TCVN **TIÊU CHUẨN QUỐC GIA**

TCVN 11367-2:2016
ISO/IEC 18033-2:2006

Xuất bản lần 1

CÔNG NGHỆ THÔNG TIN - CÁC KỸ THUẬT AN TOÀN - THUẬT TOÁN MẬT MÃ - PHẦN 2: MẬT MÃ PHI ĐỐI XỨNG

Information technology -- Security techniques -- Encryption algorithms -- Part 2: Asymmetric ciphers

HÀ NỘI - 2016

Mục Lục

Lời nói đầu

TCVN 11367-2:2016 hoàn toàn tương đương với ISO/IEC 18033-2:2006.

TCVN 11367-2:2016 do Cục Quản lý mật mã dân sự và Kiểm định sản phẩm mật mã biên soạn, Ban Cơ yếu Chính phủ đề nghị, Tổng cục Tiêu chuẩn Đo lường Chất lượng thẩm định, Bộ Khoa học và Công nghệ công bố.

Bộ tiêu chuẩn TCVN 11367 *Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán mật mã* gồm 04 phần:

- TCVN 11367-1:2016 (ISO/IEC 18033-1:2015) Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán mật mã – Phần 1: Tổng quan.

- TCVN 11367-2:2016 (ISO/IEC 18033-2:2006) Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán mật mã – Phần 2: Mật mã phi đối xứng.

- TCVN 11367-3:2016 (ISO/IEC 18033-3:2010) Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán mật mã – Phần 3: Mã khối.

- TCVN 11367-4:2016 (ISO/IEC 18033-4:2011) Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán mật mã – Phần 4: Mã dòng.

Giới thiệu

Tổ chức tiêu chuẩn hóa quốc tế (ISO) và Ủy ban kỹ thuật điện quốc tế (IEC) hướng tới thực tế việc tuân thủ tiêu chuẩn này có thể liên quan tới việc sử dụng các bằng sáng chế

ISO và IEC không liên quan đến các bằng chứng tính hợp lệ và phạm vi áp dụng của các bản quyền sáng chế này. Người sở hữu bản quyền sáng chế phải tự đảm bảo ISO và IEC rằng họ sẵn sàng đàm phán giấy phép theo các điều khoản và không phân biệt một cách hợp lý và các điều kiện với người yêu cầu trên toàn thế giới. Trong khía cạnh này, tuyên bố của người sáng chế phải được đăng kí với ISO và IEC. Thông tin có thể tìm được từ:

ISO/IEC JTC 1/SC 27 Standing Document 8 (SDH) "*Patent Information*"

Tài liệu hiện hành 8 (SD8) được công bố công khai tại: http://www.ni.din.de/sc27

Chú ý rằng khả năng một số yếu tố của tiêu chuẩn này có thể là đối tượng của bản quyền sáng chế khác với những điều đã xác định ở trên. ISO và IEC sẽ không chịu trách nhiệm xác định bất kỳ hoặc tất cả các bản quyền sáng chế như vậy.

TIÊU CHUẨN QUỐC GIA **TCVN 11367-2:2016**

Công nghệ thông tin - Các kỹ thuật an toàn - Thuật toán mật mã - Phần 2: Mật mã phi đối xứng

Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric ciphers

1 Phạm vi áp dụng

Tiêu chuẩn này đặc tả một số mật mã phi đối xứng. Các đặc tả này qui định các giao diện chức năng và các phương pháp đúng đắn sử dụng các mật mã loại này nói chung, cũng như chính xác hóa chức năng và định dạng bản mã cho một số mật mã phi đối xứng (mặc dù có thể chọn các hệ thống phù hợp và các định dạng khác để lưu trữ và truyền bản mã).

Phụ lục A cung cấp cú pháp ASN.1 cho các định danh đối tượng, các khóa công khai, và các cấu trúc tham số liên kết với thuật toán được đặc tả trong phần này của ISO/IEC 18033.

Tuy nhiên, các đặc tả này không qui định các giao thức thu được một cách tin cậy khóa công khai, để chứng minh việc sở hữu khóa mật, hay để xác nhận khóa công khai hoặc khóa mật; xem ISO/IEC 117700-3 hướng dẫn các vấn đề quản lí khóa.

Các mật mã phi đối xứng được đặc tả trong tiêu chuẩn này (của bộ TCVN 11367 (ISO/IEC 18033)) được chỉ ra tại Điều 7.6.

CHÚ THÍCH Một cách vắn tắt, mật mã phi đối xứng gồm:

- ECIES-HC; PSEC-HC; ACE-HC: Hệ mật lai ghép tổng quát dựa trên mật mã Elgamal;

- RSA-HC: Mật mã lai ghép dựa trên biến đổi RSA;

- RSAES: Lược đồ đệm OAEP dựa trên biến đổi RSA;

- HIME(R): Lược đồ dựa trên tính khó của bài toán phân tích số.

2 Tài liệu viện dẫn

Các tài liệu viện dẫn sau rất cần thiết cho việc áp dụng tiêu chuẩn này. Đối với các tài liệu viện dẫn ghi năm công bố thì áp dụng phiên bản được nêu. Đối với các tài liệu viện dẫn không ghi năm công bố thì áp dụng phiên bản mới nhất, bao gồm cả các sửa đổi, bổ sung (nếu có).

ISO/IEC 9797-1:1999, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Machanisms using a block cipher*

ISO/IEC 9797-2:2002, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Machanisms using a dedicated hash functions*

ISO/IEC 10118-2:2000, *Information technology – Security techniques – Hash functions – Part 2: hash functions using an n-block cipher*

ISO/IEC 10118-3:2004, *Information technology – Security techniques – Hash functions – Part 3: dedicated hash functions*

ISO/IEC 18033-3:2005, *Information technology – Security techniques – Encryption algorithms – Part 3: blocks ciphers*

3 Định nghĩa

Trong tiêu chuẩn này áp dụng các thuật ngữ và định nghĩa dưới đây:

CHÚ THÍCH Trong tiêu chuẩn này, ở những nơi phù hợp các tham chiếu tiếp theo được đưa ra trong các Điều mà chứa những định nghĩa và/hoặc soạn thảo chi tiết hơn.

3.1

Mật mã phi đối xứng (asymmetric cipher)

Hệ thống dựa vào các kỹ thuật mật mã phi đối xứng, ở đó phép biến đổi khóa công khai được sử dụng để mã hóa và phép biến đổi khóa riêng dùng để giải mã.

[ISO/IEC 9798-1:2010, 3.2]

CHÚ THÍCH Xem Điều 7.

3.2

Kỹ thuật mật mã phi đối xứng (asymmetric cryptigraphic technique)

Kỹ thuật mật mã phi đối xứng sử dụng hai phép biến đổi liên quan đến nhau, phép biến đổi công khai (được xác định bởi khóa công khai) và biến đổi mật (được xác định bởi khóa mật). Cả hai phép biến đổi này có tính chất là cho biết phép biến đổi công khai, về mặt tính toán không thể có khả năng xác định được phép biến đổi bí mật.

[ISO/IEC 11770-1:1996]

3.3

Cặp khóa phi đối xứng (asymmetric key pair)

Cặp khóa liên quan với nhau, khóa công khai và khóa bí mật, ở đây khóa công khai xác định phép biến đổi công khai, khóa bí mật xác định phép biến đổi bí mật

[ISO/IEC 9798-1:1977]

CHÚ THÍCH Xem Điều 7, 8.1.

3.4

Bit (bit)

Một trong hai kí hiệu '0' và '1'

CHÚ THÍCH Xem Điều 5.2.1.

3.5

Xâu bit (bit string)

Dãy các bit

CHÚ THÍCH Xem Điều 5.2.1.

3.6

Khối (block)

Xâu bit có độ dài xác định

[ISO/IEC 18033-1]

CHÚ THÍCH Trong phần này của ISO/IEC 18033, khối được giới hạn là xâu bộ tám (được minh họa một cách tự nhiên như những xâu bit).

3.7

Mã khối (block cipher)

Mã đối xứng với tính chất là thuật toán mã hóa thao tác trên các khối của bản rõ, nghĩa là trên xâu bit có độ dài xác định, kết quả cho ra khối bản mã

[ISO/IEC 18033-1]

CHÚ THÍCH 1 Xem Điều 6.4.

CHÚ THÍCH 2 Trong phần này của ISO/IEC 18033, các khối của bản rõ/ bản mã chỉ giới hạn là các xâu bộ tám (được minh họa một cách tự nhiên như những xâu bit).

3.8

Mật mã (cipher)

Kỹ thuật mật mã dùng để bảo mật dữ liệu, bao gồm từ ba quá trình thành phần: thuật toán mã hóa, thuật toán giải mã, phương pháp tạo khóa

[ISO/IEC 18033-1]

3.9

Bản mã (cipher text)

Dữ liệu được biến đổi để che giấu nội dung thông tin trong đó

[ISO/IEC 10116:1997]

3.10

Nhóm cụ thể (concrete group)

Sự mô tả dưới dạng tường minh nhóm abel hữu hạn, cùng với các thuật toán để thực thi các phép toán trên nhóm và mã hóa, giải mã các phần tử nhóm dưới dạng xâu octet

CHÚ THÍCH Xem Điều 10.1.

3.11

Hàm băm mật mã (cryptographic hash function)

Hàm ánh xạ các xâu bộ tám có độ dài bất kì vào xâu bộ tám có độ dài cố định, sao cho bằng tính toán không thể tìm ra mối quan hệ giữa đầu vào và đầu ra, và sao cho, cho trước một phần đầu ra, nhưng không cho trước đầu vào, bằng tính toán không thể đoán được bất kì bit nào của phần đầu ra còn lại. Yêu cầu chính xác về độ an toàn phụ thuộc vào ứng dụng.

CHÚ THÍCH Xem Điều 6.1.

3.12

Cơ chế bọc dữ liệu (data encapsulation mechanism)

Cơ chế mật mã dựa trên kỹ thuật mật mã đối xứng, bảo vệ tính bí mật và tính toàn vẹn của dữ liệu

CHÚ THÍCH Xem Điều 8.2.

3.13

Giải mã (decryption)

Ngược với phép mã hóa tương ứng

[ISO/IEC 11770-1:1996]

3.14

Thuật toán giải mã (decryption algorithm)

Quá trình biến đổi bản mã thành bản rõ

[ISO/IEC 18033-1]

3.15

Mã hóa (encryption)

Phép biến đổi khả nghịch dữ liệu bằng thuật toán mật mã để tạo ra bản mã, nghĩa là che giấu nội dung thông tin của dữ liệu

[ISO/IEC 9797-1]

3.16

Trường hữu hạn cho dưới dạng tường minh (explicitly given finite field)

Trường hữu hạn được biểu diễn dưới dạng tường minh, theo thuật ngữ các đặc trưng của nó và một bảng phép nhân làm cơ sở trên một trường nguyên tố

CHÚ THÍCH Xem điều 5.3.

3.17

Thuật toán mã hóa (encryption algorithm)

Quá trình biến đổi bản rõ thành bản mã

[ISO/IEC 18033-1]

3.18

Tùy chọn mã hóa (encryption option)

Tùy chọn sao cho có thể được chuyển đến thuật toán mã hóa của hệ mật phi đối xứng, hoặc của cơ chế bọc khóa, để kiểm soát việc định dạng bản mã đầu ra

CHÚ THÍCH Xem điều 7.8.1.

3.19

Trường (field)

Khái niệm toán học của trường, tức là một tập hợp các phần tử cùng với các phép toán hai ngôi là phép cộng và phép nhân trên tập đó, sao cho thỏa mãn các tiên đề về trường

3.20

Nhóm abel hữu hạn (finite abelian group)

Nhóm với tính chất là tập hợp cơ sở gồm hữu hạn phần tử và phép toán hai ngôi cơ sở trên đó có tính giao hoán

3.21

Trường hữu hạn (finite group)

Trường với tập hợp cơ sở của nó là hữu hạn

3.22

Nhóm (group)

Khái niệm toán học về nhóm, tức tập hợp các phần tử với phép toán nhị phân trên tập đó thỏa mãn các tiên đề thông thường về nhóm

3.23

Hệ mật lai ghép (hybird cipher)

Mật mã phi đối xứng kết hợp cả kỹ thuật mật mã phi đối xứng và đối xứng

3.24

Khóa (key)

Dãy các kí hiệu điều khiển hoạt động của phép biến đổi mật mã (ví dụ, phép mã hóa, giải mã)

[ISO/IEC 11770-1:1996]

3.25

Hàm dẫn xuất khóa (key derivation function)

Hàm ánh xạ xâu bộ tám có độ dài bất kì thành xâu bộ tám có độ dài xác định, sao cho bằng tính toán không thể tìm ra mối quan hệ giữa đầu vào và đầu ra và sao cho nếu biết một phần đầu ra, nhưng không biết đầu vào, hoàn toàn không có khả năng bằng tính toán, đoán được bất kì bit nào của phần đầu ra còn lại.

CHÚ THÍCH Xem Điều 6.2.

3.26

Cơ chế bọc khóa (key encapsulation mechanism)

Tương tự mật mã phi đối xứng, nhưng thuật toán mã hóa nhận đầu vào là khóa công khai, tạo ra khóa bí mật và mã hóa khóa bí mật đó

CHÚ THÍCH Xem Điều 8.1.

3.27

Thuật toán tạo khóa (key generation algorithm)

Phương pháp tạo cặp khóa phi đối xứng

CHÚ THÍCH Xem Điều 7, 8.1.

3.28

Nhãn (label)

Xâu bộ tám là đầu vào của cả thuật toán mã hóa và giải mã trong hệ mật phi đối xứng, và đầu vào của cơ chế đóng gói dữ liệu. Nhãn là thông tin công khai được gắn vào bản mã theo một cách không thể thay đổi được

CHÚ THÍCH Xem Điều 7, 8.2.

3.29

Độ dài (length)

Độ dài của xâu các chữ số hay biểu diễn một số nguyên

Cụ thể:

(1) Độ dài của xâu bit là số bit của xâu

CHÚ THÍCH Xem Điều 5.2.1.

(2) Độ dài của xâu bộ tám là số lượng octet của xâu

CHÚ THÍCH Xem Điều 5.2.2.

(3) Độ dài của số nguyên không âm n là số bit trong biểu diễn nhị phân của n, nghĩa là $dlog_2(n+1)$.

CHÚ THÍCH Xem Điều 5.2.4.

(4) Độ dài octet của số nguyên không âm n là số lượng chữ số trong biểu diễn của n theo cơ số 256, tức $dlog_{256}(n+1)$.

CHÚ THÍCH Xem Điều 5.2.4.

3.30

Mã xác thực thông báo (message authentication code) (MAC)

Xâu bit là đầu ra của thuật toán MAC

[ISO/IEC 9797-1]

CHÚ THÍCH 1 Xem Điều 6.3.

CHÚ THÍCH 2 Trong phần này của ISO/IEC 18033, MAC được hạn chế là xâu bộ tám (được minh họa một cách tự nhiên như một xâu bit).

3.31

Thuật toán MAC (MAC algorithm)

Thuật toán tính hàm là ánh xạ các xâu bit và khóa bí mật vào các xâu bit có độ dài cố định, thỏa mãn hai tính chất sau:

- Với khóa và xâu đầu vào bất kì, có thể tính hàm một cách hiệu quả;

- Với khóa cố định bất kì và không biết trước khóa, bằng tính toán không có khả năng tính được giá trị hàm tại bất kì xâu đầu vào mới nào và không biết trước khóa không thể tính được giá trị hàm tại bất kì xâu đầu vào mới nào, thậm chí cho biết tập hợp các xâu đầu vào và giá trị tương ứng của hàm, trong đó giá trị của đầu vào thứ i có thể chọn sau khi quan sát $i-1$ giá trị đầu vào của hàm.

[ISO/IEC 9797-1]

CHÚ THÍCH 1 Xem Điều 6.3.

CHÚ THÍCH 2 Trong phần này của ISO/IEC 18033, xâu đầu và đầu ra chỉ hạn chế là những xâu bộ tám (được minh họa một cách tự nhiên như những xâu bit).

3.32

Bộ tám (octet)

Xâu bit có độ dài bằng 8

CHÚ THÍCH Xem Điều 5.2.2.

3.33

Xâu bộ tám (octet string)

Dãy các octet

CHÚ THÍCH Xem Điều 5.2.2.

CHÚ THÍCH Khi cần, xâu bộ tám có thể được minh họa như một xâu bit bằng cách ghép tất cả các octet thành phần.

3.34

Bản rõ (plaintext)

Thông tin chưa được mã hóa

[ISO/IEC 10116:1997]

3.35

Tập hợp phi tiền tố (prefix free set)

Tập hợp S các xâu bit/octet sao cho không tồn tại các xâu $x \neq y \in S$ sao cho x là tiền tố của y

3.36

Hàm nguyên thủy [nguyên thủy] (primitive)

Hàm dùng để biến đổi lẫn nhau giữa các dạng dữ liệu

3.37

Khóa bí mật (private key)

Khóa thuộc cặp khóa phi đối xứng của thực thể chỉ được sử dụng bởi thực thể

[ISO/IEC 11770-1:1996]

CHÚ THÍCH xem Điều 7, 8.1.

3.38

Khóa công khai (public key)

Khóa thuộc cặp khóa phi đối xứng có thể được công khai

[ISO/IEC 11770-1:1996]

CHÚ THÍCH Xem Điều 7, 8.1.

3.39

Khóa bí mật (secret key)

Khóa được sử dụng trong kỹ thuật mật mã đối xứng bởi một tập hợp xác định các thực thể

[ISO/IEC 11770-3:1999]

3.40

Mật mã đối xứng (symmetric cipher)

Mật mã dựa trên kỹ thuật mật mã đối xứng, sử dụng cùng một khóa bí mật cho cả thuật toán mã hóa và giải mã

[ISO/IEC 18033-1]

3.41

Các tham số hệ thống (system parameters)

Sự lựa chọn tham số là chọn một lược đồ hoặc một hàm mật mã cụ thể từ một họ các lược đồ hoặc các hàm mật mã

4 Ký hiệu và khái niệm

Trong tiêu chuẩn này áp dụng các ký hiệu và khái niệm dưới đây.

CHÚ THÍCH Trong tiêu chuẩn này, ở những nơi phù hợp các tham chiếu tiếp theo được đưa ra trong các điều mà chứa những định nghĩa và/hoặc soạn thảo chi tiết hơn.

$\lfloor x \rfloor$ Số nguyên lớn nhất nhỏ hơn hoặc bằng số thực x. Ví dụ:
$\lfloor 5 \rfloor = 5, \lfloor 5.3 \rfloor = 5,$ và $\lfloor -5.3 \rfloor = -6$

$\lceil x \rceil$ Số nguyên nhỏ nhất lớn hơn hoặc bằng số thực x. Ví dụ:
$\lceil 5 \rceil = 5, \lceil 5.3 \rceil = 6$ và $\lceil -5.3 \rceil = -6$

$[a..b]$ Khoảng các số nguyên từ a đến b, tính cả a và b

$[a..b)$ Khoảng số nguyên từ a đến b, tính cả a nhưng không tính b

$|X|$ Là lực lượng của X, nếu X là tập hợp hữu hạn; là tập hợp phần tử cơ sở nếu X là nhóm abel hoặc trường hữu hạn; là giá trị tuyệt đối nếu X là số thực; là độ dài của xâu bit hoặc xâu bộ tám nếu X là xâu bit hay xâu bộ tám

CHÚ THÍCH Xem Điều 5.2.1; 5.2.2.

$x \oplus y$ Phép cộng XOR, nếu x và y là xâu bit/octet có cùng độ dài

CHÚ THÍCH Xem Điều 5.2.1, 5.2.2.

$\langle x_1, ..., x_l \rangle$ Xâu bit/octet độ dài l, ở đó $x_1, x_2, ..., x_l$ là bit/octet

CHÚ THÍCH Xem Điều 5.2.1, 5.2.2.

$x \| y$ Phép ghép hai xâu bit/octet x và y, kết quả được xâu với phần đầu là x, phần kế sau là y

CHÚ THÍCH Xem Điều 5.2.1 và 5.2.2.

gcd(a,b) Ước số chung lớn nhất của hai số nguyên a và b, nghĩa là số nguyên dương lớn nhất chia hết cho cả a và b (hoặc 0 nếu a = b = 0)

$a \mid b$ Quan hệ giữa các số nguyên a và b sao cho a chia hết b, tức tồn tại số c để $b = ac$

$a \equiv b (\mathrm{mod}\, n)$ Quan hệ giữa hai số nguyên a và b sao cho a và b đồng dư theo modulo n, nghĩa là $n \mid a - b$., trong đó n là số nguyên khác 0

$a \bmod n$ Số nguyên duy nhất $r \in [0, ..., n)$ sao cho $r \equiv a (\mathrm{mod}\, n)$

$a^{-1} \bmod n$	Số nguyên duy nhất b $b \in [0,...,n)$ sao cho $ab = 1 (\bmod n)$, trong đó a là số nguyên, n số nguyên dương
F^*	Nhóm nhân các đơn vị của F, trong đó F là trường hữu hạn
0_F	Phần tử đồng nhất theo phép cộng (phần tử 0) của trường F.
1_F	Phần tử đồng nhất theo phép nhân của trường F.
BS2IP	Nguyên thủy biến đổi xâu bit thành số nguyên CHÚ THÍCH Xem Điều 5.2.5.
EC2OSP	Nguyên thủy biến đổi đường cong elliptic thành xâu bộ tám *(xem Điều 5.4.3)*
FE2OSP	Nguyên thủy biến đổi phần tử trường thành xâu bộ tám *(xem Điều 5.3.1.)*
FE2IP	Nguyên thủy biến đổi phần tử trường thành số nguyên *(xem Điều 5.3.1.)*
I2BSP	Nguyên thủy biến đổi số nguyên thành xâu bit *(xem Điều 5.2.5.)*
I2OSP	Nguyên thủy biến đổi số nguyên thành xâu bộ tám *(xem Điều 5.2.5.)*
OS2ECP	Nguyên thủy biến đổi xâu bộ tám thành đường cong elliptic *(xem Điều 5.4.3.)*
OS2FEP	Nguyên thủy biến đổi xâu bộ tám thành phần tử trường *(xem Điều 5.3.1.)*
OS2IP	Nguyên thủy biến đổi xâu bộ tám thành số nguyên *(xem Điều 5.2.5.)*
Oct(m)	Bộ tám có giá trị nguyên bằng m *(xem Điều 5.2.4.)*
E(n)	Độ dài của số nguyên n tính bằng octet *(xem Điều 5.2.5.)*

5 Cơ sở toán học

Điều này mô tả cơ sở toán học được sử dụng trong phần này của ISO/IEC 18033, bao gồm việc trình bày các đối tượng toán học và các nguyên thủy cho biến đổi các dạng dữ liệu.

5.1 Hàm và thuật toán biến đổi

Để tiện cho trình bày, các hàm và các hàm ngẫu nhiên (tức những hàm mà giá trị của nó không chỉ phụ thuộc vào giá trị đầu vào, mà còn vào việc chọn giá trị ngẫu nhiên bổ trợ), thường được xác định ở dạng thuật toán. Ngoại trừ những trường hợp sẽ được nói rõ, người thực thi có thể chọn để khai thác bất kì thuật toán tương đương nào (nghĩa là, chọn thuật toán dẫn đến cùng một hàm hoặc hàm ngẫu nhiên). Ngoài ra trong trường hợp hàm ngẫu nhiên, khi mà thuật toán mô tả hàm chỉ ra giá trị ngẫu nhiên, người thực thi có thể sử dụng bộ tạo ngẫu nhiên tương ứng để sinh ra giá trị đó (xem ISO/IEC 18031 với hướng dẫn chi tiết hơn về vấn đề này).

Trong mô tả hàm bằng ngôn ngữ thuật toán, quy ước sau đây được chấp nhận. Một thuật toán có thể tính toán cho ra được giá trị, hoặc ngược lại, có thể **thất bại** (tức không tính được giá trị). Theo quy

ước, nếu thuật toán **thất bại**, thì trừ khi có ghi chú thêm, một thuật toán khác, sử dụng thuật toán này như một chương trình con, cũng sẽ **thất bại**.

CHÚ THÍCH Như vậy khái niệm **thất bại** cũng tương tự như khái niệm "đưa ra ngoại lệ" (throwing an exception) trong nhiều ngôn ngữ lập trình. Tuy nhiên điều này cũng có thể coi như thuật toán trả về một giá trị đặc biệt, khác với tất cả các giá trị khác được thuật toán đưa ra khi nó không **thất bại**. Với cách cắt nghĩa như vậy về **thất bại**, thuật toán vẫn được coi là mô tả hàm. Chi tiết về việc, làm thế nào để thực thi hiệu quả khi thuật toán thất bại, không trình bày ở đây. Tuy nhiên, trong các thực thi thông thường, thuật toán có thể trả về một số dạng "mã lỗi" cho môi trường thực thi của nó và như vậy nó chỉ ra nguyên nhân thất bại. Cũng cần lưu ý rằng trong một số trường hợp, vì lí do bảo mật, trong khi thực thi cần quan tâm để không để lộ lí do chính xác của một số dạng lỗi nhất định.

5.2 Xâu bit và xâu bộ tám

5.2.1 Bit và xâu bit

Bit là một trong hai kí hiệu '0' và '1'.

Xâu bit là một dãy bit. Cho các bit $x_1,...,x_l$, $\langle x_1,...,x_l\rangle$ là kí hiệu xâu bit độ dài l, gồm các bit $x_1,...,x_l$ xếp theo thứ tự đã cho.

Với xâu bit $x=\langle x_1,...,x_l\rangle$, độ dài l của x được kí hiệu là $|x|$, và nếu $l > 0$ thì x_1 được gọi là bit *đầu*, x_l được gọi là bit *cuối* của x.

Với hai xâu bit x và y, $x\|y$ kí hiệu kết quả ghép x và y, bởi vậy nếu $x=\langle x_1,...,x_l\rangle$, $y=\langle y_1,...,y_m\rangle$ thì $x\|y=\langle x_1,...,x_l,y_1,...,y_m\rangle$.

Nếu x và y là hai xâu bit có cùng độ dài l, thì $x\oplus y$ kí hiệu phép cộng bit XOR của x và y.

Xâu bit có độ dài bằng 0 được gọi là xâu bit rỗng.

CHÚ THÍCH Không có một toán tử nào được định nghĩa dành riêng cho xâu bit, nên nếu x là xâu bit, thì x_i không nhất thiết kí hiệu một bit cụ thể của x.

5.2.2 Bộ tám (octet) và xâu bộ tám (octet)

Bộ tám là xâu bit có độ dài bằng 8.

Xâu bộ tám là dãy các bộ tám.

Cho các octet $x_1,...,x_l$, thì $\langle x_1,...,x_l\rangle$ là kí hiệu xâu bộ tám độ dài l gồm các bộ tám $x_1,...,x_l$, theo thứ tự đã cho.

Cho xâu bộ tám x = $\langle x_1,...,x_l\rangle$, độ dài l của x được kí hiệu là $|x|$, nếu $l > 0$ thì x_1 được gọi là bộ tám *đầu*, x_l được gọi là bộ tám *cuối* của x.

Với hai xâu bộ tám x và y, $x\|y$ là kí hiệu kết quả ghép x và y, bởi vậy nếu $x=\langle x_1,...,x_l\rangle$, $y=\langle y_1,...,y_l\rangle$ thì $x\|y=\langle x_1,...,x_l,y_1,...,y_m\rangle$.

Nếu x và y là hai xâu bộ tám có cùng độ dài, thì $x\oplus y$ kí hiệu phép cộng bit XOR của x và y.

Xâu bộ tám có độ dài bằng 0 được gọi là xâu bộ tám rỗng.

CHÚ THÍCH 1 Không có toán tử nào được xác định dành riêng cho tập các xâu octet. Do đó nếu x là xâu bộ tám, thì x_i không nhất thiết kí hiệu một bộ tám cụ thể của x.

CHÚ THÍCH 2 Để ý rằng, vì bộ tám là xâu bit có độ dài bằng 8, nên nếu x và y là các bộ tám, thì $x \| y$ có độ dài bằng 16, nếu $\langle x \rangle$ và $\langle y \rangle$ mỗi *xâu* có độ dài bằng 1 thì $\langle x \rangle \| \langle y \rangle = \langle x, y \rangle$ là xâu bộ tám độ dài bằng 2.

5.2.3 Chuyển đổi giữa xâu bit và xâu bộ tám

Các nguyên thủy $OS2BSP$ và $BS2OSP$ chuyển đổi giữa các xâu bit và xâu bộ tám được định nghĩ như sau :

Hàm $OS2BSP(x)$: đầu vào là xâu bộ tám $x = \langle x_1, ..., x_l \rangle$ và đầu ra là xâu bit $y = x_1 \| x_2 ... \| x_l$.

Hàm $BS2OSP(y)$: đầu vào là xâu bit y với độ dài là bội số của 8, đầu ra là xâu bộ tám duy nhất x sao cho $y = OS2BSP(x)$.

5.2.4 Chuyển đổi giữa xâu bit và số nguyên

Các nguyên thủy $BS2IP$ và $I2BSP$ thực hiện chuyển đổi giữa xâu bit và số nguyên được định nghĩa như sau.

Hàm $BS2IP\ (x)$ ánh xạ xâu bit x vào giá trị nguyên x', như sau. Nếu $x = \langle x_{l-1}, ..., x_0 \rangle$, ở đây $x_0, ..., x_{l-1}$ là các bit, khi đó giá trị x' bằng

$$x' = \sum_{\substack{0 \le i < l \\ x_i = \text{'1'}}} 2^i$$

Hàm $I2BSP(m, l)$ nhận đầu vào là hai số nguyên không âm m và l, đầu ra là xâu bit duy nhất x độ dài l sao cho $BS2IP(x) = m$, nếu tồn tại x như vậy. Ngược lại, hàm **thất bại**.

Độ dài tính bằng bit của số nguyên không âm n là số bit trong biểu diễn nhị phân của nó, tức

$\lceil \log 2(n + 1) \rceil$.

Để tiện kí hiệu, đặt $Oct(m) = I2BSP(m, 8)$.

CHÚ THÍCH Lưu ý rằng $I2BSP(m, l)$ **thất bại** khi và chỉ khi độ dài của m tính bằng bit lớn hơn l.

5.2.5 Chuyển đổi giữa xâu bộ tám và số nguyên

Các phép biến đổi nguyên thủy $OS2IP$ và $I2OSP$ chuyển đổi giữa xâu bộ tám và số nguyên được xác định như sau.

Hàm $OS2IP(x)$: đầu vào là xâu bộ tám, đầu ra là số nguyên $BS2IP(OS2BSP(x))$.

Hàm $I2OSP(m, l)$: đầu vào là hai số nguyên không âm m, l, đầu ra là xâu bộ tám (octet) duy nhất x, độ dài l, thỏa mãn hệ thức $OS2IP(x) = m$, nếu số nguyên x như vậy tồn tại. Ngược lại, hàm **thất bại**.

Độ dài tính bằng octet của số nguyên không âm n là số chữ số trong biểu diễn cơ số 256, nghĩa là $\lceil log_{256}(n + 1) \rceil$; đại lượng này được kí hiệu là $\mathcal{L}(n)$.

CHÚ THÍCH Chú ý rằng $I2OSP(m, l)$ **thất bại** khi và chỉ khi độ dài của m tính bằng bộ tám (octet) lớn hơn l.

5.3 Trường hữu hạn

Điều này mô tả một cấu trúc rất khái quát để mô tả các trường hữu hạn đặc biệt. Trường hữu hạn được xác định bằng cách này được gọi là *trường hữu hạn được cho dưới dạng tường minh*, và được xác định bởi dữ liệu hiện.

Với trường hữu hạn F có lực lượng $q = p^e$, ở đây p là số nguyên tố và $e \geq 1$, dữ liệu hiện của F bao gồm p và e, cùng với "bảng nhân" là ma trận $T = (T_{ij})_{1 \leq i,j \leq e}$, trong đó mỗi T_{ij} là bộ e phần tử (e-tuple) từ tập hợp $[0,\ldots,p)$.

Tập hợp các phần tử của trường F là tập hợp tất cả các bộ e phần tử trên $[0,\ldots,p)$. Đầu vào của T được coi như các phần tử của F.

Phép cộng trong F được định nghĩa như sau:

Nếu

$$a = (a_1, \ldots, a_e) \in F \text{ và } b = (b_1, \ldots, b_e) \in F,$$

Khi đó $a + b = c$, *với*

$$c = (c_1, \ldots, c_e) \text{ và } c_i = (a_i + b_i) \bmod p \ (1 \leq i \leq e).$$

Phép nhân vô hướng trong F (nhân một phần tử của F với một số thuộc $[0,\ldots,p)$) như sau:

Nếu

$$a = (a_1, \ldots, a_e) \in F \text{ và } d \in [0 \ldots p),$$

Khi đó $d.a = c$ với

$$c = (c_1, \ldots, c_e) \text{ và } c_i = (d.a_i) \bmod p \ (1 \leq i \leq e).$$

Phép nhân trên trường F được xác định thông qua bảng nhân T, như sau:

Nếu

$$a = (a_1, \ldots, a_e) \in F \text{ và } b = (b_1, \ldots, b_e) \in F,$$

thì $$a.b = \sum_{i=1}^{e} \sum_{j=1}^{e} (a_i b_j \bmod p) T_{ij},$$

Trong công thức trên cách tích $(a_i\, b_i \bmod p)T_{ij}$ được xác định theo qui tắc tích vô hướng đã nêu trên, và chúng được cộng lại theo qui tắc phép cộng trong F. Giả thiết là bảng nhân xác định một cấu trúc đại số thỏa mãn các tiên đề thông thường của trường; nói riêng, tồn tại các phần tử đồng nhất đối với phép cộng và phép nhân (tức phần tử trung hòa và đơn vị), mỗi phần tử của trường có phần tử nghịch đảo đối với phép cộng (phần tử đối) và phần tử nghịch đảo đối với phép nhân.

Nhận thấy rằng phần tử đồng nhất đối với phép cộng của F, kí hiệu là 0_F, chính là bộ e phần tử gồm các phần tử 0 và phần tử đồng nhất đối với phép nhân được kí hiệu là 1_F, là bộ e phần tử khác không với định dạng chính xác của 1_F phụ thuộc vào T.

CHÚ THÍCH 1 Trường F là không gian véc tơ có số chiều là e, xác định trên trường nguyên tố F', có lực lượng bằng p, với phép nhân vô hướng đã được định nghĩa trên đây. Số nguyên tố p được gọi là đặc số của trường F. Với $1 \leq i \leq e$, kí hiệu θ_i là bộ e-phần tử (e-tuple) trên F' sao cho thành phần thứ i của nó là 1 và tất cả thành phần còn lại bằng 0. Các phần tử $\theta_1, \ldots, \theta_e$ tạo thành cơ sở của không gian véc tơ F trên F'. Lưu ý rằng với mọi $1 \leq i, j \leq e$, ta có $\theta_i \theta_j = T_{ij}$.

CHÚ THÍCH 2 Với $e > 1$, có hai kiểu cơ sở cùng được sử dụng trong thực thi các phép toán số học trên trường hữu hạn:

– $\theta_1, ..., \theta_e$ được gọi là cơ sở đa thức cho F trên F', nếu tồn tại θ thuộc F sao cho $\theta_i = \theta^{e-1}$ với mọi $1 \le i \le e$. Rõ ràng ta có $1_F = \theta_e$.

– $\theta_1, ..., \theta_e$ được gọi cơ sở chuẩn tắc cho F trên F', nếu tồn tại θ thuộc F sao cho $\theta_i = \theta^{p^{i-1}}$ với mọi $1 \le i \le e$. Rõ ràng trong trường hợp này, ta có $1_F = c\sum_{i=1}^{e} \theta_i$ với $c \in [1, ..., p)$; Nếu p = 2, khi đó chỉ có một lựa chọn duy nhất cho c là 1; hơn nữa luôn luôn có thể chọn cơ sở chuẩn tắc với c = 1.

CHÚ THÍCH 3 Định nghĩa trường hữu hạn dưới dạng tường minh được trích dẫn từ [23].

5.3.1 Chuyển đổi giữa xâu bộ tám và số nguyên/trường hữu hạn

Các phép biến đổi nguyên thủy $OS2FEP_F$ và $FE2OSP_F$ giữa xâu bộ tám và các phần tử của trường hữu hạn được cho dưới dạng tường minh F, cũng như phép biến đổi nguyên thủy $FE2IP_F$ chuyển đổi các phần tử của F thành số nguyên, được định nghĩa như sau.

Hàm $FE2IP_F$ ánh xạ phần tử $a \in F$ vào giá trị nguyên a', như sau. Nếu lực lượng của F là $q = p^e$, ở đây p là số nguyên tố và $e \ge 1$, khi đó phần tử a của trường F là bộ e phần tử $(a_1, ..., a_e), a_i \in [0, ..., p), 1 \le i \le e,$ và giá trị a' được xác định theo công thức:

$$a' = \sum_{i=1}^{e} a_i p^{i-1}$$

Hàm $FE2OSP_F(a)$ nhận đầu vào là phần tử a của trường, đầu ra là xâu bộ tám $I2OSP(a', l)$, $a' = FE2OSP_F(a)$ và l là độ dài tính bằng bộ tám của $|F|-1$, nghĩa là $l = \lceil \log_{256} |F| \rceil$. Như vậy đầu ra của $FE2OSP_F(a)$ luôn là xâu bộ tám có độ dài chính xác bằng $l = \lceil \log_{256} |F| \rceil$.

Hàm $OS2FEP_F(x)$ nhận đầu vào là xâu bộ tám x, đầu ra (duy nhất) là phần tử trường $a \in F$ sao cho $FE2OSP_F(a) = x$, nếu tồn tại a như thế, và ngược lại thì thất bại. Lưu ý rằng $OS2FEP_F(x)$ thất bại khi và chỉ khi hoặc x không có độ dài đúng bằng $l = \lceil \log_{256} |F| \rceil$, hoặc $OS2IP(x) \ge |F|$.

5.4 Đường cong elliptic

Đường cong elliptic trên trường hữu hạn dạng tường minh là tập hợp các điểm $P = (x, y)$, ở đây x và y là các phần tử của trường F, thỏa mãn phương trình xác định, với "điểm tại vô cùng" được kí hiệu là O. Trong phần này của tiêu chuẩn ISO/IEC- 18033, đường cong elliptic E được xác định bởi hai phần phần tử của trường $a, b \in F$, các phần tử này được gọi là các *hệ số* của E.

Giả sử P là đặc trưng của trường F.

Nếu $p > 3$ thì a và b thỏa mãn điều kiện $4a^3 + 27b^2 \ne 0_F$, và mỗi điểm $P = (x, y)$ trên E (khác phần tử O) thỏa mãn phương trình

$$y^2 = x^3 + ax + b.$$

Nếu p = 2, thì a và b thỏa mãn điều kiện $b \ne 0_F$, và mỗi điểm $P = (x, y)$ trên E (khác phần tử O) thỏa mãn phương trình

$$y^2 + xy = x^3 + ax^2 + b.$$

Nếu $p = 3$, thì a và b thỏa mãn điều kiện $a \neq 0_F$ và $b \neq 0_F$, và mỗi điểm $P = (x, y)$ trên E (khác phần tử O) thỏa mãn phương trình

$$y^3 = x^3 + ax^2 + b.$$

Các điểm nằm trên đường cong elliptic tạo thành nhóm abel, với phần tử không O của E là phần tử trung hòa. Tồn tại những thuật toán hiệu quả để thực thi các phép toán nhóm của đường cong elliptic, tuy nhiên việc thực thi các thuật toán này nằm ngoài phạm vi của phần này của ISO/IEC 18033.

CHÚ THÍCH Xem ISO/IEC 15946-1, cũng như [9], để biết thêm thông tin về cách thực thi hiệu quả phép toán nhóm trên đường cong elliptic.

5.4.1 Các điểm nén của đường cong elliptic

Giả sử E là đường cong elliptic trên trường hữu hạn dạng tường minh F, ở đây F có đặc trưng bằng p.

Điểm $P \neq O$ có thể được biểu diễn dưới *dạng nén*, *không nén* và *hoặc lai ghép*.

Nếu $P = (x, y)$, thì (x, y) là dạng không nén của P.

Giả sử $P = (x, y)$ là điểm trên đường cong elliptic E. *Dạng nén* của P là cặp $(x, \tilde{y})$, ở đây $\tilde{y} \in \{0,1\}$ được xác định như sau:

– Nếu $p \neq 2$ và $y = 0_F$, thì $\tilde{y} = 0$.

– Nếu $p \neq 2$ và $y \neq 0_F$ thì $\tilde{y} = ((y' / p^f) \bmod p) \bmod 2$, ở đây $y' = FE2IP_F(y)$ và f là số nguyên không âm lớn nhất sao cho $p^f \mid y'$.

– Nếu $p = 2$ và $x = 0_F$ thì $\tilde{y} = 0$.

– Nếu $p = 2$ và $x \neq 0_F$ thì $\tilde{y} = \left\lfloor z' / 2^f \right\rfloor \bmod 2$, ở đây $z = y/x$ và $z' = FE2IP_F(y)$ và f là số nguyên không âm nhỏ nhất thỏa mã điều kiện 2^f chia hết $FE2IP_F(1_F)$.

Dạng lai ghép của $P = (x, y)$ là bộ ba $(x, \tilde{y}, y)$ với $\tilde{y}$ được xác định như ở Điều trước.

5.4.2 Các thuật toán giải nén điểm

Tồn tại các thuật toán hiệu quả giải nén điểm, tức là tính $\tilde{y}$ từ $(x, \tilde{y})$. Dưới đây sẽ mô tả vắn tắt các thuật toán đó.

– Giả sử $p \neq 2$ và $(x, \tilde{y})$ là dạng nén của (x, y). Điểm (x, y) thỏa mãn phương trình $y^2 = f(x)$ với $f(x)$ là đa thức trên F. Nếu $f(x) = 0_F$, thì chỉ có một khả năng chọn y, chính là $y = 0_F$. Ngược lại, nếu $f(x) \neq 0$ thì có hai khả năng chọn y, các khả năng này chỉ khác nhau về dấu, việc chọn đúng được xác định bởi $\tilde{y}$. Có hai thuật toán tính căn bậc hai trên trường hữu hạn được biết đến nhiều, và do đó hai lựa chọn y dễ dàng tính được.

– Giả sử $p = 2$ và $(x, \tilde{y})$ là dạng nén của (x, y). Các điểm (x, y) thỏa mãn phương trình $y^2 + xy = x^3 + ax^2 + b$. Nếu $x = 0_F$, khi đó $y^2 = b$, từ đây y được tính toán dễ dàng và duy nhất. Ngược lại nếu $x \neq 0_F$ khi đó đặt $z = y/x$, ta có $z^2 + z = g(x)$ ở đây $g(x) = (x + a + bx^{-2})$. Giá trị của y được xác định một cách duy nhất và dễ dàng tính ra được từ các giá trị z và x, do đó chỉ cần tính z. Để tính z, ta lưu ý rằng với x cố định, nếu z là một trong các nghiệm của phương trình $z^2 + z = g(x)$, khi đó có đúng một nghiệm khác, chính là $z + 1_F$. Dễ dàng tính được hai giá trị ứng cử viên của z và dễ thấy rằng việc chọn đúng z được xác định bởi $\tilde{y}$.

5.4.3 Chuyển đổi giữa xâu bộ tám và đường cong elliptic

Các nguyên thủy $EC2OSP_E$ và $OS2ECP_E$ chuyển đổi giữa các điểm trên đường cong elliptic và xâu bộ tám được xác định như sau.

Giả sử E là đường cong elliptic trên trường hữu hạn được cho ở dạng hiện F.

Hàm $EC2OSP_E(P, fmt)$ nhận đầu vào là điểm P trên E và bộ định dạng fmt là một trong các giá trị kí hiệu được nén, không được nén, hoặc lai ghép. Đầu ra là xâu bộ tám EP, được tính như sau:

– Nếu $P = O$ thì $EP = \langle Oct(0) \rangle$.

– Nếu $P = (x, y) \neq O$ với dạng nén $(x, \tilde{y})$, khi đó

$$EP = \langle H \rangle \parallel X \parallel Y,$$

ở đây,

– H là bộ tám đơn có dạng $Oct(4U + C.(2 + \tilde{y}))$, trong đó,

  – $U = 1$ nếu fmt hoặc là dạng không nén hoặc dạng lai ghép, ngược lại $U = 0$;

  – $C = 1$ nếu fmt hoặc dạng nén hoặc dạng lai ghép, ngược lại $C = 0$;

– X là xâu bộ tám $FE2OSP_F(x)$;

– Y là xâu bộ tám $FE2OSP_F(y)$, nếu fmt hoặc dạng không nén hoặc dạng lai ghép; ngược lại y là xâu bộ tám rỗng.

CHÚ THÍCH Nếu (biến) định dạng fmt không ở dạng nén, thì không cần tính giá trị $\tilde{y}$.

Hàm $OS2ECP_E(EP)$ nhận đầu vào là xâu bộ tám EP. Nếu tồn tại điểm P trên đường cong elliptic E và (biến) định định dạng fmt thỏa mãn $EC2OSP_E(P, fmt) = EP$, khi đó đầu ra của hàm là P (ở dạng không nén), ngược lại-hàm thất bại. Lưu ý rằng điểm P, nếu tồn tại, được xác định duy nhất và do đó hàm $OS2ECP_E(EP)$ đã được xác định rõ.

6 Các biến đổi mật mã

Điều này mô tả một số phép biến đổi mật mã sẽ được tham chiếu ở các Điều tiếp theo. Các dạng biến đổi đó là: *hàm băm mật mã, hàm dẫn xuất khóa, mã xác thực thông báo, mã khối,* và *mã đối xứng*. Với mỗi dạng biến đổi sẽ đưa ra các đặc trưng đầu ra/đầu vào ngắn gọn, tiếp đó sẽ đặc tả các thực thi cụ thể các phép biến đổi này, chúng được phép sử dụng trong phần này của ISO/IEC18003.

6.1 Các hàm băm mật mã

Hàm băm mật mã là hàm ánh xạ các xâu bộ tám có độ dài thay đổi vào xâu bộ tám có độ dài cố định. Chính xác hơn, hàm băm (hash function) mật mã xác định

– số nguyên dương $Hash.Len$ kí hiệu độ dài đầu ra của hàm băm,

– số nguyên dương $Hash.MaxInputLen$ kí hiệu độ dài lớn nhất của đầu vào hàm băm,

– và hàm $Hash.eval$ kí hiệu bản thân hàm băm, ánh xạ xâu bộ tám có độ dài lớn nhất $Hash.MaxInputLen$ vào xâu bộ tám độ dài $Hash.Len$.

Việc gọi hàm $Hash.eval$ thất bại khi và chỉ khi độ dài đầu ra vượt quá giá trị $Hash.MaxInputLen$.

6.1.1 Các hàm băm mật mã cho phép

Với mục đích của phần này của ISO/IEC 18033, các hàm băm mật mã là những hàm được mô tả trong ISO/IEC 10118-2 và ISO/IEC 10118-3 với những qui định sau đây:

– Hàm băm trong ISO/IEC 10118 ánh xạ xâu bit vào xâu bit, trong khi đó trong ISO/IEC 18033 hàm băm ánh xạ xâu bộ tám vào xâu bộ tám. Do đó hàm băm trong ISO/IEC 10118-2 và ISO/IEC 10118-3 được phép áp dụng vào phần này của tiêu chuẩn ISO/IEC 18033, chỉ khi độ dài tính bằng bit của đầu ra là bội số của 8, trong trường hợp này ánh xạ giữa các xâu bộ tám và xâu bit bị tác động bởi các hàm *OS2BSP* và *BS2OSP*.

– Ngược lại, nếu như hàm băm trong ISO/IEC 10118 không xác định đối với đầu vào vượt quá độ dài cho trước, thì hàm băm trong phần này của ISO/IEC 18033 được xác định là **thất bại** đối với những đầu vào như vậy.

6.2 Hàm dẫn xuất khóa

Hàm *dẫn xuất khóa* là hàm $KDF(x, l)$, nhận đầu vào là xâu bộ tám x và số nguyên $l \geq 0$, và đầu ra là xâu bộ tám độ dài l. Xâu x có độ dài bất kì, mặc dù trong khi thực thi có thể xác định độ dài cực đại (rất lớn) cho x là kích thước cực đại cho l và **thất bại** nếu các giới hạn này bị vượt qua.

CHÚ THÍCH Trong một số tài liệu và tiêu chuẩn khác, thuật ngữ "hàm tạo mặt nạ" ("mask generation function") được sử dụng thay cho "hàm dẫn xuất khóa".

6.2.1 Hàm dẫn xuất khóa được phép

Các hàm dẫn xuất khóa được phép trong phần này của ISO/IEC 18033 là *KDF1* được mô tả phía dưới, ở Điều 6.2.2 và *KDF2* được mô tả ở Điều 6.2.3.

6.2.2 *KDF1*

6.2.2.1 Các tham số hệ thống

KDF1 là họ các hàm dẫn xuất khóa được tham số hóa bởi các tham số hệ thống sau đây:

– *Hash*: hàm băm mật mã được mô tả ở Điều 6.1.

6.2.2.2 Đặc tả

Với xâu bộ tám x và số nguyên không âm l, $KDF1(x, l)$ được xác định cho l bộ tám đầu tiên của xâu

$Hash.eval(x \| I2OSP(0,4) \| ... \| Hash.eval(x \| I2OSP(k-1,4),$

ở đây $k = \lceil l / Hashlen \rceil$.

CHÚ THÍCH Hàm này sẽ **thất bại** khi và chỉ khi $k > 2^{32}$ hoặc khi $|x| + 4 > Hash.MaxInputLen$.

6.2.3 *KDF2*

6.2.3.1 Các tham số hệ thống

KDF2 là họ các hàm dẫn xuất khóa, được tham số bởi các tham số hệ thống sau:

— *Hash*: hàm băm mật mã được mô tả tại Điều 6.1.

6.2.3.2 Đặc tả

Với xâu bộ tám x và số nguyên không âm l, $KDF2(x,l)$ được xác định như là bộ tám đầu tiên của xâu

$$Hash.eval(x\|I2OSP(1,4)\|...\|Hash.eval(x\|I2OSP(k,4)).$$

ở đây $k=\lceil l/Hashlen\rceil$.

CHÚ THÍCH 1 Hàm này sẽ **thất bại** khi và chỉ khi $k>2^{32}$ hoặc $|x|+4>Hash.MaxInputLen$.

CHÚ THÍCH 2 *KDF2* cũng giống như *KDF1*, trừ việc tính từ 1 đến k thay vì từ 0 đến $k-1$.

6.3 Các thuật toán MAC

Thuật toán xác thực thông báo MA là lược đồ xác định hai số nguyên dương $MA.KeyLen$, $MA.MACLen$, cùng hàm $MA.eval(k',T)$, hàm này nhận khóa mật k' là xâu bộ tám độ dài bằng $MA.KeyLen$, và xâu bộ tám tùy ý T làm đầu vào và tính đầu ra là xâu bộ tám MAC độ dài $MA.MACLen$.

Việc thực thi có thể hạn chế giá trị cực đại cho độ dài của T và $MA.eval(k',T)$ sẽ **thất bại** nếu giới hạn này bị vượt qua.

CHÚ THÍCH Tham khảo Phụ lục B.1 xem thảo luận về các tính chất bảo mật mong muốn của các thuật toán MAC.

6.3.1 Các thuật toán MAC được phép

Với mục đích của phần này của ISO/IEC 18033, các thuật toán MAC được phép được mô tả trong ISO/IEC 9797-1 và ISO/IEC 9797-2, với các qui định sau:

— Đối với các thuật toán MAC được mô tả trong ISO/IEC 9797-1 và ISO/IEC 9797-2, đầu ra là các xâu bit, khoá bí mật và đầu ra là những xâu có độ dài cố định. Bởi vậy, một thuật toán trong ISO/IEC 9797-1 hoặc ISO/IEC 9797-2 được phép áp dụng trong phần này của ISO/IEC 18033 chỉ khi độ dài tính bằng bit của MAC và khoá mật là bội số của 8, trong trường hợp này ánh xạ giữa các xâu bộ tám và các xâu bit bị ảnh hưởng bởi $OS2BSP$ và $BS2OSP$.

— Nếu các thuật toán trong ISO/IEC 9797-1 và ISO/IEC 9797-2 không xác định đối với đầu vào vượt quá độ dài cho trước, thì thuật toán MAC trong phần này của ISO/IEC 18033, được xác định là **thất bại** đối với những đầu vào như vậy.

6.4 Mã khối

Mã khối BC đặc tả như sau:

— số nguyên dương $BC.KeyLen$, là độ dài của khóa mật tính bằng octet,

— số nguyên dương $BC.BlockLen$, là độ dài tính bằng octet của khối bản mã hoặc bản rõ,

— hàm $BC.Encrypt(k,b)$, nhận đầu vào là khóa bí mật k,với k là xâu bộ tám, độ dài $BC.KeyLen$ và khối bản rõ b là xâu bộ tám độ dài $BC.BlockLen$, và đầu ra là khối mã b' với b' là xâu bộ tám độ dài $BC.BlockLen$, và

— hàm $BC.Decrypt(k,b')$, nhận đầu vào là khóa bí mật là xâu bộ tám, độ dài $BC.KeyLen$, và khối bản mã b' là xâu bộ tám độ dài $BC.BlockLen$, và đầu ra là khối bản rõ là xâu bộ tám, độ dài $BC.BlockLen$.

Với khóa bí mật cố định bất kì k, hàm $b \rightarrow BC.Encrypt(k,b)$ tác động như một phép hoán vị trên tập các xâu bộ tám độ dài $BC.BlockLen$, và hàm $b' \rightarrow BC.Decrypt(k,b)$ tác động như một phép hoán vị nghịch đảo.

CHÚ THÍCH Xem thảo luận trong Phụ lục B.2 về tính chất bảo mật mong muốn của mã khối.

6.4.1 Mã khối được phép

Trong phần này của ISO/IEC 18033, mã khối được phép là những mã khối được mô tả trong ISO/IEC 18033-3, với các qui định sau:

— Trong ISO/IEC 18033-3, các khối bản rõ/bản mã và khóa bí mật là những xâu bit có độ dài cố định, còn trong phần này của ISO/IEC 18033, chúng là những xâu bộ tám có độ dài cố định. Bởi vậy, mã khối trong ISO/IEC 18033-3 được phép áp dụng trong phần này của ISO/IEC 18033 chỉ khi độ dài tính bằng bit của các khối bản rõ/bản mã và của khóa bí mật là bội số của 8, trong trường hợp này ánh xạ giữa các xâu bộ tám và xâu bị tác động bởi các hàm $OS2OSP$ và $BS2OSP$.

6.5 Mã đối xứng

Mã đối xứng SC xác định độ dài khóa $SC.KeyLen$, cùng với các thuật toán mã hóa, giải mã:

— Thuật toán mã hóa $SC.Encrypt(k,M)$ nhận đầu vào là khóa bí mật k, khoá này là xâu bộ tám độ dài $SC.KeyLen$ và bản rõ M là xâu bộ tám độ dài bất kì, đầu ra là bản mã c là xâu bộ tám.

Thuật toán mật mã có thể **thất bại**, nếu độ dài của M vượt quá một giá trị lớn nào đó, được xác định bởi quá trình thực thi.

— Thuật toán giải mã $SC.Decrypt(k,c)$ nhận đầu vào là khóa bí mật k, khoá này là xâu bộ tám độ dài $SC.KeyLen$, và bản mã c là xâu bộ tám độ dài bất kì, đầu ra là bản rõ M là xâu bộ tám.

Thuật toán giải mã có thể thất bại trong một số tình huống nào đó.

Các thuật toán mã hóa và giải mã là tất định. Đồng thời, đối với tất cả khóa bí mật k và tất cả bản rõ M, nếu M không vượt quá giới hạn độ dài của thuật toán mã hóa và nếu $c = SC.Encryt(k,M)$, thì $SC.Decrypt\ (k,c)$ không **thất bại** và $SC.Decrypt\ (k,c) = M$.

CHÚ THÍCH Xem thảo luận trong phụ lục B.3 về các tính chất bảo mật mong muốn của mã đối xứng.

6.5.1 Mật mã đối xứng được phép

Mật mã đối xứng được phép sử dụng trong phần này của ISO/IEC 18033 là:

— $SC1$, được mô tả tại Điều 6.5.2, và

— $SC2$, được mô tả tại Điều 6.5.3.

6.5.2 *SC1*

Mã khối đối xứng này đạt được bằng cách sử dụng mã khối ở chế độ CBC (cipher block chaining – xem ISO/IEC 10116), cùng với lược đồ đệm (padding) cụ thể, để bổ sung bản rõ sao cho độ dài của chúng là bội số của kích thước khối của mã khối cơ sở.

6.5.2.1 Các tham số hệ thống

SC1 là họ mã khối đối xứng được tham số hóa bằng các tham số hệ thống sau đây:

– *BC*: mã khối, được mô tả trong Điều 6.4.

Nói một cách chặt chẽ, cần hạn chế sao cho $BC.BlockLen < 256$. Và hạn chế này luôn được đáp ứng trên thực tế.

6.5.2.2 Đặc tả

$SC1.KeyLen = BC.KeyLen.$

Hàm $SC1.Encrypt(k, M)$ làm việc như sau:

a) Đặt $PadLen = BC.BlockLen - (|M| \bmod BC.BlockLen)$.

b) Giả sử $P_1 = Oct(padLen)$.

c) Giả sử P_2 là xâu bộ tám được hình thành bằng cách lặp lại octet P_1 với tổng số lần lặp bằng $PadLen$ (bởi vậy $|P_2| = padLen$).

d) Giả sử $M' = M \| P_2$.

e) Tạo cú pháp cho M': $M' = M'_1 \| ... \| M'_l$, ở đây với $1 \le i \le l, M'_i$ là xâu bộ tám độ dài $BC.BlockLen$.

f) Giả sử c_0 là xâu bộ tám, bao gồm các bản sao $BC.BlockLen$ của octet $Oct(0)$ và với $1 \le i \le l$, đặt $c_i = BC.Encrypt(k, M'_i \oplus c_{i-1})$.

g) Giả sử $c = c_1 \| \cdots \| c_l$.

h) Đưa ra c.

Hàm $SC1.Decrypt(k, c)$ làm việc như sau:

a) Nếu $|c|$ không phải là bội số khác không của $BC.BlockLen$, hàm sẽ **thất bại**.

b) Tạo cú pháp cho c, $c = c_1 \| \cdots \| c_l$ với $1 \le i \le l$ là xâu bộ tám độ dài $BC.BlockLen$, đồng thời giả sử c_0 là xâu bộ tám bao gồm các bản sao $BC.BlockLen$ của $Oct(0)$.

c) Với $1 \le i \le l$, đặt $M'_i = BC.Decrypt(k, c_i) \oplus c_{i-1}$.

d) Giả sử P_1 là octet cuối cùng của M'_l, và giả sử $padLen = BS2IP(P_1)$.

e) Nếu $padLen \notin [1..BC.BlockLen]$ thì sẽ **thất bại**.

f) Kiểm tra nếu bộ tám cuối cùng $padLen$ của M'_l bằng P_1 hay không; nếu không bằng thì **thất bại**.

g) Giả sử M''_l là xâu bộ tám bao gồm từ những octet đầu tiên $BC.BlockLen - padlen$ của M'_l.

h) Đặt $M = M'_l \| \cdots \| M'_{l-1} \| M''_l$.

i) Đưa ra M.

6.5.3 *SC2*

6.5.3.1 Các tham số hệ thống

SC2 là họ các mã đối xứng, được tham số hóa bởi các tham số hệ thống sau đây:

– KDF: Hàm dẫn xuất khóa được mô tả tại Điều 6.2;

– $KeyLen$: số nguyên dương.

6.5.3.2 Đặc tả

Giá trị của $SC2.KeyLen$ bằng giá trị của tham số hệ thống $KeyLen$.

Hàm $SC2.Encrypt(k, M)$ làm việc như sau:

a) Đặt $mask = KDF(k, |M|)$.

b) Đặt $c = mask \oplus M$.

c) Đưa ra c.

Hàm $SC2.Decrypt(k, c)$ làm việc như sau:

d) Đặt $mask = KDF(k, |c|)$.

e) Đặt $M = mask \oplus c$.

f) Đưa ra M.

7 Mật mã phi đối xứng

Mật mã phi đối xứng AC bao gồm từ ba thuật toán:

– Thuật toán tạo khóa $AC.KeyGen()$, với đầu ra là cặp khóa công khai/khóa bí mật (PK, pk). Cấu trúc của PK và pk phụ thuộc vào mật mã cụ thể.

– Thuật toán mã hóa $AC.Encrypt\ (PK, L, M, opt)$, nhận đầu vào là khóa công khai, nhãn L, bản rõ M và tuỳ chọn opt, đầu ra là bản mã C. Lưu ý rằng L, M và C là các xâu bộ tám. Xem thêm Điều 7.2 phía dưới về *nhãn*. Xem Điều 7.4 để biết thêm về *tuỳ chọn mật mã*.

Thuật toán mã hóa có thể **thất bại**, nếu độ dài L hoặc M vượt quá giới hạn trong thực thi.

– Thuật toán giải mã $AC.Decrypt(pk, L, C)$, nhận đầu vào là khóa riêng pk, nhãn L, bản mã C và đầu ra là bản rõ M.

Thuật toán giải mã có thể **thất bại** trong một số tình huống nào đấy.

Nhìn chung, các thuật toán tạo khóa, mã hóa là các thuật toán xác suất, còn thuật toán giải mã là thuật toán tất định.

CHÚ THÍCH 1 Mục đích của tất cả mật mã phi đối xứng trong phần này của ISO/IEC 18033 là cung cấp độ an toàn thích hợp chống lại kiểu tấn công chọn bản mã thích hợp (được định nghĩa trong [30], và điều này tương đương với khái niệm "khả năng không bị uốn" ("non-malleability"), được định nghĩa trong [17]). Khái niệm an toàn này nhìn chung được cộng đồng nghiên cứu mật mã đánh giá như một hình thức an toàn phù hợp mà mật mã phi đối xứng cần cung cấp. Định nghĩa hình thức của khái niệm an toàn này được trình bày tại phụ lục B.4, và đã được điều chỉnh phù hợp với bản rõ có độ dài thay đổi và vai trò của nhãn; đồng thời cũng định nghĩa một khái niệm yếu hơn một ít về tính an toàn, được gọi là "dễ bị uốn nhẹ" (benign

malleability). Khái niệm "dễ bị uốn thích hợp", nếu không phải với tất cả, thì cũng với đại đa số ứng dụng của mật mã phi đối xứng, và một số mật mã phi đối xứng trong phần này của ISO/IEC 18033 chỉ đạt được mức an toàn này.

CHÚ THÍCH 2 Yêu cầu cơ bản của bất kì mật mã phi đối xứng là tính đúng đắn: Với bất kì cặp khóa công khai/khóa riêng (PK, pk), với cặp bất kì nhãn/bản rõ (L, M) sao cho độ dài L, M không vượt quá giới hạn do thực thi quyết định, bất kì sự mã hóa bản rõ M với nhãn L và PK được giải mã bằng nhãn L và pk để nhận được bản rõ.

CHÚ THÍCH 3 Một ví dụ mật mã phi đối xứng chứng tỏ yêu cầu trên đây về tính đúng đắn không phải luôn luôn thoả mãn là mật mã dựa trên RSA với $n = pq$, p và q là hai số nguyên tố. Thuật toán tạo khóa có thể sử dụng các thuật toán xác suất để kiểm tra liệu p và q có phải là các số nguyên tố không, và thuật toán này có thể đưa ra kết quả sai với xác suất không đáng kể. Nếu xảy ra điều đó, thì thuật toán giải mã có thể không phải là thuật toán nghịch đảo của thuật toán mã hóa.

7.1 Độ dài bản rõ

Một điều quan trọng cần chú ý là, bản rõ có thể có độ dài thay đổi và bất kì, mặc dù khi thực thi có thể áp đặt một giới hạn trên (thường là rất lớn) cho độ dài này.

Tuy vậy, có hai dạng mật mã phi đối xứng suy biến, được định nghĩa như sau :

— *Mật mã phi đối xứng AC* độ dài bản rõ cố định chỉ mã hóa những bản rõ có độ dài (tính bằng số bộ tám) bằng một giá trị cố định $AC.MsgLen$.

— *Mật mã phi đối xứng AC* độ dài bản rõ bị giới hạn chỉ mã hóa những bản rõ có độ dài (tính bằng bộ tám) nhỏ hơn hoặc bằng một giá trị cố định $AC.MaxMsgLen(PK)$. Ở đây độ dài cực đại của bản rõ có thể phụ thuộc vào khóa công khai của bản mã.

CHÚ THÍCH Ngoại trừ mật mã phi đối xứng độ dài bản rõ cố định và độ dài bản rõ bị giới hạn, phép mã hóa bản rõ nói chung không che giấu được độ dài bản rõ. Bởi vậy khi áp dụng mật mã phi đối xứng, cần thiết phải đảm bảo, có thể bằng cách sử dụng lược đồ đệm bản rõ, để không một thông tin nhạy cảm nào được mã hóa ở dạng ẩn có độ dài bằng độ dài bản rõ.

7.2 Sử dụng nhãn

Nhãn là xâu bộ tám mà giá trị của nó được sử dụng bởi các thuật toán mã hóa và giải mã. Nhãn có thể chứa dữ liệu công khai được che giấu khỏi ngữ cảnh và không cần thiết được mã hóa, nhưng nó lại phải liên quan đến bản mã.

Nhãn là xâu bộ tám có nhiều ý nghĩa đối với các ứng dụng sử dụng mật mã phi đối xứng và độc lập với quá trình thực thi mật mã phi đối xứng.

Nhãn có thể tùy ý, có độ dài thay đổi, mặc dù với mật mã cụ thể có thể chọn giới hạn trên (rất lớn) cho độ dài của nó.

Dạng suy biến của mật mã phi đối xứng được định nghĩa như sau:

— Mật mã phi đối xứng với độ dài nhãn cố định là mật mã với các thuật toán mã hóa và giải mã chỉ chấp nhận nhãn có độ dài (tính bằng bộ tám) bằng một giá trị cố định $AC.LabelLen$.

CHÚ THÍCH 1 Khái niệm an toàn truyền thống chống lại tấn công chọn bản mã, được mở rộng trong phụ lục B.4, sao cho bằng trực giác, đối với mật mã phi đối xứng, thuật toán mã hóa cần gắn nhãn vào bản mã theo một kiểu phù hợp "không có khả năng uốn".

CHÚ THÍCH 2 Ví dụ, có những giao thức trao đổi khóa, trong đó một phía, chẳng hạn A mã hóa khóa phiên bằng khóa công khai của phía thứ hai, chẳng hạn B. Để giao thức được an toàn, định danh của phía A (hoặc khóa công khai hoặc chứng thư

số) phải "không có khả năng bị uốn" đối với bản mã. Một cách để thực hiện điều này là, đơn giản chỉ cần gắn định danh vào bản rõ. Tuy vậy, điều này làm cho bản mã trở nên dài không cần thiết, vì thường thì *A* đã biết định danh của *B* theo ngữ cảnh cụ thể của giao thức. Việc thực thi tốt cơ chế dán nhãn sẽ mang lại cùng hiệu quả, mà không cần tăng kích thước bản mã.

7.3 Định dạng bản mã

Mật mã phi đối xứng được đề xuất trong phần này của ISO/IEC 18033 mô tả chính xác cách định dạng bản mã như một xâu bộ tám. Tuy vậy, việc thực thi có thể tiến hành độc lập với lưu trữ và/hoặc truyền bản bản mã trong định dạng đã lựa chọn để thay thế, nếu như điều đó là thuận tiện. Hơn nữa, quá trình mã hóa bản rõ và biến đổi bản mã thu được theo định dạng đã chọn, có thể suy biến thành một quá trình đơn tương đương về chức năng. Tương tự như thế quá trình biến đổi từ định dạng đã chọn và giải mã bản mã có thể suy biến thành một quá trình đơn, tương đương về chức năng. Như vậy, trong hệ thống cho trước, bản mã nhất thiết không được xuất hiện ở dạng đã qui định ở đây.

CHÚ THÍCH Bên cạnh việc tăng cường tính liên tác, việc qui định định dạng của bản mã là cần thiết để đưa ra các mệnh đề có ý nghĩa và lập luận cứ cho tính an toàn của mật mã phi đối xứng chống lại các tấn công chọn bản mã phù hợp.

7.4 Lựa chọn mật mã

Một số mật mã phi đối xứng cho phép một số dạng tùy chọn cụ thể ở dạng lược đồ để chuyển sang thuật toán mật mã, điều này cắt nghĩa vì sao biến tùy chọn mật mã bổ sung *opt* lại được cho phép trong giao diện trừu tượng cho mật mã phi đối xứng.

Một số mật mã phi đối xứng được trình bày ở đây, một cách tự nhiên có thể xem như không chứa bất kì tùy chọn mật mã, trong những trường hợp như thế mật mã được coi là không có tùy chọn.

Hệ thống có thể cung cấp giá trị "mặc định" của biến *opt*. Tuy vậy, vấn đề này nằm ngoài phạm vi của phần này.

CHÚ THÍCH Trong các mật mã phi đối xứng được mô tả trong phần này của ISO/IEC 18033, chỉ có mật mã dựa trên đường cong elliptic là sử dụng chế độ tùy chọn mật mã, chế độ này được sử dụng nhằm chỉ dẫn định dạng mong muốn để mã hóa các điểm trên đường cong elliptic.

7.5 Phương pháp vận hành mật mã phi đối xứng

Thông thường, thuật toán tạo khóa do một bên thực hiện, đó là chủ nhân của cặp khóa, hoặc là bên được tin cậy được chủ nhân ủy thác. Khóa công khai được tạo ra cho tất cả các bên, những người muốn gửi thông báo được mã hóa đến cho chủ nhân, nhưng khóa riêng thì không được tiết lộ cho bất kì bên nào trừ chủ nhân của nó. Về cơ chế và các giao thức tạo khóa công khai cho các bên khác nằm ngoài phạm vi của phần này. Về vấn đề này xem hướng dẫn tại ISO/IEC 11770.

Mỗi một mật mã phi đối xứng được trình này ở phần này của ISO/IEC 18033 là thành phần của họ các mật mã phi đối xứng, ở đấy mật mã cụ thể được chọn từ họ mật mã bằng cách chọn giá trị cụ thể của các tham số hệ thống, các tham số này xác định họ các mật mã.

Đối với mỗi mật mã cụ thể được chọn từ họ mật mã, các giá trị cụ thể được chọn trước khi tạo khóa. Tùy vào các qui ước được sử dụng để mã hóa khóa công khai, một số lựa chọn của tham số hệ thống có thể được nhúng vào quá trình mã hóa khóa công khai. Những tham số hệ thống này sẽ được cố định suốt cả thời gian sống của khóa công khai.

CHÚ THÍCH Ví dụ, nếu mã phi đối xứng có thể được tham số hóa theo thuật ngữ của hàm băm mật mã, thì việc chọn hàm *hash* nên được cố định và nói chung tại một điểm nào đó trước khi tạo cặp khóa công khai/khóa riêng, và các thuật toán mã hóa, giải mã nên sử dụng để chọn hàm băm suốt thời gian sống của khóa công khai. Bỏ qua nguyên tắc này không chỉ làm chệch việc thực thi, mà còn làm mất ý nghĩa của việc phân tích tính an toàn của mật mã, và có thể trong một số trường hợp, đẩy việc thực thi vào những rủi ro nghiêm trọng.

7.6 Mật mã phi đối xứng được phép

Người dùng muốn khai thác mật mã phi đối xứng trong phần này của ISO/IEC 18033 sẽ chọn một trong số các mật mã sau đây:

— Mật mã kết hợp vạn năng, được chọn từ họ HC các mật mã lai ghép, được mô tả tại Điều 8.3;

— Mã phi đối xứng với độ dài bản mã giới hạn từ họ $RSAES$ các mật mã, được mô tả tại Điều 11.4;

— Mã phi đối xứng với độ dài bản rõ bị giới hạn thuộc họ $HIME(R)$ được mô tả tại Điều 12.3.

CHÚ THÍCH Vì HC, $RSAES$ và $HIME$ là họ mật mã được tham số hóa bởi các tham số hệ thống khác nhau, người dùng sẽ phải chọn các giá trị cụ thể của các tham số hệ thống từ tập các tham số hệ thống được phép, được xác định trong các Điều tương ứng, trong đó mỗi họ mật mã được mô tả.

8 Mật mã lai ghép tổng quát

Khi thiết kế mật mã phi đối xứng hiệu quả, một cách tiếp cận hữu ích là thiết kế *mật mã lai ghép*, ở đó có thể sử dụng kỹ thuật mật mã phi đối xứng để mã hóa khóa bí mật, khóa bí mật này sau đó được sử dụng để mã thông báo, sử dụng kỹ thuật mật mã đối xứng. Điều này mô tả một dạng mật mã lai đặc biệt, được gọi là *mật mã lai ghép tổng quát*. Mật mã lai ghép tổng quát được xây dựng từ hai "khối kiến tạo" mức thấp hơn: *cơ chế bọc khóa* và *cơ chế bọc dữ liệu*. Điều 8.3 đặc tả chi tiết họ HC các mật mã lai ghép tổng quát.

8.1 Cơ chế bọc khóa

Cơ chế bọc khóa KEM bao gồm ba thuật toán:

— Thuật toán tạo khóa $KEM.KeyGen()$, với đầu ra là cặp khóa công khai/khóa riêng (PK, pk). Cấu trúc của PK và pk phụ thuộc vào lược đồ cụ thể.

— Thuật toán mã hóa $KEM.Encrypt(PK, opt)$, nhận đầu vào là khóa công khai PK và tùy chọn mật mã opt, đầu ra là cặp khóa mật/bản mã (K, C_0). K và C_0 là các xâu bộ tám. Vai trò của opt ở đây cũng tương tự như trong mật mã phi đối xứng (xem Điều 7.4).

— Thuật toán giải mã $KEM.Decrypt\ (pk, C_0)$ với đầu vào là khóa riêng pk và bản mã C_0, đầu ra là khóa bí mật K, K và C_0 là các xâu bộ tám.

Thuật toán giải mã có thể thất bại trong một số bối cảnh nhất định.

Cơ chế bọc khóa cũng xác định một số nguyên dương $KEM.KeyLen$ - độ dài khóa bí mật là đầu ra của $KEM.Encrypt$ và $KEM.Decrypt$.

CHÚ THÍCH Cơ chế bọc khóa cần thỏa mãn tính đúng đắn, tương tự tính đúng đắn của mật mã phi đối xứng: với bất kì cặp khóa công khai/khóa riêng (PK, pk), bất kì đầu ra $(K,\ C_0)$ của thuật toán mã hóa với đầu vào (PK, opt), bản mã C_0 có thể

được giải mã nhờ pk để thu được K. Yêu cầu này có thể được giảm nhẹ, bởi vậy nói chung chỉ nó chỉ giữ một phần không đáng kể cặp khóa công khai/bí mật.

8.1.1 Tính chất phi tiền tố

Ngoài ra, cơ chế bọc khóa còn phải thỏa mãn tính chất sau đây. Tập hợp tất cả các đầu ra - các bản mã có thể - của thuật toán mã hóa là tập con của tập ứng cử viên các xâu bộ tám (có thể phụ thuộc vào khóa công khai), sao cho tập ứng cử viên này là phi tiền tố và các phần tử của tập này dễ dàng được nhận dạng (hoặc cho trước khóa công khai hoặc khóa riêng).

8.1.2 Các cơ chế bọc khóa được phép

Cơ chế bọc khóa được phép trong phần này của tiêu chuẩn ISO/IEC 18033 gồm:

– $ECIES - KEM$ (được mô tả tại Điều 10.2),

– $PSEC - KEM$ (được mô tả tại Điều 10.3),

– $ACE - KEM$ (được mô tả tại Điều 10.4), và

– $RSA - KEM$ (được mô tả tại Điều 11.5).

CHÚ THÍCH 1 Để thuận tiện, các mật mã lai ghép tổng quát tương ứng được xây dựng từ các cơ chế bọc khóa trên thông qua cấu trúc lai ghép tổng quát tại Điều 8.3 tương ứng được gọi là $ECIES - HC, PSEC - HC, ACE - HC,$ và $RSA - HC$.

CHÚ THÍCH 2 Nói một cách đại thể, cơ chế bọc khóa làm việc tương tự như mật mã phi đối xứng, ngoại trừ một điều là thuật toán mã hóa không nhận đầu vào nào khác ngoài khóa công khai của người nhận: thay vì nhận đầu vào là thông báo và tạo ra bản mã, thuật toán mã hóa tạo ra cặp khóa bí mật/bản mã (K, C_0), ở đây K là xâu bộ tám có độ dài xác định và C_0 là mã hóa của K, như vậy thuật toán giải mã áp dụng vào C_0 và đưa ra K.

CHÚ THÍCH 3 Luôn luôn có thể sử dụng mật mã phi đối xứng (với độ dài bản rõ cố định hay độ dài bản rõ bị giới hạn) để tạo ra xâu bộ tám ngẫu nhiên K và sau đó mã hóa nó bằng khóa công khai của người nhận (và tùy chọn nào đó để thu được C_0). Tuy nhiên, cũng có thể thiết kế cơ chế bọc khóa theo cách khác hiệu quả hơn.

CHÚ THÍCH 4: Để xây dựng mật mã lai ghép tổng quát an toàn chống lại tấn công chọn bản mã phù hợp, tồn tại khái niệm an toàn tương ứng cho cơ chế bọc khóa. Điều này được thảo luận chi tiết tại Phụ lục B.5.

8.2 Các cơ chế bọc dữ liệu

Cơ chế bọc dữ liệu DEM xác định độ dài khóa $DEM.KeyLen$, và các thuật toán mã hóa và giải mã.

– Thuật toán mã hóa $DEM.Encrypt(K, L, M)$ nhận đầu vào là khóa bí mật K, nhãn L và bản rõ M. Thuật toán cho đầu ra là bản mã C_1. Ở đây K, L, M và C_1 là các xâu bộ tám, trong đó L và M có thể có độ dài tùy ý, còn K có độ dài $DEM.KeyLen$.

Thuật toán mã hóa có thể bị **thất bại**, nếu độ dài L hoặc M vượt qua giới hạn (rất lớn) được xác định bởi quá trình thực thi.

– Thuật toán giải mã $DEM.Decrypt(K, L, C_1)$ nhận đầu vào là khóa bí mật K, nhãn L và bản mã C_1 cho đầu ra là bản rõ M.

Thuật toán giải mã có thể thất bại trong một số bối cảnh nhất định.

CHÚ THÍCH Các thuật toán mã hóa và giải mã nên là thuật toán tất định, thỏa mãn yêu cầu sau đây về tính đúng đắn: với mọi khóa bí mật, tất cả L và tất cả bản rõ, đều có độ dài L và M không vượt quá giới hạn được xác định bởi thực thi,

$$DEM.Decrypt(K, L, DEM.Encrypt(K, L, M)) = M.$$

8.2.1 Các dạng cơ chế bọc dữ liệu suy biến

Có hai dạng cơ chế bọc dữ liệu suy biến được xác định như sau:

– *Cơ chế bọc dữ liệu với độ dài nhãn cố định* là cơ chế mà thuật toán mã hóa và giải mã chỉ chấp nhận nhãn có độ dài bằng giá trị cố định $DEM.LabelLen$.

– *Cơ chế bọc dữ liệu với độ dài bản rõ cố định* là cơ chế mà thuật toán mã hóa chỉ chấp nhận các bản rõ có độ dài bằng giá trị cố định $DEM.MsgLen$.

8.2.2 Cơ chế bọc dữ liệu được phép

Cơ chế bọc dữ liệu trong phần này của tiêu chuẩn ISO/IEC 18033 được mô tả tại Điều 9.

CHÚ THÍCH 1 Nói đại thể, cơ chế bọc dữ liệu cung cấp "phong bì số", bảo vệ cả tính bí mật lẫn tính toàn vẹn của dữ liệu sử dụng kỹ thuật mật mã đối xứng. Nó đồng thời gắn dữ liệu vào nhãn công khai.

CHÚ THÍCH 2 Để xây dựng mã lai ghép tổng quát, chống được tấn công chọn bản mã, tồn tại khái niệm tương ứng về an toàn cho cơ chế bọc dữ liệu. Điều này được xem xét chi tiết tại Phụ lục B.6.

8.3 HC

8.3.1 Các tham số hệ thống

HC là họ mật mã phi đối xứng được tham số hóa như sau:

– KEM: cơ chế bọc khóa được mô tả tại Điều 8.1;

– DEM: cơ chế bọc khóa được mô tả tại Điều 8.2.

Mọi sự kết hợp giữa KEM và DEM đều có thể sử dụng, nếu đảm bảo điều kiện $KEM.KeyLen = DEM.KeyLen$.

CHÚ THÍCH 1 Nếu DEM là cơ chế bọc dữ liệu với độ dài nhãn cố định, trong đó độ dài nhãn bị hạn chế bởi $DEM.LabelLen$, thì HC là mã đối xứng với độ dài nhãn cố định và $HC.LabelLen = DEM.KeyLen$.

CHÚ THÍCH 2 Nếu DEM là cơ chế bọc dữ liệu với độ dài bản rõ cố định, trong đó bản rõ có độ dài hạn chế bởi $DEM.MsgLen$, thì HC là mã đối xứng với độ dài bản rõ cố định và $HC.MsgLen = DEM.MsgLen$.

CHÚ THÍCH 3 Với tất cả sự lựa chọn được phép của KEM, giá trị của $KEM.KeyLen$ là tham số hệ thống có thể được chọn bằng $DEM.KeyLen$. Như vậy mọi sự kết hợp có thể của KEM và DEM có thể được thực hiện bằng cách chọn thích hợp các tham số hệ thống.

8.3.2 Tạo khóa

Thuật toán tạo khóa, khóa công khai và khóa riêng cho HC cũng như cho KEM; Tùy chọn mật mã của HC cũng như của KEM.

Giả sử *(PK, pk)* là cặp khóa công khai/khóa riêng.

8.3.3 Mã hóa

Thuật toán mã hóa $HC.Encrypt$ nhận đầu vào là khóa công khai PK, nhãn L, bản rõ M và tùy chọn mật mã opt. Thuật toán này chạy như sau:

a) Tính $(K, C_0) = KEM.Encrypt(PK, opt)$.

b) Tính $C_1 = DEM.Encrypt(K, L, M)$.

c) Tạo $C = C_0 \parallel C_1$.

d) Đưa ra C.

8.3.4 Giải mã

Thuật toán giải mã $HC.Decrypt$ nhận đầu vào là khóa riêng pk, nhãn L và bản mã C. Thuật toán chạy như sau:

a) Sử dụng tính chất phi tiền tố của bản mã liên kết với KEM (xem Điều 8.1.1), tạo cú pháp C, $C = C_0 \| C_1$, ở đây C_0 và C_1 là các xâu bộ tám sao cho C_0 là phần tử của tập hợp ứng cử viên của các bản mã có thể liên kết với KEM. Bước này sẽ **thất bại** nếu C không được tạo cú pháp.

b) Tính $K = KEM.Decrypt(pk, C_0)$.

c) Tính $M = DEM.Decrypt(K, L, C_1)$.

d) Đưa ra M.

CHÚ THÍCH Độ an toàn của HC được xem xét tại phụ lục B.7. Tại đây lưu ý rằng chừng nào KEM và DEM thỏa mãn các tính chất an toàn tương ứng, thì HC sẽ là an toàn đối với tấn công chọn bản mã thích hợp.

9 Thiết kết các cơ chế bọc dữ liệu

Điều khoản này đưa ra các cơ chế bọc dữ liệu, gồm:

– $DEM1$, được mô tả trong Điều 9.1,

– $DEM2$, được mô tả trong Điều 9.2, và

– $DEM3$, được mô tả trong Điều 9.3.

9.1 $DEM1$

9.1.1 Các tham số hệ thống

$DEM1$ là họ các cơ chế bọc dữ liệu, được tham số hóa bằng các tham số hệ thống sau:

– SC: mật mã đối xứng, được mô tả tại Điều 6.5;

– MA: thuật toán MAC được mô tả trong Điều 6.3.

Giá trị của $DEM1.KeyLen$ được xác định như sau: $DEM1.KeyLen = SC.KeyLen + MA.KeyLen$.

9.1.2 Mã hóa

Thuật toán $DEM1.Encrypt$ nhận đầu vào là khóa bí mật K, nhãn L, và bản rõ M. Thuật toán chạy như sau:

a) Tạo cú pháp K, $K = k \| k'$, ở đây k và k' là các xâu bộ tám sao cho $|k| = SC.KeyLen$ và $|k'| = MA.KeyLen$.

b) Tính $c = SC.Encrypt(k, M)$.

c) Giả sử $T = c \| L \| I2OSP(8.|L|, 8)$.

d) Tính $MAC = MA.eval(k', T)$.

e) Thiết lập $C_1 = c \parallel MAC$.

f) Đưa ra C_1.

9.1.3 Giải mã

Thuật toán $DEM1.Decrypt$ nhận đầu vào là khóa bí mật K, nhãn L, và bản mã C_1. Thuật toán chạy như sau:

a) Tạo cú pháp $K, K = k \| k'$; ở đấy k và k' là các xâu bộ tám với $|k| = SC.KeyLen$ và $|k'| = MA.KeyLen$.

b) Nếu $|C_1| < MA.MACLen$ thì **thất bại**.

c) Tạo cú pháp $C_1 : C_1 = c \| MAC$, ở đây c và MAC là các xâu bộ tám với $|MAC| = MA.MACLen$.

d) Giả sử $T = c \| L \| I2OSP(8.|L|, 8)$.

e) Tính $MAC' = MA.eval(k', T)$.

f) Nếu $MAC \neq MAC'$ thì **thất bại**.

g) Tính $M = SC.Decrypt(k, c)$.

h) Đưa ra M.

CHÚ THÍCH Việc xem xét chi tiết tính an toàn của cấu trúc này được đưa ra trong Phụ lục B.6.1. Ở đây chỉ lưu ý là các *SA* và *MA* cơ sở được cung cấp thoả mãn các yêu cầu tương ứng, sau đó là cả *DEM1* cũng thỏa mãn các yêu cầu này.

9.2 *DEM2*

9.2.1 Các tham số hệ thống

DEM2 là họ các cơ chế bọc dữ liệu với độ dài nhãn cố định, được tham số hóa bằng các tham số hệ thống sau đây:

— *SC*: mã đối xứng được mô tả tại Điều 6.5;

— *MA*: thuật toán MAC, được mô tả tại Điều 6.3;

— *LabelLen*: số nguyên không âm.

Giá trị $DEM2.LabelLen$ được xác định là bằng giá trị của tham số hệ thống $LabelLen$.

Giá trị $DEM2.KeyLen$ được xác định bằng: $DEM2.KeyLen = SC.KeyLen + MA.KeyLen$.

9.2.2 Mã hóa

Thuật toán $DEM2.Encrypt$ nhận đầu vào là khóa bí mật K, nhãn L độ dài $LabelLen$, bản rõ M. Thuật toán chạy như sau:

a) Tạo cú pháp K, $K = k \| k'$ ở đấy k và k' là các xâu bộ tám sao cho $|k| = SC.KeyLen$ và $|k'| = MA.KeyLen$.

b) Tính $c = SC.Encrypt(k, M)$.

c) Đặt $T = c \| L$.

d) Tính $MAC = MA.eval\ (k', T)$.

e) Thiết lập $C_1 = c \parallel MAC$.

f) Đưa ra C_1.

9.2.3 Giải mã

Thuật toán $DEM2.Decrypt$ nhận đầu vào là khóa bí mật K, nhãn L độ dài $LabelLen$ và bản mã C_1. Thuật toán chạy như sau:

a) Tạo cú pháp K, $K = k \| k'$ ở đấy k và k' là các xâu bộ tám sao cho $|k| = SC.KeyLen$ và $|k'| = MA.KeyLen$.

b) Nếu $|C_1| < MA.MACLen$ thì **thất bại**.

c) Tạo cú pháp $C_1, C_1 = c \| MAC$, ở đây c và MAC là các xâu bộ tám sao cho $|MAC| = MA.MACLen$.

d) Giả sử $T = c \parallel L$.

e) Tính $MAC' = MA.eval\ (k', T)$.

f) Nếu $MAC' \neq MAC$ thì **thất bại**.

g) Tính $M = SC.Decrypt(k, c)$.

h) Đưa ra M.

CHÚ THÍCH 1 Việc xem xét chi tiết tính an toàn của thiết kế trên được trình bày trong Phụ lục B.6.1. Ở đây chỉ lưu ý là các SC và MA thỏa mãn các yêu cầu an toàn tương ứng, khi đó $DEM2$ cũng thỏa mãn các yêu cầu này.

CHÚ THÍCH 2 $DEM2$ được cung cấp chủ yếu để so sánh với các tiêu chuẩn khác.

9.3 *DEM3*

9.3.1 Các tham số hệ thống

$DEM3$ là họ các cơ chế bọc dữ liệu với độ dài bản rõ cố định, được tham số hóa bởi các tham số hệ thống sau đây:

– MA: thuật toán MAC được mô tả ở Điều 6.3;

– $MsgLen$: số nguyên dương.

Giá trị của $DEM3.MsgLen$ được định nghĩa bằng giá trị của tham số hệ thống $MsgLen$.

Giá trị của $DEM3.KeyLen$ được xác định bằng,

$$DEM3.KeyLen = MsgLen + MA.KeyLen.$$

9.3.2 Mã hóa

Thuật toán $DEM3.Encrypt$ nhận đầu vào là khóa bí mật K, nhãn L, bản rõ M với độ dài $MsgLen$. Thuật toán chạy như sau:

a) Tạo cú pháp K, $K = k \| k'$ ở đấy k và k' là các xâu bộ tám (octet) với $|k| = MsgLen$ và $|k'| = MA.KeyLen$.

b) Tính $c = k \oplus M$.

c) Giả sử $T = c \| L$.

d) Tính $MAC = MA.eval(k', T)$.

e) Đặt $C_1 = c \| MAC$.

f) Đưa ra C_1.

9.3.3 Giải mã

Thuật toán $DEM3.Decrypt$ nhận đầu vào là khóa bí mật K, nhãn L và bản mã C_1. Thuật toán chạy như sau:

a) Tạo cú pháp cho K, $K = k \| k'$ ở đấy k và k' là các xâu bộ tám với $|k| = MsgLen$ và $|k'| = MA.KeyLen$.

b) Nếu $C_1 \neq MsgLen + MA.MACLen$ thì **thất bại**.

c) Tạo cú pháp $C_1 : C_1 = c \| MAC$, ở đây c và MAC là các xâu bộ tám với $|c| = MsgLen$ và $|MAC| = MA.MACLen$.

d) Đặt $T = c \| L$.

e) Tính $MAC' = MA.eval(k', T)$.

f) Nếu $MAC \neq MAC'$ thì **thất bại**.

g) Tính $M = k \oplus c$.

h) Đưa ra M.

CHÚ THÍCH 1 Việc xem xét chi tiết tính an toàn của thiết kế này được trình này ở Phụ lục B.6.1. Cần lưu ý ở đây là nếu *SC* và *MA* thỏa mã các yêu cầu an toàn tương ứng thì *DEM3* cũng thỏa mãn các yêu cầu đó.

CHÚ THÍCH 2 *DEM3* được cung cấp chủ yếu để so sánh với các tiêu chuẩn khác.

10 Cơ chế bọc khóa dựa vào ElGamal

Điều này mô tả một số cơ chế bọc khóa dựa trên bài toán logarit rời rạc:

— *ECIES-KEM* được mô tả trong Điều 10.2;

— *PSEC-KEM* được mô tả trong Điều 10.3;

— *ACE-KEM* được mô tả trong Điều 10.4.

CHÚ THÍCH Tất cả các lược đồ trên đều là những biến thể của lược đồ mật mã gốc ElGamal [18].

10.1 Các nhóm cụ thể

Mật mã ElGamal dựa trên các phép toán số học trên nhóm hữu hạn. Để mô tả cơ chế bọc khóa dựa trên mật mã ElGamal, khái niệm nhóm được mô tả như một kiểu dữ liệu trừu tượng. Việc mô tả và phân tích các lược đồ dựa vào giao diện trừu tượng này, tuy nhiên trong phần này của tiêu chuẩn ISO/IEC 18033, chỉ cho phép sử dụng một số dạng nhóm nhất định bằng cách cụ thể hóa kiểu dữ liệu trừu tượng này.

Để tiện, khái niệm phép cộng luôn được sử dụng cho nhóm. Đồng thời các phần tử của nhóm sẽ được viết bằng chữ cái thường, đậm nét và **0** kí hiệu phần tử đồng nhất của nhóm.

Nhóm cụ thể Γ là bộ ($\mathcal{H}$, $\mathcal{G}$, g, μ, ν, $\mathcal{E}$, $\mathcal{D}$, $\mathcal{E}'$, $\mathcal{D}'$),

— $\mathcal{H}$ là nhóm Abel hữu hạn. Chú ý rằng nhóm không nhất thiết là tuần hoàn.

— $\mathcal{G}$ là nhóm con tuần hoàn của $\mathcal{H}$.

— g là phần tử sinh của $\mathcal{G}$.

— μ là bậc của $\mathcal{G}$, và ν là chỉ số của $\mathcal{G}$ trong $\mathcal{H}$, nghĩa là $\nu = |\mathcal{H}|/\mu$.

Ở đây đòi hỏi μ phải là số nguyên tố. Trong một số lược đồ mật mã, đòi hỏi thêm điều kiện $gcd(\mu, \nu) = 1$.

— $\mathcal{E}(a, fmt)$ là hàm "mã hóa" (encoding function), ánh xạ phần tử nhóm $a \in \mathcal{H}$ vào xâu bộ tám; biến thứ hai fmt là bộ định dạng, được sử dụng để chọn một trong số lượng nhỏ các định dạng có thể để mã hóa phần tử của nhóm. Các giá trị được phép của biến fmt phụ thuộc vào nhóm.

Các yêu cầu sau phải được thỏa mãn:

— Tập tất cả các đầu ra $\mathcal{E}$ là tập phi tiền tố.

— Phần tử trung hòa được mã hóa (encoding) duy nhất, bởi vậy với tất cả các biến định dạng fmt, fmt' ta có: $\mathcal{E}(0, fmt) = \mathcal{E}(0, fmt')$.

— Trừ phần tử trung hòa, hàm mã hóa là ánh xạ một-một, do đó với tất cả a và $a' \in \mathcal{H}$ và tất cả các biến định dạng fmt, fmt', nếu $(a, fmt) \neq (a', fmt')$ và nếu $a \neq 0$ hoặc $a' \neq 0$, thì $\mathcal{E}(a, fmt) \neq \mathcal{E}(a', fmt')$.

Xâu bộ tám x được gọi là *mã hợp lệ* (valid encoding) của phần tử nhóm a $\in \mathcal{H}$, nếu $x = \mathcal{E}(a, fmt)$ với một biến định dang fmt nào đó.

— $\mathcal{D}(x)$ là hàm sao cho nó sẽ trả về nếu x không phải là mã hợp lệ của phần tử thuộc $\mathcal{H}$, ngược lại, hàm hoàn lại phần tử nhóm duy nhất a $\in \mathcal{H}$, sao cho $\mathcal{E}(a, fmt) = x$ với một biến định dạng fmt nào đó.

— $\mathcal{E}'(a)$ là hàm "mã từng phần", ánh xạ mỗi phần tử nhóm $a \in \mathcal{H}$, vào xâu bộ tám.

Điều này đòi hỏi tập hợp tất cả đầu ra của $\mathcal{E}'$ là phi tiền tố.

Xâu bộ tám x được gọi là *mã từng phần hợp lệ* (valid partial encoding) của phần tử nhóm a nếu $x = \mathcal{E}'(a)$.

— $\mathcal{D}'(x)$ là hàm sao cho hoặc là **thất bại** nếu x không phải là mã cụ thể của phần tử thuộc $\mathcal{H}$; ngược lại, nó sẽ trả về một tập hợp chứa tất cả các phần tử nhóm $a \in \mathcal{H}$, sao cho $\mathcal{E}'(a) = x$. Giả thiết kích thước của tập hợp này bị giới hạn bởi một hằng số nhỏ.

Giả thiết có thể thực hiện hiệu quả các phép toán số học trong $\mathcal{H}$. Đồng thời, tất cả các thuật toán trên đây đều được thực thi hiệu quả. Hàm $\mathcal{D}'$ sẽ không được sử dụng bởi bất kì lược đồ nào, nhưng sự tồn tại của hàm này là cần cho việc phân tích tính an toàn các lược đồ đó.

Giả sử có thể kiểm tra hiệu quả, liệu một phần tử của $\mathcal{H}$ có nằm trong nhóm con $\mathcal{G}$ hay không. Lưu ý rằng nếu tất cả các phần tử của $\mathcal{H}$, có bậc μ nằm trong $\mathcal{G}$, thì có thể kiểm tra $a \in \mathcal{G}$ bằng cách kiểm tra điều kiện $\mu . a = 0$. Bởi vậy phép kiểm tra này có thể áp dụng được nếu bản thân $\mathcal{H}$ là nhóm tuần hoàn và $gcd(\mu, v) = 1$. Với các nhóm cụ thể, có thể có các phép kiểm tra hiệu quả hơn về việc phần tử cho trước có thuộc nhóm con hay không.

Tập hợp $\{\mathcal{E}(a_1, fmt_1), \ldots, \mathcal{E}(a_m, fmt_m)\}$ các mã hợp lệ được gọi là tập bền vững, nếu mã (encodings) của các phần tử nhóm không phải là phần tử trung hòa sử dụng cùng một biến định dạng fmt; bởi vậy với mọi $1 \leq i, j \leq m$, nếu $a_i \neq 0$, $a_j \neq 0$ thì $fmt_i = fmt_j$. Với những giả thiết như trên có thể kiểm tra một cách hiệu quả một tập hợp các mã hợp lệ cho trước có bền vững hay không.

CHÚ THÍCH Các ứng dụng mật mã khác nhau sẽ đưa ra các giả thiết khác nhau về tính khó giải của nhóm. Những giả thiết này được xem xét tại Phụ lục B.8.

10.1.1 Các nhóm cụ thể được phép

Phần này của ISO/IEC 18033 chỉ cho phép hai họ nhóm cụ thể sau đây, được mô tả tại các Điều 10.1.2 và 10.1.3.

10.1.2 Nhóm con của các trường hữu hạn được cho dưới dạng tường minh

Giả sử F là trường hữu hạn dạng tường minh, được định nghĩa tại Điều 5.3 và F^* là nhóm nhân các đơn vị của F. Giả sử $\mathcal{H}$ là kí hiệu của F^*, $\mathcal{G}$ là nhóm con của F^* có bậc nguyên tố, g là phần tử sinh của $\mathcal{G}$. Đặt $\mu = |\mathcal{G}|$ và $v = (|F| - 1)/\mu$.

Vì $\mathcal{H}$ là nhóm tuần hoàn, suy ra $\mathcal{G}$ chứa tất cả các phần tử của $\mathcal{H}$ có bậc chia hết μ, thậm chí khi $gcd(\mu, v) \neq 1$. Như vậy, luôn luôn có thể kiểm tra phần tử $a \in \mathcal{H}$ có thuộc $\mathcal{G}$ hay không bằng cách kiểm tra hệ thức $\mu . a = 0$. Tuy nhiên, còn có thể có những phép kiểm tra hiệu quả hơn, ví dụ nếu F là trường hữu hạn nguyên tố và $v = 2$, thì phép kiểm tra này có thể thực hiện bằng cách tính kí hiệu Jacobi.

Ánh xạ mã hóa $\mathcal{E}$ (encoding map) được thực thi bằng cách sử dụng hàm $FE2OSP_F$, bởi vậy tất cả các phần tử nhóm được mã hóa dưới dạng các xâu bộ tám có độ dài $\lceil \log_{256} |F| \rceil$. Chỉ cho phép một định dạng. Ánh xạ $\mathcal{D}$ được thực thi bằng cách sử dụng $OS2FEP_F$ và thất bại nếu $OS2FEP_F$ thất bại hoặc đưa ra 0_F. Hàm $\mathcal{E}'$ cũng giống như $\mathcal{E}$ và $\mathcal{D}'$ cũng giống như $\mathcal{D}$.

10.1.3 Nhóm con của đường cong Elliptic

Giả sử E là đường cong elliptic trên trường hữu hạn dạng tường minh F, được mô tả tại Điều 5.4. Kí hiệu $\mathcal{H}$ là nhóm các điểm trên E, $\mathcal{G}$ là nhóm con bậc nguyên tố của $\mathcal{H}$, g là phần tử sinh của $\mathcal{G}$; giả sử μ là bậc của $\mathcal{G}$, v là chỉ số của nó trong $\mathcal{H}$.

Ta thấy, nói chung $\mathcal{H}$ không phải là nhóm tuần hoàn. Nếu $gcd(\mu, v) = 1$ thì có thể kiểm tra phần tử $a \in \mathcal{H}$ có nằm trong $\mathcal{G}$ hay không, bằng cách kiểm tra hệ thức $\mu . a = 0$. Nếu $gcd(\mu, v) \neq 1$ khi đó yêu cầu phải có nhiều thông tin hơn về cấu trúc nhóm của E để xây dựng nên phép kiểm tra hiệu quả về việc phần tử thuộc $\mathcal{H}$ có nằm trong $\mathcal{G}$ hay không.

Các ánh xạ mã hóa/ giải mã $\mathcal{E}$ và $\mathcal{D}$ được thực thi bằng cách sử dụng các hàm $EC2OSP_E$ và $OS2ECP_E$. Như vậy mã hóa của điểm là xâu bộ tám có độ dài hoặc bằng 1 hoặc bằng $1+\lceil \log_{256}|F| \rceil$, hoặc $1+2\lceil \log_{256}|F| \rceil$. Tập tất cả các biến định dạng được phép có thể được chọn là tập con không rỗng bất kỳ của tập {*không nén, nén, lai ghép*}. Như vậy nhóm cụ thể được xác định nhờ sử dụng đường cong elliptic, có thể, nhưng không nhất thiết, cho phép sử dụng các định dạng đa mã hóa.

Ánh xạ mã hóa cụ thể $\mathcal{E}'$ được xác định như sau: Cho điểm P trên E, nếu $P = O$ thì đầu ra là $FE2OSP_F(0_F)$ và nếu $P = (x,y) \neq O$, ở đấy $x, y \in F$, thì đầu ra là $FE2OSP_F(x)$. Như vậy đầu ra của $\mathcal{E}'$ là xâu bộ tám độ dài $\lceil \log_{256}|F| \rceil$.

10.2 *ECIES-KEM*

Điều này mô tả cơ chế bọc khóa $ECIES-KEM$.

CHÚ THÍCH $ECIES-KEM$ dựa trên công trình của Adballa, Bellare và Rogaway [1,2].

10.2.1 Các tham số hệ thống

$ECIES-KEM$ là họ các cơ chế bọc khóa, được tham số hóa bởi các tham số hệ thống sau đây:

— Γ: nhóm cụ thể

$$\Gamma = (\mathcal{H}, \mathcal{G}, g, \mu, \nu, \mathcal{E}, \mathcal{D}, \mathcal{E}', \mathcal{D}'),$$

như mô tả trong Điều 10.1;

— KDF: Hàm dẫn xuất khóa, như đã mô tả tại Điều 6.2;

— $CofactorMode$: một trong hai giá trị 0 hoặc 1.

— $OldCofactorMode$: một trong hai giá trị 0 hoặc 1.

— $CheckMode$:một trong hai giá trị 0 hoặc 1.

— $SingleHashMode$: một trong hai giá trị 0 hoặc 1.

— $KeyLen$: số nguyên.

Bất kì một tổ hợp nào các tham số hệ thống đều cho phép, trừ các hạn chế sau:

— Nhiều nhất một trong các tham số $CofactorMode$, $OldCofactorMode$ và $CheckMode$ có thể là 1.

— Nếu $\nu > 1$ và $CheckMode = 0$, khi đó phải có $gcd(\mu, \nu) = 1$.

Giá trị $ECIES-KEM.KeyLen$ được định nghĩa bằng giá trị của tham số hệ thống $Keylen$.

CHÚ THÍCH Các giá trị của $CofactorMode$ và $CheckMode$ chỉ được sử dụng bởi thuật toán giải mã.

10.2.2 Tạo khóa

Thuật toán tạo khóa $ECIES-KEM.KeyGen$ không nhận đầu vào, và chạy như sau:

a) Tạo số ngẫu nhiên $x \in [1 \ldots \mu)$.

b) Tính $h = x.g$.

c) Đưa ra khóa công khai đầu ra:

– h: phần tử khác không của $\mathcal{G}$.

d) Đưa ra khoá riêng đầu ra:

– x: số nguyên thuộc tập hợp [1, ..., μ).

10.2.3 Mã hóa

Thuật toán $ECIES-KEM.Encrypt$ nhận đầu vào là khóa công khai, gồm phần tử $h \in \mathcal{G} \setminus \{0\}$ cùng với tùy chọn mật mã fmt xác định định dạng dùng để mã hóa các phần tử của nhóm. Thuật toán được thực hiện như sau:

a) Tạo số ngẫu nhiên $r \in [1 \ldots \mu)$.

b) Nếu $OldCofactorMode = 1$ thì đặt $r' = r.v \bmod \mu$, ngược lại thì đặt $r' = r$.

c) Tính $\tilde{g} = r.g$ và $\tilde{h} = r'.h$.

d) Đặt $C_0 = \mathcal{E}(\tilde{g}, fmt)$.

e) Nếu $SingleHashMode = 1$ thì Z là xâu bộ tám gồm toàn phần tử 0; ngược lại đặt $Z = C_0$.

f) Đặt $PEH = \mathcal{E}'(\tilde{h})$.

g) Đặt $K = KDF(Z \parallel PEH, KeyLen)$.

h) Đưa ra bản mã C_0 và khóa bí mật K.

10.2.4 Giải mã

Thuật toán giải mã $ECIES-KEM.Decrypt$ nhận đầu vào là khóa bí mật K, gồm phần tử $r \in [1 \ldots \mu)$, và bản mã C_0. Thuật toán thực hiện như sau:

a) Đặt $\tilde{g} = \mathcal{D}(C_0)$; bước này sẽ **thất bại** nếu C_0 không phải là mã hợp lệ của phần tử thuộc $\mathcal{H}$.

b) Nếu $CheckMode = 1$, kiểm tra $\tilde{g} \in \mathcal{G}$, nếu không thoả mãn thì **thất bại**.

c) Nếu $CofactorMode = 1$ hoặc $OldCofactorMode = 1$, đặt $\hat{g} = v.\tilde{g}$, nếu ngược lại thì đặt $\hat{g} = \tilde{g}$.

d) Nếu $Cofactor = 1$, khi đó $\hat{x} = v^{-1}x \bmod \mu$, ngược lại đặt $\hat{x} = x$.

e) Tính $\tilde{h} = \hat{x}.\hat{g}$.

f) Nếu $\tilde{h} = 0$ thì **thất bại**.

g) Nếu $SingleHashMode = 1$ thì cho Z là xâu bộ tám rỗng, ngược lại giả sử $Z = C_0$.

h) Đặt $PEH = \mathcal{E}'(\tilde{h})$.

i) Đặt $K = KDF(Z \parallel PEH, KeyLen)$.

j) Đưa ra là khóa bí mật K

CHÚ THÍCH Sử dụng $CofactorMode = 1$ hoặc $OldCofactorMode = 1$ có thể mang lại lợi ích đáng kể trong thực thi, nếu v thật sự nhỏ. Ưu điểm của việc sử dụng $CofactorMode = 1$ là ở chỗ hành vi của thuật toán mã hóa không bị ảnh hưởng bởi giá trị của $CofactorMode$.

CHÚ THÍCH 2 Khi sử dụng $CofactorMode = 1$, việc thực thi có thể đơn giản là tính trước và lưu giá trị $\hat{x}$ thay vì tính x.

CHÚ THÍCH 3 Khi sử dụng $SingleHashMode = 1$, thậm chí nếu Γ hỗ trợ định dạng đa mã, giá trị của fmt được sử dụng trong khi mã hóa không ảnh hưởng lên bất kì tính toán nào, ngoại trừ định dạng cho bản mã cuối cùng. Như vậy, nếu bản mã cho trước C_0 là kết quả mã hóa của phần tử nhóm $\tilde{g}$, thì một bản mã khác C'_0 đồng thời cũng là kết quả mã hóa (encoding) của $\tilde{g}$ cũng sẽ được giải mã bằng cách như với C_0.

CHÚ THÍCH 4: Việc xem xét tính an toàn của lược đồ này được trình bày trong Phụ lục B.9.

10.3 *PSEC-KEM*

Điều này mô tả cơ chế bọc khóa $PSEC-KEM$.

CHÚ THÍCH $PSEC-KEM$ dựa trên công trình của Fujisaki và Okamoto [26].

10.3.1 Các tham số hệ thống

$PSEC-KEM$ là họ các cơ chế bọc khóa, được tham số hóa bởi các tham số hệ thống sau đây:

— Γ: nhóm cụ thể

$$\Gamma = (\mathcal{H}, \mathcal{G}, g, \mu, v, \mathcal{E}, \mathcal{D}, \mathcal{E}', \mathcal{D}'),$$

được mô tả ở Điều 10.1;

— KDF: Hàm xuất khóa được mô tả ở Điều 6.2;

— $SeedLen$: số nguyên dương;

— $KeyLen$: số nguyên dương.

10.3.2 Tạo khóa

Thuật toán tạo $PSEC-KEM.KeyGen$ không có đầu vào, thực hiện như sau:

a) Tạo số ngẫu nhiên $x \in [0 \ldots \mu)$.

b) Tính $h = x.g$.

c) Đưa ra khóa công khai:

— h: phần tử của $\mathcal{G}$.

d) Đưa ra khóa bí mật:

— x: số nguyên thuộc tập hợp $[0,\ldots,\mu)$.

10.3.3 Mã hóa

Giả sử $I0 = I2OSP(0,4)$ và $I1 = I2OSP(1,4)$.

Thuật toán mã hóa $PSEC-KEM.Encrypt$ nhận đầu vào là khóa công khai, gồm phần tử $h \in \mathcal{G}$, cùng với tùy chọn mật mã fmt, đưa ra định dạng dùng để mã hóa phần tử của nhóm. Thuật toán được thực hiện như sau:

a) Tạo xâu bộ tám ngẫu nhiên $seed$ độ dài $SeedLen$.

b) Tính $t = KDF(I0 \| seed, \lceil \log_{256} \mu \rceil + 16 + KeyLen)$,

là xâu bộ tám độ dài $\lceil \log_{256} \mu \rceil + 16 + KeyLen$

c) Tạo cú pháp cho t, $t = u \parallel K$, ở đây u và K là các xâu bộ tám sao cho $|u| = \lceil log_{256}\mu \rceil + 16$ và $|K| = KeyLen$.

d) Tính $r = OS2IP(u) \bmod \mu$.

e) Đặt $\tilde{g} = r.g, \tilde{h} = r.h$.

f) Đặt $EG = \mathcal{E}(\tilde{g}, fmt)$ và $PEH = \mathcal{E}'(\tilde{h})$.

g) Đặt $SeedMask = KDF(I1 \parallel EG \parallel PEH, SeedLen)$.

h) Đặt $MaskedSeed = seed \oplus SeedMask$.

i) Đặt $C_0 = EG \parallel MaskedSeed$.

j) Đưa ra là bản mã C_0 và khóa bí mật K.

10.3.4 Giải mã

Giả sử $I0 = I2OSP(0, 4)$ và $I1 = I2OSP(1, 4)$.

Thuật toán giải mã $PSEC - KEM.Decrypt$ nhận đầu vào là khóa bí mật K, gồm phần tử $x \in [0 \dots \mu)$ và bản mã C_0. Thuật toán thực hiện như sau:

a) Tạo cú pháp C_0, $C_0 = EG \parallel MaskedSeed$, EG và $MaskedSeed$ là các xâu bộ tám sao cho $|MaskedSeed| = SeedLen$; bước này sẽ **thất bại** nếu $|C_0| < SeedLen$.

b) Đặt $\bar{g} = \mathcal{D}(EG)$; bước này sẽ thất bại nếu EG không phải là mã hợp lệ của phần tử nhóm.

c) Tính $\tilde{h} = x.\tilde{g}$.

d) Đặt $PEH = \mathcal{E}'(\tilde{h})$.

e) Đặt $SeedMask = KDF(I1 \parallel EG \parallel PEH, SeedLen)$.

f) Đặt $seed = MaskedSeed \oplus SeedMask$.

g) Tính

$$t = KDF(I0 \| seed, \lceil \log_{256} \mu \rceil + 16 + KeyLen)$$

là xâu bộ tám độ dài $\lceil log_{256}\mu \rceil + 16 + KeyLen$.

h) Tạo cú pháp t, $t = u \,||K$ ở đây u và K là xâu bộ tám thỏa mãn $|u| = \lceil log_{256}\mu \rceil + 16$ và $|K| = KeyLen$.

i) Tính $r = OS2IP(u) \bmod \mu$.

j) Tính $\bar{g} = r.g$.

k) Kiểm tra $\bar{g} = \tilde{g}$, nếu không phải thì **thất bại**.

l) Đưa ra khóa bí mật K.

CHÚ THÍCH Việc xem xét tính an toàn của lược đồ trên có thể tìm tại phụ lục B.10.

10.4 *ACE-KEM*

Điều này mô tả cơ chế bọc khóa $ACE - KEM$.

CHÚ THÍCH $ACE - KEM$ được xây dựng dựa trên công trình của Cramer và Shoup [13,14].

10.4.1 Các tham số hệ thống

$ACE - KEM$ là họ các cơ chế bọc khóa, được tham số hoá bởi các tham số hệ thống sau:

— Γ: nhóm cụ thể

$$\Gamma = (\mathcal{H}, \mathcal{G}, g, \mu, \nu, \mathcal{E}, \mathcal{D}, \mathcal{E}', \mathcal{D}'),$$

được mô tả tại Điều 10.1;

— KDF: hàm xuất khóa được mô tả ở Điều 6.2;

— Hàm băm mật mã được mô tả ở Điều 6.1;

— $CofactorMode$: một trong hai giá trị 0 hoặc 1.

— $KeyLen$: số nguyên dương.

Bất kì sự kết hợp nào của các tham số hệ thống được phép đều cho phép, ngoại trừ các hạn chế sau:

— $HashLen$: phải nhỏ hơn $log_{256}\mu$.

— Nếu $\nu = 1$ thì $CofactorMode$ sẽ bằng 0.

— Nếu $\nu > 1$ $CofactorMode$ có thể bằng 1 với điều kiện $gcd(\mu, \nu) = 1$.

CHÚ THÍCH Giá trị của $Cofactormode$ chỉ được sử dụng bởi thuật toán giải mã.

10.4.2 Tạo khóa

Thuật toán tạo khóa $ACE - KEM.KeyGen$ không có đầu vào, thực hiện như sau:

a) Tạo số ngẫu nhiên $w, x, y, z \in [0 \dots \mu)$.

b) Tính các phần tử của nhóm

$$g' = w.g, c = x.g, d = y.g, h = z.g.$$

c) Đưa ra khóa công khai:

— g', c, d, h: các phần tử thuộc $\mathcal{G}$.

d) Đưa ra khóa riêng:

— w, x, y, z: các số nguyên thuộc tập $[0 \dots \mu)$.

10.4.3 Mã hóa

Thuật toán mã hóa $ACE = KEM.Encrypt$ nhận đầu vào là khóa công khai, gồm

$$g', c, d, h \in \mathcal{G},$$

cùng tùy lựa chọn mật mã fmt xác định định dạng dùng để mã hóa phần tử nhóm. Thuật toán thực hiện như sau:

a) Tạo số ngẫu nhiên $r \in [0..\mu)$.

b) Tính các phần tử nhóm

$$u = r.g, u' = r.g', \bar{h} = r.h.$$

c) Tính các xâu bộ tám

$$EU = \mathcal{E}(u, fmt), EU' = \mathcal{E}(u', fmt).$$

d) Tính số nguyên

$$\alpha = OS2IP\big(Hash.eval(EU \parallel EU')\big).$$

e) Tính số nguyên

$$r' = \alpha.r \bmod \mu.$$

f) Tính phần tử nhóm

$$v = r.c + r'.d.$$

g) Đặt $EV = \mathcal{E}(v, fmt)$.

h) Đặt $PEH = \mathcal{E}'(\tilde{h})$.

i) Đặt $C_0 = EU \parallel EU' \parallel EV$.

j) Đặt $K = KDF(EU \parallel PEH, KeyLen)$.

k) Đưa ra là bản mã C_0 và khóa bí mật K.

10.4.4 Giải mã

Thuật toán $ACE - KEM.Decrypt$ nhận đầu vào là khóa bí mật, gồm $w, x, y, z \in [0...\mu)$ và bản mã C_0. Thuật toán chạy như sau:

a) Tạo cú pháp cho C_0, $C_0 = EU \parallel EU' \parallel EV$, ở đấy EU, EU', và EV là các xâu bộ tám sao cho đối với một số phần tử nhóm (xác định duy nhất) $u, u', v \in H$, ta có $u = \mathcal{D}(EU)$, $u' = \mathcal{D}(EU')$, $v = \mathcal{D}(EV)$. Bước này sẽ **thất bại** nếu C_0 không được tạo cú pháp như vậy.

b) Kiểm tra tập hợp $\{EU, EU', EV\}$ có phải là tập bền vững các mã hợp lệ không, nếu không thì **thất bại**.

c) Nếu $CofactorMode = 1$, đặt

$\hat{u} = v.u, \hat{w} = v^{-1}w \bmod \mu, \hat{x} = v^{-1}x \bmod \mu, \hat{y} = v^{-1}y \bmod \mu, \hat{z} = v^{-1}z \bmod \mu$;

Ngược lại, thì đặt $\hat{u} = u, \hat{w} = w, \hat{x} = x, \hat{y} = y, \hat{z} = z$.

d) Nếu $CofactorMode \neq 1$ và $v > 1$: kiểm tra $u \in \mathcal{G}$ nếu $u \notin \mathcal{G}$ thì **thất bại**.

e) Tính số nguyên

$$\alpha = OS2IP\big(Hash.eval(EU \parallel EU')\big).$$

f) Tính số nguyên $t = \hat{x} + \hat{y}\alpha \bmod \mu$

g) Kiểm tra

$$\hat{w}.\hat{u} = u'; t.\hat{u} = v.$$

nếu không thỏa mãn, thì **thất bại**.

h) Tính phần tử nhóm $\tilde{h} = \hat{z}.\hat{u}$.

i) Đặt $PEH = \mathcal{E}'(\tilde{h})$.

j) Đặt $K = KDF(EU \parallel PEH, KeyLen)$.

k) Đưa ra là khóa bí mật K.

Vì lí do an toàn, khuyến cáo rằng việc thực thi không được để lộ bất kì thông tin nào về lý do sinh ra lỗi tại bước g. Nói riêng, việc thực thi có thể đưa ra cùng một thông báo lỗi tại cùng thời điểm bất kể lý do sinh ra lỗi.

CHÚ THÍCH 1 Sử dụng $CofactorMode = 1$ có thể mang lại lợi ích cho thực thi nếu v đủ nhỏ. Chú ý rằng ở chế độ này việc thực thi có thể tính toán trước và lưu trữ giá trị $\hat{w},\hat{x},\hat{y},\hat{z}$ thay vì tính các giá trị w, x, y, z.

CHÚ THÍCH 2 Việc thực thi không phụ thuộc vào việc sử dụng phiên bản tương đương về chức năng sau đây của thuật toán giải mã. Trong thực thi không cần thiết phải tính u' và v tại bước a của thật toán giải mã, mà đơn giản là tạo ra cú pháp C_0 sau khi nhận được EU, EU', và EV, và chỉ biến đổi EU thành phần tử nhóm u. Bước b có thể bỏ qua.Tiếp đó việc kiểm tra tại bước g của thuật toán giải mã thực hiện như sau: Nếu $u = 0$ thì kiểm tra EU' và EV có phải là mã (duy nhất) của 0 hay không; nếu không thì giả sử fmt là biến định dạng của EU và kiểm tra các điều kiện $\mathcal{E}(w.\hat{u}, fmt) = EU'$ và $\mathcal{E}(t,\hat{u}, fmt) = EV$.

CHÚ THÍCH 3 Việc xem xét chi tiết tính an toàn của lược đồ này được đề cập ở Phụ lục B.11.

11 Mật mã phi đối xứng dựa trên RSA và các cơ chế bọc khóa

Điều này mô tả mật mã phi đối xứng và các cơ chế bọc khóa dựa trên biến đổi RSA. Mã $RSAES$ được mô tả tại Điều11.4; cơ chế bọc khóa $RSA - KEM$ được mô tả ở Điều 11.5.

CHÚ THÍCH 1 Các lược đồ này là sự cải biên của lược đồ mật mã gốc RSA [31].

CHÚ THÍCH 2 Trong một số tiêu chuẩn ISO khác, thuật ngữ "phân tích số" được sử dụng ở những nơi "dựa vào RSA"; tuy nhiên, vì tiêu chuẩn này xác định một số lược đồ khác nhau dựa trên phân tích số, nên nó chấp nhận thỏa thuận đặt tên mới.

11.1 Các thuật toán tạo khóa RSA

Thuật toán tạo khóa RSA, $RSAKeyGen()$ là thuật toán xác suất không có đầu vào, đầu ra là bộ ba (n, e, d), ở đây:

– Số nguyên n là tích của hai số nguyên tố p và q, có độ dài tương tự nhau, $p \neq q$,

– e là số nguyên dương thỏa mãn điều kiện $\gcd(e,(p-1)(q-1)) = 1$ và

– d là số nguyên dương thỏa mãn $ed \equiv 1 (\mathrm{mod}\, \lambda(n))$ ở đây $\lambda(n)$ là bội số chung nhỏ nhất của $(p-1)$ và $(q-1)$.

Phân phối đầu ra của thuật toán tạo khóa RSA phụ thuộc vào thuật toán cụ thể. Thuật toán này được phép cho kết quả đầu ra không thỏa mãn các điều kiện trên đây, chừng nào điều đó còn xảy ra với xác suất không đáng kể.

CHÚ THÍCH 1 Trong việc mô tả mật mã dựa trên RSA, các mật mã này được tham số hóa bằng thuật ngữ $RSAKeyGen$; ví dụ $RSAKeyGen$ được coi như một tham số hệ thống của mật mã. Trong những thực thi thông thường, thuật toán tạo khóa cụ thể có thể được chọn từ họ các thuật toán, được tham số hóa bởi "tham số an toàn" (ví dụ, độ dài của n).

CHÚ THÍCH 2 Xem ISO/IEC 18033 hướng dẫn thuật toán tạo các số nguyên tố p và q được đề cập trên đây.

11.2 Biến đổi RSA

Thuật toán $RSATransform\ (X, \alpha, n)$ nhận đầu vào là

— Xâu bộ tám X,

— Số nguyên α, và

— Số nguyên n,

Và đầu ra là xâu bộ tám. Thuật toán thực hiện như sau:

a) Kiểm tra việc thực hiện $|X| = \mathcal{L}(n)$; nếu không thực hiện thì **thất bại**.

b) Đặt $x = OS2IP(X)$.

c) Kiểm tra $x < n$; nếu không thỏa mãn thì **thất bại**.

d) Đặt $y = x^{\alpha} \ mod \ n$.

e) Đặt $Y = OS2IP(y, \mathcal{L}(n))$.

f) Đưa ra là Y.

CHÚ THÍCH Như đã biết, nếu (*n, e, d*) là đầu ra của thuật toán tạo khóa RSA và $X = I2OSP(x, \mathcal{L}(n))$ với số nguyên *x* nào đó thỏa mãn 0 ≤ x ≤ n, khi đó

$$RSATransform\ (RSATransform(X, e, n), d, n) = X.$$

11.3 Các cơ chế mã hóa RSA

Cơ chế mã hóa RSA, REM xác định hai thuật toán:

— $REM.Encode(M, L, ELen)$ nhận đầu vào là bản rõ M, nhãn L và độ dài đầu ra là $ELen$. Ở đây, M và L là các xâu bộ tám với độ dài bị hạn chế được mô tả dưới đây. Đầu ra là xâu bộ tám E độ dài $Elen$.

— $REM.Decode(E, L)$ nhận đầu vào là xâu bộ tám E, nhãn L và tìm bản rõ M thỏa mãn $REM.Encode(M, L, |E|) = E$. Nếu không đưa ra được M như vậy thì **thất bại**.

Ngoài ra, cơ chế cũng nên xác định giới hạn $REM.Bound$, sao cho khi gọi $REM.Encode(M, L, ELen)$, thì điều kiện $|M| \leq ELen - REM$ sẽ được thỏa mãn. Nếu không thuật toán mã hóa sẽ thất bại. Ngoài ra thuật toán mã hóa cũng có thể thất bại, nếu $|L|$ vượt quá giới hạn được xác định bởi quá trình thực thi.

Nói chung thuật toán $REM.Encode$ là thuật toán xác suất, bởi cùng một bản rõ có thể được mã hóa bằng một số cách khác nhau. Đồng thời vì lí do kỹ thuật, yêu cầu bộ tám đầu tiên của đầu ra phải là $Oct(0)$.

11.3.1 Các cơ chế mã hóa RSA được phép

Trong phần này của ISO/IEC, Cơ chế mã hóa RSA được phép là $REM1$, được mô tả dưới đây ở Điều 11.3.2.

11.3.2 *REM1*

Điều này mô tả cơ chế mã hóa cụ thể được gọi là $REM1$.

CHÚ THÍCH $REM1$ được xây dựng trên cơ sở OAEP của Bellare và Rogaway [8].

11.3.2.1 Các tham số hệ thống

$REM1$ là họ các cơ chế mã hóa RSA được tham số hóa bởi các tham số hệ thống sau đây:

– $Hash$: hàm băm mật mã được mô tả tại Điều 6.1;

– KDF: hàm dẫn xuất khóa được mô tả ở Điều 6.2.

Giá trị $REM.Bound$ được xác định bằng:

$$REM1.Bound = 2.Hash.len + 2.$$

11.3.2.2 Hàm mã hóa

Hàm mã hóa $REM1.Encode(M, L, ELen)$ chạy như sau:

a) Kiểm tra $|M| \leq ELen - 2.Hashlen - 2$, nếu không thì **thất bại**.

b) Giả sử phần đệm *(pad)* là xâu bộ tám độ dài $ELen - |M| - 2.Hashlen - 2$ gồm một xâu bộ tám $Oct(0)$.

c) Tạo mầm là xâu bộ tám ngẫu nhiên mầm độ dài $Hash.len$.

d) Đặt $check = Hash.eval(L)$.

e) Đặt $DataBlock = check \| pad \| \langle Oct(1) \rangle \| M$

f) Đặt $DataBlockMask = KDF(seed, Elen - Hash.len - 1)$.

g) Đặt $MaskedDataBlock = DataBlockMask \oplus DataBlock$.

h) Đặt $SeedMask = KDF(MaskedDataBlock, Hash.len)$.

i) Đặt $MaskedSeed = SeedMask \oplus seed$.

j) Đặt $E = \langle Oct(0) \rangle \| MaskedSeed \| MaskedDataBlock$.

k) Đưa ra E.

11.3.2.3 Hàm giải mã (Decoding function)

Thuật toán $REM1.Decode(E, L)$ chạy như sau:

a) Giả sử $Elen = |E|$.

b) Kiểm tra $ELen \geq 2.Hashlen + 2$; nếu không thì **thất bại**.

c) Đặt $Check = Hash.eval(L)$.

d) Tạo cú pháp E, $E = \langle X \rangle \| MaskedSeed \| MaskedDataBlock$; ở đây X là bộ tám và $MaskedSeed$ và $MaskedDataBlock$ là các bộ tám thỏa mãn hệ thức $|MaskedSeed| = Hashlen, |MaskedDataBlock| = Elen - Hashlen - 1$.

e) Đặt $SeedMask = KDF(MaskedDataBlock, Hash.len)$.

f) Đặt $seed = MaskedSeed \oplus SeedMask$.

g) Đặt $DataBlockMask = KDF(seed, Elen - Hash.len - 1)$.

h) Đặt $DataBlock = MaskedDataBlock \oplus DataBlockMask$.

i) Tạo cú pháp cho $DataBlock$, $DataBlock = check' \| M'$; $Check'$ và M' là các xâu bộ tám với $|check'| = Hash.len; |M'| = ELen - 2.Hash.len - 1$.

j) Giả sử $M' = \langle M_1, M_2, ..., M_l \rangle$ là các bộ tám và $l = Elen - 2.Hash.len - 1$; đồng thời giả sử m là số nguyên dương lớn nhất sao cho $m \leq l$, và $M_1 = M_2 = \cdots M_{m-1} = Oct(0)$ và T là kí hiệu bộ tám M_m, giả sử M là kí hiệu xâu bộ tám $\langle M_{m+1}, ..., M_l \rangle$.

k) Nếu $check' \neq check$, $X \neq Oct(0)$, hoặc $T \neq Oct(1)$ thì **thất bại**.

l) Đưa ra M.

Vì lí do an toàn, điều quan trọng là trong quá trình thực thi không để lộ bất kì thông tin nào về nguyên nhân phát sinh lỗi tại bước k. Nói riêng, thực thi có thể cho ra thông báo cùng lỗi tại cùng thời điểm, bất kể nguyên nhân việc sinh ra lỗi.

11.4 *RSAES*

11.4.1 Các tham số hệ thống

RSAES là họ các mã phi đối xứng với độ dài bản rõ bị hạn chế, được tham số hóa bởi các tham số hệ thống sau:

— $RSAKeyGen$: Thuật toán tạo khóa RSA được mô tả tại Điều 11.1;

— REM: cơ chế mã hóa RSA được mô tả tại Điều 11.3.

Bất kì một tổ hợp các tham số hệ thống nào cũng được phép với các hạn chế sau:

— Độ dài tính bằng octet của đầu ra n của $RSAKeyGen()$ phải luôn luôn lớn hơn $REM.Bound$.

11.4.2 Tạo khóa

Thuật toán $RSAES.KeyGen$ không có đầu vào, chạy như sau:

a) Tính $(n, e, d) = RSAKeyGen()$.

b) Khóa công khai đầu ra PK:

— n: số nguyên dương.

— e: số nguyên dương.

c) Đưa ra khóa riêng pk:

— n: số nguyên dương.

— d: số nguyên dương.

RSAES là mật mã phi đối xứng với độ dài bản rõ bị hạn chế. Đối với khóa công khai cho trước $PK = (n, e)$, giá trị của $RSAES.MaxMsgLen(PK)$ là $\mathcal{L}(n) - REM.Bound$.

Các thuật toán mã hóa và giải mã đều sử dụng biến đổi thuật toán RSA được định nghĩa tại Điều 11.2.

11.4.3 Mã hóa

Thuật toán $RSAES.Encrypt$ nhận đầu vào;

– Khóa công khai, gồm số nguyên dương n và số nguyên dương e,

– Nhãn L,

– Bản rõ M với độ dài lớn nhất $\mathcal{L}(n) - REM.Bound$, và

– Không có tùy chọn mật mã.

Thuật toán chạy như sau:

a) Đặt $E = REM.Encode(M, L, \mathcal{L}(n))$.

b) Đặt $C = RSATransform(E, e, n)$.

c) Đưa ra C.

11.4.4 Giải mã

Thuật toán $RSAES.Decrypt$ nhận đầu vào:

– Khóa riêng, gồm số nguyên dương n và số nguyên dương e,

– Nhãn L, và

– Bản mã C.

Thuật toán chạy như sau:

a) Đặt $E = RSATransform(C, d, n)$; chú ý rằng bước này cũng có thể **thất bại**.

b) Đặt $M = REM.Decode(E, L)$; chú ý rằng bước này cũng có thể **thất bại**.

c) Đưa ra M.

CHÚ THÍCH Tính an toàn của *RSAES* được xem xét tại Phụ lục B.13.

11.5 *RSA-KEM*

11.5.1 Các tham số hệ thống

$RSA - KEM$ là họ các cơ chế bọc khóa, được tham số hóa bởi các tham số hệ thống sau đây:

– $RSAKeyGen$: Thuật toán tạo khóa RSA, được mô tả ở Điều 11.1;

– KDF: hàm dẫn xuất khóa được mô tả ở Điều 6.2;

– $KeyLen$: số nguyên dương.

Giá trị của $RSA - KEM.KeyLen$ được xác định bằng giá trị của tham số hệ thống $KeyLen$.

11.5.2 Tạo khóa

Thuật toán $RSA - KEM.KeyGen$ không có đầu vào, chạy như sau:

a) Tính $(n, e, d) = RSAKeyGen()$.

b) Đưa ra khóa công khai PK:

– n: số nguyên dương.

– e: số nguyên dương.

c) Đưa ra khóa riêng pk:

— n: số nguyên dương.

— d: số nguyên dương.

Thuật toán mã hóa và giải mã sử dụng thuật toán biến đổi RSA được định nghĩa tại Điều 11.2.

11.5.3 Mã hóa

Thuật toán mã hóa $RSA - KEM.Encrypt$ nhận đầu vào là:

— Khóa công khai, bao gồm các số nguyên dương n và e, và

— Không có tùy chọn mật mã.

Thuật toán chạy như sau:

a) Tạo số ngẫu nhiên $r \in [0..n)$.

b) Đặt $R = I2OSP(r, \mathcal{L}(n))$.

c) Đặt $C_0 = RSATransform(R, e, n)$.

d) Tính $K = KDF(R, KeyLen)$.

e) Đưa ra là bản rõ C_0 và khóa bí mật K.

11.5.4 Giải mã

Thuật toán $RSA - KEM.Decrypt$ nhận đầu vào:

— Khóa mật gồm các số nguyên dương n và d

— Bản mã C_0.

Thuật toán chạy như sau:

a) Đặt $R = RSATransform(C_0, d, n)$; chú ý rằng bước này cũng có thể **thất bại**.

b) Tính $K = KDF(R, KeyLen)$.

c) Đầu ra là khóa bí mật K.

CHÚ THÍCH Tính an toàn của $RSA - KEM$ được xem xét tại phụ lục B.14.

12 Mật mã dựa trên phép bình phương modulo

Điều này mô tả họ các mật mã phi đối xứng dựa trên phép bình phương modulo. Mã $HIME(R)$ được mô tả tại Điều 12.3.

12.1 Các thuật toán tạo khóa *HIME*

Với số dương l và d >1 và thuật toán tạo khóa l –bit $HIME - HIMEKeyGen$ là thuật toán xác suất không có đầu vào, đầu ra là các số nguyên dương (p, q, n, d),ở đấy:

— p là số nguyên tố, với $2^{l-1} \leq p < 2^l$; $p \equiv 3 \ (mod\ 4)$,

– q là số nguyên tố với $2^{l-1} \leq q < 2^l$; $q \equiv 3 \pmod 4$ và $p \neq q$,

– $n = p^d q$.

Phân phối đầu ra của thuật toán tạo khóa *HIME* bit phụ thuộc vào thuật toán cụ thể. Thuật toán này được phép cho kết quả đầu ra không thỏa mãn các điều kiện trên đây, chừng nào điều đó còn xảy ra với xác suất không đáng kể.

CHÚ THÍCH 1 Trong mô tả mật mã dựa trên *HIME*, những lược đồ này được tham số hóa theo thuật ngữ của *HIMEKeyGen*, nghĩa là *HIMEKeyGen* được xem như tham số hệ thống của mật mã.

CHÚ THÍCH 2 Tham khảo ISO/IEC 18032 về hướng dẫn thiết kế thuật toán tạo các số nguyên tố p và q ở trên.

12.2 Các cơ chế mã hóa *HIME*

Cơ chế mã hóa theo *HIME*, kí hiệu *HIME* là *HEM* đưa ra hai thuật toán:

– $HEM.Encode(M, L, ELen, KLen)$ nhận đầu vào là bản rõ M, nhãn L, độ dài đầu ra $Elen$, và số nguyên dương $KLen$. M và L là các xâu bộ tám với độ dài bị giới hạn, được mô tả dưới đây. $KLen$ thỏa mãn điều kiện $1 \leq KLen \leq 8$. Đầu ra là xâu bộ tám E độ dài $ELen$.

– $HEM.Decode(E, L, KLen)$ nhận đầu vào là xâu bộ tám E, nhãn L, và số nguyên dương $KLen$. Đầu ra là bản rõ M sao cho $HEM.Encode(M, L, |E|, KLen) = E$. Thuật toán sẽ đưa ra M, nếu không thì thất bại.

12.2.1 Các cơ chế mã hóa *HIME* được phép

Trong phần này của ISO/IEC 18033, cơ chế mã hóa *HIME* được phép là *HEM1*, được mô tả ở Điều 12.2.2 dưới đây.

12.2.2 *HEM1*

Điều này mô tả một cơ chế mã hóa *HIME* cụ thể, được gọi là *HEM1*.

CHÚ THÍCH *HEM1* dựa trên OAEP do Bellare và Rogaway xây dựng [8].

12.2.2.1 Các tham số hệ thống

HEM1 là họ các cơ chế mã hóa *HEM* được tham số hóa bởi các tham số hệ thống sau:

– $Hash$: hàm băm mật mã được mô tả tại Điều 6.1;

– KDF: hàm dẫn xuất khóa được mô tả tại Điều 6.2.

Độ lớn của $HEM1.Bound$ được định nghĩa bằng

$$HEM1.Bound = 2.Hash.len + 2.$$

12.2.2.2 Hàm mã hóa

Thuật toán $HEM1.Encode(M, L, ELen, KLen)$ chạy như sau:

a) Kiểm tra điều kiện $|M| \leq ELen - 2.Hashlen - 2$, nếu không thỏa mãn thì **thất bại**.

b) Giả sử phần *pad* là xâu bộ tám độ dài $ELen - |M| - 2.HashLen - 2$ gồm một xâu octet Oct(0).

c) Tạo xâu bộ tám ngẫu nhiên *seed* với độ dài $Hash.len + 1$.

d) Xóa đi $KLen - bit$ có nghĩa nhất của *seed*.

e) Đặt $check = Hash.eval(L)$.

f) Đặt $DataBlock = check \| pad \| \langle Oct(1) \rangle \| M$

g) Đặt $DataBlockMask = KDF(seed, ELen - HashLen - 1)$.

h) Đặt $MaskedDataBlock = DataBlockMask \oplus DataBlock$.

i) Đặt $SeedMask = KDF(MaskedDataBlock, Hash.len + 1)$.

j) Xóa đi $KLen - bit$ có nghĩa nhất của $SeedMask$.

k) Đặt $MaskedSeed = SeedMask \oplus seed$.

l) Đặt $E = MaskedSeed \| MaskedDataBlock$.

m) Đưa ra E.

12.2.2.3 Hàm giải mã

Thuật toán $HEM1.Decode(E, L, KLen)$ chạy như sau:

a) Đặt $ELen = |E|$.

b) Đặt $check = Hash.eval(L)$.

c) Tạo cú pháp E, $E = MaskedSeed \| MaskedDataBlock$, $MaskedDataBlock$ và $MaskedSeed$ là các xâu bộ tám thỏa mãn điều kiện $|MaskedSeed| = Hash.len + 1$ và $|MaskedDataBlock| = Elen - Hash.len - 1$.

d) Đặt $SeedMask = KDF(MaskedDataBlock, Hash.len + 1)$.

e) Xóa $Klen - bit$ có nghĩa nhất của $SeedMask$.

f) Đặt $seed = MaskedSeed \oplus SeedMask$.

g) Đặt $DataBlockMask = KDF(seed, ELen - Hash.len - 1)$.

h) Đặt $DataBlock = MaskedDataBlock \oplus DataBlockMask$.

i) Tạo cú pháp cho $DataBlock$, $DataBlock = check' || M'$ ở đây $check'$ và M' là các xâu bộ tám thỏa mãn điều kiện $|check'| = Hash.len$ và $|M'| = ELen - 2.Hash.len - 1$.

j) Đặt $M' = \langle M_1, ..., M_l \rangle$, ở đây $M_1, ..., M_l$ là các bộ tám và $l = Elen - 2.Hash.len - 1$; đồng thời giả sử m là số nguyên dương lớn nhất sao cho $m \leq l$, và $M_1 = M_2 = ... = M_{m-1} = Oct(0)$ và T là kí hiệu M_m, M kí hiệu xâu bộ tám $\langle M_{m+1}, ..., M_l \rangle$.

k) Nếu $check' \neq check$, phần $KLen - bit$ có nghĩa nhất nhất của $seed \neq$ xâu bit gồm các số 0, hoặc $T \neq Oct(1)$, thì **thất bại**.

l) Đưa ra M.

Vì lí do an toàn, điều quan trọng là trong quá trình thực thi không để lộ bất kì thông tin nào về nguyên nhân phát sinh lỗi tại bước k. Nói riêng, việc thực thi có thể đưa ra cùng thông báo lỗi tại cùng thời điểm bất kể nguyên nhân sinh ra lỗi.

12.3 *HIME (R)*

12.3.1 Các tham số hệ thống

$HIME(R)$ là họ các mật mã phi đối xứng với độ dài bản rõ bị hạn chế, được tham số hóa bởi các tham số hệ thống sau:

— d: số nguyên >1,

— $HIMEKeyGen$: thuật toán tạo khóa $HIME$ l -bit được mô tả tại Điều 12.1;

— HEM: cơ chế mã hóa $HIME$ được mô tả ở Điều 12.2.

12.3.2 Tạo khóa

Thuật toán $HIME(R).KeyGen$ không có đầu vào, chạy như sau:

a) Tính $(p, q, n) = HIMEKeyGen()$.

b) Đưa ra khóa công khai PK:

— n: số nguyên dương.

c) Đưa ra khóa riêng pk:

— n, p, q: các số nguyên dương.

12.3.3 Mã hóa

Thuật toán mã hóa $HIME(R).Encrypt$ nhận đầu vào là

— khóa công khai là số nguyên dương n,

— Nhãn L,

— Bản rõ M với độ dài lớn nhất là $\mathcal{L}(n) - HEM.Bound$, và

— Không có tùy chọn mật mã.

Thuật toán chạy như sau:

a) Đặt $k = 8.\mathcal{L}(n) - (\text{độ dài bit của } n) + 1$.

b) Đặt $E = HEM.Encode(M, L, \mathcal{L}(n), k)$.

c) Đặt $e = OS2IP\ (E)$.

d) Đặt $c = e^2 mod\ n$.

e) Đặt $C = I2OSP(c, \mathcal{L}(n))$.

f) Đưa ra C.

12.3.4 Giải mã

Thuật toán giải mã $HIME(R).Decrypt$ nhận đầu vào là:

– Khóa riêng gồm các số nguyên dương n, p, q,

– Nhãn L,

– Bản mã C.

Thuật toán chạy như sau:

a) Đặt $c = OS2IP(C)$.

b) Đặt k= 8.$\mathcal{L}(n)$-(độ dài bit của n) + 1.

c) Đặt $z = p^{-1} \bmod q$.

d) Đặt $c_p = c \bmod p$, $c_q = c \bmod q$.

e) Đặt $\alpha_1 = c_p^{\frac{p+1}{4}} \bmod p, \alpha_2 = p - \alpha_1$.

f) Đặt $\beta_1 = c_q^{\frac{p+1}{4}} \bmod q, \beta_2 = q - \beta_1$.

g) Đặt

1) $u_0^{(1)} = \alpha_1$, và $u_1^{(1)} = \left(\beta_1 - u_0^{(1)}\right) z \bmod q$.

2) $u_0^{(2)} = \alpha_1$, và $u_1^{(2)} = \left(\beta_2 - u_0^{(2)}\right) z \bmod q$.

3) $u_0^{(3)} = \alpha_2$, và $u_1^{(3)} = \left(\beta_1 - u_0^{(3)}\right) z \bmod q$.

4) $u_0^{(4)} = \alpha_2$, và $u_1^{(4)} = \left(\beta_2 - u_0^{(4)}\right) z \bmod q$.

h) Với i từ 1 đến 4, thực hiện:

1) Đặt $v_1^{(i)} = u_0^{(i)} + u_1^{(i)} p$.

2) Với t từ 2 đến d, thực hiện:

i) Đặt $u_t^{(i)} = ((c - {v_{t-1}^{(i)}}^2 \bmod p^t q)/(p^{t-1} q))(2u_0^{(i)})^{-1} \bmod p$.

ii) Đặt $v_t^{(i)} = v_{t-1}^{(i)} + u_t^{(i)} p^{t-1} q$.

3) Đặt $x_i = u_0^{(i)} + u_1^{(i)} p + \sum_{t=2}^{d} u_t^{(i)} p^{t-1} q$.

i) Với i từ 1 đến 4, đặt $X_i = I2OSP\left(x_i, \mathcal{L}(n)\right)$.

j) Nếu tồn tại duy nhất i sao cho $HEM.Decode(X_i, L, k)$ không bị thất bại và $x_i^2 \bmod n = c$, thì với i đó đặt $M = HEM.Decode(X_i, L, k)$, nếu không thì **thất bại**.

k) Đưa ra M.

CHÚ THÍCH Việc xem xét tính an toàn của lược đồ này được trình bày tại Phụ lục B.15.

Phụ lục A

(Quy định)

Cú pháp ASN.1 cho các bộ định danh đối tượng

Phụ lục cung cấp cú pháp cho các bộ định danh đối tượng, khóa công khai, và các cấu trúc tham số đi kèm với các thuật toán được đặc tả trong phần này của ISO/IEC 18033.

```
--#####################################################################
EncryptionAlgorithms-2 {

     iso(1) standard(0) encryption-algorithms(18033) part(2)

          asn1-module(0) algorithm-object-identifiers(0) }

     DEFINITIONS EXPLICIT TAGS ::= BEGIN

--xuất ra toàn bộ --

IMPORTS

BlockAlgorithms

FROM EncryptionAlgorithms-3 { iso(1) standard(0)

encryption-algorithms(18033) part(3)

asn1-module(0) algorithm-object-identifiers(0) }

HashFunctionAlgs, id-sha1, NullParms

FROM DedicatedHashFunctions { iso(1) standard(0)

hash-functions(10118) part(3)

asn1-module(1) dedicated-hash-functions(0) };


--#####################################################################

-- Định nghĩa oid --

OID ::= OBJECT definition -- alias

--đồng nghĩa--

is18033-2 OID ::= { iso(1) standard(0) is18033(18033) part2(2)}

id-ac OID : = { is18033-2 asymmetric-cipher(1) }

id-kem OID := { is18033-2 key-encapsulation-mechanism(2) }

id-dem OID := { is18033-2 data- encapsulation -mechanism(3) }

id-sc OID := { is18033-2 symmetric-cipher(4) }

id-kdf OID = { is18033-2 key-derivation-function(5) }

id-rem OID = { is18033-2 rsa-encoding-method(5) }
```

```
id-hem OID := { is18033-2 himer-encoding-method(7) }

id-ft OID : = { is18033-2 fleid-type(8) }

--Hệ mật phi đối xứng--

id-ac-rsaes OId ::= { id-ac rsaes(1) }

id-ac-generic-hybrid OID ::= { id-ac generic-hybrid(2) }

id-ac-himer OID ::= { id-ac himer(3) }

-- Cơ chế đóng gói khóa --

id-kem-ecies OID ::= { id-kem ecies(1) }

id-kem-psec OID ::= { id-kem psec(2) }

id-kem-ace OID ::= { id-kem ace(3) }

id-kem-rsa OID ::= { id-kem rsa(4) }

-- Cơ chế đóng gói dữ liệu --

id-dem-dem1 OID ::= { id-dem dem1(1) }

id-dem-dem2 OID ::= { id-dem dem2(2) }

id-dem-dem3 OID ::= { id-dem dem3(3) }

-- Hệ mật đối xứng --

id-sc-sc1 OID ::= { id-sc sc1(1) }

id-sc-sc2 OID ::= { id-sc sc2(2) }

-- Hàm khóa gốc --

id-kdf-kdf1 OID ::= { id-kdf kdf1(1) }

id-kdf-kdf2 OID ::= { id-kdf kdf2(2) }

-- Phương pháp mã hóa rsa --

id-rem-rem1 OID ::= { id-rem rem1(1) }

-- Phương pháp mã hó hime(r)--

id-hem-hem1 OID ::= { id-hem hem1(1) }

-- Các dạng trường mới oids

-- id-ft-prime-field OID ::= { id-ft prime-field(1) }

-- Chỉ sử dụng để xác định dạng cơ sở mới

id-ft-characteristic-two OID ::= { id-ft characteristic-two(2) }

id-ft-odd-characteristic OID ::= { id-ft odd-characteristic(3) }
```

```
id-ft-characteristic-two-basis OID ::=

{ id-ft-characteristic-two basisType(1) }

charTwoPolynomialBasis OID ::=

{ id-ft-characteristic-two-basis

charTwoPolynomialBasis(1) }

id-ft-odd-characteristic-basis OID ::= { id-ft-odd-characteristic
basisType(1)}

oddCharPolynomialBasis OID ::= {id-ft-odd-characteristic-basis
oddCharPolynomialBasis(1)}

--#####################################################################

-- Chú thích viện dẫn:

-- Bất cứ khi nào giá trị của cấu trúc khóa công khai

-- được tiến hành trong cấu trúc SubjectPublicKeyinfo được quy định

-- tại X.509

-- giá trị của subjectPublicKey sẽ là xâu bit

-- tương ứng với bảng mã DER của cấu trúc khóa công khai và

-- giá trị của trường thuật toán sẽ là định danh thuật toán đó

-- (được xác định trong mô-đun này)với khóa công khai

-- được dự kiến

--#####################################################################


-- hệ mật phi đối xứng RSAES

rsaes ALGORITHM ::= {

OID id-RSAES-OAEP PARMS RsaesParameters

}

RsaesPublicKey ::= RSAPublicKey

-- nhận được từ PKCS#1 RSAPublicKey ::= SEQUENCE {

modulus INTEGER,-- n

publicExponent INTEGER -- e

}
```

```
-- Trường pSource trong PKCS #1 được bỏ qua vì
-- giá trị mặc định (trống)
-- và không có vai trò trong thuật toán mã hóa
RsaesParameters ::= SEQUENCE {
hashFunction [0] HashFunction DEFAULT alg-sha1,
keyDerivationFunction [1] RsaesKeyDerivationFunction
DEFAULT alg-mgf1-sha1
}
RsaesKeyDerivationFunction ::=
AlgorithmIdentifier {{ RKDFAlgorithms }}
RKDFAlgorithins ALGORITHM ::= {
KDFAlgorithms    |
{ OID id-mgf1 PARMS HashFunction }
}
-- MGF1 trong PKCS #1 là tương đương với KDF1 ở đây
-- id-mgf1 được sử dụng thay thế cho id-kdf-kdf1 để tương thích
-- với khai triển hiện tại
alg-mgf1-sha1 RsaesKeyDerivationFunction ::= {
      algorithm id-mgf1,
      parameters HashFunction : alg-sha1
}
alg-sha1 HashFunction ::= {
algorithm id-sha1,
parameters NullParms : NULL }
--#####################################################################
-- Hệ mật phi đối xứng HIME(R)
himer ALGORITHM ::= {
OID id-ac-himer PARMS HimerParameters
}
```

```
HimerPublicKey INTEGER

HimerParameters::=SEQUENCE{

d INTEGER(2..MAX),

encodingMethod HimerEncodingMethod

}

HimerEncodingMethod ::= Algorithmldentifier {{ HemAlgorithms}}

HemAlgorithms ALGORITHM ::= {

{ OID id-hem-heml PARMS HemlParameters },

-- các thuật toán mong đợi khác --

}

HemlParameters ::= SEQUENCE {

hashFunction HashFunction,

keyDerivationFunction KeyDerivationFunction

}

--#################################################################

-- Hệ mật phi đối xứng HC lai ghép chung ALGORITHM ::= {

OID id-ac-generic-hybrid PARMS GenericHybridParameters

}

GenericHybridParameters ::= SEQUENCE {

kem KeyEncapsulationMechanism,

dem DataEncapsulationMechanism

}

--#################################################################

-- chú thích viện dẫn:

-- Cấu trúc SubjectPublicKeyInfo được xác định trong X.509, trường thuật
toán

-- sẽ thực hiện theo cú pháp lai ghép chung, và

-- trương subjectPublicKey sẽ là xâu bit tương ứng với một

-- giá trị của EciesKemPublicKey, PsecKemPublicKey,

-- AceKemPublicKey, hoặc RsaKemPublicKey, theo trường kem của
```

```
-- tham số lai ghép chung --
#####################################################################
-- thông tin của KEM
KeyEncapsulationMechanism ::= AlgorithmIdentifier {{ KEMAlgorithms }}
KEMAlgorithms ALGORITHM ::= {
{ OID id-kem-ecies PARMS EciesKemParameters } |
{ OID id-kem-psec PARMS PsecKemParameters } |
{ OID id-kem-ace PARMS AceKemParameters } |
{ OID id-kem-rsa PARMS RsaKemParameters },
-- các thuật toán mong đợi khác --
}
--#####################################################################
-- ECIES-KEM
-- phải là một nhân tố khác 0 của nhóm được đưa ra trong
-- EciesKemParameters EciesKemPublicKey ::= FieldElement
EciesKemParameters ::= SEQUENCE { group Group OPTIONAL,
keyDerivationFunction KeyDerivationFunction,
oldCofactorMode BOOLEAN,
singleHashMode BOOLEAN,
keyLength KeyLength
}
--#####################################################################
-- PSEC-KEM
-- một nhân tố của nhóm được đưa ra (có thể =0)
PsecKemPublicKey ::= FieldElement
PsecKemParameters ::= SEQUENCE {
group Group OPTIONAL,
keyDerivationFunction KeyDerivationFunction,
seedLength INTEGER (1..MAX),
keyLength KeyLength }
```

```
--#####################################################################
-- ACE-KEM
-- toàn bộ các thành phần của khóa công khai của nhóm được đưa ra trong
-- AceKemParameters
AceKemPublicKey ::= SEQUENCE {
gPrime FieldElement,
c FieldElement,
d FieldElement,
h FieldElement
}
AceKemParameters ::= SEQUENCE {
group Group OPTIONAL,
keyDerivationFunction KeyDerivationFunction,
hashFunction HashFunction,
keyLength KeyLength
}
--#####################################################################
-- RSA-KEM
RsaKemPublicKey ::= RSAPublicKey
EsaKemParameters ::= SEQUENCE {
keyDerivationFunction KeyDerivationFunction,
keyLength KeyLength
}
--#####################################################################
--#####################################################################
-- Đặc tả DEM
DataEncapsulationMechanism ::= AlgorithmIdentifier {{DEMAlgorithms}}
DEMAlgorithms ALGORITHM ::= {
{ OID id-dem-deml PARMS DemlParameters } |
{ OID id-dem-dem2 PARMS Dem2Parameters } |
```

```
{ OID id-dem-dem3 PARMS Dem3Parameters },
-- Các thuật toán mong đợi khác --
}
Dem1Parameters ::= SEQUENCE{
symmetricCipher SymmetricCipher,
mac MacAlgorithm
}
Dem2Parameters ::= SEQUENCE{
symmetricCipher SymmetricCipher,
mac MacAlgorithm,
labelLength INTEGER (0..MAX)
}
Dem3Parameters ::= SEQUENCE{
mac MacAlgorithm,
msgLength INTEGER (0..MAX)
}
--########################################################################
-- trường hữu hạn, nhóm, và đặc các mô tả của đương cong elliptic
Group ::= CHOICE {
groupOid OBJECT IDENTIFIER,
groupHashId OCTET STRING,
-- được xác định trong RFC2528 groupParameters GroupParameters
}
GroupParameters ::= CHOICE {
explicitFiniteFieldSubgroup
[0] ExplicitFiniteFieldSubgroupParameters,
ellipticCurveSubgroup
[1] EllipticCurveSubgroupParameters }
ExplicitFiniteFieldSubgroupParameters ::= SEQUENCE {
fieldID FieldID {{FieldType}},
```

```
generator FieldElement,
subgroupOrder INTEGER,
subgroupIndex INTEGER
}
FIELD-ID ::= TYPE-IDENTIFIER
FieldID { FIELD-ID:IOSet } ::= SEQUENCE {
fieldType FIELD-ID.&id({IOSet}) ,
parameters FIELD-ID.&Type({IOSet}{@fieldType}) OPTIONAL
FieldTypes FIELD-ID ::= {
{ Prime-p IDENTIFIED BY prime-field } |
{ Characteristic-two IDENTIFIED BY characteristic-two-field }|
{ Odd-characteristic IDENTIFIED BY id-ft-odd-characteristic },
-- các trường mong đợi khác
}
-- Trường cơ bản Prime-p ::= INTEGER
-- đặc trưng 2 trường CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
-- cơ sở là tối ưu khi nó là gnBasis
-- cơ sở bình thường của dạng T ở đây T được xác định như sau:
-- nếu tồn tại một ONB của dạng 2 với giá trị m cho trước thì
-- T là 2, ngược lại nếu tồn tại một ONB của dạng 1
-- với giá trị m cho trước thì T là 1, ngược lại T sẽ là
-- giá trị nhỏ nhất với một ONB tồn tại dạng T
-- với giá trị m cho trước
-- khi cơ sở là gnBasis thì m không chia hết cho 8
-- chú ý: quy tắc trên từ ANSI X9.62
-- chú ý: với m và T cho trước thì ONB là duy nhất
Characteristic-two SEQUENCE {
m INTEGER,-- extension degree
basis CHARACTERISTIC-TWO.&id({BasisTypes}),
parameters CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
```

```
}
BasisTypes CHARACTERISTIC-TWO ::= {
{ NULL IDENTIFIED BY gnBasis } |
{ Trinomial IDENTIFIED BY tpBasis } |
{ Pentanomial IDENTIFIED BY ppBasis } |
{ CharTwoPolynoinial IDENTIFIED BY charTwoPolynomialBasis },
... -- dạng cơ sở khác được chờ đợi }
Trinomial ::= INTEGER
Pentanomial ::= SEQUENCE {
k1 INTEGER,
k2 INTEGER,
k3 INTEGER
}
-- thức bất khả quy chung characteric two biểu điễn
-- đa thức bất khả quy
-- a(n)*x^n + a(n-1)*x^(n-1) + ... + a(1)*x + a(0)
-- được mã hóa trong xâu bít với a(n) là bit đầu tiên
-- theo sau là các hệ số tiếp theo đối với vị trí từng bit và a(0)
-- là bit cuối cùng của xâu bit (có thể bỏ qua a(n) và a(0)
-- nhưng nếu để lại chúng thì sẽ đơn giản và ít gây lỗi hơn
-- trong bản mã)
-- bậc của đa thức là chiều dài
-- của xâu bit CharTwoPolynomial ::= BIT STRING
-- phần mở rộng của trường odd characteristic
ODD-CHARACTERISTIC ::= TYPE-IDENTIFIER
Odd-characteristic ::= SEQUENCE {
characteristic INTEGER(3..MAX),
degree INTEGER(2..MAX),
basis ODD-CHARACTERISTIC.&id({OddCharBasisTypes}),
parameters ODD-CHARACTERISTIC.&Type ({OddCharBasisTypesH@basis})
```

```
}
OddCharBasisTypes ODD-CHARACTERISTIC ::= {
{ OddCharPolynomial IDENTIFIED BY oddCharPolynomialBasis },
... --các trường cơ sở khác được chờ đợi
}
-- các đa thức bất khả quy monic được mã hóa như sau
-- hệ số đầu được bỏ qua
-- các hệ số còn lại được xác định là các phần tử của trường hữu hạn
-- được mã hóa trong một xâu octet sử dụng FE20SP OddCharPolynomial ::=
FieldElement
EllipticCurveSubgroupParameters ::= SEQUENCE {
version INTEGER { ecpVer1(1) } (ecpVer1),
fieldID FieldID {{ FieldTypes }},
curve Curve,
generator ECPoint,
subgroupOrder INTEGER,
subgroupIndex INTEGER,
}
Curve ::= SEQUENCE {
aCoeff FieldElement,
bCoeff FieldElement,
seed BIT STRING OPTIONAL }
--#####################################################################
-- một số định nghĩa phụ trợ
FieldElement ::= OCTET STRING -- obtained through FE20SP
ECPoint ::= OCTET STRING -- obtained through EC20SP
KeyLength ::= INTEGER (1..MAX)
MacAlgorithm ::= AlgorithmIdentifier {{ MACAlgorithms }}
MACAlgorithms ALGORITHM ::= {
{ OID hMAC-SHA1 PARMS NULL } ,
```

```
-- một số thuật toán được chờ đợi --
}
HashFunction ::= AlgorithmIdentifier {{ HashFunctionAlgorithms }}
HashFunctionAlgorithms ALGORITHM ::= {
HashFunctionAlgs,-- from 10118-3
-- một số thuật toán được chờ đợi
}
KeyDerivationFunction ::= AlgorithmIdentifier {{ KDFAlgorithms }}
KDFAlgorithms ALGORITHM ::= {
{ OID id-kdf-kdf1 PARMS HashFunction } |
{ OID id-kdf-kdf2 PARMS HashFunction } ,
... -- một số thuật toán được chờ đợi --
}
SymmetricCipher ::= AlgorithmIdentifier {{ SymmetricAlgorithms }}
SymmetricAlgorithms ALGORITHM ::= {
{ OID id-sc-sc1 PARMS BlockCipher } |
{ OID id-sc-sc2 PARMS BlockCipher },
-- một số thuật toán được chờ đợi --
}
BlockCipher ::= AlgorithmIdentifier {{ BlockAlgorithms }}
--#####################################################################
-- các OID ngoài
-- RSA-OAEP
pkcs-1 OID ::= { iso(1) member-body(2)
                        us(840) rsadsi(113549) pkcs(1) 1 }
id-RSAES-OAEP OID ::= { pkcs-1 7 }
id-mgf1 OID ::= { pkcs-1 8 }
-- HMAC-SHA1
hMAC-SHA1 OID ::= {
iso(1) identified-organization(3) dod(6) internet(1) security(5)
```

```
    mechanisms(5) 8 12}
--các dạng trường hữu hạn và trường cơ sở X9.62
ansi-X9-62 OID ::= { iso(1) member-body(2) us(840) 10045 }
id-fieldType OID ::= { ansi-X9-62 fieldType(1) }
prime-field OID ::= { id-fieldType 1 }
characteristic-two-field OID ::= { id-fieldType 2 }
-- cơ sở characteristic two
id-characteristic-two-basis OID ::= { characteristic-two-field basisType(3)
}
gnBasis OID ::= { id-characteristic-two-basis 1 }
tpBasis OID ::= { id-characteristic-two-basis 2 }
ppBasis OID ::= { id-characteristic-two-basis 3 }
--########################################################################
-- nhận dạng các thuật toán mật mã --
ALGORITHM CLASS {
&id OBJECT IDENTIFIER UNIQUE,
&Type OPTIONAL
}
WITH SYNTAX {OID &id [PARMS &Type] }
AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
algorithm ALGORITHM.&id( {IOSet} ),
parameters ALGORITHM.&Type( {IOSet}{@algorithm} ) OPTIONAL
}
END -- EncryptionAlgorithms-2 --
```

Phụ lục B

(Tham khảo)

Xem xét tính an toàn

Trong Phụ lục này xem xét các đặc tính an toàn của các lược đồ mật mã khác nhau đã được mô tả trong phần này của tiêu chuẩn ISO/IEC 18033. Đối với mỗi loại lược đồ (ví dụ, mã phi đối xứng, thuật toán MAC,...) đưa ra định nghĩa hình thức tương ứng và với mỗi lược đồ cụ thể sẽ xem xét phạm vi mà định nghĩa đưa ra được thỏa mãn.

Tính an toàn của mỗi lược đồ có thể được chứng minh một cách hình thức, dựa trên các giả thuyết nhất định về tính khó, hoặc trên một giả thuyết khác là các cơ chế ở mức thấp hơn là an toàn. Các chứng minh này là "các phép qui dẫn", trong đó chỉ ra làm thế nào để qui dẫn vấn đề kẻ địch A phá được lược đồ về lược đồ kẻ địch A' giải bài toán được giả thiết là khó, hoặc phá một cơ chế được coi là an toàn. Trong phần lớn các trường hợp, "chất lượng" của "phép qui dẫn" được chỉ ra bằng mô tả cách định lượng mối quan hệ giữa các yêu cầu về tài nguyên (ví dụ thời gian chạy thuật toán) và ưu thế (ví dụ xác suất thành công) của A và yêu cầu về tài nguyên và ưu thế của A'. Phép qui dẫn được gọi là "chặt chẽ", nếu các yêu cầu về tài nguyên của A' lớn hơn không đáng kể so với các yêu cầu về tài nguyên của A, và nếu ưu thế của A' cũng nhỏ hơn không đáng kể so với của A.

Phương pháp tiếp cận tính an toàn ở đây là "cụ thể", như trong [6], so với cách tiếp cận có tính "tiệm cận": phép qui dẫn của tính an toàn được phát biểu bằng thuật ngữ của các lược đồ cụ thể, chứ không phải bằng thuật ngữ của họ các lược đồ được chỉ số hóa bởi "tham số an toàn" và chỉ số này có xu hướng tiến ra vô cùng. Tuy nhiên, một số ước lượng định lượng được biểu thị bằng khái niệm "O lớn", nhưng hoàn toàn là những hằng số bé.

Một số chứng minh tính an toàn thuộc mô hình được gọi là "tiên tri ngẫu nhiên", mô hình này đầu tiên được hình thức hóa trong [7] và từ đó được sử dụng để phân tích nhiều lược đồ mật mã. Trong mô hình tiên tri ngẫu nhiên, hàm băm và hàm dẫn xuất khóa được mô hình như những hàm ngẫu nhiên, mà đối với mọi thuật toán hay mọi kẻ tấn công, chúng chỉ là những "hộp đen". Các loại chứng minh tính an toàn theo mô hình tiên tri ngẫu nhiên như thế tốt nhất là nên xem như những chứng minh mới ở mức tìm tòi (heuristic). Có thể, một lược đồ là an toàn theo mô hình tiên tri ngẫu nhiên, nhưng lại bị phá mà không cần giải được bài toán khó cơ sở, hoặc các giả thuyết an toàn và không cần phải tìm ra bất kì điểm yếu cụ thể nào trong hàm băm hay hàm dẫn xuất khóa nào [12]. Nói cách khác, chứng minh tiên tri ngẫu nhiên đã loại bỏ mất một lớp tấn công rộng.

B.1 Thuật toán MAC

Điều này mô tả đặc tính an toàn cơ bản của thuật toán MAC đã được trình bày trong phần này của tiêu chuẩn ISO/IEC 18033.

Xét thuật toán MA thuộc các thuật toán MAC được mô tả tại Điều 6.3.

Xét kịch bản tấn công sau đây. Kẻ tấn công chọn ngẫu nhiên một xâu bộ tám T^* và khóa bí mật k'. Giả sử kẻ tấn công cũng sở hữu giá trị $MAC^* = MA.eval(k', T^*)$. Kẻ tấn công tạo ra một danh sách các cặp (T, MAC), ở đây T là xâu bộ tám với $T \neq T^*$ (và cũng không nhất thiết có cùng độ dài với T^*), và MAC là xâu bộ tám độ dài $MA.MACLen$. Ưu thế của kẻ tấn công được định nghĩa là xác suất sao cho với mỗi cặp (T, MAC), ta có $MA.eval(k', T) = MAC$.

Với kẻ tấn công A và thuật toán MA, lợi thế được kí hiệu là $Adv_{MA}(A)$. Nếu kẻ tấn công chạy trong khoảng thời gian nhiều nhất là t và tạo ra được một danh sách nhiều nhất là l cặp và T^* và tất cả T về độ dài bị giới hạn bởi l', thì A được gọi là kẻ tấn công $MA[t, l, l']$.

Tính an toàn ở đây có nghĩa là lợi thế này là không đáng kể đối với bất kì kẻ tấn công hiệu quả nào.

Mặc dù mô hình tấn công "đơn thông báo" ở đây được xem là đủ cho việc thiết kế các cơ chế bọc dữ liệu an toàn, nhưng với nhiều ứng dụng khác là không đủ, và cần xem xét cả mô hình tấn công "đa thông báo". Trong mô hình tấn công "đa thông báo", thay vì thu được giá trị của $MA.eval(k',.)$ tại đầu vào đơn lẻ T^*, kẻ tấn công được phép thu được giá trị của $MA.eval(k',.)$ tại nhiều đầu vào (được chọn phù hợp), $T_1^*, \ldots, T_s^*$. Vì trước đó kẻ tấn công đã lập được danh sách các cặp (T, MAC), nhưng với hạn chế là $T \neq T_i^*$ với $1 \leq i \leq s$.

Điều 6.3.1 cho phép sử dụng các thuật toán MAC được mô tả tại ISO/IEC 9797-1 và ISO/IEC 9797-2, tất cả các thuật toán này được thiết kế an toàn trong mô hình tấn công "đa thông báo" và một số trong số đó được chứng minh là an toàn theo mô hình này, dựa trên những giả thiết nhất định về hàm băm mật mã cơ sở.

B.2 Mã khối

Điều này mô tả đặc tính an toàn cơ bản của mã khối được trình bày trong phần này của ISO/IEC 18033.

Xét mã khối BC được xác định trong Điều 6.4.

BC được gọi là hoán vị giả ngẫu nhiên, nếu kẻ tấn công khó phân biệt được hoán vị ngẫu nhiên của xâu bộ tám độ dài $BC.BlockLen$ với hoán vị $b \mapsto BC.Encrypt(k, b)$ với khóa bí mật k được chọn ngẫu nhiên. Trong tấn công này, kẻ tấn công được cho ở dạng tiếp cận tiên tri (oracle access) tới hoán vị hoặc tới hoán vị ngẫu nhiên hoặc tới mã khối và phải đoán được đang tiếp cận trường hợp nào. Hoán vị được coi là giả ngẫu nhiên, nếu với bất kì kẻ tấn công hiệu quả nào, khả năng đoán thành công của nó gần bằng ½, với sai số không đáng kể.

Điều 6.4.1 cho phép sử dụng các mã khối được mô tả trong ISO/IEC 18033-3. Mặc dù có một ít chứng minh hình thức, nhưng kinh nghiệm cho thấy những mã khối này trong thực tế cũng hành xử như các phép hoán vị giả ngẫu nhiên.

B.3 Mã đối xứng

Điều này mô tả đặc tính an toàn cơ bản được yêu cầu đối với mã đối xứng trong phần này của ISO/IEC 18033.

Xét mã đối xứng MC được xác định trong Điều 6.5.

Xét kịch bản tấn công sau. Kẻ tấn công tạo ra hai bản rõ (hai xâu bộ tám) M_0, M_1 có độ dài bằng nhau, khóa bí mật k và chọn một bit ngẫu nhiên b, và M_b được mã hóa theo khóa k. Kẻ tấn công biết bản mã c. Gọi $\hat{b}$ là phỏng đoán bit b của kẻ tấn công. Ưu thế của kẻ tấn công được định nghĩa bằng $|\Pr[\hat{b} = b] - 1/2|$.

Giả sử A là kẻ tấn công và SC là mã đối xứng, ưu thế của A được kí hiệu là $Adv_{SC}(A)$. Nếu kẻ tấn công chạy thời gian nhiều nhất là t và đầu ra của thuật toán mã hóa có độ dài lớn nhất là l bộ tám, thì A được gọi là kẻ tấn công $SC[t, l]$.

Tính an toàn ở đây có nghĩa là ưu thế này là không đáng kể đối với bất kì kẻ tấn công hiệu quả nào.

Mặc dù mô hình tấn công "đơn bản rõ" ở đây được xem là đủ cho việc thiết kế các cơ chế bọc dữ liệu an toàn, nhưng với nhiều ứng dụng khác là không đủ và cần xem xét thêm mô hình tấn công "đa bản rõ", ở đó đó kẻ tấn công được phép có được nhiều phép mã hóa phù hợp để lựa chọn. Dạng tấn công này cũng được gọi là tấn công "chọn bản rõ". Một dạng tấn công khác được gọi là tấn công "chọn bản mã", ở đó kẻ tấn công có được kết quả giải mã các bản mã đã lựa chọn.

B.3.1 An toàn của *SC1*

Điều này xem xét tính an toàn của *SC1* được định nghĩa tại Điều 6.5.2.

Đấy là mã đối xứng được tham số hóa theo thuật ngữ của mã khối *BC*.

Chế độ móc xích mã khối mã *(CBC)* với giá trị khởi đầu (IV) được phân tích trong [4], tại đây đã chỉ ra chế độ *CBC* là an toàn đối với tấn công "đa bản rõ" như đã thảo luận ở trên, với giả thiết *BC* là hoán vị giả ngẫu nhiên (xem Phụ lục B.2). Mã *SC1* sử dụng giá trị khởi đầu cố định. Tuy vậy, dễ dàng điều chỉnh chứng minh tính an toàn trong [4] để chỉ ra rằng *SC1* là an toàn chống lại tấn công "đơn bản rõ" vốn phù hợp với các cấu trúc trong tài liệu này.

Để ý rằng trong bài báo [35] trình bày một số tấn công lên *SC1*. Tuy nhiên, các tấn công trong [35] là tấn công "chọn bản mã" và do đó không liên quan ở đây. Tóm lại lược đồ bộ đệm (padding) đóng vai trò an toàn trong mã hóa *CBC* chỉ khi xem xét các tấn công chọn bản mã.

B.3.2 An toàn của *SC2*

Phần này thảo luận tính an toàn của *SC2* được định nghĩa trong Điều 6.5.3.

Hiện nay chưa có một phép qui dẫn hình thức nào qui tính an toàn của *SC2* về tính an toàn của một số cơ chế khác hoặc tính khó của bài toán khác. Tuy nhiên, nếu muốn mô hình hóa hàm dẫn xuất khóa như một tiên tri ngẫu nhiên thì có thể yên tâm rằng *SC2* là mật mã đối xứng an toàn.

B.4 Mật mã phi đối xứng

Điều này mô tả đặc tính an toàn cơ bản được yêu cầu đối với mật mã phi đối xứng.
Xét mật mã phi đối xứng *AC*, được định nghĩa ở Điều 7.
Xét kịch bản tấn công "chọn bản mã phù hợp" sau đây.

Bước 1: Chạy thuật toán tạo khóa, tạo ra khóa công khai và khóa riêng. Kẻ tấn công, hiển nhiên, biết được khóa công khai nhưng không biết khóa riêng.

Bước 2: Kẻ tấn công đưa ra một loạt câu truy vấn tùy ý để tiên tri giải mã. Mỗi câu chất vấn là một cặp nhãn/bản mã (L, C), cặp này được giải mã bởi tiên tri giải mã sử dụng khóa riêng. Kết quả giả mã được trao cho kẻ tấn công. Ngoài ra, nếu thuật toán **thất bại** thì thông tin này được trao cho kẻ tấn công và việc tấn công được tiếp tục. Kẻ tấn công tự do thiết kế các cặp nhãn/bản mã này theo một cách tùy ý — hiển nhiên không cần sử dụng thuật toán mã hóa để tính toán chúng.

Bước 3: Kẻ tấn công chuẩn bị nhãn L^* và hai bản rõ "đích" M_0, M_1 có cùng độ dài, và nạp chúng vào tiên tri mã hóa. Nếu lược đồ hỗ trợ các tùy chọn mật mã nào đó, thì kẻ tấn công

cũng sẽ chọn chúng. Tiên tri mã hóa chọn ngẫu nhiên $b \in \{0,1\}$ mã hóa M_b sử dụng nhãn L^*, chuyển kết quả là bản mã "đích" C^* cho kẻ tấn công.

Bước 4: Kẻ tấn công tiếp tục chuyển cặp nhãn /bản mã (L, C) tới tiên tri giải mã, chỉ với điều kiện ràng buộc là $(L, C) \neq (L^*, C^*)$.

Bước 5: Kẻ tấn công đưa ra $\hat{b} \in \{0,1\}$, và **dừng lại**.

Ưu thế của A được định nghĩa bằng giá trị $\left|\Pr[\hat{b} = b] - 1/2\right|$. Với kẻ tấn công A và mã phi đối xứng AC, ưu thế của A được kí hiệu là $Adv_{AC}(A)$. Nếu kẻ địch trong thời gian thực hiện t, thực hiện nhiều nhất là q truy vấn giải mã tiên tri, tất cả các bản mã là đầu ra từ mã hóa tiên tri và đầu vào của giải mã tiên tri có độ dài lớn nhất là l, và các nhãn đầu vào của mã hóa và giải mã tiên tri lớn nhất là l', khi đó A được gọi là kẻ tấn công $-AC[t, q, l, l']$.

Tính an toàn có nghĩa là ưu thế đối với tất cả kẻ tấn công hiệu quả là không đáng kể.

Định nghĩa này, ở dạng hơi khác một ít, lần đầu tiên được đề xuất bởi Rackoft và Simon trong [30], trong công trình này định nghĩa trên đã được khái quát cho trường hợp độ dài bản rõ tùy ý và tính đến vai trò của nhãn. Nói chung cộng đồng nghiên cứu mật mã nhất trí rằng đây là thuộc tính an toàn "ổn" cho mật mã phi đối xứng tổng quát. Khái niệm an toàn này kéo theo các đặc tính có lợi khác như "không dễ uốn" (xem [15,16]). Một cách trực quan, tính không dễ uốn có nghĩa là khó biến đổi một cặp nhãn/bản mã (L, C), mã hóa bản rõ M thành cặp khác (L', C'), sao cho việc giải mã C' bằng nhãn L' liên quan đến M theo một cách nào đó. Tham khảo thêm các khái niệm khác an toàn đối với mã đối xứng trong [11,14,5,15, 16].

Trong [27] trình bày một định nghĩa yếu hơn về an toàn, đôi lúc được gọi là an toàn chống lại các tấn công "thời gian ăn trưa". Trong thiết kế này, an toàn cũng được định nghĩa giống như ở trên, chỉ trừ một điểm là kẻ tấn công không được phép thực hiện bất kì truy vấn tiên tri giải mã nào ở bước 4. Mặc dù, điều này có vẻ giống với một định nghĩa tự nhiên về an toàn, song nó lại không tương thích với nhiều ứng dụng và không phải là khái niệm an toàn phù hợp cho các mã phi đối xứng với mục tiêu tổng quát.

Ngoài ra trong [21] cũng trình bày một định nghĩa yếu hơn về an toàn, được gọi là an toàn "ngữ nghĩa" Trong thiết kế này khái niệm an toàn cũng được định nghĩa giống như ở đây, trừ một điểm là nói chung kẻ tấn công không được phép thực hiện bất kì truy vấn tiên tri giải mã nào.

B.4.1 Che giấu độ dài bản rõ

Lưu ý rằng trong cuộc tấn công, đòi hỏi kẻ tấn công phải chuyển cho tiên tri mã hóa hai bản rõ đích có cùng độ dài. Hạn chế này lên kẻ tấn công phản ánh một sự thật là, không thể che giấu kẻ tấn công độ dài của bản rõ được mã hóa - đối với nhiều hệ mật mã thì độ dài bản rõ được thấy rõ từ độ dài bản mã. Nói chung là điều này được áp dụng trong ứng dụng mật mã để đảm bảo là độ dài bản rõ không để lộ thông tin nhạy cảm.

Với mật mã độ dài bản rõ bị giới hạn, khái niệm an toàn cũng giống như trong các trường hợp thông thường, trừ một điểm là không yêu cầu kẻ tấn công trình các bản rõ đích có cùng độ dài cho tiên tri mã hóa. Điều này phản ánh một sự thật là những lược đồ này có thể che giấu độ dài của bản rõ đối với kẻ tấn công.

Với mật mã phi đối xứng độ dài bản rõ cố định, thì vấn đề trên không bị phát sinh.

B.4.2 Tính dễ bị uốn nhẹ: Khái niệm yếu hơn một ít về an toàn

Định nghĩa tính an toàn trên đây có thể coi là mạnh đến mức không cần thiết. Ví dụ, lấy một mật mã phi đối xứng AC thỏa mã định nghĩa trên và thay đổi nó để nhận được mật mã mới AC' như sau: AC' cũng

giống như AC, chỉ trừ một điểm là một bộ tám ngẫu nhiên được gắn vào bản mã sau khi mã hóa và loại bộ tám bổ sung đó trước khi giải. Về phương diện kỹ thuật, AC' không thỏa mãn định nghĩa trên đối với tính an toàn bản mã lựa chọn thích hợp mà dường như điều này là phi trực quan. Thật vậy, mặc dù về mặt kỹ thuật AC' là "dễ bị uốn", nhưng chỉ dễ bị uốn theo một kiểu "nhẹ": có thể tạo ra các bản mã khác nhau của cùng một bản rõ, và các bản mã khác nhau đó đều dễ dàng bị nhận dạng là bản thay thế của một bản rõ.

Điều này mô tả khái niệm hình thức về tính an toàn, phản ánh chính xác khái niệm trực giác của "tính dễ uốn nhẹ".

Đối với mật mã phi đối xứng cụ thể AC, hàm giá trị 0/1, thời gian đa thức $Equiv$, được gọi là thuộc tính tương đương (equivalence predicate) cho AC, nếu với xác suất trội, đầu ra của $AC.KeyGen$ là cặp (PK, pk) sao cho với nhãn bất kì L và hai bản mã bất kì C và C', ta có:

$$Equiv(PK, L, C, C') = 1 \text{ suy ra } AC.Decrypt(pk, L, C) = AC.Decrypt(pk, L, C').$$

Mã đối xứng AC được gọi là dễ uốn nhẹ, nếu tồn tại thuộc tính tương đương $Equiv$ và nếu nó thỏa mãn định nghĩa an toàn ở trên về tính an toàn chống lại tấn công chọn bản mã thích hợp, nhưng với cải biên sau đây trong cuộc chơi tấn công: khi kẻ tấn công trình cặp nhãn/bản mã (L, C) cho tiên tri giải mã ở bước 4, thì thay vì yêu cầu cặp (L, C) ≠ (L*, C*), đòi hỏi $L^* \neq L$ hoặc $Equiv(PK, L, C, C^*) = 0$. Đối với kẻ tấn công A, ưu thế của nó trong cấu trúc này kí hiệu bằng $Adv'_{AC}(A)$.

B.5 Các cơ chế bọc khóa

Điều này mô tả thuộc tính an toàn cơ bản đòi hỏi đối với cơ chế bọc khóa.

Xét cơ chế bọc khóa KEM được định nghĩa trong Điều 8.1.

Xét kịch bản tấn công "chọn bản mã thích hợp" sau:

Bước 1: Chạy thuật toán tạo khóa, tạo ra khóa công khai và khóa riêng. Kẻ tấn công, hiển nhiên, biết được khóa công khai nhưng không biết khóa riêng.

Bước 2: Kẻ tấn công đưa ra một loạt câu truy vấn cho tiên tri giải mã. Mỗi câu truy vấn là bản mã C_0, được giải mã bởi tiên tri giải mã, sử dụng khóa riêng. Kết quả giải mã được trao cho kẻ tấn công, hơn nữa nếu thuật toán **thất bại** thì thông tin này được trao cho kẻ tấn công và tấn công được tiếp tục. Kẻ tấn công tự do kiến thiết các bản mã này theo một cách tùy ý — không cần sử dụng thuật toán mã hóa để tính toán chúng.

Bước 3: Kẻ tấn công gọi tiên tri mã hóa bằng cách cung cấp tùy chọn mật mã nào đó, nếu lược đồ hỗ trợ điều này. Tiên tri mã hóa thực hiện như sau:

a) Chạy thuật toán mã hóa, tạo ra cặp (K*,C_0^*).

b) Tạo ra xâu bộ tám ngẫu nhiên $\tilde{K}$ độ dài $KEM.KeyLen$.

c) Chọn ngẫu nhiên $b \in \{0,1\}$.

d) Nếu b = 0, đưa ra (K*, C_0^*); ngược lại đưa ra là ($\tilde{K}$, C_0^*).

Bước 4: Kẻ tấn công tiếp tục chuyển bản mã C_0 cho tiên tri giải mã với điều kiện ràng buộc $C_0 \neq C_0^*$.

Bước 5: Kẻ tấn công đưa ra $\hat{b} \in \{0,1\}$ và dừng lại.

Ưu thế của A được định nghĩa bằng giá trị $\left|\Pr[\hat{b} = b] - 1/2\right|$. Với kẻ tấn công A, và cơ chế bọc khóa KEM, ưu thế của A được kí hiệu là $Adv_{KEM}(A)$. Nếu kẻ tấn công thực hiện trong thời gian t, thực hiện nhiều nhất là q truy vấn tiên tri giải mã, khi đó A được gọi là kẻ tấn công – $KEM[t, q]$.

Tính an toàn có nghĩa là ưu thế đối với tất cả kẻ tấn công hiệu quả là không đáng kể.

B.5.1 Tính dễ uốn nhẹ

Điều này định nghĩa khái niệm tính dễ uốn nhẹ cho cơ chế bọc khóa tương ứng với khái niệm tính dễ uốn đối với mã phi đối xứng, như trong Phụ lục B.4.2.

Đối với cơ chế bọc khóa cụ thể *KEM*, hàm giá trị 0/1, thời gian đa thức Equiv được gọi là thuộc tính tương đương đối với *KEM*, nếu với xác suất trội, đầu ra của $KEM.KeyGen$ là cặp (PK, pk) sao cho với hai bản mã bất kì C_0 và C_0', ta có:

$Equiv(PK, C_0, C_0') = 1$ suy ra $KEM.Decrypt(pk, C_0) = KEM.Decrypt(pk, C_0')$.

Cơ chế bọc khóa *KEM* được gọi là dễ uốn nhẹ, nếu tồn tại thuộc tính tương đương *Equiv* và nếu nó thỏa mãn định nghĩa an toàn ở trên về tính an toàn chống lại tấn công chọn bản mã thích hợp, nhưng với thay đổi sau đây trong cuộc tấn công: khi kẻ tấn công gửi cặp bản mã C_0 cho tiên tri giải mã ở bước 4, thì yêu cầu $C_0 \neq C_0^*$ được thay bởi $Equiv(PK, C_0, C_0^*) = 0$. Đối với kẻ tấn công *A*, lợi thế của nó trong cấu trúc này kí hiệu bằng $Adv'_{KEM}(A)$.

B.6 Các cơ chế bọc dữ liệu

Điều này mô tả thuộc tính an toàn cơ bản được yêu cầu đối với cơ chế bọc dữ liệu.

Xét cơ chế bọc khóa *DEM* được định nghĩa trong Điều 8.2.

Xét kịch bản tấn công sau. Kẻ tấn công tạo ra hai bản rõ (hai xâu bộ tám) M_0, M_1 có độ dài bằng nhau và nhãn L^*. Khóa bí mật K được tạo một cách ngẫu nhiên. Chọn ngẫu nhiên bit b và M_b được mã hóa bằng khóa bí mật K. Bản mã thu được C_1^* được trao cho kẻ tấn công. Tiếp đó kẻ tấn công đưa ra một loạt các truy vấn cho tiên tri giải mã: mỗi truy vấn là một cặp nhãn/bản mã $(L, C_1) \neq (L^*, C_1^*)$ và tiên tri giải mã tương ứng với bản giải mã của C_1 với nhãn L và khóa K. Kẻ tấn công phỏng đoán giá trị $\hat{b}$ của b. Ưu thế của kẻ tấn công được định nghĩa bằng $\left|\Pr[\hat{b} = b] - 1/2\right|$.

Giả sử A là kẻ tấn công A và *DEM* là cơ chế bọc dữ liệu. Ưu thế của A được kí hiệu là $Adv_{DEM}(A)$. Nếu kẻ tấn công chạy trong thời gian t, thực hiện nhiều nhất là q truy vấn tiên tri, tất cả các bản mã là đầu ra từ mã hóa tiên tri và đầu vào của giải mã tiên tri có độ dài lớn nhất là l bộ tám và các nhãn đầu vào của mã hóa và giải mã tiên tri lớn nhất là l', khi đó A được gọi là kẻ tấn công $-DEM[t, q, l, l']$.

Tính an toàn có nghĩa là ưu thế trên không đáng kể đối bất kì kẻ tấn công hiệu quả nào.

B.6.1 An toàn *DEM1, DEM2* và *DEM3*

Điều khoản này xem xét tính an toàn của các cơ chế bọc dữ liệu *DEM1* (xem Điều 9.1), *DEM2* (xem Điều 9.2), và *DEM3* (xem Điều 9.3).

Xét cơ chế *DEM1*. Lược đồ này được tham số hóa bởi mã đối xứng *SC* và thuật toán xác thực thông báo *MA*. Có thể chỉ ra rằng nếu *SC* thỏa mãn định nghĩa an toàn tại Phụ lục B.3 và *MA* thỏa mãn định nghĩa an toàn tại Phụ lục B.1, khi đó *DEM1* thỏa mãn định nghĩa an toàn tại Phụ lục B.6.

Cụ thể hơn, với bất kì kẻ tấn công $A - DEM1[t, q, l, l']$ ta có:

$$Adv_{DEM1}(A) \leq Adv_{SC}(A_1) + Adv_{MA}(A_2),$$

ở đây,

– A_1 là kẻ tấn công $[t_1, l'']$, với $t_1 \approx t$,

– A_2 là kẻ tấn công $MA[t_2, q, l'']$ với $t_2 \approx t$, và

– $l'' = l - MA.MACLen$.

Tương tự, với bất kì tấn công $A - DEM2[t, q, l, l'']$, ở đây cần thỏa mãn $l' = DEM2.LabelLen$, chúng ta có

$$Adv_{DEM2}(A) \leq Adv_{SC}(A_1) + Adv_{MA}(A_2),$$

ở đây

– A_1 là kẻ tấn công $SC[t_1, l'']$, với $t_1 \approx t$,

– A_2 là kẻ tấn công $MA[t_2, q, l'' + l']$ với $t_2 \approx t$, và
– $l'' = l - MA.MACLen$.
Tương tự, với bất kì kẻ địch $A - DEM3[t, q, l, l']$, ở đây cần thỏa mãn $l = DEM3.MsgLen + MA.MACLen$, chúng ta có

$$Adv_{DEM3}(A) \leq Adv_{MA}(A_2),$$

Ở đây
– A_2 là kẻ tấn công $MA[t_2, q, DEM3.MsgLen + l']$, với $t_2 \approx t$.
Dễ dàng xác lập các giới hạn này từ định nghĩa. Xem, chẳng hạn [14], để chứng minh cho *DEM2* với $LabelLen = 0$. Chứng minh cho những trường hợp khác có thể tiến hành theo những mạch suy luận tương tự như trong [14].

B.7 An toàn của *HC*

Điều khoản này mô tả độ an toàn của mật mã lai ghép HC, được quy định tại điều 8.3. Lược đồ này được tham số hóa liên quan tới cơ chế bọc khóa *KEM* và cơ chế bọc dữ liệu *DEM*.
Có thể chỉ ra rằng, nếu *KEM* thỏa mãn định nghĩa tính an toàn tại phụ lục B.5 và *DEM* thỏa mãn định nghĩa an toàn tại Phụ lục B.6, khi đó *HC* thỏa mãn định nghĩa tại Điều B.4.
Cụ thể hơn, với mọi kẻ tấn công $A - HC[t, q, l, l']$, ta có

$$Adv_{HC}(A) \leq 2.Adv_{DEM}(A_1) + Adv_{KEM}(A_2).$$

Ở đây,
– A_1 là kẻ tấn công $[t_1, q]$, với $t_1 \approx t$,
– A_2 là kẻ tấn công $DEM[t_2, q, l, l']$ với $t_2 \approx t$, và
Bất đẳng thức trên không tính đến khả năng $KEM.KeyGen$ đưa ra cặp khoá "tồi" (ví dụ một trong những cặp như thế là thuật toán giải mã không vận hành như thuật toán nghịch đảo của thuật toán mã hóa) với xác suất khác không. Trong trường hợp này, đơn giản là cần bổ sung xác suất này (được coi là không đáng kể) vào bên phải của bất đẳng thức trên.
Giới hạn này dễ dàng thiết lập từ các định nghĩa. Xem, chẳng hạn trong [14], chứng minh chi tiết trong trường hợp không có nhãn. Chứng minh trong trường hợp có nhãn có thể tiến hành theo các mạch suy luận tương tự như trong [14]. Nếu *KEM* là dễ uốn nhẹ (xem Phụ lục B.5.1), thì dễ dàng chỉ ra rằng *HC* cũng dễ uốn nhẹ (xem Phụ lục B.4.2) với cùng một giới hạn an toàn như trên.

B.8 Các giả thuyết về tính khó liên quan đến các nhóm cụ thể

Điều này xác định một số giả thuyết về tính khó liên quan đến các nhóm cụ thể.

Giả sử $\Gamma = (H, G, g, \mu, \nu, \mathcal{E}, \mathcal{D}, \mathcal{E}', \mathcal{D}')$ là nhóm cụ thể được định nghĩa ở Điều 10.1.

B.8.1 Bài toán tính toán Diffie-Hellman.

Bài toán tính toán Diffie-Hellman (CDH) đối với Γ phát biểu như sau. Cho đầu vào *(xg, yg)* với $x, y \in [0, ..., \mu)$. Hãy tính $xy.g$, giả thiết đầu vào ngẫu nhiên, tức x và y được chọn ngẫu nhiên từ tập hợp$[0, ..., \mu)$.

Giả thiết *CDH* là giả thiết rằng bài toán tính toán Diffie-Hellman là bài toán khó.

Lưu ý rằng rất khó, thậm chí, nhận dạng lời giải đúng cho bài toán *CDH* (đây được gọi là bài toán quyết định Diffie-Hellman–xem phần tiếp theo dưới đây). Trong khi phân tích các hệ mật, các dạng thuật

toán để giải bài toán CDH mà phần lớn là phát sinh một cách tự nhiên là những thuật toán cho ra một danh sách các lời giải có thể đối với trường hợp đã cho của bài toán CDH. Với thuật toán A bất kì giải bài toán CDH, đầu ra của nó là một danh sách các phần tử nhóm, ta kí hiệu $AdvCDH_\Gamma(A)$ là xác suất để danh sách đó chứa lời giải đúng đối với đầu vào bài toán. Nếu A chạy trong thời gian t và tạo ra được danh sách nhiều nhất là l phần tử nhóm, thì A được gọi là *kẻ tấn công* $CDH_\Gamma(t, l)$.

Trong [32] đã chỉ ra, làm thế nào để, kẻ tấn công $CDH_\Gamma(t, l)$ với $\epsilon = AdvCDH_\Gamma(A)$ và giá trị δ cho trước, biến đổi thành kẻ tấn công $A' - CDH_\Gamma[t', 1]$, sao cho $AdvCDH_\Gamma(A')$ = 1-δ và sao cho t' tương đương $O(t \cdot \epsilon^{-1} \log(1/\delta))$ cộng với thời gian thực hiện số các phép toán nhóm là:

$$O(\epsilon^{-1} l \log(1/\delta) \log \mu + (\log \mu)^2)$$

B.8.2 Bài toán quyết định Diffie-Hellman

Bài toán quyết định Diffie–Hellman (DDH) đối với Γ được phát biểu như sau:

Xác định hai phân bố:

Phân bố **R** gồm bộ ba (xg, yg, zg), ở đây x, y, z được chọn ngẫu nhiên từ tập hợp $[0, ..., \mu)$. Giả sử X_R là biến ngẫu nhiên được lấy mẫu từ phân bố này.

Phân bố **D** bao gồm bộ ba (xg, yg, zg),ở đây x, y được chọn ngẫu nhiên từ tập hợp $[0, ..., \mu)$, còn $z = xy \bmod \mu$. Kí hiệu X_D là biến ngẫu nhiên với mẫu được lấy từ phân bố **D**.

Bài toán quyết định Diffie-Hellman là phân biệt hai phân bố nói trên (**R** và **D**)

Với thuật toán A mà đầu ra hoặc là 0 hoặc là 1, "Ưu thế DDH" của nó được định nghĩa bằng

$$AdvCDH_\Gamma(A) = |\Pr[A(X_R) = 1] - \Pr[A(X_D) = 1]|.$$

Nếu A chạy trong thời gian t thì A được gọi là kẻ tấn công DDH_Γ[t].

Giả thiết DDH là: Ưu thế này là không đáng kể đối với tất cả các thuật toán hiệu quả.

Tham khảo thêm [10, 25, 26] về DDH và các vấn đề liên quan.

B.8.3 Bài toán Gap-CDH

Bài toán Gap-CDH là bài toán giải bài toán CDH với sự hỗ trợ của tiên tri cho bài toán DDH. Trong trường hợp này, vì thuật toán cho bài toán này sử dụng tiên tri DDH, có thể giả thiết là đầu ra của thuật toán là phần tử nhóm đơn lẻ, chứ không phải một danh sách các phần tử nhóm.

Giả thiết rằng bài toán Gap-CDH là bài toán khó.

Với bất kì thuật toán "tiên tri" A, $AdvGapCDH_\Gamma(A)$ được định nghĩa là xác suất để cho ra lời giải đúng cho trường hợp ngẫu nhiên của bài toán CDH, sử dụng tiên tri đối với Γ. Nếu A chạy trong thời gian nhiều nhất là t, và thực hiện nhiều nhất q truy vấn đối với DDH-tiên tri, thì A được gọi là *kẻ tấn công* $GapCDH_\Gamma[t, q]$.

Tham khảo [29] để biết thêm chi tiết về giả thiết này.

B.9 Tính an toàn của *ECIES-KEM*

Điều này xem xét tính an toàn của cơ chế bọc khóa $ECIES - KEM$ được định nghĩa ở Điều 10.2.

Lược đồ này được tham số hóa theo thuật ngữ của nhóm cụ thể Γ (xem Điều 10.1) và hàm dẫn xuất khóa *KDF* (xem Điều 6.2).

Có thể chứng minh lược đồ này là an toàn theo mô hình tiên tri ngẫu nhiên, nơi *KDF* được mô hình hóa như tiên tri ngẫu nhiên, với giả thiết là bài toán Gap-CDH là bài toán khó.

Cụ thể hơn, giả sử rằng các tham số hệ thống của $ECIES-KEM$ được chọn sao cho $SingleHashMode = 0$ và

$$CheckMode + CofactorMode + OldCofactorMode > 0.$$

Khi đó nếu A là kẻ tấn công $ECIES-KEM[t,q]$ và thực hiện nhiều nhất q' truy vấn tiên tri ngẫu nhiên, khi đó ta có

$$Adv_{ECIES-KEM}(A) = O\left(AdvGapCDH_{\Gamma}(A')\right),$$

ở đấy,

– A' là kẻ tấn công $GapCDH_{\Gamma}(t', O(q'), t' \approx t$.

Giới hạn này về cơ bản đã được chứng minh trong [14], ít nhất cũng cho trường hợp khi $CheckMode = 1$ và các phần tử nhóm có mã duy nhất. Các trường hợp khác có thể chứng minh bằng lập luận tương tự.

Giả sử các tham số hệ thống của $ECIES-KEM$ được chọn sao cho $SingleHashMode = 1$ và

$$CheckMode + CofactorMode + OldCofactorMode > 0.$$

Trong trường hợp này $ECIES-KEM$ sẽ không còn an toàn chống lại tấn công chọn bản mã, nhưng lại dễ uốn nhẹ (tham khảo Phụ lục B.5.1). Nếu khi đó nếu A là kẻ tấn công $ECIES-KEM[t,q]$ và thực hiện nhiều nhất q' truy vấn tiên tri ngẫu nhiên, khi đó ta có

$$Adv'_{ECIES-KEM}(A) = O\left(AdvGapCDH_{\Gamma}(A')\right),$$

ở đây,

– A' là kẻ tấn công $GapCDH_{\Gamma}(t', O(q.q'), t' \approx t$.

Ngoài ra, chỉ thỏa mãn định nghĩa yếu về an toàn, thì phép qui dẫn này sẽ không chặt chẽ như trong trường hợp ở đó $SingleHashMode = 0$. Đồng thời chất lượng của phép qui dẫn bị suy giảm, thậm chí còn giảm hơn với $SingleHashMode = 1$, khi xem xét mô hình đa bản rõ được định nghĩa một cách hình thức trong [3], ngược lại phép qui dẫn không suy giảm đáng kể khi $SingleHashMode = 0$.

Nếu

$$CheckMode + CofactorMode + OldCofactorMode = 0,$$

khi đó, trong cả hai ước lượng trên, thuật ngữ $AdvGapCDH_{\Gamma}(A')$, cần được thay thế bởi $v.AdvGapCDH_{\Gamma}(A')$.

Bởi vậy, sự lựa chọn tham số hệ thống này chỉ nên sử dụng khi v rất bé.

Thay vì phân tích $ECIES-KEM$ theo thuật ngữ của giả thiết Gap-CDH trong mô hình tiên tri ngẫu nhiên, có thể phân tích nó mà không cần sử dụng các tiên tri ngẫu nhiên, nhưng với giả thiết rất cụ thể và không chuẩn tắc. Tham khảo tại [1, 2] để biết thêm chi tiết.

B.10 Tính an toàn của *PSEC-KEM*

Điều này xem xét tính an toàn của cơ chế bọc khóa $PSEC-KEM$ được định nghĩa tại Điều 10.3.

Lược đồ này được tham số hóa bằng thuật ngữ của nhóm cụ thể Γ (xem Điều 10.1) và hàm dẫn xuất khóa KDF (Điều 6.2).

Có thể chứng minh sự an toàn của lược đồ này theo mô hình tiên tri ngẫu nhiên, coi KDF như tiên tri ngẫu nhiên và giả thiết bài toán CDH là bài toán khó.

Cụ thể hơn, với giá trị cho trước của tham số hệ thống $SeedLen$ và với A-kẻ tấn công $PSEC-KEM[t,q]$ thực hiện nhiều nhất q' truy vấn ngẫu nhiên, ta có

$$Adv_{PSEC-KEM}(A) = O(AdvCDH_{\Gamma}(A') + (q+q')(\mu^{-1} + 2^{-SeedLen})),$$

ở đây A' là kẻ tấn công $AdvCDH_{\Gamma}[t', O(q+q')]$, $t' \approx t$.

Giới hạn này được chứng minh trong [41].

Đồng thời tính an toàn cũng không bị suy giảm đáng kể trong mô hình đa bản rõ được nghĩa trong [10].

B.11 Tính an toàn của *ACE-KEM*

Điều này xem xét tính an toàn của cơ chế bọc khóa $ACE-KEM$, được định nghĩa trong 10.4.

Lược đồ này được tham số hóa bằng thuật ngữ của nhóm cụ thể Γ (xem Điều 10.1), hàm dẫn xuất khóa KDF (xem Điều 6.2) và hàm băm $hash$ (Điều 6.1).

Lược đồ này có thể được chứng minh là an toàn, với giả thiết bài toán DDH là bài toán khó-cần nhấn mạnh là chứng minh về tính an toàn này không có trong mô hình tiên tri ngẫu nhiên. Thay vì điều này, một số giả thiết cụ thể và tương đối chuẩn tắc đã được đưa ra về hàm dẫn xuất khóa và hàm băm.

Cụ thể hơn, với bất kì kẻ tấn công A, $ACE-KEM\ [t,q]$, ta có

$$Adv_{ACE-KEM}(A) = O\ (AdvDDH_{\Gamma}(A_1) + Adv_{Hash}(A_2) + Adv_{KDF}(A_3) + q.\mu^{-1}),$$

ở đây:

— A_1, A_2, A_3 là những kẻ tấn công, thực hiện với lượng thời gian như A.

— $Adv_{Hash}(A_2)$ là kí hiệu xác suất mà kẻ tấn công A_2, với mã cho trước $EU1^*$ và $EU2^*$ của hai phần tử độc lập, ngẫu nhiên thuộc $\mathcal{G}$, có thể tìm thấy mã $EU1$, và $EU2$ của các phần tử thuộc $\mathcal{G}$, sao cho ($EU1$, $EU2$)≠ ($EU1^*$, $EU2^*$), nhưng

$$Hash.eval\ (EU1\ ||EU2) = Hash.eval\ (EU1^*\ ||\ EU2^*).$$

Nếu nhóm hỗ trợ đa mã hóa, thì kẻ tấn công có thể chọn định dạng theo ý muốn, khi EU1* và EU2* được tạo ra. Hơn nữa kẻ tấn công có thể chọn sử dụng các định dạng như nhau hoặc khác nhau trong hai lựa chọn của nó là $EU1$ và $EU2$. Tuy nhiên $EU1^*$ và $EU2^*$ phải là các mã hóa bền vững, và $EU1$ và $EU2$ cũng sở hữu các tính chất đó.

Nếu $CofactorMode = 1$, thì kẻ tấn công có thể chọn $EU1$ làm mã của phần tử thuộc $\mathcal{H}$ mà không nhất thiết thuộc $\mathcal{G}$.

Lưu ý rằng bài toán này là bài toán xung đột tiền ảnh thứ hai, bài toán này nói chung là bài toán khó giải hơn nhiều so với bài toán tìm một cặp bất kì đầu vào xung đột.

– $Adv_{KDF}(A_3)$ kí hiệu ưu thế của A_3 trong việc phân biệt giữa hai phân bố sau. Giả sử u_1 và $\tilde{h}$ là các phần tử độc lập, ngẫu nhiên trong $\mathcal{G}$ và giả sử $EU1$ là mã của u_1. Giả sử R là xâu các bộ tám ngẫu nhiên độ dài $KeyLen$. Phân bố thứ nhất là $(R, EU1)$, và phân bố thứ hai là $(KDF(EU1 \parallel \mathcal{E}'(\tilde{h}), KeyLen), EU1)$.

Độc giả có thể tham khảo [14] để xem chứng minh chi tiết cho trường hợp $CofactorMode = 0$ và các phần tử nhóm có mã duy nhất. Dễ dàng chuyển chứng minh sang các trường hợp khác, sử dụng yếu tố là thuật toán giải mã kiểm tra các mã hóa bền vững.

Trong [14] đồng thời cũng chỉ ra rằng, độ an toàn của $ACE-KEM$ không kém hơn $ECIES-KEM$, tức là với bất kì kẻ tấn công - $ACE-KEM$ $[t,q]$ A tồn tại một kẻ tấn công - $ECIES-KEM$ $[t',q]$ A' với $t' \approx t$ và $Adv_{ECIES-KEM}(A') \approx Adv_{ACE-KEM}(A)$. Chứng minh trong [14] chỉ cho trường hợp $CofactorMode = 0$ và các phần tử của nhóm có mã hóa duy nhất. Chứng minh này dễ dàng thích hợp để xử lý các trường hợp khác, một lần nữa làm cho việc sử dụng thực tế là kiểm tra thuật toán giải mã cho mã hóa thích hợp.

Trong [14] đồng thời cũng chỉ ra rằng, nếu coi KDF như tiên tri ngẫu nhiên, thì tính an toàn của $ACE-KEM$ có thể được chứng minh dựa trên giả thiết CDH. Tuy nhiên, phép qui dẫn này về tính an toàn cũng không hoàn toàn chặt chẽ. Chứng minh trong [14] chỉ để cho trường hợp $CofactorMode = 0$ và các phần tử nhóm có mã duy nhất. Chứng minh này dễ dàng chuyển sang cho các trường hợp khác.

Như đã chỉ ra trong Điều 10.4.4, nên lưu ý đến việc thực thi $ACE-KEM.Decrypt$. Đặc biệt, việc thực thi $ACE-KEM.Decrypt$ không nên để lộ nguyên nhân của sai sót trong bước g. Nếu kẻ tấn công có thể thu được thông tin từ tiên tri giải mã, việc chứng minh tính an toàn trong giả thiết DDH sẽ không còn giá trị nữa. Tuy nhiên, thậm chí nếu thông tin này tiếp cận được, thì hiện chưa biết tấn công nào vào lược đồ này, và hơn nữa lược đồ vẫn không kém an toàn hơn so với *ECIES-KEM*.

B.12 Bài toán ngược RSA

Điều này xem xét bài toán ngược RSA.

Giả sử $RSAKeyGen$ là thuật toán tạo khóa RSA (xem Điều 11.1).

Bài toán ngược RSA phát biểu như sau: Cho đầu vào n và e của thuật toán $RSAKeyGen()$, cho $x \in [0 \ldots n)$ hãy tính y sao cho $y^e \equiv x \pmod n$. Với thuật toán A bất kì và thuật toán tạo khóa cho trước $RSAKeyGen$, $Adv_{RSAKeyGen}(A)$ kí hiệu xác suất của A giải thành công bài toán ngược nói trên. Nếu A chạy trong lượng thời gian nhiều nhất t, thì được gọi là *kẻ tấn công* $RSAKeyGen[t]$.

Giả thiết RSA đối với $RSAKeyGen$ là giả thiết: $Adv_{RSAKeyGen}(A)$ không đáng kể đối với bất kì thuật toán hiệu quả A nào.

B.13 Tính an toàn của *RSAES*

Điều này xem xét tính an toàn của mật mã có độ dài bản rõ hạn chế *RSAES*, được định nghĩa trong Điều 11.4.

Trong [8] đã phân tích một cài đặt tổng quát hơn, trong đó (phương án thứ yếu của) cơ chế mã hóa *REM1* (được định nghĩa tại Điều 11.3.2) được áp dụng cho phép "hoán vị cửa sập một chiều" tổng quát,

chứ không chỉ cho hàm RSA một chiều cụ thể. Việc phân tích được tiến hành theo mô hình tiên tri ngẫu nhiên, ở đấy hàm dẫn xuất khóa và hàm băm được mô hình như những tiên tri.

Trong [8] cũng đã chứng minh rằng lược đồ thu được thỏa mãn một tính chất kỹ thuật, được gọi là "nhận thức bản rõ" (plaintext awareness), với giả thiết rằng phép hoán vị cơ sở thực sự là hàm một phía. Ngoài ra, như đã chỉ ra trong [33], việc nhận biết bản rõ không suy ra được tính an toàn chống lại tấn công chọn bản mã – nó chỉ suy ra khái niệm yếu hơn về an toàn, đó là an toàn chống lại tấn công "thời gian ăn bữa trưa" (xem Phụ lục B.4). Hơn nữa, trong [33] cũng đã chứng minh rằng nói chung *REM1* không tạo ra mật mã an toàn chống lại tấn công chọn bản mã, nếu phép hoán vị cơ sở là tùy ý. Kết quả phủ định này không suy ra rằng *RSAES* là không an toàn chống lại tấn công chọn bản mã - chỉ suy ra rằng phân tích trong [8] không đạt được điều này.

Trong [33], đã chỉ ra rằng *RSAES* là an toàn, nếu số mũ mã hoá e là rất bé (ví dụ $e = 3$).Kết quả này đã được tổng quát trong [20] cho số mũ mã hóa tổng quát. Tuy nhiên có thể chỉ ra rằng phép qui dẫn của tính an toàn trong [20] là rất không chặt chẽ — thật vậy, nói chung chưa có thể nói gì về tính an toàn của *RSAES* đối với modulo RSA kích thước tới hàng ngàn bit. Phép qui dẫn an toàn trong [33] cho số mũ mã hóa nhỏ là tốt hơn đáng kể, nhưng vẫn không hoàn toàn chặt chẽ như đòi hỏi.

Như đã chỉ ra trong Điều 11.3.2.3, cần quan tâm đến vấn đề thực thi *RSAES*. Đặc biệt là việc thực thi $REM1.Decode$ không nên để lộ nguyên nhân sai sót tại bước thứ k; Nếu kẻ tấn công có thể thu được thông tin này từ tiên tri giải mã, thì lược đồ có thể dễ dàng bị phá, như mô tả trong [24].

B.14 Tính an toàn của *RSA-KEM*

Điều này xem xét tính an toàn của cơ chế bọc khóa *RSA-KEM*, được định nghĩa trong Điều 11.5.

Có thể chứng minh tính an toàn của lược đồ này theo mô hình tiên tri ngẫu nhiên, nơi các tham số hệ thống *KDF* được mô hình hóa dưới dạng tiên tri ngẫu nhiên và giả thiết bài toán ngược RSA là bài toán khó.

Cụ thể hơn, với bất kì thuật toán tạo khóa $RSAKeyGen$, sao cho đầu ra (n, e, d) luôn luôn thỏa mãn điều kiện $n \geq nBound$, và với mỗi kẻ tấn công $A, RSA - KEM[t, q]$ ta có

$$Adv_{RSA-KEM}(A) \leq Adv_{RSAKeyGen}(A') + q/nBound.$$

ở đây,

– A' là kẻ tấn công $RSAKeyGen[t']$ với $t' \approx t$.

Bất đẳng thức này không tính đến khả năng của biến cố $RSAKeyGen$ tạo ra khóa RSA "tồi" với xác suất khác không. Trong trường hợp này, đơn giản là cần bổ sung xác suất này (giả thiết là không đáng kể) vào vế phải của bất đẳng thức trên.

Xem chứng minh tại [34].

Phép qui dẫn này hoàn toàn chặt chẽ, khác với phép qui dẫn cho *RSAES* được thảo luận ở trên trong Phụ lục B.13. Ngoài ra, trong mô hình đa bản rõ được định nghĩa trong [3], tính an toàn của *RSA-KEM* nói chung không suy giảm do tính tự qui dẫn ngẫu nhiên của bài toán ngược RSA. Ngược lại, tính an toàn của *RSAES* suy giảm một cách tuyến tính theo số lượng bản rõ, vì không may là tính chất tự qui dẫn ngẫu nhiên không thể khai thác được trong bối cảnh này.

Đồng thời, không giống như *RSAES*, không có cảm giác là *RSA-KEM* dễ bị tổn thương đối với các tấn công "thực thi", như tấn công trong [24].

B.15 Tính an toàn của *HIME (R)*

Có thể chỉ ra rằng theo mô hình tiên tri ngẫu nhiên, nơi hàm *Hash* và *KDF* trong *HEM1* được mô hình hóa dưới dạng tiên tri ngẫu nhiên, rằng *HIME(R)* là an toàn chống lại tấn công chọn bản rõ phù hợp, với giả thiết là bằng tính toán để không thể phân tích số nguyên ở dạng mà ra bởi thuật toán *HIMEKeyGen*. Để xem chi tiết, tham khảo [29, 35] - lưu ý là [35] đã chỉnh sửa một số sai sót trong [29].

PHỤ LỤC C

(Tham khảo)

Các véc tơ kiểm tra

Phụ lục này đưa ra các véc tơ kiểm tra cho lược đồ mã hóa được đặc tả trong phần này của tiêu chuẩn ISO/IEC 18033.

Đối với các cơ chế bọc khóa dựa trên ELGamal, nhóm "Modp" là một nhóm con của Z_p^* với số nguyên tố p cho trước; nhóm "ECModp" là các đường cong elliptic trên Z_p, đôi khi được gọi là "P192" trong các tiêu chuẩn khác; nhóm "ECGF2" là các đường cong elliptic trên trường hữu hạn gồm có 2^{163} phần tử, đôi khi được gọi là "B163" trong các tiêu chuẩn khác (các phần tử trong trường được mô tả với quan hệ của đa thức cơ sở cho đa thức bất khả quy p đã cho).

C.1 Véc tơ kiểm tra cho *DEM1*

C.1.1 Véc tơ kiểm tra

```
DEM1
SC=SC1(BC=AES(keylen=32))
MAC=HMAC(Hash=Sha1(), keylen=20, outlen=20)
----------
Dấu hiệu mã hóa DEM1
----------
Thông báo trong ASCII = "the rain in spain falis mainly on the plain"
Thông báo trong xâu = 0x746865207261696e20696e20737061696e2066616c6c
                      73206d61696e6c79206f6e2074686520706c61696e
Nhãn trong ASCII = "test"
Nhãn trong xâu octet = 0x74657374
k = 0x6434363064303334306635613764353333643739636535636535396235633737
k' = 0x3863323837346633333330653033653032303536
c = 0x0745c5f99ad56fe3ae4ebbeddc5385493cf67a8fa3e3fcdda5d8c82308a8e2b04c
    a4ac32241b1036f20fbe1f3aed19a3
T = 0x0745c5f99ad56fe3ae4ebbeddc5385493cf67a8fa3e3fcdda5d8c82308a8e2b04c
    a4ac32241b1036f20fbe1f3aed19a3746573740000000000000020
MAC = 0x016072f3d5cd979bb49a7c350b233b724f64bba9
C1 = 0x0745c5f99ad56fe3ae4ebbeddc5385493cf67a8fa3e3fcdda5d8c82308a8e2b04
     ca4ac32241b1036f20fbe1f3aed19a3016072f3d5cd979bb49a7c350b233b724f64
     bba9
```

C.1.2 Véc tơ kiểm tra

```
DEM1
```

SC=SC2(Kdf=Kdf1(Hash=Sha1()), keylen=32)

MAC=HMAC(Hash=Sha1(), keylen=20, outlen=20)

Dấu hiệu mã hóa DEM1

Nhãn trong ASCI = "the rain in spain fails mainly on the plain"

Nhãn trong xâu octet = 0x746865207261696e20696e20737061696e2066616c6c
73206d61696e6c79206f6e2074686520706c61696e

Nhãn trong ASCII = "test"

Nhãn trong xâu octet = 0x74657374

k = 0x6434363064303334306635613764353333643739636535636535396235633737

k1 = 0x3863323837346633333330653033653032303536

c = 0xae747466b1f160cf196d2ebe16ac9a70b6ff57c614436cf3de67ea38324f275791
164cfcaea866b0024db7

T = 0xae747466b1f160cf196d2ebe16ac9a70b6ff57c614436cf3de67ea38324f275791
164cfcaea866b0024db774657374000000000000020

MAC = 0xa3462a9d5997aeaac247b33b6c13d748511e0f20

C1 = 0xae747466b1f160cf196d2ebe16ac9a70b6ff57c614436cf3de67ea38324f27579
1164cfcaea866b0024db7a3462a9d5997aeaac247b33b6c13d748511e0f20

C.2 Véc tơ kiểm tra cho *ECIES-KEM*

C.2.1 Véc tơ kiểm tra

ECIES-KEM

Kdf=Kdf1(Hash=Sha1())

Keylen=128

CofactorMode=0

OIDCofactorMode=0

CheckMode=1

SingleHashMode=0

Group=Modp-Group:

p = 0x8a1b8d83ef967f4e8dc0a423a178b33f31a3aeb743fb332dc020970b44ba95bd29
38eb60365ee9c1b1bda579d8276553758e84eb2a8f89c21f8c08ae12f2aacf

g = 0x5e769d3a6fc9b82acf30800c8afe9631c2b9a1bdee398fd0a920704560513898d9
4e40f3f6fc6a773249d63fc74bba14ceadc203b49f2344a6a22a0a8904c60b

mu = 0xdf0235fe94e74d2d70dbbc887389e5af9ec9ccd7

nu = 0x9e89f7f68e9a2e44b68affab0e53d03763d829685af48fa6405ce08865be6c7ee
7221781300459df024b33e2

Khóa công khai

h = 0x61ddb01fad54cffe21a3a68c1cf388c23493699e74519931e42b8576a9652e47dc
c65f7cd297039268d4a7d6b0337466415647a6f6204b6604d3659127f5c69f

---

Khóa riêng

x = 0x4a401de389f502aa4e1fb066b940a6784626a429

Dấu hiệu mã hóa ECIES-KEM

---

r = 0x83bd99b480f6e3ab8b9dc4f410470949f9c9361d

r' = 0x83bd99b480f6e3ab8b9dc4f410470949f9c9361d

g^ = 0x5110f7e54f656e70c71ea2067c901570088a1eb1b230000abba1b2df4b774bed5
43c0325b7083f2b477d5c02ddcafdfec0725672da2cbed39baf75f02dc078d0

h^ = 0x4e9752632f973db43ed3d06ffd5bd9e741af0f855cbc556b73ab530affd7850ca
4c93d4b91d73b47db8718c05e296151e036cf9ba980cef6563af244438cac1b

PEH = 0x4e9752632f973db43ed3d06ffd5bd9e741af0f855cbc556b73ab530affd7850c
a4c93d4b91d73b47db8718c05e296151e036cf9ba980cef6563af244438cac1b

z = 0x5110f7e54f656e70c71ea2067c901570088a1eb1b230000abba1b2df4b774bed54
3c0325b7083f2b477d5c02ddcafdfec0725672da2cbed39baf75f02dc078d0

C0 = 0x5110f7e54f656e70c71ea2067c901570088a1eb1b230000abba1b2df4b774bed5
43c0325b7083f2b477d5c02ddcafdfec0725672da2cbed39baf75f02dc078d0

K = 0x23e41472d780bfbb2daafd85a8fcdf8641fdca4d9f539a4ad175c473ca0f498728
931bc311baa2c957ab528935aa22954075a2899ab1ce8ff5ba90a049aeba8cbb9019
beefc5c24c815ac8a1106e163936597b5d06ba4b52377ca48d82621b2768373a2103
88998b964c11b0a2780c12c49cdea2cb454543fb3b725b026443d9

C.2.2 Véc tơ kiểm tra

ECIES-KEM

Kdf=Kdf1(Hash=Sha1())

Keylen=128

CofactorMode=0

OldCofactorMode=0

CheckMode=0

SingleHashMode=0

Group=ECModp-Group:

p = 0xffffffffffffffffffffffffffffffffefffffffffffffff

a = 0xfffffffffffffffffffffffffffffffefffffffffffffffc

b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

mu = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831

nu = 0x01

g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012

g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811

Khóa công khai

h(x) = 0x1cbc74a41b4e84a1509f935e2328a0bb06104d8dbb8d2130

h(y) = 0x7b2ab1f10d76fde1ea046a4ad5fb903734190151bb30cec2

Khóa riêng x = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3f3

Dấu hiệu mã hóa ECIES-KEM

Encoding format = uncompressed_fmt

r = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8f21

r' = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8f21

g~(x) = 0xccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df

g~(y) = 0x047b2e07dd2ffb89359945f3d22ca8757874be2536e0f924

h~(x) = 0xcdec12c4cf1cb733a2a691ad945e124535e5fc10c70203b5

h~(y) = 0x0cae66e42ae0dd8857ab670c6397c93c1769f9a5f5b9d36d

PEH = 0xcdec12c4cf1cb733a2a691ad945e124535e5fc10c70203b5

z = 0x04ccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df047b2e07dd2ffb89
359945f3d22ca8757874be2536e0f924

C0 = 0x04ccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df047b2e07dd2ffb8
9359945f3d22ca8757874be2536e0f924

K = 0x9a709adeb6c7590ccfc7d594670dd2d74fcdda3f8622f2dbcf0f0c02966d5d9002
db578c989bf4a5cc896d2a11d74e0c51efe1f8ee784897ab9b865a7232b5661b7cac
87cf4150bdf23b015d7b525b797cf6d533e9f6ad49a4c6de5e7089724c9cadf0adf1
3ee51b41be6713653fc1cb2c95a1d1b771cc7429189861d7a829f3

C.2.3 Véc tơ kiểm tra *ECIES-KEM*

ECIES-KEM

Kdf = Kdf1(Hash=Sha1())

Keylen=128

CofactorMode=0

```
OldCofactorMode=0
CheckMode=0
SingleHashMode=0
Group=ECModp-Group:
p = 0fffffffffffffffffffffffffffffffeffffffffffffffff
a = 0xfffffffffffffffffffffffffffffffefffffffffffffffc
b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
mu = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831
nu = 0x01
g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811
-----
Khóa công khai
h(x) = 0x1cbc74a41b4e84a1509f935e2328a0bb06104d8dbb8d2130
h(y) = 0x7b2ab1f10d76fde1ea046a4ad5fb903734190151bb30cec2
-----
Khóa riêng
x = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3f3

--------------------------------------------------
 Dấu hiệu mã hóa ECIES-KEM
--------------------------------------------------
Encoding format = compressed_fmt
r = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8f21
r'= 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8f21
g~(x) = 0xccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df
g~(y) = 0x047b2e07dd2ffb89359945f3d22ca8757874be2536e0f924
h~(x) = 0xcdec12c4cf1cb733a2a691ad945e124535e5fc10c70203b5
h~(y) = 0x0cae66e42ae0dd8857ab670c6397c93c1769f9a5f5b9d36d
PEH = 0xcdec12c4cf1cb733a2a691ad945e124535e5fc10c70203b5
z = 0x02ccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df
C0 = 0x02ccc9ea07b8b71d25646b22b0e251362a3fa9e993042315df
K = 0x8fbe0903fac2fa05df02278fe162708fb432f3cbf9bb14138d22be1d279f74bfb9
    4f0843a153b708fcc8d9446c76f00e4ccabef85228195f732f4aedc5e48efcf2968c
    3a46f2df6f2afcbdf5ef79c958f233c6d208f3a7496e08f505d1c792b314b45ff647
    237b0aa186d0cdbab47a00fb4065d62cfc18f8a8d12c78ecbee3fd
```

C.2.4 Véc tơ kiểm tra *ECIES-KEM*

ECIES-KEM

Kdf=Kdf1(Hash=Sha1())

Keylen=128

CofactorMode=0

OldCofactorMode=0

CheckMode=0

SingleHashMode=0

Group=ECGF2-Group:

p = 0x080000000000000000000000000000000000000c9

a = 0x01

b = 0x020a601907b8c953ca1481eb10512f78744a3205fd

mu = 0x040000000000000000000292fe77e70c12a4234c33

nu = 0x01

g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36

g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1

Khóa công khai

h(x) = 0x03d401df33470c1eb3611ed1b9fd4dd12ffb48cbc1

h(y) = 0x057b470f90c82a900cc4daa27567d15b05d8bdbcb0

Khóa riêng

x = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c933

Dấu hiệu mã hóa ECIES-KEM

Encoding format = uncompressed_fmt

r = 0xa9836a84a1583f601a2f9b2b2432a0aff42c8541

r' = 0xa9836a84a1583f601a2f9b2b2432a0aff42c8541

g~(x) = 0x0619b155dea55122f456a0b4741093a244893c91df

g~(y) = 0x03c75545c65707dd31d9a1a583aba4f107c0c2af51

h~(x) = 0x93c4a6f28021e71e1af8c9da440ab0317e12febd

h~(y) = 0x048d83cad5c3da366af4b7da10f5e13ec45eb1d65d

PEH = 0x0093c4a6f28021e71e1af8c9da440ab0317e12febd

```
z = 0x040619b155dea55122f456a0b4741093a244893c91df03c75545c65707dd31d9a1
    a583aba4f107c0c2af51
C0 = 0x040619b155dea55122f456a0b4741093a244893c91df03c75545c65707dd31d9a
    1a583aba4f107c0c2af51
K = 0x970d1027a42bb88402797cadc8b0822849218339f25189a624c1c7881a09814ede
    d59a9baafafd2ceb516d43b7c6594d1db583ac478bec07bfe37cc3d216a9a2929658
    fae29a7023e266abbdecff6ccecd19bd1f8e51d4db6329af82cae0c07ee093eb3188
    3c57511800057e60407d7d67210ba7366ae3b8b6877a9e81ecb774
```

C.2.5 Véc tơ kiểm tra *ECIES-KEM*

```
-----------------
 ECIES-KEM
-----------------
Kdf=Kdf1(Hash=Sha1())
Keylen=128
CofactorMode=0
OldCofactorMode=0
CheckMode=0
SingleHashMode=0
Group=ECGF2-Group:
p = 0x0800000000000000000000000000000000000000c9
a = 0x01
b = 0x020a601907b8c953ca1481eb10512f78744a3205fd
mu = 0x040000000000000000000292fe77e70c12a4234c33
nu = 0x01
g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36
g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1
-----
Khóa công khai
h(x) = 0x03d401df33470c1eb3611ed1b9fd4dd12ffb48cbc1
h(y) = 0x057b470f90c82a900cc4daa27567d15b05d8bdbcb0
-----
Khóa riêng
x = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c933
-------------------------------------------------
 Dấu hiệu mã hóa ECIES-KEM
-------------------------------------------------
Encoding format = compressed_fmt
r = 0xa9836a84a1583f601a2f9b2b2432a0aff42c8541
```

r' = 0xa9836a84a1583f601a2f9b2b2432a0aff42c8541

g^(x) = 0x0619b155dea55122f456a0b4741093a244893c91df

g^(y) = 0x03c75545c65707dd31d9a1a583aba4f107c0c2af51

h^(x) = 0x93c4a6f28021e71e1af8c9da440ab0317e12febd

h^(y) = 0x048d83cad5c3da366af4b7da10f5e13ec45eb1d65d

PEH = 0x0093c4a6f28021e71e1af8c9da440ab0317e12febd

z = 0x030619b155dea55122f456a0b4741093a244893c91df

C0 = 0x030619b155dea55122f456a0b4741093a244893c91df

K = 0xdc66d10d56868d338b147186fdac210c351150862f94ff3ffcf4fc34b96c2117f1
2e8cf39527419a96066ce00fd856b1742f3ec1865614d901b87ea7b89102417f9b62
775e5806870e73db128fe00a0edd3efe21d93e84a4ae9609ade5838c96da784104db
20170f74b430acde310785d4b66edd09d37f9f32c54ae44442c41f

C.3 Véc tơ kiểm tra cho *PSEC-KEM*

C.3.1 Véc tơ kiểm tra

PSEC-KEM

Kdf=Kdf1(Hash=Sha1())

Keylen=128

Seedlen=64

Group=Modp-Group:

p = 0x8a1b8d83ef967f4e8dc0a423a178b33f31a3aeb743fb332dc020970b44ba95bd29
38eb60365ee9c1b1bda579d8276553758e84eb2a8f89c21f8c08ae12f2aacf

g = 0x5e769d3a6fc9b82acf30800c8afe9631c2b9a1bdee398fd0a920704560513898d9
4e40f3f6fc6a773249d63fc74bba14ceadc203b49f2344a6a22a0a8904c60b

mu = 0xdf0235fe94e74d2d70dbbc887389e5af9ec9ccd7

nu = 0x9e89f7f68e9a2e44b68affab0e53d03763d829685af48fa6405ce08865be6c7ee
7221781300459df024b33e2

Khóa công khai

h = 0x61ddb01fad54cffe21a3a68c1cf388c23493699e74519931e42b8576a9652e47dc
c65f7cd297039268d4a7d6b0337466415647a6f6204b6604d3659127f5c69f

---

Khóa riêng

x = 0x4a401de389f502aa4e1fb066b940a6784626a429

Dấu hiệu mã hóa PSEC-KEM

```
seed = 0x79878e0f7ef84d47753bf4b9a4fa5c33ec1bfa66fa140a3d998770496c613ad
       f8b9b6fdc083d4ac64f1960a9836a84a1583f601b1222a45b9ec718604eb67048

t = 0x583e88b2d550ec4b00419221470e635a63eb0ec74cb9fb6295b57c360e8b68eba9
    631b4e58bd6f118861b03d4dc8b12a3f2cb2e74a5a47e733f34e875891e980963615
    bad107bd2430e8e0d00c4f2d8f9306195b079ba4276900541f0fc7816815366b5190
    34810f6b0d6a6632e251a5ab70d176077701a9c048658a87178a4b94430190607b3a
    52cf66002e4d0251d2cf09f9b19cfbf4793251f7caf9d852a13ad7e37f

u = 0x583e88b2d550ec4b00419221470e635a63eb0ec74cb9fb6295b57c360e8b68eba9
    631b4e

r = 0x0a3b085c410f14847aa9c17ecae644cff418369e

g~ = 0x6e60226637400270f589f53577f00641538d241462441652cb18ffb244414789f
     6cfe71770e5248e74d80524927acd9b0242d273844f8415c4199d1b7037613f

h~ = 0x4ebe32dd0b9aa56cfb712581e7dcf9d8b5a4413544cbf6d09b074fa0d332ff335
     682de79a9a27cfae7a362f84c3e8ab15fca0ce2d1aae6aafc659438225c5559

EG = 0x6e60226637400270f589f53577f00641538d241462441652cb18ffb244414789f
     6cfe71770e5248e74d80524927acd9b0242d273844f8415c4199d1b7037613f

PEH = 0x4ebe32dd0b9aa56cfb712581e7dcf9d8b5a4413544cbf6d09b074fa0d332ff33
      5682de79a9a27cfae7a362f84c3e8ab15fca0ce2d1aae6aafc659438225c5559

SeedMask = 0xeab31c0d24a50c663d7e14d767cc2c4b5e2470deb00b09eab870d28ad0e
           a7c3a3cd05e998ce08c5a6f77a04e2d2b3b84c22d1747f36d5aff7794fbb0
           e27b7a80

MaskedSeed = 0x933492025a5d41214845e06ec3367078b23f8ab84a1f03d721f7a2c3b
             C8b46e5b74b314584ddc69c206ec0e7ae41bf259a12775ce14ffea4e953
             e3d0accd0ac8

C0 = 0x6e60226637400270f589f53577f00641538d241462441652cb18ffb244414789f
     6cfe71770e5248e74d80524927acd9b0242d273844f8415c4199d1b7037613f9334
     92025a5d41214845e06ec3367078b23f8ab84a1f03d721f7a2c3bc8b46e5b74b314
     584ddc69c206ec0e7ae41bf259a12775ce14ffea4e953e3d0accd0ac8

K = 0x58bd6f118861b03d4dc8b12a3f2cb2e74a5a47e733f34e875891e980963615bad1
    07bd2430e8e0d00c4f2d8f9306195b079ba4276900541f0fc7816815366b51903481
    0f6b0d6a6632e251a5ab70d176077701a9c048658a87178a4b94430190607b3a52cf
    66002e4d0251d2cf09f9b19cfbf4793251f7caf9d852a13ad7e37f
```

C.3.2 Véc tơ kiểm tra

```
----------
 PSEC-KEM
----------
Kdf=Kdf1(Hash=Sha1())

Keylen=128

Seedlen=64

Group=ECMoap-Group:

p = 0xffffffffffffffffffffffffffffffffefffffffffffffff
```

a = 0xfffffffffffffffffffffffffffffffefffffffffffffffc

b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

mu = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831

nu = 0x01

g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012

g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811

Khóa công khai

h(x) = 0x1cbc74a41b4e84a1509f935e2328a0bb06104d8dbb8d2130

h(y) = 0x7b2ab1f10d76fde1ea046a4ad5fb903734190151bb30cec2

Khóa riêng

x = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3f3

Dấu hiệu mã hóa PSEC-KEM

Encoding format = uncompressed_fint

seed = 0xae8aeaf179878e0f7ef84d47753bf4b9a4fa5c33ec1bfa66fa140a3d9987704
96c613adf8b9b6fdc083d4ac64f1960a9836a84a1583f601b1222a45b9ec71860

t = 0x336bbe43a45e8bb835c7fe866cf3501e9eff51d26d6d1dc10ae0775897f2f7a63f
9d18df8a6880f99ed846a35852323b31b3b24eb1778db73a1195641b815990cf51ed
62dd220189d600927c0fd9b19f8ddf5bde2305332cdbb202f915c76dca22bce645ea
70b039ebbc12ac76d93590c4884062fca8a33ad29580fea2ddbf72e3746a334b8f5e
f1f772aa09a6b7242df1fc806e605fcd45f50128f6d03db4c0581132f917f4e59d

u = 0x336bbe43a45e8bb835c7fe866cf3501e9eff51d26d6d1dc10ae0775897f2f7a63f
9d18df8a6880f9

r = 0x9a53172304b54d475de3654019156aa4214a478cec066668

g~(x) = 0x87256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc0

g~(y) = 0x0c8e9ddf435a593e775339ed77b9f5f5bcc5097d0819c4b1

h~(x) = 0xb444acd74621f37573fcd0e79eb3a300fefd174b88cee971

h~(y) = 0x393eb322bac28badc949896dbff834da61954c1ebec59885

EG = 0x0487256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc00c8e9ddf435a593
e775339ed77b9f5f5bcc5097d0819c4b1

PEH = 0xb444acd74621f37573fcd0e79eb3a300fefd174b88cee971

SeedMask = 0xda2ab7c99faf1b81e0ad09604c08b0978ebdef27be5bdce29c950fc061a
3bb527eeb1aaae03e4082ba67effefa35fb8fb63c6457b049a1f5dcc0c321
59530f7d

```
MaskedSeed = 0x74a05d38e628958e9e5544273933442e2a47b31452402684668105fdf
             824cb1b128a20756ba52f5eb25aa538b52c9b263556e0f6e876c1eecee2
             677ac794171d

C0 = 0x0487256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc00c8e9ddf435a593
     e775339ed77b9f5f5bcc5097d0819c4b174a05d38e628958e9e5544273933442e2a
     47b31452402684668105fdf824cb1b128a20756ba52f5eb25aa538b52c9b263556e
     0f6e876c1eecee2677ac794171d

 K = 0x9ed846a35852323b31b3b24eb1778db73a1195641b815990cf51ed62dd220189d6
     00927c0fd9b19f8ddf5bde2305332cdbb202f915c76dca22bce645ea70b039ebbc12
     ac76d93590c4884062fca8a33ad29580fea2ddbf72e3746a334b8f5ef1f772aa09a6
     b7242df1fc806e605fcd45f50128f6d03db4c0581132f917f4e59d
```

C.3.3 Véc tơ kiểm tra

```
-----------------
 PSEC-KEM
-----------------
Kdf=Kdf1(Hash=Sha1())

Keylen=128

Seedlen=64

Group=ECModp-Group;

p = 0xfffffffffffffffffffffffffffffffeffffffffffffffff

a = 0xfffffffffffffffffffffffffffffffefffffffffffffffc

b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

mu = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831

nu = 0x01

g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012

g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811

-----

Khóa công khai

h(x) = 0x1cbc74a41b4e84a1509f935e2328a0bb06104d8dbb8d2130

h(y) = 0x7b2ab1f10d76fde1ea046a4ad5fb903734190151bb30cec2

-----

Khóa riêng

x = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3f3
-------------------------------------------------
 Dấu hiệu mã hóa PSEC-KEM
-------------------------------------------------
Encoding format = compressed_fmt
```

```
seed = 0xae8aeaf179878e0f7ef84d47753bf4b9a4fa5c33ec1bfa66fa140a3d998770
       496c613adf8b9b6fdc083d4ac64f1960a9836a84a1583f601b1222a45b9ec718
       60

t = 0x336bbe43a45e8bb835c7fe866cf3501e9eff51d26d6d1dc10ae0775897f2f7a63f
    9d18df8a6880f99ed846a35852323b31b3b24eb1778db73a1195641b815990cf51ed
    62dd220189d600927c0fd9b19f8ddf5bde2305332cdbb202f915c76dca22bce645ea
    70b039ebbc12ac76d93590c4884062fca8a33ad29580fea2ddbf72e3746a334b8f5e
    f1f772aa09a6b7242df1fc806e605fcd45f50128f6d03db4c0581132f917f4e59d

u = 0x336bbe43a45e8bb835c7fe866cf3501e9eff51d26d6d1dc10ae0775897f2f7a63f
    9d18df8a6880f9

r = 0x9a53172304b54d475de3654019156aa4214a478cec066668

g^(x) = 0x87256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc0

g^(y) = 0x0c8e9ddf435a593e775339ed77b9f5f5bcc5097d0819c4b1

h^(x) = 0xb444acd74621f37573fcd0e79eb3a300fefd174b88cee971

h^(y) = 0x393eb322bac28badc949896dbff834da61954c1ebec59885

EG = 0x0387256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc0

PEH = 0xb444acd74621f37573fcd0e79eb3a300fefd174b88cee971

SeedMask = 0xe63cf131069307ca1a2296e0ac3fa1afa25a6476a01254e56903c7301a5
           5dde0bd2cd68a28f2c94c867a0b8e4d6f825c041e63e463f6cabb1a9d290bf
           4c20673

MaskedSeed = 0x48b61bc07f1489c564dadba7d904551606a038454c09ae839317cd0d8
             3d2ada9d14dec55a369a6908e4741480276e2f58774e7453bc9aaa008bf
             8d506a051e13

C0 = 0x0387256b492f43b0cf7cf192faeb26ea354a0e19d1d9bdbbc048b61bc07f1489c
     564dadba7d904551606a038454c09ae839317cd0d83d2ada9d14dec55a369a6908e
     4741480276e2f58774e7453bc9aaa008bf8d506a051e13

K = 0x9ed846a35852323b31b3b24eb1778db73a1195641b815990cf51ed62dd220189d6
    00927c0fd9b19f8ddf5bde2305332cdbb202f915c76dca22bce645ea70b039ebbc12
    ac76d93590c4884062fca8a33ad29580fea2ddbf72e3746a334b8f5ef1f772aa09a6
    b7242df1fc806e605fcd45f50128f6d03db4c0581132f917f4e59d
```

C.3.4 Véc tơ kiểm tra

```
-----------------
 PSEC-KEM
-----------------
Kdf=Kdf1(Hash=Sha1())

Keylen=128

Seedlen=64

-----

Group=ECGF2-Group:
```

p = 0x0800000000000000000000000000000000000000c9

a = 0x01

b = 0x020a601907b8c953ca1481eb10512f78744a3205fd

mu = 0x040000000000000000000292fe77e70c12a4234c33

nu = 0x01

g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36

g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1

Khóa công khai

h(x) = 0x03d401df33470c1eb3611ed1b9fd4dd12ffb48cbc1

h(y) = 0x057b470f90c82a900cc4daa27567d15b05d8bdbcb0

Khóa riêng

x = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c933

Dấu hiệu mã hóa PSEC-KEM

Encoding format = uncompressed.fmt

seed = 0xf179878e0f7ef84d47753bf4b9a4fa5c33ec1bfa66fa140a3d998770496c613
adf8b9b6fdc083d4ac64f1960a9836a84a1583f601b1222a45b9ec718604eb670

t = 0xc6836e810a973cb54f73dc4b573505e2f1fe2b80c67633494fd53af386c73e42c5
c4508d75b270dd95d81fff0518e500e42925ae1f699f498e8273e4884f31407b8a3a
26aa6ee547d4f6b8448b72e9b05f51803bce733cf773bac707fb6127476ba914f74a
5ad10ac0a7b87b59b9699a707a326924528af109U386c65388aebe88ebefa8ee2a1c
9cca32a6d00d9833ca055f0437ee06379416cc139a7fb1900b8d3cadde2

u = 0xc6836e810a973cb54f73dc4b573505e2f1fe2b80c67633494fd53af386c73e42c5
c4508d75

r = 0x02f40b3321460743cc5722182f8529f93ed53cc58c

g~(x) = 0x067ba0d66f34b80ade98971eaec46ae7df42e41864

g~(y) = 0x051879a0b595dacd15353f307a61f741467f1be232

h~(x) = 0x031878816c68b18a57a4528f1ae4247a33a319d4f5

h~(y) = 0x037b354c91ad6607a52fc1222972610dd4d0df1361

EG = 0x04067ba0d66f34b80ade98971eaec46ae7df42e41864051879a0b595dacd15353
f307a61f741467f1be232

PEH = 0x031878816c68b18a57a4528f1ae4247a33a319d4f5

SeedMask = 0x4de1b17b54d897920299ffc57d414cc2f533521f737633dcc953ca8fd86
e087722b7dae4df95d940c29d56fa08c6ead9f418786f092c993e5d6a314f
fc6c6994

```
MaskedSeed = 0xbc9836f55ba66fdf45ecc431c4e5b69ec6df49e5158c27d6f4ca4dff9
             102694dfd3c418b039de40a04d24f9aa145805d5540470f123ebb9a06f4
             f6579c22dfe4
C0 = 0x04067ba0d66f34b80ade98971eaec46ae7df42e41864051879a0b595dacd15353
     f307a61f741467f1be232bc9836f55ba66fdf45ecc431c4e5b69ec6df49e5158c27d
     6f4ca4dff9102694dfd3c418b039de40a04d24f9aa145805d5540470f123ebb9a06f
     4f6579c22dfe4
K = 0xb270dd95d81fff0518e500e42925ae1f699f498e8273e4884f31407b8a3a26aa6e
    e547d4f6b8448b72e9b05f51803bce733cf773bac707fb6127476ba914f74a5ad10a
    c0a7b87b59b9699a707a326924528af10911386c65388aebe88ebefa8ee2a1c9cca3
    2a6d00d9833ca055f0437ee06379416cc139a7fb1900b8d3cadde2
```

C.3.5 Véc tơ kiểm tra

```
-----------------
 PSEC-KEM
-----------------
Kdf=Kdf1(Hash=Sha1())
Keylen=128
Seedlen=64
-----
Group = ECGF2-Group:
p = 0x080000000000000000000000000000000000000c9
a = 0x01
b = 0x020a601907b8c953ca1481eb10512f78744a3205fd
mu = 0x040000000000000000000292fe77e70c12a4234c33
nu = 0x01
g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36
g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1
-----
Khóa công khai
h(x) = 0x03d401df33470c1eb3611ed1b9fd4dd12ffb48cbc1
h(y) = 0x057b470f90c82a900cc4daa27567d15b05d8bdbcb0
-----
Khóa riêng
x = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c933

-------------------------------------------------
 Dấu hiệu mã hóa PSEC-KEM
-------------------------------------------------
```

```
Encoding format = compressed_fmt
seed = 0xf179878e0f7ef84d47753bf4b9a4fa5c33ec1bfa66fa140a3d998770496c613
       adf8b9b6fdc083d4ac64f1960a9836a84a1583f601b1222a45b9ec718604eb670
t = 0xc6836e810a973cb54f73dc4b573505e2f1fe2b80c67633494fd53af386c73e42c5
    c4508d75b270dd95d81fff0518e500e42925ae1f699f498e8273e4884f31407b8a3a
    26aa6ee547d4f6b8448b72e9b05f51803bce733cf773bac707fb6127476ba914f74a
    5ad10ac0a7b87b59b9699a707a326924528af10911386c65388aebe88ebefa8ee2a1
    c9cca32a6d00d9833ca055f0437ee06379416cc139a7fb1900b8d3cadde2
u = 0xc6836e810a973cb54f73dc4b573505e2f1fe2b80c67633494fd53af386c73e42c5
    c4508d75
r = 0x02f40b3321460743cc5722182f8529f93ed53cc58c
g~(x) = 0x067ba0d66f34b80ade98971eaec46ae7df42e41864
g~(y) = 0x051879a0b595dacd15353f307a61f741467f1be232
h~(x) = 0x031878816c68b18a57a4528f1ae4247a33a319d4f5
h~(y) = 0x037b354c91ad6607a52fc1222972610dd4d0df1361
EG = 0x03067ba0d66f34b80ade98971eaec46ae7df42e41864
PEH = 0x031878816c68b18a57a4528f1ae4247a33a319d4f5
SeedMask = 0xefce9dd9b8e3ebd1f563ead211fc08e3a21dca27d0a56ef447c201e85f3
           f33e144f6281fa60d1f94f8d31ee0bb791b276ede83dcda51d37ee35b1bb6
           1f349211
MaskedSeed = 0x1eb71a57b79d139cb216d126a858f2bf91f1d1ddb65f7afe7a5b86981
             65352db9b7db3707a0522de3e9c078012fa71a3cf86bcbcc143f1dab8c5
             dcae7f7a2461
C0 = 0x03067ba0d66f34b80ade98971eaec46ae7df42e418641eb71a57b79d139cb216d
     126a858f2bf91f1d1ddb65f7afe7a5b8698165352db9b7db3707a0522de3e9c0780
     12fa71a3cf86bcbcc143f1dab8c5dcae7f7a2461
K = 0xb270dd95d81fff0518e500e42925ae1f699f498e8273e4884f31407b8a3a26aa6e
    e547d4f6b8448b72e9b05f51803bce733cf773bac707fb6127476ba914f74a5ad10a
    c0a7b87b59b9699a707a326924528af10911386c65388aebe88ebefa8ee2a1c9cca3
    2a6d00d9833ca055f0437ee06379416cc139a7fb1900b8d3cadde2
```

C.4 Véc tơ kiểm tra cho *ACE-KEM*

C.4.1 Véc tơ kiểm tra

```
ACE-KEM
Kdf=Kdf1(Hash=Sha1()) Hash=Sha1()
Keylen=128
CofactorMode=0
Group=Modp-Group:
```

p = 0x8a1b8d83ef967f4e8dc0a423a178b33f31a3aeb743fb332dc020970b44ba95bd29
38eb60365ee9c1b1bda579d8276553758e84eb2a8f89c21f8c08ae12f2aacf

g = 0x5e769d3a6fc9b82acf30800c8afe9631c2b9a1bdee398fd0a920704560513898d9
4e40f3f6fc6a773249d63fc74bba14ceadc203b49f2344a6a22a0a8904c60b

mu = 0xdf0235fe94e74d2d70dbbc887389e5af9ec9ccd7

nu = 0x9e89f7f68e9a2e44b68affab0e53d03763d829685af48fa6405ce08865be6c7ee
7221781300459df024b33e2

Khóa công khai

g' = 0x32785f2307a7cb33cdf124e4349e8e6037040950e51171a4e3d47e0b7280b4798
ec799752e8761d48de565a13962ad951a6322441074a3a7e001dd5bee6448e9

c = 0x84e3b74b067c33ea7ab19ac8e61863e704d56c43e96b14acfb2f2a056f4e72a413
889732006a11bbd34e487e36084fab09c9ec7828308b76412d6a4753e55d31

d = 0x39967584286a71b1dc7fa5a486b26b9cfad2731a5902a8dcc611a5f37eae8d6e9c
c8ad0948344e8edbe80fa607d1c35b2395487ff1aa94b66af9693e20a28027

h = 0x46d73cf934f674c1c9549c7b3e9460c826e2a52c31fd4c5d4cb8da9caddce1b493
eec79ca9a9d6ec5377cf42d8d2968a28c4b183acc9a3bf0590d5bd147e1c14

Khóa riêng

w = 0x4a401de389f502aa4e1fb066b940a6784626a349

x = 0x83bd99b480f6e3ab8b9dc4f410470949f9c9355a

y = 0xa881357fe37c1047061a8192e51b5ebef3a34c23

z = 0x87b8cdd4253bbab89fae7e5c67b5dac6d637f3e7

Dấu hiệu mã hóa ACE-KEM

---

r = 0x346dbd3e7b9fe6b6aebdfcb4077b9b0c6351e94e

u = 0x8a17046e6e2417994139c5b57fb1f8700062fb67d435b5ddfcf4a9d44f6c52fceb
6eb10372486c1c9d01587ad776d285e6b02cdda1d5a80993b6f6d2fc356ac8

u' = 0x7e150711098af13547d25ab9f85615a892faa3842778d8442729dd00cf72687a2
b86af2de61622ebae0823a03656501a01370da1cef809c9809ef2b749c09e0e

h~ = 0x31c724131f8f c689de7a23e51320d265321b1f33db2e161b75f35b66e63064115
648a39c8b28345a3be4290bde2a9d93d6c87ca01f455e1912de76fd5672c755

EU = 0x8a17046e6e2417994139c5b57fb1f8700062fb67d435b5ddfcf4a9d44f6c52fce
b6eb10372486c1c9a01587ad776d285e6b02cdda1d5a80993b6f6d2fc356ac8

EU' = 0x7e150711098af13547d25ab9f85615a892faa3842778d8442729dd00cf72687a
2b86af2de61622ebae0823a03656501a01370da1cef809c9809ef2b749c09e0e

alpha = 0x7265603f0ff462e1940a060c68dd864b16b9ce22

r' = 0xc2114e9865736183434568cd3526c4e00dcc2b52

v = 0x378c692bb3450c9a506348f345019053ef00afd2d436b0e2f435722ecadbf728a3
adda54806d9d759618d5be331907276d87a051c8260e0357c9a0130a8d43e5

EV = 0x378c692bb3450c9a506348f345019053ef00afd2d436b0e2f435722ecadbf728a
3adda54806d9d759618d5be331907276d87a051c8260e0357c9a0130a8d43e5

PEH = 0x31c724131f8fc689de7a23e51320d265321b1f33db2e161b75f35b66e6306411
5648a39c8b28345a3be4290bde2a9d93d6c87ca01f455e1912de76fd5672c755

C0 = 0x8a17046e6e2417994139c5b57fb1f8700062fb67d435b5ddfcf4a9d44f6c52fce
b6eb10372486c1c9d01587ad776d285e6b02cdda1d5a80993b6f6d2fc356ac87e15
0711098af13547d25ab9f85615a892faa3842778d8442729dd00cf72687a2b86af2
de61622ebae0823a03656501a01370da1cef809c9809ef2b749c09e0e378c692bb3
450c9a506348f345019053ef00afd2d436b0e2f435722ecadbf728a3adda54806d9
d759618d5be331907276d87a051c8260e0357c9a0130a8d43e5

K = 0x72c0f34359abf9cbeebb3e52cf1273d14066479a43ef9c93f9fd6f4080a5f27916
98ab80c57d163192b51dc2efa27740d7625db9eb5cfeb6af370e85af5832a035facf
2e2a150cb847338eb173438cdf7126162230917e258cc8a5eee6cb006ec5493ce69d
c91fe3aa2c3c5792e19fea7eeec3bef3db66c4e0b4b36b08507f4e

C.4.2 Véc tơ kiểm tra

ACE-KEM

Kdf=Kdf1(Hash=Sha1())

Hash=Sha1()

Keylen=128

CofactorMode=0

Group=ECModp-Group:

p = 0xffffffffffffffffffffffffffffffffeffffffffffffff

a = 0xffffffffffffffffffffffffffffffffeffffffffffffffc

b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

mu = 0xfffffffffffffffffffffff99def836146bc9b1b4d22831

nu = 0x01

g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012

g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811

Khóa công khai

g'(x) = 0x5a9d4f57936977adcade30ca2350d00096bab728d97499a8

g'(y) = 0xb521a9a56bac905bdf8673a9e83a25ded725bf7a53631b90

c(x) = 0x48dd5e86ac11435b355f9e42ddf6c4509d4d00ed4dc7eb83

c(y) = 0xc4f840332c46a887c58f7e0731ec0f4b11433ea220ee078f

d(x) = 0x603a3be96761734ec5a11096686ec2d252ce79ebc4b9dd5d

d(y) = 0x7aa5a1a995563856c3eb8b03e7c40157009f86e03793dd35

h(x) = 0x28437b3ff9b4371d4eeabf4ca150a5366eb8b950ab779072

h(y) = 0x6569c7ce2e2020768c9ee52e7100e46a06c81365821d2b13

Khóa riêng

w = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3a4

x = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8e44

y = 0xb9a4fa5c33eclbfa66fal46b9514f3e4d2b023da873d4cbb

z = 0xd8b41a0eb3f5f88ce888aed452af12a8e096873e563a9203

Dấu hiệu mã hóa ACE-KEM

Encoding format = uncompressed.fmt

r = 0x9658ad41da2d788ddec09a0265990ccbe903be34126c26a9

u(x) = 0xfd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb02944

u(y) = 0x07eb4a06d8c64b8032a60394736c4d645003bcf412516fdf

u'(x) = 0x83123745fa28135677da40c250bb4254bd0cba6a1c2e2585

u'(y) = 0x6bdf0ade4befa54a9ed1aa7cd9831383a8d17ed3498a19df

h~(x) = 0x456af30e1cbacbb6d069244aa8d1f191ff3ebacdcfaf539b

h~(y) = 0x3c9a22e32c801a9ec37d9e8d6b8a90e5a41ba007204cb4ff

EU = 0x04fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb0294407eb4a06d8c64b8
032a60394736c4d645003bcf412516fdf

EU' = 0x0483123745fa28135677da40c250bb4254bd0cba6a1c2e25856bdf0ade4befa5
4a9ed1aa7cd9831383a8d17ed3498a19df

alpha = 0xa1fd1f8238f51ea06ad52d55df7da4772f730e94

r' = 0x716a5800d4de6612fcf75653538c5eb5571a83040f2d47a4

v(x) = 0x1544105c84f3765f8f1fd490b271a18b0ed1c45e6ecc5071

v(y) = 0xf44c386f466f43eaa29e0434395bb20a218d21715d15316c

EV = 0x041544105c84f3765f8f1fd490b271a18b0ed1c45e6ecc5071f44c386f466f43e
aa29e0434395bb20a218d21715d15316c

PEH = 0x456af30e1cbacbb6d069244aa8d1f191ff3ebacdcfaf539b

C0 = 0x04fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb0294407eb4a06d8c64b8
032a60394736c4d645003bcf412516fdf0483123745fa28135677da40c250bb4254
bd0cba6a1c2e25856bdf0ade4befa54a9ed1aa7cd9831383a8d17ed3498a19df041

544105c84f3765f8f1fd490b271a18b0ed1c45e6ecc5071f44c386f466f43eaa29e
0434395bb20a218d21715d15316c

K = 0x94a6b23344a026db8e3f2669562ad8fC06a529befb032d89a192a460d0340f5a7d
533d79ce5ce59b5c778c2874f3330e03e02056b92d6ae1ad5d9749babe116620b168
d77de156ab53b52b328b0b42c12ef7c74887805ee3fa82c0fb88e6e27ef65e669fa9
43844124c9d5de423d08766dbfa44686fbb5d179239d9096520034

C.4.3 Véc tơ kiểm tra *ACE-KEM*

ACE-KEM

Kdf=Kdf1(Hash=Sha1())

Hash=Sha1()

Keylen=128

CofactorMode=0

Group=ECModp-Group:

p = 0xffffffffffffffffffffffffffffffffeffffffffffffff

a = 0xffffffffffffffffffffffffffffffffefffffffffffffc

b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1

mu = 0xfffffffffffffffffffffff99def836146bc9b1b4d22831

nu = 0x01

g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012

g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811

Khóa công khai

g'(x) = 0x5a9d4f57936977adcade30ca2350d00096bab728d97499a8

g'(y) = 0xb521a9a56bac905bdf8673a9e83a25ded725bf7a53631b90

c(x) = 0x48dd5e86ac11435b355f9e42ddf6c4509d4d00ed4dc7eb83

c(y) = 0xc4f840332c46a887c58f7e0731ec0f4b11433ea220ee078f

d(x) = 0x603a3be96761734ec5a11096686ec2d252ce79ebc4b9dd5d

d(y) = 0x7aa5a1a995563856c3eb8b03e7c40157009f86e03793dd35

h(x) = 0x28437b3ff9b4371d4eeabf4ca150a5366eb8b950ab779072

h(y) = 0x6569c7ce2e2020768c9ee52e7100e46a06c81365821d2b13

Khóa riêng

w = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3a4

x = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8e44

y = 0xb9a4fa5c33ec1bfa66fa146b9514f3e4d2b023da873d4cbb

z = 0xd8b41a0eb3f5f88ce888aed452af12a8e096873e563a9203

Dấu hiệu mã hóa ACE-KEM

Encoding format = compressed_fmt

r = 0x9658ad41da2d788ddec09a0265990ccbe903be34126c26a9

u(x) = 0xfd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb02944

u(y) = 0x07eb4a06d8c64b8032a60394736c4d645003bcf412516fdf

u'(x) = 0x83123745fa28135677da40c250bb4254bd0cba6a1c2e2585

u'(y) = 0x6bdf0ade4befa54a9ed1aa7cd9831383a8d17ed3498a19df

h~(x) = 0x456af30e1cbacbb6d069244aa8d1f19Iff3ebacdcfaf539b

h~(y) = 0x3c9a22e32c801a9ec37d9e8d6b8a90e5a41ba007204cb4ff

EU = 0x03fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb02944

EU' = 0x0383123745fa28135677da40c250bb4254bd0cba6a1c2e2585

alpha = 0xf3af4f830f0cdb0f2c3dd05a2ceca58edb37c97f

r'= 0x8088d4e192dc432148f02aa124d31f0d0ea82c0ab3fb96ea

v(x) = 0x7f0963883bed2203445a315a3d5ca1bb68d3ec74ede13f4f

v(y) = 0x37a45b48bde10a956a0f19fbdf9b2796d33c2be5330b7cf9

EV = 0x037f0963883bed2203445a315a3d5ca1bb68d3ec74ede13f4f

PEH = 0x456af30e1cbacbb6d069244aa8d1f191ff3ebacdcfaf539b

C0 = 0x03fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb029440383123745fa281
35677da40c250bb4254bd0cba6a1c2e2585037f0963883bed2203445a315a3d5ca1
bb68d3ec74ede13f4f

K = 0xd29e265d98f2b3051f2f516ac3cbb96852bec0518bc82ba8660bc5d406a4c82fcd
dc311d935f847963f7a8ea8c0e661109d4bb18306d868aa2a70fcade78d51b0a9468
b309a59ca8d33774caf4966adc156a27243d2added6ee47551eb26f0b9c68c0715e5
d8751ba4ec02e959bbb8b3278468228d2695156ae59f01eca85b58

C.4.4 Véc tơ kiểm tra

ACE-KEM

Kdf=Kdf1(Hash=Sha1())

Hash=Sha1()

Keylen=128

CofactorMode=0

```
Group=ECGF2-Group:
p = 0x080000000000000000000000000000000000000000c9
a = 0x01
b = 0x020a601907b8c953ca1481eb10512f78744a3205fd
mu = 0x040000000000000000000292fe77e70c12a4234c33
nu = 0x01
g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36
g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1
-----
Khóa công khai
g'(x) = 0x052248912facadbe4995dc17e15c2760dca33bef9c
g'(y) = 0x0132e6b3cdf5a6fc94af4bcff2320cIe673e2897df
c(x) = 0x0537639a8b5c088e9c4960986961fc0e7c531df742
c(y) = 0x0733205990c58c743f14aed5550fa5f9a44af020e7
d(x) = 0x013344cd624a8d3af7b38fc6103d795792d951d2a6
d(y) = 0xb47079579331c06ae15065d4cf0b436a20c77f6e
h(x) = 0x059adc6998e2b481aa7d65739ae772187fcc94a933
h(y) = 0x03294c9d5168906f47fe504d5121542a8962fa945b
-----
Khóa riêng
w = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c902
x = 0xa9836a84a1583f601a2f9b2b2432a0aff42c84e8
y = 0x02140a3d998770496c5cbec836b6e8d38e47cc0575
z = 0x02f179878e0f7ef84d45966f119bc634d0f246beec
-----
 Dấu hiệu mã hóa ACE-KEM
-----
Encoding format = uncompressed_fmt
r = 0x015897ecb2c932fa1bb876e25442682b342fab391c
u(x) = 0x05cf2e1de9dcf32160bef47df954851b52a226f463
u(y) = 0x06c65878cff713a57fa53bbfc87497ac73067ed3aa
u'(x) = 0x04783f61a7493d83d76b8178c0935a1830b8708ea8
u'(y) = 0x02aa698207027836dd768207089af0ee1b556aa9d3
h^(x) = 0x0b420ea755ce20f5fa8ea1015d0d2cbf5860767f
h^(y) = 0x055fe3d3d923afdb92c3e44a1e9ae34c249b7f3eb1
```

```
EU = 0x0405cf2e1de9dcf32160bef47df954851b52a226f46306c65878cff713a57fa53
     bbfc87497ac73067ed3aa

EU' = 0x0404783f61a7493d83d76b8178c0935a1830b8708ea802aa698207027836dd76
      8207089af0ee1b556aa9d3

alpha = 0x4a159752a3b5fad5725dce4b7a626e93021de7d5

r' = 0x8aeed29f26765252b9b6fa8e7419c3db8b2766aa

v(x) = 0x01452f7abbd59e15c528aa67738c03829a4facb9d3

v(y) = 0x0374bb51467dc126d5af50e6360f29b8a1427d01c9

EV = 0x0401452f7abbd59e15c528aa67738c03829a4facb9d30374bb51467dc126d5af5
     0e6360f29b8a1427d01c9

PEH = 0x000b420ea755ce20f5fa8ea1015d0d2cbf5860767f

C0 = 0x0405cf2e1de9dcf32160bef47df954851b52a226f46306c65878cff713a57fa53
     bbfc87497ac73067ed3aa0404783f61a7493d83d76b8178c0935a1830b8708ea802
     aa698207027836dd768207089af0ee1b556aa9d30401452f7abbd59e15c528aa677
     38c03829a4facb9d30374bb51467dc126d5af50e6360f29b8a1427d01c9

K = 0x472984597505cf1aec33eeb7477b7546ab14490e65106fce3842a55adbc6aa9828
    e0be5b74785fdf3583023352961ae5d49827a61898e458e4b5b4571472ec6fa05558
    fe870d2954814d49b8560f0d02b039398a5bbd8742d37a463a4056488db1bae29b89
    c5a532e16a4ca8dcd3ab0a9d1fd4a1c42ab27c031a81dc1e53b9ba
```

C.4.5 Véc tơ kiểm tra

```
----------------
 ACE-KEM
----------------
Kdf =Kdf1 (Hash=Sha1())

Hash=Sha()

Keylen=128

CofactorMode=0

-----

Group=ECGF2-Group:

p = 0x080000000000000000000000000000000000000c9

a = 0x01

b = 0x020a601907b8c953ca1481eb10512f78744a3205fd

mu = 0x040000000000000000000292fe77e70c12a4234c33

nu = 0x01

g(x) = 0x03f0eba16286a2d57ea0991168d4994637e8343e36

g(y) = 0xd51fbc6c71a0094fa2cdd545b11c5c0c797324f1

-----
```

Khóa công khai

g'(x) = 0x052248912facadbe4995dc17e15c2760dca33bef9c

g'(y) = 0x0132e6b3cdf5a6fc94af4bcff2320c1e673e2897df

c(x) = 0x0537639a8b5c088e9c4960986961fc0e7c531df742

c(y) = 0x0733205990c58c743f14aed5550fa5f9a44af020e7

d(x) = 0x013344cd624a8d3af7b38fc6103d795792d951d2a6

d(y) = 0xb47079579331c06ae15065d4cf0b436a20c77f6e

h(x) = 0x059adc6998e2b481aa7d65739ae772187fcc94a933

h(y) = 0x03294c9d5168906f47fe504d5121542a8962fa945b

Khóa riêng

w = 0x028d2d26a73f713d3f9d0d5b8ce30d76f4d151c902

x = 0xa9836a84a1583f601a2f9b2b2432a0aff42c84e8

y = 0x02140a3d998770496c5cbec836b6e8d38e47cc0575

z = 0x02f179878e0f7ef84d45966f119bc634d0f246beec

Dấu hiệu mã hóa ACE-KEM

Encoding format = compressed_fmt

r = 0x015897ecb2c932fa1bb876e25442682b342fab391c

u(x) = 0x05cf2e1de9dcf32160bef47df954851b52a226f463

u(y) = 0x06c65878cff713a57fa53bbfc87497ac73067ed3aa

u'(x) = 0x04783f61a7493d83d76b8178c0935a1830b8708ea8

u'(y) = 0x02aa698207027836dd768207089af0ee1b556aa9d3

h~(x) = 0x0b420ea755ce20f5fa8ea1015d0d2cbf5860767f

h~(y) = 0x055fe3d3d923afdb92c3e44a1e9ae34c249b7f3eb1

EU = 0x0305cf2e1de9dcf32160bef47df95485Ib52a226f463

EU' = 0x0204783f61a7493d83d76b8178c0935a1830b8708ea8

alpha = 0xd8e475b97184ee436903685198f494fbaa979816

r' = 0x0267bffb82048609976b545bc4311c57e0869cf07c

v(x) = 0x06904e3cfb1b97cda28216f7caeca93fb005122cd3

v(y) = 0x05d0ae5a5d32e563575bfe4f59a2e5a18151163070

EV = 0x0306904e3cfb1b97cda28216f7caeca93fb005122cd3

PEH = 0x000b420ea755ce20f5fa8ea1015d0d2cbf5860767f

```
C0 = 0x0305cf2e1de9dcf32160bef47df954851b52a226f4630204783f61a7493d83d76
     b8178c0935a1830b8708ea80306904e3cfb1b97cda28216f7caeca93fb005122cd3

K = 0xa0e34391d4fd70e6e780d7edb112ab475d88d3cd9782fd6365aca96b67cfeee964
    1bf7ee8176ec16db20623729a5001ec8e69779ecb3d25e9d128fe22aa4fc3056a032
    969279bb2eeaa2af3e9e5708e8b2b92d2d3f8932adeac7181c7ae03b663883fac467
    e54579cc7531dd3226fd94504c8a8bb60c2ad8cdb2aca4ef8664c9
```

C.5 Véc tơ kiểm tra cho *RSAES*

C.5.1 Véc tơ kiểm tra

```
----------
 RSAES
----------
Rem=Rem1(Hash=Sha1(), Kdf=Kdf1(Hash=Sha1()))

-----

Khóa công khai

n = 109676931776753394141394564514720734236796584022842820507613945978304
    098920529412415619708851314423671483225500317195833435789174491417 8
    718642603750662788855742326532564254342961137739738745427333226003 65
    156233965235292281146938652303374751525426102732530711430473466903 65
    642884618438728252895009596756788538 1

e = 65537

-----

Khóa riêng

n = 109676931776753394141394564514720734236796584022842820507613945978304
    098920529412415619708851314423671483225500317195833435789174491417 8
    718642603750662788855742326532564254342961137739738745427333226003 65
    156233065235292281146938652303374751525426102732530711430473466903 65
    642884618438728252895009596756788538 1

d = 366047199101717654157914355193323865901925886496428126548825829742 50
    355611162241753007459781570721713937967733775394425511406024286292 98
    123635453535902951929063823956409864791888891362846194448669544519 31
    908099484465962690429667141337642117077430417892475442846608311778 34
    928904892102326793526435136473548141

--------------------------------------------------

 Dấu hiệu mã hóa RSAES

--------------------------------------------------

thông báo trong ASCII = "This is a test message !!!"

Thông báo trong xâu octet = 0x20546869732069732061207465737420 6d6573736167
                            6520212121

Nhãn trong ASCII = "label"

Nhãn trong xâu octet = 0x4c6162656c
```

seed = 0xd6e168c5f256a2dcff7ef12facd390f393c7a88d

DataBlock = 0x74341e3c271df3c784e595b804b1f90be0f8042900000000000000000
000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000012054686973206973206120746573
74206d65737361676520212121

DataBlockMask = 0xc325ebbb41a82551d5d0ad4834870a05ef3918c8caae38873f07dc
a43127a4dee36a6ca5970f6c06926037de7df79c4915d83ff705821d
2c46a1fa7bb81b73e27176feb7fd3a45e40b843f1aaebccb1ef4fa7e
e3b9b491a342f43eaaa435efded41e0a3a6ec2eff1f2ed95

MaskedDataBlock = 0xb711f58766b5d696513538f03036f30e0fc11ce1caae38873f07
dca43127a4dee36a6ca5970f6c06926037de7df79c4915d83ff705
821d2c46a1fa7bb81b73e27176feb7fd3a45e40b843f1aaebccb1f
d4ae168aca94f8d062951edec1469bfeb97b79490fa58ad1d3ccb4

SeedMask = 0x281d7cb2d7d5531ed1f9382152d9be9a89a1df09

MaskedSeed = 0xfefc14772583fIc22e87c90efe0a2e691a667784

E = 0x00fefc14772583fIc22e87c90efe0a2e691a667784b711f58766b5d696513538f0
3036f30e0fc11ce1caae38873f07dca43127a4dee36a6ca5970f6c06926037de7df7
9c4915d83ff705821d2c46a1fa7bb81b73e27176feb7fd3a45e40b843f1aaebccb1f
d4ae168aca94f8d062951edec1469bfeb97b79490fa58ad1d3ccb4

C = 0x4712734b1d3c9e43bc8ca30f4d93c88b6273075cb59a63ed2de383cf1a719afc42
99919813f3b775153ef66121fea89821e6ef57427cbb03628884db2aed8e980bce93
1205efdd3d6ee2e2ffc32a8266176ceee26dda7e3ed664c70c97c21187e97e1ccafa
0c1b2e504552ff81d2aa683d89c77b37e9f7818aaf09b7fb585daf

C.5.2 Véc tơ kiểm tra

RSAES

Rem=Rem1(Hash=Sha1(), Kdf=Kdf2(Hash=Sha1()))

Khóa công khai

n = 109676931776753394141394564514720734236796584022842820507613945978304
098920529412415619708851314423671483225500317195833435789174491417871
864260375066278885574232653256425434296113773973874542733322600365156
233965235292281146938652303374751525426102732530711430473466903656428
8461843872825289500959675678853 81

e = 65537

Khóa riêng

n = 109676931776753394141394564514720734236796584022842820507613945978304
098920529412415619708851314423671483225500317195833435789174491417871
86426037506627888557423265325642543429611377397387454273332260036 5

```
    1562339652352922811469386523033747515254261027325307114304734669036
    5
    6428846184387282528950095967567885381

d = 3660471991017176541579143551933238659019258864964281265488258297425
    0
    3556111622417530074597815707217139379677337753944255114060242862929
    8
    1236354535359029519290638239564098647918888913628461944486695445193
    1
    9080994844659626904296671413376421170774304178924754428466083117783
    4
    9289048921023267935264351364735481 41

-------------------------------------------------
 Dấu hiệu mã hóa RSAES
-------------------------------------------------

thông báo trong ASCII = " This is a test message !!!"

Thông báo trong xâu octet = 0x2054686973206973206120746573742 06d6573736167
                            6520212121

Nhãn trong ASCII = "label"

Thông báo trong xâu octet = 0x4c6162656c

seed = 0xd6e168c5f256a2dcff7ef12facd390f393c7a88d

DataBlock = 0x74341e3c271df3c784e595b804b1f90be0f8042900000000000000000
            0000000000000000000000000000000000000000000000000000000000000
            0000000000000000000000000000000000000000000120546869732069732 0
            612074657374206d65737361676520212121

DataBlockMask = 0xcaae38873f07dca43127a4dee36a6ca5970f6c06926037de7df79c
                4915d83ff705821d2c46a1fa7bb81b73e27176feb7fd3a45e40b843f
                1aaebccb1ef4fa7ee3b9b491a342f43eaaa435efded41e0a3a6ec2ef
                fIf2ed951285c5776e259a31024b20beab5cfa02db497746

MaskedDataBlock = 0xbe9a26bb181a2f63b5c23166e7db95ae77f7682f926037de7df7
                  9c4915d83ff705821d2c46a1fa7bb81b73e27176feb7fd3a45e40b
                  843f1aaebccb1ef4fa7ee3b9b491a342f43eaaa435efded41e0a3b
                  4e96879881cdfc61a5a4571a40e945222645cdd83d9d67fb685667

SeedMask = 0x0bfaec4d57584c957e242aa0ef72860f3e109d42

MaskedSeed = 0xdd1b8488a50eee49815adb8f43a116fcadd735cf

E = 0x00dd1b8488a50eee49815adb8f43a116fcadd735cfbe9a26bb181a2f63b5c23166
    e7db95ae77f7682f926037de7df79c4915d83ff705821d2c46a1fa7bb81b73e27176
    feb7fd3a45e40b843f1aaebccb1ef4fa7ee3b9b491a342f43eaaa435efded41e0a3b
    4e96879881cdfc61a5a4571a40e945222645cdd83d9d67fb685667

C = 0x7e72db6f8d55e9ef81e7486a891dd6f3399cd6275f817cf2978a64577fc276e8a8
    b0108d42d671867e22fd76ee2b59cca834a548aeb7b8f1e635ad719a9530b435d2bc
    8d2b15eeb2e162e9573d9765bcc9e4fbededdf6f1ef277aed2449214ffcb998734e1
    d1ba948e84e79f67d2c2a441509899222de4131819718bde30c471
```

C.5.3 Véc tơ kiểm tra

```
-------------------
 RSAES
-------------------
Rem=Rem1(Hash=Sha256( outlen=20), Kdf=Kdf1(Hash=Sha256( outlen=20)))
-----
Khóa công khai
n = 10967693177675339414139456451472073423679658402284282050761394597830
    40989205294124156197088513144236714832255003171958334357891744914178
    71864260375066278885574232653256425434296113773973874542733322600365
    15623396523529228114693865230337475152542610273253071143047346690365
    642884618438728252895009596756788 5381
e = 65537
-----
Khóa riêng
n = 10967693177675339414139456451472073423679658402284282050761394597830
    40989205294124156197088513144236714832255003171958334357891744914178
    71864260375066278885574232653256425434296113773973874542733322600365
    15623396523529228114693865230337475152542610273253071143047346690365
    6428846184387282528950095967567885381
d = 36604719910171765415791435519332386590192588649642812654882582974250
    35561116224175300745978157072171393796773377539442551140602428629298
    12363545353590295192906382395640986479188889136284619444866954451931
    90809948446596269042966714133764211707743041789247544284660831177834
    9289048921023267935264351364735481 41
--------------------------------------------------
 Dấu hiệu mã hóa RSAES
--------------------------------------------------
thông báo trong ASCII = " This is a test message !!!"
Thông báo trong xâu octet = 0x205468697320697320612074657374206d6573736167
                            6520212121
Nhãn trong ASCII = "label"
Nhãn trong xâu octet = 0x4c6162656c
seed = 0xd6e168c5f256a2dcff7ef12facd390f393c7a88d
DataBlock = 0x0e66373f45dcf3dd656151e519f7ee5e3d558d9c000000000000000000
            000000000000000000000000000000000000000000000000000000000000
            000000000000000000000000000000000000000001205468697320697320
            612074657374206d6573736167652021 2121
DataBlockMask = 0x0742ba966813af75536bb6149cc44fc256fd6406df79665bc31dc5
                a62f70535e52c53015b9d37d412ff3c1193439599e1b628774c50d9c
                Cb78d82c425e4521ee47b8c36a4bcffe8b8112a89312fc04420a39de
                99223890e74ce10378bc515a212b97b8a6447ba6a8870278
```

```
MaskedDataBlock = 0x09248da92dcf5ca8360ae7f18533a19c6ba8e99adf79665bc31d
                  c5a62f70535e52c53015b9d37d412ff3c1193439599e1b628774c5
                  0d9ccb78d82c425e4521ee47b8c36a4bcffe8b8112a89312fc0443
                  2a6db6f05118f9946c80230cd9222e0146f2cbd5251cc388a62359

SeedMask = 0x6f0195f38eed2417aa6eb7a365245073e58711db

MaskedSeed = 0xb9e0fd367cbb86cb5510468cc9f7c0807640b956

E = 0x00b9e0fd367cbb86cb5510468cc9f7c0807640b95609248da92dcf5ca8360ae7f1
    8533a19c6ba8e99adf79665bc31dc5a62f70535e52c53015b9d37d412ff3c1193439
    599e1b628774c50d9ccb78d82c425e4521ee47b8c36a4bcffe8b8112a89312fc0443
    2a6db6f05118f9946c80230cd9222e0146f2cbd5251cc388a62359

C = 0x04652a946c1b2a9cade87c46f1995f1a531008cd04bc10d46c850094234c856dd8
    57140a46f9d4d059b5e184dc5f57baac374655911eba712d0b2cd4a92af5ebcfd5ef
    cdc484b8236e85f237c2eec163fe836ad4c002d6604fe0021f4b3835028c98af97e3
    c37c646227e1c488bff9bfd4dc430ac04b4aadc7a9cf6d335e4913
```

C.5.4 Véc tơ kiểm tra

```
-----------------
 RSAES
-----------------
Rem=Rem1(Hash=Sha256( outlen=20), Kdf=Kdf2(Hash=Sha256( outlen=20)))

-----

Khóa công khai

n = 109676931776753394141394564514720734236796584022842820507613945978304
    098920529412415619708851314423671483225500317195833435789174491417871
    864260375066278885574232653256425434296113773973874542733322600365156
    233965235292281146938652303374751525426102732530711430473466903656428
    846184387282528950095967567885381

e = 65537

-----

Khóa riêng

n = 109676931776753394141394564514720734236796584022842820507613945978304
    098920529412415619708851314423671483225500317195833435789174491417871
    864260375066278885574232653256425434296113773973874542733322600365156
    233965235292281146938652303374751525426102732530711430473466903656428
    846184387282528950095967567885381

d = 366047199101717654157914355193323865901925886496428126548825829742503
    556111622417530074597815707217139379677337753944255114060242862929812
    363545353590295192906382395640986479188889136284619444866954451931908
    099484465962690429667141337642117077430417892475442846608311778349289
    04892102326793526435136473548141
```

Dấu hiệu mã hóa RSAES

---

thông báo trong ASCII = " This is a test message !!!"

Thông báo trong xâu octet = 0x20546869732069732061207465737420 6d6573736167
6520212121

Nhãn trong ASCII = "label"

Nhãn trong xâu octet = 0x4c6162656c

seed = 0xd6e168c5f256a2dcff7ef12facd390f393c7a88d

DataBlock = 0x0e66373f45dcf3dd656151e519f7ee5e3d558d9c000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000120546869732069732061207465737420 6d6573736167
6520212121

DataBlockMask = 0xdf79665bc31dc5a62f70535e52c53015b9d37d412ff3c119343959
9e1b628774c50d9ccb78d82c425e4521ee47b8c36a4bcffe8b8112a8
9312fc04420a39de99223890e74ce10378bc515a212b97b8a6447ba6
a8870278f0262727ca041fa1aa9f7b5d1cf7f308232fe861

MaskedDataBlock = 0xd11f516486c1367b4a1102bb4b32de4b8486f0dd2ff3c1193439
599e1b628774c50d9ccb78d82c425e4521ee47b8c36a4bcffe8b81
12a89312fc04420a39de99223890e74ce10378bc515a212b97b8a7
642fcec1f4221183064607be616cd58af21e2e6f96946d030ec940

SeedMask = 0xaed67204e89d4e7fc20317fe06684bc794aad260

MaskedSeed = 0x78371ac11acbeca33d7de6d01aabbdb34076d7aed

E = 0x0078371ac11acbeca33d7de6d1aabbdb34076d7aedd11f516486c1367b4a1102bb
4b32de4b8486f0dd2ff3c1193439599e1b628774c50d9ccb78d82c425e4521ee47b8
c36a4bcffe8b8112a89312fc04420a39de99223890e74ce10378bc515a212b97b8a7
642fcec1f4221183064607be616cd58af2Ie2e6f96946d030ec940

C = 0x4565d8b8edd717044fbee766d4e7b20e17ac060db1a3cc7087cf4dee0adc68eeb1
b91958c83187419730595237a31ddb24277754705db809da5b4b3c2a9a0e711aad62
2fc1e334785d2eb2ea673f883d2036247ac3caac578eb14915126000cbb06a8ad716
a4b39a80c184387e3b170193d2df02864672f5abca52ac0a638419

C.6 Véc tơ kiểm tra cho *RSA-KEM*

C.6.1 Véc tơ kiểm tra

RSA-KEM

Kdf=Kdf1(Hash=Sha1())

keylen=128

Khóa công khai

n = 588811333250269125176193643100928488496664075717980233749P5464783262
385371073265968008202375971398248691849906387495562697857970655080 97
452399642780486933

```
e = 65537

-----

Khóa riêng

n = 588811333250269125176193643100928488496664075717980233749054647832623853710732659680082023759713982486918499063874955626978579706550809745239964278048693
d = 320231355585994818631537452447417399567978358039214023704434972804647939603752030898135380889546180639556447463912452544604470870525967584021098954647926
5
----------------------------------------------------
 Trace for RSA-KEM encrypt
----------------------------------------------------
r = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d764374152e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

R = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d764374152e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

C0 = 0x4603e5324cab9cef8365c817052d954d44447b1667099edc69942d32cd594e4ff
     cf268ae3836e2c35744aaa53ae201fe499806b67dedaa26bf72ecbd117a6fc0

K = 0x5f8de105b5e96b2e490ddecbd147dd1def7e3b8e0e6a26eb7b956ccb8b3bdc1ca975bc57c3989e8fbad31a224655d800c46954840ff32052cdf0d640562bdfadfa263cfccf3c52b29f2af4a1869959bc77f854cf15bd7a25192985a842dbff8e13efee5b7e7e55bbe4d389647c686a9a9ab3fb889b2d7767d3837eea4e0a2f04
```

C.6.2 Véc tơ kiểm tra

```
----------------
 RSA-KEM
----------------
Kdf=Kdf2(Hash=Sha1())

keylen=128

-----

Khóa công khai

n = 588811333250269125176193643100928488496664075717980233749054647832623853710732659680082023759713982486918499063874955626978579706550809745239964278048693

e = 65537

-----

Khóa riêng

n = 588811333250269125176193643100928488496664075717980233749054647832623853710732659680082023759713982486918499063874955626978579706550809745239964278048693
```

d = 3202313555859948186315374524474173995679783580392140237044349728046479396037520308981353808895461806395564474639124525446044708705259675840210989546479265

Dấu hiệu mã hóa RSA-KEM

---

r = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f061Ib8f785d764374152e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

R = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d764374152e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

C0 = 0x4603e5324cab9cef8365c817052d954d44447b1667099edc69942d32cd594e4ffcf268ae3836e2c35744aaa53ae201fe499806b67dedaa26bf72ecbd117a6fc0

K = 0x0e6a26eb7b956ccb8b3bdc1ca975bc57c3989e8fbad31a224655d800c46954840ff32052cdf0d640562bdfadfa263cfccf3c52b29f2af4a1869959bc77f854cf15bd7a25192985a842dbff8e13efee5b7e7e55bbe4d389647c686a9a9ab3fb889b2d7767d3837eea4e0a2f04b53ca8f50fb31225cIbe2d0126c8c7a4753b0807

C.6.3 Véc tơ kiểm tra

RSA-KEM

Kdf=Kdf1(Hash=Sha256(outlen=20))

keylen=128

Khóa công khai

n = 5888113332502691251761936431009284884966640757179802337490546478326238537107326596800820237597139824869184990638749556269785797065508097452399642780486933

e = 65537

Khóa riêng

n = 5888113332502691251761936431009284884966640757179802337490546478326238537107326596800820237597139824869184990638749556269785797065508097452399642780486933

d = 3202313555859948186315374524474173995679783580392140237044349728046479396037520308981353808895461806395564474639124525446044708705259675840210989546479265

Dấu hiệu mã hóa RSA-KEM

---

r = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d764374152e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

R = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d7643741
52e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

C0 = 0x4603e5324cab9cef8365c817052d954d44447b1667099edc69942d32cd594e4ff
cf268ae3836e2c35744aaa53ae201fe499806b67dedaa26bf72ecbd117a6fc0

K = 0x09e2decf2a6e1666c2f6071ff4298305e2643fd510a2403db42a8743cb989de86e
668d168cbe604611ac179f819a3d18412e9eb45668f2923c087c12fee0c5a0d2a8aa
70185401fbbd99379ec76c663e875a60b4aacb1319fa11c3365a8b79a44669f26fb5
55c80391847b05eca1cb5cf8c2d531448d33fbaca19f6410ee1fcb

C.6.4 Véc tơ kiểm tra

RSA-KEM

Kdf=Kdf2(Hash=Sha256(outlen=20))

keylen=128

Khóa công khai

n = 588811333250269125176193643100928488496664075717980233749054647832623853710732659680082023759713982486918499063874955626978579706550809745239964278048693

e = 65537

Khóa riêng

n = 588811333250269125176193643100928488496664075717980233749054647832623853710732659680082023759713982486918499063874955626978579706550809745239964278048693

d = 320231355585994818631537452447417399567978358039214023704434972804647939603752030898135380889546180639556447463912452544604470870525967584021098954647926 5

Dấu hiệu mã hóa RSA-KEM

---

r = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d7643741
52e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

R = 0x032e45326fa859a72ec235acff929b15d1372e30b207255f0611b8f785d7643741
52e0ac009e509e7ba30cd2f1778e113b64e135cf4e2292c75efe5288edfda4

C0 = 0x4603e5324cab9cef8365c817052d954d44447b1667099edc69942d32cd594e4ff
cf268ae3836e2c35744aaa53ae201fe499806b67dedaa26bf72ecbd117a6fc0

K = 0x10a2403db42a8743cb989de86e668d168cbe604611ac179f819a3d18412e9eb456
68f2923c087c12fee0c5a0d2a8aa70185401fbbd99379ec76c663e875a60b4aacb13
19fa11c3365a8b79a44669f26fb555c80391847b05eca1cb5cf8c2d531448d33fbac
a19f6410ee1fcb260892670e0814c348664f6a7248aaf998a3acc6

C.7 Véc tơ kiểm tra cho *HC*

Kết hợp mỗi *KEM* với *DEM* là khá đơn giản, nhưng liệt kê tất cả các kết hợp khác nhau có thể khó và mất nhiều thời gian. Chúng tôi cung cấp ở đây chỉ là một vector thử nghiệm như là một minh họa.

C.7.1 Véc tơ kiểm tra

```
-----------------------------------
 Hệ mật lai ghép
-----------------------------------

---------------
 DEM1
---------------
SC=SC1(BC=AES(keylen=32))
MAC=HMAC(Hash=Sha1(),keylen=20, outlen=20)
---------------
 ACE-KEM
---------------
Kdf=Kdf1(Hash=Sha1())
Hash=Sha1()
Keylen=52
COfactorMode=0
-----
Group=ECModp-Group:
p = 0xffffffffffffffffffffffffffffffffffffffffffffffff
a = 0xfffffffffffffffffffffffffffffffffffffffffffffffc
b = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
mu = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831
nu = 0x01
g(x) = 0x188da80eb03090f67cbf20eb43a18800f4ff0afd82ff1012
g(y) = 0x07192b95ffc8da78631011ed6b24cdd573f977a11e794811
-----
Khóa công khai
g'(x) = 0x5a9d4f57936977adcade30ca2350d00096bab728d97499a8
g'(y) = 0xb521a9a56bac905bdf8673a9e83a25ded725bf7a53631b90
c(x) = 0x48dd5e86ac11435b355f9e42ddf6c4509d4d00ed4dc7eb83
c(y) = 0xc4f840332c46a887c58f7e0731ec0f4b11433ea220ee078f
d(x) = 0x603a3be96761734ec5a11096686ec2d252ce79ebc4b9dd5d
d(y) = 0x7aa5a1a995563856c3eb8b03e7c40157009f86e03793dd35
```

h(x) = 0x28437b3ff9b4371d4eeabf4ca150a5366eb8b950ab779072

h(y) = 0x6569c7ce2e2020768c9ee52e7100e46a06c81365821d2b13

Khóa riêng

w = 0xb67048c28d2d26a73f713d5ebb994ac92588464e7fe7d3a4

x = 0x083d4ac64f1960a9836a84f91ca211a185814fa43a2c8e44

y = 0xb9a4fa5c33ec1bfa66fa146b9514f3e4d2b023da873d4cbb

z = 0xd8b41a0eb3f5f88ce888aed452af12a8e096873e563a9203

Dấu hiệu mã hóa HC

Dấu hiệu mã hóa ACE-KEM

Encoding format = uncompressed_fmt

r = 0x9658ad41da2d788ddec09a0265990ccbe903be34126c26a9

u(x) = 0xfd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb02944

u(y) = 0x07eb4a06d8c64b8032a60394736c4d645003bcf412516fdf

u'(x) = 0x83123745fa28135677da40c250bb4254bd0cba6a1c2e2585

u'(y) = 0x6bdf0ade4befa54a9ed1aa7cd9831383a8d17ed3498a19df

h~(x) = 0x456af30e1cbacbb6d069244aa8d1f191ff3ebacdcfaf539b

h~(y) = 0x3c9a22e32c801a9ec37d9e8d6b8a90e5a41ba007204cb4ff

EU = 0x04fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb0294407eb4a06d8c64b8
032a60394736c4d645003bcf412516fdf

EU' = 0x0483123745fa28135677da40c250bb4254bd0cba6a1c2e25856bdf0ade4befa5
4a9ed1aa7cd9831383a8d17ed3498a19df

alpha = 0xa1fd1f8238f51ea06ad52d55df7da4772f730e94

r' = 0x716a5800d4de6612fcf75653538c5eb5571a83040f2d47a4

v(x) = 0x1544105c84f3765f8fIfd490b271a18b0ed1c45e6ecc5071

v(y) = 0xf44c386f466f43eaa29e0434395bb20a218d21715d15316c

EV = 0x041544105c84f3765f8fIfd490b271a18b0ed1c45e6ecc507If44c386f466f43e
aa29e0434395bb20a218d21715d15316c

PEH = 0x456af30e1cbacbb6d069244aa8d1f191ff3ebacdcfaf539b

C0 = 0x04fd5dd4aa91d2c67b57bfd32f103e5432605f8b903fb0294407eb4a06d8c64b8
032a60394736c4d645003bcf412516fdf0483123745fa28135677da40c250bb4254
bd0cba6a1c2e25856bdf0ade4befa54a9ed1aa7cd9831383a8d17ed3498a19df041
544105c84f3765f8fIfd490b271a18b0ed1c45e6ecc507If44c386f466f43eaa29e
0434395bb20a218d21715d15316c

K = 0x94a6b23344a026db8e3f2669562ad8fc06a529befb032d89a192a460d0340f5a7d
533d79ce5ce59b5c778c2874f3330e03e02056

Dấu hiệu mã hóa DEMI

Thông báo trong ASCII = "the rain in Spain falls mainly on the plain"

Thông báo trong xâu octet = 0x746865207261696e20696e20737061696e2066616c6
c73206d61696e6c79206f6e2074686520706c61696e

Nhãn trong ASCII = "test"

Nhãn trong xâu octet = 0x74657374

k = 0x94a6b23344a026db8e3f2669562ad8fC06a529befb032d89a192a460d0340f5a

k' = 0x7d533d79ce5ce59b5c778c2874f3330e03e02056

c = 0x4e11a54ddf582716b4d46b75adcd446a173ca235b70a944901d2e6f8a583a01993
bfebf63d92496654e5fe271784a310

T = 0x4e11a54ddf582716b4d46b75adcd446a173ca235b70a944901d2e6f8a583a01993
bfebf63d92496654e5fe271784a3107465737400000000000000020

MAC = 0xd4a406ce2e48b63c3d054b91c354b4eeb4a16941

C1 = 0x4e11a54ddf582716b4d46b75adcd446a173ca235b70a944901d2e6f8a583a0199
3bfebf63d92496654e5fe271784a310d4a406ce2e48b63c3d054b91c354b4eeb4a1
6941

C.8 Véc tơ kiểm tra cho *HIME(R)*

C.8.1 Véc tơ kiểm tra

HIME(R) Test Vector N 0.1 (1023-bit)

Hash.eval: SHA1

Hash.len: 20

KDF: KDF1-SHA1

n: 1023-bit

p,q: 341-bit

d: 2

Khóa công khai

---

n = 45 79 68 9f 45 3a 81 0f 87 bd 83 9e bd 99 3e a0
67 0e d9 06 dd 20 23 e6 69 51 7a fa 79 6a 77 9c
7f de 32 26 81 49 15 f7 7c 08 c1 ed fa 7c da 7d
ad be d0 51 66 11 b2 63 bd 4c 96 32 7d 5e 08 b4
03 a5 f9 eb 31 35 c7 87 d3 fe 5e ef 7f 7e ad 42
07 3a fa ad fa cb f0 36 7d 11 2e 08 b3 8f 56 4e

```
      1a 6e c7 9f b7 6f 6a d6 94 9f 88 25 d8 7b 88 8d
      9a 92 9f 0f ab 05 9d f7 04 75 08 6a 08 23 76 4f
Khóa riêng
-----------
P =   18 a0 e0 be 46 81 ae 1a 96 67 de e8 fe 53 8c 39
      3f 47 a5 49 0e 14 aa 67 0a dc 80 2b 8a b2 8d d3
      76 3d 07 d8 34 8a b3 23 2d 65 4f
q =   1d 52 60 8f 7d 37 84 55 85 ff 0a 67 cb 11 cf 2f
      52 84 fc 04 b9 2f e4 b 0 a5 37 16 55 c5 e1 6d 66
      3e 6a 6b 52 8a b7 52 cb 50 42 af
Thông điệp được mã
----------------------
M =   fb e8 e3 3b 8a e5 35 a3 56 45 d8 04 89 75 8e ed
      50 26 34 dc 5d a1 e7 84 71 a7 37 d9 84 72 81 19

Các tham số mã hóa
------------------
L= (xâu rỗng)
Từng bước mã hóa HEM1 của M
-------------------------------

        Check           =   Hash.eval(L)

        Seed            =   random string of octets

        seed'           =   the most significant Klen-bit
                            cleared seed

        DataBlockMask   =   KDF(seed', Elen-Hash.len-1)

        MaskedDataBlock =   DataBlock xor DataBlockMask

        SeedMask        =   KDF(MaskedDataBlock, Hash,len+1

        SeedMask'       =   the most significant Klen-bit
                            cleared SeedMask

        Maskedseed      =   seed' xor SeedMask'
```

```
seed =              86 9d 91 55 d2 66 54 c4 41 c9 18 26 d4 6a b4 d4
                    32 12 6f a7 67

check =             da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90
                    af d8 07 09

DataBlock =         da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90
                    af d8 07 09 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 01 fb e8 e3 3b 8a
                    e5 35 a3 56 45 d8 04 89 75 8e ed 50 26 34 dc 5d
                    a1 e7 84 71 a7 37 d9 84 72 81 19

 DataBlockMask =    2e b9 b8 94 75 9c 38 f1 4f b1 ed d0 42 40 3b 89
                    66 f3 e2 63 75 d7 bf bd 19 eb 14 67 97 dc d7 c1
                    80 bd e2 40 ff 8c 22 16 e5 83 49 0a f0 19 af 5d
                    ec 1d 9c 51 0b f4 cf f1 05 11 8c 48 b5 3c fd 13
                    48 49 32 b7 7f 8f 81 6e 64 dc 39 70 57 63 ed d6
                    f7 4c 2e 4e 0b 1e be bd 93 4d a6 e2 a0 29 5e 95
                    f5 0f 50 04 40 50 54 7e 6b d5 b1

MaskedDataBlock =   f4 80 1b 7a 2b f7 73 fc 7d e4 52 3f d7 20 23 19
                    c9 2b e5 6a 75 d7 bf bd 19 eb 14 67 97 dc d7 c1
                    80 bd e2 40 ff 8c 22 16 e5 83 49 0a f0 19 af 5d
                    ec 1d 9c 51 0b f4 cf f1 05 11 8c 48 b5 3c fd 13
                    48 49 32 b7 7f 8f 81 6e 64 dc 38 8b bf 80 d6 5c
                    12 79 8d 18 4e c6 b8 34 e6 c3 4b b2 86 1d 82 c8
                    54 e8 d4 75 e7 67 8d fa 19 54 a8

SeesMask =          34 59 3e 07 a1 9a 32 d6 08 51 f6 22 c2 1d 90 c2
                    60 03 99 2e a9

SeesMask' =         34 59 3e 07 a1 9a 32 d6 08 51 f6 22 c2 1d 90 c2
                    60 03 99 2e a9

MaskedSeed =        32 c4 af 52 73 fc 66 12 49 98 ee 04 16 77 24 16
                    52 11 f6 89 ce f4 80 1b 7a 2b f7 73 fc 7d e4 52
                    3f d7 20 23 19 c9 2b e5 6a 75 d7 bf bd 19 eb 14
                    67 97 dc d7 c1 80 bd e2 40 ff 8c 22 16 e5 83 49
                    0a f0 19 af 5d ec 1d 9c 51 0b f4 cf f1 05 11 8c
                    48 b5 3c fd 13 48 49 32 b7 7f 8f 81 6e 64 dc 38
                    8b bf 80 d6 5c 12 79 8d 18 CD c6 C 0 34 e6 c3 4b
                    b2 86 1d 82 c8 54 e8 d4 75 e7 67 8d fa 19 54 a8
```

Bản mã

```
C =                 15 73 07 76 08 64 8e c0 d3 66 b6 b5 78 7d 18 c5
                    0c 58 7f 42 ea 50 88 cc 04 c2 68 24 ea e4 8f cb
                    dd 0c c6 47 ce 93 5f 14 2b 8c eb ea b5 1f 78 4b
                    2d 9e 15 80 63 55 bb 9a ed ce 2e 23 a8 ec 7d 8b
                    a6 ae bf bb 1e 54 b7 64 e6 76 51 e9 c5 b6 e7 c2
                    e6 be 38 fc c9 da 94 20 a9 64 d6 92 07 3a e1 8a
                    3c 66 61 f2 45 74 d5 5a 3e a4 ee b0 c2 73 ef f9
                    d9 2e 16 0f 6f d9 3c a4 74 8d 16 01 d9 36 4f e8

Từng bước giải mã HEM1 của E
-------------------------------

Các giá trị trung gian cũng tương tự như trong mã hóa HEM1 của M
```

C.8.2 Véc tơ kiểm tra

Véc tơ kiểm tra HIME(R) No.2 (1023-bit)

```
Hash.eval: SHA1
Hash.len: 20
KDF: KDF1-SHA1
n: 1023-bit
p,q: 341-bit
d: 2
L=(xâu rỗng )

Khóa công khai
----------

n=                  45 79 68 9f 45 3a 81 0f 87 bd 83 9e bd 99 3e a0
                    67 0e d9 06 dd 20 23 e6 69 51 7a fa 79 6a 77 9c
                    7f de 32 26 81 49 15 f7 7c 08 c1 ed fa 7c da 7d
                    ad be d0 51 66 11 b2 63 bd 4c 96 32 7d 5e 08 b4
                    03 a5 f9 eb 31 35 c7 87 d3 fe 5e ef 7f 7e ad 42
                    07 3a fa ad fa cb f0 36 7d 11 2e 08 b3 8f 56 4e
                    1a 6e c7 9f b7 6f 6a d6 94 9f 88 25 d8 7b 88 8d
                    9a 92 9f 0f ab 05 9d f7 04 75 08 6a 08 23 76 4f

Khóa riêng
----------

p=                  18 a0 e0 be 46 81 ae 1a 96 67 de e8 fe 53 8c 39
                    3f 47 a5 49 0e 14 aa 67 0a dc 80 2b 8a b2 8d d3
                    76 3d 07 d8 34 8a b3 23 2d 65 4f

q=                  1d 52 60 8f 7d 37 84 55 85 ff 0a 67 cb 11 cf 2f
                    52 84 f c04 b9 2f e4 b0 a5 37 16 55 c5 e1 6d 66
                    3e 6a 6b 52 8a b7 52 cb 50 42 af
```

Ví dụ 2.1

M= d8 5a 93 45 a8 60 51 e7 30 71 62 00 56 b9 20 e2
19 00 58 55 a2 13 a0 f2 38 97 cd cd 73 1b 45 25
7c 77 7f e9

seed= dd dd 87 71 fe c4 8b 83 a3 1e e6 f 5 92 c4 cf d4
be 88 17 4f 3b

C= 0e 38 31 4b 1e f6 49 fd d1 d6 1c 81 7b 21 d4 30
d5 b3 79 ca 3e 05 34 Id d9 53 34 d4 2a e9 7e 73
11 32 83 6b 71 52 91 2b 7b d2 b7 45 85 c3 c9 1f
c0 20 dd 34 8c fe d0 a3 f2 3c e1 4c 27 ca da 07
d1 8c cc a8 ad ff 6b 2c bb 85 48 2c ae 3e cb ac
6a 27 f9 5f f8 45 19 41 19 60 d7 1b 9e 8b 2c 34
0c 71 11 7c 0c 25 58 80 e9 2f 8c 02 62 51 0f 36
22 04 el 9e fe 5c 73 a0 11 47 13 36 09 d2 d5 2f

C.8.3 Véc tơ kiểm tra

Véc tơ kiểm tra HIME(R) No.3 (1023-bit)

Hash.eval: SHA1
Hash.len: 20
KDF: KDF1-SHA1
n: 1023-bit
p,q: 341-bit
d: 2
L= (Xâu rỗng)

Khóa công khai
---

n = 64 46 ed 6c 4b a2 a1 ad e0 c3 6f ba 47 1c 02 8c
3a fc ba 00 3b 8f 8f 52 f2 1e c7 8f fb 42 33 83
4f a5 75 f4 ff 7d bb 1a be c8 b5 93 57 e9 69 dc
30 44 a5 7d 3a b9 1d bd 1a 49 2e 87 fd f7 57 21
c0 52 43 4f 1a 00 63 f 6 c5 18 be d1 35 93 3a f4
e3 c9 7b 29 30 c0 5b d1 1c bb c2 06 b4 fd b3 50
eb e0 5b d4 38 36 a0 ec b6 5b af f2 d0 da 1a 47
08 4e 8d f2 b7 a4 35 f1 6b 9e 5f 3d ad b6 62 97

Khóa riêng
---

p = 1c a0 e0 be 46 91 ae 1a 96 67 de e8 fe 53 8c
3f 47 a5 49 0e 14 aa 67 0a dc 80 2b 8a b2 8d
76 3d 07 d8 34 8a b3 23 2d 66 c7

```
q =             1f 52 60 C0 7d 37 84 55 85 ff 0a 6b cb 11 cf
                52 84 fc 04 b9 2f e4 b0 a5 37 16 55 c5 e1 6d
                3e 6a 6b 52 8a b7 52 cb 50 47 c7

Ví dụ 3.1
-----------

M =             ab 3c d9 d8 8d 98 40 3b 38 b4 09 95 fd 6f f4 1a
                1a cc 8a da

seed =          6a 90 87 4e ef ce 8f 2c cc 20 e4 f2 74 1f b0 a3
                3a 38 48 ae c9

C =             5d c8 e9 ec 7c 33 f6 a4 5e 63 26 43 e8 b6 57 32
                38 76 b7 17 ac dd 30 a2 fc f3 fd 8d 22 35 f1 ec
                3e 75 ac e1 c8 e8 4e 2e dd 5a b9 6f 9e bd 48 68
                79 3f 20 f2 af 90 60 57 da b1 2e f1 d3 07 5e 0d
                b3 39 39 f6 50 50 1c 27 8f 59 25 4d 3d 64 81 3d
                8c da e2 98 37 c5 76 65 53 43 15 eb 54 e9 3f ac
                b3 53 7b 95 b7 c0 0e 92 36 38 f9 7b 4b 1d 6d 5e
                0a bb a3 4c 8a b5 13 4b 2a 7b 47 4c 37 af 01 c3

Ví dụ 3.2
-----------

M =             ef f2 9d da 4f 2d 51 64 73 f1 10 64 c9 96 fa 26
                56 d2 c3 c0 93 44 05 af be de 22 2d 9d 31 7c f2
                39 fe 8a 16

seed =          95 29 7b 0f 95 a2 fa 67 d0 07 07 d6 09 df d4 fc
                05 c8 9d af c2

C =             41 57 55 2e 9d ab ff 3c 08 54 6e 5b c8 0d e0 07
                c9 64 7a 09 be e5 9c a0 8b 9b f9 f6 be 14 58 e0
                8b 0a 1d 47 ff 2c 7d 5f 42 75 30 32 3b 1c ce dc
                c3 ac c3 8e 43 be ca be 57 39 4f 29 8d 8e dd cd
                7d 26 bf 72 1f 6a 3f c5 55 19 c9 04 cd eb 94 f4
                3f c8 15 a6 fb fc 43 0a 0e 25 05 14 5d ac 22 b6
                c9 a5 57 27 89 ca 0e 52 4d fb 87 91 71 fe 80 f7
                d8 78 c4 3a b0 d7 7c 3e 66 39 c6 ac 28 c3 85 8f
```

C.8.4 Véc tơ kiểm tra

Véc tơ kiểm tra HIME(R) No.4 (1023-bit)

Hash.eval: SHA1

Hash.len: 20

KDF: KDF1-SHA1

```
n: 1344-bit
p,q:=448-bit
d: 2

Khóa công khai
----------

n =             87 e5 d1 89 c4 66 b7 98 0f 48 50 1f 36 10 80 2d
                86 9d 91 55 d2 66 54 c4 41 c9 18 26 d4 6a b4 d4
                32 12 6f 6a 0d 8c 39 22 f2 6b 35 42 80 1b 0b cf
                fb e8 e3 3b 8a e5 35 a3 56 45 d8 04 89 75 8e ed
                50 26 34 dc 5d a1 e7 84 71 a7 37 d9 84 72 81 19
                92 d2 09 25 dc 4b a8 1f 7f ee 3a f5 71 cb b3 f2
                c6 68 a6 93 56 c0 72 aa ba 4b 3e 4d 19 9a e1 84
                07 33 e8 e4 df 9f 43 89 c4 f0 4c fc ad 99 63 67
                70 cb d8 9c 70 a7 87 eb 73 5f 91 f6 cb fc 5a 22
                b9 72 63 1d e2 28 3d 1e 84 9c 82 0f f6 2a d9 42
                c4 c5 fd ea 0b 8a 66 63

Khóa riêng
-----------

p =             d1 ad c7 92 77 f0 e7 16 de bf 8f a0 42 be 80 8c
                55 cc da 53 34 2c fd a4 60 d6 d5 68 e3 85 b7 89
                36 b3 28 ad fa 67 9b 80 b0 ec 94 5c 9c da 67 1b
                63 da 57 f 9 e4 66 87 03

q =             ca 92 db 5a 3a 53 22 0a 4a 92 1e e6 e9 af da ce
                12 7e a7 67 0e df df c2 68 42 a1 ad 62 79 05 1b
                e6 82 00 c0 09 83 db a2 36 ab e1 6d 37 41 45 cd
                b1 f6 da b2 cc 81 d8 0b

Thông điệp được mã
----------------------

M =             fb e8 e3 3b 8a e5 35 a3 56 45 d8 04 89 75 8e ed
                50 26 34 dc 5d a1 e7 84 71 a7 37 d9 84 72 81 19

Các tham số mã hóa
-------------------

L = (Xâu rỗng)

Từng bước mã hóa HEM1 của M
-------------------------------

        check               = Hash.eval1(1)
```

```
        Seed                = random string of octets

        seed'               = the most significant Klen-bit cleared seed

        DataBlockMask       = KDF(seed', Elen-Hash.len-1)

        MaskedDataBlock     = DataBlock xor DataBlockMask

        SeedMask            = KDF(MaskedDataBlock, Hash,len+1

        SeedMask'           = the most significant Klen-bit cleared
                              SeedMask

        Maskedseed          = seed' xor SeedMask'

seed =              86 9d 91 55 d2 66 54 c4 41 c9 18 26 d4 6a b4 d4
                    32 12 6f a7 67

seed'=              06 9d 91 55 d2 66 54 c4 41 c9 18 26 d4 6a b4 d4
                    32 12 6f a7 67

check =             da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90
                    af d8 07 09

DataBlock =         da 39 a3 ee 5e 6b 4b 0d 32 55 bf ef 95 60 18 90
                    af d8 07 09 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                    00 00 01 fb e8 e3 3b 8a e5 35 a3 56 45 d8 04 89
                    75 8e ed 50 26 34 dc 5d a1 e7 84 71 a7 37 d9 84
                    72 81 19

DataBlockMask =     2e b9 b8 94 75 9c 38 f1 4f b1 ed d0 42 40 3b 89
                    66 f3 e2 63 75 d7 bf bd 19 eb 14 67 97 dc d7 c1
                    80 bd e2 40 ff 8c 22 16 e5 83 49 0a f0 19 af 5d
                    ec 1d 9c 51 0b f4 cf f1 05 11 8c 48 b5 3c fd 13
                    48 49 32 b7 7f 8f 81 6e 64 dc 39 70 57 63 ed d6
                    f7 4c 2e 4e 0b 1e be bd 93 4d a6 e2 a0 29 5e 95
                    f5 0f 50 04 40 50 54 7e 6b d5 b1 7a 0d 04 a8 ca
                    27 26 10 33 dc 95 e2 e4 5b 32 08 d1 7e 13 90 61
                    9b 4b d7 c4 15 04 ea 4e fd fc b7 24 dc 0e 75 01
                    a3 19 f9

MaskedDataBlock =   f4 80 1b 7a 2b f7 73 fc 7d e4 52 3f d7 20 23 19
                    c9 2b e5 6a 75 d7 bf bd 19 eb 14 67 97 dc d7 c1
                    80 bd e2 40 ff 8c 22 16 e5 83 49 0a f0 19 af 5d
                    ec 1d 9c 51 0b f4 cf f1 05 11 8c 48 b5 3c fd 13
```

```
                48 49 32 b7 7f 8f 81 6e 64 dc 39 70 57 63 ed d6
                f7 4c 2e 4e 0b 1e be bd 93 4d a6 e2 a0 29 5e 95
                f5 0f 50 04 40 50 54 7e 6b d5 b1 7a 0d 04 a8 ca
                27 26 11 c8 34 76 d9 6e be 07 ab 87 3b cb 94 e8
                ee c5 3a 94 33 30 36 13 5c 1b 33 55 7b 39 ac 85
                d1 98 e0

SeedMask =      6d ad 57 05 63 99 d1 1f 44 a1 20 bd 1a ff 15 47
                31 f0 79 59 ea

SeedMask' =     6d ad 57 05 63 99 d1 1f 44 a1 20 bd 1a ff 15 47
                31 f0 79 59 ea

MaskedSeed =    6b 30 c6 50 b1 ff 85 db 05 68 38 9b ce 95 a1 93
                03 e2 16 fe 8d

M =             6b 30 c6 50 b1 ff 85 db 05 68 38 9b ce 95 a1 93
                03 e2 16 fe 8d f4 80 1b 7a 2b f7 73 fc 7d e4 52
                3f d7 20 23 19 c9 2b e5 6a 75 d7 bf bd 19 eb 14
                67 97 dc d7 c1 80 bd e2 40 ff 8c 22 16 e5 83 49
                0a f0 19 af 5d ec 1d 9c 51 0b f4 cf f1 05 11 8c
                48 b5 3c fd 13 48 49 32 b7 7f 8f 81 6e 64 dc 39
                70 57 63 ed d6 f7 4c 2e 4e 0b 1e be bd 93 4d a6
                e2 a0 29 5e 95 f5 0f 50 04 40 50 54 7e 6b d5 b1
                7a 0d 04 a8 ca 27 26 11 c8 34 76 d9 6e be 07 ab
                87 3b cb 94 e8 ee c5 3a 94 33 30 36 13 5c 1b 33
                55 7b 39 ac 85 d1 98 e0

Bản mã
--------------

C =             79 67 be c3 42 fc bb 72 e2 75 e6 68 73 bf 3c 68
                c9 81 05 df cd 08 82 28 0d cd 0a 45 26 1c 7d 68
                ee 46 79 6b 49 b3 66 74 81 9c c8 84 05 98 5a 44
                8b f9 17 4d 93 c5 ce d4 fc b2 00 ff 65 e5 fd 58
                cb 3c f1 1f 65 d8 3c 24 f2 78 c6 ba 88 44 79 32
                86 af 8e b3 9b f5 9d 02 65 21 b8 2b 09 c5 40 05
                92 1e a7 4c 56 87 57 4b 3a 9f c1 8b 86 c2 90 eb
                21 34 17 bb 3b e2 b4 b4 a6 be d1 54 d8 6f 1e b4
                d6 be cd 65 05 a6 00 23 40 31 d8 c3 a7 ce c1 03
                b8 14 17 30 37 8a 33 05 c6 3b 1d bd ed e7 bf ac
                6f 8d e8 22 34 90 db 7a
```

Từng bước giải mã HEM1 của E

Các giá trị trung gian cung tương tự như trong mã hóa HEM1 của M

C.8.5 Véc tơ kiểm tra

Véc tơ kiểm tra HIME(R) No.5 (1344-bit)

Hash.eval: SHA1

Hash.len: 20

KDF: KDF1-SHA1

n: 1344-bit

p,q:=448-bit

d: 2

L= (xâu rỗng)

Khóa công khai

---

n = 87 e5 d1 89 c4 66 b7 98 0f 48 50 1f 36 10 80 2d
86 9d 91 55 d2 66 54 c4 41 c9 18 26 d4 6a b4 d4
32 12 6f 6a 0d 8c 39 22 f2 6b 35 42 80 1b 0b cf
fb e8 e3 3b 8a e5 35 a3 56 45 d8 04 89 75 8e ed
50 26 34 dc 5d a1 e7 84 71 a7 37 d9 84 72 81 19
92 d2 09 25 dc 4b a8 1f 7f ee 3a f5 71 cb b3 f2
c6 68 a6 93 56 c0 72 aa ba 4b 3e 4d 19 9a e1 84
07 33 e8 e4 df 9f 43 89 c4 f0 4c fc ad 99 63 67
70 cb d8 9c 70 a7 87 eb 73 5f 91 f6 cb fc 5a 22
b9 72 63 1d e2 28 3d 1e 84 9c 82 0f f6 2a d9 42
c4 c5 fd ea 0b 8a 66 63

Khóa riêng

---

p = d1 ad c7 92 77 f0 e7 16 de bf 8f a0 42 be 80 8c
55 cc da 53 34 2c fd a4 60 d6 d5 68 e3 85 b7 89
36 b3 28 ad fa 67 9b 80 b0 ec 94 5c 9c da 67 1b
63 da 57 f 9 e4 66 87 03

q = ca 92 db 5a 3a 53 22 0a 4a 92 1e e6 e9 af da ce
12 7e a7 67 0e df df c2 68 42 a1 ad 62 79 05 1b
e6 82 00 c0 09 83 db a2 36 ab e1 6d 37 41 45 cd
b1 f6 da b2 cc 81 d8 0b

Ví dụ 5.1

M = ab 3c d9 d8 8d 98 40 3b 38 b4 09 95 fd 6f f4 1a
1a cc 8a da

seed = 6a 90 87 4e ef ce 8f 2c cc 20 e4 f2 74 1f b0 a3
3a 38 48 ae c9

```
C =             71 9d 83 07 f8 75 1e b4 51 be d2 20 28 d0 6a 2e
                e4 25 16 fe 5b 3d bd e2 3f a5 50 85 6f 06 7e 2c
                3b f7 0a ca ae 1a 8d 36 39 13 36 4a e3 1f 9f 74
                b1 ed 53 b7 81 97 95 9a 9b e3 c4 fd 33 46 5c 4b
                55 53 b5 a5 ba 21 07 36 7e ba ec f9 9f 2a 15 80
                6c 9b 8a d0 f5 70 2e 61 2c 12 26 6a 56 90 7c 9e
                91 e9 2a d7 b2 3a 14 43 fa 95 94 b5 23 b5 96 0f
                d5 5d 6d 8f 1b f9 fe fc bf dd 72 cf 3c bd 21 5f
                7c dc 01 51 c6 50 04 b1 f6 ae eb 4c a8 b9 be f4
                f6 11 9c fd d9 09 34 14 8f 1e af 1d 54 6e 63 af
                41 27 74 54 5f d8 19 39

Ví dụ 5.2
-----------

M =             ef f2 9d da 4f 2d 51 64 73 f1 10 64 c9 96 fa 26
                56 d2 c3 c0 93 44 05 af be de 22 2d 9d 31 7c f2
                39 fe 8a 16

seed =          95 29 7b 0f 95 a2 fa 67 d0 07 07 d6 09 df d4 fc
                05 c8 9d af c2

C =             1b 17 e0 25 92 33 25 83 0b 77 1f cf 9e 17 53 57
                5b 6e 07 f2 59 5a c4 50 8a fd 00 58 3c b4 32 9c
                95 7a 46 a1 4c 21 68 99 bd 35 93 d9 96 8a 1f 7f
                e8 f4 bb 9a fe 35 89 01 5b 57 c9 0a e5 a0 00 ec
                57 9c 0c e8 1e b4 fc 51 81 d9 fe ba e0 35 38 bd
                60 f1 4e 6c fb 04 28 e7 43 d6 0f 26 ce a6 40 30
                0b 7d 1f 08 d8 18 22 94 ed 3f 53 1c ab eb ee b4
                89 f7 37 85 88 be 1b ba 8a 45 a1 ee 5e 69 fd ed
                31 d4 18 65 5a a5 c8 ae 9d 17 4b fc c2 31 68 cb
                e7 d5 f4 1e 5b 55 ee f1 f1 9e 1c c6 cd 9e a1 31
                fe d9 5e d0 56 52 cc 8b
```

C.8.6 Véc tơ kiểm tra

Véc tơ kiểm tra HIME(R) No.6 (1344-bit)

Hash.eval: SHA1

Hash.len: 20

KDF: KDF1-SHA1

n: 1344-bit

p,q:=448-bit

d: 2

L= (Xâu rỗng)

Khóa công khai

---

n = d1 10 0b fe 0e 2c f5 d0 76 12 57 dc 34 e4 13 bb
02 21 f7 c4 27 cd ee 41 d1 b0 78 28 ac b3 c8 80
cc b5 26 db ea d4 39 58 fb 71 07 e6 28 1b 73 0f
6c f1 64 be 28 9f df b9 f2 9b ae 88 e4 e3 a3 38
c8 45 1e 70 c3 0b b1 f1 44 87 e8 49 9b 67 55 11
e5 ac 3d ff 50 91 05 3b 46 59 cc 3b 23 c2 99 a7
0f a3 ed ea e4 63 45 6e e8 35 99 90 68 c3 a4 5e
b7 45 47 7c 79 d8 ed ac c5 8a cc 16 9a 67 7d 96
ab f9 4f 7a 1b 55 3b 56 35 c0 62 37 21 0d 9d 48
63 c1 f1 27 64 15 b9 ba 1e ae a0 73 d2 f9 3d 32
11 8a 73 b2 61 bd 13 1b

Khóa riêng

---

p = f1 ad c7 92 77 f0 e7 16 de bf 8f a0 42 be 80 8c
55 cc da 53 34 2c fd a4 60 d6 d5 68 e3 85 b8 89
36 b3 28 ad fa 67 9b 80 b0 ec 94 5c 9c da 67 1b
63 da 57 f9 e4 cc 6f 6b

q = ea 92 db 5a 3a 53 22 0a 4a 92 1e e6 e9 af da ce
12 7e b7 67 0e df df c2 68 42 a1 ad 62 79 05 1b
e6 82 00 c0 09 83 db a2 36 ab e1 6d 37 41 45 cd
b1 f6 da b2 cc 8a 2e 73

Ví dụ 6.1

M = d8 5a 93 45 a8 60 51 e7 30 71 62 00 56 b9 20 e2
19 00 58 55 a2 13 a0 f2 38 97 cd cd 73 1b 45 25
7c 77 7f e9

seed = dd dd 87 71 fe c4 8b 83 a3 1e e6 f5 92 c4 cf d4
be 88 17 4f 3b

C = 6b ab 8d 82 90 ec 05 7c d3 e8 b2 7f e9 e5 38 d1
25 cd c6 13 38 7d e4 e5 f0 df 23 24 fe 58 3d 91
91 79 71 f5 1a 93 ae 13 29 9d 47 5d 4c be 45 be
9a 8d f1 71 5c 32 c6 cc da 85 ea b6 ca ed da 1b
92 48 42 a8 4a f9 aa 40 9f 02 97 f9 73 74 b2 71
18 18 85 07 9e c4 02 1a 95 f2 18 08 28 69 be 00
9d da c2 45 b3 f7 fe be 58 07 d3 65 67 59 d4 b2
14 f3 d2 1a e1 e5 10 49 90 f8 7d e5 70 02 13 87
96 b0 cc b9 15 3b 6b 2a 0f 25 52 90 a1 d4 1c 44
45 c1 07 ae 90 da 54 fb 3b b4 44 88 fc d6 1e 26

```
                2b a7 62 ce a8 7f df 91

Ví dụ 6.2
-----------

M =             da fb f0 38 e1 80 d8 37 c9 63 66 df 24 c0 97 b4
                ab 0f ac 6b df 59 0d 82 1c 9f 10 64 2e 68 1a d0

seed =          cc 88 53 d1 d5 4d a6 30 fa c0 04 f4 71 f2 81 c7
                b8 98 2d 82 24

C =             b6 dd 2f 33 ee 04 52 1f a9 17 d3 30 9e f1 2f a1
                cb 93 de 75 4c fe a3 b1 bf e7 dd 9c a4 51 44 85
                bf dc 57 31 c1 6d c2 78 7c 14 6d f6 f4 98 68 45
                b8 ce aa c6 59 cf 42 12 f9 31 07 4c e6 e5 26 9e
                62 96 06 46 db 64 88 12 42 fe 7f c7 6b 7a ed f9
                0d 30 ca 8f 03 62 23 69 25 cb ce 11 d9 4a 76 e4
                6e c7 bf c9 e7 b6 02 a0 f5 32 73 3b a8 b1 e6 f5
                7c 07 45 28 bf b2 df c0 3c d7 1f 7f d6 3b 7e 4b
                66 68 f5 89 e4 c8 d8 df 3c 0f 40 c1 04 ce b5 7b
                6e 45 06 e9 a1 f5 be 79 c1 40 be e8 f2 11 dd 85
                96 43 81 a6 ef d5 14 7b
```

Thư mục tài liệu tham khảo

[1]. SO/IEC 10116:1997, *Information technology — Security techniques — Modes of operation for an n-bit block cipher*

[2]. ISO/I EC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

[3]. ISO/IEC11770 (all parts), *Information technology — Security techniques — Key management*

[4]. ISO/IEC 15946-1:2002, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part I: General*

[5]. ISO/IEC 18031:2005, *Information technology — Security techniques — Random bit generation*

[6]. ISO/IEC 18032:2005, *Information technology — Security techniques — Prime number generation*

[7]. ISO/IEC18033-1:2005, *Information technology — Security techniques — Encryption algorithms — Part I: General*

[8]. [8] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie-Hellman problem. Cryptology ePrint Archive, Report 1999/007, 1999. <http://eprint.iacr.org>

[9]. M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In Topics in Cryptology - CT-RSA 2001, pages 143-158, 2001. Springer LNCS 2045

[10]. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: security proofs and improvements. In Advances in Cryptology - Eurocrypt 2000, pages 259-274, 2000

[11]. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: analysis of the DES modes of operation. In 38th Annual Symposium on Foundations of Computer Science, 1997

[12]. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In Advances in Cryptology-Crypto '98, pages 26-45, 1998

[13]. M. Bellare, J. Kilian, and P. Rogaway. On the security of cipher block chaining. In Advances in Cryptology - Crypto '94, pages 341-358, 1994

[14]. M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In First ACM Conference on Computer and Communications Security, pages 62- 73, 1993

[15]. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In Advances in Cryptology - Eurocrypt '94, pages 92-111, 1994

[16]. I. Blake, G. Seroussi, and N. Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999

[17]. D. Boneh. The Decision Diffie-Hellman Problem. In Ants-III, pages 48-63, 1998. Springer LNCS 1423

[18]. R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. <http://eprint.iacr.org>

[19]. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In 30th Annual ACM Symposium on Theory of Computing, pages 209-218, 1998

[20]. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Advances in Cryptology-Crypto '98, pages 13-25, 1998

[21]. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Cryptology ePrint Archive, Report 2001/108, 2001. <http://eprint.iacr.org>

[22]. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In 23rd Annual ACM Symposium on Theory of Computing, pages 542-552, 1991

[23]. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography, 1998. Manuscript (updated, full length version of STOC paper)

[24]. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. SIAM J. Comput., 30(2):391 -437, 2000

[25]. T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31:469-472, 1985

[26]. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Advances in Cryptology-Crypto '99, pages 537-554, 1999

[27]. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In Advances in Cryptology-Crypto 2001, pages 260-274, 2001

[28]. S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28:270-299, 1984

[29]. Self-evaluation report: HIME(R) cryptosystem, Oct. 2003. Available from <http://www.sdl.hitachi.co.jp/crypto/hime/index.html>

[30]. H. W. Lenstra. Finding isomorphisms between finite fields. Math. Comp., 56:329-347, 1991

[31]. J. Manger. A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS # 1 v2.0. In Advances in Cryptology-Crypto 2001, pages 230-238, 2001

[32]. U. Maurer and S. Wolf. The Diffie-Hellman protocol. Designs, Codes, and Cryptography, 19:147-171, 2000

[33]. M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In 38th Annual Symposium on Foundations of Computer Science, 1997

[34]. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In 22nd Annual ACM Symposium on Theory of Computing, pages 427-437, 1990

[35]. M. Nishioka, H. Satoh, and K. Sakuri. Design and analysis of fast provably secure public-key cryptosystems based on a modular squaring. In Proc. ICISC 2001, LNCS 2288, pages 81-102, 2001

[36]. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In Proc. 2001 International Workshop on Practice and Theory in Public Key Cryptography (PK.C 2001), 2001

[37]. C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In Advances in Cryptology-Crypto '91, pages 433-444, 1991

[38]. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2): 120-126, 1978

[39]. V. Shoup. Lower bounds for discrete logarithms and related problems. In Advances in Cryptology - Eurocrypt '97, pages 256-266, 1997

[40]. V. Shoup. OAEP reconsidered. In Advances in Cryptology-Crypto 2001, pages 239-259, 2001

[41]. V. Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001 <http://eprint.iacr.org>

[42]. S. Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS. In Advances in Cryptology-Eurocrypt 2002, 2002