chữ ký số

1. Giả sử M là không gian bản rõ của lược đồ chữ ký.
2. Giả sử C = M, không gian chữ ký S.
3. Giả sử (e, d) là cặp khoá của lược đồ mã hoá khoá công khai.
4. Định nghĩa hàm ký S_A là D_d. Điều này có nghĩa là chữ ký của một bản rõ $m \in M$ là $s = D_d(m)$.
5. Định nghĩa hàm kiểm tra ký V_A bởi

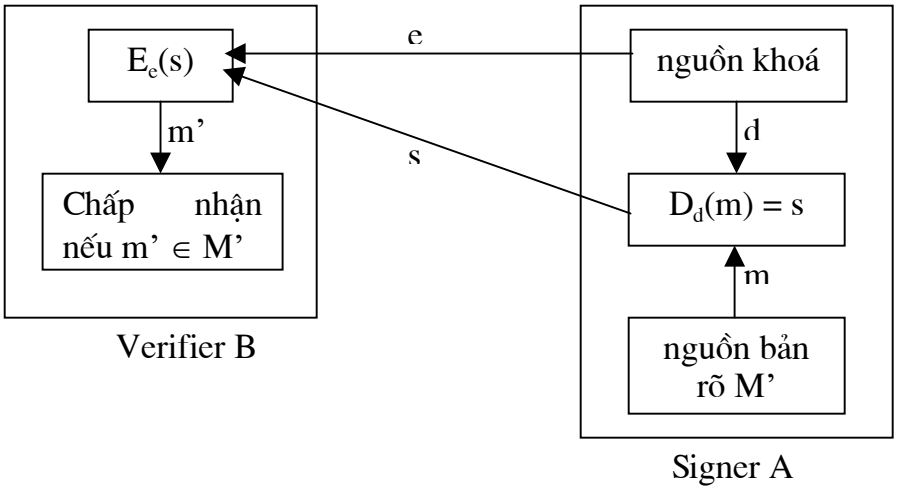
$$V_A(m,s) = \begin{cases} true, & \text{nếu } E_e(s) = m, \\ false, & \text{còn lại.} \end{cases}$$

Lược đồ chữ ký có thể được đơn giản hơn nếu A chỉ ký các bản rõ có cấu trúc đặc biệt, và cấu trúc này là được biết công khai. Cho M' là một tập con của M với các phần tử của M' có cấu trúc đặc biệt đã định nghĩa, do đó, M' chỉ chứa phần không đáng kể các bản rõ. Ví du, giả sử rằng M bao gồm tất cả các chuỗi nhị phân với độ dài là 2t, với t là số nguyên dương. Cho M' là tập con của M bao gồm tất cả các chuỗi mà t bits đầu tiên được lặp lại t bits còn lại (ví dụ, 101101 là thuộc tập M' với t=3). Nếu A chỉ ký các bản rõ nằm trong tập con M', điều này được dễ dàng nhận biết bởi một người kiểm tra ký.

Định nghĩa lại hàm kiểm tra ký V_A là

$$V_A(m,s) = \begin{cases} true, & \text{nếu } \mathrm{E}_e(s) \in M', \\ false, & \text{còn lại.} \end{cases}$$

Với các giả thiết mới này, A chỉ cần chuyển giao chữ ký s vì bản rõ m = E_e(s) có thể được khôi phục lại bằng cách áp dụng hàm kiểm tra ký. Một lược đồ như vậy được gọi là *digital signature scheme with message recovery*. Hình sau minh hoạ cách sử dụng hàm chữ ký này. Đặc điểm của sự lựa chọn các bản rõ với cấu trúc đặc biệt được tham khảo như là việc chọn các bản rõ có phần dư (*redundancy*).



Lược đồ chữ ký số khôi phục bản rõ.

Sự sửa đổi được trình bày ở trên (đưa độ dư vào) là một cái gì đó nhiều hơn phép thu gọn không gian được ký; nó là hoàn toàn cốt yếu nếu một ai đó hy vọng thoả mãn yêu cầu của tính chất (b) đối với các hàm ký và kiểm tra ký đã nêu ra ở trên. Hãy xem tại sao lại như vậy, chú ý rằng một thực thể B có thể chọn ngẫu nhiên một phần tử s ∈ S như là một chữ ký và áp dụng E_e để lấy u = E_e(s), vì S = M và E_e là công khai. B có thể lấy bản rõ m = u và chữ ký trên m là s, sau đó chuyển giao (m, s). Dễ dàng kiểm tra rằng s sẽ được chấp nhận như là một chữ ký đã được tạo bởi A cho m, nhưng trong khi đó A không đóng vai trò gì. Trong trường hợp này chúng ta nói rằng B đã giả mạo chữ ký của A. Đây là một ví dụ được gọi *existential forgery*. (B đã tạo ra chữ ký của A trên bản rõ mà bản rõ này không theo sự lựa chọn của B).

Nếu M' chỉ chứa một phần nhỏ các bản tin từ M, thì xác suất để một ai đó giả mạo được chữ ký của A theo cách này là rất nhỏ.

Nhận xét (chữ ký số so với sự tin cậy) Mặc dù các lược đồ chữ ký số dựa trên mã hoá khoá công khai thuận nghịch là rất hấp dẫn, chúng yêu cầu một phương pháp mã hoá như là một gốc mật mã. Có những tình huống mà một kỹ thuật chữ ký số

được yêu cầu nhưng sự mã hoá bị ngăn cấm. Trong những trường hợp như vậy thì các lược đồ chữ ký số này không thích hợp.

Chữ ký số trong thực tế

Với các chữ ký số thực sự các tác dụng trong thực tế, các phương án cụ thể của các khái niệm trước đó chắc chắn phải có thêm các tính chất khác. Một chữ ký số phải

1. dễ dàng tính toán bởi người ký (hàm ký là dễ dàng áp dụng);
2. dễ dàng kiểm tra bởi người khác (hàm kiểm tra ký là dễ dàng áp dụng); và
3. có khoảng thời gian phù hợp, ví dụ, an toàn về phương diện tính toán tránh được giả mạo cho đến khi chữ ký không cần thiết cho mục đích của nó.

Giải quyết tranh chấp

Mục đích của một chữ ký số (hoặc phương pháp ký bất kỳ) là để cho phép giải quyết các trạnh chấp. Ví dụ, một thực thể A phủ nhận đã ký lên một bản rõ hoặc thực thể B khẳng định sai một chữ ký trên một bản rõ là được tạo ra bởi A. Để khắc phục các vấn đề như vậy thì một Tổ chức tin cậy thứ ba (Trusted Third Party, TTP) hoặc quan toà (judge) được yêu cầu. TTT cần phải là một thực thể mà tất cả các bên tham gia đồng ý công nhận từ trước.

Nếu A phủ nhận rằng bản rõ m đang lưu giữ ở B là đã được ký bởi A, thì B có thể đưa ra chữ ký s_A trên m tới TTP cùng với m. Các quyết định của TTP sẽ ủng hộ B nếu $V_A(m, s_A)=true$ và ủng hộ A nếu ngược lại. B sẽ chấp nhận quyết định đó nếu B tin cậy rằng TTP có cùng phép ánh xạ kiểm tra ký V_A như A đã có. A sẽ chấp nhận quyết định đó nếu A tin cậy rằng TTP đã sử dụng V_A và tin rằng S_A không bị phá. Do vậy, việc giải quyết hợp lý tranh chấp yêu cầu các tiêu chuẩn sau phải được thoả mãn.

Những yêu cầu để giải quyết các chữ ký bị tranh chấp

1. S_A và V_A có các tính chất (a) và (b) đã nói trên.
2. TTP có bản sao đúng của V_A.
3. Phép ánh xạ ký S_A phải được giữ bí mật và vẫn còn an toàn.

Các tính chất này là cần thiết nhưng trong thực tế thì có thể không đảm bảo được chúng. Ví dụ, giả thiết cho rằng S_A và V_A có các đặc điểm như yêu cầu trong tính chất 1 có thể là không cho một lược đồ chữ ký đặc biệt. Một khả năng khác đó là A khẳng định sai rằng S_A đã bị phá. Để vượt qua các vấn đề này yêu cầu một phương pháp thoả thuận để phê chuẩn chu kỳ thời gian mà A sẽ chấp nhận trách nhiệm đối với ánh xạ kiểm tra. Một hoàn cảnh tương tự có thể xảy ra đối với việc huỷ bỏ thẻ tín dụng. Người sử dụng card chịu trách nhiệm đến tận khi thông báo với công ty phát hành card rằng card đã bị mất hoặc đã bị đánh cắp.

3 - Các định nghĩa và phân loại

Các định nghĩa

1. Chữ ký số (*digital signature*) là một chuỗi dữ liệu làm nhiệm vụ liên kết

một thông điệp (ở dạng số) với thực thể tạo ra nó.

2. Thuật toán sinh chữ ký số (*digital signature generation algorithm* hoặc *signature generation algorithm*) là một phương pháp tạo ra một chữ ký số.
3. Thuật toán kiểm tra chữ ký số (*digital signature verification algorithm* hoặc *verification algorithm*) là phương pháp để kiểm tra rằng một chữ ký số là đáng tin (tức là thực sự đã được tạo bởi thực thể đã được chỉ ra).
4. Lược đồ chữ ký số (*digital signature scheme* hoặc *mechanism*) bao gồm thuật toán sinh chữ ký và thuật toán kiểm tra chữ ký đi kèm.
5. Quy trình sinh chữ ký số (*digital signature signing process* hoặc *procedure*) bao gồm một thuật toán sinh chữ ký số (toán học), đi cùng với một phương pháp định khuôn dạng dữ liệu cho thông điệp để có thể ký được.
6. Tiến trình kiểm tra chữ ký số (*digital signature verification process* hoặc *procedure*) bao gồm một thuật toán kiểm tra ký, đi cùng với một phương pháp khôi phục dữ liệu từ thông điệp.

Trong chương này, hầu hết các mục (như chữ ký ElGamal, chữ ký Rabin) chỉ liên quan đơn thuần đến các lược đồ chữ ký số. Nhưng để sử dụng được một lược đồ chữ ký số trong thực tế thì còn cần nhiều hơn thế (chỉ có lược đồ chữ ký thôi thì chưa đủ), có nghĩa là cần đến quy trình sinh chữ ký số (thêm vào cách padding chẳng hạn). Có nhiều quy trình liên quan đến rất nhiều lược đồ khác nhau đã nổi lên như là các chuẩn thương mại thực sự; 2 quy trình như vậy, đó là ISO 9796 và PKCS #1, được trình bày trong riêng cho hệ chữ ký số RSA. Ký hiệu sử dụng cho phần còn lại của chương này được cung cấp trong bảng sau. Các tập và các hàm đã liệt kê trong bảng này là được công bố công khai.

Ký hiệu	ý nghĩa
M	tập các phần tử được gọi là không gian bản rõ.
M_S	tập các phần tử được gọi là không gian ký.
S	tập các phần tử được gọi là không gian chữ ký.
R	ánh xạ 1-1 từ M tới M_S gọi là hàm phần dư.
M_R	ảnh của R (tức là, $M_R=Im(R)$).
R^{-1}	nghịch ảnh của R (tức là, $R^{-1}: M_R->M$).
$\mathscr{R}$	tập các phần tử gọi là tập chỉ số chữ ký (*indexing set for signing*).
h	hàm một chiều trên miền M.
M_h	ảnh của h (tức là, $h: M->M_h$); $M_h \subseteq M_S$ được gọi là không gian giá trị băm.

Ký hiệu cho các kỹ thuật chữ ký số.

Chú ý (giải thích bảng trên)

(i) (không gian bản rõ) M là tập các phần tử mà từ đó một người ký có thể thêm vào chữ ký số.

(ii) (không gian ký) M_S là tập các phần tử mà từ đó các phép ánh xạ chữ ký (sẽ được trình bày sau này) được áp dụng. Các phép ánh xạ chữ ký không được áp dụng trực tiếp vào tập M.

(iii) (không gian chữ ký) S là tập các phần tử tương ứng với các bản rõ trong M. Những phần tử này được sử dụng để ràng buộc người ký với bản rõ.

(iv) (tập chỉ số) $\mathscr{R}$ được sử dụng để định danh các phép ánh xạ ký cụ thể.

Phân loại các lược đồ chữ ký số

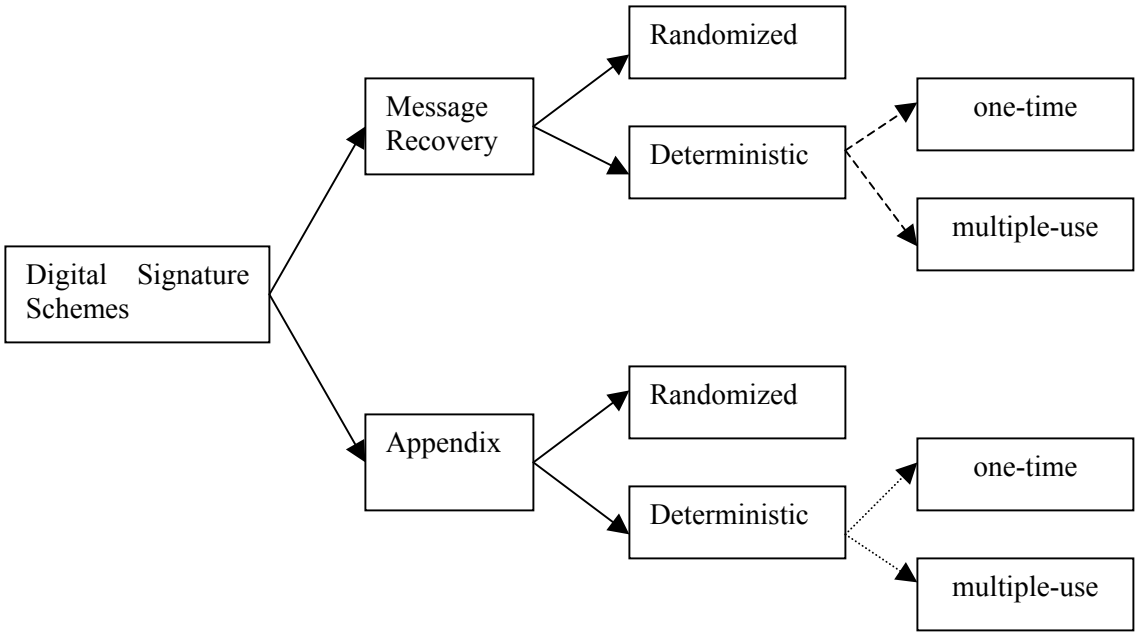
Trong hai mục sau sẽ trình bày 2 lớp tổng quát của các lược đồ chữ ký số, mà có thể tổng kết ngắn gọn như sau:

(i) Các lược đồ chữ ký số có phần phụ lục (*digital signature scheme with appendix*) yêu cầu bản rõ gốc có ở đầu vào của thuật toán kiểm tra chữ ký.

(ii) Các lược đồ chữ ký khôi phục bản rõ (*digital signature scheme with message recovery*) không yêu cầu bản rõ gốc trong đầu vào cho thuật toán kiểm tra chữ ký. Trong trường hợp này, bản rõ gốc tự được khôi phục lại từ chữ ký.

Định nghĩa Lược đồ chữ ký (khôi phục bản rõ hoặc có phần phụ lục) được gọi là lược đồ chữ ký số có tính ngẫu nhiên (*randomized digital signature scheme*) nếu $|\mathscr{R}| > 1$; và ngược lại, lược đồ chữ ký được gọi là tất định (*deterministic*).

Hình sau minh hoạ sự phân loại này. Kỹ thuật chữ ký số tất định có thể bị chia nhỏ hơn thành các lược đồ: *one-time signature schemes* và *multiple-use schemes*.



Phân loại các lược đồ chữ ký số.

4- Lược đồ chữ ký số cùng phụ lục

Lược đồ chữ ký số có phần phụ lục, như đã trình bày ở trên, được sử dụng phổ biến trong thực tế. Chúng dựa trên các hàm hash mật mã hơn là trên các hàm phần dư, và ít bị nguy hiểm hơn đối với các tấn công *existenial forgery*.

Định nghĩa Các lược đồ chữ ký số mà yêu cầu bản rõ là đầu vào của thuật toán kiểm tra ký được gọi là lược đồ chữ ký số có phần phụ lục (*digital signature schemes with appendix)*.

Các ví dụ về kỹ thuật tạo chữ ký số có phần phụ lục là các lược đồ chữ ký số DSA, ElGamal và Schnorr.

Thuật toán

Tóm tắt: từng thực thể tạo một khoá bí mật để ký các thông điệp, và tương ứng là một khoá công khai được sử dụng để các thực thể khác kiểm tra chữ ký.

1. Thực thể A lựa chọn một khoá bí mật, khoá này xác định tập $S_A=\{S_{A,k}: k\in\mathscr{R}\}$ của các phép ánh xạ. Mỗi $S_{A,k}$ là ánh xạ 1-1 từ M_h vào S và được gọi là một ánh xạ ký.
2. S_A định ra ánh xạ tương ứng V_A từ $M_h x\, S$ vào $\{true, false\}$ như sau:

$$V_A(\tilde{m}, s^*) = \begin{cases} true, \text{ nếu } \mathrm{S}_{\mathrm{A,k}}(\tilde{m}) = s^*, \\ false, \text{ trường hợp còn lại,} \end{cases}$$

   với $\forall\ \tilde{m} \in M_h, s^* \in S;\ \tilde{\mathrm{m}} = \mathrm{h(m)}$ với $\mathrm{m} \in \mathrm{M}$. V_A được gọi là ánh xạ kiểm tra ký và được tạo ra sao cho nó có thể tính được mà không cần biết gì về khoá bí mật của người ký.
3. Khoá công khai của A là V_A và khoá bí mật của A là tập S_A.

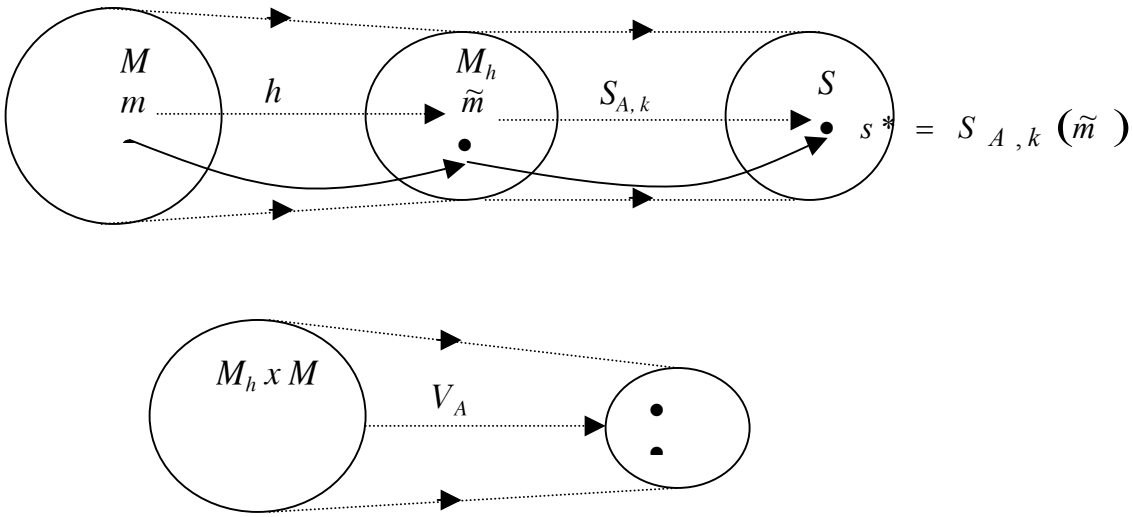
Thuật toán

Tóm tắt: thực thể A tạo ra một chữ ký $s \in S$ cho thông điệp $m \in M$, chữ ký này được kiểm tra bởi thực thể B về sau này.

1. *Sinh chữ ký*. Thực thể A thực hiện như sau;
   (a) Chọn một phần tử $k \in R$.
   (b) Tính $\tilde{m} = h(m)$ và $\mathrm{s}^* = \mathrm{S}_{\mathrm{A,k}}(\tilde{m})$.
   (c) Chữ ký của A trên m là s*. Cả m và s* là có thể tiếp cận được bởi các thực thể khác muốn kiểm tra chữ ký.
2. *Kiểm tra ký*. Thực thể B thực hiện như sau:
   (a) Nhận được khoá công khai $\mathrm{V_A}$ của A (có xác thực)
   (b) Tính $\tilde{m} = h(m)$ và $\mathrm{u} = \mathrm{V_A}(\tilde{\mathrm{m}}, \mathrm{s}^*)$.
   (c) Chấp nhận chữ ký nếu và chỉ nếu $u=true$.

Hình sau đưa ra biểu đồ của một lược đồ chữ ký số có phần phụ lục. Các tính chất dưới đây được yêu cầu đối với các ánh xạ ký và kiểm tra:

(i) với mỗi $k \in \mathcal{K}$, $S_{A,k}$ có thể tính được một cách hiệu quả;
(ii) V_A tính được có hiệu quả; và
(iii) nó không thể tính toán được đối với các thực thể khác A để tìm $m \in M$ và $s^* \in S$ thoả mãn $V_A(\tilde{m}, s^*) = true$, với $\tilde{m} = h(m)$.



Lược đồ chữ ký số cùng phụ lục.

Chú ý (về sử dụng các hàm băm) Hầu hết các lược đồ chữ ký số khôi phục bản rõ được áp dụng cho các thông điệp có độ dài xác định, trong khi đó thì lược đồ chữ ký số cùng phụ lục lại được áp dụng cho các thông điệp với độ dài tuỳ ý. Hàm một chiều (*one-way function*) h trong trên được chọn là hàm băm không va chạm (collision-free hash function). Một phương án khác thay cho việc băm là ngắt thông điệp thành các khối với độ dài xác định rồi từ đó có thể được ký riêng từng khối sử dụng lược đồ chữ ký số khôi phục bản rõ. Vì việc sinh chữ ký là tương đối chậm đối với phần lớn các lược đồ, và vì việc sắp xếp lại thứ tự nhiều khối đã ký có sự rủi ro an toàn, nên phương pháp được ưu tiên là sử dụng hàm băm.

5-Lược đồ chữ ký khôi phục thông báo

Các lược đồ chữ ký số trình bày trong mục này có đặc điểm đó là các thông điệp đã ký có thể được khôi phục lại từ chữ ký của nó. Trong thực tế, đặc điểm này chỉ sử dụng cho các thông điệp ngắn.

Định nghĩa Một lược đồ chữ ký khôi phục thông báo là một lược đồ chữ ký số mà việc biết trước thông điệp là không được yêu cầu cho thuật toán kiểm tra chữ ký.

Các ví dụ về các kỹ thuật cung cấp chữ ký số khôi phục thông báo là các lược đồ

ký khoá công khai: RSA, Rabin và Nyberg-Rueppel.

Thuật toán

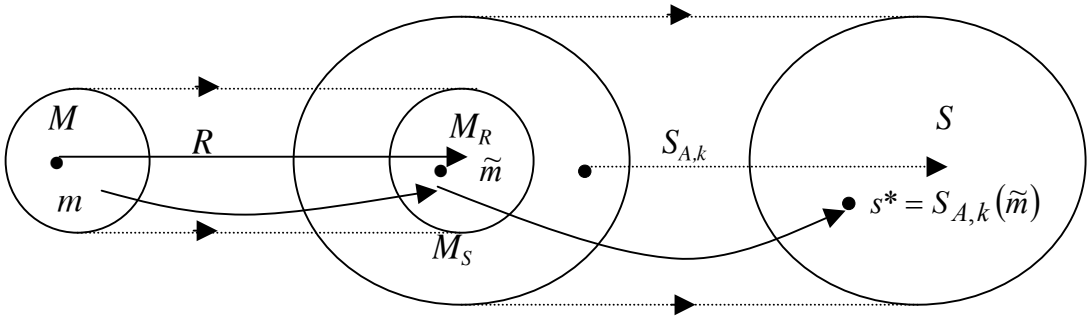
Tóm tắt: Mỗi một thực thể tạo một khoá bí mật được sử dụng ký các thông điệp, và một khoá công khai tương ứng được sử dụng bởi các thực thể khác để kiểm tra các chữ ký.

1. Thực thể A lựa chọn một tập các ánh xạ $S_A=\{S_{A,k}: k \in \mathscr{R}\}$. Mỗi $S_{A,k}$ là một ánh xạ 1-1 từ M_S tới S và được gọi là ánh xạ ký.
2. S_A định nghĩa một ánh xạ tương ứng V_A có tính chất sao cho $V_A o S_{A,k}$ là ánh xạ đồng nhất trên M_S với $\forall\ k \in \mathscr{R}$. V_A được gọi là ánh xạ kiểm tra và được tạo sao cho nó có thể được tính toán mà không cần biết gì về khoá bí mật của người ký.
3. Khoá công khai của A là V_A và khoá bí mật của A là tập S_A.

Thuật toán

Tóm tắt: thực thể A tạo một chữ ký $s \in S$ cho thông điệp $m \in M$, nó có thể được kiểm tra bởi thực thể B về sau. Thông điệp m được khôi phục lại từ s.

1. *Sinh chữ ký*. Thực thể A thực hiện như sau:
   (a) Chọn một phần tử $k \in \mathscr{R}$.
   (b) Tính $\tilde{m} = R(m)$ và $s^* = S_{A,k}(\tilde{m})$. (R là một hàm phần dư)
   (d) Chữ ký của A là s^*; nó có thể dùng được bởi các thực thể khác muốn kiểm tra chữ ký và khôi phục bản rõ m.
2. *Kiểm tra*. Thực thể B thực hiện như sau:
   (a) Nhận được khoá công khai V_A của A (có xác thực)
   (b) Tính $\tilde{m} = V_A(s^*)$.
   (c) Kiểm tra rằng $\tilde{m} \in M_R$. (Nếu $\tilde{m} \notin M_R$ thì bác bỏ chữ ký).
   (d) Khôi phục bản rõ m từ $\tilde{m}$ bằng cách tính $R^{-1}(\tilde{m})$.



Lược đồ chữ ký số khôi phục thông báo

Hình trên cung cấp môt biểu đồ về lược đồ chữ ký khôi phục thông báo. Dưới đây là các tính chất yêu cầu đối với các ánh xạ ký và kiểm tra ký.

(i) với mỗi $k \in \mathcal{K}$, $S_{A,k}$ tính được một cách hiệu quả;
(ii) V_A tính được hiệu quả; và
(iii) nó không thể tính toán được đối với các thực thể khác A để tìm $s^* \in M$ thoả mãn $V_A(s^*) \in M_R$.

Chú ý (*về hàm phần dư*) Hàm phần dư R và hàm ngược R^{-1} của nó là biết công khai. Chọn hàm phần dư R thích hợp là rất quan trọng để đảm bảo an toàn hệ thống. Để minh hoạ cho vấn đề này, giả sử rằng $M_R=M_S$. Giả sử R và $S_{A,k}$ là các song ánh từ M vào M_R và từ M_S vào S. Điều này nói lên rằng M và S có cùng số lượng phần tử. Do vậy, với mỗi $s^* \in S$, $V_A(s^*) \in M_R$, và sẽ dễ dàng tìm được các thông điệp m và các chữ ký s^* tương ứng mà sẽ được chấp nhận bởi thuật toán kiểm tra ký như sau:

1. Chọn ngẫu nhiên $k \in \mathcal{K}$ và ngẫu nhiên $s^* \in S$.
2. Tính $\tilde{m} = V_A(s^*)$.
3. Tính $m = R^{-1}(\tilde{m})$.

Phần tử s^* là chữ ký đúng của thông điệp m và được tạo ra mà không cần biết về tập các ánh xạ ký S_A.

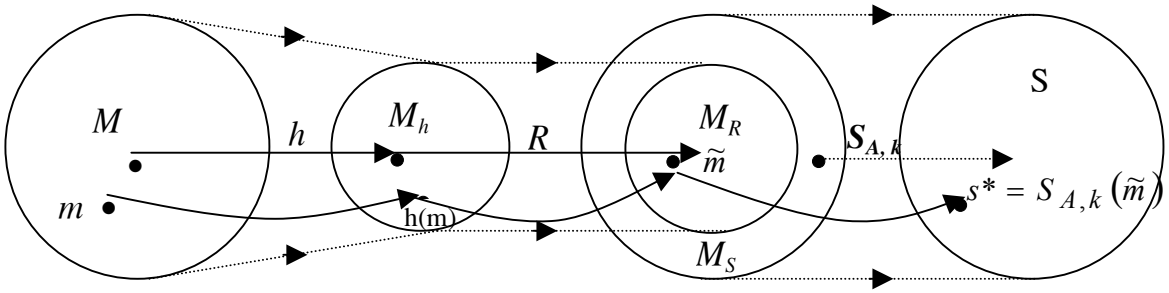
Ví dụ (*về hàm phần dư*) Giả sử $M = \{m: m \in \{0, 1\}^n\}$ với một số nguyên dương xác định n và $M_S = \{t: t \in \{0, 1\}^{2n}\}$. Định nghĩa $R: M \to M_S$ bởi $R(m) = m||m$, (với ký hiệu || là phép ghép dãy); như vậy, $M_R = \{m||m: m \in M\} \subseteq M_S$. Với giá trị n lớn, thì tỷ số $|M_R|/|M_S| = (1/2)^n$ là không đáng kể. Hàm phần dư này là thích hợp đảm bảo rằng không có cách chọn khôn ngoan nào cho s* về phía kẻ tấn công sẽ có được xác suất không nhỏ sao cho $V_A(s^*) \in M_R$.

Nhận xét (*về chọn hàm phần dư*) Mặc dù hàm phần dư R là được biết công khai và R^{-1} là dễ tính toán, sự lựa chọn R là rất quan trọng và không được làm độc lập với sự lựa chọn của các ánh xạ ký S_A. Mục "Các tấn công có thể đối với chữ ký RSA" cung cấp một ví dụ về hàm phần hàm phần dư làm tổn thương đến lược đồ chữ ký. Một ví dụ về hàm phần dư mà được chấp nhận là một chuẩn quốc tế được đưa ra ở sau này. Hàm phần dư này không phù hợp với tất cả các lược đồ chữ ký khôi phục thông báo, nhưng áp dụng được cho lược đồ chữ ký số RSA và Rabin.

Nhận xét (*về một lớp đặc biệt của các lược đồ chữ ký số*) ở trên đã trình bày một lớp các lược đồ chữ ký khôi phục thông báo được phát triển từ các phương pháp mã hoá khoá công khai thuận nghịch. Các ví dụ bao gồm các lược đồ mã hoá RSA và Rabin. Các kỹ thuật ký tương ứng được trình bày ở phía sau.

Chú ý (*tạo các chữ ký có phần phụ lục từ các lược đồ ký khôi phục thông báo*) Một lược đồ chữ ký khôi phục thông báo bất kỳ có thể biến thành một lược đồ một

lược đồ chữ ký cùng phụ lục đơn giản bằng cách băm (hash) các thông điệp và sau đó ký lên các giá trị hash đó. Như vậy, thông báo phải được yêu cầu như đầu vào của thuật toán kiểm tra ký. Biểu đồ cho tình huống này được minh hoạ trong hình sau. Hàm phần dư R là không quan trọng nữa cho đối với độ an toàn của lược đồ chữ ký, và có thể là các hàm 1-1 bất kỳ từ M_h tới M_S.



Lược đồ chữ ký khôi phục thông báo nhận được từ lược đồ chữ ký cùng phụ lục.

6-Các kiểu tấn công trên lược đồ ký

Mục đích chính của một kẻ tấn công là giả mạo chữ ký; điều đó có nghĩa là tạo các chữ ký mà sẽ được chấp nhận như là các chữ ký của các thực thể khác. Dưới đây đưa ra tập hợp các tiêu chuẩn theo ý nghĩa để phá vỡ lược đồ ký.

1. *total break (phá hoàn toàn)*. Kẻ tấn công không những có thể tính toán thông tin về khoá bí mật của người ký, mà còn tìm ra thuật toán ký có hiệu lực tương đương với thuật toán ký của người ký.
2. *selective forgery*. Kẻ tấn công có thể tạo ra được một chữ ký hợp lệ cho một thông điệp đặc biệt hoặc lớp các thông điệp ưu tiên đã lựa chọn. Việc tạo các chữ ký không trực tiếp đòi hỏi đến chính người ký.
3. *existential forgery*. Kẻ tấn công có thể giả mạo một chữ ký cho tối thiểu là một thông điệp. Kẻ tấn công có rất ít hoặc không điều khiển được thông điệp mà chữ ký của chúng là có được, và người ký có thể tham gia vào quá trình giả mạo.

Có 2 kiểu tấn công cơ bản dựa vào các lược đồ chữ ký số khoá công khai.

1. *key-only attacks*. Trong các tấn công kiểu này, kẻ tấn công chỉ biết khoá công khai của người ký.
2. *message attacks*. Kẻ tấn công có thể kiểm tra các chữ ký tương ứng không những các messages đã được biết mà còn được lựa chọn. Kiểu tấn công này có thể được chia nhỏ thành 3 lớp dưới đây:
   (a) *known-message attack*. Kẻ tấn công có các chữ ký của tập các thông điệp đã được biết nhưng không phải là được chọn bởi anh ta.
   (b) *chosen-message attack*. Kẻ tấn công lấy các chữ ký đúng từ danh sách các thông điệp đã được chọn trước khi tiến hành phá vỡ lược

đồ chữ ký. Tấn công này là *non-adaptive* (không thích nghi) theo nghĩa các messages được chọn trước khi có được bất kỳ một chữ ký nào. Các tấn công messages lựa chọn chống lại lược đồ chữ ký là tương tự với các tấn bản mã lựa chọn dựa vào các lược đồ mã hoá khoá công khai.

(c) *adaptive chosen-message attack*. Kẻ tấn công được phép sử dụng người ký như là một oracle; kẻ tấn công có thể yêu cầu các chữ ký của các messages tuỳ thuộc vào khoá công khai của người ký và hắn ta có thể yêu cầu các chữ ký của các messages phụ thuộc vào các thông báo hoặc các chữ ký đã có trước đó.

Chú ý (*về adaptive chosen-message attack*) Nói chung, *adaptive chosen-message attack* là dạng tấn công rất khó chống nhất. Có thể tưởng tượng rằng khi đã nhận đủ một lượng các messages và các chữ ký tương ứng, kẻ tấn công có thể từ đó rút ra mẫu và sau đó giả mạo chữ ký theo lựa chọn của mình. Trong khi *adaptive chosen-message attack* có thể không thể thực hiện dựng lên trong thực tế, một lược đồ ký được thiết kế tốt dù sao cũng phải được thiết kế để bảo vệ chống lại được khả năng này.

Chú ý (*xem xét về sự an toàn*) Mức độ an toàn đã yêu cầu trong một lược đồ chữ ký số có thể thay đổi tuỳ theo ứng dụng. Ví dụ, trong các trường hợp mà kẻ tấn công chỉ có thể sắp đặt tấn công *key-only attack* thì chỉ cần thiết kế một lược đồ ngăn chặn kẻ tấn công không thành công theo kiểu *selective forgery*. Trong các trường hợp kẻ tấn công có khả năng tấn công kiểu *messgage attack*, thì cần thiết phải đề phòng khả năng *existential forgery*.

Chú ý (*về các hàm hash và các tiến trình tạo chữ ký số*) Khi một hàm hash *h* được sử dụng trong một lược đồ chữ ký số (trường hợp thường xảy ra), thì *h* nên là một phần cố định của quy trình ký do vậy kẻ tấn công không thể lấy được một chữ ký đúng, thay thế *h* bằng một hàm hash yếu, và sau đó sắp đặt tấn công kiểu *selective forgery attack*.

7- Hàm băm

Một trong những thành tố cơ bản trong mật mã hiện đại là các hàm hash mật mã, thường được gọi là hàm băm một chiều (*one-way hash function*). Một định nghĩa được đơn giản hoá sẽ được trình bày ở dưới đây.

Định nghĩa Một hàm hash là một hàm tính toán hiệu quả thực hiện ánh xạ các chuỗi nhị phân với độ dài tuỳ ý vào các chuỗi nhị phân có độ dài cố định, được gọi là các giá trị hash (*hash-values*).

Với hàm hash có đầu ra là các giá trị hash *n*-bit (ví dụ, n = 128 hoặc 160) và có các tính chất mong muốn, xác suất để một chuỗi được chọn ngẫu nhiên ánh xạ vào một giá trị hash n-bit (ảnh) cụ thể là 2^{-n}. ý tưởng cơ bản là một giá trị hash dùng như là

một đại diện đầy đủ của một chuỗi đầu vào. Để sử dụng trong mật mã, hàm hash h thường được chọn sao cho về mặt tính toán là không thể tìm được 2 đầu vào khác nhau mà có cùng chung một giá trị hash (tức là, 2 đầu vào va chạm *x* và *y* thoả mãn *h(x)=h(y)*), và khi nhận được một giá trị hash cụ thể thì nó không thể tính toán để tìm ra một đầu vào (pre-image) x thoả mãn *h(x)=y*.

Nói chung, trong mật mã thường ứng dụng các hàm hash trong các chữ ký số và toàn vẹn dữ liệu. Trong các chữ ký số, một message dài thường được băm (sử dụng hàm hash công bố công khai, đã thoả thuận 2 bên) và chỉ giá trị hash được ký. Bên nhận nhận message và sau đó băm message đã nhận được, và kiểm tra rằng chữ ký nhận được là đúng với giá trị hash này. Điều này tiết kiệm cả không gian và thời gian so với việc ký trực tiếp vào message, tức là bao gồm chia các message thành các khối có kích thước phù hợp và ký từng khối riêng biệt. Điều cần chú ý ở đây rằng không có khả năng tìm 2 messages với cùng một giá trị hash là một yêu cầu an toàn, vì nếu khác điều này, thì chữ ký trên một message đã hash có thể giống với message đã hash khác, điều này cho phép người ký thực hiện ký một message và tại một thời điểm sau lại khẳng định đã ký một message khác (tráo đổi message).

Các hàm hash có thể được sử dụng cho việc toàn vẹn dữ liệu như sau. Giá trị hash tương ứng với một đầu vào đặc biệt được tính tại một thời điểm. Tính toàn vẹn của giá trị hash này là được bảo vệ bằng một cách nào đó. Tại một thời điểm tiếp theo, để kiểm tra rằng dữ liệu đầu vào là không bị thay đổi, giá trị hash được tính lại sử dụng message ban đầu và được so sánh là bằng với giá trị hash cũ. Những ứng dụng cụ thể bao gồm cả việc bảo vệ virus và phân phối phần mềm.

Một ứng dụng thứ 3 của các hàm hash là sử dụng trong các giao thức liên quan cam kết trước, bao gồm các lược đồ chữ ký số và các giao thức nhận dạng.

Các hàm hash được bàn tới ở trên được biết công khai và không có khoá bí mật (secret key). Khi được sử dụng để nhận biết xem message đầu vào đã bị thay đổi hay chưa thì chúng được gọi là *modification detection codes* (MDCs). Liên quan tới chúng còn có các hàm hash gồm có một secret key, và cung cấp tính xác thực nguyên vẹn ban đầu như là toàn vẹn dữ liệu, những hàm băm này được gọi là *message authentication codes* (MACs).

Chương II
Lược đồ chữ ký số RSA

1- Lược đồ chữ ký RSA

Không gian bản rõ và không gian bản mã của lược đồ mã hoá khoá công khai RSA đều là $Z_n = \{0, 1, 2, ..., n-1\}$ với $n = pq$ là tích của 2 số nguyên tố khác nhau được chọn ngẫu nhiên. Vì ánh xạ mã hoá là một song ánh, nên các chữ ký có thể được tạo bằng cách đảo ngược vai trò của mã hoá và giải mã. Lược đồ chữ ký RSA là một lược đồ chữ ký số tất định mà cung cấp việc khôi phục message. Không gian ký M_S và không gian chữ ký S đều là Z_n. Hàm phần dư $R: M->Z_n$ được lựa chọn và được biết công khai.

Thuật toán

Tóm tắt: Mỗi thực thể tạo một khoá công khai RSA và một khoá bí mật RSA tương ứng. Mỗi thực thể A thực hiện như sau:

1. Sinh ngẫu nhiên 2 số nguyên tố lớn khác nhau p và q, xấp xỉ về kích thước .
2. Tính $n = pq$ và $\phi = (p-1)(q-1)$.
3. Chọn ngẫu nhiên một số nguyên e, $1 < e < \phi$, thoả mãn $gcd(e, \phi) = 1$.
4. Sử dụng thuật toán Euclidean mở rộng để tìm số nguyên duy nhất d, $1 < d < \phi$, thoả mãn $ed \equiv 1 \pmod{\phi}$.
5. Khoá công khai của A là (n, e); Khoá bí mật của A là d.

Thuật toán

Tóm tắt: Thực thể A ký một message $m \in M$. Thực thể B kiểm tra chữ ký của A và khôi phục lại message m từ chữ ký.

1. *Tạo chữ ký*. Thực thể A thực hiện như sau:
   (a) Tính $\tilde{m} = R(m)$, là một số nguyên trong khoảng $[0, n-1]$.
   (b) Tính $s = \tilde{m}^d \bmod n$.
   (c) Chữ ký của A trên m là s.
2. *Kiểm tra ký*. Để kiểm tra chữ ký s của A và khôi phục message m, B làm như sau:
   (a) Nhận được khoá công khai của A là (n, e). (có xác thực)
   (b) Tính $\tilde{m} = s^e \bmod n$.
   (c) Kiểm tra rằng $\tilde{m} \in M_R$; nếu sai, không chấp nhận chữ ký.
   (d) Khôi phục $m = R^{-1}(\tilde{m})$.

Chứng minh việc kiểm tra chữ ký. Nếu s là một chữ ký của message m, thì $\tilde{m} = s^e \bmod n$, với $\tilde{m} = R(m)$. Vì $ed \equiv 1 \pmod{\phi}$, nên $s^e \equiv \tilde{m}^{ed} \equiv \tilde{m} \pmod n$. Cuối cùng tìm m, $R^{-1}(\tilde{m}) = R^{-1}(R(m)) = m$.

Ví dụ (*tạo chữ ký RSA với các tham số nhỏ*)
Sinh khoá. A chọn 2 số nguyên tố $p = 7927$, $q = 6997$, và tính $n = pq = 55465219$ và $\phi = 7796 \times 6996 = 55450296$. Chọn $e = 5$ và giải $ed = 5d \equiv 1 \pmod{55450296}$, tìm được $d = 44360237$. Khoá công khai của A là ($n = 55465219$, $e = 5$); và khoá bí mật của A là $d = 44360237$.

Tạo chữ ký. Để đơn giản, giả sử rằng $M = Z_n$ và hàm phần dư $R: M \to Z_n$ là ánh xạ $R(m) = m$ với $\forall\ m \in M$. Để ký lên message $m = 31229978$, A tính $\tilde{m} = R(m) = 31229978$, và tính chữ ký $s = \tilde{m}^d \bmod n = 31229978^{44360237} \bmod 55465219 = 30729435$.

Kiểm tra ký. B tính $\tilde{m} = s^e \bmod n = 30729435^5 \bmod 5465219 = 31229978$. Cuối cùng, B chấp nhận chữ ký vì $\tilde{m}$ đã đúng yêu cầu hàm phần dư (tức là, $\tilde{m} \in M_R$,). và khôi phục $m = R^{-1}(\tilde{m}) = 31229978$.

2-Các tấn công đối với chữ ký RSA

(i) Phân tích số nguyên

Nếu kẻ tấn công có thể phân tích *public modulus n* của thực thể A, thì sau đó hắn có thể tính ϕ và tiếp đó, sử dụng thuật toán Euclidean mở rộng, suy ra *private key d* từ ϕ và luỹ thừa công khai e bằng cách giải bài toán $ed \equiv 1 \pmod{\phi}$. Như vậy, có nghĩa là đã phá vỡ hoàn toàn hệ thống (*total break attack*). Để chống lại điều này, A phải chọn p và q để việc phân tích n là không thể thực hiện được.

(ii) Tính chất nhân của RSA

Lược đồ ký RSA (cũng như phương pháp mã hoá) có tính chất nhân, đôi khi được xem như tính chất đồng cấu (*homomorphic property*). Nếu $s_1 = m_1^d \bmod n$ và $s_2 = m_2^d \bmod n$ là các chữ ký trên các messages m_1 và m_2, tương ứng (hoặc hơn nữa là các chữ ký trên các messages sau khi đã sử dụng hàm phần dư), thì $s = s_1 s_2 \bmod n$ có tính chất là $s = (m_1 m_2)^d \bmod n$. Nếu $m = m_1 m_2$ có tính dư đúng (tức là, $m \in M_R$) thì s là chữ ký đúng của message m. Do đó, một điều rất quan trọng đó là hàm phần dư R không có tính chất nhân, tức là, với tất cả các cặp $a, b \in M$, $R(a.b) \neq R(a).R(b)$. Ví dụ sau chỉ ra rằng điều kiện này cho R là cần thiết nhưng không phải là điều kiện đủ để đảm bảo an toàn.

Ví dụ (*về hàm phần dư không an toàn*) Cho *n* là một RSA modulus và *d* là private key. Gọi $k = \lceil lg\ n \rceil$ là độ dài bit của *n*, và cho *t* là một số nguyên dương cố định thoả mãn $t < k/2$. Cho $w = 2^t$ và các messages là các số nguyên $m \in [1, n2^{-t}-1]$. Hàm phần dư R có dạng $R(m) = m2^t$ (*t* bits nhỏ nhất trong biểu diễn nhị phân của $R(m) = 0$). Như vậy, với các sự lựa chọn *n*, thì *R* sẽ không có tính chất nhân. Tấn công *existential forgery attack* trình bày ở trên có xác suất thành công là $(1/2)^t$. Nhưng với hàm phần dư này, sẽ bị tấn công kiểu *selective forgery attack* (rất nghiêm trọng), nó sẽ được trình bày dưới đây.

Giả sử một kẻ tấn công muốn giả mạo chữ ký trên message *m*. Kẻ tấn công biết *n* nhưng không biết *d*. Hắn có thể sắp đặt tấn công theo *chosen-message attack* để lấy chữ ký trên *m*. Sử dụng thuật toán Euclidean mở rộng cho *n* và $\tilde{m} = \mathrm{R(m)} = \mathrm{m2^t} = \mathrm{mw}$. Tại mỗi bước trong thuật toán mở rộng Euclidean, các số nguyên *x, y,* và *r* được tính sao cho thoả mãn biểu thức $xn + y\tilde{m} = r$. Có thể chỉ ra rằng tại một bước nào đó tồn tại *y* và *r* thoả mãn $|y| < n/w$ và $r < n/w$ với $w \le \sqrt{n}$. Nếu $y > 0$, lấy các số nguyên $m_2 = rw$ và $m_3 = yw$. Nếu $y < 0$, thì lấy các số nguyên $m_2 = rw$ và $m_3 = -yw$. Trong cả 2 trường hợp, m_2 và m_3 đều có độ dư cần thiết. Nếu các chữ ký $s_2 = m_2^d \bmod n$ và $s_3 = m_3^d \bmod n$ là chữ ký đúng từ người ký, thì kẻ tấn công có thể tính chữ ký cần giả mạo chữ ký trên m như sau:

- Nếu y > 0, tính $\dfrac{s_2}{s_3} = \dfrac{\mathrm{m_2^d}}{m_3^d} = \left(\dfrac{\mathrm{rw}}{\mathrm{yw}}\right)^d = \left(\dfrac{\mathrm{r}}{\mathrm{y}}\right)^d = \tilde{\mathrm{m}}^d \bmod \mathrm{n}$;
- Nếu y < 0, tính $\dfrac{s_2}{-s_3} = \dfrac{\mathrm{m_2^d}}{-m_3^d} = \left(\dfrac{\mathrm{rw}}{\mathrm{yw}}\right)^d = \left(\dfrac{\mathrm{r}}{\mathrm{y}}\right)^d = \tilde{\mathrm{m}}^d \bmod \mathrm{n}$.

Như vậy, kẻ tấn công có một message đã được ký theo lựa chọn của mình với độ dư cần thiết. Tấn công này là một ví dụ của tấn công *chosen-message attack* cung cấp *selective forgery*. Điều này nhấn mạnh rằng phải thận trọng khi lựa chọn hàm phần dư *R*.

3- Chữ ký RSA trong thực tế

(i) Bài toàn reblocking

Một đề nghị sử dụng RSA trong thực tế là thực hiện việc ký message và sau đó mã hoá chữ ký đó. Như vậy, điều quan tâm ở đây là quan hệ giữa các kích thước của các modulus khi thực hiện cài đặt thủ tục này. Giả sử rằng A muốn ký và sau đó mã hoá một message cho B. Và giả sử rằng (n_A, e_A) và (n_B, e_B) tương ứng là public key của A và B. Nếu $n_A > n_B$, thì có khả năng message m không được khôi phục bởi B, như là minh hoạ trong ví dụ dưới đây.

Ví dụ (*về bài toán reblocking*) Cho $n_A = 8377 \times 7499 = 62894113$, $e_A = 5$, và $d_A = 37726937$; và $n_B = 55465219$, $e_B = 5$, $d_B = 44360237$. Chú ý rằng $n_A > n_B$. Giả sử

rằng m = 1368797 là message cùng với phần dư để ký bằng private key của A và sau đó được mã hoá sử dụng public key của B. A thực hiện tính toán như sau:

1. $s = m^{d_A} \bmod n_A = 136879^{37726937} \bmod 62894113 = 59847900.$
2. $c = s^{d_B} \bmod n_B = 59847900^5 \bmod 55465219 = 38842235.$

Để khôi phục message m và kiểm tra chữ ký, B tính toán như sau:

1. $\hat{s} = c^{d_B} \bmod n_B = 38842235^{44360237} \bmod 55465219 = 4382618.$
2. $\hat{m} = \hat{s}^{e_A} \bmod n_A = 4382618^5 \bmod 62894113 = 54383568.$

Nhận thấy rằng $m \neq \tilde{m}$. Lý do ở đây là s lớn hơn modulus n_B. Xác suất xảy ra tình huống này là $(n_A - n_B)/n_A \approx 0.12$.

Có nhiều cách để khắc phục vấn đề reblocking.

1. *reordering*. Vấn đề giải mã sai sẽ không bao giờ xảy ra nếu phép toán sử dụng modulus nhỏ hơn được thực hiện trước. Có nghĩa là, nếu $n_A > n_B$, thì thực thể A nên mã hoá message sử dụng public key của B, và sau đó ký lên bản mã nhận được bằng private key của A. Tuy nhiên, trật sự của các phép toán lại là ký message trước và sau đó mã hoá chữ ký; Nếu để A mã hoá trước và sau đó mới ký, thì kẻ tấn công có thể bỏ được chữ ký và thay thế nó bằng chữ ký của hắn ta. Mặc dù kẻ tấn công sẽ không biết cái gì đã được ký (message m), nhưng cũng có thể có các tình huống có lợi cho hắn ta. Do vậy, reordering là một giải pháp không khôn ngoan.
2. *2 moduli cho một thực thể*. Mỗi thực thể sinh ra các moduli riêng biệt cho việc mã hoá và ký. Nếu modulus ký của mỗi người dùng là nhỏ hơn tất cả các moduli mã có thể, thì việc giải mã sai không bao giờ xảy ra. Điều này có thể được bảo đảm bởi yêu cầu moduli giải mã là các số (t + 1)-bits và các số để ký là t-bits.
3. *Quy định dạng của modulus*. Trong phương pháp này, chọn các số nguyên tố p và q sao cho modulus n có một dạng đặc biệt: bit lớn nhất là 1 và k bits sau đó lấy giá trị 0. Một modulus n t-bits có thể được tìm theo cách sau. Với n có dạng theo yêu cầu thì $2^{t-1} \leq n < 2^{t-1} + 2^{t-k-1}$. Chọn ngẫu nhiên số nguyên tố p có $\lceil t/2 \rceil$-bits, và tìm số nguyên tố q trong khoảng $\lceil 2^{t-1}/p \rceil$ và $\lfloor (2^{t-1} + 2^{t-k-1})/p \rfloor$; khi đó n = pq là một modulus có dạng đã yêu cầu (xem ví dụ sau). Sự lựa chọn này cho modulus n không hoàn toàn chống được vấn đề giải mã sai, nhưng làm cho xác suất xảy ra là một số nhỏ không đáng kể. Giả sử n_A là thoả mãn các điều kiện trên và $s = m^{d_A} \bmod n_A$ là chữ ký trên m. Hơn nữa giả sử rằng có s có giá trị 1 ở một trong k + 1 bits cao, nhưng không phải bit cao nhất. Khi đó s, vì nó nhỏ hơn n_A, phải có giá trị 0 ở vị trí bit cao nhất và tất yếu phải nhỏ hơn modulus khác có dạng tương tự. Xác suất để s không có giá trị 1 ở k + 1 bit cao, ngoài vị trí cao nhất, là nhỏ hơn $(1/2)^k$, sẽ là rất nhỏ nếu chọn k khoảng 100.

Ví dụ (về *quy định dạng của modulus*) Giả sử muốn tạo một modulus n 12-bits thoả mãn bit cao nhất là 1 và các bits tiếp theo là k = 3 bits lấy giá trị 0. Bắt đầu chọn số nguyên tố p 6-bits p = 37. Chọn số nguyên tố q nằm trong khoảng $\lceil 2^{11}/p \rceil$

= 56 và $\lfloor (2^{11} + 2^8)/p \rfloor$ = 62. Như vậy, các khả năng có thể của p là 59 và 61. Nếu chọn p = 59, thì n = 37 x 59 = 2183, có biểu diễn nhị phân là 100010000111. Nếu chọn q = 61, thì n = 37 x 61 = 2257, có biểu diễn nhị phân là 100011010001.

(ii) Hàm phần dư

Để tránh được tấn công kiểu existential forgery attack trên lược đồ chữ ký RSA, yêu cầu phải có một hàm phần dư R phù hợp. Mục sau đưa ra một hàm phần dư như vậy mà được chấp nhận như là một chuẩn quốc tế. Lựa chọn cẩn thận một hàm phần dư là cốt yếu để an toàn hệ thống.

(iii) Lược đồ chữ ký RSA cùng phụ lục

Trên đây đã trình bày cách sửa lược đồ chữ ký khôi phục thông báo để nhận được lược đồ chữ ký cùng phụ lục. Ví dụ, nếu sử dụng thuật toán hash MD5 để hash các messages có độ dài bit tuỳ ý thành chuỗi bits có độ dài 128, sau đó Thuật toán ký RSA được sử dụng để ký các giá trị hash này. Nếu RSA modulus n có độ dài k-bit, thì một hàm phần dư phù hợp *R* với yêu cầu đầu vào là số nguyên 128-bit và đầu ra là số nguyên k-bit. Sau đây có trình bày một phương pháp để làm việc này mà thường được sử dụng trong thực tế (PKCS#1).

(iv) Các đặc điểm hiệu xuất của việc sinh chữ ký và kiểm tra chữ ký

Cho $n = pq$ là 2k-bit RSA modulus với p và q là 2 số nguyên tố k-bit. Tính một chữ ký $s = m^d \bmod n$ cho message m yêu cầu $O(k^3)$ phép toán (sẽ được trình bày ở chương IV). Vì người ký biết p và q, anh ta có thể tính $s_1 = m^d \bmod p$, $s_2 = m^d \bmod q$, và xác định s sử dụng Định lý phần dư Trung hoa. Mặc dù độ phức tạp tính toán của thủ tục này vẫn còn $O(k^3)$, nhưng nó có kết quả tốt hơn nhiều trong một số trường hợp.

Việc kiểm tra các chữ ký nhanh hơn nhiều so với ký nếu luỹ thừa công khai được chọn là một số nhỏ. Nếu thực hiện như vậy, thì kiểm tra ký chỉ còn $O(k^2)$ phép toán. Các giá trị được gợi ý cho luỹ thừa công khai e trong thực tế là 3 và $2^{16} + 1$ (việc chọn $e = 2^{16} + 1$ được dựa trên thực tế đó là e là một số nguyên tố và $\tilde{m}^e \bmod n$ có thể được tính chỉ bằng 16 phép bình phương modular và 1 phép tính nhân); tất nhiên, p và q phải được chọn thoả mãn điều kiện $\gcd(e, (p - 1)(q - 1)) = 1$.

Lược đồ ký RSA là lý tưởng phù hợp với một số trường hợp mà việc kiểm tra chữ ký là phép toán được thực hiện nhiều lần. Ví dụ, khi Trusted Third Party (TTP) tạo ra một chứng chỉ số (public-key certificate) cho một thực thể A, nó chỉ yêu cầu sinh một chữ ký, và chữ ký này có thể được kiểm tra nhiều lần bởi các thực thể khác nhau.

(v) Lựa chọn tham số

Đến năm 1996, số bit nhỏ nhất là 768 bits được giới thiệu cho RSA signature moduli. Một modulus tối thiểu là 1024 bits được giới thiệu cho các chữ ký còn

được sử dụng trong một thời gian dài hơn nữa hoặc chúng là quan trọng cho sự an toàn tổng thể của một mạng lớn. Còn lại là thận trọng để nhận thức sự phát triển của bài toán phân tích số nguyên, và để chuẩn bị điều chỉnh các tham số tương ứng.

Không có điểm yếu trong lược đồ ký RSA được thông báo khi sử dụng luỹ thừa công khai e được chọn là số nhỏ như là 3 và $2^{16} + 1$. Không khuyến cáo chỉ ra giới hạn độ lớn của luỹ thừa bí mật d để tăng hiệu lực của việc sinh chữ ký.

(vi) Bandwidth efficiency

Bandwidth efficiency của các chữ ký với lược đồ chữ ký khôi phục thông báo dựa trên tỷ số của logarithm (cơ số 2) kích thước không gian ký M_S với logarithm (cơ số 2) kích thước không gian ảnh của hàm phần dư M_R. Do đó, b*andwidth efficiency* được xác định bởi hàm phần dư R. Với lược đồ chữ ký số RSA (và Rabin), hàm phần dư chỉ ra trong chuẩn ISO/IEC 9796 lấy các messages k-bit và mã chúng thành các phần tử 2k-bit trong M_S, từ đó có được chữ ký dạng 2k-bit. *Bandwidth efficiency* trong trường hợp này là 1/2. Ví dụ, với một modulus có kích thước là 1024 bits, thì kích thước lớn nhất của một message có thể được ký là 512 bits.

(vii) System-wide parameters

Mỗi thực thể phải có một RSA modulus riêng biệt; sẽ không an toàn khi sử dụng một modulus trên toàn hệ thống gọi là *system-wide parameters*, khi đó sẽ bị tấn công kiểu common modulus attack. Luỹ thừa công khai e có thể là một *system-wide parameter*, giống như trong một số ứng dụng khác.

(viii) Sự đối lập giữa các messages ngắn và dài

Giả sử n là RSA modulus 2k-bit mà được sử dụng để ký các messages k-bit (tức là, bandwidth efficiency là 1/2). Giả sử thực thể A muốn ký một message *m kt*-bit. Có một cách là chặt *m* ra các khối *k*-bit sao cho $m = m_1||m_2||... ||m_t$ và ký từng khối riêng biệt. Như vậy, yêu cầu *bandwidth* sẽ là 2*kt* bits. Thay cho cách này, A thực hiện hash message *m* thành các chuỗi bit có độ dài $l \leq k$ và ký lên giá trị hash. Yêu cầu *bandwidth* của chữ ký này là $kt + 2k$, với *kt* là độ dài bit của message *m*. Vì $kt + 2k \leq 2kt$, với $\forall\ t \geq 2$, như vậy, hầu hết các phương pháp có bandwidth hiệu quả nhất là để sử dụng chữ ký RSA cùng phụ lục. Với một message có độ dài bit tối đa là *k*-bits, thì nên sử dụng lược đồ ký RSA khôi phục thông báo.

4-Định dạng chuẩn ISO/IEC 9796

ISO/IEC 9796 được công bố vào năm 1991 bởi Tổ chức các chuẩn quốc tế (ISO - International Standards Organization) như là một chuẩn quốc tế đầu tiên về các chữ ký số. Nó chỉ rõ một quy trình chữ ký số sử dụng một kỹ thuật chữ ký số hỗ trợ khôi phục message.

Các đặc điểm chính của chuẩn ISO/IEC 9796 như sau:

(i) Nó được dựa trên mật mã khoá công khai;

(ii) thuật toán chữ ký cụ thể không được chỉ ra, nhưng nó phải ánh xạ k bits tới k bits;
(iii) nó được sử dụng để ký các messages có độ dài bị giới hạn và không sử dụng hàm hash mật mã;
(iv) nó hỗ trợ khôi phục message và
(v) nó chỉ rõ cách kéo dài message (padding), khi cần thiết.

Các ví dụ về các kỹ thuật phù hợp với chuẩn này là RSA và Rabin cải biên. Các phương pháp cụ thể sử dụng cho padding, hàm phần dư, và cắt ngắn trong chuẩn ISO/IEC 9796 ngăn ngừa nhiều phương pháp giả mạo chữ ký bình thường. Bảng sau cung cấp các ký hiệu cho mục này.

Ký hiệu	**ý nghĩa**
k	độ dài bit của chữ ký.
d	độ dài bit của message m sẽ được ký; thoả mãn $d \le 8\lfloor (k+3)/16 \rfloor$.
z	số bytes của message đã được kéo dài; $z = \lceil d/8 \rceil$.
r	số bits thêm vào; $r = 8z - d + 1$.
t	số nguyên nhỏ nhất thoả mãn rằng một chuỗi $2t$ bytes có tối thiểu k - 1 bits; $t = \lceil (k-1)/16 \rceil$.

Ký hiệu cho chuẩn ISO/IEC 9796.

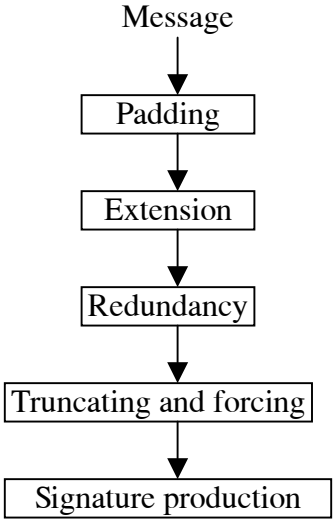
Ví dụ (*về các giá trị của tham số chuẩn ISO/IEC 9796*) Bảng dưới đây đưa ra danh sách ví dụ các giá trị của các tham số trong quy trình ký một message 150-bit và một chữ ký 1024-bit.

Tham số	k (bits)	d (bits)	z (bytes)	r (bits)	t (bytes)
Giá trị	1024	150	19	3	64

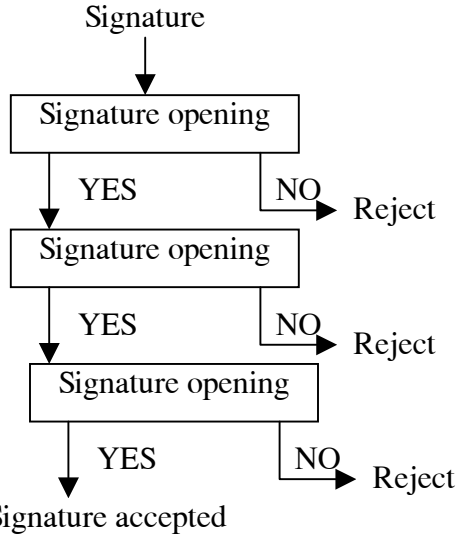
(i) Quy trình ký của chuẩn ISO/IEC 9796

Quy trình ký bao gồm 5 bước như ở phía bên trái hình sau.

(a) ISO/IEC 9796 signature process



(b) ISO/IEC 9796 verification process



1. *padding*. Nếu m là một message, tạo message đã được kéo dài MP = $0^{r-1}||m$ với $1 \le r \le 8$, sao cho số bits trong MP là bội của 8. Số bytes của MP là z: MP = $m_z||m_{z-1}||...||m_2||\ m_1$ với mỗi m_i là 1 byte.
2. *message extension* (mở rộng message). Message đã được mở rộng, ký hiệu là ME, nhận được từ MP bằng cách nối lặp lại MP về phía bên trái của nó cho đến khi được chuỗi có t bytes: ME = $ME_t||ME_{t-1}||...||ME_2||ME_1$ (với mỗi ME_i là 1 byte). Nếu t không là bội của z, thì các bytes cuối cùng được nối vào là một tập con các bytes của MP, tập con này gồm các bytes liên tiếp của Mp tính từ bên phải. Chính xác hơn, $ME_{i+1} = m_{(i \bmod z)+1}$ với $0 \le i \le t - 1$.
3. *message redundancy* (phần dư message). Phần dư được thêm vào ME để nhận được chuỗi các byte MR = $MR_{2t}||MR_{2t-1}||...||MR_2||MR_1$ như sau. MR có được bằng cách chèn t bytes của ME với t bytes phần dư và và sau đó điều chỉnh byte MR_{2z} của chuỗi thu được. Chính xác hơn, $MR_{2i-1} = ME_i$ và $MR_{2i} = S(MR_i)$ với $1 \le i \le t$, ở đây S(u) được gọi là *shadow function* của byte u, và được định nghĩa như sau. Nếu $u = u_2||u_1$ với u_1 và u_2 là các chuỗi nhỏ (các chuỗi có độ dài bit là 4), thì $S(u) = \pi(u_2)||\ \pi(u_1)$ với π là phép hoán vị $\pi = \begin{pmatrix} 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ A\ B\ C\ D\ E\ F \\ E\ 3\ 5\ 8\ 9\ 4\ 2\ F\ 0\ D\ B\ 6\ 7\ A\ C\ 1 \end{pmatrix}$. (Để ngắn gọn, π được viết bằng các chuỗi bít ngắn được biểu diễn bởi các kí tự hexa.) Cuối cùng MR được lấy bởi thay thế MR_{2z} bằng $r \oplus MR_{2z}$. (Mục đích của MR_{2z} là để cho phép người kiểm tra chữ ký có thể khôi phục được độ dài d của message. Vì d = 8z - r + 1, nên chỉ cần biết z và r. Những giá trị này có thể suy ra từ MR.)
4. *truncation and forcing* (cắt ngắn và ràng buộc). Tạo số nguyên trung gian k-bit IR từ MR như sau:
   (a) nối thêm một bit có giá trị 1 vào bên trái của k - 1 bits thấp nhất của MR.

(b) sửa byte thấp nhất $u_2||u_1$ của kết quả, thay thế nó bởi $u_1||0110$. (Làm việc này để đảm bảo rằng IR ≡ 6 (mod 16).)

5. *signature production* (tạo chữ ký). Một kỹ thuật ký được sử dụng để ánh xạ các số nguyên k-bit thành các số nguyên k-bit (và cho phép khôi phục message). IR được ký sử dụng kỹ thuật này; ký hiệu s là kết quả ký.

Chú ý (*về RSA, Rabin*) Chuẩn ISO/IEC 9796 có ý định dùng cho các kỹ thuật chữ ký số RSA và Rabin. Với các lược đồ đặc biệt này, tạo chữ ký được mô tả rõ ràng. Cho e là luỹ thừa công khai của các thuật toán RSA và Rabin, n là modulus, và d là luỹ thừa bí mật. Trước hết, tạo phần tử đại diện RR như sau:

(i) bằng IR, nếu e là lẻ, hoặc nếu e là chẵn và ký hiệu Jacobi của IR (coi như một số nguyên) đối với modulus n là 1;

(ii) bằng IR/2 nếu e là chẵn và ký hiệu Jacobi của IR đối với n là -1.

Chữ ký của m là $s = (RR)^d \bmod n$. Chuẩn ISO/IEC 9796 qui định rằng chữ ký s nên nhỏ hơn $(RR)^d \bmod n$ và $n - ((RR)^d \bmod n)$.

(ii) Qui trình kiểm tra của chuẩn ISO/IEC 9796

Qui trình kiểm tra một chữ ký số theo chuẩn ISO/IEC 9796 có thể được chia ra 3 giai đoạn, như ở bên phải hình trên.

1. *signature opening*. Giả sử s là chữ ký cần kiểm tra. Thì các bước sau được thực hiện:
   (a) áp dụng ánh xạ kiểm tra công khai cho s để khôi phục số nguyên IR'.
   (b) Không chấp nhận chữ ký nếu IR' không phải một chuỗi k bits với bit lớn nhất là 1, hoặc nếu 4 bit nhỏ nhất không có giá trị 0110.
2. *message recovery*. Chuỗi MR' 2t bytes được xây dựng từ IR' bằng cách thực hiện các bước sau:
   (a) Cho X là chuỗi k - 1 bits nhỏ nhất của IR'.
   (b) Nếu $u_4||\ u_3||\ u_2||0110$ là 4 chuỗi có 4 bit nhỏ nhất của X, thay thế byte nhỏ nhất của X bởi $\pi^{-1}(u_4)||u_2$.
   (c) MR' được lấy bằng cách thêm vào X các bit 0 (từ 0 đến 15 số) để chuỗi kết quả là 2t bytes.

   Các giá trị z và r được tính như sau:
   (a) Từ 2t bytes của MR', tính t tổng $MR'_{2i} \oplus S(MR'_{2i-1})$, $1 \le i \le t$. Nếu các tổng đều bằng 0, thì không chấp nhận chữ ký.
   (b) Cho z là giá trị nhỏ nhất của i để $MR'_{2i} \oplus S(MR'_{2i-1}) \neq 0$.
   (c) Cho r là chuỗi 4-bit thấp nhất (least significant) của tổng đã tìm được ở bước (b). Không chấp nhận chữ ký nếu giá trị hexa của r không nằm trong khoảng 1 và 8.

   Từ MR', chuỗi z-byte MP' được xây dựng như sau:
   (a) $MP'_{2i} = MR'_{2i-1}$ với $1 \le i \le z$.
   (b) Không chấp nhận chữ ký nếu r - 1 bits lớn nhất của MP' không phải tất cả có giá trị 0.
   (c) Giả sử M' là chuỗi 8z - r + 1 bits nhỏ nhất của MP'.

3.redundancy checking. Chữ ký s được kiểm tra như sau:

(a) Từ M' xây dựng chuỗi MR'' bằng cách áp dụng các bước message padding, message extension, và message redundancy của tiến trình ký.
(b) Chấp nhận chữ ký nếu và chỉ nếu k - 1 bits nhỏ nhất của MR'' là bằng k - 1 bits nhỏ nhất của MR'.

5-Định dạng chuẩn PKCS #1

Các chuẩn mật mã khoá công khai (PKCS) là một bộ các đặc tả bao gồm các kỹ thuật cho mã hoá và chữ ký RSA. Trong mục này trình bày quy trình tạo chữ ký số trong chuẩn PKCS #1 ("RSA Encryption Standard").

Kỹ thuật chữ ký số trong chuẩn PKCS #1 không sử dụng đặc tính khôi phục message của lược đồ chữ ký RSA. Nó yêu cầu một hàm hash (hoặc MD2 hoặc MD5), do đó, là lược đồ chữ ký số cùng phụ lục. Bảng sau liệt kê các ký hiệu sử dụng trong mục này. Các chữ hoa sử dụng cho các chuỗi các ký tự hex. Nếu X là một chuỗi các ký tự hex, thì X_i là ký tự hex thứ i tính từ phía trái lại.

Ký hiệu	ý nghĩa	Ký hiệu	ý nghĩa
k	độ dài octet của n ($k \geq 11$)	EB	khối mã
n	modulus, $2^{8(k-1)} \leq n < 2^{8k}$	ED	dữ liệu đã mã
p, q	các thừa số nguyên tố của n	octet	chuỗi bit có độ dài 8
e	luỹ thừa công khai	ab	giá trị octet dạng hexa
d	luỹ thừa bí mật	BT	kiểu khối
M	message	PS	chuỗi padding
MD	message digest	S	chữ ký
MD'	so sánh với message digest	||X||	độ dài octets của X

Các ký hiệu cho chuẩn PKCS #1.

(i) Định dạng dữ liệu chuẩn PKCS #1

Dữ liệu là một octet string D, với $||D|| \leq k-11$. BT là một octet mã biểu diễn hexa là 00 hoặc 01. PS là một octet string với $||PS|| = k-3-||D||$. Nếu BT = 00, thì tất cả các octets trong PS là 00; nếu BT = 01, thì tất cả các octets trong PS là ff. Khối dữ liệu đã định dạng (được gọi là khối mã) là EB = 00||BT||PS||00||D.

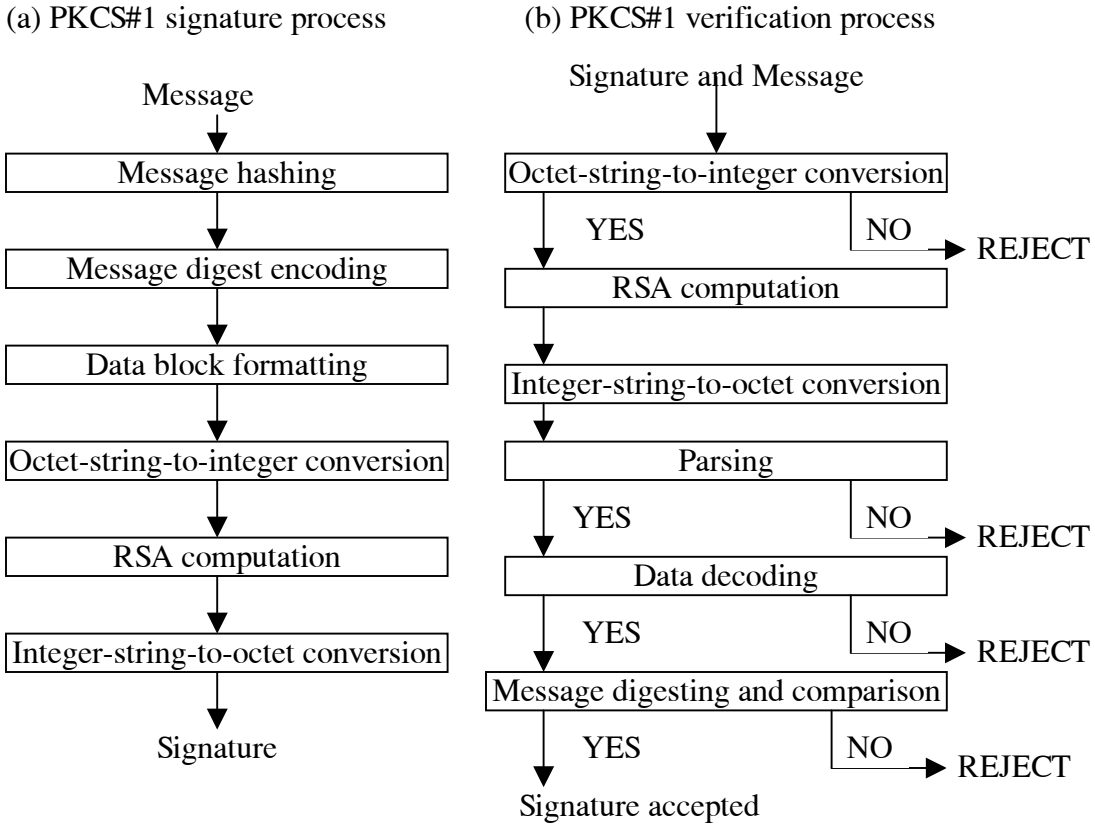
Chú ý (*về sự hợp lý trong định dạng dữ liệu*)

(i) Đầu khối là 00 để chắc chắn rằng octet string EB, khi được hiểu như số nguyên, là nhỏ hơn modulus n.
(ii) Nếu kiểu khối là BT = 00, thì hoặc D phải bắt đầu bằng một octet khác 0 hoặc độ dài của nó phải biết, để cho phép phân tích EB không mơ hồ.
(iii) Nếu BT = 01, thì phân tích không mơ hồ là luôn có thể.
(iv) Với lý do ở (iii), và để cản trở một số tấn công nào đó trên kỹ thuật chữ ký, BT = 01 được khuyến cáo.

Ví dụ (*về định dạng dữ liệu chuẩn PKCS #1 cho các giá trị đặc biệt*) Giả sử rằng modulus n là 1024-bit (k = 128). Nếu ||D|| = 20 octets, thì ||PS|| = 105 octets, và ||EB|| = 128 octets.

(ii) Quy trình ký của chuẩn PKCS #1

Tiến trình ký bao gồm các bước như ở bên trái hình sau.



Đầu vào của quy trình ký là message M, luỹ thừa bí mật của người ký là d và modulus n.

1. *message hashing*. Hash message M sử dụng thuật toán message-digest đã chọn để nhận được octet string MD.
2. *message digest encoding*. MD và định danh thuật toán hash được kết hợp thành một giá trị ASN.1 (Abstract Syntax Notation One) và sau đó được mã dạng BER (Basic Encoding Rules) để nhận được một chuỗi dữ liệu dạng octet string D.
3. *data block formating*. Với chuỗi dữ liệu vào là D, sử dụng định dạng dữ liệu ở trên để tạo ra octet string EB.
4. *octet-string-to-integer conversion*. Giả sử các octets của EB là $EB_1 \| EB_2 \| ... \| EB_k$. Định nghĩa $\tilde{EB}_i$ là số nguyên mà biểu diễn nhị phân là octet EB_i (bit nhỏ nhất ở bên phải). Biểu diễn số nguyên EB là $\sum_{i=1}^{k} 2^{8(k-i)} \tilde{EB}_i$ (vì $EB_1 = 00$ và $n \geq 2^{8(k-1)}$, nên $0 \leq m < n$.).
5. *RSA computation*. Tính $s = m^d \bmod n$.

6. *integer-to-octet-string conversion*. Chuyển số nguyên s thành một octet string ED = ED_1|| ED_2||...|| ED_k, với các octets ED_i thoả mãn $s = \sum_{i=1}^{k} 2^{8(k-i)} \widetilde{ED}_i$. Chữ ký là S = ED.

(iii) Quy trình kiểm tra chữ ký của chuẩn PKCS #1

Quy trình kiểm tra chữ ký bao gồm các bước ở bên phải hình trên. Đầu vào của quy trình kiểm tra là message M, chữ ký S, luỹ thừa công khai e, và modulus n.

1. *octet-string-to-integer conversion*.
   (a) Bác bỏ S nếu độ dài bit của S không phải là bội của 8.
   (b) Chuyển S thành số nguyên s như trong bước 4 của tiến trình ký.
   (c) Không chấp nhận chữ ký nếu s > n.
2. *RSA computation*. Tính $m = s^e \bmod n$.
3. *integer-to-octet-string*. Chuyển m thành một octet string EB với độ dài là k octets, như bước 6 của tiến trình ký.
4. *parsing*. Phân tích EB thành khối kiểu BT, padding string PS, dữ liệu D.
   (a) Không chấp nhận chữ ký nếu EB không được phân tích không mập mờ.
   (b) Không chấp nhận chữ ký nếu BT không là 00 hoặc 01.
   (c) Không chấp nhận chữ ký nếu PS có số octet nhỏ hơn 8 octets hoặc mâu thuẫn với BT.
5. *data decoding*.
   (a) Giải mã BER dữ liệu D để lấy message digest MD và một định danh thuật toán hash.
   (b) Không chấp nhận chữ ký nếu định danh thuật toán hash không xác định là MD2 hoặc MD5.
6. *message digest and comparion*.
   (a) Thực hiện hash message M với thuật toán message-digest đã chọn để nhận được MD'.
   (b) Chấp nhận chữ ký S trên M nếu và chỉ nếu MD' = MD.

Chương III
Module thực hiện ký và kiểm tra chữ ký số sử dụng chứng chỉ số

Trong chương này chúng tôi giới thiệu hai phần chính, phần thứ nhất giới thiệu về một số chuẩn được sử dụng trong quá trình sinh chữ ký và kiểm tra chữ ký, phần thứ hai giới thiệu về các thư viện cung cấp các hàm thực hiện ký và kiểm tra chữ ký trên một tệp dữ liệu (trong đó có cả ví dụ về việc xây dựng hai module chương trình ký/kiểm tra chữ ký sử dụng các thư viện đã nêu). Chúng ta có thể hình dung các bước snhư sau: khoá bí mật RSA được sinh theo PKCS#1, lưu dưới định dạng PKCS#8, khoá bí mật và chứng chỉ số được sử dụng để thực hiện ký một tệp dữ liệu, kết quả được lưu dưới định dạng PKCS#7.

1-Một số chuẩn mật mã khoá công khai

1.1-Giới thiệu về PKCS#1: Chuẩn mã hoá RSA

Chuẩn PKCS#1 mô tả một phương pháp mã hoá dữ liệu sử dụng hệ thống mật mã khoá công khai RSA. Chuẩn này bao gồm các phần sau: key generation, key syntax, encryption process, decryption process, signature algorithms.

1.1.1- Sinh khoá

Một thực thể (ví dụ, user) sẽ sinh bộ tham số RSA (n, e, d), với n = pq (p, q là 2 số nguyên tố lớn) được gọi là modulus, de-1 chia hết cho cả (p-1) và (q-1), e được gọi là public exponent và d được gọi là private exponent. Gọi k là độ dài octet của n thì nó phải thoả mãn điều kiện: $2^{8(k-1)} \le n \le 2^{8k}$, như vậy k_{min} = 12 octets thì mới thoả mãn chuẩn PKCS#1.

Chúng ta có một số nhận xét sau:

- public exponent có thể được chuẩn hoá trong một số ứng dụng: e lấy giá trị 3 và F_4 (65537) đã được chú ý trong X.509 phụ lục C.
- Một số điều kiện để chọn số nguyên tố là probable prime, nên các điều kiện an toàn không nằm trong chuẩn này. Như vậy chúng ta phải xem xét lại cách sinh số nguyên tố (provable prime).

1.1.2- Cú pháp khoá

Public-key syntax theo kiểu ASN.1 (Abstract Syntax Notation One) như sau:

```
RSAPublicKey ::= SEQUENCE {
        modulus INTEGER, -- n
        publicExponent INTEGER -- e
}
```

Private-key syntax theo kiểu ASN.1 như sau:

```
RSAPublicKey ::= SEQUENCE {
        version Version,
```

modulus INTEGER, -- n
publicExponent INTEGER, -- e
privateExponent INTEGER, -- d
prime1 INTEGER, -- p
prime2 INTEGER, -- q
exponent1 INTEGER, -- d mod (p-1)
exponent2 INTEGER, -- d mod (q-1)
coefficient INTEGER -- (inverse of q) mod p
}
Version ::= INTEGER

1.1.3- Tiến trình mã hoá

Tiến trình mã hoá bao gồm 4 bước: định dạng encrytion-block, chuyển đổi octet-string-to-integer, phép tính RSA, và chuyển đổi integer-to-octet-string (để thêm thông tin tham khảo tài liệu PKCS#1). Gọi dữ liệu cần mã hoá là một octet string D, modulus n, public-key e, private key d. Kết thúc tiến trình mã hoá chúng ta nhận được encrypted data là một octet string ED. Length(D) phải lớn hơn k-11 octets (với k lớn hơn bằng 12 octets), chuỗi padding PS tối thiểu là 8 octets (đây là một trong những điều kiện bảo mật của chuẩn PKCS#1). Chúng ta có một số chú ý sau:

- Trong một số ứng dụng của chuẩn này để mã hoá content-encryption keys và message digests thì length(D) ≤ 30, do vậy RSA modulus phải tối thiểu là 328 bits (41 octets) (là một trong những điều kiện an toàn).
- Tiến trình mã hoá không cung cấp kiểm tra tính toàn vẹn của dữ liệu đã được mã hoá có thể bị thay đổi khi lưu chuyển. Tuy nhiên, cấu trúc của khối mã hoá bảo đảm rằng xác suất mà sự thay đổi đó không được nhận ra là nhỏ hơn 2^{-16}.

1.1.4- Tiến trình giải mã

Tiến trình giải mã bao gồm 4 bước (ngược lại tiến trình mã hoá): chuyển đổi octet-string-to-integer, tính toán RSA, chuyển đổi integer-to-octet-string, và phân tích encryption-block. Để có thêm thông tin mời các bạn tham khảo tài liệu PKCS#1.

1.1.5- Các thuật toán chữ ký số

Các thuật toán chữ ký số trong PKCS#1 được dựa trên tiến trình mã hoá và giả mã RSA (ở trên). Tiến trình ký bao gồm 4 bước: message digesting, data encoding, RSA encryption, và octet-strintg-to-bit-string conversion. Sự khác nhau giữa tiến trình ký trong PKCS#1 và PKCS#7 trong phần encrypted message digest đó là việc thể hiện bit string đối với PKCS#1 và thể hiện octet string trong PKCS#7. Do vậy, trong mục này chúng tôi sẽ không trình bày mà sẽ chi tiết trong phần PKCS#7 (mục 3).

Nhận xét:

- Với PKCS#1 v2.0 thì phần RSA encryption/decryption được hỗ trợ thêm kiểu RSAES-OAEP (RSA Encryption Scheme - Optimal Asymmetric Encryption Scheme). Lược đồ mã hoá này được xem như là an toàn chứng minh được.
- Với PKCS#1 v2.1 phần RSA signature/verification được hỗ trợ thêm kiểu RSASSA-PSS (RSA Signature Scheme with Appendix - Probabilistic Signature Scheme). Lược đồ chữ ký này được xem là an toàn chứng minh được, độ khó giả mạo chữ ký tương đương với việc tính hàm ngược trong thuật toán RSA. Lược đồ đưa thêm các hàm sinh hash và mask được xem như là các black boxes hoặc random oracles. Nhưng trên lý thuyết thì một kẻ tấn công cũng có thể thay thế một hàm hash khác hàm hash mà signer đã chọn, nên gợi ý rằng hàm sinh mask của EMSA-PSS (hàm encoding của thuật toán RSASSA-PSS) giống hàm hash. Như vậy, theo kiểu này thì tất cả các encoded message sẽ phụ thuộc vào hàm hash và nó sẽ khó khăn cho đối phương thực hiện việc thay thế hàm hash khác.

Để kết thúc phần giới thiệu về PKCS#1 chúng tôi đưa ra một số thông tin về RSA encryption/decryption và RSA signature/verification để các bạn tiện tham khảo.

- Version 1.5: RSA encryption/decryption theo kiểu RSAES-PKCS1-V1_5, RSA signature/verification theo kiểu RSASSA-PCKS1-V1_5.
- Version 2.0: RSA encryption/decryption được hỗ trợ thêm kiểu RSAES-OAEP (RSA Encryption Scheme - Optimal Asymmetric Encryption Scheme).

Version 2.1: RSA signature/verification được hỗ trợ thêm kiểu RSASSA-PSS (RSA Signature Scheme with Appendix - Probabilistic Signature Scheme).

1.2-Giới thiệu về định dạng PKCS#7

Chuẩn PKCS#7 mô tả một định dạng tổng quát cho dữ liệu mà có thể có phần mật mã áp dụng vào đó, cụ thể là: các chữ ký số và các bọc số (digital envelope, trước tiên nội dung được mã hoá bởi khoá content-encryption key với thuật toán content-encryption algorithm, và content-encryption key được mã hoá bởi RSA public key của người nhận, sau đó kết hợp nội dung đã mã hoá và khoá đã mã hoá theo chuẩn PKCS#7). Định dạng này cho phép sự đệ quy, ví dụ, có thể ký lên dữ liệu số đã bị bọc trước đó. Chuẩn này cho phép có thể có các thuộc tính tuỳ ý như thời gian ký, được xác thực đi cùng với nội dung của một message. Trong trường hợp đơn giản của định dạng là việc phát hành các certificates và các CRLs.

Chuẩn này có thể hỗ trợ nhiều kiến trúc quản lý khoá dựa trên certificate (ví dụ, dạng PEM trong RFC 1422). Sự quyết định của kiến trúc như là: người phát hành certificate nào được xem như là top-level, các thực thể certificate nào được uỷ quyền chứng thực, distinguished name nào được chấp nhận, và người phát hành certificate dựa vào các policy nào.

Trong chuẩn này đưa ra cú pháp tổng quát cho 6 kiểu nội dung sau: data, signed data, enveloped data, signed-and-enveloped data, digested data, và encrypted data. Trước khi mô tả 6 kiểu nội dung này chúng tôi đưa ra cú pháp tổng quát (nó sẽ thay đổi phụ thuộc vào kiểu nội dung dữ liệu) của chuẩn PKCS#7 như sau:

ContentInfo ::= SEQUENCE {
contentType ContenType,
content [0] EXPLICIT ANY DEFINED BY contenType OPTIONAL
}
ContentType ::= OBJECT IDENTIFIER

Giải thích:
contentType xác định kiểu của nội dung dữ liệu (6 kiểu). Nó là một định danh object, tức là nó là một chuỗi số nguyên duy nhất để biểu diễn kiểu nội dung dữ liệu.
content nội dung dữ liệu.

Dưới đây chúng tôi sẽ đi mô tả chi tiết 6 kiểu nội dung dữ liệu.

1.2.1- Data content type
Data content type là một octet string, định nghĩa theo ASN.1 kiểu Data đơn giản như sau:

Data ::= OCTET STRING

1.2.2- Signed-data content type
Signed-data conten type bao gồm nội dung của kiểu nào đó và tương ứng message digest đã được mã hoá không cho ai hoặc cho nhiều người ký. Chúng ta hiểu message digest được mã hoá cho một signer là một chữ ký trên nội dung của signer đó. Kiểu của nội dung có thể được ký bởi nhiều người lên đó (parallel). Hơn nữa, trong trường hợp đơn giản nhất tức là không có signer nào ký trên nội dung đó, nghĩa là phát hành các certificates và các CRLs.

Điều này nói lên rằng với ứng dụng riêng biệt sử dụng signed-data content type với một chữ ký số của một signer lên nội dung của kiểu nội dung dữ liệu. Với ứng dụng riêng biệt khác sử dụng trường hợp suy biến để phát hành các certificates và các CRLs.

Quá trình ký dữ liệu được xây dựng theo các bước sau:

- Tạo message digest. Với mỗi signer thì message digest được thực hiện trên nội dung với một thuật toán băm xác định. Nếu signer cần xác thực một số thông tin khác thì các thông tin này cũng được băm bởi thuật toán trên.
- Message digest được mã bằng private key của signer.

- Encrypted message digest được đặt vào giá trị SignerInfo. Certificates và CRLs cho từng signer, hoặc không thuộc signer nào cũng được đưa vào SignerInfo.
- Đưa tất cả các thuật toán message-digest algorithm của các signer và các giá trị SignerInfo tương ứng vào SigneData.

Người nhận sẽ kiểm tra chữ ký bằng cách giải mã encrypted message digest của từng signer với public key của signer tương ứng, sau đó so sánh với message digest tìm được với message digest đã tính độc lập.

Kiểu nội dung dữ liệu được ký được định nghĩa trong ASN.1 là SignedData như sau:

```
SignedData ::= SEQUENCE {
        version Version,
        digestAlgorithms DigestAlgorithmIdentifiers,
        contentInfo ContenInfo,
        certificates [0] IMPLICIT ExtendedCertifictesAndCertificates OPTIONAL,
        crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
        signerInfos SignerInfos
}
DigestAlgorithmIdentifiers ::= SER OF DigestAlgorithmIdentifier
SignerInfos ::= SET OF SignerInfo
```

Kiểu thông tin signer được định nghĩa trong ASN.1 là SignerInfo như sau:

```
SignerInfo ::= SEQUENCE {
        version Version,
        issuerAndSerialNumber IssuerAndSerialNumber,
        digestAlgorithm DigestAlgorithmIdentifier,
        authenticateAttributes [0] IMPLICIT Attributes OPTIONAL,
        digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,
        encryptedDigest EncryptedDigest,
        unauthenticateAttributes [1] IMPLICIT Attributes OPTIONAL
}
EncryptedDigest ::= OCTET STRING
```

Quá trình tạo message digest và mã hoá message digest được chi tiết trong mục 9.3 và 9.4 trong tài liệu PKCS#7. Dưới đây chúng tôi chỉ đưa ra một số chú ý của các quá trình này.

- Sự khác nhau giữa quá trình ký trong chuẩn PKCS#7 và PKCS#1 là chữ ký trong chuẩn PKCS#1 được biểu diễn ở dạng bit string (có thể sử dụng X.509 macro SIGNED).

- Định danh thuật toán băm trong DigestInfo thay vì thoả thuận một thuật toán băm sẽ giảm được mối nguy hiểm. Ví dụ, kẻ tấn công có thể tìm ra được message với message digest nhận được. Hắn sẽ giả mạo chữ ký bằng cách lấy một message với cùng thuật toán băm mà đã được ký trước đó, và biểu diễn chữ ký trước đó lên message mới.

1.2.3- Enveloped-data content type

Envelope-data content type bao gồm nội dung đã được mã (một số kiểu) và các khoá mã nội dung được mã (encrypted content-encryption keys) cho một hoặc nhiều người nhận. Sự kết hợp nội dung đã được mã và khoá mã nội dung đã được mã cho một người nhận được gọi là digital envelope cho người đó. Envelope-data content type được áp dụng để biểu diễn digital envelope của một hoặc nhiều người trên nội dung của dữ liệu, digested-data, hoặc signed-data content type. Quá trình dữ liệu được bọc theo các bước sau:

- Sinh ngẫu nhiên khoá content-encryption key (với một thuật toán content-encryption algorithm).
- Với từng người nhận, conten-encryption key được mã hoá bởi public key tương ứng (kết quả là encrypted conten-encryption key).
- Với từng người nhận, encrypted conten-encryption key và một số thông tin khác được đưa vào giá trị RecipientInfo.
- Nội dung được mã hoá sử dụng conten-encryption key, thu được encrypted content.
- Tập hợp các giá trị RecipientInfo với encrypted content vào giá trị EnvelopedData.

Người nhận mở vỏ bọc số bằng cách giải mã một trong các encrypted content-encryption keys với private key của mình, kết quả nhận được conten-encryption key và giải mã encrypted content với conten-encryption key đã tìm được. Private key của người nhận được tham chiếu bởi một tên phát hành và một số serial duy nhất định danh certificate cho public key tương ứng.

Envelope-data content type định nghĩa trong ASN.1 có kiểu EnvelopedData như sau:

```
EnvelopedData ::= SEQUENCE {
        version Version,
        recipientInfos RecipientInfos,
        encryptedContentInfo EncryptedContentInfo
}
RecipientInfos ::= SET OF RecipientInfo
EncryptedContentInfo ::= SEQUENCE {
        contentType ContentType,
        contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
        encryptedContent [0] IMPLICIT Encrypted Content OPTIONAL
```

}
EncryptedContent ::= OCTET STRING

RecipientInfo type được định nghĩa trong ASN.1 là kiểu RecipientInfo như sau:

RecipientInfo ::= SEQUENCE {
version Version,
issuerAndSerialNumber IssuerAndSerialNumber,
keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
encryptedKey EncryptedKey
}
EncryptedKey ::= OCTET STRING

Quá trình mã hoá khoá và mã hoá nội dung đã được chi tiết trong mục 10.3 của chuẩn PKCS#7. Chúng tôi chỉ đưa ra chú ý sau: Một số thuật toán content-encryption algorithm làm việc với độ dài đầu vào là bội của k octets, với k > 1, và ứng dụng đưa ra một phương pháp đơn giản điều khiển độ dài đầu vào khi mà độ dài không là bội của k. Với phương này sẽ thực hiện pad thêm vào đầu vào bằng kéo dài phần cuối với k - (l mod k) octets với tất cả các giá trị là k - (l mod k), với l là độ dài của đầu vào. Như vậy, phương pháp này chỉ tốt khi k < 256 (< 2048 bits), với k lớn hơn vẫn là một bài toán mở (PKCS#7 v1.5). Trên windows sử dụng tools để update giúp cho việc import certificate với khoá có độ dài lớn hơn 2048, phải chăng đã giải quyết được vấn đề này.

1.2.4- Signed-and-enveloped-data content type
Signed-and-enveloped-data content type bao gồm encrypted content, encrypted content-encryption key cho một hoặc nhiều người và cặp encrypted message digests (một bản được mã bởi private key của signer và một bản mã content-encryption key) cho một hoặc nhiều signers. Như vậy, định dạng này là sự kết hợp của 2 phần: Signed-data content type và enveloped-data content type. Quá trình xây dựng dữ liệu signed-and-enveloped-data content type theo các bước sau:

- Khoá content-encryption key (cho một thuật toán content-encryption algorithm) được sinh ngẫu nhiên.
- Với từng người nhận, content-encryption key được mã hoá với public key tương ứng.
- Với từng người nhận, encrypted content-encryption key và một số thông tin về người nhận cùng được đưa vào giá trị RecipientInfo.
- Với từng signer, message digest được thực hiện trên nội dung với thuật toán băm đã chỉ ra bởi signer.
- Với từng signer, message digest và một số thông tin khác được mã hoá với private key của signer, và kết quả thu được encrypted content-encryption key.
- Với từng signer, cặp encrypted message digest và một số thông tin khác cùng được đưa vào giá trị SignerInfo.

- Nội dung được mã với content-encryption key, nhận được encrypted content.
- Hợp tất cả các thuật toán băm, các giá trị SignerInfo, và các giá trị RecipientInfo cùng với encrypted content vào giá trị SignedAndEnvelopedData.

Khi người nhận muốn mở vỏ bọc số và kiểm tra các chữ ký thì thực hiện lần lượt như mục 3.4 và 3.3, chúng tôi sẽ không nhắc lại ở đây và chỉ chú ý rằng: phương pháp này không có các thuộc tính xác thực nên nó được ứng dụng để bọc thông tin về người ký hơn là việc thực hiện ký.

Signed-and-enveloped-data content type định nghĩa trong ASN.1 là kiểu SignedAndEnvelopedData như sau:

```
SignedAndEnvelopedData ::= {
  version Version,
  recipientInfos RecipientInfos,
  digestAlgoritms DigestAlgorithmIdenfifiers,
  encryptedContentInfo EncryptedContentInfo,
  certificates [0]  IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
  crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos
}
```

Quá trình digest-encryption được thực hiện như mục 3.2 nhưng có điểm khác đó là: quá trình này thực hiện 2 bước. Bước thứ nhất thực hiện như mục 3.2. Bước thứ 2 lấy kết quả của bước 1 nhưng không mã dạng DER. Mục đích của việc này là chống việc một kẻ tấn công khôi phục được message digest của nội dung. Nếu việc này thực hiện được thì kẻ đó sẽ xác định được danh sách các ứng viên nội dung và sau đó xác định được nội dung thực sự bằng các so sánh các message digest của họ với message digest đúng.

1.2.5- Digested-data content type

Digested-data content type bao gồm nội dung và một message digest của nội dung đó. Công việc này được yêu cầu để đảm bảo tính toàn vẹn của nội dung dữ liệu. Quá trình thực hiện băm dữ liệu theo các bước sau:

- Một message digest được thực hiện trên nội dung với một thuật toán message digest algorithm.
- Message digest algorithm và message digest cùng được đưa vào giá trị DigestedData.

Như đã đề cập ở trên, người nhận kiểm tra message digest bằng cách so sánh message digest này với một message digest được tính độc lập trên nội dung.

Digested-data content type được định nghĩa trong ASN.1 là kiểu DigestedData như sau:

```
DigestedData ::= SEQUENCE {
        version Version,
        digestAlgorithm DigestAlgorithmIdentifier,
        contentInfo ContentInfo,
        digest Digest
}
Digest ::= OCTET STRING
```

1.2.6- Encrypted-data content type

Encrypted-data content type gồm có nội dung đã được mã hoá encrypted content. Kiểu này được định nghĩa trong ASN.1 là kiểu EncryptedData như sau:

```
EncryptedData ::= SEQUENCE {
        version Version,
        encryptedContentInfo EncryptedContentInfo
}
```

1.3- PKCS#8: Private-Key information Syntax Standard

Chuẩn PKCS#8 mô tả một cú pháp về thông tin của private-key. Thông tin này bao gồm một private key cho thuật toán khoá công khai và tập hợp các thuộc tính của nó. Đồng thời chuẩn này cũng mô tả cú pháp của các private key đã được mã hoá. Thuật toán mã hoá dựa vào password (đã được mô tả trong PKCS#5) cũng được sử dụng để mã hoá thông tin về private key.

Chuẩn này đưa ra tập hợp các thuộc tính là nhằm mục đích cung cấp một cách đơn giản cho một user thiết lập tin cậy về thông tin: distinguished nam hoặc public key của top-level CA. Trong khi đó sự tin cậy này có thể được thiết lập với một chữ ký số, mã hoá với một secret key (user đã biết). Trong chuẩn này chúng tôi trình bày 2 mục: private-key information syntax (PrivateKeyInfo) và encrypted private-key information syntax (EncryptedPrivateKeyInfo).

1.3.1- Private-key information syntax

Cú pháp về thông tin private key có kiểu PrivateKeyInfo (ASN.1) như sau:

```
PrivateKeyInfo ::= SEQUENCE {
        version Version,
        privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
        privateKey PrivateKey,
        attributes [0] IMPLICIT Attributes OPTIONAL
}
Version ::= INTEGER
```

PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
PrivateKey ::= OCTET STRING
Attributes ::= SET OF Attribute

Giải thích:
version số phiên bản của cú pháp (để tương thích với lần sửa tiếp theo), trong chuẩn này thì số version là 0.
privateKeyAlgorithm xác định thuật toán sinh ra private key (ví dụ, rsaEncryption).
privateKey một octet string mà nội dung của nó là giá trị của private key.
attributes tập các thuộc tính, chúng là các thông tin mở rộng đã được mã hoá đi cùng với thông tin private key.

1.3.2- Encrypted private-key information syntax
Cú pháp về thông tin private key có kiểu EncryptedPrivateKeyInfo (ASN.1) như sau:

EncryptedPrivateKeyInfo ::= SEQUENCE {
encryptionAlgorithm EncryptionAlgorithmIdentifier,
encryptedData EncryptedData
}
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING

Giải thích:
encryptionAlgorithm xác định thuật toán mà thông tin về private key được mã hoá (ví dụ, pbeWithMD5AndDES-CBC).
encryptedData dữ liệu private key đã được mã hoá.

Quá trình mã hoá theo 2 bước:

- Thông tin private key được mã (encode) dạng BER, sau bước này thu được một octet string.
- Octet string (thu được của bước 1) được mã hoá với secret key, sau bước này thu được một octet string là kết quả của quá trình mã hoá.

2-Module thực hiện việc ký/kiểm tra chữ ký
2.1-Module thực hiện ký một tệp dữ liệu sử dụng chứng chỉ số
2.1.1-Các thư viện cung cấp các hàm thực hiện việc ký
Để xây dựng module thực hiện ký một tệp dữ liệu chúng ta chỉ cần sử dụng hai thư viện libcrypto.a, sign.o và một tệp C header là sign.h. Trong đó thư viện libcrypto.a cung cấp các hàm tác vụ cho việc thực hiện xây dựng hàm thực hiện ký một tệp dữ liệu trong thư viện sign.o.
Dưới đây là một số cấu trúc dữ liệu và các hàm chính được cung cấp trong hai thư viện trên:

Thư viện libcrypto.a:

-Cấu trúc lưu chứng chỉ số

```
X509 *x509;
```

-Cấu trúc lưu khoá bí mật

```
EVP_PKEY *pkey;
```

-Cấu trúc lưu đối tượng dữ liệu PKCS#7

```
EVP_PKEY *pkey;
```

-Cấu trúc lưu các thông tin liên quan đến chữ ký số (thời gian, đối tượng ký, ...)

```
EVP_PKEY *pkey;
```

-Hàm đọc chứng chỉ số từ một file dạng PEM

```
if ((x509=PEM_read_bio_X509(incert,NULL,NULL,NULL)) == NULL) goto
err;
```

-Hàm đọc khoá bí mật số từ một file dạng PEM

```
if ((pkey=PEM_read_bio_PrivateKey(inkey,NULL,NULL,NULL)) == NULL)
goto err;
```

-Hàm khởi tạo một đối tượng PKCS7 mới

```
p7=PKCS7_new();
```

-Hàm bổ sung chứng chỉ số cho một đối tượng PKCS7

```
PKCS7_add_certificate(p7,x509);
```

-Hàm bổ sung dữ liệu cần ký cho một đối tượng PKCS7

```
PKCS7_content_new(p7,NID_pkcs7_data);
```

-Hàm thực hiện ký đối với một đối tượng PKCS7

```
if (!PKCS7_dataFinal(p7,p7bio)) goto err;
```

-Hàm thực hiện ghi kết quả ra một tệp

```
PEM_write_PKCS7(fb,p7);
```

Thư viện sign.o:

Thư viện này chỉ cung cấp một hàm duy nhất từ việc sử dụng các hàm trong thư viện libcrypto.a:

```
int signdata(char *infile,char *outfile, char *certfile, char *keyfile)
```

Trong đó

infile: đường dẫn và tên tệp cần ký

outfile: đường dẫn và tên tệp lưu kết quả chữ ký số.

certfile: đường dẫn và tên tệp chứng chỉ số dưới dạng PEM

keyfile: đường dẫn và tên tệp khoá bí mật dưới đinh dạng PEM

Giá trị trả về: 1 khi quá trình ký thành công

0 khi quá trình ký gặp lỗi

Tệp sign.h

Khai báo hàm signdata() như sau:

```
#include <stdio.h>
#include <string.h>
int signdata(char *infile,char *outfile, char *certfile, char
*keyfile);
```

Các thư viện libcrrypto.a, sign.o và sign.h được lưu trong thư mục Lib\libsign trên CD-ROM.

2.1.2-Chương trình ví dụ thực hiện việc ký một tệp dữ liệu

Để mô tả cụ thể hơn việc xây dựng chương trình thực hiện gọi hàm ký dữ liệu sử dụng chứng chỉ số và khóa bí mật, chúng tôi xây dựng một module chương trình ví dụ thực hiện ký một tệp dữ liệu. Module chương trình ví dụ được lưu trong thư mục Example\sign, nội dung thư mục này gồm các tệp dưới đây:

- Tệp \Lib\libcrypto.a: thư viện cung cấp các dịch vụ mật mã khóa công khai.
- Tệp \Lib\sign.o: thư viện cung cấp hàm ký tệp dữ liệu.
- Tệp \sign.h:
- Tệp \data.txt: ví dụ về tệp dữ liệu sẽ được ký.
- Tệp \signdata.c: tệp chương trình thực hiện tác vụ ký
- Tệp \user1.crt: tệp chứng chỉ số được sử dụng khi ký.
- Tệp \user1.key: tệp khóa bí mật tương ứng với tệp chứng chỉ số user1.crt.
- Tệp \chuky.p7: tệp kết quả dưới định dạng PKCS#7
- Tệp \Makefile: sử dụng để biên dịch chương trình ví dụ
- Tệp \Readme: hướng dẫn biên dịch và sử dụng chương trình.

Dưới đây là toàn bộ nội dung tệp chương trình nguồn signdata.c

```
/*Sử dụng các tệp header*/
#include <stdio.h>
#include <string.h>
#include "sign.h"
int main(int argc, char **argv)
{

/*Khai báo các biến nhận tham số đầu vào*/
int i;
char *infile=NULL;
char *outfile=NULL;
char *certfile=NULL;
char *keyfile=NULL;

/*Nhận các tham số đầu vào*/
      argv++;
      argc--;
      for (;;)
            {
            if (argc <= 0) break;
            if (strcmp(*argv,"-in") == 0)
                  {
                  if (--argc < 1) goto bad;
                  infile= *(++argv);
                  }
            else if (strcmp(*argv,"-out") == 0)
                  {
                  if (--argc < 1) goto bad;
                  outfile= *(++argv);
                  }
            else if (strcmp(*argv,"-cert") == 0)
                  {
```

```
                  if (--argc < 1) goto bad;
                  certfile= *(++argv);
                  }

            else if (strcmp(*argv,"-key") == 0)
                  {
                  if (--argc < 1) goto bad;
                  keyfile= *(++argv);
                  }
            else goto bad;
            argv++;
            argc--;
            }

/*Kiểm tra các tham số đầu vào*/
      if ((infile==NULL) || (outfile==NULL) || (keyfile==NULL) ||
(certfile==NULL))
            {
bad:
            printf("Su dung: sign [args] \n");
            printf(" -in            Ten tep du lieu can ky\n");
            printf(" -out           Ten dau ra duoi dang PKCS7\n");
            printf(" -cert          Ten tep chung chi (PEM)\n");
            printf(" -key           Ten tep khoa (PEM)\n");
            goto err;
            }

/*Gọi hàm thực hiện việc ký lên tệp dữ liệu*/
    i=signdata(infile,outfile,certfile,keyfile);

/*Quá trình ký không thành công*/
    if (i==0){
    printf ("Qua trinh ky tep %s khong thanh cong!\n",infile);
    }

/*Quá trình ký thành công*/
    else
    {
    printf ("Qua trinh ky tep %s thanh cong!\n",infile);
    }
    return 1;
err:
printf("Wrong args command");
}
```

Để biên dịch module chương trình chúng ta thực hiện lệnh sau:

make

khi đó tiện ích signdata sẽ được sinh, để sử dụng tiện ích chúng ta thực hiện lệnh:

./signdata -in data.txt -out chuky.p7 -cert user1.crt -key user1.key

Nếu quá trình ký thành công kết quả cho ta tệp chuky.p7k, nội dung của tệp này như dưới đây:

```
-----BEGIN PKCS7-----
MIIEOgYJKoZIhvcNAQcCoIIEKzCCBCcCAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3
```

```
DQEHAaCCAo8wggKLMIIB9KADAgECAgIBFzANBgkqhkiG9w0BAQUFADBoMR8wHQYJ
KoZIhvcNAQkBFhBSb290Q0FAeWFob28uY29tMQ8wDQYDVQQDEwZSb290Q0ExEjAQ
BgNVBAsTCU15Q0EgVXNlcjETMBEGA1UEChMKTXlDQSBHcm91cDELMAkGA1UEBhMC
Vk4wHhcNMDQwMTAxMjM1OTM3WhcNMDQwNDEwMjM1OTM3WjBaMR0wGwYJKoZIhvcN
AQkBFg51c2VyMUBwdmtoLmNvbTEOMAwGA1UEAxMFdXNlcjExDTALBgNVBAsTBHB2
a2gxDTALBgNVBAoTBG1hbmcxCzAJBgNVBAYTAnRhMIGeMA0GCSqGSIb3DQEBAQUA
A4GMADCBiAKBgEAACAAB4AAaAAKgAq6hBR5GYpeFA7mRDb0EyzpTa7TAx1GqXs76
zwsvi8U2SJ/foi+CKJ7EfdyRslXiX5VPiT+3G/3XsCN0vVH2Rjb7+zNye3ar4NXq
BiPYtNqnk8yTSMikkC2FOEWLCVQOg6WNfBPe5fblJo1/83fYi66lybWZAgMBAAGj
UzBRMAkGA1UdEwQCMAAwEQYJYIZIAYb4QgEBBAQDAgWgMAsGA1UdDwQEAwIF4DAk
BglghkgBhvhCAQ0EFxYVTXlDQSBVc2VyIENlcnRpZmljYXRlMA0GCSqGSIb3DQEB
BQUAA4GBADjgQGRJFzZB8X3vfrQeP5AAax5cNf/Utp5Grona6XQkYvMyQCIBNieG
vJvN6PmINLlfb6JQRujE9Ovo6QBNtuLna11OcFDTxaA2xzkHrtNK3YvuTMR/ENfF
ZSuzxk7T4XtW4CE+vtD7Q0ttgMPzUpfCEC35j3cHEu/xgHAD0lYlMYIBczCCAW8C
AQEwbjBoMR8wHQYJKoZIhvcNAQkBFhBSb290Q0FAeWFob28uY29tMQ8wDQYDVQQD
EwZSb290Q0ExEjAQBgNVBAsTCU15Q0EgVXNlcjETMBEGA1UEChMKTXlDQSBHcm91
cDELMAkGA1UEBhMCVk4CAgEXMAkGBSsOAwIaBQCgXTAYBgkqhkiG9w0BCQMxCwYJ
KoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0wNDA0MTUxNzA1NDlaMCMGCSqGSIb3
DQEJBDEWBBQtHNJmIpAGMPkFbZwQPpjgPuDT5jANBgkqhkiG9w0BAQEFAASBgBq3
YjHdPZX7FjGLNSqQTs8LxWnlfTebP/tHovPrF80glilF3vPRhrb6xDeQ0ms8jdLu
4/Qex/icxRwELMzeQAyAazt3pFaScKuwMZ+VLR3NVwv9P2xgggOYQa5xm85C3dRX
NBLAlC9L9feca9mmNGSs1fIKweI015+/rJpp0S3o
-----END PKCS7-----
```

2.2-Module thực hiện việc kiểm tra chữ ký số

2.2.1-Các thư viện cung cấp các hàm thực hiện việc kiểm tra chữ ký

Để xây dựng module thực hiện kiểm tra chữ ký chúng ta chỉ cần sử dụng hai thư viện libcrypto.a, verify.o và một tệp C header là verify.h. Trong đó thư viện libcrypto.a cung cấp các hàm tác vụ cho việc thực hiện xây dựng hàm thực hiện kiểm tra chữ ký lên một tệp dữ liệu trong thư viện verify.o.

Dưới đây là một số cấu trúc dữ liệu và các hàm chính được cung cấp trong hai thư viện trên:

Thư viện libcrypto.a:

-Cấu trúc dữ liệu PKCS7

```
PKCS7 *p7;
```

-Cấu trúc dữ liệu lưu thông tin chữ ký số dưới định dạng PKCS7

```
PKCS7_SIGNER_INFO *si;
```

-Cấu trúc dữ liệu lưu chứng chỉ số theo chuẩn X509

```
X509_STORE_CTX cert_ctx;
```

-Các hàm thực hiện mở các tệp dữ liệu và tệp PKCS7

```
data=BIO_new(BIO_s_file());
detached=BIO_new(BIO_s_file());
```

-Các hàm thực hiện đọc các tệp dữ liệu và tệp PKCS7

```
if (!BIO_read_filename(detached,datafile)){
                    goto err;
               }
if (!BIO_read_filename(data,sigfile)){
                    goto err;
               }
```

-Hàm tạo đối tượng PKCS7 từ một buffer dữ liệu dạng PKCS7

```
if ((p7=PEM_read_bio_PKCS7(data,NULL,NULL,NULL)) == NULL) goto
err;
```

-Các hàm tạo đối tượng lưu chứngchỉ số của RootCA từ một từ một tệp dưới định dạng PEM

```
cert_store=X509_STORE_new();
X509_STORE_set_default_paths(cert_store);
X509_STORE_load_locations(cert_store,cafile,NULL);
```

-Hàm kiểm tra tính hợp lệ của chứng chỉ

```
X509_STORE_set_verify_cb_func(cert_store,verify_callback);
```

-Hàm kiểm tra chữ ký số trong đó có cả quá trình kiểm tra hiện trạng của chứng chỉ số được sử dụng để ký

```
i=PKCS7_dataVerify(cert_store,&cert_ctx,p7bio,p7,si);
```

Thư viện verify.o:

Thư viện này chỉ cung cấp một hàm duy nhất từ việc sử dụng các hàm trong thư viện libcrypto.a:

```
int verifydata(char *datafile,char *sigfile, char *cafile)
```

Trong đó:

datafile: đường dẫn và tên tệp dữ liệu đã được ký

sigfile: đường dẫn và tên tệp chữ ký dưới định dạng PKCS7

cafile: đường dẫn và tên tệp chứng chỉ số CA đưới định dạng PEM.

Giá trị trả về: hàm trả về giá trị 1 nếu kiểm tra chữ ký thành công

hàm trả về giá trị 0 nếu quá trình kiểm tra chữ ký không hợp lệ (chữ ký sai, chứng chỉ số CA hoặc chứng chỉ số đươc sử dụng để ký không hợp lệ)

Tệp verify.h:

Khai báo hàm verifydata như sau:

```
#include <stdio.h>
#include <string.h>
int verifydata(char *datafile,char *sigfile,char *cafile);
```

Cả ba tệp trên được lưu trong thư mục /Lib/verify trên CD-ROM.

2.2.2-Module chương trình ví dụ việc kiểm tra chữ ký

Để mô tả cụ thể hơn việc xây dựng chương trình thực hiện gọi hàm kiểm tra chữ ký sử dụng, chúng tôi xây dựng một module chương trình ví dụ thực hiện kiểm tra chữ ký lên một tệp dữ liệu. Module chương trình ví dụ được lưu trong thư mục Example\verify, nội dung thư mục này gồm các tệp dưới đây:

- Tệp \Lib\libcrypto.a: thư viện cung cấp các dịch vụ mật mã khóa công khai.
- Tệp \Lib\verify.o: thư viện cung cấp hàm kiểm tra chữ ký lên một tệp dữ liệu.
- Tệp \verify.h: tệp header khai báo hàm kiểm tra chữ ký
- Tệp \data.txt: ví dụ về tệp dữ liệu đã được ký.
- Tệp \verifydata.c: tệp chương trình thực hiện tác vụ kiểm tra chữ ký
- Tệp \RootCA.crt: tệp chứng chỉ số CA được sử dụng để kiểm tra chữ ký.
- Tệp \chuky.p7: tệp chữ ký dưới định dạng PKCS#7

- Tệp \Makefile: sử dụng để biên dịch chương trình ví dụ
- Tệp \Readme: hướng dẫn biên dịch và sử dụng chương trình.

Dưới đây là toàn bộ nội dung tệp chương trình verifydata.c

```
/*Sử dụng tệp verify.h*/
#include "verify.h"
int main(int argc, char **argv)
{
/*Khai báo các biến lưu đường dẫn và tên các tệp là tham số đầu vào*/
char *datafile=NULL;
char *sigfile=NULL;
char *cafile=NULL;
int i;

/*Nhận tham số đầu vào*/
      argv++;
      argc--;
      for (;;)
            {
            if (argc <= 0) break;
            if (strcmp(*argv,"-datafile") == 0)
                  {
                  if (--argc < 1) goto bad;
                  datafile= *(++argv);
                  }
            else if (strcmp(*argv,"-sigfile") == 0)
                  {
                  if (--argc < 1) goto bad;
                  sigfile= *(++argv);
                  }
            else if (strcmp(*argv,"-cafile") == 0)
                  {
                  if (--argc < 1) goto bad;
                  cafile= *(++argv);
                  }
            else goto bad;
            argv++;
            argc--;
            }

/*Kiểm tra các tham số đầu vào*/

if ((datafile==NULL) || (sigfile==NULL) || (cafile==NULL))
            {
bad:
            printf("Su dung: verify [args] \n");
            printf(" -datafile      Ten tep du lieu da duoc ky\n");
            printf(" -sigfile       Ten tep chu ky dang PKCS7\n");
            printf(" -cafile        Ten tep chung chi CA (PEM)\n");
            goto err;
            }

/*Gọi hàm kiểm tra chữ ký*/
i=verifydata(datafile,sigfile,cafile);
```

```
/*Kiểm tra chữ ký thành công*/
if (!i){
printf("\nVerify Failed!!!\n\n");
}

/*Chư ký không hợp lệ*/
else
{
printf("\nSignature OK!!!\n\n");
}
return 0;
err:
exit(0);
}
```

Để biên dịch module chương trình chúng ta thực hiện lệnh sau:

make

khi đó tiện ích verifydata sẽ được sinh, để sử dụng tiện ích chúng ta thực hiện lệnh:

./verifydata -datafile data.txt -sigfile chuky.p7 -cafile RootCA.crt