TCVN TIÊU CHUẨN QUỐC GIA

TCVN 7635 : 2007

Xuất bản lần 1

KỸ THUẬT MẬT MÃ - CHỮ KÝ SỐ

Cryptography technique - Digital signature

HÀ NỘI – 2007

Mục lục **Trang**

Lời nói đầu 2
Lời giới thiệu 5
1 Phạm vi áp dụng 7
2 Tài liệu viện dẫn 7
3 Thuật ngữ và định nghĩa, thuật ngữ viết tắt 7
3.1 Thuật ngữ và định nghĩa 7
3.1.1 Thông điệp dữ liệu *(data message)* 7
3.1.2 Chữ ký số (digital signature) 7
3.1.3 Hàm băm *(hash function)* 8
3.1.4 Quá trình tạo chữ ký số *(digital signature generation)* 8
3.1.5 Quá trình kiểm tra chữ ký số *(digital signature verification)* 8
3.1.6 Bộ tạo số giả ngẫu nhiên *(Pseudorandom number generator)* 8
3.2 Từ viết tắt 8
4 Khái quát 9
5 Thuật toán chữ ký số RSA-PSS 10
Các ký hiệu 10
5.2 Thuật toán RSA 11
5.2.1 Khoá công khai RSA 11
5.2.2 Khoá bí mật RSA 11
5.3 Các hàm cơ sở chuyển đổi dữ liệu 12
5.3.1 Hàm cơ sở chuyển đổi từ dạng số nguyên sang dạng chuỗi octet 12
5.3.2 Hàm cơ sở chuyển đổi từ dạng chuỗi octet về dạng số nguyên 12
5.4 Các phép toán mật mã cơ sở 12
5.4.1 Phép toán cơ sở RSASP 12
5.4.2 Phép toán cơ sở RSAVP 13
5.5 Lược đồ chữ ký RSA kèm phụ lục theo PSS 14
5.5.1 Thao tác tạo chữ ký 14
5.5.2 Thao tác kiểm tra chữ ký 15
5.6 Phương pháp định dạng cho chữ ký kèm phụ lục theo PSS (EMSA-PSS) 16
5.6.1 Thao tác định dạng 16
5.6.2 Thao tác kiểm tra 17
5.6.3 Hàm tạo mặt nạ MGF dựa vào hàm băm 18
6 Hàm băm SHA–256 19
6.1 Một số khái niệm và thuật ngữ 19
6.1.1 Biến, tham số 19
6.1.2 Các ký hiệu tính toán 20
6.1.3 Chuỗi các bit và các số nguyên 20
6.1.4 Các phép tính với các từ 20
6.2 Thuật toán 21
6.2.1 Các hàm và các hằng được sử dụng cho thuật toán 21
6.2.2 Bước tiền xử lý 21
6.2.3 Thuật toán tính giá trị băm 22
6.2.4 Dữ liệu kiểm tra 24
7 Bộ tạo số giả ngẫu nhiên dùng AES-128 24
7.1 Một số ký hiệu 24
7.2 Thuật toán 25
8 Tiêu chuẩn tham số sử dụng trong chữ ký số RSA-PSS 25
8.1 Các yêu cầu chung 25
8.2 Yêu cầu đối với các khoá RSA 26

Lời nói đầu

TCVN 7635 : 2007 do Tiểu Ban kỹ thuật tiêu chuẩn TCVN/JTC 1/SC 27 "*Các kỹ thuật mật mã*" biên soạn trên cơ sở dự thảo đề nghị của Ban cơ yếu Chính phủ, Tổng cục Tiêu chuẩn Đo lường Chất lượng đề nghị, Bộ Khoa học và Công nghệ công bố.

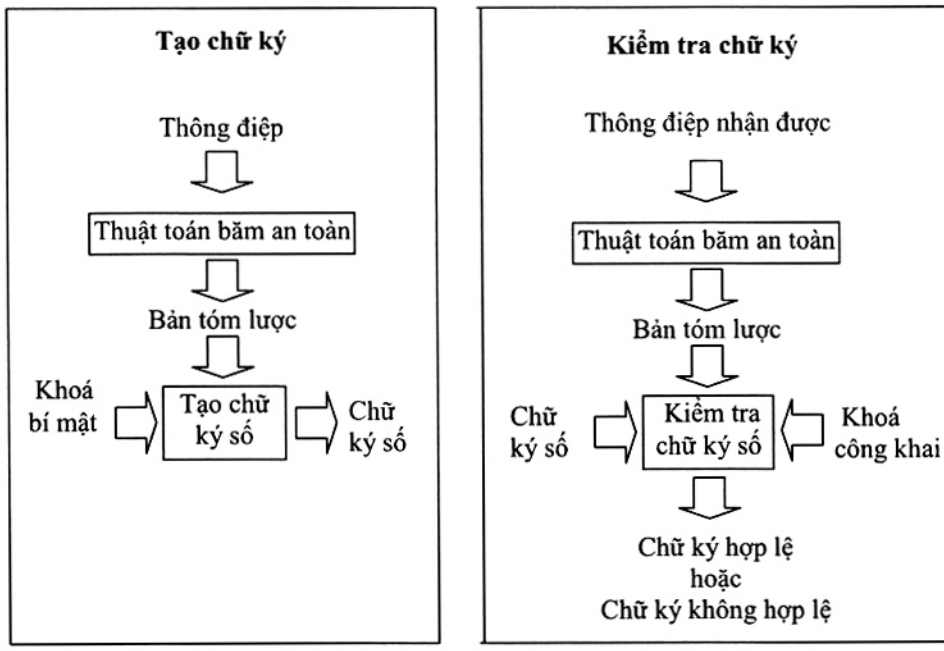
Lời giới thiệu

Trong giao dịch giấy tờ truyền thống chữ ký tay là phương tiện để xác thực nguồn gốc và nội dung của văn bản. Chữ ký tay còn có khả năng chống chối bỏ, nghĩa là người gửi sau khi đã ký vào văn bản thì không thể chối bỏ chữ ký của mình và văn bản sau khi được ký thì không thể thay đổi được nội dung. Đối với văn bản điện tử chữ ký tay không còn đảm bảo được các tính năng nói trên, vì vậy chữ ký số điện tử (gọi tắt là chữ ký số) được sử dụng để thay thế vai trò của chữ ký tay.

Chữ ký số được biểu diễn dưới dạng một chuỗi số nhị phân. Nó được tạo ra trên cơ sở sử dụng tập hợp các quy tắc và tập hợp các tham số để xác định danh tính người gửi (người ký) cũng như kiểm tra tính toàn vẹn của dữ liệu. Phương tiện cơ bản để thực hiện chữ ký số là kỹ thuật mật mã. Mỗi chữ ký số được thể hiện trên một lược đồ xác định gọi là lược đồ chữ ký số. Lược đồ này bao gồm ba thành tố: thuật toán chữ ký số, thuật toán hàm băm và thuật toán tạo số giả ngẫu nhiên. Thuật toán chữ ký số gồm thuật toán ký và thuật toán kiểm tra. Người gửi sử dụng thuật toán ký và khóa bí mật để tạo ra chữ ký số, người nhận (người kiểm tra) sử dụng thuật toán kiểm tra và khóa công khai tương ứng để kiểm tra đồng thời tính chân thực của thông điệp dữ liệu và tính chân thực của chữ ký số do người gửi tạo ra. Mỗi người sở hữu một cặp khóa bao gồm khoá công khai (giả thiết là được công bố một cách công khai) và khoá bí mật (được giữ bí mật tuyệt đối). Bất kỳ ai cũng có thể kiểm tra chữ ký số của một người nào đó bằng cách sử dụng khóa công khai của người này. Quá trình tạo chữ ký số chỉ có thể được thực hiện bởi người sở hữu khoá bí mật. Thuật toán hàm băm là biến đổi toán học dùng để thu gọn văn bản ban đầu (còn gọi là thông điệp dữ liệu) nhằm tạo ra bản tóm lược của thông điệp. Bản tóm lược này sẽ là đầu vào của thuật toán tạo chữ ký số. Chữ ký số được đính kèm với thông điệp dữ liệu đã được ký. Trong quá trình kiểm tra chữ ký số, thuật toán hàm băm cũng được áp dụng như trong quá trình tạo chữ ký. Thuật toán tạo số giả ngẫu nhiên dùng để tạo ra các số giả ngẫu nhiên (độc lập, đồng xác suất) làm tham số trong lược đồ chữ ký số (xem Hình 1). Tiêu chuẩn này quy định 3 thuật toán đó.

Thuật toán chữ ký số mô tả trong Tiêu chuẩn này được xây dựng dựa trên tài liệu: Tiêu chuẩn mật mã RSA: *PKCS#1 v2.1: RSA Cryptography Standard*, ban hành ngày 14 tháng 6 năm 2002.

Thuật toán hàm băm mô tả trong Tiêu chuẩn này được xây dựng dựa trên tài liệu: Tiêu chuẩn hàm băm an toàn (Mỹ): *FIPS 180-2: Secure Hash Standard*, ban hành ngày 1 tháng 8 năm 2001.



Hình 1 - Mô tả quá trình tạo và kiểm tra chữ ký số

Xuất bản lần 1

Kỹ thuật mật mã - Chữ ký số

Cryptography technique - Digital Signature

1 Phạm vi áp dụng

Tiêu chuẩn này áp dụng cho các chữ ký số sử dụng trong hoạt động giao dịch điện tử của mọi tổ chức, công dân Việt Nam và tổ chức, công dân nước ngoài có quan hệ kinh tế - xã hội với tổ chức, công dân Việt Nam .

2 Tài liệu viện dẫn

FIPS 197: Advanced Encryption Standard (Tiêu chuẩn Mã hóa tiên tiến của Mỹ), ban hành ngày 26 tháng 11 năm 2001.

3 Thuật ngữ và định nghĩa, thuật ngữ viết tắt

3.1 Thuật ngữ và định nghĩa

Tiêu chuẩn này sử dụng các thuật ngữ và định nghĩa sau đây:

3.1.1

Thông điệp dữ liệu *(data message)*

Dữ liệu sẽ được ký. Trong quá trình kiểm tra chữ ký thì đó là dữ liệu đã được ký.

Trong tiêu chuẩn này, thông điệp dữ liệu thường được gọi tắt là thông điệp.

3.1.2

Chữ ký số (digital signature)

Một chuỗi số, kết quả của phép biến đổi mật mã trên thông điệp dữ liệu nhằm cung cấp một phương tiện để kiểm tra tính xác thực của nguồn gốc thông điệp dữ liệu, tính toàn vẹn của dữ liệu và tính không thể chối bỏ của người đã ký.

3.1.3

Hàm băm (hash function)

Một thuật toán chuyển đổi mỗi thông điệp biểu diễn dưới dạng bit có độ dài bất kỳ thành một chuỗi bit có độ dài cố định. Chuỗi bit có độ dài cố định đó được gọi là "giá trị băm", "mã băm", hay đơn giản là "tóm lược" của thông điệp đầu vào. Các thuật toán băm mật mã được thiết kế sao cho thoả mãn các tính chất sau:

1. tính một chiều hay tính kháng tiền ảnh: Không thể tìm được trong thời gian cho phép một thông điệp có giá trị băm cho trước;
2. tính kháng tiền ảnh thứ hai: Cho trước thông điệp M1, không thể tìm được trong thời gian cho phép một thông điệp M2 khác M1 sao cho giá trị băm của M1 và M2 là như nhau;
3. tính kháng xung đột: Không thể tìm được hai chuỗi bit khác nhau có cùng một giá trị băm.

3.1.4

Quá trình tạo chữ ký số (digital signature generation)

Quá trình sử dụng thuật toán chữ ký số và khoá bí mật để tạo ra chữ ký số cho thông điệp.

3.1.5

Quá trình kiểm tra chữ ký số (digital signature verification)

Quá trình dùng thuật toán chữ ký số và khoá công khai để kiểm tra chữ ký số của thông điệp.

3.1.6

Bộ tạo số giả ngẫu nhiên (pseudorandom number generator)

Thiết bị hay thuật toán dùng để tạo ra một dãy các số giả ngẫu nhiên xuất hiện một cách độc lập thống kê và đồng xác suất.

3.2 Từ viết tắt

AES	Tiêu chuẩn mã hóa tiên tiến (*Advanced Encryption Standard*)
EM SA	Phương pháp định dạng sử dụng trong thuật toán chữ ký số kèm phụ lục (*Encoding Method for Signatures with Appendix*)
FIPS	Tiêu chuẩn xử lý thông tin Liên bang Mỹ (*Federal Information Processing Standard*) được ban hành bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Mỹ (*National Institute of Standard and Technology-NIST*)
GCD	Ước số chung lớn nhất (*Greatest Common Divisor*)
hexa	Biểu diễn theo hệ cơ số 16
I2OSP	Hàm cơ sở chuyển đổi từ dạng số nguyên sang chuỗi octet (*Integer-to-Octet-String Primitive*)

LCM	Bội số chung nhỏ nhất (*Least Common Multiplier*)
MGF	Hàm tạo mặt nạ (*Mask generation function*)
octet	Bộ 8 bit (còn có thể hiểu là một byte), được xem như một ký tự trong hệ đếm cơ số 256 biểu diễn dưới dạng một cặp chữ số của hệ đếm cơ số 16 (hexa)
OS2IP	Hàm cơ sở chuyển đổi từ chuỗi octet sang số nguyên (*Octet-String-to-Integer-Primitive*)
PKCS	Tiêu chuẩn mật mã khoá công khai (*Puplic Key Cryptography Standard*) do Phòng thí nghiệm RSA (Mỹ) ban hành
PSS	Lược đồ ký xác suất (*Probabilistic Signature Scheme*)
RSA	Tên của hệ mã do ba nhà toán học Rivest, Shamir và Adleman sáng tạo ra
RSAVP	Phép toán cơ sở phục vụ cho kiểm tra chữ ký RSA
RSASP	Phép toán ký RSA cơ sở
RSASSA	Lược đồ ký RSA kèm phụ lục (*RSA Signature Scheme with Appendix*)
SHA	Thuật toán băm an toàn (*Secure Hash Algorithm*)
Word	Từ (32 bit)

4 Khái quát

4.1 Tiêu chuẩn này quy định 3 thành phần cần thiết cho lược đồ chữ ký số. Thành phần thứ nhất là thuật toán chữ ký số RSA-PSS được mô tả trong Phần 5. Thành phần thứ 2 là thuật toán hàm băm SHA-256 được mô tả trong Phần 6 và thành phần cuối cùng là thuật toán tạo số giả ngẫu nhiên dùng AES-128 được mô tả trong Phần 7. Những tiêu chuẩn về các tham số sử dụng trong thuật toán chữ ký số RSA-PSS được đề cập tới trong Phần 8.

4.2 Chữ ký số là một dạng của chữ ký điện tử được sử dụng để xác minh người đã ký thông điệp và tính nguyên vẹn của nó.

4.3 Thuật toán chữ ký số là thuật toán cho phép người ký tạo ra được chữ ký số trên dữ liệu và cho phép người kiểm tra xác minh được tính xác thực của chữ ký. Mỗi người có một cặp khóa bao gồm khoá bí mật và khoá công khai. Khóa bí mật được sử dụng trong quá trình tạo chữ ký còn khóa công khai sử dụng trong quá trình kiểm tra chữ ký. Trong quá trình tạo và kiểm tra chữ ký, thông điệp M (dạng dữ liệu) được thu gọn nhờ áp dụng thuật toán băm an toàn (SHA-256). Một người không biết khóa bí mật của người đã ký thông điệp thì không thể tạo ra được chữ ký đó. Như vậy, chữ ký không thể bị giả mạo. Khi sử dụng khóa công khai của người ký, bất kỳ ai cũng có thể kiểm tra được tính chân thực của chữ ký số trên thông điệp đã được ký.

4.4 Khi khoá công khai được sử dụng để kiểm tra chữ ký không tương ứng với khoá bí mật đã được dùng trong việc tạo ra chữ ký thì không thể kết luận về tính chân thực của chữ ký đó.

Một cơ chế cần được thiết lập để gắn kết khoá công khai với danh tính của người giữ khoá bí mật tương ứng. Cơ chế này có thể đạt được nhờ sự chứng thực bởi bên thứ ba được tin cậy. Những nội dung về dịch vụ chứng thực không thuộc phạm vi của tiêu chuẩn này.

5 Thuật toán chữ ký số RSA-PSS

Các ký hiệu

≡	Ký hiệu phép đồng dư, ví dụ: a ≡ b (mod c) có nghĩa là a và b có cùng số dư khi chia cho c
c	Biểu diễn của bản mã dưới dạng một số nguyên trong khoảng từ 0 đến *n*-1
C	Bản mã ở dạng chuỗi Octet
d	Số mũ bí mật RSA
dP	Nghịch đảo của *e* theo môđun *p*-1 ($e \cdot dP \equiv 1 \bmod (p-1)$)
dQ	Nghịch đảo của *e* theo môđun *q*-1 ($e \cdot dQ \equiv 1 \bmod (q-1)$)
e	Số mũ công khai RSA
EM	Thông điệp đã được định dạng (theo *EM* SA), chuỗi octet (encoded message)
GCD(.,.)	Ước chung lớn nhất của 2 số nguyên không âm
emBits	Độ dài dự kiến theo bit của *EM*
emLen	Độ dài dự kiến theo octet của *EM*
hLen	Độ dài đầu ra của hàm băm theo octet
k	Độ dài của môđun RSA (số *n*) theo octet
K	Khoá bí mật RSA
LCM(., . ,.)	Bội số chung nhỏ nhất của một danh sách các số nguyên không âm
m	Biểu diễn của thông điệp dưới dạng một số nguyên trong khoảng từ 0 đến *n*-1
M	Thông điệp dưới dạng chuỗi octet
mask	Mặt nạ, đầu ra của hàm MGF dưới dạng chuỗi octet
maskLen	Độ dài dự kiến của chuỗi mặt nạ theo octet
MGF	Hàm tạo mặt nạ
mgfSeed	Mầm tạo mặt nạ, là một chuỗi octet
mLen	Độ dài của thông điệp *M* theo octet
n	Môđun RSA
(*n*, *e*)	Khoá công khai RSA

p, q Hai nhân tử nguyên tố của môđun RSA ($n = p \cdot q$)

qInv Là số nguyên dương nghịch đảo của q theo môđun p, tức là $q \cdot qInv \equiv 1 \bmod p$

s Biểu diễn của chữ ký dưới dạng số nguyên giữa 0 và n-1

S Chữ ký, ở dạng chuỗi octet

sLen Độ dài phần phụ thêm của *EM* SA-PSS theo octet

x Số nguyên không âm

X Biểu diễn của *x* dưới dạng chuỗi octet

xLen Độ dài chủ định của chuỗi octet *X* thu được từ *x*

$\lambda(n)$ LCM(p-1, q-1)

|| Toán tử nối

5.2 Thuật toán RSA

5.2.1 Khoá công khai RSA

Khoá công khai RSA bao gồm 2 thành phần:

n Môđun RSA, là một số nguyên dương

e Số mũ công khai RSA, là một số nguyên dương

n là tích của hai số nguyên tố lẻ khác nhau là *p* và *q*, *e* là số nguyên giữa 3 và n-1 thỏa mãn điều kiện GCD(e, $\lambda(n)$) = 1 với $\lambda(n)$ = LCM(p-1, q-1). Sau đây chúng ta qui ước rằng *p* lớn hơn *q*.

5.2.2 Khoá bí mật RSA

Khoá bí mật RSA có một trong hai dạng biểu diễn:

5.2.2.1 Dạng biểu diễn thứ nhất gồm cặp (*n, d*) với:

n Môđun RSA, là một số nguyên dương

d Số mũ bí mật RSA, là một số nguyên dương

Số mũ bí mật *d* là số nguyên dương nhỏ hơn $\lambda(n)$ thỏa mãn

$$e \cdot d \equiv 1 \pmod{\lambda(n)}.$$

5.2.2.2 Dạng biểu diễn thứ hai chính là (*p, q, dP, dQ, qInv*), trong đó

p Nhân tử thứ nhất, là số nguyên dương

q Nhân tử thứ hai, là số nguyên dương

dP Là số nguyên dương sao cho $e \cdot dP \equiv 1 \bmod (p\text{-}1)$

dQ Là số nguyên dương sao cho $e \cdot dQ \equiv 1 \bmod (q\text{-}1)$

qInv Số nguyên dương nghịch đảo của q theo môđun p, tức là $q \cdot qInv \equiv 1 \bmod p$

5.3 Các hàm cơ sở chuyển đổi dữ liệu

5.3.1 Hàm cơ sở chuyển đổi từ dạng số nguyên sang dạng chuỗi octet

I2OSP (*x*, *xLen*)

Chức năng: Chuyển số nguyên không âm *x* thành một chuỗi octet có độ dài *xLen*

Đầu vào: *x* Số nguyên không âm cần chuyển đổi

Đầu ra: *X* Chuỗi octet tương ứng có độ dài *xLen*

Thông báo lỗi: "số nguyên quá lớn"

Các bước:

1. nếu $x \geq 256^{xLen}$, cho ra thông báo lỗi "số nguyên quá lớn" và dừng
2. viết số nguyên *x* duy nhất gồm *xLen* chữ số với cơ số 256:

$$x = x_{xLen-1} \cdot 256^{xLen-1} + x_{xLen-2} \cdot 256^{xLen-2} + \ldots + x_1 \cdot 256 + x_0$$

Với $0 \leq x_i < 256$ (chú ý rằng một hay nhiều chữ số đầu sẽ bằng 0 nếu x nhỏ hơn 256^{xLen-1}).

3. cho octet X_i giá trị nguyên x_{xLen-i} với $1 \leq i \leq xLen$. Cho ra chuỗi octet

$$X = X_1 X_2 \ldots X_{xLen}$$

5.3.2 Hàm cơ sở chuyển đổi từ dạng chuỗi octet về dạng số nguyên

OS2IP (X)

Chức năng: Chuyển chuỗi octet thành một số nguyên không âm

Đầu vào: *X* Chuỗi octet cần chuyển đổi

Đầu ra: *x* Số nguyên không âm tương ứng

Các bước:

1. cho $X_1X_2...X_{xLen}$ là các octet của *X* từ octet đầu tiên tới octet cuối cùng, x_{xLen-i} là giá trị nguyên của octet X_i với $1 \leq i \leq xLen$;
2. cho $x = x_{xLen-1} \cdot 256^{xLen-1} + x_{xLen-2} \cdot 256^{xLen-2} + \ldots + x_1 \cdot 256 + x_0$;
3. xuất ra *x*.

5.4 Các phép toán mật mã cơ sở

5.4.1 Phép toán cơ sở RSASP

RSASP (*K*, *m*)

Đầu vào: *K* Khoá bí mật RSA, với *K* có một trong hai dạng sau:

- cặp (*n*, *d*);
- bộ năm (*p*, *q*, *dP*, *dQ*, *qInv*);

m Biểu diễn của thông điệp, dưới dạng số nguyên giữa 0 và *n*-1

Đầu ra: *s* Biểu diễn của chữ ký, là số nguyên giữa 0 và *n*-1

Thông báo lỗi: "biểu diễn thông điệp ở ngoài miền hợp lệ"

Giả thiết: K là một khoá bí mật RSA hợp lệ

Các bước:

1. nếu biểu diễn của thông điệp *m* không nằm giữa 0 và *n*-1, cho ra thông báo lỗi "biểu diễn thông điệp ở ngoài miền hợp lệ" và dừng lại;
2. biểu diễn của chữ ký được tính như sau:
   a. nếu dạng thứ nhất (*n*, *d*) của *K* được sử dụng thì $s = m^d \bmod n$;
   b. nếu dạng thứ hai (*p*, *q*, *dP*, *dQ*, *qInv*) của *K* được sử dụng thì tiến hành như sau:
      i. Lấy $s_1 = m^{dP} \bmod p$ và $s_2 = m^{dQ} \bmod q$
      ii. Đặt $h = (s_1 - s_2) \cdot qInv \bmod p$
      iii. Đặt $s = s_2 + q \cdot h$
   c. xuất ra *s*.

5.4.2 Phép toán cơ sở RSAVP

RSAVP ((*n*, *e*), *s*)

Đầu vào: (*n*, *e*) Khoá công khai RSA

s Biểu diễn của chữ ký, là số nguyên giữa 0 và *n*-1

Đầu ra: *m* Biểu diễn của thông điệp, là số nguyên giữa 0 và *n*-1

Thông báo lỗi: "biểu diễn chữ ký ở ngoài miền hợp lệ"

Giả thiết: Khoá công khai RSA (*n*, *e*) là hợp lệ

Các bước:

1. nếu biểu diễn của chữ ký *s* không nằm giữa 0 và *n*-1, cho ra "biểu diễn chữ ký ở ngoài miền hợp lệ" và dừng lại;
2. đặt $m = s^e \bmod n$;
3. xuất ra *m*.

5.5 Lược đồ chữ ký RSA kèm phụ lục theo PSS

Các thao tác tạo chữ ký số áp dụng một thao tác định khuôn dạng vào một thông điệp trước khi nó được chuyển thành một biểu diễn thông điệp ở dạng số nguyên. Phép toán cơ sở RSASP được áp dụng vào biểu diễn thông điệp này để tạo ra chữ ký số. Đảo ngược quá trình này, các thao tác kiểm tra chữ ký áp dụng một phép toán cơ sở RSAVP vào một chữ ký để khôi phục một biểu diễn thông điệp, sau đó nó được chuyển thành thông điệp đã được định dạng ở dạng chuỗi octet. Thao tác kiểm tra được áp dụng vào thông điệp ban đầu và thông điệp đã được định dạng để xác định xem chúng có tương ứng với nhau hay không.

5.5.1 Thao tác tạo chữ ký

RSASSA-PSS-SIGN(*K*, *M*)

Đầu vào: *K* Khoá bí mật RSA của người ký

M Thông điệp sẽ được ký, là một chuỗi octet

Đầu ra: *S* Chữ ký, chuỗi octet có độ dài *k*, với *k* là độ dài của môđun RSA theo octet

Thông báo lỗi: "văn bản quá dài", "lỗi định dạng "

Các bước:

1. *mã hoá EMSA-PSS* : Áp dụng thao tác EMSA-PSS-ENCODE (được giới thiệu ở phần sau) vào văn bản *M* để tạo ra thông điệp được định dạng *EM* có độ dài $\lceil (modBits-1)/8 \rceil$ octet sao cho độ dài bit của số nguyên OS2IP (*EM*) nhiều nhất là *modBits*-1, với *modBits* là độ dài theo bit của số n (môđun RSA):

   EM = EMSA-PSS-ENCODE (*M*, *modBits*-1)

   Chú ý rằng độ dài octet của *EM* sẽ bằng *k* –1 nếu *modBits*-1 chia hết cho 8 và bằng *k* nếu *modBits*-1 không chia hết cho 8. Nếu hàm EMSA-PSS-ENCODE cho ra thông báo lỗi "văn bản quá dài" thì RSASSA-PSS-SIGN cũng cho ra thông báo lỗi "văn bản quá dài" và dừng lại. Nếu EMSA-PSS-ENCODE cho ra thông báo "lỗi định dạng" thì RSASSA-PSS-SIGN cũng cho ra thông báo "lỗi định dạng" và dừng lại.

2. *chữ ký RSA*:

   a. chuyển thông điệp đã được định dạng (chuỗi octet) *EM* thành biểu diễn thông điệp ở dạng số nguyên *m*;

      m = OS2IP(*EM*)

   b. áp dụng phép toán cơ sở RSASP với K là khoá bí mật RSA và biểu diễn thông điệp *m* để tạo ra biểu diễn chữ ký là số nguyên *s*:

      s = RSASP(*K*, *m*);

   c. chuyển chữ ký s dạng số nguyên thành chữ ký S dạng chuỗi octet có độ dài k:

      S = I2OSP(*s*, *k*)

3. *xuất ra chữ ký S.*

5.5.2 Thao tác kiểm tra chữ ký

RSASSA-PSS-VERIFY((*n, e*), *M, S*)

Đầu vào: (*n, e*) Khoá công khai RSA của người ký

M Thông điệp mà chữ ký của nó cần được kiểm tra, là chuỗi octet

S Chữ ký được kiểm tra, chuỗi octet có độ dài *k*, với k là độ dài theo octet của số *n*, môđun RSA

Đầu ra: "chữ ký hợp lệ" hoặc "chữ ký không hợp lệ"

Các bước:

1. kiểm tra độ dài: Nếu độ dài của chữ ký *S* không là *k* octet, cho ra thông báo lỗi "chữ ký không hợp lệ" và dừng;
2. kiểm tra chữ ký RS;
   a. chuyển chữ ký *S* thành biểu diễn chữ ký ở dạng số nguyên *s*;

      s = OS2IP (S)

   b. áp dụng phép toán cơ sở RSAVP với khoá công khai RSA là (*n, e*) và biểu diễn chữ ký *s* để tạo ra m là số nguyên biểu diễn thông điệp;

      m = RSAVP ((n, e), s)

   c. chuyển biểu diễn thông điệp *m* thành thông điệp đã được định dạng *EM* có độ dài *emLen* = $\lceil$ (*modBits*-1)/8$\rceil$ octet, với *modBits* là độ dài theo bit của số *n* (môđun RSA):

      EM = I2OSP (m, *emLen*)

Chú ý rằng *emLen* sẽ bằng *k*-1 nếu *modBits*-1 chia hết cho 8 và bằng *k* nếu *modBits*-1 không chia hết cho 8. Nếu I2OSP cho ra thông báo lỗi "số nguyên quá lớn" thì RSASSA-PSS-VERIFY cho ra thông báo lỗi "chữ ký không hợp lệ" và dừng lại.

3. *kiểm tra EMSA-PSS*: Áp dụng thao tác kiểm tra EMSA-PSS-VERIFY (sẽ được mô tả ở phần 5.6 dưới đây) vào thông điệp *M* và thông điệp đã được định dạng *EM* để xác định xem chúng có tương ứng với nhau hay không;

   Result = EMSA-PSS-VERIFY (*M, EM, modBits*-1)

4. nếu kết quả (Result) là "phù hợp" thì cho ra "chữ ký hợp lệ". Ngược lại sẽ cho ra "chữ ký không hợp lệ".

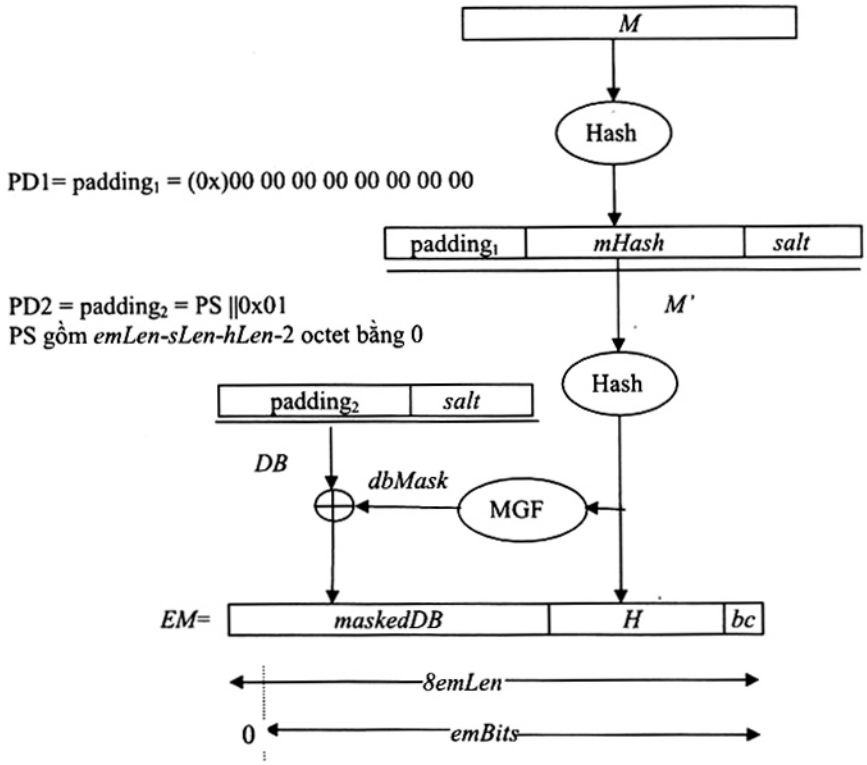
5.6 Phương pháp định dạng cho chữ ký kèm phụ lục theo PSS (EMSA-PSS)

5.6.1 Thao tác định dạng

Phương pháp định dạng được tham số hoá bằng cách chọn:

- hàm băm (cố định với khoá RSA đã cho);
- hàm tạo mặt nạ (cố định với khoá RSA đã cho) và;
- độ dài phần phụ thêm (có thể thay đổi với khoá RSA đã cho).

Các hàm băm và hàm tạo mặt nạ được đề xuất sẽ được mô tả trong phần 6 và 5.6.3. Hình 2 minh họa thao tác định dạng.



Hình 2 - Minh họa thao tác định dạng

Công thức để tính *EM:*

$((\mathrm{PD2}||r) \oplus \mathrm{MGF}(h(\mathrm{PD1}||h(\mathrm{M})||r))) \,||\, h(\mathrm{PD1}||h(\mathrm{M})||r)) \,||\, 0xbc$

EMSA-PSS-ENCODE (*M, emBits*)

Lựa chọn : *h* Hàm băm (độ dài đầu ra của nó theo octet là *hLen*)

MGF Hàm tạo mặt nạ

sLen Độ dài chủ định của phần phụ thêm theo octet

Đầu vào: *M* Văn bản để mã hoá, là một chuỗi octet

emBits độ dài tối đa theo bit của số nguyên OS2IP (*EM*), ít nhất bằng *8hLen* + *8sLen* + *9*

Đầu ra: *EM* Văn bản đã được mã, đó là chuỗi octet có độ dài *emLen* = ⌈*emBits*/8⌉

Thông báo lỗi: "lỗi định dạng"; "văn bản quá dài"

Các bước:

1. nếu độ dài của *M* lớn hơn giới hạn đầu vào cho hàm băm (2^{64}-1 đối với SHA-256) thì cho ra thông báo lỗi "văn bản quá dài" và dừng;
2. lấy *mHash* = *h(M)*, đó là một chuỗi octet dài *hLen*;
3. nếu *emLen* < *hLen* + *sLen* + 2, cho ra thông báo "lỗi định dạng" và dừng;
4. tạo ra chuỗi octet ngẫu nhiên salt có độ dài *sLen*; nếu *sLen* = 0 thì salt không có;
5. đặt *M'* = (0x)00 00 00 00 00 00 00 00 || *mHash* ||*salt;*
6. lấy *H* = *Hash(M')*, đó là một chuỗi octet dài *hLen;*
7. lấy *PS* là một chuỗi octet bằng 0 dài *emLen – hLen –sLen –2;*
8. lấy *DB* = *PS* || 0x01 || *salt* ; *DB* là một chuỗi octet dài *emLen-hLen –1;*
9. lấy *maskedDB* = *DB* ⊕ *dbMask;*
10. đặt *8emLen-emBits* bit đầu tiên bên trái của octet đầu tiên bên trái trong *maskedDB* bằng 0;
11. lấy *EM* = *maskedDB* ||*H* || *0xbc;*
12. xuất ra *EM.*

5.6.2 Thao tác kiểm tra

EMSA-PSS-VERIFY (*M, EM, emBits*)

Lựa chọn: *h* Hàm băm (độ dài đầu ra của nó theo octet là *hLen*)

MGF Hàm tạo mặt nạ

sLen Độ dài dự kiến của phần thêm theo octet

Đầu vào: *M* Thông điệp cần kiểm tra chữ ký, là chuỗi octet

EM Thông điệp đã được định dạng, là chuỗi octet có độ dài *emLen* = ⌈*emBits/8*⌉

emBits Độ dài tối đa theo bit của số nguyên OS2IP(*EM*), tối thiểu là *8hLen* + 8*sLen* + 9

Đầu ra: "phù hợp" hoặc "không phù hợp".

Các bước:

1. nếu độ dài của *M* lớn hơn giới hạn đầu vào của hàm băm (2^{64} – 1 octet đối với SHA-256), thì đưa ra thông báo "không phù hợp" và dừng;
2. đặt *mHash* = *h(M)*, là chuỗi octet có độ dài *hLen*;
3. nếu emBits < 8*hLen*+8*sLen*+9, đưa ra thông báo "không phù hợp" và dừng;
4. nếu octet đầu tiên bên phải của *EM* không chứa giá trị *bc*, đưa ra thông báo "không phù hợp" và dừng;
5. đặt *maskedDB* là *emLen-hLen-1* octet đầu tiên bên trái của *EM*, và *H* là *hLen* octet tiếp theo;
6. nếu *8emLen-emBits* bit đầu tiên bên trái của octet đầu tiên bên trái trong *maskedDB* không phải tất cả bằng 0, đưa ra thông báo "không phù hợp" và dừng;
7. đặt *dbMask* = MGF(*H, emLen-hLen-1*);
8. đặt *DB* = *maskedDB* ⊕ *dbMask*;
9. thiết lập *8emLen-emBits* bit đầu tiên bên trái của *DB* bằng 0;
10. nếu *emLen-hLen-sLen*-2 octet đầu tiên bên trái của *DB* không phải bằng 0 hoặc nếu octet tại vị trí thứ *emLen-hLen-sLen-1* không bằng 0x01, đưa ra thông báo "không phù hợp" và dừng;
11. đặt *salt* bằng *sLen* octet cuối cùng của *DB*;
12. đặt *M'* = 00 00 00 00 00 00 00 00 ‖ *mHash* ‖ *salt*

    M' là chuỗi octet có độ dài *8+hLen+sLen* với 8 octet bằng 0 khởi đầu;
13. đặt *H'* = *h(M')*, là chuỗi octet có độ dài *hLen*;
14. nếu *H* = *H'*, đưa ra thông báo "phù hợp". Ngược lại, đưa ra thông báo "không phù hợp".

5.6.3 Hàm tạo mặt nạ MGF dựa vào hàm băm

MGF(*mgfSeed, maskLen*)

Lựa chọn:	*h*	Hàm băm (độ dài đầu ra của nó theo octet là *hLen*)
Đầu vào:	*mgfSeed*	Mầm được dùng để tạo mặt nạ, là chuỗi octet
	maskLen	Độ dài chủ ý theo octet của mặt nạ, nhiều nhất là 2^{32}*hLen*
Đầu ra:	*mask*	Mặt nạ, là chuỗi octet có độ dài *maskLen*

Thông báo lỗi: "mặt nạ quá dài"

Các bước:

1. nếu *maskLen* > 2^{32} *hLen*, cho ra thông báo lỗi "mặt nạ quá dài" và dừng;

2. lấy *T* là chuỗi octet rỗng ;

3. với *counter* chạy từ 0 tới $\lceil maskLen/hLen \rceil$ -1, thực hiện các bước.

   a. chuyển *counter* thành một chuỗi octet C có độ dài 4 octet;

   $$C = I2OSP\ (counter, 4)$$

   b. nối mầm *mgfSeed* với *C*, tính hàm băm của chuỗi này. Sau đó nối chuỗi octet *T* với giá trị băm vừa thu được.

   $$T = T \parallel h(mgfSeed \parallel C)$$

4. xuất ra *maskLen* octet đầu tiên của *T* như là chuỗi octet *mask*.

6 Hàm băm SHA–256

6.1 Một số khái niệm và thuật ngữ

6.1.1 Biến, tham số

a, b, c, …, h	Các biến làm việc, các biến này là các từ 32 bit được sử dụng để tính toán các giá trị băm $H^{(i)}$
$H^{(i)}$	Giá trị băm thứ *i*. $H^{(0)}$ là giá trị băm khởi tạo. $H^{(N)}$ là giá trị băm cuối cùng được sử dụng để xác định bản tóm lược của văn bản
$Hj^{(i)}$	Từ thứ *j* của giá trị băm thứ *i*, $H_0^{(i)}$ là từ ngoài cùng bên trái của giá trị băm thứ *i*
Kt	Hằng số được sử dụng cho vòng lặp thứ *t* của quá trình băm
k	Số lượng bit 0 được bổ sung cho thông điệp trong bước bổ sung dữ liệu
l	Độ dài của thông điệp (ký hiệu là *M*) theo đơn vị bit
m	Số lượng bit trong một khối thông điệp $(M^{(i)})$
M	Thông điệp cần băm
$M^{(i)}$	Khối thông điệp thứ*l*
$M_j^{(i)}$	Từ thứ *j* của khối thông điệp thứ *i*. $M_0^{(i)}$ là từ ngoài cùng bên trái của khối thông điệp thứ*l*
n	Số lượng các bit quay vòng hoặc dịch đi khi xử lý một từ
N	Số khối của bản thông điệp sau khi đã được bổ sung
T	Biến tạm thời lưu từ (32 bit) trong quá trình tính toán
W_t	Từ (32 bit) thứ *t* trong chuỗi thông điệp.

6.1.2 Các ký hiệu tính toán

^	Phép AND bit
∨	Phép OR bit
⊕	Phép XOR bit
¬	Phép bù bit
+	Phép cộng mođun 2^{32}
<<	Phép dịch trái, $x<<n$ có nghĩa là dịch x đi n bit sang trái (loại bỏ n bit ngoài cùng bên trái), bổ sung n bit 0 vào bên phải;
>>	Phép dịch phải, $x>>n$ có nghĩa là dịch x đi n bit sang phải (loại bỏ n bit ngoài cùng bên phải), bổ sung n bit 0 vào bên trái.

6.1.3 Chuỗi các bit và các số nguyên

- một chữ số hexa là một phần tử thuộc tập {0, 1,..., 9, a, ..., f};
- một từ là một chuỗi 32 bit có thể được biểu diễn bởi dãy các số hexa (8 chữ số hexa). Để chuyển đổi một chuỗi bit thành dạng một chuỗi hexa chúng ta có thể lần lượt chuyển đổi từng bộ 4 bit thành một số hexa tương ứng;
- một số nguyên lớn hơn bằng 0 và nhỏ hơn 2^{64} có thể được biểu diễn như một từ (nếu nhỏ hơn 2^{32}) hoặc một cặp từ (nếu lớn hơn hoặc bằng 2^{32}). Chúng ta có thể dùng hai từ, để biểu diễn độ dài của thông điệp theo bit;
- đối với SHA-256, mỗi khối thông điệp gồm 512 bit, nó được biểu diễn dưới dạng 16 từ (32 bit).

6.1.4 Các phép tính với các từ

- các phép tính logic từng bit với từ: ∧ , ∨ , ⊕, và ¬;
- cộng hai từ môđun 2^{32};
- phép dịch phải $SHR^n(x)$, với x là một từ (32 bit) và n là một số nguyên $0 \le n < 32$, được định nghĩa như sau;

  $SHR^n(x) = x >> n.$

- phép dịch vòng phải $ROTR^n(x)$, với x là một từ (32 bit) và n là một số nguyên $0 \le n < 32$, được định nghĩa như sau.

$$ROTR^n(x) = (x >> n) \vee (x << 32 - n)$$

Tức là lấy n bit loại ra ở phía cuối bổ sung vào phía đầu theo đúng thứ tự đã có từ trước.

6.2 Thuật toán

6.2.1 Các hàm và các hằng được sử dụng cho thuật toán

6.2.1.1 Các hàm

SHA-256 sử dụng 6 hàm logic, mỗi hàm đều thực hiện trên các từ (32 bit), các từ này được biểu diễn bởi các biến x, y, z. Kết quả đầu ra của các hàm này là một từ 32 bit mới.

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(z, y, x) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$$\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

6.2.1.2 Các hằng số

SHA-256 sử dụng chuỗi 64 từ (32 bit) làm hằng số, $K_0^{\{256\}}, K_1^{\{256\}}, ..., K_{63}^{\{256\}}$. Các từ này lần lượt là 32 bit đầu tiên của phần thập phân khi lấy căn bậc ba 64 số nguyên tố đầu tiên. Khi biểu diễn dưới dạng hexa các hằng số có giá trị như liệt kê dưới đây:

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

6.2.2 Bước tiền xử lý

Tiền xử lý được thực hiện trước khi bắt đầu tính toán giá trị băm. Bước tiền xử lý được chia làm 3 bước nhỏ: bổ sung thông điệp, chia thông điệp đã được bổ sung thành các khối, và thiết lập các giá trị băm khởi đầu $H^{(0)}$.

6.2.2.1 Bổ sung thông điệp

Giả sử thông điệp *M* có độ dài là *l* bit, bổ sung bit "1" vào cuối thông điệp, tiếp theo là *k* bit 0, với *k* thoả mãn $l+1+k = 448 \bmod 512$. Cuối cùng bổ sung một khối 64 bit để lưu giá trị *l* (độ dài thật của thông điệp). Như vậy thông điệp sau khi đã được bổ sung có độ dài là bội của 512 bit.

6.2.2.2 Chia thông điệp thành khối sau khi đã được bổ sung

Thông điệp sau khi đã được bổ sung được chia thành *N* khối 512 bit, $M^{(1)}$, $M^{(2)}$, ... , $M^{(N)}$. Mỗi khối thông điệp gồm 16 từ (32 bit). 32 bit đầu tiên của khối thông điệp thứ *i* là $M_0^{(i)}$, 32 bit tiếp theo là $M_1^{(i)}$ và 32 bit cuối cùng của khối thông điệp thứ *i* là $M_{15}^{(i)}$.

6.2.2.3 Thiết lập các giá trị băm khởi đầu $H^{(0)}$

Các giá trị băm khởi đầu gồm 8 từ (32 bit):

$H_0^{(0)}$ = 6a09e667

$H_1^{(0)}$ = bb67ae85

$H_2^{(0)}$ = 3c6ef372

$H_3^{(0)}$ = a54ff53a

$H_4^{(0)}$ = 510e527f

$H_5^{(0)}$ = 9b05688c

$H_6^{(0)}$ = 1f83d9ab

$H_7^{(0)}$ = 5be0cd19

6.2.3 Thuật toán tính giá trị băm

SHA-256 được sử dụng để tính giá trị băm của một thông điệp có độ dài là *l*, với $0 \le l < 2^{64}$. Thuật toán sử dụng một chuỗi 64 từ (32 bit) được tạo ra từ một khối thông điệp đầu vào, 8 biến làm việc cho mỗi từ 32 bit, giá trị băm trung gian gồm 8 từ (32 bit), kết quả cuối cùng của SHA-256 là 256 bit mã băm hay còn gọi là bản tóm lược thông điệp.

Các từ tạo ra từ khối thông điệp đầu vào được ký hiệu là $W_0, W_1, ..., W_{63}$, tám biến làm việc được ký hiệu là *a, b, c, d, e, f, g* và *h*. Các từ của kết quả băm được ký hiệu là $H_0^{(t)}, H_1^{(t)}, ..., H_7^{(t)}$, chúng được gán các giá trị băm khởi đầu, $H^{(0)}$, và sẽ được thay thế bởi các giá trị băm trung gian (sau khi mỗi khối thông điệp được xử lý), $H^{(i)}$, và cuối cùng là giá trị băm, $H^{(N)}$.

6.2.3.1 Tiền xử lý SHA-256

Thông điệp *M* được xử lý như mục 6.2.2.

6.2.3.2 Tính toán giá trị băm SHA-256

Việc tính toán giá trị băm SHA-256 sử dụng các hàm và hằng được định nghĩa trong phần 6.2.1. Phép tính "+" được xem là cộng môđun 2^{32}.

Sau khi hoàn thành bước tiền xử lý, mỗi khối thông điệp, $M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$ tuần tự được xử lý theo các bước dưới đây:

Với *i* từ 0 đến *n*

{

1. Tính các từ W_t từ khối thông điệp

$$W_t = M_t^{(i)} \qquad \text{với } 0 \le t \le 15$$

$$W_t = \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_1^{\{256\}}(W_{t-15}) + W_{t-16} \qquad \text{với } 16 \le t \le 63$$

2. Khởi gán tám biến làm việc a, b, c, d, e, f, g và h bằng các giá trị băm thứ (i-1)

$a = H_0^{(i-1)}$

$b = H_1^{(i-1)}$

$c = H_2^{(i-1)}$

$d = H_3^{(i-1)}$

$e = H_4^{(i-1)}$

$f = H_5^{(i-1)}$

$g = H_6^{(i-1)}$

$h = H_7^{(i-1)}$

3. với t từ 0 đến 63, tính

{

$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$

$T_2 = \Sigma_0^{\{256\}}(a) + Maj(a,b,c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

4. Tính giá trị băm trung gian thứ i, $H^{(i)}$

$H_0^{(i)} = a + H_0^{(i-1)}$

$H_1^{(i)} = b + H_1^{(i-1)}$

$H_2^{(i)} = c + H_2^{(i-1)}$

$H_3^{(i)} = d + H_3^{(i-1)}$

$H_4^{(i)} = e + H_4^{(i-1)}$

$H_5^{(i)} = f + H_5^{(i-1)}$

$H_6^{(i)} = g + H_6^{(i-1)}$

$H_7^{(i)} = h + H_7^{(i-1)}$

}

Sau khi xử lý *N* lần (tương ứng với *N* khối thông điệp), kết quả đầu ra hàm băm SHA-256 của thông điệp *M* là:

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

6.2.4 Dữ liệu kiểm tra

Để giúp các nhà lập trình kiểm tra tính đúng đắn của chương trình do mình xây dựng, TCVN này đưa ra các giá trị được sử dụng để kiểm tra:

6.2.4.1 Thông điệp đầu vào chỉ có một khối

Dữ liệu đầu vào *M*: "abc"

Kết quả đầu ra khi băm *M*: "ba7816bf8f01cfea414140de5dae2223"

"b00361a396177a9cb410ff61f20015ad"

6.2.4.2 Thông điệp đầu vào gồm nhiều khối

Dữ liệu đầu vào *M*: "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"

Kết quả đầu ra khi băm *M*: "248d6a61d20638b8e5c026930c3e6039"

"a33ce45964ff2167f6ecedd419db06c1",

6.2.4.3 Đầu vào là một thông điệp dài

Dữ liệu đầu vào *M*: Một triệu chữ "a"

Kết quả đầu ra khi băm *M*: "cdc76e5c9914fb9281a1c7e284d73e67"

"f1809a48a497200e046d39ccc7112cd0"

7 Bộ tạo số giả ngẫu nhiên dùng AES-128

Phần này sẽ mô tả bộ tạo số giả ngẫu nhiên sử dụng thuật toán mã hoá AES-128.

7.1 Một số ký hiệu

$AES_K(M)$	Hàm mã hoá AES-128. Thực hiện việc mã khối thông điệp *M* (128 bit) bởi khoá *K* (128 bit). Trả về 128 bít dữ liệu đã mã của *M* (Chi tiết về hàm AES-128 được nêu trong tài liệu FIPS 197 "Advanced Encryption Standard")
DT_j	Giá trị 128 bit, là ngày tháng/thời gian (date/time) của hệ thống
XOR	Phép toán XOR bit
$\lceil x \rceil$	Số nguyên bé nhất lớn hơn hay bằng x, ví dụ: $\lceil 6 \rceil = \lceil 5.1 \rceil = 6$
$\leftarrow$	Phép gán giá trị; ví dụ: a $\leftarrow$ b có nghĩa là gán b cho a

7.2 Thuật toán

Đầu vào:

L	Số bit cần tạo ngẫu nhiên
V_0	128 bit ngẫu nhiên, lựa chọn bởi người dùng
DT_j	128 bit là ngày tháng/thời gian của hệ thống
K	128 bit khoá cho AES-128

Đầu ra:

Số giả ngẫu nhiên p có L bit

Các bước:

p = null

Với j từ 1 đến $\lceil L/128 \rceil$, thực hiện các bước sau:

$I_j = AES_K(DT_j)$

$x_j = AES_K(I_j \text{ XOR } V_{j-1})$

$V_j = AES_K(I_j \text{ XOR } x_j)$

$p \leftarrow p \,||\, x_j$

$p \leftarrow$ Lấy L bit bên trái của p

8 Tiêu chuẩn tham số sử dụng trong chữ ký số RSA-PSS

Để sử dụng lược đồ chữ ký số RSA-PSS an toàn cần phải tuân thủ các yêu cầu sau đây:

8.1 Các yêu cầu chung

1. cặp khoá RSA dùng để ký thì không được dùng cho mục đích khác (chẳng hạn dùng lại để mã thông điệp);
2. hai số nguyên tố p, q và số mũ bí mật d cần phải được giữ bí mật tránh việc bị truy cập bất hợp pháp, làm lộ hoặc sửa đổi. Môđun n và số mũ công khai e phải được công bố công khai;
3. mỗi người sử dụng cần có môđun n riêng;
4. độ dài của môđun n *(nlen)* không được nhỏ hơn 1024 bit và nên được thay đổi theo thời gian như sau.

Thời gian sử dụng	**Độ an toàn**	***nlen* tối thiểu**
Tới năm 2010	80	1024
Tới năm 2030	112	2048
Sau 2030	128	3072

Trong đó, độ an toàn (*security_strength*) là một số nguyên biểu thị lượng tính toán cần thiết để phá hệ mã.

Vì các phương pháp phá hệ mã thường xuyên được hoàn thiện nên cần phải định kỳ 3 đến 5 năm một lần xem xét lại *nlen* tối thiểu (có thể tham khảo chi tiết yêu cầu này trong tài liệu *NIST Special Publication 800-57: Recommendation for Key Management – Part1: general, May,2006*).

8.2 Yêu cầu đối với các khoá RSA

1. số mũ công khai *e* cần phải được chọn với các ràng buộc sau:
   a) số mũ công khai *e* cần được chọn trước khi tạo số mũ bí mật *d*;
   b) số mũ công khai *e* cần phải là số nguyên dương lẻ sao cho: $65,537 \le e < 2^{nlen-2 \times security\_strength}$

Với *nlen* là độ dài của môđun *n* theo bit.

Chú ý rằng *e* có thể là giá trị bất kỳ mà thoả mãn ràng buộc 1(b); *p* và *q* sẽ được chọn (trong mục 2) sao cho *e* là nguyên tố cùng nhau với cả (*p*-1) và (*q*-1).

2. hai số nguyên tố *p* và *q* được tạo ngẫu nhiên và giữ bí mật cần phải được chọn với các ràng buộc sau:
   a) (*p*-1) và (*q*-1) cần phải nguyên tố cùng nhau với số mũ công khai *e*;
   b) mỗi một số trong bốn số (*p* + 1), (*p* - 1) và (*q* + 1), (*q* - 1) cần phải có các nhân tử nguyên tố lớn hơn $2^{security\_strength+20}$;
   c) nhân tử nguyên tố bí mật *p*, *q* cần phải được chọn ngẫu nhiên từ các số nguyên tố thoả mãn $(\sqrt{2})(2^{(nlen/2)-1}) \le q < p \le \left(2^{nlen/2} - 1\right)$;
3. số mũ bí mật d cần phải được lựa chọn sau khi tạo *p* và *q* với các ràng buộc:
   a) số mũ d cần phải lớn hơn $2^{nlen/2}$, và
   b) $d = e^{-1} \bmod (\text{LCM}((p-1), (q-1)))$.

(Chi tiết về hàm tạo các tham số RSA có thể tham khảo trong tài liệu *FIPS 186-3: Digital Signature Standard*).