TCVN TIÊU CHUẨN QUỐC GIA

TCVN 12853 : 2020

ISO/IEC 18031 : 2011

WITH AMENDMENT 1:2017

Xuất bản lần 1

CÔNG NGHỆ THÔNG TIN – CÁC KỸ THUẬT AN TOÀN – BỘ TẠO BIT NGẪU NHIÊN

Information technology – Security techniques – Random bit generation

HÀ NỘI – 2020

Mục Lục

Lời nói đầu

TCVN 12853 : 2020 hoàn toàn tương đương với ISO/IEC 18031:2011, sửa đổi 1:2011 và đính chính 1:2017.

TCVN 12853 : 2020 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.

TIÊU CHUẨN QUỐC GIA **TCVN 12853:2020**

Công nghệ thông tin – Các kỹ thuật an toàn – Bộ tạo bit ngẫu nhiên

Information technology – Security techniques – Random bit generation

1 Phạm vi áp dụng

Tiêu chuẩn này quy định một mô hình khái niệm cho bộ tạo bit ngẫu nhiên dùng trong mật mã cùng với các phần tử của mô hình này.

Tiêu chuẩn này

- quy định đặc tính của các phần tử chính được yêu cầu đối với một bộ tạo bit ngẫu nhiên bất định,
- quy định đặc tính của các phần tử chính được yêu cầu đối với một tạo bit ngẫu nhiên tất định,
- thiết lập các yêu cầu an toàn cho cả bộ sinh bit ngẫu nhiên bất định và tất định.

Trường hợp có yêu cầu phải tạo ra chuỗi các số ngẫu nhiên từ xâu bit ngẫu nhiên, phụ lục B sẽ trình bày hướng dẫn về cách thức thực hiện.

Các kỹ thuật kiểm tra thống kê đối với các bộ tạo bit ngẫu nhiên dùng để xác thực hoặc kiểm tra hợp lệ một cách độc lập và thiết kế chi tiết cho các bộ tạo đó nằm ngoài phạm vi của tiêu chuẩn này.

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).

TCVN 11495-2 (ISO/IEC 9797-2), Công nghệ thông tin - Các kỹ thuật an toàn - Mã xác thực thông điệp (MACs) - Phần 2: Cơ chế sử dụng một hàm băm chuyên dụng.

TCVN 12213 (ISO/IEC 10116), Công nghệ thông tin - Các kỹ thuật an toàn - Chế độ hoạt động cho mã khối n-bit

TCVN 11816-3 (ISO/IEC 10118-3), Công nghệ thông tin - Các kỹ thuật an toàn - Hàm băm - Phần 3: Hàm băm chuyên dụng.

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

TCVN 11295 (ISO/IEC 19790), Công nghệ thông tin - Các kỹ thuật an toàn - Các yêu cầu an toàn cho mô đun mật mã

ISO/IEC 18032, Information technology — Security techniques — Prime number generation

3 Thuật ngữ và định nghĩa

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

3.1

Thuật toán (algorithm)

Quá trình tính toán được quy định rõ ràng để tính toán một bộ quy tắc, trả về một kết quả nhất định nếu tuân thủ đúng.

3.2

An toàn cho trước đây (backward secrecy)

Đảm bảo rằng các giá trị trước đó không thể xác định được từ các thông tin của giá trị hiện tại hoặc các giá trị tiếp theo.

3.3

Nguồn phân bố lệch (biased source)

Nguồn các xâu bit (hoặc số) lấy từ một không gian mẫu sao cho một số xâu bit (hoặc số) được chọn nhiều hơn so với một số xâu bit (hoặc số) khác.

CHÚ THÍCH 1 Tương tự, nếu không gian mẫu gồm r phần tử, thì một số phần tử sẽ xuất hiện với xác suất khác $1/r$.

CHÚ THÍCH 2 Thuật ngữ này hoàn toàn trái ngược với nguồn không chệch (3.35).

3.4

Dòng bit (bit stream)

Đầu ra các bit liên tục từ một thiết bị hoặc một cơ chế.

3.5

Xâu bit (bit string)

Chuỗi hữu hạn các số 0 và 1.

3.6

Hộp đen (black box)

Cơ chế lý tưởng chấp nhận các giá trị đầu vào và tạo ra các giá trị đầu ra, nhưng được thiết kế sao cho người quan sát không thể thấy bên trong chiếc hộp hoặc xác định chính xác những gì đang xảy ra bên trong chiếc hộp đó.

CHÚ THÍCH Thuật ngữ này hoàn toàn trái ngược với hộp trắng (3.14).

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.

[TCVN 11367-1]

3.8

Ranh giới mật mã (cryptographic boundary)

Đường bao khép kín liên tục được xác định ở dạng hiển, thiết lập các ranh giới lôgic và/hoặc vật lý của mô-đun mật mã và chứa tất cả các thành phần phần cứng, phần mềm và/hoặc phần sụn của mô-đun mật mã.

[TCVN 11295]

3.9

Thuật toán tất định (deterministic algorithm)

Đặc trưng của một thuật toán sao cho cùng một đầu vào cho trước luôn tạo ra một đầu ra giống nhau.

3.10

Bộ tạo bit ngẫu nhiên tất định (deterministic random bit generator)

DRBG

Bộ tạo bit ngẫu nhiên tạo ra một chuỗi bit xuất hiện ngẫu nhiên bằng cách sử dụng một thuật toán tất định từ một giá trị khởi tạo ngẫu nhiên thích hợp được gọi là mầm và có thể thêm một số đầu vào thứ cấp mà không ảnh hưởng đến độ an toàn của bộ tạo bit ngẫu nhiên.

CHÚ THÍCH Đặc biệt, các nguồn bất định cũng có thể là một phần của các đầu vào thứ cấp này.

3.11

Entropy (entropy)

Độ đo sự hỗn loạn, tính ngẫu nhiên hoặc biến đổi trong một hệ thống đóng.

CHÚ THÍCH Độ bất định của một biến ngẫu nhiên X là một đại lượng toán học chỉ lượng thông tin thu được khi quan sát biến X.

3.12

Nguồn bất định (entropy source)

Thành phần, thiết bị hoặc sự kiện tạo ra đầu ra theo một số cách nhất định nào đó, tạo ra một xâu bit chứa entropy.

3.13

An toàn cho sau đó (forward secrecy)

Đảm bảo rằng thông tin của các giá trị tiếp theo (tương lai) không thể xác định được từ các giá trị hiện tại hoặc các giá trị trước đó.

3.14

Hộp trắng (glass box)

Cơ chế lý tưởng chấp nhận các giá trị đầu vào và tạo ra các giá trị đầu ra và được thiết kế sao cho người quan sát có thể thấy bên trong và xác định chính xác những gì đang xảy ra.

CHÚ THÍCH Thuật ngữ này hoàn toàn trái ngược với hộp đen (3.6).

3.15

Hàm băm (hash-function)

Hàm ánh xạ các chuỗi bit thành các chuỗi bit có độ dài cố định, thỏa mãn hai đặc tính sau đây:

- Không thể tính toán để tìm ra đầu vào tương ứng với một đầu ra cho trước.
- Không thể tính toán để tìm ra đầu vào thứ hai có chung đầu ra với đầu vào đó.

CHÚ THÍCH Khả năng tính toán phụ thuộc vào các yêu cầu an toàn và môi trường cụ thể.

[TCVN 11816-1 (ISO/IEC 10118-1)]

3.16

Nguồn bất định con người (human entropy source)

Nguồn bất định có một số thành phần ngẫu nhiên có vai trò của con người.

3.17

Bộ tạo bit ngẫu nhiên tất định lai ghép (hybrid *D*RBG)

Bộ tạo bit ngẫu nhiên tất định sử dụng nguồn bất định không xác định như là một nguồn bất định bổ sung.

3.18

Bộ tạo bit ngẫu nhiên bất định lai ghép (hybrid NRBG)

Bộ tạo bit ngẫu nhiên bất định lấy một giá trị mầm làm một nguồn bất định bổ sung.

CHÚ THÍCH Bộ tạo bit ngẫu nhiên bất định lai có thể là vật lý hoặc phi vật lý.

3.19

Giá trị khởi tạo (initialisation value)

Giá trị được sử dụng để xác định điểm bắt đầu của thuật toán mật mã.

CHÚ THÍCH Giá trị khởi tạo được sử dụng trong hàm băm hoặc một thuật toán mã hóa là một ví dụ.

3.20

Hộp Kerckhoffs (Kerckhoffs box)

Hệ mật lý tưởng mà kẻ tấn công biết được thiết kế và khóa công khai nhưng không biết khóa bí mật và/hoặc thông tin bí mật khác.

CHÚ THÍCH Đối với kẻ tấn công, hộp Kerckhoffs là khái niệm lai giữa hộp đen và hộp trắng.

3.21

Kiểm tra với câu trả lời đã biết (known-answer test)

Phương pháp kiểm tra một cơ chế tất định xử lý một đầu vào nhất định và kết quả đầu ra được so sánh với một giá trị tương ứng đã biết.

CHÚ THÍCH Kiểm tra với câu trả lời đã biết của một cơ chế tất định cũng có thể bao gồm kiểm tra tính toàn vẹn của phần mềm thực thi cơ chế tất định. Ví dụ: nếu phần mềm thực thi cơ chế tất định là chữ ký số, thì chữ ký có thể được tính lại và so sánh với giá trị chữ ký đã biết.

3.22

Độ bất định nhỏ nhất (min-entropy)

Giới hạn dưới của độ bất định có hiệu quả trong việc xác định giá trị ước lượng trong trường hợp xấu nhất của độ bất định lấy mẫu.

CHÚ THÍCH Xâu bit X (hoặc chính xác hơn là biến ngẫu nhiên tương ứng mô phỏng các chuỗi bit ngẫu nhiên loại này) có độ bất định nhỏ nhất là k nếu k là giá trị lớn nhất sao cho $\Pr[X = x] \leq 2^{-k}$. Tức là X bao gồm k bit của độ bất định nhỏ nhất hoặc của tính ngẫu nhiên

3.23

Bộ tạo bit ngẫu nhiên bất định (non-deterministic random bit generator)

NRBG

Bộ sinh bit ngẫu nhiên có độ an toàn phụ thuộc vào việc lấy mẫu một nguồn entropy.

CHÚ THÍCH Nguồn bất định sẽ được lấy mẫu bất cứ khi nào bộ tạo bit ngẫu nhiên tạo đầu ra và có thể với tần suất lớn hơn.

3.24

Hàm một chiều (one-way function)

Hàm có tính chất dễ dàng tính được đầu ra đối với đầu vào cho trước nhưng lại không thể tìm được đầu vào tương ứng nếu cho trước đầu ra.

[ISO/IEC 11770-3]

3.25

Hàm tạo đầu ra (output generation function)

Hàm trong một bộ tạo bit ngẫu nhiên tính toán đầu ra của bộ tạo bit ngẫu nhiên từ trạng thái bên trong của bộ tạo này.

3.26

Chuỗi bit giả ngẫu nhiên (pseudorandom sequence of bits)

Chuỗi bit hoặc một số được chọn một cách ngẫu nhiên ngay cả khi quá trình lựa chọn được thực hiện bởi một thuật toán tất định.

3.27

Bộ tạo bit ngẫu nhiên tất định thuần túy (pure DRBG)

Bộ tạo bit ngẫu nhiên tất định trong đó nguồn bất định chính là mầm.

3.28

Bộ tạo bit ngẫu nhiên bất định thuần túy (pure NRBG)

Bộ tạo bit ngẫu nhiên bất định trong đó nguồn bất định là không xác định.

3.29

Bộ tạo bit ngẫu nhiên (random bit generator)

RBG

Thiết bị hoặc thuật toán có đầu ra là một chuỗi bit xuất hiện độc lập và đồng xác suất.

3.30

Thay mầm mới (reseeding)

Chức năng chuyển đổi trạng thái bên trong chuyên biệt bằng cách cập nhật trạng thái bên trong khi một giá trị mầm mới được cung cấp.

CHÚ THÍCH Việc sử dụng thuật ngữ "Thay mầm mới" không chỉ có trong tài liệu. Một số tác giả cũng sử dụng "Thay mầm mới" để chỉ cơ chế thay thế giá trị hiện thời của trạng thái bên trong bằng một giá trị mới. Tiêu chuẩn này tuân theo định nghĩa này. Tuy nhiên, thường thì người ta phân biệt giữa "Thay mầm mới" và "Cập nhật mầm". Thuật ngữ "Thay mầm mới" chỉ bao gồm các cơ chế thay thế trạng thái bên trong bằng một giá trị mới mà không phụ thuộc vào giá trị hiện thời (đa phần là một quá trình tạo mầm mới). Ngược lại, "Cập nhật mầm" dùng để chỉ một cơ chế tính toán trạng thái bên trong mới từ giá trị hiện thời và các dữ liệu khác (thường là bất định) (Tham khảo 9.6, mục 3).

3.31

Tham số bí mật (secret parameter)

Đầu vào của một bộ tạo bit ngẫu nhiên trong quá trình khởi tạo, cung cấp entropy bổ sung trong trường hợp nguồn entropy bị lỗi hoặc bị làm tổn hại.

3.32

Mầm (seed)

Chuỗi bit được sử dụng làm đầu vào của một bộ tạo bit ngẫu nhiên tất định.

CHÚ THÍCH Mầm sẽ xác định một phần trạng thái của bộ tạo bit ngẫu nhiên tất định.

3.33

Vòng đời của mầm (seedlife)

Khoảng thời gian từ khi khởi tạo bộ tạo bit ngẫu nhiên từ một mầm đến khi thay mầm mới (khởi tạo đầy đủ) cung cấp cho bộ tạo bit ngẫu nhiên một giá trị mầm khác.

3.34

Trạng thái (state)

Điều kiện của một bộ tạo bit ngẫu nhiên hoặc thành phần bất kỳ của bộ tạo ứng với thời gian và tình huống cụ thể.

3.35

Nguồn phân bố đều (unbiased source)

Nguồn các xâu bit (hoặc số) lấy từ một không gian mẫu sao cho tất cả các xâu bit (hoặc số) được chọn với cùng một xác suất.

CHÚ THÍCH 1 Tương tự, nếu không gian mẫu gồm r phần tử, thì tất cả các phần tử sẽ xuất hiện với xác suất $1/r$.

CHÚ THÍCH 2 Thuật ngữ này hoàn toàn trái ngược với nguồn phân bố lệch (3.3).

4 Ký hiệu

Trong tiêu chuẩn này áp dụng các ký hiệu dưới đây:

Ký hiệu **Ý nghĩa**

$Pr[x]$	Xác suất xuất hiện của x.
IV	Giá trị khởi tạo.
$\lceil X \rceil$	Làm tròn lên: số nguyên nhỏ nhất lớn hơn hoặc bằng X. Ví dụ: $\lceil 5 \rceil = 5$, $\lceil 5{,}3 \rceil = 6$.
$X \oplus Y$	Phép cộng bit loại trừ (hay được gọi là cộng bit theo mod 2) của hai xâu bit X và Y có cùng độ dài.
$X \parallel Y$	Phép nối hai xâu bit X và Y theo thứ tự.
$\lvert a \rvert$	Độ dài bit của xâu a.
$x \bmod n$	Số dư duy nhất $r, 0 \leq r \leq n-1$ khi số nguyên x chia cho n. Ví dụ: $23 \bmod 7 = 2$.
	Được sử dụng trong hình để minh họa bộ "chuyển đổi" giữa các nguồn đầu vào.

5 Đặc điểm và yêu cầu đối với bộ tạo bit ngẫu nhiên

5.1 Đặc điểm của bộ tạo bit ngẫu nhiên

Các tính chất ngẫu nhiên có thể được mô phỏng bằng cách tung một đồng xu lên không khí và quan sát mặt hướng lên trên khi nó rơi xuống đất, trong đó một mặt được gọi là "mặt ngửa" (H) và mặt còn lại được gọi là "mặt sấp" (T). Đồng xu cũng có gờ, tuy nhiên xác suất để gờ đồng xu chạm đất hiếm khi xảy ra nên có thể bỏ qua trường hợp này.

Tung đồng xu nhiều lần tạo ra một chuỗi có thứ tự kết quả của việc tung đồng xu ký hiệu là một chuỗi H và T. Ví dụ: chuỗi "HTTHT" (đọc từ trái sang phải) nghĩa là mặt ngửa rồi đến mặt sấp, tiếp theo là mặt sấp, mặt ngửa rồi đến mặt sấp. Chuỗi kết quả tung đồng xu này có thể chuyển đổi thành một chuỗi nhị phân bằng cách gán H bằng số một nhị phân ('1') và T bằng số không nhị phân ('0'); chuỗi nhị phân kết quả là '10010'.

Các tính chất yêu cầu của tính ngẫu nhiên có thể được kiểm tra bằng cách sử dụng ví dụ tung đồng xu được mô tả ở trên. Kết quả của mỗi lần tung đồng xu là:

1. Không thể đoán trước được: Trước khi tung, không thể biết được đồng xu rơi xuống đất bằng mặt ngửa hay mặt sấp. Ngoài ra, nếu kết quả tung được giữ bí mật, thì không thể xác định được kết quả tung đồng xu là gì nếu biết một kết quả tung đồng xu sau đó bất kỳ. Tính không thể đoán trước được phụ thuộc vào việc người quan sát có thể quan sát được kết quả của việc tung đồng xu hay không. Khái niệm độ bất định định lượng số lượng kết quả không thể đoán trước được hoặc không chắc chắn liên quan đến người quan sát và sẽ được thảo luận kỹ hơn ở phần sau của tiêu chuẩn này.
2. Không chệch: mỗi kết quả thu được có cùng khả năng xảy ra;
3. Độc lập: quá trình tung đồng xu được cho là quá trình không nhớ; bất cứ điều gì xảy ra trước khi tung đồng xu hiện tại không ảnh hưởng đến nó.

Vì vậy, chuỗi kết quả của việc tung đồng xu có thể trực tiếp áp dụng cho một bộ tạo bit ngẫu nhiên. Các bộ tạo bit ngẫu nhiên được quy định trong tiêu chuẩn này mô phỏng chuỗi kết quả của việc tung đồng xu lý tưởng.

Như đã nêu ở trên, tính không thể đoán trước là thuộc tính bắt buộc của một bộ tạo bit ngẫu nhiên. Do đó, không thể dự đoán đầu ra của một bộ tạo bit ngẫu nhiên được thực thi và làm việc đúng cách. Tính an toàn về phía trước liên quan đến khả năng dự đoán đầu ra tiếp theo của bộ tạo bit ngẫu nhiên dựa trên thông tin về các giá trị đầu ra trước đó và/hoặc các trạng thái bên trong. Việc không xác định được đầu ra trước đó của bộ tạo bit ngẫu nhiên khi biết thông tin về đầu ra hiện tại hoặc tương lai của bộ tạo bit được gọi là tính an toàn về phía sau.

Quyết định kết hợp tính an toàn về phía trước với an toàn về phía sau phụ thuộc vào yêu cầu của ứng dụng.

Các yếu tố sau đây cần được xem xét khi quyết định kết hợp tính an toàn về phía trước với an toàn về phía sau:

1. Trong một số trường hợp, việc đạt được tính an toàn về phía sau quan trọng hơn việc đạt được tính bí mật về phía trước. Ví dụ: nếu một hệ mật bị lộ, kẻ tấn công có thể đọc được các thông điệp cũ được xử lý bằng hệ mật đó. Tính an toàn về phía trước không được quan tâm vì hệ mật không còn được sử dụng bởi người sở hữu đầu. Việc đạt được tính an toàn về phía sau là đơn giản (ví dụ: bằng cách sử dụng hàm một chiều trong thiết kế) mặc dù có thể ảnh hưởng đến hiệu suất do thực hiện tính chất này và phụ thuộc vào thiết kế.
2. Việc đạt được tính an toàn về phía trước có thể không thích hợp đối với một số hệ mật. Ví dụ: thẻ thông minh được khởi tạo tại nơi sản xuất với giá trị mầm có độ bất định đầy đủ và sẽ hết hạn sau một khoảng thời gian nhất định (ví dụ: hai hoặc ba năm). Trong trường hợp này, có thể dễ dàng thay thế thẻ cũ bằng một thẻ mới có giá trị mầm khác thay cho việc phải đảm bảo tính an toàn về phía trước trong thiết kế bộ tạo bit ngẫu nhiên.
3. Trong một số trường hợp, việc đạt được tính an toàn về phía trước có thể quan trọng hơn việc đạt được tính an toàn về phía sau. Ví dụ: việc tạo giá trị nonce an toàn. Thuật toán tạo bit ngẫu nhiên không cần đảm bảo tính an toàn về phía sau vì tất cả các giá trị đầu ra trước đó sẽ được biết đến. Tuy nhiên, tính an toàn về phía trước cần phải đảm bảo để ngăn chặn kẻ tấn công với những thông tin đã biết về bộ tạo có thể dự đoán được các giá trị đầu ra sau này.

5.2 Yêu cầu đối với bộ tạo bit ngẫu nhiên

Các yêu cầu sau đây được áp dụng cho tất cả các bộ tạo bit ngẫu nhiên cả tất định và bất định.

Đây là các yêu cầu cơ bản nhằm đảm bảo an toàn cho các cơ chế mật mã yêu cầu đầu vào ngẫu nhiên.

Ngưỡng giữa tính khả thi và không khả thi sẽ được xác định theo yêu cầu chung đối với mức an toàn mật mã tối thiểu chấp nhận được của ứng dụng.

1. Căn cứ vào một số giả thiết nhất định, không thể phân biệt đầu ra của một bộ tạo bit ngẫu nhiên từ các bit ngẫu nhiên thực có phân bố đều. Thông thường, tất cả các đầu ra có thể xảy ra với xác suất bằng nhau và một chuỗi kết quả đầu ra xuất hiện với phân bố đều.
2. Cho trước một chuỗi các bit đầu ra, không thể tính toán hoặc dự đoán bất kỳ bit đầu ra nào đã xuất hiện hoặc sẽ xuất hiện.
3. Trong suốt thời gian có hiệu lực của bộ tạo bit ngẫu nhiên, không thể dự đoán sự lặp lại của chuỗi đầu ra.
4. Bộ tạo bit ngẫu nhiên sẽ không làm rò rỉ thông tin bí mật (ví dụ: trạng thái bên trong của một bộ tạo bit ngẫu nhiên) thông qua đầu ra của nó.
5. Bộ tạo bit ngẫu nhiên sẽ không làm rò rỉ thông tin bí mật cho kẻ tấn công.

CHÚ THÍCH 1 Tấn công thời gian là một ví dụ cho việc không thỏa mãn yêu cầu 5.

6. Bộ tạo bit ngẫu nhiên sẽ không tạo ra các bit nếu bộ tạo không được đánh giá là có độ bất định đầy đủ. Tiêu chuẩn về tính đầy đủ của độ bất định quan trọng hơn các yêu cầu của tiêu chuẩn này và yêu cầu của ứng dụng.
7. Khi phát hiện có lỗi, bộ tạo bit ngẫu nhiên phải (a) chuyển sang trạng thái lỗi vĩnh viễn hoặc (b) có thể khôi phục từ một độ bất định tổn thất hoặc thỏa hiệp nếu trạng thái lỗi vĩnh viễn không được chấp nhận trong các yêu cầu của ứng dụng. Các yêu cầu này được thỏa mãn trong thiết kế của bộ tạo.

CHÚ THÍCH 2 Xem mục 8.8 và 9.8 để có thông tin khi bị lỗi và các phép kiểm thử của bộ tạo bit ngẫu nhiên tất định và bất định.

8. Thiết kế và thực thi một bộ tạo bit ngẫu nhiên phải có một ranh giới bảo vệ xác định. Ranh giới bảo vệ được quy định trong TCVN 11295 (ISO/IEC 19790) (xem phụ lục I).
9. Xác suất để bộ tạo bit ngẫu nhiên có đầu ra không đáp ứng yêu cầu (ví dụ: đầu ra cố định hoặc có chu kỳ nhỏ tức là lặp lại cùng một đầu ra) phải đủ nhỏ. Điều đó có nghĩa là xác suất xảy ra lỗi phải phù hợp với yêu cầu tổng thể về hoạt động chính xác của bộ tạo bit ngẫu nhiên, tuy nhiên không đồng nhất với yêu cầu về độ an toàn mật mã.
10. Thiết kế của bộ tạo bit ngẫu nhiên phải bao gồm các phương pháp để ngăn chặn ảnh hưởng, thao tác có thể dự đoán được hoặc dự đoán đầu ra của bộ tạo bằng cách quan sát các đặc tính vật lý của nó (ví dụ: mức tiêu hao năng lượng, thời gian hoặc sự phát xạ).
11. Quá trình thực thi phải được thiết kế cho phép xác nhận tính hợp lệ, bao gồm những khẳng định thiết kế cụ thể về việc bộ tạo không thực hiện. Việc kiểm tra tính hợp lệ của một bộ tạo bit bất định có nghĩa là xem xét bộ tạo có hoạt động như mong đợi hay không, không chỉ trong điều kiện hoạt động bình thường mà còn ở các ranh giới của điều kiện hoạt động dự kiến. Các đoạn mã liên quan đến an toàn trong mã nguồn giúp điều chỉnh hành vi trong điều kiện đặc biệt (ví dụ: khởi tạo, kiểm thử không đạt,...) phải được xác nhận bằng cách thử tất cả các điều kiện lỗi xảy ra trong quá trình kiểm tra.
12. Phải có căn cứ thiết kế (lý thuyết, thực nghiệm hoặc cả hai) để hỗ trợ tất cả các yêu cầu an toàn cho bộ tạo bit ngẫu nhiên bao gồm việc bảo vệ trước các hành vi sai.

5.3 Yêu cầu tùy chọn đối với bộ tạo bit ngẫu nhiên

Các yêu cầu tùy chọn cho một bộ tạo bit ngẫu nhiên như sau:

1. Nếu bộ tạo bit ngẫu nhiên có khả năng hoạt động ở nhiều chế độ, thì nó phải trả về thông tin của chế độ đang hoạt động theo yêu cầu.
2. Ứng dụng có thể yêu cầu bộ tạo bit ngẫu nhiên chạy ở chế độ kiểm thử, ví dụ: khi sử dụng giá trị mầm không bí mật, do đó có khả năng tái tạo lại các bit mà nó sinh ra. Khi một bộ tạo bit ngẫu nhiên chạy ở chế độ kiểm thử, nó không được sử dụng để tạo ra các bit bí mật. Khi sử dụng một giá trị mầm bí mật, bộ tạo sẽ không hoạt động ở chế độ kiểm thử.
3. Được coi như một hộp trắng, bộ tạo bit ngẫu nhiên phải đảm bảo tính an toàn về phía sau. Nếu thuộc tính này được hỗ trợ thì nghĩa là cho trước tất cả các thông tin có thể tiếp cận được về bộ tạo bit ngẫu nhiên (bao gồm một số tập hợp các giá trị đầu vào, thuật toán và đầu ra), không thể tính toán hoặc dự đoán (tính theo độ an toàn quy định) bất kỳ bit đầu ra nào trước đó.
4. Được coi như một hộp trắng, bộ tạo bit ngẫu nhiên phải đảm bảo tính an toàn về phía trước. Nếu hỗ trợ, thì nghĩa là cho trước tất cả các thông tin có thể tiếp cận được về bộ tạo bit ngẫu nhiên (bao gồm một số tập hợp các giá trị đầu vào, thuật toán và đầu ra), không thể tính toán hoặc dự đoán (tính theo độ an toàn quy định) bất kỳ bit đầu ra sắp tới nào tính từ thời điểm yêu cầu tính an toàn về phía trước.

6 Mô hình bộ tạo bit ngẫu nhiên

6.1 Mô hình chức năng, khái niệm để tạo bit ngẫu nhiên

Hình 1 mô tả mô hình chức năng, khái niệm cho quá trình tạo bit ngẫu nhiên. Mô hình này và các yêu cầu liên quan cũng như mục tiêu mà bộ tạo bit ngẫu nhiên phải đạt được không có ràng buộc hay bắt buộc thực hiện cho dù là bộ tạo ngẫu nhiên bất định hay tất định. Vì không phải tất cả các khía cạnh quan trọng của việc tạo bit ngẫu nhiên có thể xác định theo thuật toán, do đó quan điểm chức năng của việc tạo bit ngẫu nhiên là trung tâm của định nghĩa về các bộ tạo bit ngẫu nhiên trong tiêu chuẩn này.

Mô hình chức năng bao gồm mọi thứ cần thiết để tạo các bit ngẫu nhiên. Cách tiếp cận toàn diện này là cần thiết để đảm bảo rằng đầu ra của bộ tạo bit ngẫu nhiên là ngẫu nhiên như mong muốn.

Khi một bộ tạo bit ngẫu nhiên hoặc một thành phần của hệ thống không hết hợp trực tiếp tất cả các thành phần chức năng hoặc giải quyết tất cả các yêu cầu chức năng được quy định ở đây, thì bộ tạo bit ngẫu nhiên đó có thể vẫn phù hợp với tiêu chuẩn này nếu thành phần của nó được sử dụng trong một hệ thống cung cấp các yếu tố còn thiếu và đáp ứng các yêu cầu còn lại. Một bộ tạo bit ngẫu nhiên như vậy sẽ có các yêu cầu chức năng bổ sung. Những yêu cầu đó trở thành một điều kiện tiên quyết cho việc sử dụng bộ tạo bit ngẫu nhiên.

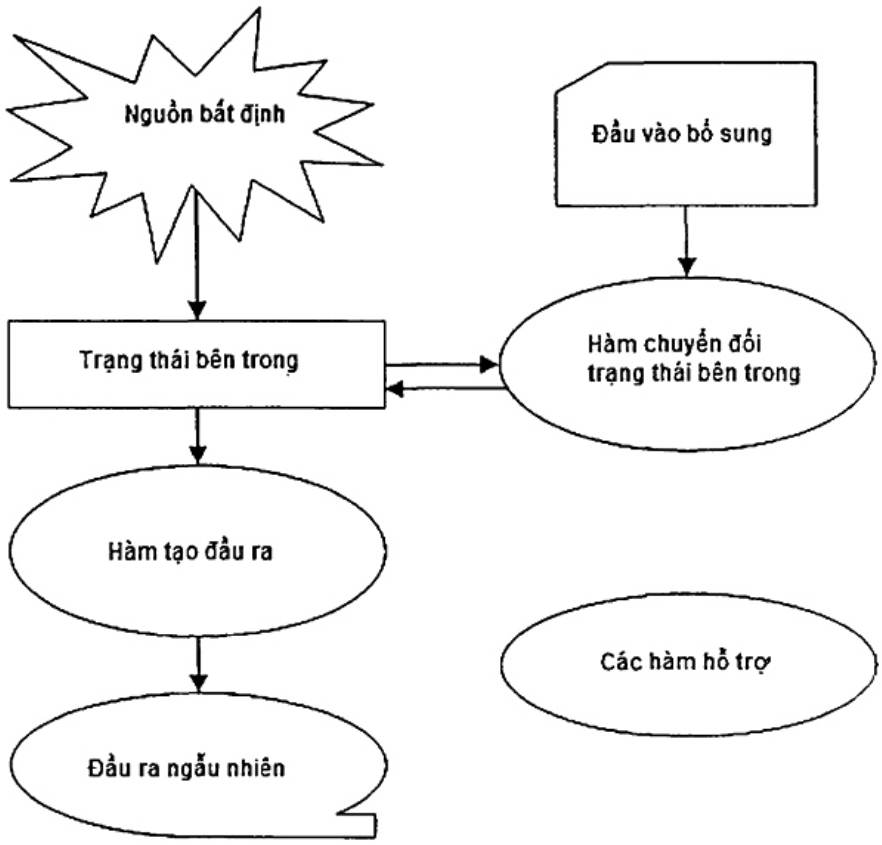
CHỦ THÍCH Có những bộ tạo bit ngẫu nhiên không kết hợp tất cả các thành phần chức năng như được mô tả trong hình 1. Ví dụ: Các bộ tạo bit ngẫu nhiên bất định vật lý không nhất thiết cần thêm đầu vào bổ sung.

6.2 Các thành phần cơ bản của bộ tạo bit ngẫu nhiên

6.2.1 Giới thiệu về các thành phần cơ bản của bộ tạo bit ngẫu nhiên

Mô hình được mô tả trong hình 1 có 6 thành phần cơ bản, tuy nhiên đây là trường hợp đáp ứng tất cả các yêu cầu chức năng mà không cần kết hợp tất cả các thành phần cơ bản. Đó là:

a) Nguồn bất định;
b) Đầu vào bổ sung;
c) Trạng thái bên trong;
d) Hàm chuyển đổi trạng thái bên trong;
e) Hàm tạo đầu ra; và
f) Các hàm hỗ trợ.



Hình 1: Mô hình chức năng của bộ tạo bit ngẫu nhiên

6.2.2 Nguồn bất định

6.2.2.1 Giới thiệu về nguồn bất định

Nguồn bất định là nguồn các bit không thể dự đoán trước được. Các bit này có thể bị lệch và phục thuộc vào nhau ở một mức độ nào đó; trên thực tế điều này thường xảy ra. Đối với bộ tạo bit ngẫu nhiên tất định, nguồn bất định có thể là một bộ tạo bit ngẫu nhiên bất định riêng biệt, từ xa. Tuy nhiên, đối với một bộ tạo bit ngẫu nhiên bất định, thành phần nguồn bất định là kết hợp của sự tồn tại của hoạt động tạo ra độ bất định, sự phát hiện của hoạt động này và cơ chế số hóa.

Nguồn bất định tạo ra các bit với độ bất định khác 0. Đây là thành phần duy nhất trong mô hình tạo ra độ bất định. Nguồn bất định bao gồm tất cả mọi thứ không xác định trong mô hình bộ tạo bit ngẫu nhiên, cùng với mọi yêu cầu cho nguồn tự biểu thị theo các bit (trái ngược với các tín hiệu tương tự hoặc hoạt động chưa số hóa khác). Trong một bộ tạo bit ngẫu nhiên tất định, nguồn bất định là không xác định mặc dù tính chất không thể dự đoán được của bộ tạo phụ thuộc hoàn toàn vào độ bất định của một đầu vào đặc biệt được gọi là mầm. Mô hình không giả định bất cứ điều gì về khả năng dự đoán, độ chệch hoặc tính độc lập của các bit được tạo ra bởi nguồn, khác biệt só với nguồn bất định có độ bất định khác 0. Tuy nhiên, mọi thiết kế bộ tạo bit ngẫu nhiên cụ thể sẽ cần thiết lập các điều kiện xác định nhằm đảm bảo thiết kế sẽ hoạt động.

Mô hình cho phép nguồn bất định bị phá hủy theo những cách khác nhau. Một cách đó là phá hủy đệ quy. Đặc biệt, nguồn bất định cho một bộ tạo bit ngẫu nhiên có thể là một bộ tạo bit ngẫu nhiên (bên ngoài) riêng biệt khác. Thật vậy, bộ tạo bit ngẫu nhiên ngoài có thể sử dụng một bộ tạo bit ngẫu nhiên khác như một nguồn bất định. Không có giới hạn nào cho việc xây dựng đệ quy, tuy nhiên quá trình đệ quy như vậy phải kết thúc bằng một nguồn bất định (không xác định) thật sự, nếu không toàn bộ cấu

trúc bị hủy bỏ. Ngoài ra, việc xây dựng phải đảm bảo độ bất định của hệ thống đầy đủ để hỗ trợ các tiêu chí đầu vào cho từng bộ tạo bit ngẫu nhiên riêng biệt.

6.2.2.2 Các yêu cầu đối với nguồn bất định

Các yêu cầu đối với nguồn bất định của một bộ tạo bit ngẫu nhiên như sau:

1. Nguồn bất định phải dựa vào các nguyên tắc được thiết lập tốt hoặc các hành vi đặc trưng phổ biến.
2. Tỷ lệ độ bất định phải được đánh giá hoặc phải tự điều chỉnh quá trình thu thập sao cho lượng độ bất định trên một đơn vị hoặc sự kiện thu thập phải đạt được hoặc vượt quá giới hạn dưới được thiết kế.
3. Nguồn bất định phải được thiết kế sao cho bất cứ thao tác nào (như khả năng kiểm soát nguồn bất định), ảnh hưởng (như khả năng tạo độ chệch cho nguồn bất định) hoặc quá trình quan sát bởi một số thực thể bên ngoài trái phép phải được giảm thiểu tối đa theo các yêu cầu của ứng dụng.
4. Phải phát hiện được việc mất mát hoặc suy giảm nghiêm trọng nguồn bất định.

6.2.3 Đầu vào bổ sung

6.2.3.1 Giới thiệu về đầu vào bổ sung

Đầu vào bổ sung có thể được sử dụng để cá nhân hóa đầu ra của bộ tạo bit ngẫu nhiên dựa trên các thông số như thời gian/ngày tháng, sử dụng dữ liệu và thống tin được người dùng khác cung cấp, và/hoặc dữ liệu cần thiết để kiểm soát chức năng bên trong của bộ tạo bit ngẫu nhiên. Các đầu vào này thường bao gồm các lệnh và/hoặc các tham số biến thiên theo thời gian như xung hoặc bộ đếm bên trong. Độ bất định của đầu ra phải không phụ thuộc vào bất cứ đặc tính nào của đầu vào bổ sung.

6.2.3.2 Các yêu cầu đối với đầu vào bổ sung

Hình thức và việc sử dụng đầu vào bổ sung phải không làm giảm độ bất định của bộ tạo bit ngẫu nhiên.

6.2.4 Trạng thái bên trong

6.2.4.1 Giới thiệu về trạng thái bên trong

Trạng thái bên trong chứa bộ nhớ của bộ tạo bit ngẫu nhiên. Có thể bao gồm mầm, bộ đếm, giá trị xung, đầu vào người dùng, độ an toàn của đầu ra... Trạng thái bên trong bao gồm tất cả các tham số, biến và các giá trị lưu trữ khác được phần tất định của bộ tạo bit ngẫu nhiên sử dụng hoặc hoạt động. Trạng thái bên trong thường chứa trong đặc tả về các hàm tất định; mô hình làm cho trạng thái trở nên rõ ràng để giải quyết tốt hơn vấn đề an toàn mà nó có thể ảnh hưởng.

Về mặt chức năng, trạng thái bên trong đóng hai vai trò khác nhau. Đầu tiên là để đảm bảo rằng đầu ra xuất hiện ngẫu nhiên ngay cả khi độ bất định của đầu vào không đủ để tạo ra một chuỗi với độ dài nhất định thực sự là ngẫu nhiên. Vì hàm chuyển đổi trạng thái bên trong và hàm tạo đầu ra là các hàm tất định và do các hàm tất định có tính chất là cho cùng một giá trị đầu ra với cùng một giá trị đầu vào, nên cần phải có một biến trạng thái được xây dựng để thay đổi từng khối đầu ra của bộ tạo bit ngẫu nhiên. Mục đích của biến này là đảm bảo rằng đầu ra xuất hiện ngẫu nhiên ngay cả khi không có hoặc có ít độ bất định thực sự được thêm vào hệ thống.

Điều này đặc biệt phù hợp với một bộ tạo bit ngẫu nhiên tất định. Đối với bộ tạo bit ngẫu nhiên tất định, nếu mầm hoặc tham số bí mật không được cập nhật bởi một thiết bị bên ngoài, thì có thể tạo ra một chuỗi lặp lại của trạng thái bên trong. Vì vậy, mầm (hoặc tham số bí mật) phải được cập nhất định kỳ hoặc thiết bị phải không hoạt động được sau một khoảng thời gian xác định trước.

Phần trạng thái bên trong hỗ trợ chức năng này được gọi là trạng thái làm việc. Phải đảm bảo rằng giá trị của trạng thái làm việc không bị ảnh hưởng bởi bất kỳ sự kiện bên ngoài nào (ví dụ: bằng cách khởi động lại bộ tạo bit ngẫu nhiên).

Vai trò thứ hai của trạng thái bên trong là tham số hóa các hàm chuyển đổi trạng thái bên trong tất định và/hoặc hàm tạo đầu ra mà không biết đến tham số này, hàm tất định thể hiện các thuộc tính cần thiết của nó. Ví dụ, nó có thể tham số hóa hàm tạo đầu ra theo cách không thể suy ra trạng thái bên trong từ đầu ra khi không biết tham số bí mật. Tham số bí mật (đôi khi được gọi là khóa của bộ tạo bit ngẫu nhiên) thường là một hoặc nhiều khóa mật mã và được tạo ra một cách ngẫu nhiên. Khi tồn tại một tham số bí mật như vậy, nó sẽ có thời hạn an toàn được xác định rõ ràng bao gồm các quy định về việc tạo, phân phối, cập nhật và hủy.

CHÚ THÍCH 1 Tham số bí mật không bắt buộc cho tất cả các bộ tạo bit ngẫu nhiên.

CHÚ THÍCH 2 Vấn đề quản lý khóa được quy định trong ISO/IEC 11770-1 [6].

6.2.4.2 Các yêu cầu đối với trạng thái bên trong

Các yêu cầu đối với trạng thái bên trong của một bộ tạo bit ngẫu nhiên như sau:

1. Trạng thái bên trong phải được bảo vệ phù hợp với việc sử dụng và độ nhạy của đầu ra.
2. Trạng thái bên trong phải duy trì chức năng một cách hợp lý khi có sự cố về nguồn điện, khởi động lại... hoặc quay về một điều kiện an toàn trước khi tạo ra đầu ra (nghĩa là phải đảm bảo tính toàn vẹn của trạng thái bên trong hoặc phải khởi tạo lại trạng thái bên trong).
3. Các thành phần trạng thái tích lũy hoặc mang độ bất định cho bộ tạo bit ngẫu nhiên phải có ít nhất x bit độ bất định trong đó x là độ dài tính bằng bit của độ an toàn mong muốn. Thành phần trạng thái phải tích lũy nhiều hơn số bit tối thiểu của độ bất định nhằm đảm bảo và giảm nguy cơ sử dụng các giá trị giống hệt nhau trong hai hệ mật khác nhau. Vì vậy, nên giữ trạng thái bên trong lớn hơn x, ví dụ: $x + 64$ thường cung cấp một giá trị biên đủ lớn, trong khi nếu kẻ tấn công có thể tính toán trước và có lượng bộ nhớ rất lớn, thì phải sử dụng kích cỡ $2x$ (xem ví dụ [10]).
4. Phần bí mật của trạng thái bên trong phải có một thời hạn xác định cụ thể sau khi bộ tạo bit ngẫu nhiên ngừng hoạt động hoặc được nâng cấp lại với độ bất định bổ sung đầy đủ. Các hoạt động sử dụng giá trị mầm cũ phải chấm dứt sau khi hết thời hạn quy định. Xem ISO/IEC 11770-1 để biết thêm thông tin chi tiết về vòng đời của khóa.
5. Không được sử dụng lại trạng thái bên trong, trừ trường hợp vô tình. Vì cùng một giá trị mầm giống nhau sẽ không là đầu vào cho một trường hợp khác của bộ tạo bit ngẫu nhiên tất định.

6.2.4.3 Yêu cầu tùy chọn đối với trạng thái bên trong

Trạng thái bên trong được sử dụng để tạo ra dữ liệu công khai (ví dụ: giá trị nonce và giá trị khởi tạo) phải hoàn toàn độc lập với trạng thái được sử dụng để tạo ra dữ liệu bí mật như khóa mật mã.

6.2.5 Hàm chuyển đổi trạng thái bên trong

6.2.5.1 Giới thiệu về hàm chuyển đổi trạng thái bên trong

Hàm chuyển đổi trạng thái bên trong làm thay đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên. Trong một bộ tạo bit ngẫu nhiên tất định, hàm chuyển đổi trạng thái bên trong bao gồm một hoặc nhiều thuật toán mật mã, chính là một phần của bộ tạo. Hàm chuyển đổi trạng thái bên trong bao gồm mọi thứ trong bộ tạo bit ngẫu nhiên có thể thiết lập hoặc thay đổi trạng thái bên trong. Theo quy ước, nguồn bất định hoặc bất kỳ đầu vào dữ liệu của bộ tạo bit ngẫu nhiên không được hoạt động trực tiếp trên trạng thái bên trong. Thay vào đó, nguồn bất định và các đầu vào dữ liệu khác của bộ tạo bit ngẫu nhiên là đối số cho các hàm hoạt động trên trạng thái bên trong.

Vì trạng thái bên trong có thể bao gồm một số thành phần riêng biệt, các hàm chuyển đổi trạng thái bên trong có thể bao gồm nhiều tập hàm khác nhau, phân biệt bởi thành phần trạng thái mà chúng tác động. Các hàm trong tập đó có thể đơn giản như hàm định danh hoặc phức tạp như một hàm mật mã. Đối với một thanh ghi xung hoặc thanh ghi bộ đếm, hàm sắp xếp thanh ghi như một xung hoặc bộ đếm là một thành phần của hàm chuyển đổi trạng thái bên trong. Vì đầu ra của các hàm này luôn điều khiển một thành phần của trạng thái bên trong, nên đầu vào có thể là nguồn bất định, một đầu vào bộ tạo bit ngẫu nhiên bên ngoài khác, các thành phần khác nhau của trạng thái bên trong hoặc bất kỳ sự kết hợp nào.

Hàm chuyển đổi trạng thái bên trong cũng phải cho phép thao tác an toàn (ví dụ: cập nhật) đối với tham số bí mật. Việc kết nối tới các hàm này phải được kiểm soát chặt chẽ.

Trong một bộ tạo bit ngẫu nhiên tất định, hàm chuyển đổi trạng thái bên trong có chức năng an toàn nhiều hơn cho bộ tạo, bao gồm cả lượng đầu ra mà bộ tạo bit ngẫu nhiên có thể tạo ra với một đầu vào nguồn bất định cho trước.

Trong một bộ tạo bit ngẫu nhiên bất định, hàm chuyển đổi trạng thái bên trong xác định cách thức quá trình thu thập độ bất định ảnh hưởng đến trạng thái bên trong của bộ tạo bit ngẫu nhiên và có thể hoạt động theo nhiều cách: các hàm này có thể là các thuật toán đơn giản, phi trạng thái hoặc có thể kết hợp các trạng thái trước đó với độ bất định mới thu thập được để tích lũy độ bất định.

6.2.5.2 Các yêu cầu đối với hàm chuyển đổi trạng thái bên trong

Các yêu cầu đối với hàm chuyển đổi trạng thái bên trong của một bộ tạo bit ngẫu nhiên như sau:

1. Hàm chuyển đổi trạng thái bên trong phải được kiểm tra qua kiểm thử đã biết câu trả lời.
2. Hàm chuyển đổi trạng thái bên trong phải phụ thuộc hoàn toàn vào độ bất định do trạng thái bên trong mang đến.
3. Hàm chuyển đổi trạng thái bên trong phải chống lại quan sát và phân tích thông qua việc tiêu thụ điện năng, thời gian, bức xạ hoặc các kênh kề phù hợp khác.
4. Không thể (cố ý hoặc vô ý) làm cho hàm chuyển đổi trạng thái bên trong quay về trạng thái trước đó khi đang hoạt động bình thường (không bao gồm kiểm thử và xác minh có ủy quyền đầu ra của bộ tạo bit ngẫu nhiên).

6.2.5.3 Yêu cầu tùy chọn đối với hàm chuyển đổi trạng thái bên trong

Hàm chuyển đổi trạng thái bên trong có thể cho phép bộ tạo bit ngẫu nhiên khôi phục lại từ sự thỏa hiệp về trạng thái bên trong (tức là cung cấp tính an toàn về phía trước) thông qua sự kết hợp độ bất định định kỳ.

6.2.6 Hàm tạo đầu ra

6.2.6.1 Giới thiệu về hàm tạo đầu ra

Hàm tạo đầu ra (Output Generation Function - OGF) hoạt động trên trạng thái bên trong để tạo ra các bit đầu ra theo yêu cầu của ứng dụng. Một yêu cầu gửi đến OGF chứa (ít nhất) số bit đầu ra yêu cầu và độ an toàn tối thiểu cần thiết cho các bit đó. Yêu cầu như vậy có thể làm cho trạng thái bên trong được cập nhật bởi hàm chuyển đổi trạng thái bên trong. OGF chấp nhận thành phần trạng thái làm việc của trạng thái bên trong như là đầu vào và tạo ra đầu ra của bộ tạo bit ngẫu nhiên. Nó có thể đơn giản như hàm định danh hoặc phức tạp như một hàm mật mã. OGF có thể định dạng hoặc chặn đầu ra để phù hợp với quy ước về giao diện bên ngoài. Thông thường, OGF là hàm một chiều, do đó rất khó để tìm ra hàm ngược của OGF nhằm khôi phục một phần của trạng thái bên trong.

OGF là tất định và luôn luôn tạo ra cùng một đầu ra cho bất kỳ giá trị cụ thể nào của trạng thái bên trong. Vì vậy, mối quan hệ giữa OGF và hàm chuyển đổi trạng thái bên trong vô cùng quan trọng. Hàm chuyển đổi trạng thái bên trong phải được gọi ít nhất một lần để cập nhật trạng thái làm việc giữa các hoạt động liên tiếp của hàm tạo đầu ra.

6.2.6.2 Các yêu cầu đối với hàm tạo đầu ra

Các yêu cầu đối với hàm tạo đầu ra của một bộ tạo bit ngẫu nhiên như sau:

1. Hàm tạo đầu ra là tất định (với mọi đầu vào cho trước) và phải được kiểm tra bằng kiểm thử đã biết câu trả lời. Kết quả của việc kiểm thử đã biết câu trả lời được tách ra từ đầu ra hoạt động.
2. Hàm tạo đầu ra phải sử dụng thông tin từ trạng thái bên trong chứa độ bất định đủ lớn theo độ an toàn được yêu cầu.
3. Đầu ra sẽ bị chặn lại cho đến khi trạng thái bên trong nhận được độ bất định được đánh giá là phù hợp.
4. Khi một trạng thái bên trong cụ thể được sử dụng làm đầu ra, thì trạng thái bên trong phải được thay đổi để tạo ra nhiều đầu ra hơn.
5. Hàm tạo đầu ra phải chống lại quan sát và phân tích thông qua việc tiêu thụ điện năng, thời gian, bức xạ hoặc các kênh kề phù hợp khác.

6.2.6.3 Yêu cầu tùy chọn đối với hàm tạo đầu ra

Hàm tạo đầu ra phải bảo vệ trạng thái bên trong, do đó việc phân tích đầu ra của bộ tạo bit ngẫu nhiên không tiết lộ thông tin hữu ích gì về trạng thái bên trong.

6.2.7 Hàm hỗ trợ

6.2.7.1 Giới thiệu về hàm hỗ trợ

Các hàm hỗ trợ liên quan đến việc đánh giá chất lượng của bộ tạo bit ngẫu nhiên: không thay đổi trạng thái bên trong. Hàm hỗ trợ bao gồm các hàm đánh giá độ bất định của đầu vào, hàm đánh giá chất lượng thống kê của đầu ra và hàm kiểm tra xem các hàm bên trong có bị xâm nhập hay không.

Hàm bên trong (tức là "chất lượng" của phần tất định trong bộ tạo bit ngẫu nhiên) có thể được kiểm tra hiệu quả nhất bằng kiểm thử đã biết câu trả lời. "Chất lượng" của nguồn bất định và chất lượng đầu ra cần phải được kiểm tra bằng các kỹ thuật hoặc kiểm thử thống kê.

Thông tin cụ thể về việc đánh giá chất lượng của bộ tạo bit ngẫu nhiên tất định và bất định có trong mục 8 và 9.

6.2.7.2 Các yêu cầu đối với hàm hỗ trợ

Các yêu cầu đối với hàm hỗ trợ của một bộ tạo bit ngẫu nhiên như sau:

1. Bộ tạo bit ngẫu nhiên phải được thiết kế để cho phép kiểm thử nhằm đảm bảo rằng bộ tạo hoạt động chính xác.
2. Khi một bộ tạo bit ngẫu nhiên kiểm thử thất bại, bộ tạo bit ngẫu nhiên phải chuyển sang trạng thái lỗi và đưa ra cảnh báo lỗi. Bộ tạo bit ngẫu nhiên không được thực hiện bất cứ hoạt động nào khi đang trong trạng thái lỗi.

7 Phân loại bộ tạo bit ngẫu nhiên

7.1 Giới thiệu về các loại bộ tạo bit ngẫu nhiên

Mỗi bộ tạo bit ngẫu nhiên sẽ có một nguồn bất định chính. Bộ tạo bit ngẫu nhiên được chia thành hai loại cơ bản phục thuộc vào tính chất của nguồn bất định chính. Nguồn bất định là nguồn bất định không xác định hoặc nguồn bất định xác định.

Nếu nguồn bất định là một hệ thống có thể trích xuất một lượng độ bất định bất kỳ bằng cách lấy mẫu, với điều kiện là hệ thống này hoạt động trong một khoảng thời gian đủ dài, thì nó được coi là một nguồn bất định không xác định. Cụ thể là không có thuật toán tất định nào có thể dự đoán đầu ra của một nguồn bất định không xác định.

Nếu nguồn bất định là một giá trị mầm, thì được gọi là nguồn bất định xác định. Giá trị mầm (xem 9.3) có thể được cung cấp cho bộ tạo bit ngẫu nhiên từ hệ thống bên ngoài hoặc từ bên trong bộ tạo bit ngẫu nhiên. Nếu giá trị mầm được cung cấp từ ngoài bộ tạo bit ngẫu nhiên, giả thiết rằng bộ tạo có độ bất định đầy đủ và được rút ra từ một tập cụ thể. Mặc khác, nếu giá trị mầm được cung cấp từ bên trong bộ tạo bit ngẫu nhiên, nó được đánh giá là có đủ độ bất định và được rút ra từ một tập cụ thể.

Phải đảm bảo rằng kẻ tấn công không thể kiểm soát hoàn toàn giá trị mầm nhằm làm suy giảm độ bất định của đầu ra bộ tạo bit ngẫu nhiên.

Một bộ tạo bit ngẫu nhiên được gọi là bộ tạo bit ngẫu nhiên bất định nếu nguồn bất định chính là không xác định. Một bộ tạo bit ngẫu nhiên được gọi là bộ tạo bit ngẫu nhiên tất định nếu nguồn bất định chính là xác định. Bộ tạo bit ngẫu nhiên cũng có thể sử dụng nguồn bất định bổ sung có thể là tất định hoặc bất định.

7.2 Bộ tạo bit ngẫu nhiên bất định

Bộ tạo bit ngẫu nhiên bất định có thể được phân loại dựa vào tính chất của nguồn bất định không xác định. Một nguồn bất định không xác định là nguồn vật lý hoặc phi vật lý.

Nguồn bất định không tất định vật lý là nguồn bất định mà trong đó phần cứng chuyên dụng được sử dụng để đo các đặc tính vật lý của một số sự kiện trong thế giới thực. Các thiết bị như vậy tiếp tục cung cấp đầu ra tạo nguồn điện cho thiết bị đo. Ví dụ về nguồn bất định không xác định vật lý bao gồm đo thời gian giữa những lần phân rã phóng xạ của một nguyên tử không ổn định, đo các đặc tính tiếng ồn của đi-ốt "nhiễu" hoặc không ổn định. Xem mục 8.3.2 để biết thêm thông tin chi tiết về các yêu cầu đối với nguồn bất định vật lý.

Một nguồn bất định không xác định phi vật lý là nguồn bất định không xác định bất kỳ không phải là một nguồn bất định không xác định vật lý. *Ví dụ về nguồn bất định không xác định phi vật lý bao gồm đo* thời gian giữa các lần ấn bàn phím hoặc lấy mẫu dữ liệu trong các phần bộ nhớ RAM sử dụng thường xuyên. Xem mục 8.3.3 để biết thêm thông tin chi tiết về các yêu cầu đối với nguồn bất định phi vật lý.

Bộ tạo bit ngẫu nhiên tất định vật lý là một bộ tạo bit ngẫu nhiên có nguồn bất định không xác định vật lý. Bộ tạo bit ngẫu nhiên bất định phi vật lý là một bộ tạo bit ngẫu nhiên có nguồn bất định không xác định phi vật lý.

Một bộ tạo bit ngẫu nhiên bất định (vật lý hoặc phi vật lý) được gọi là "thuần túy" nếu tất cả các nguồn bất định của nó là không xác định.

Một bộ tạo bit ngẫu nhiên bất định lai ghép phải thỏa mãn tất cả các tiêu chuẩn an toàn áp dụng cho các bộ tạo bit ngẫu nhiên bất định thuần túy và cũng phải thỏa mãn các yêu cầu an toàn bổ sung (xem 8.3.5) thường áp dụng cho các bộ tạo bit ngẫu nhiên tất định.

Ưu điểm của việc sử dụng một giá trị mầm làm nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên bất định là cho phép người dùng tham số hóa đầu ra của bộ tạo bit ngẫu nhiên bất định. Do đóa, đầu ra của bộ tạo bit ngẫu nhiên bất định có thể được cá nhân hóa và thậm chí có thể cho phép những người dùng khác nhau cùng sử dụng chung một nguồn bất định không xác định mà không ảnh hưởng đến an toàn của hệ thống. Ngay cả khi không có nhiều người dùng cùng sử dụng chung một nguồn bất định, thì giá trị mầm bí mật được coi là một tính năng an toàn bổ sung.

CHÚ THÍCH 1 Thông tin thêm về việc phân loại các bộ tạo bit ngẫu nhiên có trong [5].

CHÚ THÍCH 2 Ví dụ: một giá trị mầm bí mật có thể là một chức năng an toàn điển hình của bộ tạo bit ngẫu nhiên tất định. Nếu bộ tạo bit ngẫu nhiên bất định đáp ứng các yêu cầu của tiêu chuẩn ngay cả với một hàm chuyển đổi trạng thái đơn giản và hàm đầu ra đơn giản (xem mục 8.2 và E.1). Hàm chuyển đổi trạng thái mật mã hoặc hàm đầu ra mật mã có thể là một chức năng an toàn điển hình của bộ tạo bit ngẫu nhiên tất định.

7.3 Bộ tạo bit ngẫu nhiên tất định

Bộ tạo bit ngẫu nhiên tất định cũng được chia làm hai loại là thuần túy và lai ghép. Một bộ tạo bit ngẫu nhiên tất định được gọi là "thuần túy" nếu tất cả các nguồn bất định của nó là giá trị mầm và là "lai ghép" nếu nó sử dụng một nguồn bất định không xác định là nguồn bất định bổ sung.

Một bộ tạo bit ngẫu nhiên tất định lai ghép phải thỏa mãn tất cả các điều kiện an toàn áp dụng cho các bộ tạo bit ngẫu nhiên tất định thuần túy và cũng phải thỏa mãn các yêu cầu an toàn bổ sung nhất định (xem 9.3.4).

CHÚ THÍCH 1 Độ an toàn của một bộ tạo bit ngẫu nhiên bất định lai ghép dựa trên thành phần không xác định còn độ an toàn của các bộ tạo bit ngẫu nhiên tất định lai ghép (xem 9.3.4) về cơ bản dựa trên các thành phần xác định. Nếu một bộ tạo bit ngẫu nhiên được xem vừa là bộ tạo bit ngẫu nhiên bất định lai ghép vừa là bộ tạo bit ngẫu nhiên tất định lai ghép, thì ký hiệu nó là "bộ tạo bit ngẫu nhiên lai ghép". Rõ ràng, các bộ tạo bit ngẫu nhiên lai ghép có hai quan điểm về an toàn.

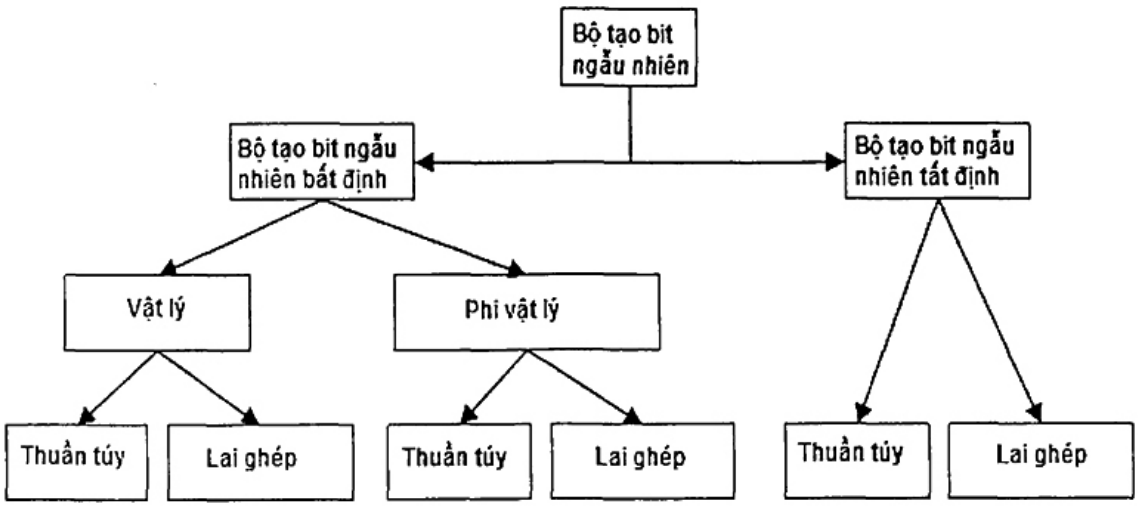
Ưu điểm của việc sử dụng một nguồn bất định không xác định làm nguồn bất định bổ sung là cho phép đầu ra không xác định và giúp ngăn chặn việc phân tích mã và/hoặc thêm các tính năng an toàn như tính an toàn về phía trước hoặc phía sau.

CHÚ THÍCH 2 Thông tin thêm về việc phân loại các bộ tạo bit ngẫu nhiên tất định có trong [4].

7.4 Phân loại bộ tạo bit ngẫu nhiên

Việc phân loại bộ tạo bit ngẫu nhiên bất định và bộ tạo bit ngẫu nhiên tất định được mô tả trong hình 2.

Sự phân biệt giữa một bộ tạo bit ngẫu nhiên bất định và một bộ tạo bit ngẫu nhiên tất định là sự phân loại của một loạt lựa chọn thiết kế. Ở một đầu là một bộ tạo bit ngẫu nhiên tất định được thiết kế để sử dụng một giá trị mầm cho toàn bộ vòng đời của nó. Ở đầu kia là một bộ tạo bit ngẫu nhiên bất định đơn giản được thiết kế với đầu ra của một nguồn bất định không xác định, mạnh và ngẫu nhiên. Các thiết lế bộ tạo bit ngẫu nhiên trung gian cho phép thay mầm mới định kỳ cho bộ tạo bit ngẫu nhiên tất định hoặc đưa các nguồn bất định không xác định vào bộ tạo bit ngẫu nhiên tất định. Các thiết kế bộ tạo bit ngẫu nhiên trung gian cho phép các bộ tạo bit ngẫu nhiên bất định có đầu ra cũng phụ thuộc vào giá trị mầm hoặc các bộ tạo bit ngẫu nhiên bất định xử lý đầu ra của nguồn bất định một cách phức tạp tương tự như cách một bộ tạo bit ngẫu nhiên tất định xử lý giá trị mầm. Lựa chọn đúng đắn cho một bộ tạo bit ngẫu nhiên là sự đánh đổi giữa chi phí và lợi ích tùy thuộc vào yêu cầu của ứng dụng.



Hình 2: Phân loại bộ tạo bit ngẫu nhiên

8 Giới thiệu và yêu cầu đối với bộ tạo bit ngẫu nhiên bất định

8.1 Giới thiệu về bộ tạo bit ngẫu nhiên bất định

Trong phần lớn các ứng dụng, bộ tạo bit ngẫu nhiên tất định sẽ được chọn sử dụng để tạo ra bit ngẫu nhiên. Do đó, việc sử dụng một bộ tạo bit ngẫu nhiên bất định đáp ứng tiêu chuẩn này sẽ tạo ra các giá trị mầm khởi tạo ngẫu nhiên cho một bộ tạo bit ngẫu nhiên tất định. Tuy nhiên, tiêu chuẩn này không loại trừ việc sử dụng một bộ tạo bit ngẫu nhiên bất định để tạo ra tất cả các bit ngẫu nhiên cần thiết của ứng dụng.

Theo mục tiêu của tiêu chuẩn này, một bộ tạo bit ngẫu nhiên bất định phải đáp ứng các yêu cầu quy định trong mục 5 và 6. Tuy nhiên, các mục tiêu và yêu cầu duy nhất của bộ tạo bit ngẫu nhiên bất định vượt quá các điểm quy định tại mục 5 và 6 được áp dụng cho các thiết kế của bộ tạo bit ngẫu nhiên bất định, được trình bày cụ thể từ mục 8.3 đến 8.9. Các ví dụ về bộ tạo bit ngẫu nhiên bất định có trong phụ lục E.

8.2 Mô hình chức năng của bộ tạo bit ngẫu nhiên bất định

Mục này sẽ giới thiệu đặc tả về các thành phần và mô hình chung cho một bộ tạo bit ngẫu nhiên bất định. Nó sẽ mô tả hoạt động chung của một bộ tạo bit ngẫu nhiên bất định, bao gồm các mục tiêu của mỗi thành phần chức năng của bộ tạo bit ngẫu nhiên bất định cần đạt được.

Hình 3 minh họa sơ đồ khối chức năng mô tả một bộ tạo bit ngẫu nhiên bất định khái niệm đáp ứng tiêu chuẩn này. Trong sơ đồ này, các đường nét đứt chỉ ra thành phần là tùy chọn phụ thuộc vào các yếu tố khác nhau. Cần lưu ý rằng các thành phần hiển thị không nhất thiết phải được thực thi như là thành phần vật lý thực tế, nhưng phải thực thi chức năng của nó.

Mỗi thành phần cũng như mục tiêu và yêu cầu của thành phần đó giúp ngăn chặn những điểm yếu an toàn liên quan đến quá trình tạo bit ngẫu nhiên trong các ứng dụng và môi trường mật mã. Nhìn chung, mỗi thành phần sau đây sẽ được yêu cầu trong một bộ tạo bit ngẫu nhiên bất định. Trong một số ứng dụng, có thể lập luận rằng không có yêu cầu nào được áp dụng đối với một thành phần nhất định. Nếu điều này được chứng minh và được ghi nhận, thì thành phần đó có thể loại bỏ khỏi bộ tạo bit ngẫu nhiên bất định, vì thành phần đó không có trong ứng dụng hoặc vì mục tiêu của thành phần đó đã được đáp ứng hoặc được giải quyết bằng các thành phần khác.

Sau đây là tổng quan về cách thức mà các thành phần này tương tác để tạo ra đầu ra ngẫu nhiên. Nguồn bất định không xác định hoạt động theo xác suất thường không tạo ra đầu ra ngẫu nhiên chấp nhận được.

Một hàm chuyển đổi trạng thái bên trong dựa trên một hoặc nhiều hàm mật mã tất định kết hợp một số lượng nhất định dữ liệu nguồn bất định với dữ liệu trạng thái làm việc để tạo ra một trạng thái làm việc mới. Hàm chuyển đổi trạng thái bên trong hoàn thành quá trình này bằng cách liên kết với mỗi kết hợp có thể của chuỗi đầu vào và trạng thái làm việc hiện tại để tạo ra một giá trị của trạng thái làm việc tiếp theo được xác định bằng giá trị hiện tại của tham số bí mật. Nếu độ dài của dữ liệu đầu vào nguồn bất định được sử dụng cho mỗi lần chuyển đổi trạng thái bên trong là n bit và kích thước của trạng thái làm việc là m bit, thì hàm chuyển đổi trạng thái bên trong là một hàm ánh xạ từ không gian của các chuỗi $(n+m)$ bit sang không gian của các chuỗi m bit. Kích thước tham số sử dụng trong một bộ tạo bit ngẫu nhiên bất định sẽ làm cho số lượng đầu vào có thể của hàm này lớn hơn số lượng đầu ra có thể của hàm. Điều này dẫn đến một số lượng lớn các kết hợp của chuỗi đầu vào và trạng thái làm việc hiện tại được gán cho bất kỳ đầu ra cụ thể nào đối với trạng thái làm việc mới. Ngoài ra, vì các hàm mật mã dựa trên các hàm chuyển đổi trạng thái bên trong điển hình đã được phân tích mật mã kỹ lưỡng, nên có thể giả định rằng hàm chuyển đổi trạng thái bên trong sẽ ánh xạ không gian đầu vào đến không gian đầu ra một cách đồng nhất (tức là mỗi đầu ra có cùng số lượng tiền ảnh giống nhau). Những giả định này đáp ứng mục tiêu tạo ra các chuỗi nhị phân có phân bố đều trong trạng thái làm việc. Đặc điểm để phân biệt giữa bộ tạo bit ngẫu nhiên tất định và bộ tạo bit ngẫu nhiên bất định là bộ tạo bit ngẫu nhiên bất định sẽ thêm độ bất định mới vào trạng thái của nó với tốc độ lớn hơn hoặc bằng với tốc độ mà nó tạo ra độ bất định. Do đó, bất cứ khi nào ứng dụng đòi hỏi đầu ra ngẫu nhiên, bộ tạo bit ngẫu nhiên bất định đảm bảo rằng trạng thái bên trong chứa đủ độ bất định chưa được sử dụng để tạo ra đầu ra ngẫu nhiên, sau đó sử dụng hàm tạo đầu ra để xử lý trạng thái làm việc hiện tại để tạo ra đầu ra ngẫu nhiên.

Hàm tạo đầu ra có trạng thái bên trong, cần phải giữ bí mật đối với kẻ tấn công và tạo ra đầu ra ngẫu nhiên. Hàm tạo đầu ra cũng thường là một hàm mật mã hoặc hàm khác có phân phối đều. Một đối số tương tự như đối với hàm chuyển đổi trạng thái bên trong dẫn đến kết luận rằng với các lựa chọn kích cỡ tham số phù hợp, hàm tạo đầu ra sẽ tạo ra đầu ra nhị phân có phân phối đều.

Đối với nhiều nguồn bất định vật lý, độ bất định của mỗi bit phải đủ lớn để các bit này có thể ứng dụng trực tiếp hoặc ứng dụng được sau một thao tác xử lý sau đơn giản (ví dụ: XOR đầu ra của nguồn bất định với trạng thái trong hiện tại). Ngoài ra, hoạt động của nguồn bất định vật lý thường có thể được mô tả bằng cách sử dụng một mô hình ngẫu nhiên (xem ví dụ [7]). Mô hình như vậy cho phép kiểm tra thống kê để phát hiện lỗi của nguồn (tức là khi nguồn tạo ra đầu ra không đủ ngẫu nhiên cho ứng dụng). Vì vậy có thể lựa chọn các hàm chuyển đổi trạng thái và hàm tạo đầu ra rất đơn giản, điều này tạo điều kiện cho việc xác định tính ngẫu nhiên của các bit đầu ra. Các tham số bí mật có thể không cần thiết trong trường hợp này.

Đối với các nguồn bất định phi vật lý, độ bất định của mỗi bit có thể thấp hơn so với các nguồn bất định vật lý và khó xác định chính xác thời điểm nguồn bất định bị lỗi. Trong những trường hợp như vật, cần phải có nhiều hoạt động xử lý sau phức tạp hơn để đảm bảo rằng đầu ra của bộ tạo bit ngẫu nhiên là hoàn toàn ngẫu nhiên và đối phó được với lỗi không xác định của nguồn bất định.

Một bộ tạo bit ngẫu nhiên bất định an toàn bao gồm các cơ chế được thiết kế để tăng khả năng hoạt động an toàn liên tục trong trường hợp lỗi hoặc thỏa hiệp. Lỗi phát hiện được giải quyết thông qua một loạt các kiểm thử chất lượng định kỳ đối với các thành phần khác nhau. Các lỗi hoặc thỏa hiệp không thể phát hiện được giải quyết theo hai cách. Thứ nhất, trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định bao gồm một tham số bí mật thay đổi định kỳ dùng làm tham số cho hoạt động tất định của hàm chuyển đổi trạng thái bên trong. Do thành phần này, thông tin về phần còn lại của trạng thái nội bộ (được gọi là trạng thái làm việc) của bộ tạo bit ngẫu nhiên bất định và tất cả các đầu vào cho bộ tạo bit

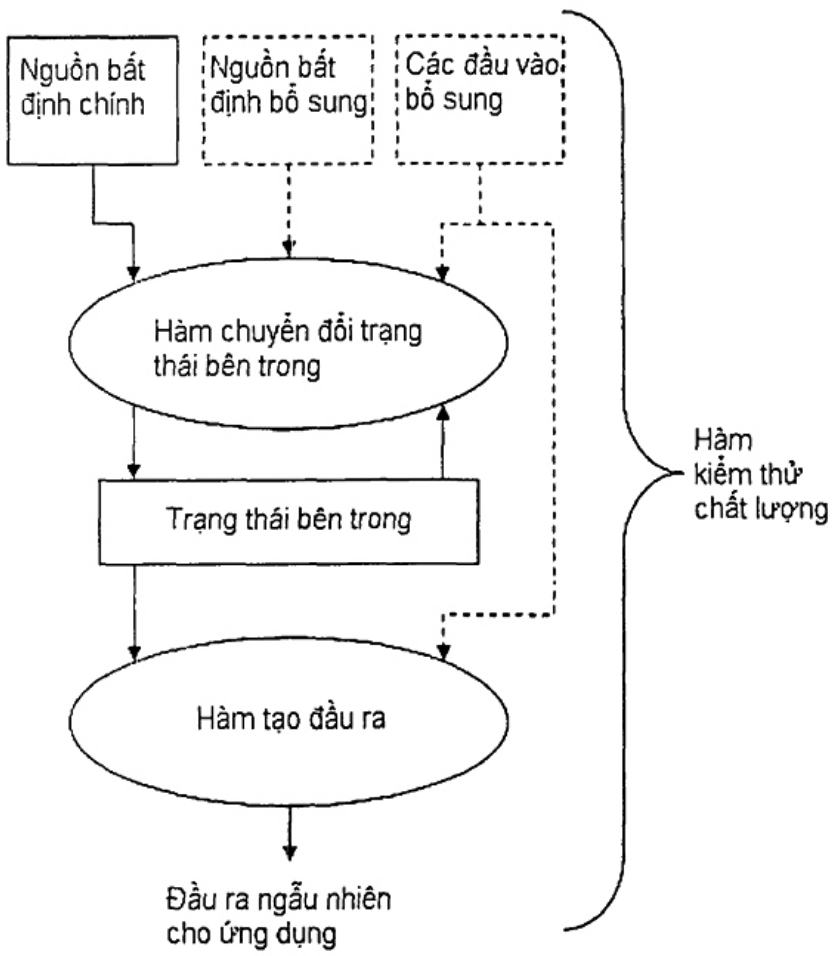
ngẫu nhiên bất định là không đủ để xác định đầu ra của bộ tạo bit ngẫu nhiên bất định. Cách thứ hai là xác định một biên độ an toàn trong việc duy trì độ bất định trong quá trình hoạt động của bộ tạo bit ngẫu nhiên bất định, do đó giảm độ bất định đầu vào có sẵn do các sự kiện bất thường hoặc mô hình thống kê không chính xác dẫn đến đầu ra ngẫu nhiên bị chệch. Mục tiêu của một bộ tạo bit ngẫu nhiên bất định phù hợp với tiêu chuẩn này là bộ tạo bit ngẫu nhiên bất định phải tiếp tục hoạt động an toàn như bộ tạo bit ngẫu nhiên tất định trong trường hợp nguồn bất định bị lỗi.

Yêu cầu bộ tạo bit ngẫu nhiên bất định phải tiếp tục hoạt động an toàn như bộ tạo bit ngẫu nhiên tất định trong trường hợp nguồn bất định bị lỗi có thể được bỏ qua nếu:

1. Có thể kiểm tra rằng kiểm thử chất lượng đối với nguồn bất định được thực hiện sớm để phát hiện các điểm yếu và lỗi có thể của nguồn bất định, có thể làm giảm chất lượng của các bit ngẫu nhiên vượt quá mức chấp nhận được; và
2. Các biện pháp thích hợp được thực hiện trong trường hợp đó (ví dụ: ngừng việc tạo ra các bit ngẫu nhiên).

CHÚ THÍCH Để đáp ứng các yêu cầu này đối với nguồn bất định vật lý, có thể sử dụng phần cứng chuyên dụng. Đề xuất chung được quy định trong [13].

Tính an toàn về phía trước và phía sau là các đặc tính của một bộ tạo bit ngẫu nhiên bất định được thực thi và hoạt động tốt vì không biết đầu vào. Trên thực tế, đó là một tính năng vốn có của một bộ tạo bit ngẫu nhiên bất định vì bộ tạo bit ngẫu nhiên bất định luôn có độ bất định mới cho mỗi lần gọi. Do đó, tính an toàn về phía trước và phía sau được tự động cung cấp nếu nguồn bất định cung cấp đủ độ bất định.



Hình 3: Sơ đồ khối của một bộ tạo bit ngẫu nhiên bất định

8.3 Nguồn bất định của bộ tạo bit ngẫu nhiên bất định

8.3.1 Nguồn bất định chính của bộ tạo bit ngẫu nhiên bất định

8.3.1.1 Giới thiệu về nguồn bất định chính của bộ tạo bit ngẫu nhiên bất định

Thành phần này hoạt động như nguồn không thể dự đoán trước trong bộ tạo bit ngẫu nhiên bất định bằng cách cung cấp dữ liệu được hàm chuyển đổi trạng thái bên trong xử lý. Nguồn không thể dự đoán này khác với nguồn trong một bộ tạo bit ngẫu nhiên tất định chỉ dựa vào giá trị mầm khởi tạo chưa biết nên không thể đoán được. Trong bộ tạo bit ngẫu nhiên bất định, tính không thể đoán trước được dựa trên việc sử dụng một hoặc nhiều nguồn bất định. Những nguồn này được gọi là nguồn bất định khác xác định và được phân loại thành nguồn vật lý và phi vật lý.

Nguồn bất định vật lý là nguồn trong đó phần cứng chuyên dụng được sử dụng để đo các đặc tính vật lý của một chuỗi sự kiện trong thế giới thực. Ví dụ: nguồn bất định vật lý bao gồm đo thời gian giữa những lần phân rã của nguyên tử không ổn định và đầu ra của một đi-ốt nhiễu, nhận được mức điện áp đầu vào liên tục và phát ra mức điện áp tương tự có phân phối đều và liên tục.

Nguồn bất định phi vật lý là nguồn bất định không xác định và không phải là nguồn vật lý. Ví dụ về nguồn bất định phi vật lý là dữ liệu hệ thống hoặc nội dung RAM trong một máy tính cá nhân, tín hiệu không liên tục dựa trên sự kiện xảy ra bất thường hoặc tương tác trong một khoảng thời gian chẳng hạn như lấy mẫu bộ đếm tốc độ cao bất cứ khi nào có thao tác nhấn phím trên bàn phím.

Thông tin thêm và những yêu cầu cụ thể cho nguồn bất định vật lý và phi vật lý có trong mục 8.3.2 và 8.3.3 tương ứng.

Tùy thuộc vào nguồn bất định, đầu ra của nguồn bất định có thể không đủ để sử dụng trực tiếp cho đầu ra ngẫu nhiên vì đầu ra không ở dạng chuỗi số nhị phân hoặc bị chệch. Những hạn chế này được khắc phục trong một bộ tạo bit ngẫu nhiên bất định bằng cách xử lý đầu ra nguồn bất định với hàm chuyển đổi trạng thái bên trong để tạo ra dữ liệu đầu ra nhị phân.

Độ bất định đầu ra từ bộ tạo bit ngẫu nhiên bất định không thể lớn hơn độ bất định của đầu vào.

8.3.1.2 Yêu cầu đối với nguồn bất định chính của bộ tạo bit ngẫu nhiên bất định

Nguồn bất định chính là nền tảng của hoạt động không xác định trong bộ tạo bit ngẫu nhiên bất định. Theo định nghĩa của một bộ tạo bit ngẫu nhiên bất định, thiết kế của nó phải bao gồm thành phần này.

Yêu cầu chức năng của nguồn bất định chính như sau:

1. Mặc dù nguồn bất định không bắt buộc phải tạo ra đầu ra không chệch và độc lập, nhưng nó phải có những tính chất xác suất tức là không thể xác định được bằng bất kỳ quy tắc thuật toán đã biết.
2. Bộ tạo bit ngẫu nhiên bất định phải có thành phần đánh giá tỷ lệ độ bất định đóng góp từ nguồn bất định. Điều này đòi hỏi hoạt động của nguồn bất định phải dựa trên các nguyên tắc được thiết lập tốt hoặc hoạt động đặc trưng phổ biến để có thể xác định một mô hình thống kê phù hợp với nguồn bất định.
3. Nguồn bất định phải phù hợp với việc kiểm thử bằng một phòng kiểm thử hoặc một quy trình kiểm tra độc lập khác để đảm bảo hoạt động hợp lý. Đặc biệt có thể thu thập một mẫu dữ liệu từ nguồn bất định trong quá trình kiểm tra hợp lệ hoặc xác minh độc lập để cho phép đánh giá mô hình thống kê đã yêu cầu, tỷ lệ độ bất định và sự phù hợp của các bài kiểm thử chất lượng đối với nguồn bất định.
4. Phải phát hiện được lỗi hoặc việc suy giảm nghiêm trọng nguồn bất định. Dựa trên tần suất thực hiện các bài kiểm thử chất lượng, việc phát hiện này có thể không phải là ngay lập tức.

5. Nguồn bất định phải được bảo vệ khỏi sự phá hoại của kẻ tấn công. Đặc biệt, kẻ tấn công không có khả năng ảnh hưởng đến nguồn bất định bằng cách làm giảm độ bất định của nguồn xuống dưới ngưỡng.

8.3.1.3 Yêu cầu tùy chọn đối với nguồn bất định chính của bộ tạo bit ngẫu nhiên bất định

Các tính năng tùy chọn nhưng được kiến nghị của nguồn bất định chính như sau:

1. Nguồn bất định phải giữ được các tính chất toán học. Nếu không giữ được thì phải có giới hạn về xác suất biến động. Hoạt động xác suất của nguồn không được thay đổi đáng kể theo thời gian. Ví dụ: nếu nguồn bất định tạo ra đầu ra từ một bảng chữ cái nhất định có phân bố thống kê, nó phải nhất quán trong độ chệch này theo thời gian. Một nguồn bất định không cố định sẽ làm phức tạp quá trình ước tính tỷ lệ đóng góp độ bất định và tăng độ khó của việc kiểm tra hợp lệ của thiết kế bộ tạo bit ngẫu nhiên bất định (trừ khi thiết kế bao gồm các nguồn bất định bổ sung đáp ứng được yêu cầu này).
2. Các bài kiểm thử chất lượng được thiết kế phù hợp với mô hình thống kê đã biết của nguồn phải chú trọng đặc biệt đến việc phát hiện hành vi có khả năng xảy ra gần ranh giới giữa môi trường hoạt động và điều kiện bất thường. Điều này đòi hỏi cần hiểu thấu đáo về hoạt động của nguồn bất định.

8.3.2 Nguồn bất định vật lý cho bộ tạo bit ngẫu nhiên bất định

8.3.2.1 Giới thiệu về nguồn bất định vật lý của bộ tạo bit ngẫu nhiên bất định

Nguồn bất định vật lý thường cung cấp một dòng liên tục giá trị đầu ra khi được cung cấp nguồn điện. Tuy nhiên, đầu ra này không nhất thiết phải là một chuỗi nhị phân đơn giản. Ví dụ: nếu nguồn bất định vật lý dựa trên thời gian giữ những lần phân rã phóng xạ, nguồn bất định không thể cung cấp một chuỗi đầu ra nhị phân đơn giản có chứa độ bất định nhưng cung cấp một loạt khoảng thời gian giữa những phần phân rã phóng xạ. Ngoài ra, nếu nguồn bất định vật lý được sử dụng dựa trên sự thay đổi điện áp trong một đi-ốt nhiễu thì đầu ra có thể là một loạt kết quả của phép đo điện áp.

Thông thường, nguồn bất định này cần được giải thích rõ ràng trước khi sử dụng. Điều này có thể được thực hiện bằng cách so sánh dữ liệu vật lý do nguồn bất định cung cấp với một loạt các giá trị ngưỡng. Dữ liệu được cung cấp bởi nguồn bất định có thể được so sánh với một giá trị ngưỡng duy nhất, do đó một lần đọc nguồn tạo ra một bit đầu ra hoặc được so sánh với một loạt các giá trị ngưỡng sao cho một lần đọc nguồn tạo ra một số bit đầu ra.

Vì một nguồn bất định vật lý thường được chứa trong một ranh giới mật mã của bộ tạo bit ngẫu nhiên bất định, điều kiện an toàn được nhấn mạnh là việc thu thập và giải thích về nguồn bất định vật lý thực sự chứ không phải là khả năng kẻ tấn công có thể lấy được thông tin và/hoặc gây ảnh hưởng quá mức đến nguồn.

Một ưu điểm của nguồn bất định vật lý là có thể mô hình hóa như một quá trình ngẫu nhiên. Điều này dẫn đến việc kiểm tra chất lượng trực tuyến luôn hiệu quả nhằm xác định chính xác thời điểm nguồn bất định vật lý bị lỗi. Các bài kiểm tra như vậy có thể được sử dụng để ngăn chặn một bộ tạo bit ngẫu nhiên bất định vật lý tiếp tục hoạt động với một nguồn bất định có thể dự đoán được và có thể loại bỏ một số hàm chuyển đổi trạng thái phức tạp hoặc hàm tạo đầu ra.

8.3.2.2 Yêu cầu đối với nguồn bất định vật lý của bộ tạo bit ngẫu nhiên bất định

Các yêu cầu chức năng đối với nguồn bất định vật lý như sau:

1. Các giá trị ngưỡng phải được chọn sao cho chuỗi đầu ra chứa một lượng độ bất định đủ lớn để ứng dụng. Lưu ý rằng có sự khác biệt giữa độ bất định do nguồn hiển thị (có thể tạo dữ liệu ở độ chính xác vô hạn) và độ bất định được cung cấp bởi biểu diễn nhị phân của nguồn đó.
2. Tổng lỗi của nguồn bất định phải được phát hiện ngay lập tức. Sự suy giảm của nguồn bất định phải được phát hiện đủ nhanh, trong đó "đủ nhanh" phụ thuộc vào mức độ suy giảm.

8.3.2.3 Yêu cầu tùy chọn đối với nguồn bất định vật lý của bộ tạo bit ngẫu nhiên bất định

Nguồn bất định phải được phân tích chính thức và các giá trị ngưỡng được chọn sao cho đầu ra chứa lượng độ bất định lớn nhất có thể. Điều này có thể xảy ra vì các nguồn bất định vật lý thật sự là tương đối đơn giản so với các nguồn bất định con người.

8.3.3 Nguồn bất định phi vật lý cho bộ tạo bit ngẫu nhiên bất định

Nguồn bất định phi vật lý thường được cung cấp bởi một hệ thống theo yêu cầu của bộ tạo bit ngẫu nhiên bất định. Do đó, các nguồn bất định phi vật lý thường nằm ngoài ranh giới bảo vệ xác định của bộ tạo bit ngẫu nhiên bất định (xem 5.4). Dữ liệu thường ở dạng nhị phân. Ví dụ: nguồn có thể là biểu diễn số của thời gian giữa những lần nhấn phím (được đo bằng hệ thống), của số liệu thống kê mạng không thể đoán trước hoặc nội dung của các phần không thể dự đoán trong bộ nhớ RAM.

Vì nguồn bất định phi vật lý có ít nhất một phần nằm ngoài sự điều khiển của bộ tạo bit ngẫu nhiên bất định, nên phải có các biện pháp ngăn chặn để giảm thiểu khả năng kẻ tấn công thu được thông tin về dữ liệu và/hoặc khả năng ảnh hưởng đến nguồn bất định. Ngoài ra, vì rất khó để mô hình hóa một nguồn bất định phi vật lý một cách chính xác như là một quá trình ngẫu nhiên, nên càng khó để xác định được liệu nguồn đó có hoạt động chính xác hay không. Do đó, bộ tạo bit ngẫu nhiên bất định phi vật lý thường sử dụng các hàm chuyển đổi trạng thái và hàm tạo đầu ra phức tạp nhằm đảm bảo rằng bộ tạo bit ngẫu nhiên ít nhất phải an toàn như một bộ tạo bit ngẫu nhiên tất định trong khoảng thời gian từ khi nguồn bất định bị lỗi đến khi lỗi được phát hiện.

Không có thêm yêu cầu an toàn cho nguồn bất định phi vật lý.

8.3.4 Nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên bất định

8.3.4.1 Giới thiệu về nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên bất định

Hoạt động của bộ tạo bit ngẫu nhiên bất định cũng bao gồm một hoặc nhiều nguồn bất định bổ sung. Một nguồn bất định bổ sung có thể hữu ích vì nhiều lý do. Nó cung cấp một lớp bảo vệ chống lại sự suy giảm đầu ra của bộ tạo bit ngẫu nhiên bất định do nguồn bất định chính lỗi hoặc có sự sai khác với mô hình thống kê đặc trưng.

Trong trường hợp nguồn bất định chính được đánh giá từ bên ngoài, thì nguồn bất định bổ sung ít có khả năng bị truy cập từ ngoài vào sẽ làm giảm lượng thông tin hữu ích mà kẻ tấn công có thể thu được từ nguồn bất định chính.

Cuối cùng, việc sử dụng nhiều nguồn bất định cung cấp khả năng chia tách quyền kiểm soát, cho phép ứng dụng mà nhiều người dùng cần truy cập vào cùng một đầu ra của bộ tạo bit ngẫu nhiên bất định nhưng không tin tưởng vào những ảnh hưởng tiềm ẩn đối với các nguồn bất định riêng lẻ. Đối với các ứng dụng như vậy, có thể thiết kế bộ tạo bit ngẫu nhiên bất định sao cho sự tin tưởng của người dùng vào một nguồn bất định duy nhất là đủ để tin tưởng vào đầu ra cuối cùng của bộ tạo bit ngẫu nhiên bất định.

Bộ tạo bit ngẫu nhiên bất định có một nguồn bất định bổ sung không phải là nguồn bất định không xác định mà là nguồn bất định xác định, tức là một giá trị mầm. Một bộ tạo bit ngẫu nhiên bất định có nguồn

bất định xác định thì được gọi là bộ tạo bit ngẫu nhiên bất định lai ghép. Bộ tạo bit ngẫu nhiên bất định lai ghép được quy định trong 8.3.5.

Mặc dù có thêm tính không thể dự đoán được đối với đầu ra của bộ tạo bit ngẫu nhiên, nhưng độ an toàn của bộ tạo bit ngẫu nhiên chỉ dựa vào nguồn bất định chính. Do đó, đầu ra của một bộ tạo bit ngẫu nhiên phải được giữ an toàn ngay cả khi kẻ tấn công biết đầu ra của tất cả các nguồn bất định bổ sung và/hoặc khi kẻ tấn công có phương pháp đo sự ảnh hưởng đối với đầu ra của các nguồn bất định bổ sung.

8.3.4.2 Yêu cầu đối với nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên bất định

Các yêu cầu chức năng đối với nguồn bất định bổ sung như sau:

Một nguồn bất định bổ sung sẽ được đưa vào nếu:

a) Nguồn bất định chính không đủ tin cậy do bị lỗi. Trong trường hợp này, nguồn bất định bổ sung phải đáp ứng các yêu cầu tương tự như nguồn bất định chính;
b) Nguồn bất định chính tạo ra độ bất định với tốc độ không bằng với tốc độ tạo bit ngẫu nhiên mong muốn. Trong trường hợp này, nguồn bất định bổ sung phải đáp ứng các yêu cầu tương tự như nguồn bất định chính. Ngoài ra, thay vì sử dụng một nguồn bất định bổ sung, giải pháp giải quyết vấn đề này được chấp nhận là sử dụng bộ tạo bit ngẫu nhiên bất định chỉ để tạo ra giá trị mầm ban đầu cho một bộ tạo bit ngẫu nhiên tất định; và
c) ứng dụng hoặc môi trường đòi hỏi bộ tạo bit ngẫu nhiên thực hiện các chức năng được cung cấp tốt nhất bởi một bộ tạo bit ngẫu nhiên bất định lai ghép, tức là bộ tạo bit ngẫu nhiên lấy giá trị mầm như một nguồn bất định. Trong trường hợp này, nguồn bất định bổ sung phải đáp ứng các điều kiện trong 8.3.5.

8.3.4.3 Yêu cầu tùy chọn đối với nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên bất định

Các tính năng tùy chọn nhưng được kiến nghị sử dụng của nguồn bất định bổ sung như sau:

1. Cho dù nguồn bất định bổ sung có cùng kiểu với nguồn bất định chính (tức là phiên bản thứ hai của cùng một nguồn bất định), hay là một thành phần hoặc quá trình hoàn toàn khác, thì nguồn này phải hoạt động độc lập với nguồn chính để đảm bảo rằng nguồn bất định kết hợp sẽ không mất độ bất định do sự phụ thuộc thống kê. Tính độc lập của các nguồn bất định cũng tạo điều kiện cho việc thiết kế và đánh giá hoặc quá trình kiểm tra độc lập bằng cách cho phép phân tách nguồn bất định chính và nguồn bất định thứ hai một cách riêng biệt và cũng làm giảm khả năng lỗi của nguồn bất định chính.
2. Đầu ra của bộ tạo bit ngẫu nhiên phải không thể dự đoán được ngay cả khi kẻ tấn công có thể lựa chọn các giá trị của nguồn bất định bổ sung; không thể dự đoán được bit tiếp theo được tạo ra bởi bộ tạo bit ngẫu nhiên với xác suất lớn hơn đáng kể 1/2.
3. Nguồn bất định thứ hai nên được đưa vào trong thiết kế của bộ tạo bit ngẫu nhiên bất định nếu một trong hai điều sau là đúng:
   a) Nguồn bất định chính là phần không ổn định (nghĩa là không nhất quán) trong hoạt động thống kê làm cho việc ước lượng độ bất định đầu vào trở nên khó khăn hơn; hoặc
   b) Kẻ tấn công có thông tin hay gây ảnh hưởng đến nguồn bất định chính. Trong trường hợp này, nguồn bất định bổ sung phải đáp ứng các yêu cầu tương tự như nguồn chính, mặc dù có thể chấp nhận có một số thành phần tất định. Nghĩa là thao tác của người sử dụng hoặc các yêu tố từ môi trường hệ thống có thể ảnh hưởng (mặc dù không hoàn toàn xác định) đầu ra từ nguồn này.

8.3.5 Bộ tạo bit ngẫu nhiên bất định lai ghép

Một bộ tạo bit ngẫu nhiên bất định được coi là một bộ tạo bit ngẫu nhiên bất định lai ghép nếu nó lấy nguồn tất định làm đầu vào bổ sung. Ưu điểm chính của bộ tạo bit ngẫu nhiên bất định lai ghép là đầu ra không thể dự đoán được của nó được tham số hóa bằng giá trị mầm. Điều này giúp tăng cường độ an toàn vì thông tin về đầu ra của nguồn bất định chính không đủ để cho phép kẻ tấn công phá vỡ lược đồ hoặc cho phép các bộ tạo bit ngẫu nhiên sử dụng một nguồn bất định không xác định duy nhất với những giá trị mầm khác nhau mà không ảnh hưởng đến độ an toàn.

Các yêu cầu chức năng bổ sung đối với một bộ tạo bit ngẫu nhiên bất định lai ghép như sau:

1. Kẻ tấn công không thể dự đoán được bit tiếp theo với xác suất lớn hơn đáng kể 1/2 ngay cả khi kẻ tấn công có toàn quyền kiểm soát giá trị mầm.
2. Không kẻ tấn công nào có thể khôi phục lại thông tin về giá trị mầm bằng cách quan sát đầu ra của bộ tạo bit ngẫu nhiên.
3. Không người dùng nào không có thẩm quyền có thể điều khiển, ảnh hưởng hoặc cập nhật giá trị mầm.

8.4 Đầu vào bổ sung của bộ tạo bit ngẫu nhiên bất định

8.4.1 Giới thiệu về đầu vào bổ sung của bộ tạo bit ngẫu nhiên bất định

Hoạt động của bộ tạo bit ngẫu nhiên bất định yêu cầu lấy một số đầu vào công khai và/hoặc do người dùng tạo ra như lệnh, dữ liệu về sự biến thiên điện năng và biến thiên thời gian như bộ đếm, xung, dữ liệu do người dùng cung cấp. Có thể giả thiết rằng những đầu vào bổ sung này có thể trực tiếp quan sát được hoặc dưới sự kiểm soát trực tiếp của kẻ tấn công. Do đó, điều quan trọng là việc điều khiển các giá trị đầu vào này không làm giảm hiệu suất của bộ tạo bit ngẫu nhiên, hoặc chỉ có một cá nhân hoặc nhân viên đã xác thực và có thẩm quyền có khả năng điều khiển những giá trị đầu vào này với một chính xác hoạt động được xác định duy nhất.

8.4.2 Yêu cầu đối với đầu vào bổ sung của bộ tạo bit ngẫu nhiên bất định

Yêu cầu chức năng của mọi đầu vào bổ sung phải bao gồm khả năng bảo vệ chống lại thao tác của kẻ tấn công (lệnh, xung, bộ hẹn giờ, nguồn điện...) gây ảnh hưởng đến độ an toàn của bộ tạo bit ngẫu nhiên.

CHÚ THÍCH Điều này có thể được thực hiện bằng cách hạn chế ảnh hưởng lên các đầu vào này trong việc kiểm soát tổng thể bộ tạo bit ngẫu nhiên bất định. Nguồn điện đầu vào là một trường hợp đặc biệt; sự gián đoạn của nguồn điện rõ ràng sẽ dẫn tới việc từ chối hoàn toàn dịch vụ. Môi trường hoạt động của bộ tạo bit ngẫu nhiên bất định phải được cung cấp điện liên tục. Đây là một vấn đề hệ thống và nằm ngoài phạm vi của tiêu chuẩn này.

8.5 Trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

8.5.1 Giới thiệu về trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Thành phần này chứa thông tin được chuyển qua lại giữa các lần gọi đến bộ tạo bit ngẫu nhiên bất định và tất cả thông tin được xử lý trong một lần yêu cầu. Vì lý do này, trạng thái bên trong là một yêu cầu bắt buộc; tuy nhiên không bắt buộc là bất kỳ phần nào của trạng thái bên trong phụ thuộc vào các trạng thái trước đó, nghĩa là không yêu cầu bắt buộc đối với bất cứ phần nào của trạng thái bên trong được chuyển sang lần gọi kế tiếp của bộ tạo bit ngẫu nhiên bất định (ví dụ: khi tung đồng xu, không có trạng thái bên trong nào được chuyển sang lần tung đồng xu tiếp theo). Trong những trường hợp như vậy, trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định hoàn toàn phụ thuộc vào đầu ra của nguồn bất định tại thời điểm bộ tạo bit ngẫu nhiên bất định được sử dụng.

Tuy nhiên, bằng cách giữ lại thông tin trạng thái này, bộ tạo bit ngẫu nhiên bất định có thể tạo đầu ra ngẫu nhiên là một hàm của đầu vào hiện tại từ nguồn bất định và của nhiều (hoặc tất cả) đầu vào trước đó. Điều này cung cấp một lớp bảo vệ chống lại lỗi hoặc sự suy giảm của nguồn bất định, cũng như sự thỏa hiệp đầu ra ngẫu nhiên của kẻ tấn công có thông tin hoặc ảnh hưởng đến nguồn bất định.

Trạng thái bên trong bao gồm hai phần. Trạng thái làm việc là một phần của trạng thái bên trong được xử lý kết hợp với dữ liệu nguồn bất định bằng hàm chuyển đổi trạng thái bên trong để tạo ra trạng thái bên trong mới. Thành phần này có thể bao gồm một "bộ trữ" ngẫu nhiên ngoài bộ đếm hoặc các giá trị khác tùy chọn. Phần thứ hai của trạng thái bên trong là một giá trị được gọi là tham số bí mật. Tham số bí mật là một đầu vào bổ sung cho hàm chuyển đổi trạng thái bên trong, tùy chỉnh hàm trong trường hợp cụ thể của bộ tạo bit ngẫu nhiên bất định. Do đó, nó đóng vai trò như một lớp bảo vệ bổ sung chống lại sự suy giảm chất lượng của nguồn bất định. Tùy thuộc vào việc xử lý các tham số bí mật, nó cũng có thể bảo vệ chống lại việc gây tổn hại đến trạng thái làm việc.

Khi khởi tạo đầu ra của nguồn bất định phải được đặt vào trạng thái bên trong, trạng thái làm việc là bắt buộc. Tất cả, một phần hoặc không có trạng thái bên trong nào có thể được chuyển sang lần gọi bộ tạo bit ngẫu nhiên bất định tiếp theo. Việc sử dụng tham số bí mật là tùy chọn, nhưng khi sử dụng, giá trị đó thường được giữ lại giữa các lần gọi bộ tạo bit ngẫu nhiên bất định và chỉ có thể được cập nhật bởi cá nhân hoặc nhân viên được xác thực và được ủy quyền trong ranh giới của chính sách hoạt động. Thông tin thêm về các vấn đề liên quan đến kiểm soát thông số bí mật được cung cấp trong ISO/IEC 11770-1 [6].

8.5.2 Yêu cầu đối với trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Các yêu cầu chức năng của trạng thái bên trong như sau:

1. Thiết kế của bộ tạo bit ngẫu nhiên bất định phải bảo vệ để kẻ tấn công biết hoặc ảnh hưởng đến trạng thái bên trong.

CHÚ THÍCH 1 Một phương tiện để đạt được yêu cầu trên bao gồm việc chỉ định trạng thái bên trong tới vùng bộ nhớ chỉ có thể truy cập vào bộ tạo bit ngẫu nhiên bất định, lưu trữ bộ tạo bit ngẫu nhiên bất định trên một máy tính hoặc thiết bị độc lập hoặc thông qua các chính sách an ninh bảo vệ thiết bị và môi trường của nó.

2. Nếu tồn tại thì tham số bí mật phải được bảo vệ khỏi kẻ tấn công trong ranh giới bảo vệ của bộ tạo bit ngẫu nhiên bất định được thiết kế để phát hiện các nỗ lực xâm nhập trái phép.
3. Giá trị khởi tạo của tham số bí mật nếu tồn tại phải chứa độ bất định đủ lớn để đáp ứng yêu cầu an toàn của ứng dụng. Tham số bí mật khởi tạo này có thể được tạo ra bởi bộ tạo bit ngẫu nhiên bất định hoặc một bộ tạo bit ngẫu nhiên bất định khác. Nếu tham số bí mật được tạo ra bởi chính bộ tạo bit ngẫu nhiên bất định, bộ tạo bit ngẫu nhiên bất định phải hoạt động ở chế độ đặc biệt dành riêng cho mục đích này, trong đó kết quả đầu ra ngẫu nhiên trở thành tham số bí mật.

   Tuy nhiên, nếu độ an toàn bổ sung được coi là một yêu cầu để bảo vệ chống lại các kịch bản kẻ tấn công thu được thông tin hoặc ảnh hưởng đến nguồn bất định có thể xảy ra, thì quá trình tạo tham số bí mật khởi tạo phải lấy dữ liệu bất định bổ sung từ một thành phần hệ thống khác hoặc thông qua tương tác của người dùng được kết hợp một cách nào đó với dữ liệu của nguồn bất định (tức là độ bất định bổ sung đóng vai trò là nguồn bất định thứ hai tạm thời).

CHÚ THÍCH 2 Ví dụ về tương tác của người dùng có thể bao gồm nhưng không giới hạn: nhấn phím, thời gian giữa các lần nhấn phím hoặc di chuyển chuột.

4. Trong trường hợp có các giá trị không nên làm tham số bí mật (ví dụ: khóa mật mã "yếu") thì tham số bí mật phải được kiểm tra để đảm bảo rằng các giá trị tham số bí mật này không được sử dụng.

5. Tham số bí mật nếu có thì phải được thay thế định kỳ. Điều này hỗ trợ mục tiêu an toàn về phía trước và phía sau. Tham số bí mật chỉ được cập nhật thông qua các lệnh của nhân viên được xác thực và được ủy quyền trong ranh giới hoạt động của chính sách an toàn.
6. Nếu một tham số bí mật tồn tại và không được lấy từ một bộ tạo bit ngẫu nhiên bất định khác, thì quá trình thay thế hoặc cập nhật tham số bí mật phải liên quan đến các nguồn bất định và hàm chuyển đổi trạng thái bên trong.

CHÚ THÍCH 3 Lược đồ thay thế chỉ đơn giản là thay thế hoặc thêm tham số bí mật hiện tại với đầu ra ngẫu nhiên từ bộ tạo bit ngẫu nhiên bất định mà không được sử dụng cho bất kỳ mục đích gì khác.

7. Nếu nguồn bất định bị lỗi và trạng thái làm việc bị ảnh hưởng, thì bộ tạo bit ngẫu nhiên bất định phải chống lại mọi khả năng để buộc nó tạo ra đầu ra sẵn có.

CHÚ THÍCH 4 Bộ tạo bit ngẫu nhiên bất định ngừng hoạt động khi (các) nguồn bất định lỗi, hoặc, nếu bộ tạo bit ngẫu nhiên bất định bao gồm một tham số bí mật bằng cách đảm bảo rằng bộ tạo bit ngẫu nhiên bất định tiếp tục hoạt động một cách an toàn như bộ tạo bit ngẫu nhiên tất định. Đặc biệt, tham số bí mật phải có đủ dài để chống lại bất kỳ hình thức tấn công mật mã nào lên bộ tạo bit ngẫu nhiên bất định bao gồm cả phương pháp vét cạn.

8.5.3 Yêu cầu tùy chọn đối với trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Các tính năng tùy chọn nhưng được khuyến nghị sử dụng đối với trạng thái bên trong như sau:

1. Kích thước tính bằng bit của trạng thái bên trong phải đủ lớn để bộ tạo bit ngẫu nhiên bất định tiếp tục hoặc động như một bộ tạo bit ngẫu nhiên tất định nếu nguồn bất định lỗi hoặc kẻ tấn công có mọi thông tin hoặc điều khiển được bộ tạo. Nếu dữ liệu của nguồn bất định là hằng số, thì chiều dài của chu kỳ tối đa được giới hạn bởi kích thước của trạng thái bên trong và đặt một giới hạn trên cho công việc mà kẻ tấn công phải thực hiện để khôi phục trạng thái bên trong (thông qua phương pháp vét cạn).
2. Tham số bí mật phải được giữ lại giữa các phiên làm việc để cung cấp cho bộ tạo bit ngẫu nhiên bất định một trạng thái duy nhất có đủ độ bất định tại mỗi lần kích hoạt khởi động mà không cần tạo ngay một giá trị tham số bí mật mới. Nếu điều này được thực hiện, nó sẽ bảo vệ bộ tạo khỏi truy cập của kẻ tấn công.

CHÚ THÍCH Quá trình bảo vệ này có thể dưới hình thức lưu trữ trong một vùng bộ nhớ chỉ có thể truy cập vào quá trình của bộ tạo bit ngẫu nhiên bất định, lưu trữ dưới dạng mã hóa hoặc lưu trữ trong một thẻ có thể tháo dời.

8.6 Hàm chuyển đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

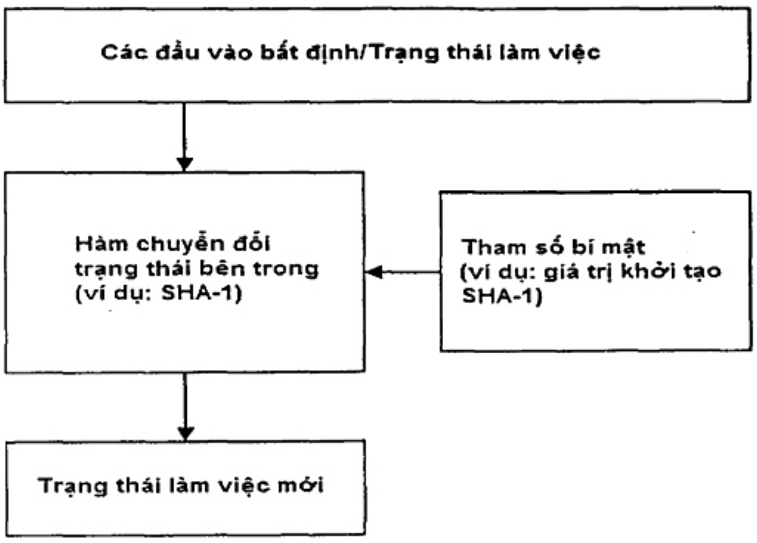
8.6.1 Giới thiệu về hàm chuyển đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Hàm chuyển đổi trạng thái bên trong điều khiển mọi hoạt động làm thay đổi trạng thái bên trong. Bao gồm các hàm bắt buộc nhằm biến đổi đầu ra của nguồn bất định vào trạng thái làm việc và phần hiện tại của trạng thái bên trong vào hàm tạo đầu ra. Thông thường các hàm này hoạt động độc lập với tham số bí mật.

Tuy nhiên, chức năng chính của các hàm chuyển đổi trạng thái bên trong là điều khiển các phần "mang đi" của trạng thái bên trong giữa các lần gọi của bộ tạo bit ngẫu nhiên bất định. Chức năng này là tùy chọn nhưng được khuyến nghị sử dụng. Một chức năng như vậy sẽ hoạt động theo hai phần.

1. Mang theo tham số bí mật mà không thay đổi.
2. Cập nhật trạng thái làm việc sử dụng hàm phụ thuộc vào trạng thái làm việc hiện tại và tham số bí mật (tùy chọn).

Hình 4 là một ví dụ về hàm chuyển đổi trạng thái bên trong với tham số bí mật.



Hình 4: Ví dụ về hàm chuyển đổi trạng thái bên trong

CHÚ THÍCH Trong hình 4, hàm chuyển đổi trạng thái bên trong là SHA-1 và tham số bí mật là hằng số 160 bit cho SHA-1.

8.6.2 Yêu cầu đối với hàm chuyển đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Việc sử dụng hàm chuyển đổi trạng thái bên trong chỉ bắt buộc ở việc thu thập và lưu trữ đầu ra của nguồn bất định. Nếu đây là hàm duy nhất mà các hàm chuyển đổi trạng thái bên trong cung cấp thì không có yêu cầu bổ sung nào mà các hàm chuyển đổi trạng thái bên trong phải đáp ứng.

Nếu hàm chuyển đổi trạng thái bên trong tạo ra một trạng thái làm việc mới bằng việc kết hợp trạng thái bên trong trước đó với đầu ra của nguồn bất định, thì yêu cầu chức năng đối với hàm chuyển đổi trạng thái bên trong như sau.

Đối với mỗi lần thay thế phần tích lũy độ bất định của trạng thái bên trong, dữ liệu của nguồn bất định được xử lý bằng hàm chuyển đổi trạng thái bên trong phải có đủ số lượng để chứa ít nhất là nhiều bit độ bất định là độ an toàn (tính theo bit) được đánh giá của bộ tạo bit ngẫu nhiên bất định.

8.6.3 Yêu cầu tùy chọn đối với hàm chuyển đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên bất định

Các tính năng tùy chọn nhưng được khuyến nghị sử dụng của hàm chuyển đổi trạng thái bên trong như sau:

1. Hàm chuyển đổi trạng thái bên trong phải đạt được tính an toàn về phía sau thông qua việc sử dụng hợp lý hàm một chiều, ví dụ: hàm băm mật mã.
2. Hàm chuyển đổi trạng thái bên trong phải có đặc tính là tất cả các bit trong trạng thái làm việc và đầu vào nguồn bất định sẽ ảnh hưởng đến các bit đầu ra của hàm chuyển đổi trạng thái bên trong.
3. Hoạt động của hàm chuyển đổi trạng thái bên trong phải bảo vệ chống lại việc quan sát và phân tích thông qua tiêu thụ điện năng, thời gian, phân rã phóng xạ hoặc các tấn công kênh kề khác. Những giá trị mà hàm chuyển đổi trạng thái bên trong làm việc (trạng thái bên trong, tham số bí mật và đầu vào nguồn bất định) là các giá trị quan trọng dựa trên tính bí mật của đầu ra ngẫu nhiên. Phân tích kênh kề có thể phá vỡ tính bí mật này.

8.7 Hàm tạo đầu ra của bộ tạo bit ngẫu nhiên bất định

8.7.1 Giới thiệu về hàm tạo đầu ra của bộ tạo bit ngẫu nhiên bất định

Thành phần này cung cấp đầu ra ngẫu nhiên cho ứng dụng yêu cầu bằng cách xử lý tất cả hoặc một tập con các bit trong trạng thái bên trong hiện tại, (cả trạng thái làm việc và tham số bí mật) và bất kỳ đầu vào bổ sung tùy chọn nào. Hàm tạo đầu ra là một thành phần quan trọng để đạt được tính an toàn về phía trước và phía sau. Thành phần này cung cấp tính an toàn về phía sau bằng cách ngăn không cho đầu ra ngẫu nhiên tiết lộ thông tin về các giá trị hiện tại hoặc trước đó của trạng thái bên trong, các đầu vào của nguồn bất định hoặc các đầu ra ngẫu nhiên.

8.7.2 Yêu cầu đối với hàm tạo đầu ra của bộ tạo bit ngẫu nhiên bất định

Các yêu cầu chức năng đối với hàm tạo đầu ra như sau:

1. Hàm tạo đầu ra phải không tạo ra độ chệch trong đầu ra ngẫu nhiên.
2. Đầu ra ngẫu nhiên từ hàm tạo đầu ra phải vượt qua các bài kiểm tra chất lượng thống kê khi được yêu cầu (xem 8.8.5).
3. Hàm tạo đầu ra phải xử lý các bit từ trạng thái bên trong ít nhất bằng số lượng các bit trong mỗi khối đầu ra ngẫu nhiên được tạo bởi hàm tạo đầu ra. Tùy thuộc vào loại hàm tạo đầu ra sử dụng, có thể cần xử lý nhiều hơn số lượng bit này từ trạng thái bên trong.
4. Hàm tạo đầu ra không được sử dụng lại dữ liệu từ phần trạng thái làm việc của trạng thái bên trong khi cung cấp dữ liệu ngẫu nhiên cho ứng dụng yêu cầu. Tức là (a) hàm chuyển đổi trạng thái bên trong sẽ thay thế các phần của trạng thái làm việc giữa những lần gọi đến hàm tạo đầu ra để đảm bảo không sử dụng lại hoặc (b) hàm tạo đầu ra sẽ sử dụng một phần chưa sử dụng trước đó của trạng thái làm việc để đảm bảo rằng dữ liệu không được sử dụng lại.
5. Hàm tạo đầu ra không được rò rỉ bất cứ thông tin nào về trạng thái bên trong có thể ảnh hưởng đến đầu ra tiếp theo.

8.7.3 Yêu cầu tùy chọn đối với hàm tạo đầu ra của bộ tạo bit ngẫu nhiên bất định

Yêu cầu sau đây là tùy chọn. Có thể có trường hợp trong một số thiết kế, việc bao gồm yêu cầu này rất được khuyến khích.

Hàm tạo đầu ra không thể có hàm ngược để tiết lộ thông tin về trạng thái bên trong. Nghĩa là thông tin về đầu ra ngẫu nhiên do hàm tạo đầu ra tạo ra không được tiết lộ thông tin gì về trạng thái hiện tại.

8.8 Kiểm tra chất lượng đối với bộ tạo bit ngẫu nhiên bất định

8.8.1 Giới thiệu về kiểm tra chất lượng đối với bộ tạo bit ngẫu nhiên bất định

Thành phần này đảm bảo rằng toàn bộ quy trình của bộ tạo bit ngẫu nhiên bất định hoạt động chính xác và đầu ra của bộ tạo bit ngẫu nhiên bất định là ngẫu nhiên một cách liên tục. Các bài kiểm tra này sẽ phát hiện ra lỗi trong bộ tạo bit ngẫu nhiên bất định và ngăn ngừa việc sử dụng bộ tạo bit ngẫu nhiên bất định cho đến khi vượt qua các bài kiểm tra chất lượng. Những bài kiểm tra này là một phần trong thiết kế của bộ tạo bit ngẫu nhiên bất định; chúng được thực hiện tự động khi bật nguồn hoặc khởi động mà không cần sự can thiệp của các ứng dụng, quá trình hoặc người dùng khác và người dùng cũng có thể yêu cầu bất cứ khi nào.

Các bài kiểm tra chất lượng được trình bày trong tiêu chuẩn này bao gồm ba bộ: kiểm tra các thành phần tất định của bộ tạo bit ngẫu nhiên bất định, kiểm tra nguồn bất định và kiểm tra đầu ra ngẫu nhiên do bộ tạo bit ngẫu nhiên bất định tạo ra.

Kiểm tra các thành phần tất định được áp dụng cho tất cả các thiết kế của bộ tạo bit ngẫu nhiên bất định. Có thể có trường hợp tỷ lệ độ bất định đầu vào hoặc đầu ra ngẫu nhiên quá thấp để có thể thực

hiện được tất cả các bài kiểm tra chất lượng thống kê trên các nguồn bất định hoặc đầu ra bộ tạo bit ngẫu nhiên bất định. Trong những trường hợp như vậy, nhà thiết kế có thể thay đổi các phép thử hoặc ngưỡng kiểm tra, cho phép kích thước mẫu nhỏ hơn trong khi vẫn giữ xác suất lỗi loại 1 tương đương nhau.

Các yêu cầu chung và các yêu cầu áp dụng cho mỗi bộ kiểm tra này được trình bày trong bốn tiểu mục dưới dây. Trong một số trường hợp, bộ tạo bit ngẫu nhiên bất định có thể có các tính năng hoặc chức năng bổ sung không được quy định trong tiêu chuẩn này và có thể bao gồm các bài kiểm tra riêng biệt. Trong những trường hợp này, nhà thiết kế phải ghi lại các mục tiêu của những tính năng bổ sung này và làm cơ sở cho các bài kiểm tra bổ sung.

Tần suất các bài kiểm tra chất lượng phải được thực hiện phụ thuộc vào thiết kế tổng thể của bộ tạo bit ngẫu nhiên. Ví dụ: nếu có thể đảm bảo rằng mọi lỗi của nguồn bất định có thể được phát hiện một cách nhanh chóng, để đảm bảo kiểm tra tần suất của nguồn bất định, thành phần tất định phải được đơn giản hóa.

8.8.2 Các yêu cầu kiểm tra chất lượng chung đối với bộ tạo bit ngẫu nhiên bất định

Các yêu cầu chức năng đối với cả ba loại kiểm tra chất lượng được nêu ở trên như sau:

1. Bộ tạo bit ngẫu nhiên bất định phải tự động thực hiện các kiểm tra chất lượng toàn diện mỗi lần bật nguồn hoặc khởi động.
2. Bộ tạo bit ngẫu nhiên bất định phải cho phép kiểm tra chất lượng (nguồn bất định, thành phần tất định và đầu ra ngẫu nhiên) được thực hiện "theo yêu cầu".
3. Tất cả đầu ra từ bộ tạo bit ngẫu nhiên phải bị ức chế trong khi thực hiện các kiểm tra chất lượng nhằm che giấu thông tin về hoạt động của bộ tạo bit ngẫu nhiên bất định và ngăn chặn việc rò rỉ thông tin nào về những lỗi có thể (bao gồm kiểm tra chất lượng về các thành phần tất định trong 8.8.3, kiểm tra chất lượng về nguồn bất định trong 8.8.4 và kiểm tra chất lượng về đầu ra ngẫu nhiên trong 8.8.5). Dữ liệu đã vượt qua tất cả các bài kiểm tra về đầu ra ngẫu nhiên có thể được sử dụng như đầu ra ngẫu nhiên sau khi hoàn thành tất cả các bài kiểm tra chất lượng.

CHÚ THÍCH Một ngoại lệ với yêu cầu này có thể tồn tại nếu kiểm tra chất lượng được liên tục áp dụng cho bộ tạo bit ngẫu nhiên bất định (ví dụ: các bài kiểm tra chất lượng về nguồn bất định hoặc kiểm tra thống kê đối với đầu ra). Trong trường hợp này, không cần thực hiện yêu cầu này.

4. Nếu bộ tạo bit ngẫu nhiên bất định được thực thi như phần mềm hoặc phần sụn, các bài kiểm tra chất lượng được thực hiện khi khởi tạo bao gồm kiểm tra tính toàn vẹn của mã thực thi (RAM, ROM hoặc thiết bị logic có thể lập trình). Ví dụ: chữ ký số hoặc mã xác thực thông báo được áp dụng cho phần mềm hoặc phần sụn.

Khi thực thi bộ tạo bit ngẫu nhiên bất định phải thực hiện các bài kiểm tra chất lượng với tần suất tăng lên trong suốt phiên hoạt động mà không làm suy giảm hiệu năng, trừ khi khởi động bật nguồn (khoảng cách hợp lý phải dựa trên tần suất yêu cầu các bit ngẫu nhiên).

8.8.3 Kiểm tra chất lượng đối với các thành phần tất định của bộ tạo bit ngẫu nhiên bất định

Mục tiêu của các kiểm tra này là để đảm bảo các thành phần tất định của bộ tạo bit ngẫu nhiên bất định tiếp tục xử lý chính xác bất kỳ tập đầu vào có thể. Vì theo định nghĩa, không thể dự đoán các thành phần này, phương pháp kiểm thử được chấp nhận là sử dụng kiểm tra với câu trả lời đã biết. Các phép kiểm thử như vậy phải khởi tạo thành phần hoặc hàm đến trạng thái khởi tạo cố định, nhập một đầu vào cố định vào hàm, sau đó so sánh kết quả đầu ra với đầu ra đúng được tính toán trước đó bằng việc thực thi hàm độc lập (ví dụ: mô phỏng máy tính kiểm tra được sử dụng trong quá trình phát triển bộ tạo bit ngẫu nhiên bất định) và được lưu trữ trong quá trình thực thi bộ tạo bit ngẫu nhiên bất định.

Các yêu cầu chức năng đối với kiểm tra chất lượng các thành phần tất định như sau:

1. Kiểm tra với câu trả lời đã biết phải bao gồm các bài kiểm tra chất lượng tổng thể được thực hiện mỗi lần khởi động và/hoặc khởi động lại và theo yêu cầu. Kiểm tra với câu trả lời đã biết phải có trong các bài kiểm tra chất lượng tổng thể thực hiện theo các khoảng thời gian định kỳ.
2. Chuỗi so sánh (kết quả được so sánh với câu trả lời đã biết) được tạo ra cho mọi bài kiểm tra với câu trả lời đã biết phải đủ dài để xác suất vượt qua kiểm thử với các thành phần lỗi hoặc bị suy giảm chấp nhận được. Vì tổng kiểm tra 32 bit được sử dụng trong nhiều ứng dụng đảm bảo thông tin, nên 32 bit là chiều dài khuyến nghị cho các giá trị khi kiểm tra với câu trả lời đã biết.
3. Kiểm tra với câu trả lời đã biết phải có trong toàn bộ bộ tạo bit ngẫu nhiên bất định, không chỉ có trong từng thành phần của bộ tạo bit ngẫu nhiên bất định mà còn có trong tổng thể các thành phần điều khiển bộ tạo bit ngẫu nhiên bất định. Có thể thực hiện bằng cách thiết lập trạng thái bên trong thành một mẫu cố định, chặn dữ liệu của nguồn bất định và thay thế nguồn bất định bằng một chuỗi bit cố định và chạy bộ tạo bit ngẫu nhiên bất định ở chế độ hoạt động. Điều này sẽ tạo ra một dãy đầu ra có chiều dài xác định ít nhất bằng chiều dài được quy định tại điểm 2 ở trên. Đầu ra sau đó được so sánh với giá trị đã xác định trước có được qua việc thực thi hoặc mô phỏng độc lập.
4. Kiểm tra với câu trả lời đã biết phải thực hiện mỗi khía cạnh của hàm đang được kiểm thử. Điều này đòi hỏi mẫu đầu vào cố định phải đủ dài để cung cấp một mẫu đại diện các đầu vào có thể cho mỗi thành phần chức năng chính của hàm được kiểm thử.
5. Tất cả đầu ra từ bộ tạo bit ngẫu nhiên bất định phải bị chặn trong khi thực hiện kiểm tra với câu trả lời đã biết để che giấu thông tin về hoạt động của bộ tạo bit ngẫu nhiên bất định và ngăn chặn việc rò rỉ thông tin về những lỗi có thể có. Dữ liệu đã vượt qua thành công kiểm tra chất lượng với câu trả lời đã biết phải không được sử dụng làm đầu ra ngẫu nhiên sau khi hoàn thành tất cả kiểm tra chất lượng. Giá trị hiện tại của trạng thái bên trong bộ tạo bit ngẫu nhiên bất định sau khi hoàn thành thành công kiểm tra với câu trả lời đã biết sẽ không được sử dụng.

Kiểm tra với câu trả lời đã biết lên các thành phần tất định của bộ tạo bit ngẫu nhiên bất định có thể được loại bỏ để thực thi bộ tạo bit ngẫu nhiên bất định với hai quy trình dự phòng và độc lập (không phải là các nguồn bất định) mà đầu ra của nó được so sánh một cách liên tục. Trong trường hợp này, kết quả không phù hợp khi so sánh được xử lý là lỗi của kiểm tra chất lượng, chuyển sang trạng thái lỗi.

8.8.4 Kiểm tra chất lượng đối với nguồn bất định của bộ tạo bit ngẫu nhiên bất định

Mục tiêu của các kiểm tra này là để phát hiện sự thay đổi trong hoạt động của nguồn bất định từ hoạt động dự định. Vì nguồn bất định được giả định không tạo ra dữ liệu nhị phân độc lập và không chệch trong hầu hết các trường hợp, những bài kiểm tra ngẫu nhiên truyền thống (ví dụ: tần số đơn, chi bình phương và chạy kiểm thử để kiểm tra giả thuyết về các bit độc lập và không chệch) sẽ luôn thất bại, do đó không hữu dụng. Nói chung, các bài kiểm tra về nguồn bất định phải được điều chỉnh cẩn thận với nguồn bất định, có tính đến hoạt động thống kê có phân bố không đều của nguồn bất định hoạt động chính xác.

Đối với các nguồn bất định không xác định tuân theo mô hình thống kê rất phức tạp và đối với các nguồn bất định phi vật lý nói riêng, có thể không khả thi để phát triển kiểm tra thống kê tương ứng chính xác với mô hình thống kê của nguồn bất định. Trong những trường hợp này, cần xác định các kiểm tra đơn giản hơn thay vì xác định liệu thống kê được tính từ một mẫu dữ liệu có nằm trong phạm vi chấp nhận được hay không, xác định liệu mẫu dữ liệu có chứa bất kỳ sự xuất hiện của các giá trị được biết liên quan đến lỗi không. Lựa chọn các mẫu được sử dụng cho các kiểm tra như vậy phải tính đến lỗi của nguồn bất định.

Các yêu cầu chức năng đối với kiểm tra chất lượng của nguồn bất định như sau:

1. Kiểm tra đối với nguồn bất định phải được bao gồm trong các bài kiểm tra chất lượng tổng thể được thực hiện mỗi lần khởi động và/hoặc khởi động lại theo khoảng thời gian định kỳ trong quá trình sử dụng và theo yêu cầu.

CHÚ THÍCH 1 Ví dụ, khoảng thời gian để kiểm tra nguồn bất định có thể được xác định bằng chính sách thực thi mô-đun.

2. Mỗi nguồn bất định tối thiểu phải được kiểm tra cho hoạt động. Nghĩa là kiểm tra phải thu thập một lượng dữ liệu từ nguồn và xác nhận rằng nó không chỉ bao gồm một đầu ra không đổi. (Các đầu ra không đổi thường là các giá trị chỉ gồm một giá trị duy nhất của đầu ra nguồn bất định được số hóa. Ví dụ: nếu nguồn đi-ốt nhiễu tạo ra giá trị 0110 mỗi lần lấy mẫu, nó sẽ bị lỗi trong bài kiểm tra hoạt động.) Kích thước của mẫu dữ liệu thu thập sẽ phụ thuộc vào các đặc tính của nguồn bất định và phải được chọn sao cho khi nguồn bất định đang hoạt động chính xác, xác suất không hoạt động trong một mẫu có kích thước đó phải thấp ở mức chấp nhận được. (10^{-4} là giá trị được khuyến nghị cho tỷ lệ lỗi loại 1).
3. Kiểm tra được áp dụng cho từng nguồn bất định và phải được ghi dưới dạng văn bản đầy đủ về lý do chọn kiểm tra đó. Cơ bản phải chỉ ra lý do tại sao các bài kiểm tra được cho là có khả năng phát hiện lỗi trong nguồn bất định.
4. Nếu kiểm tra chất lượng đối với các thành phần tất định trả về kết quả lỗi, thì bộ tạo bit ngẫu nhiên bất định phải chuyển sang trạng thái lỗi và chỉ ra một điều kiện lỗi. Bộ tạo bit ngẫu nhiên bất định phải không thực hiện việc tạo đầu ra ngẫu nhiên trong khi ở trạng thái lỗi. Để thoát khỏi trạng thái lỗi, bộ tạo bit ngẫu nhiên bất định phải yêu cầu một số hình thức can thiệp, tốt nhất là sự can thiệp của người dùng (ví dụ: nạp điện hoặc khởi động lại) thì mới vượt qua được các bài kiểm tra chất lượng.

CHÚ THÍCH 2 Để khôi phục hoặc thoát khỏi trạng thái lỗi, bộ tạo bit ngẫu nhiên bất định sẽ được yêu cầu thực hiện theo các quy trình bảo trì.

Mặc dù là tùy chọn nhưng kiểm tra đối với nguồn bất định nên bao gồm các kiểm tra các đặc tính đã biết của nguồn bất định.

8.8.5 Kiểm tra chất lượng đối với đầu ra ngẫu nhiên của bộ tạo bit ngẫu nhiên bất định

Mục tiêu của các kiểm tra này là cung cấp kiểm tra cuối cùng về tính ngẫu nhiên của đầu ra từ bộ tạo bit ngẫu nhiên bất định. Nhìn chung, việc đưa kết quả của hàm chuyển đổi trạng thái bên trong và hàm tạo đầu ra vào các bài kiểm tra chất lượng đối với đầu ra ngẫu nhiên từ một bộ tạo bit ngẫu nhiên bất định đóng vai trò ít quan trọng hơn việc kiểm tra được áp dụng trực tiếp đối với đầu ra từ nguồn bất định không xác định. Những hàm này thường thực hiện việc trộn lẫn hoàn toàn thậm chí là một lỗi lớn của nguồn bất định cũng không gây ra các bất thường thống kê có thể phát hiện được trong đầu ra ngẫu nhiên của bộ tạo bit ngẫu nhiên bất định. Đây là hệ quả của yêu cầu bộ tạo bit ngẫu nhiên bất định phải tiếp tục hoạt động như một bộ tạo bit ngẫu nhiên tất định nếu các nguồn bất định bị lỗi hoặc chịu ảnh hưởng của kẻ tấn công. Tuy nhiên, kiểm tra thống kê đối với đầu ra ngẫu nhiên vẫn còn có tác dụng và có những yêu cầu dưới đây.

Các yêu cầu chức năng đối với kiểm tra chất lượng của đầu ra ngẫu nhiên như sau:

CHÚ THÍCH 1 Các yêu cầu chức năng chủ yếu liên quan đến kiểm tra thống kê. Như đã nêu ở trên, thông thường các bài kiểm tra thống kê không có hiệu quả trong việc xác nhận chất lượng của đầu ra ngẫu nhiên khi áp dụng quá trình xử lý sau mật mã mạnh.

1. Kiểm tra đầu ra ngẫu nhiên phải được bao gồm trong kiểm tra chất lượng tổng thể được thực hiện mỗi lần khởi động và/hoặc khởi động lại theo khoảng thời gian định kỳ trong quá trình sử dụng và theo yêu cầu của người dùng.

2. Bộ tạo bit ngẫu nhiên bất định phải được kiểm tra lỗi của một giá trị cố định bằng cách thực hiện kiểm tra trên mỗi khối đầu ra ngẫu nhiên được tạo ra bằng hàm tạo đầu ra. Xem 9.8.8 để có thông tin chi tiết cụ thể về kiểm tra bộ tạo bit ngẫu nhiên.
3. Nếu kiểm tra chất lượng đối với các thành phần tất định trả về kết quả lỗi, thì bộ tạo bit ngẫu nhiên bất định phải chuyển sang trạng thái lỗi và chỉ ra một điều kiện lỗi. Bộ tạo bit ngẫu nhiên bất định phải không thực hiện việc tạo đầu ra ngẫu nhiên trong khi ở trạng thái lỗi. Để thoát khỏi trạng thái lỗi, bộ tạo bit ngẫu nhiên bất định phải yêu cầu một số hình thức can thiệp, tốt nhất là sự can thiệp của người dùng (ví dụ: nạp điện hoặc khởi động lại) thì mới vượt qua được các bài kiểm tra chất lượng.

CHÚ THÍCH 2 Để khôi phục hoặc thoát khỏi trạng thái lỗi, bộ tạo bit ngẫu nhiên bất định sẽ được yêu cầu thực hiện theo các quy trình bảo trì.

Có thể tùy chọn bài kiểm tra chất lượng sau đây để thực hiện trên đầu ra ngẫu nhiên. Khi đề xuất này được chọn, bài kiểm tra chất lượng phải đáp ứng yêu cầu chức năng số 1 ở trên. Kiểm tra chất lượng bộ tạo bit ngẫu nhiên bất định tối thiểu phải bao gồm tập hợp các phép thử sau trên một chuỗi 20.000 bit đầu ra ngẫu nhiên từ bộ tạo bit ngẫu nhiên.

Tập hợp tổng thể các bài kiểm tra đối với bộ tạo bit ngẫu nhiên bất định được coi là vượt qua nếu vượt qua được tất cả bốn phép kiểm tra riêng lẻ. Các ngưỡng chỉ định bên dưới tương ứng với xác suất lỗi loại 1 là 10^{-4}.

CHÚ THÍCH 3 Bốn phép kiểm tra bắt nguồn từ FIPS 140-2 [9] và được điều chỉnh trong AIS-31 [5].

a. Kiểm tra tần số đơn: Cho X là số lượng số 1 trong mẫu. Vượt qua phép kiểm tra nếu $9725 < X < 10275$.

b. Kiểm tra Poker: Chia chuỗi thành 5000 đoạn liên tiếp gồm 4 bit. Đếm số lần xuất hiện của 16 giá trị 4 bit có thể. Cho $f(i)$ là số lần xuất hiện của giá trị 4 bit i (sao cho $0 \leq i \leq 15$). Tính $X = \frac{16}{5000}\sum_{i=0}^{15} f(i)^2 - 5000$. Vượt qua phép kiểm tra nếu $2{,}16 \leq X \leq 46{,}17$.

c. Kiểm tra loạt: Một loạt được định nghĩa là một chuỗi dài nhất của các bit liên tiếp gồm tất cả các bit 1 hoặc tất cả các bit 0. Các lần xuất hiện của loạt gồm các bit 0 và bit 1 liên tiếp có độ dài từ 1 đến 6 được đếm và lưu trữ. Vượt qua phép kiểm tra nếu các số này đều nằm trong khoảng tương ứng được quy định trong Bảng 1 dưới đây. Điều này được áp dụng cho cả bit 0 và bit 1. Trong phép kiểm tra này, loạt có độ dài lớn hơn 6 được coi là có chiều dài 6 bit.

d. Kiểm tra loạt dài: Một loạt dài được định nghĩa là một loạt có độ dài lớn hơn hoặc bằng 27 các bit 0 hoặc 1. Vượt qua phép kiểm tra nếu không tồn tại loạt dài.

CHÚ THÍCH 4 Lý do chọn phép kiểm tra loạt và loạt dài được quy định trong phụ lục J.

Bảng 1: Các khoảng kiểm tra loạt

Độ dài loạt	Khoảng yêu cầu
1	2315 – 2685
2	1114 – 1386
3	527 – 723
4	240 – 384

5	103 – 209
6+	103 – 209

Các kiểm tra bổ sung có thể được thực hiện đối với đầu ra ngẫu nhiên nếu:

1. Cần có sự đảm bảo chắc chắn hơn. Trong trường hợp này, các bài kiểm tra trong yêu cầu tùy chọn ở trên phải được tăng cường hoặc thay thế bằng các bài kiểm tra toàn diện hơn (cỡ mẫu hoặc các bài kiểm tra thống kê bổ sung có xác suất lỗi loại 1 nhỏ hơn hoặc bằng 10^{-4}); và/hoặc
2. Kiểm tra chất lượng đối với đầu ra ngẫu nhiên trả về kết quả lỗi; bộ tạo bit ngẫu nhiên bất định phải lặp lại kiểm tra thêm một số lần nữa, nhưng không vượt quá ba lần. Nếu đầu ra ngẫu nhiên vượt qua kiểm tra trong quá trình kiểm thử này, thì bộ tạo bit ngẫu nhiên bất định có thể tiếp tục hoạt động bình thường.

8.9 Sự tương tác giữa các thành phần trong bộ tạo bit ngẫu nhiên bất định

8.9.1 Giới thiệu về sự tương tác giữa các thành phần trong bộ tạo bit ngẫu nhiên bất định

Các thành phần trong bộ tạo bit ngẫu nhiên bất định được mô tả trong tiêu chuẩn này được thiết kế để đạt được các mục tiêu an toàn nhất định bằng cách đáp ứng các mục tiêu và yêu cầu cụ thể. Phần này trình bày một số yêu cầu bổ sung liên quan đến mối quan hệ giữa các thành phần.

8.9.2 Yêu cầu đối với sự tương tác giữa các thành phần trong bộ tạo bit ngẫu nhiên bất định

1. Sự tương tác và thời gian của hàm chuyển đổi trạng thái bên trong và hàm tạo đầu ra phải đảm bảo rằng tỷ lệ hàm chuyển đổi trạng thái bên trong xử lý thêm đầu vào nguồn bất định bổ sung là đủ để cung cấp dữ liệu có thể sử dụng được cho hàm tạo đầu ra với hạn chế là hàm tạo đầu ra không được sử dụng lại dữ liệu từ trạng thái làm việc. Lưu ý rằng điều này không nhất thiết yêu cầu nguồn bất định và hàm chuyển đổi trạng thái bên trong hoạt động khi hàm tạo đầu ra tạo ra đầu ra. Ví dụ: trạng thái làm việc có thể lớn hơn nhiều chiều dài tối thiểu được chấp nhận của nó (tức là chiều dài của khối đầu ra ngẫu nhiên) cung cấp đủ số lượng bit cho việc tạo ra nhiều khối đầu ra ngẫu nhiên.
2. Quá trình xử lý đầu vào nguồn bất định và trạng thái bên trong (cả trạng thái làm việc và tham số bí mật) bằng hàm chuyển đổi trạng thái bên trong phải đảm bảo đầu vào nguồn bất định, trạng thái làm việc và tham số bí mật độc lập trong việc góp phần tạo độ bất định cho trạng thái làm việc sau mỗi lần thực hiện hàm chuyển đổi trạng thái bên trong, bất kể sự đóng góp độ bất định từ các nguồn khác nhau. Điều này đảm bảo tiêu chuẩn an toàn độc lập trong việc duy trì độ bất định trong bộ tạo bit ngẫu nhiên bất định.

8.9.3 Yêu cầu tùy chọn đối với sự tương tác giữa các thành phần trong bộ tạo bit ngẫu nhiên bất định

Các tính năng tùy chọn nhưng được khuyến nghị áp dụng cho sự tương tác giữa các thành phần trong bộ tạo bit ngẫu nhiên bất định như sau:

1. Các thành phần chức năng trong bộ tạo bit ngẫu nhiên bất định và sự tương tác giữa các thành phần này phải được thiết kế sao cho nếu nguồn bất định bị suy giảm hoàn toàn mà kiểm tra chất lượng không thể phát hiện được, thì các thành phần còn lại của bộ tạo bit ngẫu nhiên bất định phải tiếp tục hoạt động và tương tác như một bộ tạo bit ngẫu nhiên tất định. Để thực hiện điều này, có thể thiết kế bộ tạo bit ngẫu nhiên bất định như một bộ tạo bit ngẫu nhiên tất định sửa đổi để hoạt động trên đầu vào từ một hoặc nhiều nguồn bất định.
2. Bộ tạo bit ngẫu nhiên bất định phải được thiết kế sao cho trạng thái làm việc tiếp tục tích lũy ảnh hưởng từ các nguồn bất định ngay cả khi hàm tạo đầu ra không yêu cầu dữ liệu của trạng thái làm

việc mới (có thể thực hiện dưới dạng quá trình nền khi bộ xử lý hoặc tài nguyên hệ thống có sẵn). Điều này được đặc biệt khuyến nghị nếu tồn tại những khoảng thời gian dài giữa những lần yêu cầu ứng dụng cho đầu ra ngẫu nhiên, trong đó trạng thái bên trong có thể dễ bị ảnh hưởng bởi sự quan sát do thời gian tăng lên (nếu thời gian không tăng lên thì sẽ không bị thay đổi).

9 Giới thiệu và yêu cầu đối với bộ tạo bit ngẫu nhiên tất định

9.1 Giới thiệu về bộ tạo bit ngẫu nhiên tất định

Bộ tạo bit ngẫu nhiên tất định sử dụng một thuật toán tất định đã được phê duyệt để tạo ra một chuỗi bit giả ngẫu nhiên từ một giá trị khởi tạo gọi là mầm, cùng với các đầu vào có thể khác. Do tính chất tất định của quá trình, bộ tạo bit ngẫu nhiên tất định được cho là tạo ra các bit "giả ngẫu nhiên" chứ không phải là các bit ngẫu nhiên, tức là xâu bit do bộ tạo bit ngẫu nhiên tất định tạo ra là dự đoán được và có thể xây dựng lại với kiến thức có được về thuật toán, mầm và thông tin đầu vào khác. Tuy nhiên, nếu đầu vào được giữ bí mật và thuật toán được thiết kế tốt, thì xâu bit sẽ là ngẫu nhiên.

Một bộ tạo bit ngẫu nhiên tất định được chia làm hai loại phụ thuộc vào đầu vào của nó. Mặc dù một bộ tạo bit ngẫu nhiên tất định phải lấy giá trị mầm làm nguồn bất định chính nhưng cũng có thể lấy các nguồn bất định bổ sung làm đầu vào. Nếu các nguồn bất định bổ sung này đều tất định (tức là các giá trị mầm) thì bộ tạo bit ngẫu nhiên tất định được gọi là bộ tạo bit ngẫu nhiên tất định thuần túy. Nếu ít nhất có một nguồn bất định là không xác định thì bộ tạo bit ngẫu nhiên tất định được gọi là bộ tạo bit ngẫu nhiên tất định lai ghép. Một bộ tạo bit ngẫu nhiên tất định lai ghép phải thỏa mãn tất cả các yêu cầu của một bộ tạo bit ngẫu nhiên tất định thuần túy cộng thêm thỏa mãn một số yêu cầu an toàn bổ sung được quy định trong 9.3.4.

Vào bất cứ thời điểm nào sau khi bộ tạo bit ngẫu nhiên tất định khởi tạo giá trị mầm, bộ tạo bit ngẫu nhiên tất định tồn tại trong một trạng thái được xác định bởi tất cả các thông tin đầu vào trước đó. Mầm chính được coi là xác định các "trường hợp" khác nhau của một bộ tạo bit ngẫu nhiên tất định, mỗi chuỗi bit đầu ra (phụ thuộc vào một loạt đầu vào hoặc các giá trị của nguồn bất định bổ sung).

Độ an toàn của bộ tạo bit ngẫu nhiên tất định chỉ phụ thuộc vào nguồn bất định chính, mặc dù các nguồn bất định khác có thể được sử dụng để tạo ra độ bất định bổ sung giúp duy trì tính không thể dự đoán trước của đầu ra ngay cả khi giá trị mầm chính bị ảnh hưởng. Do đó, mầm chính sẽ cung cấp đủ độ bất định để đảm bảo mức an toàn mong muốn bộ tạo bit ngẫu nhiên tất định đạt được.

Các giá trị mầm cho một bộ tạo bit ngẫu nhiên tất định yêu cầu được cung cấp bởi một nguồn đầu vào bất định (ví dụ: bộ tạo bit ngẫu nhiên bất định). Độ an toàn của việc thực thi sử dụng bộ tạo bit ngẫu nhiên tất định là một vấn đề khi thực thi hệ thống; cả bộ tạo bit ngẫu nhiên tất định và nguồn đầu vào bất định phải được xem xét kỹ lưỡng.

Theo mục tiêu của tiêu chuẩn này, một bộ tạo bit ngẫu nhiên tất định phải đáp ứng các yêu cầu quy định trong mục 5 và 6. Khi mục tiêu và yêu cầu duy nhất của bộ tạo bit ngẫu nhiên tất định vượt quá các quy định trong mục 5 và 6 thì được áp dụng đối với thiết kế của bộ tạo bit ngẫu nhiên tất định quy định chi tiết trong 9.3 – 9.9. Các ví dụ về bộ tạo bit ngẫu nhiên tất định có trong phụ lục C.

9.2 Mô hình chức năng của bộ tạo bit ngẫu nhiên tất định

Phần này sẽ giới thiệu bản đặc tả của các thành phần và mô hình chung cho một bộ tạo bit ngẫu nhiên tất định. Sẽ mô tả hoạt động chung của một bộ tạo bit ngẫu nhiên tất định, bao gofm các mục tiêu mà mỗi thành phần chức năng của bộ tạo bit ngẫu nhiên tất định dự kiến sẽ đạt được.

Hình 5 cung cấp một sơ đồ khối chức năng cho một bộ tạo bit ngẫu nhiên tất định đáp ứng tiêu chuẩn này. Cần lưu ý rằng các thành phần hiển thị không nhất thiết phải được thực thi như các chương trình riêng biệt nhưng phải thực thi chức năng. Mỗi thành phần cũng như mục tiêu và yêu cầu của các thành

phần này giúp ngăn chặn những điểm yếu an toàn liên quan đến việc tạo bit ngẫu nhiên đã được biết đến trong các ứng dụng và môi trường mật mã. Nói chung, mỗi thành phần dưới đây sẽ được yêu cầu trong một bộ tạo bit ngẫu nhiên tất định. Trong một số ứng dụng, có thể lập luận rằng không có yêu cầu nào đối với một thành phần cố định. Nếu điều này được chứng minh và được ghi nhận, thì thành phần đó có thể loại bỏ khỏi bộ tạo bit ngẫu nhiên bất định, vì thành phần đó không có trong ứng dụng hoặc vì mục tiêu của thành phần đó đã được đáp ứng hoặc được giải quyết bằng các thành phần khác. Ví dụ: có hai tình huống phổ biến trong đó một bộ tạo bit ngẫu nhiên tất định muốn cập nhật giá trị mầm – do mầm được sử dụng "quá lâu" hoặc nghi ngờ rằng kẻ tấn công đã thực hiện một số kiểm soát nguy hại đối với trạng thái bên trong. Do đó, không cần thiết phải thay mầm mới nếu thiết bị ngừng hoạt động vì một số giới hạn (thời gian, số lần gọi...) và trạng thái bên trong được bảo vệ (ví dụ: bảo vệ bằng phần cứng).

Sau đây là tổng quan về cách thức các thành phần này tương tác để tạo ra đầu ra giả ngẫu nhiên. Mầm được cung cấp trực tiếp hoặc gián tiếp bởi một nguồn bất định không xác định.

Mầm được cung cấp trực tiếp bằng một trong hai cách sau. Cách đầu tiên là mầm được cung cấp trực tiếp nếu nó được tạo ra bởi một bộ tạo bit ngẫu nhiên bất định đáp ứng các yêu cầu của tiêu chuẩn này (tức là một bộ tạo bit ngẫu nhiên bất định được chấp nhận). Cách thứ hai là khi mầm được cung cấp bởi một nguồn bất định bên trong bộ tạo bit ngẫu nhiên tất định. Nguồn bất định như vậy phải được kiểm tra để đảm bảo rằng nó tạo ra đầu ra với tỷ lệ độ bất định đủ lớn tương tự như một bộ tạo bit ngẫu nhiên bất định.

Mầm được cung cấp gián tiếp nếu nó được tạo ra bởi một bộ tạo bit ngẫu nhiên tất định khác đáp ứng các yêu cầu của tiêu chuẩn này (tức là một bộ tạo bit ngẫu nhiên tất định được chấp nhận), giá trị trả về chính là mầm.

Sau khi mầm khởi tạo được cung cấp, nó được nạp vào trạng thái bên trong bằng một hàng chuyển đổi trạng thái bên trong chuyên biệt được gọi là nạp mầm. Chỉ có các bên có thẩm quyền mới có quyền truy cập vào hàm chuyển đổi trạng thái bên trong chuyên biệt đó và không thể quan sát hoặc thay đổi giá trị mầm này.

Sau khi bộ tạo bit ngẫu nhiên tất định được cung cấp một mầm khởi tạo và sẵn sàng làm việc, nó sẽ chuyển sang chế độ hoạt động và thực hiện hai việc: cập nhật trạng thái bên trong và tạo ra một khối dữ liệu ngẫu nhiên. Trạng thái bên trong được cập nhật bằng một hàm chuyển đổi trạng thái bên trong tất định, liên quan đến việc cập nhật trạng thái bên trong trong hình 5, lấy đầu vào là trạng thái bên trong hiện tại, đầu vào bổ sung được cung cấp bởi người dùng và đầu ra của nguồn bất định không xác định bất kỳ. Đặc biệt, nó sẽ không lấy mầm gốc làm đầu vào trừ khi mầm được giữ trong trạng thái bên trong cho mục đích này. Đầu ra của bộ tạo bit ngẫu nhiên tất định được tính bằng hàm tạo đầu ra tất định, lấy đầu vào là trạng thái bên trong hiện tại (vừa được cập nhật).

Trạng thái bên trong là sự kết hợp của một trạng thái làm việc được cập nhật liên tục và một loạt các tham số bí mật. Các tham số bí mật này thường điều khiển hoạt động của hàm chuyển đổi trạng thái bên trong và hàm tạo đầu ra theo cách thức tương tự như khóa mật mã (và chính là khóa mật mã). Việc kiểm soát các tham số bí mật này nằm ngoài phạm vi của tiêu chuẩn.

CHÚ THÍCH 1 Hướng dẫn quản lý khóa có trong ISO/IEC 11770. [6]

Thông thường, một bộ tạo bit ngẫu nhiên tất định cũng bao gồm một cơ chế cập nhật trạng thái bên trong khi một giá trị mầm mới được cung cấp. Quá trình cập nhật này phải được thực hiện bởi hàm chuyển đổi trạng thái bên trong chuyên biệt và được gọi là thay mầm mới. Trạng thái bên trong sau khi được thay mầm mới có thể hoặc không phụ thuộc vào trạng thái bên trong trước đó tùy thuộc vào

phương pháp thay mềm mới nhưng có thể chỉ là xóa sạch trạng thái làm việc của bộ tạo bit ngẫu nhiên tất định và nạp mầm mới giống như hoạt động tạo mầm ban đầu.

Một bộ tạo bit ngẫu nhiên tất định an toàn phải bao gồm các cơ chế được thiết kế để tăng khả năng hoạt động an toàn liên tục trong trường hợp có lỗi hoặc bị ảnh hưởng. Khả năng lỗi trong việc thực hiện được giải quyết thông qua các bài kiểm tra chất lượng trên các thành phần khác nhau, chẳng hạn như kiểm tra với câu trả lời đã biết và kiểm tra đầu ra liên tục.

Mỗi bộ tạo bit ngẫu nhiên tất định được thiết kế để cung cấp tính an toàn về phía trước và phía sau khi quan sát từ bên ngoài ranh giới bộ tạo bit ngẫu nhiên tất định với điều kiện người quan sát không biết mầm hoặc giá trị trạng thái nào.

Khi quan sát từ bên trong ranh giới của bộ tạo bit ngẫu nhiên tất định, bộ tạo bit ngẫu nhiên tất định đó phải cung cấp tính an toàn về phía sau.

Đối với bộ tạo bit ngẫu nhiên tất định, tính an toàn về phía trước phụ thuộc vào việc sử dụng mầm bí mật và chèn thêm đầu vào trong mỗi quá trình tạo với độ bất định đủ lớn để đáp ứng yêu cầu an toàn. Tính an toàn về phía trước được cung cấp cho bộ tạo bit ngẫu nhiên tất định bằng cách bổ sung đầu vào của người dùng. Tuy nhiên, mức độ an toàn về phía trước phụ thuộc vào lượng độ bất định do người dùng nhập vào. Nếu đầu vào của người dùng cung cấp độ bất định ít nhất gấp đôi độ mạnh của bộ tạo bit ngẫu nhiên tất định cho mỗi yêu cầu các bit giả ngẫu nhiên, thì sẽ đảm bảo tính an toàn về phía trước.

CHÚ THÍCH 2 Ví dụ, nếu một bộ tạo bit ngẫu nhiên tất định cung cấp độ an toàn 128 bit, thì để đảm bảo tính an toàn về phía trước cần cả mầm và từng đầu vào bổ sung phải có ít nhất 128 bit bất định.

Điều này không khả thi đối với nhiều ứng dụng. Tuy nhiên, đầu vào của người dùng với một lượng độ bất định nhỏ cung cấp một số mức an toàn về phía trước. Điều này có thể phù hợp với một ứng dụng yêu cầu người dùng nhập vào với độ bất định cao cho các ứng dụng quan trọng (ví dụ: tạo khóa cho chữ ký số).

Đối với tính an toàn về phía sau, một bộ tạo bit ngẫu nhiên tất định thực thi đúng được khởi tạo với một mầm bí mật. Mầm được sử dụng để xác định trạng thái khởi tạo của bộ tạo bit ngẫu nhiên tất định. Đầu ra là một số hàm của trạng thái bên trong và trạng thái bên trong được cập nhật mỗi lần yêu cầu bit. Nếu mầm hoặc trạng thái nào đó bị lộ, các đầu ra trước đó có thể được xác định trừ khi hàm một chiều mật mã mạnh được sử dụng trong thiết kế của bộ tạo bit ngẫu nhiên tất định để chuyển từ trạng thái bên trong sao trạng thái bên trong tiếp theo.

Hình 5: Mô hình bộ tạo bit ngẫu nhiên tất định

9.3 Nguồn bất định của bộ tạo bit ngẫu nhiên tất định

9.3.1 Nguồn bất định chính của bộ tạo bit ngẫu nhiên tất định

Nguồn bất định chính của bộ tạo bit ngẫu nhiên tất định là một giá trị mầm. Giá trị mầm này được lấy từ nguồn bất định với tốc độ đầu ra bất định cho trước giống bộ tạo bit ngẫu nhiên bất định và được nhập vào bộ tạo bit ngẫu nhiên tất định trước khi yêu cầu các bit giả ngẫu nhiên từ bộ tạo bit ngẫu nhiên tất định. Thông tin chi tiết về các yêu cầu đối với nguồn giá trị mầm có trong 9.3.2.

Mầm, kích thước của mầm và độ bất định của mầm (nghĩa là tính ngẫu nhiên) phải được lựa chọn để tối thiểu xác suất chuỗi tạo ra bằng một giá trị mầm giống với chuỗi được tạo ra bằng mầm khác, và làm giảm xác suất mầm được đoán hoặc vét cạn. Vì tiêu chuẩn này không yêu cầu độ bất định đủ lớn cho mầm nhưng yêu cầu phải có độ bất định đủ lớn, độ dài của mầm lớn hơn độ an toàn được quy định nhằm đáp ứng độ bất định cần thiết. Tức là độ dài của mầm tối thiểu phải bằng số bit trong độ an toàn được quy định, tuy nhiên độ dài mầm phải lớn hơn mức tối thiểu để tăng độ tin cậy của độ bất định và việc tái sử dụng mầm. Việc sử dụng độ dài mầm nhỏ nhất chỉ được chấp nhận nếu mầm được tạo ra bởi một nguồn bất định đầy đủ, chẳng hạn như một bộ tạo bit ngẫu nhiên bất định hoặc bộ tạo bit ngẫu nhiên tất định có mầm được cung cấp bởi bộ tạo bit ngẫu nhiên bất định. Nhìn chung mầm có thể có kích thước khác nhau, nhưng đặc tả về các thuật toán bộ tạo bit ngẫu nhiên tất định phải có giả định này cho dù quá trình thực thi cụ thể có thể được tối ưu hóa để hỗ trợ mầm có độ dài nhất định.

Phân tích độ bất định của mầm là một thành phần quan trọng đối với việc đảm bảo an toàn cho bộ tạo bit ngẫu nhiên tất định. Tầm quan trọng của việc phân tích độ bất định được thể hiện rõ nhất khi xảy ra

lỗi. Ví dụ: nếu độ an toàn mong muốn là 80 bit và được cho là đạt được, nhưng thực sự chỉ đạt được 40 bit, lượng độ bất định này đã bị tiêu hao khá nhanh chóng. Ngay cả khi độ bất định thực tế là 60 bit, thì kẻ tấn công chủ động vẫn có thể làm tiêu hao độ bất định với chi phí hợp lý.

Việc tạo ra hoặc nhập độ bất định vào trong bộ tạo bit ngẫu nhiên tất định sử dụng một phương pháp không an toàn có thể làm mất đi các bảo đảm dự kiến. Để đảm bảo tính không thể dự đoán trước, cần phải thực hiện việc thu thập và xử lý mầm. Mầm và việc sử dụng nó trong bộ tạo bit ngẫu nhiên tất định được tạo ra và xử lý như sau.

1. Cấu trúc mầm: Mầm bao gồm đầu vào bất định và chuỗi thông tin cá nhân (xem 9.4 để biết thêm thông tin về chuỗi thông tin cá nhân) khi nguồn bất định không đủ mạnh. Sự kết hợp của đầu vào bất định và chuỗi thông tin cá nhân tùy chọn được gọi là vật liệu mầm. Hàm dẫn xuất được sử dụng để phân phối đầu vào bất định trên toàn bộ mầm (ví dụ: mầm không được xây dựng với tất cả độ bất định trên một đầu của mầm) bất cứ khi nào sử dụng chuỗi thông tin cá nhân, hoặc không sử dụng chuỗi thông tin cá nhân và đầu vào bất định không độc lập và phân bố đều trong chuỗi đầu vào bất định. Giá trị mầm kết quả vẫn là duy nhất cho dù có sử dụng chuỗi thông tin cá nhân hay không.
2. Sử dụng mầm: Bộ tạo bit ngẫu nhiên tất định có thể được sử dụng để tạo ra cả thông tin bí mật và công khai. Trong cả hai trường hợp, mầm phải được giữ bí mật. Không nên sử dụng quá trình khởi tạo đơn của bộ tạo bit ngẫu nhiên tất định để tạo ra cả giá trị bí mật và công khai. Yếu tố chi phí và rủi ro phải được tính đến khi xác định có thể đưa ra các giả thuyết khác nhau về giá trị bí mật và công khai hay không.

CHÚ THÍCH 1 Nếu bộ tạo bit ngẫu nhiên tất định đảm bảo tính an toàn về phía trước và phía sau và nếu quá trình thực thi chống lại được tấn công kênh kề, có thể bỏ qua tính năng an toàn này để không sử dụng quá trình khởi tạo đơn của bộ tạo bit ngẫu nhiên tất định nhằm tạo ra cả giá trị bí mật và công khai mà không làm mất đi độ an toàn.

Mầm được sử dụng để khởi tạo trong quá trình khởi tạo bộ tạo bit ngẫu nhiên tất định sẽ không được sử dụng để thay mầm mới cho cùng một quá trình khởi tạo hoặc sử dụng làm mầm cho bộ tạo bit ngẫu nhiên tất định khác.

3. Độ bất định của mầm: Đầu vào bất định của mầm phải chứa độ bất định đầy đủ cho mức an toàn mong muốn và độ bất định phải được phân bố trên toàn mầm. Ứng dụng có thể hoặc không quan tâm đến tính chống va chạm. Để đáp ứng tính chống va chạm, mầm phải có độ bất định lớn hơn hoặc bằng 120 bit hoặc độ an toàn cần thiết cho ứng dụng (tức là độ bất định ≥ max (120, *security_strength*)). Nếu bộ tạo bit ngẫu nhiên tất định đã chọn và mầm không thể cung cấp độ an toàn như yêu cầu của ứng dụng thì phải sử dụng một bộ tạo bit ngẫu nhiên tất định khác.
4. Kích thước mầm: Kích thước tối thiểu của mầm phụ thuộc vào bộ tạo bit ngẫu nhiên tất định được chọn, độ an toàn mà ứng dụng yêu cầu và nguồn bất định. Kích thước mầm tối thiểu phải lớn hơn hoặc bằng độ an toàn yêu cầu tính bằng bit, tùy thuộc vào tỷ lệ độ bất định và sự đảm bảo nguồn bất định (xem thảo luận ở trên). Ví dụ: nếu cần 160 bit bất định, thì nguồn bất định cần mầm có kích thước 240 bit trở lên để đạt được 160 bit bất định.
5. Tính bí mật của mầm: Mầm được xử lý theo cách phù hợp với yêu cầu an toàn đối với dữ liệu mục tiêu. Ví dụ: nếu bí mật duy nhất trong hệ mật là khóa, thì mầm được sử dụng để tạo khóa cũng được coi là khóa.
6. Chu kỳ sử dụng mầm: Mầm của bộ tạo bit ngẫu nhiên tất định có thời hạn sử dụng nhất định, sau đó sẽ không còn sử dụng được nữa. Mỗi mầm sẽ có một vòng đời hữu hạn nhất định. Mầm phải được cập nhật định kỳ hoặc bị thay thế sau khi hết vòng đời của nó. Nếu biết được giá trị mầm (nghĩa là mầm bị tổn thương), các thực thể trái phép có thể xác định được đầu ra của bộ tạo bit ngẫu nhiên tất định. Thời gian sử dụng có thể tính bằng khoảng thời gian hoặc số lượng tối đa đầu ra được tạo ra bởi mầm đó.

Trong một số ứng dụng (ví dụ: smartcard), không thể thực hiện quá trình thay mầm mới vì quá trình này có thể làm giảm độ an toàn. Trong những trường hợp như vậy, biện pháp tốt nhất là thay thế bộ tạo bit ngẫu nhiên tất định, do đó có được một mầm mới (ví dụ: sử dụng thẻ smartcard mới). Trong các ứng dụng khác, thay mầm mới lại là lựa chọn thiết kế phù hợp. Thay mầm mới (tức là thay thế một mầm bằng mầm mới) là phương pháp khôi phục tính bí mật của đầu ra bộ tạo bit ngẫu nhiên tất định nếu kẻ tấn công biết được giá trị mầm. Thay mầm mới định kỳ là biện pháp đối phó tốt với mối đe dọa tiềm ẩn là mầm và đầu ra bộ tạo bit ngẫu nhiên tất định bị tổn hại. Tuy nhiên, kết quả từ việc thay mầm mới chỉ giống như bộ tạo bit ngẫu nhiên bất định được sử dụng để cung cấp mầm mới (hoặc một chuỗi bộ tạo bit ngẫu nhiên tất định được khởi tạo bởi một bộ tạo bit ngẫu nhiên bất định). Việc tạo ra quá nhiều đầu ra từ một mầm (và thông tin đầu vào khác) cung cấp thông tin quan trọng giúp dự đoán thành công đầu ra sắp tới. Thay mầm mới định kỳ sẽ giảm rủi ro về an toàn, giảm khả năng thỏa hiệp dữ liệu mục tiêu được bảo vệ bởi các cơ chế mã hóa sử dụng bộ tạo bit ngẫu nhiên tất định.

Việc thay mầm mới cho bộ tạo bit ngẫu nhiên tất định phải được thực hiện theo quy định đối với bộ tạo bit ngẫu nhiên tất định cụ thể. Khi có được mầm mới trong quá trình thay mầm, mầm mới đó phải được kiểm tra để đảm bảo rằng hai mầm liên tiếp không giống nhau. Hơn một mầm không được lưu trong bộ tạo bit ngẫu nhiên tất định. Một mầm không được lưu ở dạng ban đầu, nhưng được chuyển đổi bằng quy trình một chiều. Khi mầm mới được tạo ra và so sánh với mầm "cũ" (tức là mầm cũ đã được chuyển đổi), mầm mới phải thay thế cho mầm cũ trong bộ nhớ. Mầm cũ sẽ bị hủy. Nếu mầm mới được xác định là giống hệt mầm cũ, thì phải tạo ra một mầm mới khác.

7. Chia tách mầm: Khi tài nguyên cho phép (ví dụ: dung lượng lưu trữ), nên sử dụng mầm khác nhau (tức là không lặp lại) để tạo ra các loại dữ liệu ngẫu nhiên khác nhau (nghĩa là các "trường hợp" khác nhau của bộ tạo bit ngẫu nhiên tất định). Ví dụ, mầm được sử dụng để tạo ra giá trị công khai phải khác với mầm được sử dụng để tạo ra giá trị bí mật. Mầm được sử dụng bởi kỹ thuật của bộ tạo bit ngẫu nhiên tất định để tạo ra các cặp khóa bất đối xứng phải khác với mầm được sử dụng bởi cùng một kỹ thuật của bộ tạo bit ngẫu nhiên tất định (hoặc khác) để tạo mầm cho bộ tạo bit ngẫu nhiên tất định khác, và tương tự khác với mầm được sử dụng bởi cùng một kỹ thuật của bộ tạo bit ngẫu nhiên tất định (hoặc khác) để tạo khóa đối xứng. Mầm được sử dụng bởi kỹ thuật của bộ tạo bit ngẫu nhiên tất định để tạo ra các thách thức ngẫu nhiên phải khác với mầm được sử dụng bởi cùng một kỹ thuật của bộ tạo bit ngẫu nhiên tất định (hoặc khác) để tạo ra PINS hoặc mật khẩu. Tuy nhiên, số lượng phân tách mầm là quyết định liên quan đến chi phí/lợi ích.

CHÚ THÍCH 2 Nếu bộ tạo bit ngẫu nhiên tất định đảm bảo tính an toàn về phía trước và phía sau và nếu việc thực thi chống lại được tấn công kênh kề, có thể sử dụng số ngẫu nhiên được tạo bởi một trường hợp của bộ tạo bit ngẫu nhiên tất định cho các loại dữ liệu ngẫu nhiên khác nhau mà không làm giảm độ an toàn.

9.3.2 Tạo giá trị mầm cho bộ tạo bit ngẫu nhiên tất định

Giá trị mầm cho một bộ tạo bit ngẫu nhiên tất định được chấp nhận phải được tạo ra bằng một trong ba cách sau:

1. Mầm được tạo ra bằng một bộ tạo bit ngẫu nhiên bất định được chấp nhận để tạo ra đầu ra với tỷ lệ bất định đủ lớn.
2. Mầm được tạo ra bằng một bộ tạo bit ngẫu nhiên tất định được chấp nhận để tạo ra đầu ra với tủ lệ bất định đủ lớn. Bộ tạo bit ngẫu nhiên tất định này cũng phải có mầm được tạo ra theo các yêu cầu về tạo mầm. Do đó, một chuỗi các bộ tạo bit ngẫu nhiên tất định có thể được dự kiến tạo ra mầm cho bộ tạo bit ngẫu nhiên tất định kế tiếp. Tuy nhiên, chuỗi này luôn bắt đầu bằng một bộ tạo bit ngẫu nhiên bất định được chấp nhận hoặc một bộ tạo bit ngẫu nhiên tất định được chấp nhận để tạo ra các giá trị mầm. Nói cách khác, nếu mầm được tạo ra bởi một bộ tạo bit ngẫu nhiên tất định hoặc một chuỗi các bộ tạo bit ngẫu nhiên tất định được chấp nhận, thì bộ tạo bit ngẫu nhiên

tất định cấp cao nhất phải lấy mầm từ một bộ tạo bit ngẫu nhiên bất định hoặc một bộ tạo bit ngẫu nhiên tất định bên trong có bộ tạo bit ngẫu nhiên bất định.

3. Mầm được tạo bởi một nguồn bất định thích hợp. Nguồn này có thể bị chệch và/hoặc tạo ra các bit khác nhau. Nếu giá trị mầm được tạo ra theo cách này thì nhà phát triển phải đánh giá tỷ lệ độ bất định được tạo ra từ nguồn và đảm bảo rằng nguồn đáp ứng tất cả các yêu cầu đối với nguồn bất định có trong 6.2.2, 8.3 và 8.8.4.

9.3.3 Nguồn bất định bổ sung cho bộ tạo bit ngẫu nhiên tất định

Hoạt động của bộ tạo bit ngẫu nhiên tất định cũng bao gồm một hoặc nhiều nguồn bất định bổ sung. Một nguồn bất định bổ sung có thể hữu ích vì nhiều lý do.

Một bộ tạo bit ngẫu nhiên tất định có thể có một nguồn bất định bổ sung là nguồn bất định xác định hoặc nguồn bất định không xác định. Một bộ tạo bit ngẫu nhiên tất định có nguồn bất định không xác định được gọi là bộ tạo bit ngẫu nhiên tất định lai ghép. Các bộ tạo bit ngẫu nhiên tất định lai ghép được thảo luận trong 9.3.4.

Mặc dù có thêm tính không thể dự đoán được đối với đầu ra của một bộ tạo bit ngẫu nhiên, nhưng độ an toàn của bộ tạo bit ngẫu nhiên vẫn chỉ dựa vào nguồn bất định chính. Do đó, đầu ra của một bộ tạo bit ngẫu nhiên được giữ an toàn ngay cả khi kẻ tấn công biết được đầu ra của tất cả các nguồn bất định bổ sung và/hoặc khi kẻ tấn công có một số kết quả đo lường nhất định ảnh hưởng đến đầu ra của các nguồn bất định bổ sung.

Ưu điểm của việc sử dụng một nguồn bất định không xác định làm nguồn bất định bổ sung là cho phép đầu ra của bộ tạo bit ngẫu nhiên tất định trở nên bất định và giúp ngăn chặn phân tích mã và/hoặc thêm các tính năng an toàn như tính an toàn về phía trước và phía sau.

9.3.4 Bộ tạo bit ngẫu nhiên tất định lai ghép

Một bộ tạo bit ngẫu nhiên tất định được gọi là bộ tạo bit ngẫu nhiên tất định lai ghép nếu nó lấy nguồn bất định không xác định làm đầu vào bổ sung; ngược lại được gọi là bộ tạo bit ngẫu nhiên tất định thuần túy.

Các yêu cầu chức năng bổ sung đối với một bộ tạo bit ngẫu nhiên tất định lai ghép như sau:

1. Kẻ tấn công không thể đoán được bit tiếp theo với xác suất lớn hơn đáng kể 1/2, ngay cả khi kẻ tấn công hoàn toàn điều khiển đầu ra của nguồn bất định không xác định.
2. Không kẻ tấn công nào có thể khôi phục lại bất cứ thông tin gì về nguồn bất định không xác định bằng cách quan sát đầu ra của bộ tạo bit ngẫu nhiên.
3. Không người nào không có thẩm quyền có thể thao tác hoặc làm ảnh hưởng đến nguồn bất định không xác định.

9.4 Đầu vào bổ sung của bộ tạo bit ngẫu nhiên tất định

Hoạt động của bộ tạo bit ngẫu nhiên tất định bao gồm các đầu vào bổ sung tùy chọn. Bộ tạo bit ngẫu nhiên tất định có thể yêu cầu thông tin về đầu vào bổ sung trong quá trình khởi tạo và tạo bit. Thông tin này bao gồm các tham số đầu vào khi ứng dụng gọi bộ tạo bit ngẫu nhiên tất định và đầu vào bổ sung có thể công khai. Các đầu vào bổ sung không làm yếu bộ tạo bit ngẫu nhiên.

Tùy thuộc vào bộ tạo bit ngẫu nhiên tất định mà yêu cầu thông tin về biến thời gian, ví dụ: bộ đếm hoặc giá trị ngày tháng/thời gian.

Bộ tạo bit ngẫu nhiên tất định trong tiêu chuẩn này cho phép sử dụng một chuỗi thông tin cá nhân tùy chọn trong quá trình khởi tạo. Chuỗi thông tin cá nhân được sử dụng kết hợp với các bit bất định để tạo ra mầm. Ví dụ về dữ liệu có thể bao gồm trong chuỗi cá nhân bao gồm số sản phẩm và thiết bị,

định danh người dùng, ngày và tem thời gian, địa chỉ IP hoặc bất cứ thông tin nào khác giúp phân biệt bộ tạo bit ngẫu nhiên tất định.

Các yêu cầu chức năng đối với đầu vào bổ sung như sau:

1. Khi bộ đếm được sử dụng trong bộ tạo bit ngẫu nhiên tất định thì nó không được lặp lại trong "trường hợp" của bộ tạo bit ngẫu nhiên tất định. Khi bộ tạo bit ngẫu nhiên tất định được khởi tạo với mầm mới, bộ đếm được thiết lập với một giá trị cố định (ví dụ: đặt thành 1), nhưng phải được cập nhật cho mỗi trạng thái của bộ tạo bit ngẫu nhiên tất định và không được lặp lại.
2. Khi giá trị ngày/giờ được sử dụng trong bộ tạo bit ngẫu nhiên tất định thì không được lặp lại. Bất cứ khi nào giá trị ngày/giờ được yêu cầu bởi bộ tạo bit ngẫu nhiên tất định, thì hoặc sử dụng một giá trị ngày/giờ khác với lần sử dụng trước đó, hoặc kỹ thuật khác phải bổ sung giá trị ngày/giờ để đảm bảo tính duy nhất (ví dụ: bộ đếm nối giá trị ngày/giờ).
3. Khi chuỗi thông tin cá nhân được sử dụng, nó phải là duy nhất cho tất cả các quá trình khởi tạo của cùng một loại bộ tạo bit ngẫu nhiên tất định.
4. Bộ tạo bit ngẫu nhiên tất định phải vẫn an toàn ngay cả khi kẻ tấn công có quyền điều khiển hoàn toàn đầu vào bổ sung của bộ tạo, tức là ngay cả khi kẻ tấn công có thể lựa chọn các giá trị của đầu vào bổ sung, thì vẫn không thể dự đoán được bit tiếp theo do bộ tạo bit ngẫu nhiên tạo ra với xác suất lớn hơn đáng kể 1/2.
5. Phải đảm bảo rằng hàm chuyển đổi trạng thái bên trong cập nhật trạng thái bên trong có thuộc tính là kích thước giới hạn của nó không bị giảm đi sau những lần gọi lặp lại mà không thay mầm mới, vì điều này sẽ làm cạn kiệt độ bất định của trạng thái bên trong.

9.5 Trạng thái bên trong của bộ tạo bit ngẫu nhiên tất định

Trạng thái bên trong là bộ nhớ của bộ tạo bit ngẫu nhiên tất định và bao gồm tất cả các tham số, biến và các giá trị lưu trữ khác mà bộ tạo bit ngẫu nhiên tất định sử dụng. Trạng thái bên trong bao gồm các giá trị được thực hiện bởi hàm chuyển đổi trạng thái bên trong giữa các yêu cầu, khóa được sử dụng trong mỗi lần gọi, đầu vào của người dùng được thu thập khi yêu cầu được xử lý và các tham số biến thời gian được bộ tạo bit ngẫu nhiên tất định sử dụng. Trạng thái bên trong phụ thuộc vào bộ tạo bit ngẫu nhiên tất định cụ thể và bao gồm tất cả các thông tin được yêu cầu để tạo ra các bit giả ngẫu nhiên cho các lần yêu cầu liên tiếp. Một số phần của trạng thái bên trong được thay đổi bởi hàm chuyển đổi trạng thái bên trong trong mỗi vòng lặp của bộ tạo bit ngẫu nhiên tất định.

Trạng thái bên trong được coi là sự kết hợp giữ trạng thái làm việc được cập nhật liên tục trong quá trình thực hiện và tham số bí mật cố định trong quá trình thực hiện và chỉ được cập nhật định kỳ nếu có sự can thiệp của người dùng từ bên ngoài.

Các yêu cầu chức năng đối với trạng thái bên trong như sau:

1. Bộ tạo bit ngẫu nhiên tất định phải được khởi tạo trước khi tạo ra đầu ra. Trong quá trình khởi tạo, trạng thái khởi tạo được thiết lập cho bộ tạo bit ngẫu nhiên tất định, một phần từ mầm. Quá trình khởi tạo của bộ tạo bit ngẫu nhiên tất định có thể được thay mầm mới bất cứ lúc nào. Trạng thái của bộ tạo bit ngẫu nhiên tất định bao gồm thông tin được thực hiện và khóa (tùy chọn) được sử dụng bởi bộ tạo.
2. Trạng thái bên trong phải nằm hoàn toàn trong ranh giới của bộ tạo bit ngẫu nhiên tất định.
3. Trạng thái bên trong phải được bảo vệ ít nhất cũng như việc sử dụng đầu ra dự định của các ứng dụng.
4. Trong trường hợp có các giá trị không nên sử dụng làm tham số bí mật (ví dụ: khóa mật mã "yếu") thì tham số bí mật phải được kiểm tra để đảm bảo rằng các giá trị tham số bí mật này không được sử dụng.
5. Tham số bí mật nếu tồn tại thì phải được thay thế định kỳ.

9.6 Hàm chuyển đổi trạng thái bên trong của bộ tạo bit ngẫu nhiên tất định

Hàm chuyển đổi trạng thái bên trong sử dụng trạng thái bên trong và một hoặc nhiều thuật toán để tạo ra các bit giả ngẫu nhiên. Trong quá trình này, trạng thái bên trong của bộ tạo bit ngẫu nhiên tất định bị xóa. Thuật toán sử dụng và phương pháp xóa trạng thái bên trong phụ thuộc vào bộ tạo bit ngẫu nhiên tất định cụ thể.

Bộ tạo bit ngẫu nhiên tất định trong tiêu chuẩn này có ba hàm chuyển đổi trạng thái bên trong riêng biệt, cụ thể là:

1. Trước khi sử dụng bộ tạo bit ngẫu nhiên tất định lần đầu, nguyên liệu mầm được thu thập và xác định tất cả đầu vào khởi tạo. Đầu vào khởi tạo được dùng để xác định trạng thái khởi tạo của bộ tạo bit ngẫu nhiên tất định.
2. Mỗi yêu cầu bit giả ngẫu nhiên tạo ra các bit theo yêu cầu sử dụng trạng thái bên trong hiện tại và xác định trạng thái bên trong mới được sử dụng cho yêu cầu tiếp theo; và
3. Khi ứng dụng xác định thay mầm mới cho bộ tạo bit ngẫu nhiên tất định, hàm thay mầm mới thu thập nguyên liệu tạo mầm mới, kết hợp nó với các giá trị trạng thái bên trong hiện tại và xác định trạng thái bên trong mới cho lần yêu cầu bit giả ngẫu nhiên tiếp theo. Bằng cách kết hợp nguyên liệu tạo mầm mới với trạng thái bên trong hiện tại, độ bất định có sẵn từ trạng thái hiện tại không mất đi mà được tăng thêm bởi độ bất định của nguyên liệu tạo mầm mới.

Các yêu cầu chức năng đối với hàm chuyển đổi trạng thái bên trong như sau:

a. Bộ tạo bit ngẫu nhiên tất định phải chuyển đổi giữa các trạng thái theo yêu cầu (tức là khi bộ tạo được yêu cầu cung cấp các bit giả ngẫu nhiên mới). Bộ tạo bit ngẫu nhiên tất định cũng có thể được thực thi để chuyển đổi nhằm đối phó với các sự kiện bên ngoài (ví dụ: gián đoạn hệ thống) hoặc chuyển tiếp liên tục (ví dụ: bất cứ khi nào có thời gian để chạy bộ tạo). Tính không thể dự đoán có được khi bộ tạo chuyển tiếp giữa các trạng thái liên tục hoặc để xử lý với sự kiện bên ngoài. Tuy nhiên, khi bộ tạo bit ngẫu nhiên tất định chuyển từ trạng thái này sang trạng thái khác giữa những lần yêu cầu, việc thay mầm mới và/hoặc thay khóa cần được thực hiện thường xuyên hơn.
b. Hàm chuyển đổi trạng thái bên trong đạt được tính an toàn về phía sau bằng việc sử dụng hiệu quả hàm một chiều, chẳng hạn như hàm băm mật mã.
c. Hàm chuyển đổi trạng thái bên trong có đặc tính là tất cả các bit trong trạng thái làm việc (và đầu vào nguồn bất định không xác định nếu có) ảnh hưởng đến các bit đầu ra của hàm chuyển đổi trạng thái bên trong.
d. Hoạt động của hàm chuyển đổi trạng thái bên trong phải được bảo vệ chống lại quan sát và phân tích như tiêu thụ điện năng, thời gian, phân rã phóng xạ hoặc các tấn công kênh kề khác. Các giá trị mà hàm chuyển đổi trạng thái bên trong làm việc (trạng thái bên trong, tham số bí mật và đầu vào nguồn bất định) là những giá trị quan trọng đảm bảo tính bí mật cho đầu ra ngẫu nhiên sắp tới. Tấn công kênh kề có thể phá vỡ tính bí mật này.
e. Phải đảm bảo rằng hàm chuyển đổi trạng thái bên trong cập nhật trạng thái bên trong có đặc tính là kích thước giới hạn của nó không bị suy giảm sau khi lặp đi lặp lại nhiều lần gọi mà không cần thay mầm mới vì điều này sẽ làm mất đi độ bất định của trạng thái bên trong.

9.7 Hàm tạo đầu ra của bộ tạo bit ngẫu nhiên tất định

Hàm tạo đầu ra của bộ tạo bit ngẫu nhiên tất định tạo ra các bit giả ngẫu nhiên là một hàm của trạng thái bên trong bộ tạo bit ngẫu nhiên tất định và bất kỳ đầu vào được sử dụng trong khi hàm chuyển đổi trạng thái bên trong đang hoạt động. Những bit giả ngẫu nhiên này là xác định dựa trên thông tin đầu vào. Định dạng của các bit trước khi xuất ra là xác định bởi một quá trình thực thi cụ thể.

Các yêu cầu chức năng đối với hàm tạo đầu ra như sau:

1. Hàm tạo đầu ra cho phép kiểm tra với câu trả lời đã biết khi được yêu cầu.
2. Bộ tạo bit ngẫu nhiên tất định không tạo đầu ra cho đến khi có sẵn mầm với độ bất định đầy đủ.
3. Bộ tạo bit ngẫu nhiên tất định yêu cầu khóa không tạo đầu ra cho đến khi có khóa.
4. Hàm tạo đầu ra không làm rò rỉ thông tin về trạng thái bên trong có thể làm ảnh hưởng đến đầu ra sắp tới. Hàm tạo đầu ra phải không có hàm ngược để tiết lộ thông tin gì về trạng thái bên trong. Nghĩa là thông tin về đầu ra ngẫu nhiên được tạo ra bởi hàm tạo đầu ra không được tiết lộ thông tin gì về đầu vào của hàm.

9.8 Hàm hỗ trợ của bộ tạo bit ngẫu nhiên tất định

9.8.1 Giới thiệu về các hàm hỗ trợ của bộ tạo bit ngẫu nhiên tất định

Mặc dù không thể hiện trong hình 5, bộ tạo bit ngẫu nhiên tất định phải có các cơ chế để đo chất lượng.

Bộ tạo bit ngẫu nhiên tất định phải được thiết kế để cho phép kiểm tra đảm bảo bộ tạo được thực thi đúng và tiếp tục hoạt động chính xác. Hàm kiểm tra phải sẵn sàng cho mục đích này. Hàm kiểm tra cũng cho phép chèn các giá trị xác định trước của thông tin đầu vào để kiểm tra các kết quả mong đợi (kiểm tra với câu trả lời đã biết). Nếu kiểm tra bị lỗi, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và đưa ra cảnh báo lỗi. Bộ tạo bit ngẫu nhiên tất định sẽ không thực hiện hoạt động nào khi ở trạng thái lỗi và tất cả đầu ra bị chặn lại.

CHÚ THÍCH Trạng thái lỗi bao gồm các lỗi "cứng" chỉ ra sự cố về thiết bị đòi hỏi phải bảo trì, dịch vụ, sửa chữa hoặc thay thế bộ tạo bit ngẫu nhiên tất định hoặc các lỗi "mềm" có thể phục hồi yêu cầu khởi tạo hoặc thiết lập lại bộ tạo bit ngẫu nhiên tất định. Việc khôi phục từ trạng thái lỗi có thể thực hiện được ngoại trừ các lỗi "cứng" đòi hỏi phải bảo trì, dịch vụ, sửa chữa hoặc thay thế bộ tạo bit ngẫu nhiên tất định.

9.8.2 Kiểm tra chất lượng bộ tạo bit ngẫu nhiên tất định

Bộ tạo bit ngẫu nhiên tất định phải thực hiện kiểm tra chất lượng để đảm bảo rằng nó vẫn tiếp tục hoạt động bình thường. Kiểm tra chất lượng đối với chức năng của bộ tạo bit ngẫu nhiên phải được thực hiện khi bộ tạo bit ngẫu nhiên tất định được bật lên, theo yêu cầu (ví dụ: theo yêu cầu của ứng dụng, hoặc khi cài đặt lại, khởi động lại hoặc sạc điện) và trong các điều kiện khác nhau, điển hình là khi thực hiện hàm hoặc thao tác cụ thể (nghĩa là kiểm tra có điều kiện). Một số kiểm tra chất lượng cũng có thể được tiến hành liên tục. Bộ tạo bit ngẫu nhiên có thể thực hiện tùy ý các kiểm tra chất lượng khác đối với chức năng của bộ tạo bit ngẫu nhiên tất định bên cạnh các bài kiểm tra được quy định trong tiêu chuẩn này.

Tất cả đầu ra dữ liệu từ bộ tạo bit ngẫu nhiên tất định sẽ bị chặn lại khi thực hiện các bài kiểm tra này. Kết quả từ kiểm tra với câu trả lời đã biết không tạo ra các bit ngẫu nhiên. Tuy nhiên, các bit được sử dụng trong các loại kiểm tra khác được sử dụng như đầu ra nếu bài kiểm tra đó thành công.

Khi bộ tạo bit ngẫu nhiên tất định không vượt qua kiểm tra chất lượng, nó sẽ chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi. Bộ tạo bit ngẫu nhiên tất định phải không thực hiện bất cứ hoạt động nào khi trong trạng thái lỗi và không dữ liệu nào được xuất ra khi vẫn tồn tại trạng thái lỗi. Khi trong trạng thái lỗi, sự can thiệp của người dùng (ví dụ: sạc điện, khởi động lại bộ tạo bit ngẫu nhiên tất định) sẽ được yêu cầu để thoát khỏi trạng thái lỗi.

9.8.3 Kiểm tra thuật toán tất định của bộ tạo bit ngẫu nhiên tất định

Kiểm tra này phải được thực hiện đối với thuật toán của bộ tạo bit ngẫu nhiên tất định. Kiểm tra với câu trả lời đã biết sẽ được tiến hành khi bật nguồn, khi có yêu cầu và có thể được tiến hành định kỳ. Kiểm tra với câu trả lời đã biết bao gồm việc vận hành thuật toán trên dữ liệu mà đầu ra chính xác đã biết và kiểm tra xem đầu ra được tính toán có bằng với đầu ra mong đợi (câu trả lời đã biết) hay không. Kiểm

tra thất bại nếu đầu ra được tính toán không bằng với câu trả lời đã biết. Trong trường hợp này, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

9.8.4 Kiểm tra tính toàn vẹn phần mềm/phần sụn của bộ tạo bit ngẫu nhiên tất định

Kiểm tra này được áp dụng cho bộ tạo bit ngẫu nhiên tất định có chứa phần mềm hoặc phần sụn. Kiểm tra tính toàn vẹn phần mềm/phần sụn bằng kỹ thuật xác thực được áp dụng cho mọi phần mềm và phần sụn nằm trong bộ tạo bit ngẫu nhiên tất định khi bộ tạo bit ngẫu nhiên tất định được bật nguồn để xác định tính toàn vẹn của mã. Các kỹ thuật xác thực bao gồm mã xác thực thông báo hoặc chữ ký số bằng các thuật toán đã biết, hoặc mã phát hiện lỗi (EDC) khi thực thi mã nằm trong ranh giới mật mã. Kiểm tra này thất bại nếu kết quả tính toán không bằng kết quả được tạo ra trước đó. Trong trường hợp này, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

9.8.5 Kiểm tra các hàm quan trọng của bộ tạo bit ngẫu nhiên tất định

Tất cả các hàm quan trọng liên quan đến hoạt động an toàn của bộ tạo bit ngẫu nhiên tất định phải được kiểm tra khi bật nguồn và khi có yêu cầu. Những hàm quan trọng trong bộ tạo bit ngẫu nhiên tất định được thực hiện trong một số điều kiện cố định cụ thể phải được kiểm tra khi các điều kiện đó phát sinh.

9.8.6 Kiểm tra độ chịu tải phần mềm/phần sụn của bộ tạo bit ngẫu nhiên tất định

Kiểm tra này được thực hiện với các bộ tạo bit ngẫu nhiên tất định chứa phần mềm hoặc phần sụn. Cơ chế mật mã sử dụng kỹ thuật xác thực được chấp nhận (ví dụ: mã xác thực, thuật toán chữ ký số, hoặc HMAC) được áp dụng cho mọi phần mềm và phần sụn (ví dụ: EEPROM, RAM và mạch FPGA) được tải từ bên ngoài vào bộ tạo bit ngẫu nhiên tất định. Kiểm tra này phải xác nhận mã xác thực hoặc chữ ký số. Kết quả tính toán được so sánh với kết quả được tạo ra trước đó. Kiểm tra thất bại nếu hai kết quả này không giống nhau. Trong trường hợp này, bộ tạo bit ngẫu nhiên tất định chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

9.8.7 Kiểm tra nhập khóa thủ công vào bộ tạo bit ngẫu nhiên tất định

Khi thông tin an toàn được nhập thủ công vào bộ tạo bit ngẫu nhiên tất định (ví dụ: mầm hoặc khóa), thông tin an toàn phải bao gồm một mã phát hiện lỗi (EDC) hoặc các giá trị nhập vào trùng lặp được sử dụng để kiểm tra tính chính xác của thông tin an toàn. EDC phải dài ít nhất 16 bit. Bộ tạo bit ngẫu nhiên tất định phải kiểm tra EDC hoặc các giá trị nhập vào trùng lặp xem có phù hợp không. Nếu EDC được tính toán không bằng với EDC nhập vào, hoặc các giá trị nhập vào trùng lặp không khớp, thì không vượt qua kiểm tra. Trong trường hợp này, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

9.8.8 Kiểm tra tạo bit ngẫu nhiên liên tục của bộ tạo bit ngẫu nhiên tất định

Bộ tạo bit ngẫu nhiên tất định phải được kiểm tra với một giá trị cố định. Nếu mỗi lần gọi tạo ra các khối n bit (trong đó $n \geq 80$), khối đầu tiên được tạo ra sau khi bật nguồn, khởi tạo hoặc khởi động lại phải không được sử dụng, nhưng được lưu để so sánh với khối tiếp theo được tạo ra. Mỗi lần tạo ra một khối n bit phải được so sánh với khối được tạo ra trước đó. Kiểm tra thất bại nếu hai khối n bit được so sánh là giống nhau. Nếu kiểm tra thất bại, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

Nếu một lần gọi tới bộ tạo tạo ra ít hơn 80 bit, thì n bit đầu tiên (trong đó $n \geq 80$) được tạo ra sau khi bật nguồn, khởi tạo hoặc khởi động lại phải không được sử dụng, nhưng được lưu để so sánh với n bit tiếp theo được tạo ra. Mỗi lần tạo ra n bit phải được so sánh với n bit được tạo ra trước đó. Kiểm tra thất bại nếu hai chuỗi n bit được so sánh là giống nhau. Nếu kiểm tra thất bại, bộ tạo bit ngẫu nhiên tất định phải chuyển sang trạng thái lỗi và xuất ra một cảnh báo lỗi.

9.9 Yêu cầu bổ sung đối với khóa của bộ tạo bit ngẫu nhiên tất định

Ngoài các yêu cầu về chức năng được áp dụng cho các thành phần của bộ tạo bit ngẫu nhiên tất định, các yêu cầu khác cũng được áp dụng cho việc thực thi và sử dụng bộ tạo bit ngẫu nhiên tất định. Những yêu cầu này được kết hợp với khóa được sử dụng bởi một bộ tạo bit ngẫu nhiên tất định cho trước.

Một số bộ tạo bit ngẫu nhiên tất định yêu cầu sử dụng một hoặc nhiều khóa. Nếu không bị cấm, thì những khóa này có thể được cung cấp từ một nguồn bên ngoài (tức là từ một nguồn bên ngoài ranh giới của bộ tạo bit ngẫu nhiên tất định), hoặc bộ tạo bit ngẫu nhiên tất định có thể được thiết kế để tạo khóa từ nguyên liệu mầm. Việc sử dụng khóa được cung cấp từ ngoài vào có thể thích hợp, ví dụ như trong các ứng dụng có rủi ro thấp với các ràng buộc về bộ nhớ (ví dụ: thẻ smartcard), khi việc tạo khóa từ nguyên liệu mầm là không khả thi (ví dụ: nguồn bất định đầy đủ quá tốn kém) hoặc chất lượng của nguồn bất định trong bộ tạo bit ngẫu nhiên tất định vẫn còn là một vấn đề, trong khi đó khóa chất lượng tốt có thể có được từ bên ngoài ranh giới bộ tạo bit ngẫu nhiên tất định.

Khóa và cách sử dụng nó trong bộ tạo bit ngẫu nhiên tất định như sau:

1. Sử dụng khóa: Khóa phải được sử dụng như quy định đối với một bộ tạo bit ngẫu nhiên tất định cụ thể. Bộ tạo bit ngẫu nhiên tất định yêu cầu khóa không được tạo đầu ra cho đến khi khóa sẵn sàng.
2. Độ bất định của khóa. Độ bất định khi kết hợp các khóa ít nhất phải bằng độ an toàn mà ứng dụng yêu cầu. Ví dụ, khi độ an toàn yêu cầu của ứng dụng là 112 bit, thì khóa phải có độ bất định ít nhất là 112 bit.
3. Kích thước khóa: Kích thước khóa được lựa chọn nhằm hỗ trợ độ an toàn mong đợi trong ứng dụng.
4. Xác định khóa từ mầm: Một khóa được xác định từ mầm phải độc lập với phần còn lại của đầu vào khởi tạo được xác định bởi mầm đó. Nếu nhiều khóa được sử dụng trong bộ tạo bit ngẫu nhiên tất định, trái ngược với việc sử dụng cùng một khóa ở nhiều nơi, thì từng khóa phải độc lập với tất cả các khóa còn lại.
   a) Đối với bộ tạo bit ngẫu nhiên tất định xác định khóa từ mầm giống nhau là một giá trị khởi tạo và khóa khác (tức là một mầm đơn được sử dụng để xác định tất cả các đầu vào khởi tạo của bộ tạo bit ngẫu nhiên tất định, bao gồm khóa), mầm phải có độ bất định bằng hoặc lớn hơn độ an toàn mà ứng dụng yêu cầu.
   b) Đối với bộ tạo bit ngẫu nhiên tất định sử dụng nhiều mầm khóa để xác định một trường hợp của bộ tạo bit ngẫu nhiên tất định, từng mầm phải được sử dụng để xác định phần khác nhau của đầu vào khởi tạo (ví dụ: giá trị khởi tạo cho bộ tạo bit ngẫu nhiên tất định và từng khóa riêng biệt phải được xác định từ các mầm khác nhau). Việc kết hợp tất cả các mầ m phải có độ bất định bằng hoặc lớn hơn độ an toàn mà ứng dụng yêu cầu.
5. Khóa được cung cấp từ một nguồn bên ngoài. Khóa được tạo ra từ bên ngoài phải có độ bất định đầy đủ (tức là mỗi bit khóa phải độc lập với mọi bit khóa khác) và được tạo ra bằng cách sử dụng một bộ tạo bit ngẫu nhiên bất định hoặc một bộ tạo bit ngẫu nhiên tất định (hoặc một chuỗi bộ tạo bit ngẫu nhiên tất định) lấy mầm từ một bộ tạo bit ngẫu nhiên bất định.
6. Thay khóa: Thay khóa (tức là thay khóa này bằng một khóa mới) là phương tiện để khôi phục tính bí mật của đầu ra bộ tạo bit ngẫu nhiên tất định nếu khóa bị lộ. Việc thay khóa định kỳ là một biện pháp đối phó tốt với các mối đe dọa tiềm ẩn gây tổn hại đến khóa và đầu ra của bộ tạo bit ngẫu nhiên tất định. Tuy nhiên, kết quả từ việc thay khóa chỉ giống như sử dụng bộ tạo bit ngẫu nhiên bất định (hoặc chuỗi các bộ tạo bit ngẫu nhiên tất định được khởi tạo bởi một bộ tạo bit ngẫu nhiên bất định) để tạo ra khóa mới. Trong một số quá trình thực thi (ví dụ: thẻ smartcard), không thể thực hiện quá trình thay khóa đầy đủ và việc thay khóa có thể làm giảm độ an toàn. Trong những

trường hợp như vật, biện pháp tốt nhất là thay thế bộ tạo bit ngẫu nhiên tất định, có được khóa mới (ví dụ: sử dụng thẻ smartcard mới).

a) Sử dụng một khóa cho trước để tạo quá nhiều đầu ra có thể cung cấp thông tin hữu ích giúp đoán thành công đầu ra sắp tới. Thay khóa định kỳ sẽ giảm rủi ro về an toàn, làm giảm khả năng thỏa hiệp dữ liệu đích cần được bảo vệ bởi các cơ chế mật mã sử dụng bộ tạo bit ngẫu nhiên tất định.

7. Vòng đời khóa: Khóa có một vòng đời hữu hạn nhất định. Khóa phải được cập nhật (thay thế) định kỳ. Phải hủy khóa hết hạn hoặc khóa mà giá trị cập nhật hoặc giá trị mới được dẫn xuất. Nếu khóa bị lộ (ví dụ: mầm bị tổn hại), thì các thực thể trái phép có thể xác định được đầu ra của bộ tạo bit ngẫu nhiên tất định.
8. Phân tách khóa: Một khóa được sử dụng bởi bộ tạo bit ngẫu nhiên tất định sẽ không được cố tình sử dụng cho bất kỳ mục đích nào khác ngoài việc tạo bit ngẫu nhiên. Các trường hợp khác nhau của bộ tạo bit ngẫu nhiên tất định phải sử dụng các khóa khác nhau.

Phụ lục A
(quy định)
Kết hợp các bộ tạo bit ngẫu nhiên

Việc kết hợp các bộ tạo bit ngẫu nhiên được cho phép trong tiêu chuẩn này, miễn là nó được kết hợp theo một cách chấp nhận được. Bộ tạo bit ngẫu nhiên kết hợp phải tuân theo mô hình "không kém hơn", tức là bộ tạo bit ngẫu nhiên kết hợp được đề xuất phải "không kém hơn" một bộ tạo bit ngẫu nhiên đã được phê duyệt (và về mặt lý thuyết nó còn phải tốt hơn). Đây chính là một cách để kết hợp các bộ tạo bit ngẫu nhiên.

Một bộ tạo bit ngẫu nhiên có thể được kết hợp với bộ tạo bit ngẫu nhiên khác nếu các bộ tạo bit ngẫu nhiên không tương quan với nhau. Ví dụ: sử dụng một mầm khác nhau và/hoặc sử dụng một thuật toán khác nhau. Nếu hai thuật toán khác nhau được kết hợp, điều này sẽ giải quyết mối quan tâm về phát hiện trong tương lai có tấn công mật mã lên một trong các thuật toán.

Một bộ tạo bit ngẫu nhiên đã được phê duyệt có thể kết hợp với một bộ tạo bit ngẫu nhiên không được phê duyệt trong tiêu chuẩn này nếu hai bộ tạo bit ngẫu nhiên không tương quan và bộ tạo bit ngẫu nhiên thứ hai giảm xuống một hằng số (có nghĩa là độ bất định bằng không) thì đầu ra của bộ tạo bit ngẫu nhiên kết hợp vẫn là đầu ra của bộ tạo bit ngẫu nhiên được phê duyệt. Ví dụ: nếu hai dòng đầu ra của bộ tạo bit ngẫu nhiên được kết hợp bằng cách sử dụng phép XOR thì điều kiện này được thỏa mãn.

Việc kết hợp một bộ tạo bit ngẫu nhiên được phê duyệt với một bộ tạo bit ngẫu nhiên không xác định có thể được thực hiện để cho phép đạt được lợi thế của bộ tạo bit ngẫu nhiên không xác định, nó được cho là có chất lượng tốt hơn các thành phần thay thế được phê duyệt nhưng không được quy định trong tiêu chuẩn này. Nó cũng có thể được thực hiện khi một bên yêu cầu sử dụng phương pháp này còn bên khác lại yêu cầu sử dụng phương pháp khác.

VÍ DỤ 1: Bộ tạo bit với nguồn bất định vật lý có thể sử dụng một bộ tạo bit ngẫu nhiên tất định được khởi tạo đúng quy định để trộn "bộ trữ" các bit thu được từ nguồn vật lý.

VÍ DỤ 2: Bộ tạo bit bao gồm một bộ tạo bit ngẫu nhiên bất định "hoàn chỉnh" cung cấp đầu vào cho một bộ tạo bit ngẫu nhiên tất định "hoàn chỉnh", không gây ảnh hưởng đến dòng bit từ bộ tạo bit ngẫu nhiên bất định đến bộ tạo bit ngẫu nhiên tất định, ngoại trừ việc kiểm tra hoạt động hoặc kiểm tra đầu ra.

Phụ lục B
(quy định)
Các phương pháp chuyển đổi

B.1 Tạo số ngẫu nhiên

B.1.1 Kỹ thuật tạo số ngẫu nhiên

Tiêu chuẩn này tập trung vào tạo các chuỗi bit ngẫu nhiên. Trong một số ứng dụng mật mã, chuỗi các số ngẫu nhiên được ký hiệu là $(a_0, a_1, a_2, \dots)$ trong đó:

1. Mỗi số nguyên a_i thỏa mãn $0 \le a_i \le r - 1$, đối với một số số nguyên dương r (phạm vi của các số ngẫu nhiên);
2. Phương trình $a_i = s$ với xác suất bằng $1/r$, với mọi $i \ge 0$ và s $(0 \le s \le r - 1)$; và
3. Mọi giá trị a_i độc lập xác suất trong tập các giá trị a_j $(j \ne i)$.

Trong phần này quy định bốn kỹ thuật tạo ra các dãy số ngẫu nhiên như vậy từ các chuỗi bit ngẫu nhiên.

Nếu phạm vi của số được yêu cầu là $0 \le a_i \le r - 1$ và $a \le a_i \le b$ thì một số ngẫu nhiên trong khoảng này có thể thu được bằng cách tính $a_i + a$ trong đó a_i là một số ngẫu nhiên trong khoảng $0 \le a_i \le b - a$.

B.1.2 Phương pháp loại bỏ đơn giản

Cho m là số nguyên dương duy nhất thỏa mãn $2^{m-1} \le r \le 2^m - 1$ (m được xác định duy nhất bằng cách chọn r). Phương pháp để tạo ra số ngẫu nhiên a như sau:

1. Sử dụng bộ tạo bit ngẫu nhiên để tạo ra một chuỗi m bit ngẫu nhiên $(b_0, b_1, \dots, b_{m-1})$.
2. Cho $c = \sum_{i=0}^{m-1} 2^i b_i$.
3. Nếu $c < r$ thì đặt $a = c$, ngược lại loại bỏ c và chuyển sang bước 1.

CHÚ THÍCH Tỷ số $r/2^m$ là thước đo tính hiệu quả của kỹ thuật, và tỷ số này phải luôn luôn thỏa mãn $0{,}5 \le r/2^m < 1$. Nếu $r/2^m$ gần bằng 1 thì phương pháp trên là đơn giản và hiệu quả. Tuy nhiên, nếu $r/2^m$ gần bằng 0,5, thì phương pháp trên là kém hiệu quả và phương pháp phức tạp hơn dưới dây được khuyến nghị sử dụng.

B.1.3 Phương pháp loại bỏ phức tạp

Lựa chọn một số nguyên dương nhỏ t và cho m là số nguyên dương duy nhất thỏa mãn $2^{m-1} \le r^t \le 2^m - 1$ (m được xác định duy nhất bằng cách chọn r và t). Phương pháp này tạo ra một chuỗi t số ngẫu nhiên $(a_0, a_1, \dots, a_{t-1})$ như sau:

1. Sử dụng bộ tạo bit ngẫu nhiên để tạo ra một chuỗi m bit ngẫu nhiên $(b_0, b_1, \dots, b_{m-1})$.
2. Cho $c = \sum_{i=0}^{m-1} 2^i b_i$.
3. Nếu $c < r^t$ thì cho $(a_0, a_1, \dots, a_{t-1})$ là chuỗi giá trị duy nhất thỏa mãn $0 \le a_i \le r - 1$ sao cho $c = \sum_{i=0}^{t-1} r^i a_i$. Ngược lại loại bỏ c và quay về bước 1.

CHÚ THÍCH 1 Tỷ số $r^t/2^m$ là thước đo tính hiệu quả của kỹ thuật, và tỷ số này phải luôn luôn thỏa mãn $0,5 \leq r^t/2^m < 1$. Do đó, cho trước r, cần lựa chọn t sao cho t là giá trị nhỏ nhất thỏa mãn $r^t/2^m$ gần bằng 1. Ví dụ, nếu $r = 3$, thì lựa chọn $t = 3$ có nghĩa là $m = 5$ và $r^t/2^m = 27/32 \approx 0,84$ và lựa chọn $t = 5$ có nghĩa là $m = 8$ và $r^t/2^m = 243/256 \approx 0,95$.

CHÚ THÍCH 2 Phương pháp loại bỏ phức tạp giống với phương pháp loại bỏ đơn giản khi $t = 1$.

B.1.4 Phương pháp mô-đun đơn giản

Cho m là số nguyên dương duy nhất thỏa mãn $2^{m-1} \leq r \leq 2^m - 1$ và cho l là tham số bí mật. Phương pháp để tạo ra số ngẫu nhiên a như sau:

1. Sử dụng bộ tạo bit ngẫu nhiên để tạo ra một chuỗi gồm $m + l$ bit ngẫu nhiên, $(b_0, b_1, \dots, b_{m+l-1})$.
2. Cho $c = \sum_{i=0}^{m+l-1} 2^i b_i$.
3. Cho $a = c \bmod r$.

CHÚ THÍCH Không giống như hai phương pháp trước, phương pháp mô-đun đơn giản được đảm bảo kết thúc sau mỗi lần thực thi. Xác suất để $a = s$ đối với mọi giá trị s cụ thể ($0 \leq s \leq r - 1$) không chính xác bằng $1/r$. Nhưng đối với một giá trị l đủ lớn, chênh lệch giữa xác suất để $a = s$ đối với mọi giá trị s cụ thể và $1/r$ là không đáng kể. Đề xuất sử dụng giá trị l = 64.

B.1.5 Phương pháp mô-đun phức tạp

Chọn một số nguyên dương nhỏ t, sau đó cho m là số nguyên dương duy nhất thỏa mãn $2^{m-1} \leq r^t \leq 2^m - 1$ (m được xác định duy nhất bằng cách chọn r và t). Cho l là tham số an toàn. Phương pháp này tạo ra một chuỗi t số ngẫu nhiên $(a_0, a_1, \dots, a_{t-1})$ như sau:

1. Sử dụng bộ tạo bit ngẫu nhiên tạo ra một chuỗi gồm gồm $m + l$ bit ngẫu nhiên, $(b_0, b_1, \dots, b_{m+l-1})$.
2. Cho $c = \sum_{i=0}^{m+l-1} 2^i b_i \bmod r^t$.

Cho $(a_0, a_1, \dots, a_{t-1})$ là chuỗi giá trị duy nhất thỏa mãn $0 \leq a_i \leq r - 1$ sao cho $c = \sum_{i=0}^{t-1} r^i a_i$.

CHÚ THÍCH 1 Giống như phương pháp mô-đun đơn giản, phương pháp mô-đun phức tạp không cung cấp các số có phân bố đều, nhưng chênh lệch giữa phân bố thực tế của các số và phân bố đều là không đáng kể với một tham số an toàn đủ lớn. Đề xuất sử dụng giá trị l = 64.

CHÚ THÍCH 2 Phương pháp mô-đun phức tạp giống với phương pháp mô-đun đơn giản với t = 1.

Phụ Lục C
(quy định)
Bộ tạo bit ngẫu nhiên tất định

C.1 Ví dụ về cơ chế bộ tạo bit ngẫu nhiên tất định

Phụ lục này chứa các ví dụ về cơ chế bộ tạo bit ngẫu nhiên tất định đáp ứng các yêu cầu của tiêu chuẩn này. Đây không phải là một danh sách đầy đủ. Các cơ chế khác là khả thi nếu chúng đáp ứng các yêu cầu được nêu trong phần chính của tiêu chuẩn này.

Các bộ tạo bit ngẫu nhiên tất định trong phụ lục này được đưa ra trong tựa mã.

CHÚ THÍCH Một số ví dụ tựa mã trong phần này cũng có trong [3].

C.2 Bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm

C.2.1 Giới thiệu về bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm

Bộ tạo bit ngẫu nhiên tất định băm dựa trên hàm băm không thể đảo ngược. Bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm được cung cấp dưới đây.

Hash_DRBG (...) quy định đã được thiết kế để sử dụng bất kỳ hàm băm mật mã ISO/IEC nào và có thể được sử dụng bởi các ứng dụng yêu cầu mức độ an toàn khác nhau, miễn là sử dụng hàm băm thích hợp và cung cấp đủ độ bất định cho mầm. Những hàm băm này được quy định trong ISO/IEC 10118-3.

C.2.2 Hash_DRBG

C.2.2.1 Thảo luận

Những giả định khi thiết kế bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm (Hash_DRBG) như sau:

1. Đầu ra của hàm băm xuất hiện ngẫu nhiên nếu đầu vào là khác nhau.
2. Mầm chứa một lượng độ bất định thích hợp dựa trên các bit an toàn mong đợi, lên đến một giá trị tối đa của độ dài bit đầu ra của hàm băm.

Hash_DRBG (...) cung cấp một hàm băm ISO/IEC tạo ra một khối các bit giả ngẫu nhiên sử dụng một mầm (*seed*).

Đầu vào bổ sung tùy chọn (*additional_input*) được cung cấp trong mỗi yêu cầu của **Hash_DRBG (...)**.

Hash_DRBG (...) được thiết kế nhằm đáp ứng độ mạnh an toàn khác nhau (xem Phụ lục F) phụ thuộc vào hàm băm đã sử dụng.

Độ dài mầm (*seedlen*) ít nhất phải bằng giá trị lớn nhất của kích thước khối đầu ra hàm băm (*outlen*) và độ mạnh an toàn.

Bộ tạo **Hash_DRBG (...)** yêu cầu sử dụng hàm băm tại một số điểm trong quá trình xử lý, bao gồm các quá trình khởi tạo và thay mầm mới. Hàm băm giống nhau phải được sử dụng liên tục. Hàm băm được sử dụng phải đáp ứng hoặc vượt quá độ mạnh an toàn mong đợi mà ứng dụng yêu cầu.

Bảng C.1 cung cấp một ví dụ về hàm băm mật mã được ISO/IEC đề xuất, minh họa độ mạnh an toàn, độ bất định tối thiểu yêu cầu và độ dài mầm.

Bảng C.1 – Bảng độ mạnh an toàn cho các hàm băm

Hàm băm	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Độ mạnh an toàn hỗ trợ	80, 112, 128	80, 112, 128, 192	80, 112, 128, 192, 256	80, 112, 128, 192, 256	80, 112, 128, 192, 256
Độ bất định tối thiểu yêu cầu	**max** (120, độ mạnh an toàn)				
Độ dài mầm (*seedlen*)	440	440	440	888	888
CHÚ THÍCH Mô tả về hàm **max**(...) có trong C.2.2.2.1.					

C.2.2 Mô tả

C.2.2.2.1 Giới thiệu chung

Quá trình khởi tạo và thay mầm mới của **Hash_DRBG (...)** bao gồm việc lấy một đầu vào bất định với tối thiểu lượng độ bất định mà ứng dụng yêu cầu. Đầu vào bất định được sử dụng để dẫn xuất mầm. Mầm được dùng để dẫn xuất các phần tử của trạng thái khởi tạo bao gồm:

1. Giá trị (V) được cập nhật trong mỗi lần gọi tới bộ tạo bit ngẫu nhiên tất định;
2. Hằng số (C) phụ thuộc vào mầm;
3. Bộ đếm (*reseed_counter*) chỉ ra số lượng yêu cầu các bit giả ngẫu nhiên vì *entropy_input* mới thu được trong quá trình khởi tạo hoặc thay mầm mới;
4. Độ mạnh an toàn trong quá trình khởi tạo bộ tạo bit ngẫu nhiên tất định;
5. Độ dài mầm (*seedlen*);
6. *prediction_resistance_flag* cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định yêu cầu hay không;
7. (Tùy chọn) Chuyển đổi đầu vào độ bất định sử dụng hàm một chiều để so sánh với đầu vào độ bất định mới khi bộ tạo bit ngẫu nhiên tất định được thay mầm mới; giá trị này phải được biểu diễn nếu bộ tạo bit ngẫu nhiên tất định được thay mầm mới; có thể bỏ qua nếu bộ tạo bit ngẫu nhiên tất định không được thay mầm mới.

Các biến được sử dụng trong mô tả **Hash_DRBG (...)** như sau:

additional_input Đầu vào bổ sung tùy chọn.

C Hằng số bit *seedlen* được tính toán trong quá trình khởi tạo và thay mầm mới.

data	Dữ liệu được băm.
entropy_input	Các bit chứa độ bất định được sử dụng để xác định *seed_material* và tạo mầm.
Get_entropy *(min_entropy, min_length, max_length)*	Hàm nhận được một xâu bit từ nguồn bất định. *min_entropy* chỉ ra số lượng độ bất định tối thiểu được cung cấp trong các bit trả về; *min_length* chỉ ra số bit tối thiểu trả về; *max_length* chỉ ra số bit tối đa trả về.
Hash *(a)*	Hoạt động băm trên dữ liệu *a* sử dụng một hàm băm ISO/IEC.
Hash_df *(seed_material, seedlen)*	Hàm dẫn xuất băm một xâu đầu vào và trả về số lượng các bit theo đặc tính của hàm băm.
i	Giá trị tạm thời được sử dụng làm bộ đếm vòng lặp.
m	Số lượng vòng lặp mà hàm băm cần để thu được số lượng các bit giả ngẫu nhiên theo yêu cầu.
$\boldsymbol{max}\,(A, B)$	Hàm trả về giá trị lớn hơn A hoặc B.
max_length	Độ dài tối đa của xâu trả về từ hàm ***Get_entropy (...).***
max_request_length	Số lượng các bit giả ngẫu nhiên tối đa được yêu cầu trong một lần gọi. Giá trị này được thực thi một cách phụ thuộc.
min_entropy	Lượng độ bất định tối thiểu thu được từ nguồn bất định và được cung cấp trong mầm.
min_length	Độ dài tối thiểu của *entropy_input*.
Null	Xâu rỗng.
outlen	Độ dài của khối đầu ra hàm băm
personalisation_string	Xâu thông tin cá nhân.
prediction_resistance_flag	Cờ cho biết yêu cầu về tính an toàn phía trước có được thực hiện hay không. *prediction_resistance_flag* = 1 = cho phép = Cho phép chống lại việc dự đoán trước, 0 = không cho phép = Không chống lại việc dự đoán trước.
prediction_resistance_request_flag	Chỉ ra tính an toàn phía trước có được yêu cầu trong một lần yêu cầu các bit giả ngẫu nhiên hay không. *prediction_resistance_request_flag* = 1 = Cung cấp khả năng chống lại việc dự đoán trước, 0 = Không cung cấp khả năng chống lại việc

	dự đoán trước.
pseudorandom_bits	Số lượng các bit giả ngẫu nhiên trả về từ hàm **Hash_DRBG (...)**.
requested_no_of_bits	Số lượng các bit giả ngẫu nhiên được tạo ra.
requested_strength	Độ mạnh an toàn liên quan đến các bit giả ngẫu nhiên được yêu cầu.
reseed_counter	Đếm số lượng các yêu cầu bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới.
reseed_interval	Số lượng các yêu cầu tối đa để tạo các bit giả ngẫu nhiên trước khi yêu cầu thay mầm mới.
seed	Xâu bit chứa độ bất định được sử dụng để xác định trạng thái bên trong của bộ tạo bit ngẫu nhiên tất định trong khi khởi tạo hoặc thay mầm mới.
seedlen	Độ dài mầm chứa độ bất định yêu cầu.
seed_material	Dữ liệu được sử dụng để tạo mầm.
state(state_handle)	Một mảng trạng thái cho các quá trình khởi tạo khác nhau của bộ tạo bit ngẫu nhiên tất định. Trạng thái được thực hiện giữa những lần gọi đến bộ tạo bit ngẫu nhiên tất định. Đối với **Hash_DRBG (...)**, trạng thái cho một lần khởi tạo được xác định bằng *state (state_handle) = {V, C, reseed_counter, strength, seedlen, prediction_resistance_flag}*. Một phần tử cụ thể của state được quy định bằng *state(state_handle).element*, ví dụ: *state(state_handle).V*.
state_handle	Xử lý đối với không gian trạng thái trong quá trình khởi tạo.
status	Trạng thái trả về từ một lần gọi hàm, trong đó *status* = "Thành công" hoặc một thông báo lỗi.
strength	Độ mạnh an toàn cho bộ tạo bit ngẫu nhiên tất định.
V	Giá trị được dẫn xuất từ mầm, nhưng giả sử các giá trị mới dựa trên đầu vào bổ sung tùy chọn (*additional_input*), các bit giả ngẫu nhiên được tạo ra bằng bộ tạo (*pseudorandom_bits*), hằng số (*C*) và bộ đếm vòng lặp (*reseed_counter*).
w, W	Các giá trị trung gian.

C.2.2.2.2 Khởi tạo Hash_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để khởi tạo **Hash_DRBG (...)**.

Cho **Hash (...)** là hàm băm ISO/IEC được sử dụng và *outlen* là độ dài đầu ra của hàm băm.

Instantiate_Hash_DRBG (...):

Đầu vào: số nguyên (*requested_strength, prediction_resistance_flag*), xâu *personalisation_string*.

Đầu ra: xâu *status*, số nguyên *state_handle*.

Quá trình:

1. If (requested_strength > độ mạnh an toàn tối đa được hàm băm cung cấp), then Return (Thông báo lỗi).
2. Thiết lập độ mạnh bằng một trong năm độ mạnh an toàn.

   If (*requested_strength* ≤ 80), then *strength* = 80

   Else if (*requested_strength* ≤ 112), then *strength* = 112

   Else if (*requested_strength* ≤ 128), then *strength* = 128

   Else if (*requested_strength* ≤ 192), then *strength* = 192

   Else *strength* = 256.
3. *min_entropy* = **max**(120, *strength*).
4. *min_length* = **max**(*outlen*, *strength*).
5. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).
6. If (*status* ≠ "Thành công"), then **Return** (Thông báo lỗi).
7. *seed_material* = *entropy_input* || *personalisation_string*.
8. *seed* = **Hash_df** (*seed_material*, *seedlen*).

CHÚ THÍCH 1 Bước này đảm bảo rằng độ bất định được phân bố thông qua mầm.

9. *V* = *seed*.
10. *C* = **Hash_df** ((0x00 || *V*), *seedlen*).

CHÚ THÍCH 2 Thêm vào trước V một byte 0.

11. *reseed_counter* = 1.
12. *state*(*state_handle*) = {*V*, *C*, *reseed_counter*, *strength*, *prediction_resistance_flag*}.
13. **Return** ("Thành công", *state_handle*).

Get_entropy (...):

Mô tả cụ thể về Get_entropy (...) được giao cho người thực thi. Một ví dụ cấp cao được cung cấp dưới đây.

Đầu vào: số nguyên (*min_entropy*, *min_length*, *max_length*).

Đầu ra: xâu (*status*, *entropy_input*).

Quá trình:

1. Thu được *entropy_input* với *độ bất định* ≥ *min_entropy* và với độ dài thích hợp từ nguồn tương ứng, trong đó *min_length* ≤ *độ dài* ≤ *max_length*.
2. **Return** ("Thành công", *entropy_input*).

Hash_df (...):

Các hàm dẫn xuất được sử dụng trong quá trình khởi tạo và thay mầm mới của bộ tạo bit ngẫu nhiên tất định dẫn xuất các giá trị trạng thái hoặc phân bố độ bất định thông qua một xâu bit. Hàm băm dữa trên hàm dẫn xuất băm một xâu đầu vào và trả về số lượng bit yêu cầu. Cho **Hash (...)** là hàm băm được bộ tạo bit ngẫu nhiên tất định sử dụng và *outlen* là độ dài đầu ra.

Đầu vào: xâu *input_string*, số nguyên *no_of_bits*.

Đầu ra: xâu bit *requested_bits*.

Quá trình:

1. $temp$ = xâu rỗng.
2. $len = \left\lceil \frac{no\_of\_bits}{outlen} \right\rceil$.
3. $counter$ = một giá trị nhị phân 8 bit được biểu diễn dưới dạng hexa là 0x01.
4. For *i* = 1 to *len* do

   4.1 *temp* = *temp* || **Hash** (*counter* || *no_of_bits* || *input_string*).

CHÚ THÍCH 3 no_of_bits được biểu diễn bằng một số nguyên 32 bit.

   4.2 *counter* = *counter* + 1.

5. *requested_bits* = *no_of_bits* ngoài cùng bên trái của *temp*.
6. **Return** (*requested_bits*).

CHÚ THÍCH 4 Nếu quá trình thực thi không xử lý tất cả năm độ mạnh an toàn, thì bước 2 của **Instantiate_Hash_DRBG(...)** phải được sửa đổi cho phù hợp.

CHÚ THÍCH 5 Nếu *personalisation_string* không được cung cấp, thì tham số *personalisation_string* trong đầu vào có thể bỏ qua và bước 7 **Instantiate_Hash_DRBG(...)** được sửa thành *seed_material* = *entropy_input*.

CHÚ THÍCH 6 Nếu quá trình thực thi không cần *prediction_resistance_flag* làm tham số gọi (tức là Hash_DRBG (...) trong C.2.2.2.4 hoặc luôn luôn hoặc không bao giờ nhận được độ bất định mới trong bước 5), thì *prediction_resistance_flag* trong các tham số gọi và trong trạng thái (xem bước 12 trong **Instantiate_Hash_DRBG(...)**) có thể được bỏ qua.

C.2.2.2.3 Thay mầm mới cho quá trình khởi tạo Hash_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để thay mầm mới cho **Hash_DRBG (...)**. Cho **Hash (...)** là hàm băm ISO/IEC được sử dụng và cho *outlen* là độ dài đầu ra của hàm băm này.

Reseed_Hash_DRBG_Instantiation (...):

Đầu vào: số nguyên *state_handle*, xâu *additional_input*.

Đầu ra: xâu *status*, trong đó *status* = "Thành công" hoặc một thông báo lỗi.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi).
2. Lấy các giá trị trạng thái phù hợp, ví dụ: *V* = *state*(*state_handle*).*V*, *strength* = *state*(*state_handle*).*strength*, *seedlen* = *state*(*state_handle*).*seedlen*.
3. *min_entropy* = **max** (120, *strength*).
4. *min_length* = *min_entropy*.
5. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).
6. If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).
7. *seed_material* = 0x01 || *V* || *entropy_input* || *additional_input*.
8. *seed* = **Hash_df** (*seed_material*, *seedlen*).

CHÚ THÍCH Bước này kết hợp entropy_input mới với một byte cố định, độ bất định biểu diễn trong V và với mọi additional_input được cung cấp; thì phân bố trên mầm.

9. *V* = *seed*.
10. *C* = **Hash_df** ((0x00 || *V*), *seedlen*).
11. Cập nhật các giá trị trạng thái thích hợp.
    11.1. *state*(*state_handle*).*V* = *V*.
    11.2. *state*(*state_handle*).*C* = *C*.
    11.3. *state*(*state_handle*).*reseed_counter* = 1.
12. **Return** ("Thành công").

C.2.2.2.4 Tạo các bit giả ngẫu nhiên sử dụng Hash_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để tạo ra các bit giả ngẫu nhiên. Cho **Hash (...)** là hàm băm ISO/IEC được sử dụng (ISO/IEC 10118-3) và cho *outlen* là độ dài đầu ra của hàm băm đó.

Hash_DRBG (...):

Đầu vào: số nguyên (*state_handle*, *requested_no_of_bits*, *requested_strength*, *prediction_resistance_request_flag*), xâu *additional_input*.

Đầu ra: xâu *status*, xâu bit *pseudorandom_bits*, trong đó *status* = "Thành công" hoặc một thông báo lỗi.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (Thông báo lỗi, *Null*).
2. Lấy các giá trị trạng thái thích hợp, ví dụ: *V* = *state*(*state_handle*).*V*, *C* = *state*(*state_handle*).*C*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*, *strength* = *state*(*state_handle*).*strength*, *seedlen* = *state*(*state_handle*).*seedlen*, *prediction_resistance_flag* = *state*(*state_handle*).*prediction_resistance_flag*.
3. If (*requested_no_of_bits* > *max_request_length*), then **Return** (Thông báo lỗi, *Null*).
4. If (*requested_strength* > *strength*), then **Return** (Thông báo lỗi, *Null*).
5. If ((*prediction_resistance_request_flag* = Cho phép chống lại việc dự đoán trước) and (*prediction_resistance_flag* = Không chống lại việc dự đoán trước)), then **Return** (Thông báo lỗi, *Null*).
6. If ((*reseed_counter* > *reseed_interval*) OR (*prediction_resistance_request_flag* = Provide_prediction_resistance)) then:

   6.1 If chưa sẵn sàng thay mầm, then **Return** (Thông báo lỗi, *Null*).

   6.2 *status* = **Reseed_Hash_DRBG_Instantiation** (*state_handle*, *additional_input*).

   6.3 If (*status* ≠ "Thành công"), then **Return** (*status*, *Null*).

   6.4 *V* = *state*(*state_handle*).*V*, *C* = *state*(*state_handle*).*C*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*.

   6.5 *additional_input* = *Null*.

7. If (*additional_input* ≠ *Null*), then do

   7.1 *w* = **Hash** (0x02 || *V* || *additional_input*).

   7.2 $V = (V + w) \bmod 2^{seedlen}$.

8. *pseudorandom_bits* = **Hashgen** (*requested_no_of_bits*, *V*).
9. *H* = **Hash** (0x03 || *V*).

10. $V = (V + H + C + reseed\_counter) \bmod 2^{seedlen}$.

11. $reseed\_counter = reseed\_counter + 1$.

12. Cập nhật các giá trị đã thay đổi trong trạng thái.

    12.1 $state(state\_handle).V = V$.

    12.2 $state(state\_handle).reseed\_counter = reseed\_counter$.

13. **Return** ("Thành công", *pseudorandom_bits*).

Hashgen (...):

Đầu vào: số nguyên *requested_no_of_bits*, xâu bit *V*.

Đầu ra: xâu bit *pseudorandom_bits*.

Quá trình:

1. $m = \left\lceil \frac{requested\_no\_of\_bits}{outlen} \right\rceil$.
2. *data* = *V*.
3. *W* = xâu rỗng.
4. For *i* = 1 to *m*
5. *pseudorandom_bits* = *requested_no_of_bits* ngoài cùng bên trái của *W*.
6. **Return** (*pseudorandom_bits*).

CHÚ THÍCH 1 Nếu phần thực thi không yêu cầu *additional_input*, thì tham số đầu vào *additional_input* và bước 6 của Hash_DRBG có thể được bỏ qua.

CHÚ THÍCH 2 Nếu phần thực thi không cần *prediction_resistance_flag*, thì những gì liên quan đến *prediction_resistance_flag* trong Hash_DRBG có thể được bỏ qua.

C.2.2.2.5 Chèn độ bất định bổ sung vào trạng thái của Hash_DRBG (...)

Độ bất định bổ sung có thể được chèn vào trạng thái của **Hash_DRBG (...)** bằng bốn cách. Độ bất định bổ sung có thể được chèn vào bằng:

1. Gọi hàm **Reseed_Hash_DRBG_Instantiation (...)** bất cứ khi nào. Hàm này luôn luôn gọi đến phần thực thi phụ thuộc hàm **Get_entropy (...)** để *min_entropy* = **max** (120, *strength*) bit mới của độ bất định được thêm vào trạng thái.
2. Sử dụng tính năng thay mầm mới tự động của bộ tạo bit ngẫu nhiên tất định. Nếu số lượng tối đa các cập nhật đối với trạng thái đạt được, thì bộ tạo bit ngẫu nhiên tất định sẽ gọi **Reseed_Hash_DRBG_Instantiation (...).**
3. Thiết lập *prediction_resistance_flag* thành Cho phép chống lại việc dự đoán trước khi khởi tạo. Nếu thiết lập thành công, mỗi lần gọi đến bộ tạo bit ngẫu nhiên tất định để tạo ra các bit giả ngẫu nhiên phải bao gồm yêu cầu về tính an toàn về phía trước, trong đó lần lượt gọi đến **Get_entropy (...)** trước khi tạo ra các bit giả ngẫu nhiên mới.

4. Cung cấp đầu vào bổ sung trong mỗi lần gọi đến bộ tạo bit ngẫu nhiên tất định để tạo các bit giả ngẫu nhiên.

CHÚ THÍCH Các lần gọi thường xuyên đến hàm **Get_entropy (...)** có thể làm giảm hiệu suất một cách nghiêm trọng.

C.2.3 HMAC_DRBG

C.2.3.1 Thảo luận

HMAC_DRBG sử dụng nhiều lần hàm băm có khóa được chấp nhận dựa trên bất kỳ hàm băm mật mã ISO/IEC nào có trong ISO/IEC 10118-3. Cơ chế bộ tạo bit ngẫu nhiên tất định này sử dụng hàm cập nhật được quy định trong C.2.3.2.2 và hàm HMAC với hàm cập nhật là hàm dẫn xuất trong quá trình khởi tạo và thay mầm mới. Hàm băm mật mã giống nhau được sử dụng trong suốt quá trình khởi tạo **HMAC_DRBG**. Hàm băm được sử dụng phải đáp ứng hoặc vượt quá các yêu cầu an toàn mà ứng dụng yêu cầu.

C.2.3.2 Mô tả

C.2.3.2.1 Giới thiệu chung

Quá trình khởi tạo và thay mầm mới của **HMAC_DRBG (...)** bao gồm việc thu được mầm với lượng bất định phù hợp. Đầu vào độ bất định được sử dụng để dẫn xuất mầm, sau đó được dùng để dẫn xuất các phần tử của trạng thái khởi tạo trong bộ tạo bit ngẫu nhiên tất định. Trạng thái bao gồm:

1. Giá trị *V* được cập nhật mỗi lần *outlen* bit đầu ra khác nhau được tạo ra (trong đó *outlen* là số lượng các bit đầu ra từ hàm băm mật mã);
2. Khóa *outlen* bit được cập nhật ít nhất mỗi lần bộ tạo bit ngẫu nhiên tất định tạo ra các bit giả ngẫu nhiên;
3. Bộ đếm (*reseed_counter*) chỉ ra số lượng yêu cầu các bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới;
4. Độ mạnh an toàn khi khởi tạo bộ tạo bit ngẫu nhiên tất định; và
5. *prediction_resistance_flag* cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định yêu cầu hay không.

Các biến được sử dụng trong mô tả **Hash_DRBG (...)** như sau:

additional_input	Đầu vào bổ sung tùy chọn.
data	Dữ liệu được băm.
entropy_input	Các bit chứa độ bất định được sử dụng để xác định *seed_material* và tạo mầm.
Get_entropy *(min_entropy, min_length, max_length)*	Hàm nhận được một xâu bit từ nguồn bất định. *min_entropy* chỉ ra số lượng độ bất định tối thiểu được cung cấp trong các bit trả về; *min_length* chỉ ra số bit tối thiểu trả về; *max_length* chỉ ra số bit tối đa trả về.
HMAC (*K*, *V*)	Hàm băm có khóa được quy định trong ISO/IEC 9797-2 sử dụng hàm băm mật mã được lựa chọn cho cơ chế bộ tạo bit ngẫu nhiên tất định trong đó *K* là khóa được sử dụng và *V* là khối đầu vào.

Key	Khóa được sử dụng để tạo ra các bit giả ngẫu nhiên
max (A, B)	Hàm trả về giá trị lớn hơn A hoặc B.
max_length	Độ dài tối đa của xâu trả về từ hàm **Get_entropy (...)**.
max_request_length	Số lượng các bit giả ngẫu nhiên tối đa được yêu cầu trong một lần gọi. Giá trị này được thực thi một cách phụ thuộc.
min_entropy	Lượng độ bất định tối thiểu thu được từ nguồn bất định và được cung cấp trong mầm.
min_length	Độ dài tối thiểu của *entropy_input*.
Null	Xâu rỗng.
outlen	Độ dài của khối đầu ra hàm băm
personalisation_string	Xâu thông tin cá nhân.
prediction_resistance_flag	Cờ cho biết yêu cầu về tính an toàn về phía trước có được thực hiện hay không. *prediction_resistance_flag* = 1 = cho phép = Cho phép chống lại việc dự đoán trước, 0 = không cho phép = Không chống lại việc dự đoán trước.
prediction_resistance_request_flag	Chỉ ra tính an toàn về phía trước có được yêu cầu trong một lần yêu cầu các bit giả ngẫu nhiên hay không. *prediction_resistance_request_flag* = 1 = Cung cấp khả năng chống lại việc dự đoán trước, 0 = Không cung cấp khả năng chống lại việc dự đoán trước.
pseudorandom_bits	Số lượng các bit giả ngẫu nhiên trả về từ hàm **HMAC_DRBG (...)**.
requested_no_of_bits	Số lượng các bit giả ngẫu nhiên được tạo ra.
requested_strength	Độ mạnh an toàn liên quan đến các bit giả ngẫu nhiên được yêu cầu.
reseed_counter	Đếm số lượng các yêu cầu bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới.
reseed_interval	Số lượng các yêu cầu tối đa để tạo các bit giả ngẫu nhiên trước khi yêu cầu thay mầm mới.
seed_material	Dữ liệu được sử dụng để tạo mầm.
state(state_handle)	Một mảng trạng thái cho các quá trình khởi tạo khác nhau của bộ tạo bit ngẫu nhiên tất định. Trạng thái được thực hiện giữa những lần gọi đến bộ tạo bit ngẫu nhiên tất định. Đối với **HMAC_DRBG (...)**, trạng

	thái cho một lần khởi tạo được xác định bằng *state (state_handle) = {V, Key, reseed_counter, strength, prediction_resistance_flag}*. Một phần tử cụ thể của state được quy định bằng *state(state_handle).element*, ví dụ: *state(state_handle).V*.
state_handle	Xử lý đối với không gian trạng thái trong quá trình khởi tạo.
status	Trạng thái trả về từ một lần gọi hàm, trong đó *status* = "Thành công" hoặc một thông báo lỗi.
strength	Độ mạnh an toàn cho bộ tạo bit ngẫu nhiên tất định.
temp	Giá trị trung gian
V	Giá trị trong trạng thái được cập nhật mỗi khi các bit giả ngẫu nhiên được tạo ra.

C.2.3.2.2 Hàm bên trong: Hàm cập nhật

Hàm **HMAC_DRBG_Update** cập nhật trạng thái bên trong của **HMAC_DRBG** sử dụng *provided_data*. CHÚ THÍCH rằng đối với cơ chế bộ tạo bit ngẫu nhiên tất định này, hàm **HMAC_DRBG_Update** cũng làm việc như một hàm dẫn xuất đối với các hàm khởi tạo và thay mầm mới.

Cho **HMAC** là hàm băm có khóa trong ISO/IEC 9797-2 sử dụng hàm băm mật mã được lựa chọn từ cơ chế bộ tạo bit ngẫu nhiên tất định từ bảng C.1 trong phụ lục này.

HMAC_DRBG_Update (...):

Đầu vào: xâu bit (*provided_data, K, V*).

Đầu ra: xâu bit *(K, V)*.

Quá trình:

1. *K*= **HMAC**(*K, V*|| 0x00 || *provided_data*).
2. *V*= **HMAC** (*K, V*).
3. If (*provided_data* = *Null*), then **Return** (*K, V*).
4. *K*= **HMAC** (*K, V*|| 0x01 || *provided_data*).
5. *V*= **HMAC** (*K, V*).
6. **Return** (*K, V*).

C.2.3.2.3 Khởi tạo HMAC_DRBG

Quá trình sau đây hoặc tương đương được sử dụng để khởi tạo **HMAC_DRBG (...)**.

Cho **HMAC (...)** là hàm băm có khóa ISO/IEC được sử dụng và *outlen* là độ dài đầu ra của hàm băm có khóa.

Instantiate_HMAC_DRBG (...):

Đầu vào: số nguyên (*requested_strength, prediction_resistance_flag*), xâu *personalisation_string*.

Đầu ra: xâu *status*, số nguyên *state_handle*.

Quá trình:

1. If (*requested_strength* > độ mạnh an toàn tối đa được thuật toán MAC cung cấp), then **Return** (Thông báo lỗi).
2. Thiết lập độ mạnh bằng một trong năm độ mạnh an toàn.

   If (*requested_strength* ≤ 80), then *strength* = 80

   Else if (*requested_strength* ≤ 112), then *strength* = 112

   Else if (*requested_strength* ≤ 128), then *strength* = 128

   Else if (*requested_strength* ≤ 192), then *strength* = 192

   Else *strength* = 256.
3. *min_entropy* = **max**(120, *strength*).
4. *min_length* = **max**(*outlen*, *strength*).
5. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).
6. If (*status* ≠ "Thành công"), then **Return** (Thông báo lỗi).
7. *seed_material* = *entropy_input* || *personalisation_string*.
8. *Key* = 0x00 00...00.
9. *V* = 0x01 01...01.
10. *(Key, V)* = **HMAC_DRBG_Update** *(seed_material, Key, V)*.
11. *reseed_counter* = 1.
12. *state*(*state_handle*) = {*V*, *Key*, *reseed_counter*, *strength*, *prediction_resistance_flag*}.
13. **Return** ("Thành công", *state_handle*).

C.2.3.2.4 Thay mầm mới cho quá trình khởi tạo HMAC_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để thay mầm mới cho **HMAC_DRBG (...)**, sau khi nó được khởi tạo.

Reseed_HMAC_DRBG_Instantiation (...):

Đầu vào: số nguyên *state_handle*, xâu *additional_input*.

Đầu ra: xâu *status*, trong đó *status* = "Thành công" hoặc một thông báo lỗi.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi).
2. Lấy các giá trị trạng thái phù hợp, ví dụ: *V* = *state*(*state_handle*).*V*, *Key*= *state(state_handle).Key*, *strength* = *state*(*state_handle*).*strength*.
3. *min_entropy* = **max** (120, *strength*).
4. *min_length* = *min_entropy*.
5. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).
6. If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).
7. *seed_material* = *entropy_input* || *additional_input*.
8. *(Key, V) = HMAC_DRBG_Update (seed_material, Key, V).*
9. Cập nhật các giá trị trạng thái thích hợp.

   9.1. *state*(*state_handle*).*V* = *V*.

   9.2. *state*(*state_handle*).*Key* = *Key*.

   9.3. *state*(*state_handle*).*reseed_counter* = 1.
10. **Return** ("Thành công").

C.2.3.2.5 Tạo các bit giả ngẫu nhiên sử dụng HMAC_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để tạo các bit giả ngẫu nhiên.

HMAC_DRBG (...):

Đầu vào: số nguyên (*state_handle*, *requested_no_of_bits*, *requested_strength*, *prediction_resistance_request_flag*), xâu *additional_input*.

Đầu ra: xâu *status*, xâu bit *pseudorandom_bits*.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (Thông báo lỗi, *Null*).
2. Lấy các giá trị trạng thái thích hợp, ví dụ: *V* = *state*(*state_handle*).*V*, *Key* = *state*(*state_handle*).*Key*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*, *strength* = *state*(*state_handle*).*strength*, *prediction_resistance_flag* = *state*(*state_handle*).*prediction_resistance_flag*.
3. If (*requested_no_of_bits* > *max_request_length*), then **Return** (Thông báo lỗi, *Null*).
4. If (*requested_strength* > *strength*), then **Return** (Thông báo lỗi, *Null*).

5. *temp* = **len** (*additional_input*).
6. If (*temp* > *max_length*), then **Return** (Thông báo lỗi, *Null*).
7. If (*requested_no_of_bits* > *max_request_length*), then **Return** (Thông báo lỗi, *Null*).
8. If ((*prediction_resistance_request_flag* = Cung cấp khả năng chống lại việc dự đoán trước) and (*prediction_resistance_flag* = Không chống lại việc dự đoán trước)), then **Return** (Thông báo lỗi, *Null*).

CHÚ THÍCH 1 Nếu quá trình thực thi không cần *prediction_resistance_flag*, thì có thể bỏ qua *prediction_resistance_flag* trong tham số đầu vào và bỏ qua bước 8.

9. If ((*reseed_counter* > *reseed_interval*) OR (*prediction_resistance_request_flag* = Cung cấp khả năng chống lại việc dự đoán trước)) then:

   9.1 If chưa sẵn sàng thay mầm, then **Return** (Thông báo lỗi, *Null*).

   9.2 *status* = **Reseed_HMAC_DRBG_Instantiation** (*state_handle*, *additional_input*).

   CHÚ THÍCH 2 Nếu quá trình thực thi không cung cấp *additional_input*, thì chuỗi Null sẽ thay thế cho *additional_input* trong bước 9.2.

   9.3 If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi, *Null*).

   9.4 *V* = *state*(*state_handle*).*V*, *Key* = *state*(*state_handle*).*Key*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*.

   9.5 *additional_input* = *Null*.

10. If (*additional_input* ≠ *Null*), then:

    10.1 *(Key, V)* = **HMAC_DRBG_Update***(additional_input, Key, V).*

11. *temp* = *Null*.
12. While (**len** (*temp*) < *requested_number_of_bits*) do:

    12.1 *V*= **HMAC** (*Key, V*).

    12.2 *temp* = *temp* || *V*.

13. *pseudorandom_bits* = *requested_no_of_bits* ngoài cùng bên trái của *temp*.
14. *(Key, V)* = **HMAC_DRBG_Update** *(additional_input, Key, V).*
15. *reseed_counter* = *reseed_counter* + 1.
16. Cập nhật các giá trị đã thay đổi trong trạng thái.

    16.1 *state*(*state_handle*).*Key* = *Key*.

16.2 *state(state_handle).V = V.*

12.2 *state(state_handle).reseed_counter = reseed_counter.*

17. **Return** ("Thành công", *pseudorandom_bits*).

C.3 Bộ tạo bit ngẫu nhiên tất định dựa trên mã khối

C.3.1 Giới thiệu về bộ tạo bit ngẫu nhiên tất định dựa trên mã khối

Bộ tạo bit ngẫu nhiên tất định mã khối dựa trên một thuật toán mã khối.

Các bộ tạo bit ngẫu nhiên tất định mã khối trong tiêu chuẩn này được thiết kế để sử dụng bất kỳ thuật toán mã khối ISO/IEC đã phê duyệt và được sử dụng bởi các ứng dụng yêu cầu một số mức độ an toàn khác nhau. Các thuật toán mã khối này được quy định trong ISO/IEC 18033-3. Bộ tạo bit ngẫu nhiên tất định dựa trên các thuật toán mã khối được cung cấp sau đây:

1. **CTR_DRBG (...)** được quy định trong C.3.2.

2. **OFB_DRBG (...)** được quy định trong C.3.3.

Bảng C.2 cung cấp một ví dụ về một mã khối ISO/IEC đã phê duyệt, minh hoạt độ mạnh an toàn, các yêu cầu về độ bất định và mầm được sử dụng.

Bảng C.2 – Độ mạnh an toàn, các yêu cầu độ bất định và độ dài mầm đối với mã khối AES-128, 192 và 256

Thuật toán mã khối	Độ mạnh an toàn	Độ bất định tối thiểu yêu cầu	Độ dài mầm (tính bằng bit)
AES-128	80, 112, 128	128	256
AES-192	80, 112, 128, 192	192	320
AES-256	80, 112, 128, 192, 256	256	384

C.3.2 CTR_DRBG

C.3.2.1 Thảo luận

CTR_DRBG (...) sử dụng một thuật toán mã khối được phê duyệt ở chế độ bộ đếm và được quy định trong ISO/IEC 10116. Thuật toán mã khối và độ dài khóa giống nhau được sử dụng cho tất cả mọi hoạt động của mã khối. Thuật toán mã khối và kích thước khóa phải đáp ứng hoặc vượt quá các yêu cầu an toàn mà ứng dụng yêu cầu.

C.3.2.2 Mô tả

C.3.2.2.1 Giới thiệu chung

Quá trình khởi tạo và thay mầm mới của **CTR_DRBG (...)** bao gồm việc thu được mầm với lượng bất định phù hợp. Đầu vào độ bất định được sử dụng để dẫn xuất mầm, sau đó được dùng để dẫn xuất các phần tử của trạng thái khởi tạo trong bộ tạo bit ngẫu nhiên tất định. Trạng thái bao gồm:

1. Giá trị *V* được cập nhật mỗi lần *outlen* bit đầu ra khác nhau được tạo ra (trong đó *outlen* là số lượng các bit đầu ra từ thuật toán mã khối cơ bản);
2. Khóa được cập nhật mỗi lần số tiền xác định của các khối đầu ra được tạo ra;
3. Độ dài khóa (*keylen*) được sử dụng trong thuật toán mã khối;
4. Độ mạnh an toàn khi khởi tạo bộ tạo bit ngẫu nhiên tất định;
5. Bộ đếm (*reseed_counter*) bao gồm số lượng các yêu cầu đối với bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới; và
6. *prediction_resistance_flag* cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định yêu cầu hay không.

Các biến được sử dụng trong mô tả **CTR_DRBG (...)** như sau:

additional_input	Đầu vào bổ sung tùy chọn, có độ dài bit ≤ *max_length.*
Block_Cipher *(Key, V)*	Thuật toán mã khối, trong đó *Key* là khóa được sử dụng và *V* là khối đầu vào.
Block_Cipher_df *(a, b)*	Hàm dẫn xuất mã khối.
blocklen	Độ dài khối đầu ra của thuật toán mã khối.
entropy_input	Các bit chứa độ bất định được sử dụng để xác định *seed_material* và tạo mầm.
Get_entropy *(min_entropy, min_length, max_length)*	Hàm nhận được một xâu bit từ nguồn bất định. *min_entropy* chỉ ra số lượng độ bất định tối thiểu được cung cấp trong các bit trả về; *min_length* chỉ ra số bit tối thiểu trả về; *max_length* chỉ ra số bit tối đa trả về.
Key	Khóa được sử dụng để tạo ra các bit giả ngẫu nhiên.
keylen	Độ dài khóa cho thuật toán mã khối.
len *(x)*	Hàm trả về số lượng các bit trong xâu đầu vào x.
max_length	Độ dài tối đa của xâu thu thập độ bất định. Khi sử dụng hàm dẫn xuất, giá trị này được thực thi một cách phụ thuộc, nhưng phải ≤ 2^{35} bit (đối với mã khối 128 bit, có tối đa 2^{28} khối, tức là 2^{32} byte - 2^{35} bit). Khi không sử dụng hàm dẫn xuất, thì $max\_length = seedlen$.
max_request_length	Số lượng các bit giả ngẫu nhiên tối đa được yêu cầu trong một lần gọi; giá trị này được thực thi một cách phụ thuộc nhưng phải ≤ 2^{35} đối với mã khối 128 bit và ≤ 2^{19} bit đối với mã khối 64 bit.
min_entropy	Lượng độ bất định tối thiểu thu được từ nguồn bất định và được

	cung cấp trong mầm.
Null	Xâu rỗng.
personalisation_string	Xâu thông tin cá nhân.
prediction_resistance_flag	Cờ cho biết yêu cầu về tính an toàn về phía trước có được thực hiện hay không. *prediction_resistance_flag* = 1 = cho phép = Cho phép chống lại việc dự đoán trước, 0 = không cho phép = Không chống lại việc dự đoán trước.
prediction_resistance_request_flag	Chỉ ra tính an toàn về phía trước có được yêu cầu trong một lần yêu cầu các bit giả ngẫu nhiên hay không; *prediction_resistance_request_flag* = 1 = Cung cấp khả năng chống lại việc dự đoán trước, 0 = Không cung cấp khả năng chống lại việc dự đoán trước.
pseudorandom_bits	Các bit giả ngẫu nhiên được tạo ra trong một lần gọi đến **CTR_DRBG (...)**.
requested_no_of_bits	Số lượng các bit giả ngẫu nhiên trả về từ hàm **CTR_DRBG (...)**.
requested_strength	Độ mạnh an toàn liên quan đến các bit giả ngẫu nhiên được yêu cầu.
reseed_counter	Đếm số lượng các yêu cầu bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới.
reseed_interval	Số lượng các yêu cầu tối đa để tạo các bit giả ngẫu nhiên trước khi yêu cầu thay mầm mới. Giá trị tối đa phải $\leq 2^{32}$ đối với mã khối 128 bit và $\leq 2^{16}$ đối với mã khối 64 bit.
seedlen	Độ dài mầm, trong đó *seedlen* = *blocklen* + *keylen*.
seed_material	Dữ liệu được sử dụng để tạo mầm.
state(state_handle)	Một mảng trạng thái cho các quá trình khởi tạo khác nhau của bộ tạo bit ngẫu nhiên tất định. Trạng thái được thực hiện giữa những lần gọi đến bộ tạo bit ngẫu nhiên tất định. Đối với **CTR_DRBG (...)**, trạng thái cho một lần khởi tạo được xác định bằng *state (state_handle)* = *{V, Key, keylen, strength, reseed_counter, prediction_resistance_flag}*. Một phần tử cụ thể của state được quy định bằng *state(state_handle).element*, ví dụ: *state(state_handle).V*.
state_handle	Xử lý đối với không gian trạng thái trong quá trình khởi tạo.
status	Trạng thái trả về từ một lần gọi hàm, trong đó *status* = "Thành công" hoặc một thông báo lỗi.

strength	Độ mạnh an toàn cho bộ tạo bit ngẫu nhiên tất định.
temp	Giá trị trung gian
V	Giá trị trong trạng thái được cập nhật mỗi khi các bit giả ngẫu nhiên được tạo ra.

C.3.2.2.2 Khởi tạo CTR_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để khởi tạo **CTR_DRBG (...)**.

Instantiate_CTR_DRBG (...):

Đầu vào: số nguyên (*requested_strength, prediction_resistance_flag*), xâu *personalisation_string*.

Đầu ra: xâu *status*, số nguyên *state_handle*.

Quá trình:

1. If (*requested_strength* > độ mạnh an toàn tối đa được thuật toán mã khối cung cấp), then **Return** (Thông báo lỗi).
2. If (*requested_strength* ≤ 80), then (*strength* = 80; *keylen* = 128)

   Else if (*requested_strength* ≤ 112), then (*strength* = 112; *keylen* = 128)

   Else if (*requested_strength* ≤ 128), then (*strength* = 128; *keylen* = 128)

   Else if (*requested_strength* ≤ 192), then (*strength* = 192; *keylen* = 192)

   Else (*strength* = 256; *keylen* = 256).

CHÚ THÍCH 1 Bước này thiết lập strength bằng một trong năm độ mạnh an toàn và xác định độ dài khóa.

3. *seedlen* = *blocklen* + *keylen*.
4. *temp* = **len** (*personalisation_string*).
5. If (*temp* > *max_length*), then **Return** (Thông báo lỗi).

CHÚ THÍCH 2 Nếu không cung cấp *personalisation_string* thì có thể bỏ qua *personalisation_string* trong tham số đầu vào và bước 4 và 5.

6. Đoạn mã sau được sử dụng khi hàm dẫn xuất sẵn sang (nguồn đủ độ bất định có thể sẵn sàng hoặc chưa sẵn sàng).

   6.1 *min_entropy* = *strength* + 64.

   6.2 (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).

   6.3 If (*status* ≠ "Thành công"), then **Return** (Thông báo lỗi).

6.4 *seed_material* = *entropy_input* || *personalisation_string*.

CHÚ THÍCH 3 Nếu personalisation_string không được cung cấp trong bước 6.4 thì bước 6.4 có thể bỏ qua và bước 6.5 chuyển thành seed_material = Block_Cipher_df (entropy_input, seedlen).

6.5 *seed_material* = **Block_Cipher_df** (*seed_material, seedlen*).

7. Đoạn mã sau đây được sử dụng khi nguồn độ bất định đầy đủ đã sẵn sàng và hàm dẫn xuất không được sử dụng.

7.1 (*status, entropy_input*) = **Get_entropy** (*seedlen, seedlen, seedlen*).

7.2 If (status ≠ "Thành công"), then **Return** (Thông báo lỗi).

7.3 If (*temp* < *seedlen*), then *personalisation_string* = *personalisation_string* || $0^{seedlen - temp}$.

CHÚ THÍCH 4 Nếu *personalisation_string* quá ngắn thì phải được đệm thêm 0

7.4 *seed_material* = *entropy_input* ⊕ *personalisation_string*.

CHÚ THÍCH 5 Nếu *personalisation_string* không được cung cấp ở trên thì các bước 7.3 và 7.4 được bỏ qua và bước 7.1 trở thành: (*status, seed_material*) = **Get_entropy** (*seedlen, seedlen, seedlen*).

8. *Key* = 0^{keylen}.

9. *V* = $0^{blocklen}$.

10. *(Key, V)* = **Update** *(seed_material, keylen, Key, V)*.

11. *reseed_counter* = 1.

12. *state*(*state_handle*) = {*V*, *Key*, *keylen*, *strength*, *reseed_counter*, *prediction_resistance_flag*}.

CHÚ THÍCH 6 Nếu quá trình thực thi không cần *prediction_resistance_flag* làm tham số gọi (tức là CTR_DRBG (...) trong C.3.2.2.7 hoặc luôn luôn hoặc không bao giờ yêu cầu độ bất định mới trong bước 9), thì *prediction_resistance_flag* trong các tham số gọi và trong *state* (xem bước 12) được bỏ qua.

13. **Return** ("Thành công", *state_handle*).

C.3.2.2.3 Hàm bên trong: Hàm cập nhật

Hàm **Update (...)** cập nhật trạng thái bên trong của **CTR_DRBG (...)** sử dụng *seed_material*, phải có độ dài bit bằng *seedlen*. Quá trình sau đây hoặc tương đương được sử dụng trong hàm **Update (...)**.

Update (...):

Đầu vào: số nguyên *keylen*, xâu bit (*seed_material, Key, V*).

Đầu ra: xâu bit *(Key, V)*.

Quá trình:

1. *seedlen* = *blocklen* + *keylen*.

2. *temp* = *Null*.

3. While (**len** (*temp*) < *seedlen*) do:

   3.1 $V = (V + 1) \bmod 2^{blocklen}$.

   3.2 *output_block* = **Block_Cipher** (*Key, V*).

   3.3 *temp* = *temp* || *output_block*.

4. *temp* = *seedlen* bit ngoài cùng bên trái của *temp*.

5. *temp* = *temp* ⊕ *seed_material*.

6. *Key* = *keylen* bit ngoài cùng bên trái của *temp*.

7. *V* = *blocklen* bit ngoài cùng bên phải của *temp*.

8. **Return** (*Key, V*).

C.3.2.2.4 Hàm dẫn xuất sử dụng một thuật toán mã khối

Cho **CBC_MAC** là hàm được quy định trong C.3.2.2.5. Cho **Block_Cipher** là một hoạt động mã hóa ở chế độ ECB sử dụng thuật toán mã khối đã lựa chọn. Cho *outlen* là khối đầu ra của nó, cho *keylen* là độ dài khóa.

Đầu vào:

1. Xâu bit *input_string*: Xâu được sử dụng.

2. Số nguyên *no_of_bits*: Số lượng bit trả về bởi **Block_Cipher_df**.

Đầu ra: Xâu bit *requested_bits*: kết quả của việc thực hiện **Block_Cipher_df**.

Quá trình:

1. *L* = **len** (*input_string*) / 8.

CHÚ THÍCH 1 *L* là xâu bit biểu diễn số nguyên là kết quả của **len** (*input_string*) / 8. *L* được biểu diễn bằng một số nguyên 32 bit.

2. *N* = *no_of_bits* / 8.

CHÚ THÍCH 2 *N* là xâu bit biểu diễn số nguyên là kết quả của *no_of_bits* / 8. *N* được biểu diễn bằng một số nguyên 32 bit.

3. *S* = *L* || *N* || *input_string* || 0x80.

CHÚ THÍCH 3 Bước này thêm độ dài xâu và độ dài đầu ra yêu cầu vào *input_string*. Nếu cần thi đệm 0 vào *S*.

4. While (**len** (*S*) mod *outlen* ≠ 0) *S* = *S* || 0x00.

5. *temp* = xâu rỗng.

6. *i* = 0.

CHÚ THÍCH 4 *i* được biểu diễn bởi một số nguyên 32 bit, tức là **len**(*i*) = 32.

7. *K* = *keylen* bit ngoài cùng bên trái của 0x00010203 ... 1F.

8. While (**len** (*temp*) < *keylen* + *outlen*) do:

   8.1 $IV = i \,||\, 0^{outlen - len\,(i)}$.

   CHÚ THÍCH 5 Bước này đệm thêm biểu diễn số nguyên của *i* và đệm thêm 0 để đạt *outlen* bit.

   8.2 *temp* = *temp* || **CBC_MAC** (*K*,(*IV* || *S*)).

   8.3 *i* =*i* + 1.

9. *K* = *keylen* bit ngoài cùng bên trái của *temp*.

10. *X* = *outlen* bit tiếp theo của *temp*.

11. *temp* = xâu rỗng.

12. While (**len** (*temp*) < *no_of_bits*) do:

    12.1 *X* = **Block_Cipher** (*K*, *X*).

    12.2 *temp* = *temp* || *X*.

13. *requested_bits* = *no_of_bits* ngoài cùng bên trái của *temp*.

14. **Return** (*requested_bits*).

C.3.2.2.5 Hàm CBC_MAC

Hàm CBC_MAC là một phương pháp để tính toán mã xác thực thông báo. Cho Block_Cipher là hoạt động mã hóa ở chế độ ECB sử dụng thuật toán mã khối đã lựa chọn. Cho *outlen* là khối đầu ra của nó, cho *keylen* là độ dài khóa.

Quá trình sau đây hoặc tương đương được sử dụng để dẫn xuất số lượng bit yêu cầu.

Đầu vào:

1. Xâu bit *Key*: Khóa được sử dụng cho hoạt động mã khối.

2. Xâu bit *data_to_MAC*: Dữ liệu được sử dụng.

Đầu ra: Xâu bit *output_block*: kết quả được trả về từ hoạt động **CBC_MAC**.

Quá trình:

1. $chaining\_value = 0^{outlen}$.

CHÚ THÍCH Bước này thiết lập giá trị chuỗi móc xích đầu tiên bằng *outlen* số 0.

2. *n* = **len** *(data_to_MAC) / outlen.*
3. *Chia data_to_MAC thành n khối có độ dài outlen được biểu diễn từ* $block_1$ *đến* $block_n$.
4. For *i* = 1 to *n* do:

   4.1 *input_block = chaining_value* $\oplus$ $block_i$.

   4.2 *chaining_value* = **Block_Cipher** (*Key, input_block*).
5. *output_block = chaining_value.*
6. **Return** *output_block.*

C.3.2.2.6 Thay mầm mới khi khởi tạo CTR_DRBG(...)

Quá trình sau đây hoặc tương đương được sử dụng để thay mầm mới ngầm định cho quá trình **CTR_DRBG (...).**

Reseed_CTR_DRBG_Instantiation (...):

Đầu vào: số nguyên *state_handle*, xâu *additional_input.*

Đầu ra: xâu *status.*

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi).
2. Lấy các giá trị trạng thái phù hợp, ví dụ: *V = state(state_handle).V, Key= state(state_handle).Key, keylen = state(state_handle).keylen, strength = state(state_handle).strength, prediction_resistance_flag = state(state_handle).prediction_resistance_flag.*
3. *seedlen = blocklen + keylen.*
4. *temp* = **len** (*additional_input*).

CHÚ THÍCH 1 Nếu quá trình thực thi không xử lý *additional_input*, thì tham số *additional_input* trong đầu vào có thể bỏ qua trong bước 4 và 5 dưới đây.

5. If (*temp > max_length*), then **Return** (thông báo lỗi).
6. Đoạn mã sau đây được sử dụng khi hàm dẫn xuất sẵn sàng (nguồn độ bất định đầy đủ có thể sẵn sàng hoặc chưa sẵn sàng).

   6.1 min_entropy = strength + 64.

   6.2 (*status, entropy_input*) = **Get_entropy** (*min_entropy, min_entropy, max_length*).

6.3 If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).

6.4 *seed_material* = *entropy_input* || *additional_input*.

CHÚ THÍCH 2 Nếu quá trình thự thi không xử lý *additional_input,* thì bước 6.4 có thể bỏ qua và bước 6.5 chuyển thành: *seed_material* = **Block_Cipher_df** (*entropy_input*, *seedlen*).

6.5 *seed_material* = **Block_Cipher_df** (*seed_material*, *seedlen*).

7. Đoạn mã sau đây được sử dụng khi nguồn độ bất định đầy đủ sẵn sàng và hàm dẫn xuất không sử dụng.

7.1 *(status, entropy_input)* = **Get_entropy** *(seedlen, seedlen, seedlen).*

7.2 If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).

7.3 If (*temp* < *seedlen*), then *additional_input* = *additional_input* || $0^{seedlen - temp}$.

CHÚ THÍCH 3 Bước này đệm với 0 nếu *additional_input* quá ngắn.

7.4 *seed_material* = *entropy_input* ⊕ *additional_input*.

CHÚ THÍCH 4 Nếu quá trình thực thi không xử lý *additional_input, t*hì các bước 7.3 và 7.4 có thể bỏ qua, và bước 7.1 chuyển đổi thành: (*status*, *seed_material*) = **Get_entropy** (*seedlen*, *seedlen*, *seedlen*).

8. *(Key, V)* = **Update** *(seed_material, keylen, Key, V).*
9. *reseed_counter = 1.*
10. *state(state_handle) = {V, Key, keylen, strength, reseed_counter, prediction_resistance_flag }.*
11. **Return** *("Thành công").*

C.3.2.2.7 Tạo các bit giả ngẫu nhiên sử dụng CTR_DRBG(...)

Quá trình sau đây hoặc tương đương được sử dụng để tạo các bit giả ngẫu nhiên.

CTR_DRBG (...):

Đầu vào: số nguyên (*state_handle*, *requested_no_of_bits*, *requested_strength*, *prediction_resistance_request_flag*), xâu *additional_input*.

Đầu ra: xâu *status*, xâu bit *pseudorandom_bits*.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (Thông báo lỗi, *Null*).
2. Lấy các giá trị trạng thái thích hợp, ví dụ: *V* = *state*(*state_handle*).*V*, *Key* = *state*(*state_handle*).*Key*, *keylen* = *state*(*state_handle*).*keylen, strength* =

state(*state_handle*).*strength*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*, *prediction_resistance_flag* = *state*(*state_handle*).*prediction_resistance_flag*.

3. If (*requested_strength* > *strength*), then **Return** (Thông báo lỗi, *Null*).
4. *seedlen* = *blocklen* + *keylen*.
5. *temp* = **len** (*additional_input*).

CHÚ THÍCH 1 Nếu quá trình thực thi không cung cấp *additional_input*, thì tham số đầu vào *additional_input* và bước 5, 6 được bỏ qua.

6. If (*temp* > *max_length*), then **Return** (Thông báo lỗi, *Null*).
7. If (*requested_no_of_bits* > *max_request_length*), then **Return** (Thông báo lỗi, *Null*).
8. If ((*prediction_resistance_request_flag* = Cung cấp khả năng chống lại việc dự đoán trước) and (*prediction_resistance_flag* = Không chống lại việc dự đoán trước)), then **Return** (Thông báo lỗi, *Null*).

CHÚ THÍCH 2 Nếu quá trình thực thi không cần *prediction_resistance_flag*, thì có thể bỏ qua *prediction_resistance_flag* trong tham số đầu vào và bỏ qua bước 8.

9. If ((*reseed_counter* > *reseed_interval*) OR (*prediction_resistance_request_flag* = Cung cấp khả năng chống lại việc dự đoán trước)) then:

   Return (Thông báo lỗi, *Null*).

   CHÚ THÍCH 3 Đây là trường hợp việc thay mầm mới chưa sẵn sàng.

   9.1 *status* = **Reseed_CTR_DRBG_Instantiation** (*state_handle*, *additional_input*).

   CHÚ THÍCH 4 Nếu quá trình thực thi không cung cấp *additional_input*, thì chuỗi Null sẽ thay thế cho *additional_input* trong bước 9.1.

   9.2 If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi, *Null*).

   9.3 *V* = *state*(*state_handle*).*V*, *Key* = *state*(*state_handle*).*Key*, *reseed_counter* = *state*(*state_handle*).*reseed_counter*.

   9.4 *additional_input* = *Null*.

10. If (*additional_input* ≠ *Null*), then *additional_input* = $0^{seedlen}$.

CHÚ THÍCH 5 Nếu quá trình thực thi không cung cấp *additional_input*, thì bỏ qua bước 10.

11. Đoạn mã sau đây được sử dụng khi hàm dẫn xuất sẵn sàng (nguồn độ bất định đầy đủ có thể sẵn sàng hoặc chưa sẵn sàng).

    If (*additional_input* ≠ *Null*), then:

    11.1 *additional_input* = **Block_Cipher_df** (*additional_input*, *seedlen*).

    CHÚ THÍCH 6 Dẫn xuất *seedlen* bit.

    11.2 (*Key, V*) = **Update** (*additional_input, keylen, Key, V*).

    CHÚ THÍCH 7 Nếu quá trình thực thi không cung cấp *additional_input*, thì bỏ qua bước 11.

12. Đoạn mã sau đây được sử dụng khi nguồn độ bất định đầy đủ đã sẵn sàng và hàm dẫn xuất không được sử dụng.

    If (*additional_input* ≠ *Null*), then:

    12.1 *temp* = **len** (*additional_input*).

    12.2 If (*temp* < *seedlen*), then *additional_input* = *additional_input* || $0^{seedlen- temp}$.

    CHÚ THÍCH 8 Nếu độ dài của *additional_input* < *seedlen*, thì đệm thêm 0 để được *seedlen* bit.

    12.3 (*Key, V*) = **Update** (*additional_input, keylen, Key, V*).

    CHÚ THÍCH 9 Nếu quá trình thực thi không cung cấp *additional_input*, thì bỏ qua bước 12.

13. *temp* = *Null*.

14. While (**len** (*temp*) < *requested_no_of_bits*) do:

    14.1 *V*= (*V*+ 1) mod $2^{blocklen}$.

    14.2 *output_block* = **Block_Cipher** (*Key, V*).

    14.2 *temp* = *temp* || *output_block*.

15. *pseudorandom_bits* = *requested_no_of_bits* ngoài cùng bên trái của *temp*.

16. *(Key, V)* = **Update** *(additional_input, keylen, Key, V)*.

CHÚ THÍCH 10 Cập nhật phần thực hiện để đạt tính an toàn về phía sau.

CHÚ THÍCH 11: Nếu quá trình thực thi không cung cấp *additional_input*, thì bước 16 trở thành (Key, V) = Update ($0^{seedlen}$, *keylen, Key, V*).

17. *reseed_counter* = *reseed_counter* + 1.

18. *state*(*state_handle*) = = {*V, Key, keylen, strength, reseed_counter, prediction_resistance_flag*}.

19. **Return** ("Thành công", *pseudorandom_bits*).

C.3.3 OFB_DRBG

C.3.3.1 Thảo luận

OFB_DRBG (...) sử dụng một thuật toán mã khối ISO/IEC ở chế độ phản hồi đầu ra và được quy định trong ISO/IEC 10116. Thuật toán mã khối và độ dài khóa giống nhau được sử dụng cho mọi hoạt động của mã khối. Thuật toán mã khối và kích thước khóa phải đáp ứng hoặc vượt quá các yêu cầu an toàn mà ứng dụng yêu cầu.

C.3.3.2 Mô tả

C.3.3.2.1 Giới thiệu chung

Quá trình khởi tạo và thay mầm mới của **OFB_DRBG (...)** bao gồm việc thu được mầm với lượng bất định phù hợp. Đầu vào độ bất định được sử dụng để dẫn xuất mầm, sau đó được dùng để dẫn xuất các phần tử của trạng thái khởi tạo trong bộ tạo bit ngẫu nhiên tất định. Trạng thái bao gồm:

1. Giá trị *V* được cập nhật mỗi lần *outlen* bit đầu ra khác nhau được tạo ra (trong đó *outlen* là số lượng các bit đầu ra từ thuật toán mã khối cơ bản);
2. Khóa được cập nhật mỗi lần số lượng tiền xác định của các khối đầu ra được tạo ra;
3. Độ dài khóa (*keylen*) được sử dụng trong thuật toán mã khối;
4. Độ mạnh an toàn khi khởi tạo bộ tạo bit ngẫu nhiên tất định;
5. Bộ đếm (*reseed_counter*) bao gồm số lượng các yêu cầu đối với bit giả ngẫu nhiên khi khởi tạo hoặc thay mầm mới; và
6. *prediction_resistance_flag* cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định yêu cầu hay không.

Các biến cho **OFB_DRBG (...)** giống như các biến được sử dụng cho **CTR_DRBG (...)** quy định trong C.3.2.2.1.

C.3.3.2.2 Hàm bên trong: Hàm cập nhật

Hàm **Update (...)** cập nhật trạng thái bên trong của **OFB_DRBG (...)** sử dụng *seed_material*, phải có độ dài bit bằng *seedlen*. Quá trình sau đây hoặc tương đương được sử dụng trong hàm **Update (...)**.

Update (...):

Đầu vào: số nguyên *keylen*, xâu bit (*seed_material, Key, V*).

Đầu ra: xâu bit *(Key, V).*

Quá trình:

1. *seedlen = blocklen + keylen.*
2. *temp = Null.*
3. While (**len** (*temp*) < *seedlen*) do:

   3.1 *V* = **Block_Cipher** (*Key, V*).

   3.2 *temp = temp* || *V.*
4. *temp = seedlen* bit ngoài cùng bên trái của *temp.*

5. *temp* = *temp* ⊕ *seed_material*.

6. *Key* = *keylen* bit ngoài cùng bên trái của *temp*.

7. *V* = *blocklen* bit ngoài cùng bên phải của *temp*.

8. **Return** (*Key*, *V*).

CHÚ THÍCH Điểm khác nhau duy nhất giữa hàm cập nhật của **OFB_DRBG (...)** và **CTR_DRBG (...)** là ở bước 3.

C.3.3.2.3 Khởi tạo OFB_DRBG (...)

Quá trình này tương tự với quá trình khởi tạo đối với **CTR_DRBG (...)** trong C.3.2.2.2.

C.3.3.2.4 Thay mầm mới khi khởi tạo OFB_DRBG(...)

Quá trình này tương tự với quá trình thay mầm mới đối với **CTR_DRBG (...)** trong C.3.2.2.6.

C.3.3.2.5 Tạo các bit giả ngẫu nhiên sử dụng OFB_DRBG(...)

Quá trình này tương tự với quá trình tạo bit đối với **CTR_DRBG (...)** trong C.3.2.2.7, ngoại trừ bước 14 như sau:

14. While (**len** (*temp*) < *requested_no_of_bits*) do:

    14.1. *V* = **Block_Cipher** (*Key*, *V*).

    14.2. *temp* = *temp* || *V*.

C.4 Bộ tạo bit ngẫu nhiên tất định dựa trên các bài toán lý thuyết số

C.4.1 Giới thiệu về bộ tạo bit ngẫu nhiên tất định dựa trên các bài toán lý thuyết số

Bộ tạo bit ngẫu nhiên tất định có thể được thiết kế để tận dụng các bài toán lý thuyết số (ví dụ: bài toán logarit rời rạc). Nếu được thực hiện đúng, các đặc tính ngẫu nhiên và/hoặc khả năng không thể dự đoán trước của bộ tạo đó sẽ được đảm bảo bằng độ khó của việc giải bài toán đã chọn. C.4.3 quy định bộ tạo bit ngẫu nhiên tất định liên quan đến bài toán RSA.

C.4.2 Bộ tạo bit ngẫu nhiên tất định Micali Schnorr (MS_DRBG)

C.4.2.1 Thảo luận

MS_DRBG (...) là một biến thể của bộ tạo RSA được định nghĩa như sau.

Cho $\gcd(x, y)$ là ký hiệu ước chung lớn nhất của hai số nguyên x và y, và $\phi(n)$ biểu diễn hàm phi Euler. Chọn n là tích của hai số nguyên tố lớn khác nhau và e là số nguyên dương sao cho $\gcd(e, \phi(n)) = 1$. Xác định $f(y) = y^e \bmod n$. Bắt đầu với mầm y_0, tạo một chuỗi có dạng $y_{i+1} = f(y_i)$ và xuất ra chuỗi bao gồm $k = \lg \quad \lg(n)$ bit có trọng số thấp nhất của từng y_i. Các bit này (tiệm cận n) được cho là an toàn như hàm RSA f.

Bộ tạo Micali-Schnorr **MS_DRBG (...)** sử dụng cùng giá trị e và n để tạo ra nhiều hơn các bit ngẫu nhiên cho mỗi vòng lặp, trong khi loại bỏ việc sử dụng lại các bit ở cả đầu ra và mầm. Từng $y_i \in [0, n)$ được xem như phép nối $s_i || z_i$ của r bit s_i và k bit z_i (trong đó $k = \lg(n) - r$). s_i được sử dụng để truyền các chuỗi số nguyên: $y_{i+1} = s_i^e \bmod n$; z_i đầu ra là các bit ngẫu nhiên. Lưu ý rằng r ít nhất bằng

$2(\min\{strength, \lg(n)/e\})$, trong đó *strength* là độ mạnh an toàn mong muốn của bộ tạo và $e \geq 3$. Mầm ngẫu nhiên r bit s_0 được sử dụng để khởi tạo quá trình.

MS_DRBG (...) an toàn về mật mã theo giả thuyết mạnh hơn rằng các chuỗi có dạng $s^e mod\, n$ giống với các chuỗi số nguyên ngẫu nhiên trong Z_n. s được giả định là các số nguyên r bit và "cùng" có nghĩa là không thể phân biệt bằng bất kỳ thuật toán có thời gian đa thức nào. Chấp nhận giả thiết mạnh hơn cho phép k là tỷ lệ phần trăm đáng kể của $\lg(n)$.

Độ dài r và k, số mô-đun RSA n và giá trị của số mũ e có thể thay đổi trong giới hạn được mô tả sau đây. Giới hạn dựa trên độ mạnh mong đợi của các bit tạo ra. Để đạt hiệu quả lớn nhất thì e phải nhỏ và k lớn. k bit được tạo ra ở mỗi bước được nối để tạo ra các xâu bit giả ngẫu nhiên có độ dài mong muốn.

Nguyên liệu mầm được cung cấp bằng hàm thực thi phụ thuộc **Get_entropy (...)**. Độ bất định nhỏ nhất yêu cầu từ hàm này được thiết lập bằng $\mathbf{max}(128, strength)$.

Tính an toàn về phía sau vốn có trong thuật toán, ngay cả khi trạng thái bên trong bị tấn công. Tính an toàn về phía trước cũng có sẵn khi quan sát từ bên ngoài ranh giới bộ tạo bit ngẫu nhiên tất định. Nếu ứng dụng quan tâm đến việc thỏa hiệp trạng thái ân khi khởi tạo **MS_DRBG (...)**, trạng thái có thể được truyền với độ bất định mới theo một số cách.

Khi sử dụng đầu vào bổ sung tùy chọn (*additional_input*), giá trị *additional_input* là tùy ý và được băm thành một xâu r bit.

C.4.2.2 Mô tả

C.4.2.2.1 Giới thiệu chung

Khi khởi tạo **MS_DRBG (...)**, các tham số MS n, e, r và k được lựa chọn như mô tả ở dưới và có được một mầm khởi tạo ngẫu nhiên s_0. Mỗi tham số trong số đó trở thành một phần của trạng thái bên trong của bộ tạo bit ngẫu nhiên tất định. Trạng thái bao gồm:

1. Các tham số MS n, e, r và k;
2. Số nguyên $S \in [0, 2^r)$ truyền chuỗi trạng thái bên trong từ các bit giả ngẫu nhiên được dẫn xuất;
3. Độ mạnh an toàn được cung cấp bởi bộ tạo bit ngẫu nhiên tất định. Để đạt hiệu quả, kích thước mô-đun nhỏ nhất $\lg(n)$ cung cấp *requested_strength* bit an toàn được lựa chọn từ bảng C.4. *strength* thực tế cung cấp được lưu trong *state*;
4. Độ bất định tối thiểu cần thiết từ một lần gọi đến **Get_entropy (...)** để có nguyên liệu mầm. Giá trị *min_entropy* bằng **max** (128, *strength*);
5. Bộ đếm (*reseed_counter*) cho biết số lượng các khối ngẫu nhiên được tạo ra bởi **MS_DRBG (...)** trong trường hợp hiện tại và từ lần thay mầm mới trước đó;
6. *prediction_resistance_flag* cho biết tính an toàn về phía trước có được yêu cầu bởi bộ tạo bit ngẫu nhiên tất định hay không; thiết lập cờ này buộc gọi đến hàm thay mầm mới mỗi lần **MS_DRBG (...)** được yêu cầu một xâu bit ngẫu nhiên; và
7. Bản ghi nguyên liệu mầm dưới dạng hàm **Hash (...)** một chiều được thực hiện trên mầm để so sánh với mầm mới khi bộ tạo bit ngẫu nhiên tất định được thay mầm mới.

Các biến được sử dụng trong mô tả **MS_DRBG (...)** như sau:

additional_input	Xâu bit đã băm dẫn xuất từ *additional_input_string* tùy chọn.
additional_input_string	Đầu vào bổ sung tùy chọn. Một mảng byte được cung cấp mỗi lần gọi các bit ngẫu nhiên. Xâu được băm thành r bit sử dụng **Hash_df (...).**
e	Số nguyên dương được sử dụng làm số mũ RSA.
entropy_input	Các bit chứa độ bất định được sử dụng để xác định *seed_material* và tạo ra mầm.
$gcd(x, y)$	Ước chung lớn nhất của hai số nguyên x và y.
Get_entropy *(min_entropy, min_length, max_length)*	Hàm nhận được một xâu bit từ nguồn bất định. *min_entropy* chỉ ra số lượng độ bất định tối thiểu được cung cấp trong các bit trả về; *min_length* chỉ ra số bit tối thiểu trả về; *max_length* chỉ ra số bit tối đa trả về. **MS_DRBG** luôn quy định *min_length* = *max_length* = r.
Get_random_modulus *($lg(n), e$)*	Hàm tạo ra số mô-đun RSA n ngẫu nhiên có kích thước $\lg(n)$ bit, thỏa mãn $\gcd(e, \phi(n)) = 1$, sử dụng một thuật toán được phê duyệt. (Xem C.4.3.2.2).
Hash *(hash_input)*	Một hàm băm ISO/IEC đã phê duyệt được quy định trong ISO/IEC 10118-3, trả về một xâu bit trong đó *hash_input* có độ dài là bội của 8 bit.
Hash_df *(hash_input, output_len)*	Hàm phân bố độ bất định trong *hash_input* thành một xâu bit có độ dài *output_len*. Hàm **Hash (...)** được sử dụng để thực hiện điều đó. *hash_input* có độ dài là bội của 8 bit; *output_len* có độ dài tùy ý.
i	Giá trị tạm thời được sử dụng làm bộ đếm vòng lặp.
k	Số bit được tạo ra mỗi vòng lặp của **MS_DRBG**. Để tiện cho quá trình thực thi, số này luôn là bội của 8 bit.
$lg(n)$	Số bit trong biểu diễn nhị phân của n.
max *(A, B)*	Hàm trả về giá trị lớn hơn A hoặc B.
max_length	Độ dài tối đa của xâu trả về từ hàm **Get_entropy (...)**.*max_length* là bội của 8 bit.
min_entropy	Lượng độ bất định tối thiểu thu được từ nguồn bất định và được cung cấp trong mầm. CHÚ THÍCH 1 Thực tế giá trị *strength* được sử dụng trong định nghĩa này và *strength* luôn ít nhất bằng *requested_strength*.
min_length	Độ dài tối thiểu của *entropy_input*.
M-S parameters	n, e, r, k
n	Mô-đun RSA; tích của hai số nguyên tố lớn khác nhau p, q.
Null	Xâu rỗng.

old_transformed_entropy_input Bản ghi *entropy_input* đã sử dụng trong trường hợp trước của bộ tạo bit ngẫu nhiên tất định.

p, q Các số nguyên tố được tạo ra sử dụng một thuật toán đã phê duyệt. Các số nguyên tố này được quy định trong ISO/IEC 18032. Những số này được tạo ra ngẫu nhiên khi khởi tạo nếu *use_random_primes* được đặt bằng 1. Ngược lại mô-đun mặc định có kích thước phù hợp được sử dụng.

***pad8** (bitstring)* Hàm có đầu vào là một xâu bit có độ dài tùy ý và trả về một bản sao của xâu bit đó được đệm các bit 0 vào bên phải thành bội của 8.

CHÚ THÍCH 2 Đây là một quá trình thực thi cho các hàm định hướng byte.

personalisation_string Một mảng byte có thể cung cấp sự đảm bảo bổ sung về tính duy nhất mầm khi khởi tạo.

prediction_resistance_flag Cờ khởi tạo cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định cung cấp hay không. Mặc định là cờ này không cần thiết lập. Thiết lập cờ này bằng 1 khiến việc thay mầm mới được thực hiện mỗi lần gọi đến **MS_DRBG (...)**.

pseudorandom_bits Các bit giả ngẫu nhiên được tạo ra bởi bộ tạo bit ngẫu nhiên tất định.

r Độ dài bit của các mầm s_i; $r = \lg(n) - k$.

CHÚ THÍCH 3 r luôn là bội của 8 bit.

R Giá trị được trích xuất từ các bit giả ngẫu nhiên.

requested_e Số mũ RSA e được yêu cầu.

requested_k Kích thước k của mỗi xâu đầu ra được yêu cầu.

requested_no_of_bits Số lượng các bit giả ngẫu nhiên được trả về từ hàm **MS_DRBG (...)**.

requested_strength Độ mạnh an toàn liên quan đến các bit giả ngẫu nhiên được yêu cầu.

reseed_counter Đếm số lượng vòng lặp của **MS_DRBG (...)** kể từ lần thay mầm mới cuối cùng.

reseed_interval Số khối tối đa của đầu ra ngẫu nhiên được tạo ra trước khi thay mầm mới cho bộ tạo bit ngẫu nhiên tất định. Sử dụng giá trị 50000.

S Một giá trị khởi tạo được xác định bằng mầm, nhưng giả định các giá trị mới trong mỗi yêu cầu các bit giả ngẫu nhiên từ bộ tạo bit ngẫu nhiên tất định.

seed_material Mầm được sử dụng để dẫn xuất giá trị khởi tạo của S.

state(state_handle) Một mảng trạng thái cho những lần khởi tạo khác nhau của bộ tạo bit ngẫu nhiên tất định. Trạng thái được thực hiện giữa các lần gọi đến bộ tạo bit ngẫu nhiên tất định. Đối với **MS_DRBG (...)**, trạng thái cho một lần khởi tạo được xác định bằng *state*(*state_handle*) = { *reseed_counter*, S, n, e, r, k, *strength*, *prediction_resistance_flag*,

	transformed_entropy_input }. Một phần tử cụ thể của trạng thái được xác định bằng *state*(*state_handle*).*element*, ví dụ: *state*(*state_handle*).*S*.
state_handle	Xử lý đối với không gian trạng thái trong quá trình khởi tạo.
status	Trạng thái trả về từ một lần gọi hàm, trong đó *status* = "Thành công" hoặc một thông báo lỗi.
strength	Độ mạnh tối đa của một trường hợp bộ tạo bit ngẫu nhiên tất định. Tối thiểu luôn bằng *requested_strength*.
transformed_entropy_input	Bản ghi *seed_material* được sử dụng trong trường hợp bộ tạo bit ngẫu nhiên tất định hiện tại.
Truncate *(bits, in_len, out_len)*	Hàm có đầu vào là một xâu bit có độ dài *in_len*, trả về một xâu bao gồm *out_len* bit ngoài cùng bên trái của đầu vào. Nếu *in_len* < *out_len*, thì xâu đầu vào được đệm thêm vào bên phải với (*out_len* - *in_len*) số 0, và trả về kết quả.
use_random_primes	Nếu bằng 1 (*Use_random_primes*), các số nguyên tố ngẫu nhiên có kích thước $\lg(n)/2$ được tạo ra khi khởi tạo sử dụng một thuật toán đã phê duyệt và có độ bất định ít nhất bằng *min_entropy*. Nếu bằng 0 (*Random_primes_not_required*), sử dụng mô-đun phù hợp từ D.1.
y_i	Số nguyên $y_i \in [0, n)$. $y_i = s_i \| \| z_i$.
z_i	k bit đầu ra của **MS_DRBG(...)** trong mỗi vòng lặp i.
$\phi(n)$	Hàm phi Euler: $\phi(n)$ = số lượng các số nguyên dương $< n$ nguyên tố cùng nhau với n. Đối với mô-đun RSA $n = pq$, $\phi(n) = (p-1)(q-1)$.

C.4.2.2.2 Lựa chọn các tham số MS

Khởi tạo **MS_DRBG (...)** bao gồm lựa chọn mô-đun RSA n và số mũ e phù hợp; kích thước r và k cho mầm và xâu đầu ra tương ứng; và một mầm bắt đầu.

Các tham số MS n, r, e và k được lựa chọn thỏa mãn sáu điều kiện sau đây dựa vào độ mạnh:

1. $1 < e < \phi(n); \gcd(e, \phi(n)) = 1;$	đảm bảo rằng ánh xạ $s \rightarrow s^e mod\ n$ là 1:1
2. $re \geq 2\lg(n);$	đảm bảo rằng phép lũy thừa yêu cầu rút gọn theo mô-đun.
3. $r \geq 2strength;$	bảo vệ chống lại tấn công tra bảng.
4. k, r là bội của 8;	thuận tiện cho thực thi
5. $k \geq 8; r + k = \lg(n);$	tất cả các bit được sử dụng
6. $n = pq$	các số nguyên tố bí mật mạnh (như trong ISO/IEC 18032).

Các tham số MS được xác định theo thứ tự này:

1. Kích thước mô-đun $\lg(n)$ được thiết lập đầu tiên. Nó phải thỏa mãn các giá trị cho trong bảng C.4 đối với độ mạnh an toàn yêu cầu.

Bảng C.4 – Độ mạnh an toàn tương đương

Các bit an toàn	RSA
80	$\lg(n) = 1024$
112	$\lg(n) = 2048$
128	$\lg(n) = 3072$
192	$\lg(n) = 7680$
256	$\lg(n) = 15360$

2. Số mũ RSA e. Quá trình thực thi phải cho phép ứng dụng yêu cầu số nguyên lẻ bất kỳ e trong khoảng $1 < e < 2^{\lg(n)-1} - 2 * 2^{1/2\lg(n)}$.

   CHÚ THÍCH 1 Không tồn tại dấu bằng đảm bảo rằng $e < \phi(n)$ khi sử dụng một thuật toán được phê duyệt để tạo các số nguyên tố p, q.

   Nếu không yêu cầu số mũ, thì sử dụng giá trị mặc định $e = 3$.

3. Số k các bit đầu ra được sử dụng cho mỗi vòng lặp. Quá trình thực thi cho phép là bội bất kỳ của 8 trong khoảng $8 \leq k \leq \min\{\lg(n) - 2strength, \lg(n) - 2\lg(n)/e\}$. Nếu không quy định, k nên được chọ là bội lớn nhất của 8 trong khoảng cho phép.

   Giá trị bất kỳ cho *requested_e* và *requested_k* nằm ngoài các khoảng này sẽ bị gắn cờ lỗi.

4. Thiết lập kích thước r của mầm: $r = \lg(n) - k$

5. Lựa chọn số mô-đun n. Ứng dụng có thể yêu cầu một số mô-đun bí mật hoặc sử dụng mô-đun mặc định có kích thước phù hợp (có trong D.1). Quá trình thực thi cho phép một trong hai dựa trên giá trị *use_random_primes*.

Nếu *use_random_primes* = 1, hai số nguyên tố p và q có kích thước $\lg(n)/2$ bit có độ bất định tối thiểu *min_entropy* và thỏa mãn $\gcd(e, (p-1)(q-1)) = 1$ được tạo ra sử dụng một thuật toán đã phê duyệt. Các thuật toán phù hợp được quy định trong ISO/IEC 18032. Quá trình thực thi phải sử dụng các số nguyên tố mạnh được định nghĩa trong tiêu chuẩn đó: mỗi $p-1, p+1, q-1$ và $q+1$ phải có một nhân tử nguyên tố lớn có độ dài ít nhất là *strength* bit.

CHÚ THÍCH 2 Một thuật toán được phê duyệt phải tạo ra số mô-đun có kích thước $\lg(n)$ bit sử dụng các số nguyên tố mạnh có kích thước $\lg(n)/2$ bit và cho phép số mũ e được chỉ định trước.

Độ khó của bài toán RSA phụ thuộc vào tính bí mật của số nguyên tố p và q có trong mô-đun. Khi các số nguyên tố bí mật được tạo ra, quá trình thực thi phải xóa sạch bộ nhớ chứa các giá trị đó trước khi thoát khỏi trình khởi tạo. Chỉ số mô-đun n được giữ trong trạng thái bên trong.

Nếu *use_random_primes* = 0 phải sử dụng số mô-đun phù hợp từ D.1.

Các số mô-đun này được tạo ra bằng cách sử dụng các số nguyên tố mạnh có dạng $p = 2p_1 + 1, q = 2q_1 + 1$, trong đó p_1 và q_1 là số nguyên tố. Ngoài ra, $p + 1$ và $q + 1$ đều có nhân tử nguyên tố lớn theo yêu cầu.

CHÚ THÍCH 3 Việc lựa chọ các số nguyên tố mạnh này về cơ bản đảm bảo rằng bất kỳ số mũ lẻ e trong khoảng cho phép có thể được yêu cầu phải nguyên tố cùng nhau với $\phi(n)$.

C.4.2.2.3 Khởi tạo MS_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để khởi tạo quá trình **MS_DRBG (...)**. Cho **Hash (...)** là hàm băm được phê duyệt cho độ mạnh an toàn hỗ trợ. Nếu bộ tạo bit ngẫu nhiên tất định được sử dụng cho nhiều độ mạnh an toàn và chỉ hàm băm đơn sẵn dàng thì hàm băm phải phù hợp với tất cả các độ mạnh an toàn hỗ trợ.

Instantiate_MS_DRBG (...):

Đầu vào: số nguyên (*requested_strength, prediction_resistance_flag, use_random_primes, requested_e, requested_k*), xâu *personalisation_string*.

Đầu ra: xâu status, số nguyên state_handle.

Quá trình:

1. If (*requested_strength* > độ mạnh an toàn tối đa được cung cấp bởi quá trình thực thi, then **Return** (thông báo lỗi).
2. Xác định kích thước mô-đun $\lg(n)$ phù hợp với độ mạnh yêu cầu sử dụng bảng C.4

   If (*requested_strength* ≤ 80) then *strength* = 80, $\lg(n)$ = 1024

   Else: if (*requested_strength* ≤ 112) then *strength* = 112, $\lg(n)$ = 2048

   Else: if (*requested_strength* ≤ 128) then *strength* = 128, $\lg(n)$ = 3072

   Else: if (*requested_strength* ≤ 192) then *strength* = 192, $\lg(n)$ = 7680

   Else: if (*requested_strength* ≤ 256) then *strength* = 256, $\lg(n)$ = 15360

   Else: **Return** (thông báo lỗi).
3. Lựa chọn kích thước số mũ e. Kích thước mặc định là $e = 3$.

   If (*requested_e* = 0), then $e = 3$

   Else: Kiểm tra giới hạn

   {

   If ($e < 3$), then **Return** (thông báo lỗi);

   CHÚ THÍCH 1 Số nguyên e ít nhất bằng 3.

   If ($e \geq 2^{\lg(n)-1} - 2^{\lg(n)/2+1}$), then **Return** (thông báo lỗi);

   CHÚ THÍCH 2 Bước này đảm bảo rằng e phải nhỏ hơn $\phi(n)$.

   If (e là số chẵn), then **Return** (thông báo lỗi);

CHÚ THÍCH 3 e phải nguyên tố cùng nhau với $\phi(n)$, nên là số lẻ.

}

4. Lựa chọn độ dài đầu ra k. **MS_DRBG (...)** sử dụng k bit có trọng số thấp nhất của $y_i = s_i||z_i$ cho từng vòng lặp. Kích thước mặc định là sử dụng giá trị lớn nhất có thể.

If (requested_k= 0), then:

$k = \min\{\lfloor \lg(n) - 2strength \rfloor, \lfloor \lg(n)(1 - 2/e) \rfloor\}$

CHÚ THÍCH 4 $3 \le e < 2^{\lg(n)-1} - 2\left(2^{1/2\lg(n)}\right) \Rightarrow 8 \le 2/3\lg(n) \le \lfloor \lg(n)\,(1 - 2/e) \rfloor \le \lg(n) - 1$

Vòng lặp giảm xuống bội của 8 :

$k = 8\lfloor k/8 \rfloor$

Else:

{

CHÚ THÍCH 5 Sau đây kiểm tra giới hạn.

k = *requested_k*

If ($k < 1$), then **Return** (thông báo lỗi)

If ($k > \min\{\lfloor \lg(n) - 2strength \rfloor, \lfloor \lg(n)(1 - 2/e) \rfloor\}$),

then **Return** (thông báo lỗi)

If (k không là bội của 8),

then **Return** (thông báo lỗi)

}

5. Thiết lập kích thước của mầm.

$r = \lg(n) - k$

CHÚ THÍCH 6 $r \ge 2strength$

6. Lựa chọn số mô-đun n. *use_random_primes* xác định sử dụng giá trị mặc định hay số mô-đun bí mật được tạo ra.

If (*use_random_primes* = *Random_primes_not_required*)

then thiết lập n dựa vào kích thước $\lg(n)$ từ danh sách trong D.1

Else

{

(*status*, n) = **Get_random_modulus** ($\lg(n), e$)

If (status ≠ "Thành công"), then **Return** (thông báo lỗi)

}

CHÚ THÍCH 7 Hàm được phê duyệt sử dụng để tạo ra số mô-đun n ngẫu nhiên có kích thước phù hợp, có các số nguyên tố mạnh làm nhân tử và với $\gcd(\phi(n), e) = 1$.

7. *min_entropy* = **max** (128, *strength*).
8. *min_length* = r.
9. (*status, entropy_input*) = **Get_entropy** (*min_entropy, min_length, max_length*).
10. If (status ≠ "Thành công"), then **Return** (thông báo lỗi).
11. *seed_material* = *entropy_input* || *personalisation_string*.
12. Sử dụng hàm băm để đảm bảo rằng độ bất định được phân bố đều trên các bit: S = **Hash_df** (*seed_material*, r).
13. Thực hiện hàm một chiều trên nguyên liệu mầm để so sánh sau: *transformed_entropy_input* = **Hash** (*entropy_input*).
14. *reseed_counter* = 0.

CHÚ THÍCH 8 *reseed_counter* được tăng lên mỗi k bit.

15. Lưu tất cả các giá trị vào trạng thái.

    state(*state_handle*) = {*reseed_counter*, S, n, e, r, k, *strength*, *prediction_resistance_flag*, *transformed_entropy_input*}.

16. **Return** ("Thành công").

C.4.2.2.4 Thay mầm mới khởi tạo MS_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để thay mầm mới cho quá trình MS_DRBG (...), sau khi được khởi tạo.

Reseed_MS_DRBG_Instantiation (...):

Đầu vào: số nguyên *state_handle*, xâu *additional_input_string*.

Đầu ra: xâu *status*.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi).
2. Lấy các giá trị trạng thái phù hợp, ví dụ: S = *state(state_handle).S*, r = *state(state_handle).r*, *old_transformed_entropy_input* = *state(state_handle).transformed_entropy_input*..
3. *min_entropy* = **max** (128, *strength*).
4. *min_length* = r.
5. (*status, entropy_input*) = **Get_entropy** (*min_entropy, min_length, max_length*).
6. If (status ≠ "Thành công"), then **Return** (thông báo lỗi).
7. Thực hiện hàm một chiều đối với nguyên liệu mầm để so sánh: *transformed_entropy_input* = **Hash** (*entropy_input*).

8. Kiểm tra nguồn bất định khả thi:

   If (*transformed_entropy_input* = *old_transformed_entropy_input*),

   then **Return** (thông báo lỗi).

9. Kết hợp *entropy_input* mới với trạng thái cũ và *additional_input*:

   seed_material = *S* || *entropy_input* || *additional_input_string*.

10. *S* = **Hash_df** (*seed_material*, *r*).

11. Cập nhật các giá trị trạng thái phù hợp và thiết lập lại *reseed_counter* về 0.

    11.1 *state(state_handle).S* = *S*.

    11.2 *state(state_handle).transformed_entropy_input* = *transformed_entropy_input*.

    11.3 *state(state_handle).reseed_counter* = *0*.

12. **Return** ("Thành công").

C.4.2.2.5 Tạo các bit giả ngẫu nhiên sử dụng MS_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để tạo các bit giả ngẫu nhiên.

MS_DRBG (...):

Đầu vào: số nguyên (*state_handle, requested_strength, requested_no_of_bits*), xâu *additional_input_string*.

Đầu ra: xâu *status*, xâu bit *pseudorandom_bits*.

Quá trình:

1. If không tồn tại trạng thái, then **Return** (thông báo lỗi, *Null*).
2. Lấy các giá trị trạng thái phù hợp, ví dụ: *S* = *state(state_handle).S*, n = *state(state_handle).n*, e = *state(state_handle).e*, k = *state(state_handle).k*, r = *state(state_handle).r*, *strength* = *state(state_handle).strength*, *reseed_counter* = *state(state_handle).reseed_counter*, *prediction_resistance_flag* = *state(state_handle).prediction_resistance_flag*.
3. Kiểm tra requested_strength không lớn hơn giá trị được cung cấp trong quá trình khởi tạo này.

   If (*requested_strength* > *strength*), then **Return** (thông báo lỗi, *Null*).

4. Kiểm tra yêu cầu cung cấp đầu vào bổ sung. Điều này sẽ chỉ được thêm vào trạng thái ở vòng lặp đầu tiên.

   If (*additional_input_string* = *Null*), then *additional_input* = 0

   CHÚ THÍCH 1 additional_input được thiết lập bằng r số 0.

   Else: *additional_input* = **Hash_df** (**pad8**(*additional_input_string*), r).

   CHÚ THÍCH 2 Băm thành r bit.

CHÚ THÍCH 3 Nếu yêu cầu chống lại việc dự đoán trước được thực hiện, các bước sau đây sẽ kích hoạt độ bất định mới vào một lần gọi để thay mầm mới cho **MS_DRBG (...)**. Quá trình thay mầm mới thiết lập lại *reseed_counter* về 0.

5. If (*prediction_resistance_flag* = 1), then:

   5.1 *status* = **Reseed_MS_DRBG_Instantiation** (*state_handle, Null*).

   5.2 If (*status* ≠ "Thành công"), then **Return** (*status, Null*).

   5.3 *S* = *state(state_handle).S*, *reseed_counter* = *state(state_handle).reseed_counter.*

6. *temp* = xâu rỗng; $i = 0$.

7. Xác định nếu thay mầm mới được yêu cầu. Reseed_MS_DRBG_Instantiation(...) thiết lập *reseed_counter* về 0.

   If (*reseed_counter* = 50000), then:

   7.1 *status* = **Reseed_MS_DRBG_Instantiation** (*state_handle, Null*).

   7.2 If (status ≠ "Thành công"), then **Return** (*status, Null*).

   7.3 *S* = *state(state_handle).S*, *reseed_counter* = *state(state_handle).reseed_counter.*

8. $s = S \oplus additional\_input$.

CHÚ THÍCH 4 : s được hiểu là số nguyên không dấu r bit.

9. $S = (s^e \bmod n)/2^k$.

CHÚ THÍCH 5 : S là một số r bit.

10. $R = (s^e \bmod n) \bmod 2^k$

CHÚ THÍCH 6 : R là một số k bit.

11. *temp* = *temp* || R.

12. *additional_input* = 0.

CHÚ THÍCH 7 r số 0 chỉ được thêm vào *additional_input_string* trong vòng lặp đầu tiên.

13. $i = i + 1$.

14. *reseed_counter* = *reseed_counter* + 1.

15. If (|*temp*| < *requested_no_of_bits*), then quay lại bước 7.

16. *pseudorandom_bits* = **Truncate** (*temp*, $i \times k$, *requested_no_of_bits*).

17. Cập nhật các giá trị thay đổi trong trạng thái.

    17.1. *state(state_handle).S* = *S*.

    17.2. *state(state_handle).reseed_counter* = *reseed_counter.*

18. **Return** ("Thành công", *pseudorandom_bits*).

C.4.2.2.6 Thêm độ bất định bổ sung vào trạng thái của MS_DRBG (...)

Độ bất định bổ sung được thêm vào trạng thái của **MS_DRBG (...)** bằng bốn cách. Độ bất định bổ sung được thêm vào bằng cách:

1. Gọi đến hàm **Reseed_MS_DRBG_Instantiation(...)**. Hàm này luôn gọi hàm thực thi phụ thuộc **Get_entropy (...)** để *min_entropy* = **max** (128, *strength*) bit mới của độ bất định được thêm vào trạng thái;

2. Sử dụng tính năng thay mầm mới tự động của **MS_DRBG (...)**. Tính năng thay mầm mới tự động sẽ buộc gọi đến **Reseed_MS_DRBG_Instantiation (...)** khi 50000 khối đầu ra được tạo ra từ lần thay mầm mới cuối;

3. Thiết lập *prediction_resistance_flag* = 1 khi khởi tạo. Buộc gọi đến **Reseed_MS_DRBG_Instantiation (...)** mỗi lần **MS_DRBG(...)** được yêu cầu; hoặc

4. Cung cấp đầu vào bổ sung mỗi lần gọi đến **MS_DRBG (...)** để có các bit ngẫu nhiên.

CHÚ THÍCH Tần suất gọi đến hàm **Get_entropy (...)** có thể gây ra giảm hiệu suất của bộ tạo bit ngẫu nhiên tất định này.

C.5 Bộ tạo bit ngẫu nhiên tất định dựa trên phương trình bậc hai đa biến

C.5.1 Giới thiệu về bộ tạo bit ngẫu nhiên tất định dựa trên phương trình bậc hai đa biến

MQ_DRBG (...) dựa trên sự lặp lại của hệ bậc hai đa biến được chọn ngẫu nhiên. Quá trình khởi tạo và thay mầm mới dựa trên hàm dẫn xuất băm **Hash_df** để phân bố độ bất định thông qua mầm. Hàm **Hash_df** được quy định trong C.2.2.2.2 và sử dụng một hàm băm ISO/IEC đã phê duyệt tuân theo độ mạnh an toàn cần thiết để khởi tạo **MQ_DRBG (...)**.

C.5.2 Bộ tạo bit ngẫu nhiên tất định bậc hai đa biến (MQ_DRBG)

C.5.2.1 Thảo luận

MQ_DRBG (...) dựa trên bài toán khó sau đây: Cho một hệ P các phương trình bậc hai đa biến trên trường nhị phân và kết quả $P(x)$ của việc đánh giá hệ trên đầu vào x, tìm x. (Bài toán này đôi khi được gọi là bài toán bậc hai đa biến).

MQ_DRBG (...) sử dụng một mầm có độ dài *state_length* bit để khởi tạo việc tạo ra các xâu *block_length* bit giả ngẫu nhiên bằng cách nhắc lại một hệ bậc hai đa biến ánh xạ các xâu *state_length* bit thành các xâu (*state_length* + *block_length*) bit. Các tham số độ dài *state_length* và *block_length* được lựa chọn sao cho là bội của 8 và gần bằng nhau. Lựa chọn các tham số độ dài nhỏ nhất có thể, tùy theo độ mạnh an toàn mà ứng dụng yêu cầu, có thể cải thiện hiệu suất hoạt động.

Quá trình khởi tạo bộ tạo bit ngẫu nhiên tất định này yêu cầu lựa chọn một hệ phù hợp các phương trình bậc hai đa biến được quy định trong C.5.2.5 cho độ mạnh an toàn mong đợi.

Mầm được sử dụng để xác định giá trị khởi tạo (S) của bộ tạo bit ngẫu nhiên tất định phải có độ bất định tối thiểu bằng giá trị lớn nhất của 128 và độ mạnh an toàn mong đợi (tức là entropy ≥ $\mathbf{max}\,(128, strength)$). Độ dài mầm phải bằng *state_length* bit. Tính an toàn về phía sau vốn có trong thuật toán ngay cả khi trạng thái bên trong bị tấn công. Tính an toàn về phía trước cũng sẵn có khi quan sát từ bên ngoài ranh giới bộ tạo bit ngẫu nhiên tất định. Nếu ứng dụng quan tâm đến việc thỏa hiệp trạng thái ẩn khi khởi tạo **MQ_DRBG (...)**, trạng thái có thể được truyền với độ bất định mới theo một số cách được trình bày trong C.5.2.2.7.

Khi đầu vào bổ sung tùy chọn (*additional_input*) được sử dụng, giá trị *additional_input* được băm thành một xâu *state_length* bit.

C.5.2.2 Mô tả

C.5.2.2.1 Giới thiệu chung

Quá trình khởi tạo và thay mầm mới của **MQ_DRBG (...)** bao gồm việc thu được đầu vào độ bất định với lượng bất định tối thiểu bằng yêu cầu của ứng dụng. Đầu vào độ bất định được sử dụng để dẫn xuất mầm. Mầm được dùng để dẫn xuất các phần tử của trạng thái khởi tạo, bao gồm:

1. Giá trị S được cập nhật trong mỗi lần gọi đến bộ tạo bit ngẫu nhiên tất định;
2. Độ dài của giá trị này *state_length*;
3. Tham số độ dài khối *block_length*;
4. Tham số *field_size*;
5. Tham số hệ thống P cung cấp các hệ số cho hệ các phương trình bậc hai đa biến;
6. Bộ đếm *reseed_counter* cho biết số lượng khối các bit giả ngẫu nhiên được tạo ra từ lần cuối đầu vào độ bất định được thêm vào trạng thái trong quá trình khởi tạo hoặc thay mầm mới;
7. Giá trị tối đa *reseed_interval* cho *reseed_counter*;
8. Độ mạnh an toàn của quá trình khởi tạo bộ tạo bit ngẫu nhiên tất định; và
9. Giá trị *prediction_resistance_flag* cho biết tính an toàn về phía trước có được yêu cầu bởi bộ tạo bit ngẫu nhiên tất định hay không.

Các biến được sử dụng trong mô tả **MQ_DRBG (...)** là:

additional_input	Đầu vào bổ sung tùy chọn.
block	Xâu bit tạm thời.
block_length	Số lượng bit trả về bởi hàm **Evaluate_MQ(...)** dùng làm các bit giả ngẫu nhiên được trả về bởi **MQ_DRBG(...)**. Tham số này phải tối thiểu bằng *requested_block_length* và tối thiểu bằng *strength*.
entropy_input	Các bit chứa độ bất định được sử dụng để xác định *seed_material* và tạo ra mầm.
field_size	Số lượng các bit của một phần tử trên trường Galois GF(2^{field_size}).
field_vector(*bits, in_len, field_size*)	Hàm có đầu vào là xâu *bits* có độ dài *in_len* là bội của *field_size*. Nó trả về một mảng *vec* có độ dài là *in_len / field_size* gồm các phần tử là các xâu bit có độ dài *field_size*, trong đó $vec[1] = bits[1] \| \cdots \| bits[field\_size], vec[2] = bits[field\_size + 1] \| \cdots \| bits[2 * field\_size]$, ..., sao cho **flatten(...)** áp dụng cho đầu ra trả về xâu bit ban đầu.
flatten(*in_array, num_elements*)	Hàm có đầu vào là một mảng num_elements xâu bit và trả về xâu bit $in\_array[1] \| in\_array[2] \| \ldots \| in\_array[num\_elements]$ tạo bởi cách nối các xâu bit theo thứ tự.

~~Get~~_entropy (*min_entropy*, *min_length*, *max_length*)	Hàm nhận được một xâu bit từ nguồn bất định. *min_entropy* chỉ ra số lượng độ bất định tối thiểu được cung cấp trong các bit trả về; *min_length* chỉ ra số bit tối thiểu trả về; *max_length* chỉ ra số bit tối đa trả về.
Hash (*hash_input*)	Một hàm băm ISO/IEC đã phê duyệt được quy định trong ISO/IEC 10118-3, trả về một xâu bit trong đó *hash_input* có độ dài là bội của 8 bit.
Hash_df (*hash_input*, *output_len*)	Hàm phân bố độ bất định trong *hash_input* thành một xâu bit có độ dài *output_len*. Hàm **Hash (...)** được sử dụng để thực hiện điều đó. *hash_input* có độ dài là bội của 8 bit; *output_len* có độ dài tùy ý.
i, j, k	Các giá trị tạm thời được sử dụng làm bộ đếm vòng lặp.
max (*A*, *B*)	Hàm trả về giá trị lớn hơn A hoặc B.
max_length	Độ dài tối đa của xâu trả về từ hàm **Get_entropy (...)**.
min_entropy	Giá trị được sử dụng trong yêu cầu đến **Get_entropy (...)** để chỉ ra độ bất định tối thiểu cần cung cấp cho nguyên liệu mầm. Giá trị này luôn cố định và bằng $\mathbf{max}(128, strength)$.
min_length	Độ dài tối thiểu của *entropy_input.*
n, m	Các giá trị nguyên tạm thời được sử dụng làm giới hạn vòng lặp.
Null	Xâu rỗng.
P	Xâu bit được sử dụng để lưu các hệ số của hệ các phương trình bậc hai đa biến được sử dụng bởi bộ tạo bit ngẫu nhiên tất định.
P_vec	Vectơ của các phần tử trường được sử dụng làm biến tạm thời.
pad8 (*bitstring*)	Hàm có đầu vào là một xâu bit có độ dài tùy ý và trả về một bản sao của xâu bit đó được đệm các bit 0 vào bên phải thành bội của 8.
personalisation_string	Một mảng byte có thể cung cấp sự đảm bảo bổ sung về tính duy nhất mầm khi khởi tạo.
prediction_resistance_flag	Cờ khởi tạo cho biết tính an toàn về phía trước có được bộ tạo bit ngẫu nhiên tất định cung cấp hay không. Mặc định là cờ này không cần thiết lập. Thiết lập cờ này bằng 1 buộc gọi đến **Reseed_MQ_DRBG_Instantiation (...)** từ trong **MQ_DRBG (...)** mỗi lần quá trình được yêu cầu.
pseudorandom_bits	Các bit giả ngẫu nhiên được tạo ra bởi bộ tạo bit ngẫu nhiên tất định.
requested_block_length	Giá trị yêu cầu đối với *block_length* khi khởi tạo.
requested_no_of_bits	Số lượng các bit giả ngẫu nhiên được trả về khi gọi đến **MQ_DRBG (...).**
requested_strength	Độ mạnh an toàn liên quan đến các bit giả ngẫu nhiên được yêu cầu.

reseed_counter	Đếm số lượng vòng lặp của **MQ_DRBG (...)** kể từ lần thay mầm mới cuối cùng.
reseed_interval	Số khối tối đa của đầu ra ngẫu nhiên được tạo ra trước khi thay mầm mới cho bộ tạo bit ngẫu nhiên tất định.
S	Một giá trị được thay mới khi khởi tạo và thay mầm mới và được cập nhật bởi hàm **Evaluate_MQ(...)**.
seed_material	Mầm được sử dụng để dẫn xuất giá trị khởi tạo của *S*.
state(*state_handle*)	Một mảng trạng thái cho những lần khởi tạo khác nhau của bộ tạo bit ngẫu nhiên tất định. Trạng thái được thực hiện giữa các lần gọi đến bộ tạo bit ngẫu nhiên tất định. Đối với **MQ_DRBG (...)**, trạng thái cho một lần khởi tạo được xác định bằng *state*(*state_handle*) = { *S*, *P*, *reseed_counter, reseed_interval, state_length, block_length, field_size, strength, prediction_resistance_flag*}. Một phần tử cụ thể của trạng thái được xác định bằng *state*(*state_handle*).*element*, ví dụ: *state*(*state_handle*).*S*.
state_handle	Xử lý đối với không gian trạng thái trong quá trình khởi tạo.
state_length	Kích thước giá trị khởi tạo *S* trong bộ tạo bit ngẫu nhiên tất định. Giá trị này luôn tối thiểu bằng *requested_block_length* và *strength*.
status	Trạng thái trả về từ một lần gọi hàm, trong đó *status* = "Thành công" hoặc một thông báo lỗi.
strength	Độ mạnh tối đa của một trường hợp bộ tạo bit ngẫu nhiên tất định.
system_length	Kích thước của xâu bit *P* chứa hệ các phương trình bậc hai được sử dụng bởi **MQ_DRBG (...)**. Hoàn toàn xác định bằng cách lựa chọn các tham số *state_length, block_length* và *field_size* được quy định trong C.5.2.4.
t	Giá trị nguyên tạm thời được sử dụng làm chỉ số cho các xâu bit.
temp	Xâu bit tạm thời.
Truncate (*bits*, *in_len*, *out_len*)	Hàm có đầu vào là một xâu bit có độ dài *in_len*, trả về một xâu bao gồm *out_len* bit ngoài cùng bên trái của đầu vào. Nếu *in_len* < *out_len*, thì xâu đầu vào được đệm thêm vào bên phải với (*out_len* - *in_len*) số 0, và trả về kết quả.
x_{vec}, y_{vec}, z_vec	Vectơ của các phần tử trường được sử dụng làm các biến tạm thời.
$x[i]$	Phần tử thứ i trong mảng gồm x xâu bit.
*	Phép nhân trên trường nhị phân có kích thước *field_size*.

C.5.2.2.2 Lựa chọn các tham số MQ_DRBG

Quá trình khởi tạo **MQ_DRBG (...)** yêu cầu lựa chọn các tham số khởi tạo phù hợp. Lựa chọn các tham số phù hợp được thực hiện dựa trên các giá trị *requested_strength* và *requested_block_length* được cho trong đầu vào của hàm khởi tạo. Các tham số được lựa chọn theo bảng sau.

Bảng C.5 – Các tham số khởi tạo MQ_DRBG dựa trên *requested_block_strength* và *requested_strength*

requested_strength	*requested_block_length*			
	1-112	113-128	129-192	193-256
1-80	*strength*=80 *state_length*=112 *block_length*=112 *field_size*=1 *reseed_interval*=2^{23} *system_length*=1417696	*strength*=80 *state_length*=128 *block_length*=128 *field_size*=4 *reseed_interval*=2^{12} *system_length*=143616	*strength*=80 *state_length*=192 *block_length*=192 *field_size*=6 *reseed_interval*=2^{12} *system_length*=215424	*strength*=80 *state_length*=256 *block_length*=256 *field_size*=8 *reseed_interval* = 2^{14} *system_length*= 287232
81-112	*strength*=112 *state_length*=120 *block_length*=112 *field_size*=1 *reseed_interval*=2^{26} *system_length*=1684552	*strength*=112 *state_length*=128 *block_length*=128 *field_size*=1 *reseed_interval*=2^{32} *system_length*=2113792	*strength*=112 *state_length*=192 *block_length*=192 *field_size*=4 *reseed_interval*=2^{12} *system_length*=470400	*strength*=112 *state_length*=256 *block_length*=256 *field_size*=4 *reseed_interval* = 2^{21} *system_length*= 1098240
113-128	ERROR	*strength*=128 *state_length*=128 *block_length*=128 *field_size*=1 *reseed_interval*=2^{28} *system_length*=2113792	*strength*=128 *state_length*=192 *block_length*=192 *field_size*=3 *reseed_interval*=2^{16} *system_length*=823680	*strength*=128 *state_length*=256 *block_length*=256 *field_size*=4 *reseed_interval*

				= 2^{17} *system_length*= 1098240
129-192	ERROR	ERROR	*strength*=192 *state_length*=200 *block_length*=192 *field_size*=1 *reseed_interval*=2^{32} *system_length*=7879592	*strength*=192 *state_length*=256 *block_length*=256 *field_size*=2 *reseed_interval* = 2^{30} *system_length* = 4293120
193-256	ERROR	ERROR	ERROR	*strength*=256 *state_length*=272 *block_length*=256 *field_size*=1 *reseed_interval* = 2^{32} *system_length* = 19604112

C.5.2.2.3 Khởi tạo MQ_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để khởi tạo quá trình **MQ_DRBG (...)**.

Instantiate_MQ_DRBG (...):

Đầu vào: số nguyên (*requested_strength, prediction_resistance_flag, requested_block_length*), xâu *personalisation_string*.

Đầu ra: xâu *status*, số nguyên *state_handle*.

Quá trình:

1. If *requested_strength* hoặc *requested_block_length* lớn hơn giá trị tối đa mà quá trình thực thi có thể cung cấp, then **Return** (thông báo lỗi). If *requested_block_length* nhỏ hơn *requested_strength* then **Return** (thông báo lỗi).

2. Xác định các tham số của không gian trạng thái. Tăng *requested_strength* lên đến kích thước có sẵn tiếp theo trong bảng C.5. Thiết lập *strength*, *state_length*, *block_length*, *field_size* và *reseed_interval* bằng các giá trị trong bảng này.

3. Lựa chọn tham số hệ *P* dựa vào *state_length* và *block_length*.

CHÚ THÍCH Định dạng và lựa chọn *P* được mô tả trong C.5.2.4 và C.5.2.5.

4. *min_entropy* = **max**(128, *strength*).

5. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).

6. If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).

7. *seed_material* = *entropy_input* || *personalisation_string*.

8. *S* = **Hash_df** (**pad8**(*seed_material*), *state_length*).

9. Khởi tạo *reseed_counter* bằng 0.

10. *state*(*state_handle*) = { *S*, *P*, *reseed_counter*, *reseed_interval*, *state_length*, *block_length*, *field_size*, *strength*, *prediction_resistance_flag* }.

11. **Return** ("Thành công", *state_handle*).

C.5.2.2.4 Thay mầm mới khởi tạo MQ_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để thay mầm mới cho quá trình MQ_DRBG, sau khi được khởi tạo.

Reseed_MQ_DRBG_Instantiation (...):

Đầu vào: số nguyên *state_handle*, xâu *additional_input*.

Đầu ra: xâu *status*, trong đó *status* = "Thành công" hoặc một thông báo lỗi

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi).

2. Lấy các giá trị trạng thái phù hợp.

   2.1 *strength* = *state(state_handle).strength*.

   2.2 *state_length* = *state(state_handle).state_length*.

   2.3 *S* = *state(state_handle).S*.

3. *min_entropy* = **max**(128, *strength*).

4. (*status*, *entropy_input*) = **Get_entropy** (*min_entropy*, *min_length*, *max_length*).

5. If (*status* ≠ "Thành công"), then **Return** (thông báo lỗi).

6. *seed_material* = *S* || *entropy_input* || *additional_input*.

7. *S* = **Hash_df** (**pad8**(*seed_material*), *state_length*).

8. Cập nhật các giá trị trạng thái phù hợp.

8.1 *state(state_handle).S* = *S*.

8.2 *state(state_handle).reseed_counter* = 0.

9. **Return** ("Thành công").

C.5.2.2.5 Tạo các bit giả ngẫu nhiên sử dụng MQ_DRBG (...)

Quá trình sau đây hoặc tương đương được sử dụng để tạo các bit giả ngẫu nhiên.

MQ_DRBG (...):

Đầu vào: số nguyên (*state_handle*, *requested_strength*, *requested_no_of_bits*), xâu *additional_input*.

Đầu ra: xâu *status* trong đó *status* = "Thành công" hoặc một thông báo lỗi, xâu bit *pseudorandom_bits*.

Quá trình:

1. If trạng thái chưa sẵn sàng, then **Return** (thông báo lỗi, *Null*).
2. Lấy các giá trị trạng thái phù hợp.

   2.1 *strength* = *state(state_handle).strength*.

   2.2 *state_length* = *state(state_handle).state_length*.

   2.3 *block_length* = *state(state_handle).block_length*.

   2.4 *field_size* = *state(state_handle).field_*.size

   2.5 *S* = *state(state_handle).S*.

   2.6 *P* = *state(state_handle).P*.

   2.7 *reseed_counter* = *state(state_handle).reseed_counter*.

   2.8 *reseed_interval* = *state(state_handle).reseed_interval*.

   2.9 *prediction_resistance_flag* = *state(state_handle).prediction_resistance_flag*.
3. If (*requested_strength* > *strength*), then **Return** (thông báo lỗi, *Null*).
4. *temp* = xâu rỗng; $i = 0$.
5. If (*prediction_resistance_flag* = 1), then:

   5.1 *status* = **Reseed_MQ_DRBG_Instantiation** (*state_handle, additional_input*).

   5.2 If (status ≠ "Thành công"), then **Return** (*status, Null*).

   5.3 *additional_input* = *Null*.
6. If (*reseed_counter* ≥ *reseed_interval*), then:

   6.1 *status* = **Reseed_MQ_DRBG_Instantiation** (*state_handle, additional_input*).

   6.2 If (*status* ≠ "Thành công"), then **Return** (*status, Null*).

6.3 *additional_input* = *Null*.

7. If (*additional_input* ≠ *Null*), then:

   7.1 *additional_input* = **Hash_df** (**pad8**(*additional_input*), *state_length*).

   7.2 $S = S \oplus additional\_input$.

   7.3 *additional_input* = *Null*.

8. (*S*, *block*) = **Evaluate_MQ** (*state_length, block_length, field_size*, P, S).

9. *temp* = *temp* || *block*.

10. $i = i + 1$.

11. *reseed_counter* = *reseed_counter* + 1.

12. If (|*temp*| < *requested_no_of_bits*), then quay lại bước 5.

13. *pseudorandom_bits* = **Truncate** (*temp*, $i \times$ *block_length*, *requested_no_of_bits*).

14. Cập nhật các giá trị thay đổi vào trạng thái.

    14.1 *state(state_handle).S* = *S*.

    14.2 *state(state_handle).reseed_counter* = *reseed_counter*.

15. **Return** ("Thành công", *pseudorandom_bits*).

C.5.2.2.6 Đánh giá phương trình bậc hai đa biến sử dụng Evaluate_MQ (...)

Quá trình sau đây hoặc tương đương được sử dụng để đánh giá một hệ các phương trình bậc hai.

Evaluate_MQ (...):

Đầu vào: số nguyên (*state_length, block_length, field_size*), xâu bit (P, x).

Đầu ra: xâu bit (y, z).

Quá trình:

1. Chuyển đổi xâu bit thành vectơ của phần tử trường.

   1.1 *P_vec* = **field_vectơr**(*P, system_length, field_size*).

   1.2 *x_vec* = **field_vectơr**(*x, state_length, field_size*).

2. Khởi tạo các vectơ tạm thời *y_vec* và *z_vec*.

   2.1 *y_vec* = **field_vector**(0^{state_length}, *state_length*, *field_size*).

   2.2 *z_vec* = **field_vector**(0^{block_length}, *block_length*, *field_size*).

3. n = *state_length* / *field_size*.

4. m = *block_length* / *field_size*.

5. $t = 1$.

6. For $i = 1$ to n do

6.1 For $j = 1$ to n do

6.1.1 For $k = j$ to n do

6.1.1.1 $y\_vec[\,i\,] = y\_vec[\,i\,] \oplus (P\_vec[\,t\,] * x\_vec[\,j\,] * x\_vec[\,k\,])$.

6.1.1.2 $t = t + 1$.

6.2 If (*field_size* > 1), then:

6.2.1 For $j = 1$ to n do

6.2.1.1 $y\_vec[\,i\,] = y\_vec[\,i\,] \oplus (P\_vec[\,t\,] * x\_vec[\,j\,])$.

6.2.1.2 $t = t + 1$.

6.3 $y\_vec[\,i\,] = y\_vec[\,i\,] \oplus P\_vec[\,t\,]$.

6.4 $t = t + 1$

7. For $i = 1$ to m do

7.1 For $j = 1$ to n do

7.1.1 For $k = j$ to n do

7.1.1.1 $z\_vec[\,i\,] = z\_vec[\,i\,] \oplus (P\_vec[\,t\,] * x\_vec[\,j\,] * x\_vec[\,k\,])$.

7.1.1.2 $t = t + 1$.

7.2 If (*field_size* > 1), then:

7.2.1 For $j = 1$ to n do

7.2.1.1 $z\_vec[\,i\,] = z\_vec[\,i\,] \oplus (P\_vec[\,t\,] * x\_vec[\,j\,])$.

7.2.1.2 $t = t + 1$.

7.3 $z\_vec[\,i\,] = z\_vec[\,i\,] \oplus P\_vec[\,t\,]$.

7.4 $t = t + 1$.

8. Chuyển đổi vectơ về xâu bit.

8.1 $y = \mathbf{flatten}(y\_vec, n)$.

8.2 $z = \mathbf{flatten}(z\_vec, m)$.

9. **Return** (y, z).

CHÚ THÍCH 1 Xâu bit đầu vào x chứa *state_length* bit; các xâu bit đầu ra y, z chứa *state_length* và *block_length* bit tương ứng. Các hệ số phương trình bậc hai tạo ra y và z được lưu trữ trong xâu bit P. Xem C.5.2.3 và C.5.2.4 về mô tả định dạng của P.

CHÚ THÍCH 2 Phép nhân $*$ đề cập đến tích trên trường nhị phân $GF(2^{field\_size})$. Các phần tử của trường nhị phân được biểu diễn và xử lý như các xâu *field_size* bit sử dụng các đa thức tối thiểu cụ thể có trong bảng C.6. Xem C.5.2.3 để biết thêm

thông tin chi tiết về cách trình bày các phần tử trường thành xâu bit. Phần tử bổ sung trong $GF(2^{field\_size})$ được thực hiện sử dụng phép XOR giữa hai xâu bit có độ dài *field_size*.

C.5.2.2.7 Thêm độ bất định bổ sung vào trạng thái của MQ_DRBG (...)

Độ bất định bổ sung được thêm vào trạng thái của **MQ_DRBG (...)** bằng ba cách. Độ bất định bổ sung được thêm vào bằng cách:

1. Gọi đến hàm **Reseed_MQ_DRBG_Instantiation(...)**. Hàm này luôn gọi hàm thực thi phụ thuộc **Get_entropy (...)** để *min_entropy* = **max** (128, *strength*) bit mới của độ bất định được thêm vào trạng thái;

2. Sử dụng tính năng thay mầm mới tự động của **MQ_DRBG (...)**. Tính năng thay mầm mới tự động sẽ buộc gọi đến **Reseed_MQ_DRBG_Instantiation (...)** cho độ bất định mới khi *reseed_interval* khối ngẫu nhiên được xuất ra từ lần gọi cuối để thay mầm mới;

3. Thiết lập *prediction_resistance_flag* = 1 khi khởi tạo. Buộc gọi đến **Reseed_MQ_DRBG_Instantiation (...)** mỗi lần **MQ_DRBG(...)** được yêu cầu; hoặc

CHÚ THÍCH Tần suất gọi đến hàm **Get_entropy (...)** có thể gây ra giảm hiệu suất của bộ tạo bit ngẫu nhiên tất định này.

C.5.2.3 Định dạng biểu diễn phần tử trường

Một phần tử của $GF(2^{field\_size})$ là một đa thức đơn biến trên GF(2) mô-đun đa thức tối thiểu có trong bảng C.6. Do đó, phần tử trường được xác định duy nhất như danh sách các hệ số (là các bit) hoặc tương đương như một xâu *field_size* bit bao gồm các hệ số GF(2) được sắp xếp theo bậc giảm dần. Biểu diễn xâu này được sử dụng cho việc lưu trữ và tính toán trên các phần tử trường trong **MQ_DRBG(...)**.

CHÚ THÍCH 1 Ví dụ, $x^3 + x + 1$ trên trường $GF(2^4)$ được biểu diễn thành xâu bit 1011.

Vì các phần tử trường được lưu trữ và xử lý như các xâu bit, mọi hoạt động được thực hiện trên trường nhị phân $GF(2^{field\_size})$ được xem là hoạt động trên xâu bit. Do đó, phép nhân và phép cộng trường trên $GF(2^{field\_size})$ lấy đầu vào là hai xâu *field_size* bit và trả về một xâu *field_size* bit.

CHÚ THÍCH 2 Một phần tử trường là một bit đơn khi *field_size* = 1.

Bảng C.6 – Đa thức tối thiểu cho trường nhị phân $GF(2^{field\_size})$

field_size	Đa thức tối thiểu
1	$x + 1$
2	$x^2 + x + 1$
3	$x^3 + x + 1$
4	$x^4 + x + 1$
6	$x^6 + x + 1$
8	$x^8 + x^4 + x^3 + x^2 + 1$

C.5.2.4 Định dạng biểu diễn hệ thống các phương trình bậc hai đa biến

Xâu bit P có trong không gian trạng thái chứa các hệ số của hệ phương trình bậc hai đa biến được sử dụng trong **MQ_DRBG (...)**. Mỗi hệ số là một phần tử của trường nhị phân $GF(2^{field\_size})$ được xử lý như một xâu *field_size* bit. Tất cả các hệ số được nối lại để tạo ra xâu bit P được mô tả như sau. Do đó, kích thước P là:

$$system\_length = (n + m) * (n * (n + 3) / 2 + 1) * field\_size$$

Nếu *field_size* > 1 hoặc

$$system\_length = (n + m) * (n * (n + 1) / 2 + 1) * field\_size$$

với *field_size* = 1, trong đó n = *state_length* / *field_size* and m = *block_length* / *field_size* tương ứng là số lượng các phần tử trường có trong trạng thái bộ tạo bit ngẫu nhiên tất định và một khối đầu ra được tạo bởi **Evaluate_MQ (...)**.

Các hệ số của một phương trình bậc hai được nối theo thứ tự từ điển với bậc giảm dần. Cụ thể, hệ số của đơn thức x_1x_1 xuất hiện đầu tiên, tiếp theo là x_1x_2 và vân vân, đến hệ số x_1x_n. Hệ số của đơn thức x_2x_2 xuất hiện tiếp theo, sau đó là x_2x_3 và vân vân, cho đến cuối cùng đạt được hệ số bậc hai là $x_{n-1}x_n$. Sau đó các hệ số tuyến tính xuất hiện, bắt đầu với hệ số của đơn thức x_1 và kết thúc bằng hệ số x_n. Khi $field\_size = 1$, các hệ số tuyến tính được bỏ qua vì dư thừa các hệ số bậc hai có dạng $x_ix_i = x_i$. Chuỗi kết thúc với hệ số không đổi của phương trình bậc hai.

Các hệ số của hệ phương trình bậc hai được ghép nối theo thứ tự tuần tự, bắt đầu với các hệ số của phương trình đầu tiên và kết thúc với phương trình thứ $(n + m)$. Kết quả là xâu *system_length* bit dạng tham số hệ P.

C.5.2.5 Lựa chọn hệ các phương trình bậc hai đa biến phù hợp

Tồn tại các trường hợp yếu của hệ các phương trình bậc hai đa biến. Bậc của một phương trình bậc hai là cấp của ma trận với đầu vào a_{ij} là hệ số của đơn thức x_ix_j trong phương trình bậc hai với $i \neq j$ và bằng 0 với $i = j$. Đầu ra của một phương trình bậc hai bậc thấp có độ chệch tiệm cận 0. Tương tự, nếu một tổ hợp tuyến tính của các phương trình bậc hai có bậc thấp, thì đầu ra của các phương trình bậc hai đó có tương quan theo nghĩa là một tổ hợp tuyến tính của đầu ra có độ chệch tiệm cận 0.

Việc lựa chọn ngẫu nhiên hệ các phương trình bậc hai sử dụng trong một trường hợp của **MQ_DRBG (...)** có đặc tính là không phương trình bậc hai hoặc tổ hợp tuyến tính của phương trình bậc hai trọng số thấp có bậc thấp. Điều này được thực hiện bằng cách lấy các giá trị *min_rank* và *max_weight* từ bảng C.7 và kiểm tra theo hai tiêu chuẩn sau:

1. Tất cả các phương trình bậc hai đa biến có bậc ít nhất bằng *min_rank*, và
2. Tổng của tất cả các phương trình bậc hai đa biến với *max_weight* cao nhất có bậc ít nhất bằng *min_rank*.

Bảng C.7 – Hệ lựa chọn các tham số dựa trên *requested_strength* và *requested_block_length*

requested_strength	*requested_block_length*			
	1-112	113-128	129-192	193-256
1 - 80	*min_rank* = 106	*min_rank* = 30	*min_rank* = 30	*min_rank* = 30

	max_weight = 4	*max_weight* = 5	*max_weight* = 5	*max_weight* = 5
81 - 112	*min_rank* = 114 *max_weight* = 4	*min_rank* = 122 *max_weight* = 4	*min_rank* = 44 *max_weight* = 5	*min_rank* = 60 *max_weight* = 5
113 – 128	ERROR	*min_rank* = 122 *max_weight* = 4	*min_rank* = 60 *max_weight* = 5	*min_rank* = 60 *max_weight* = 5
129 – 192	ERROR	ERROR	*min_rank* = 192 *max_weight* = 4	*min_rank* = 124 *max_weight* = 4
193 - 256	ERROR	ERROR	ERROR	*min_rank* = 264 *max_weight* = 4

Phụ lục D
(quy định)
Hằng số cụ thể cho ứng dụng

D.1 Mô-đun mặc định cho MS_DRBG (...)

D.1.1 Giới thiệu về mô-đun mặc định MS_DRBG (...)

Mỗi mô-đun có dạng $n = pq$ với $p = 2p_1 + 1, q = 2q_1 + 1$ trong đó p_1 và q_1 là các số nguyên tố $(\lg(n)/2 - 1)$ bit.

D.1.2 Mô-đun mặc định n kích thước 1024 bit

Giá trị ở cơ số thập lục phân của mô-đun n là:

```
b66fbfda fbac2fd8 2eb13dc4 4fa170ff c9f7c7b5 1d55b214 4cc2257b 29df3f62
b421b158 0753f304 a671ff8b 55dd8abf b53d31ab a0ad742f 21857acf 814af3f1
e126d771 a61eca54 e62bfdb5 85c311b0 58e9cd3f aab758a5 e2896849 6ec1dd51
d0355aa1 55d4d912 6140dcfa b9b03f62 a5032d06 536d8574 0988f384 27f35885
```

D.1.3 Mô-đun mặc định n kích thước 2048 bit

Giá trị ở cơ số thập lục phân của mô-đun n là:

```
c11a01f2 5daf396a a927157b af6f504f 78cba324 57b58c6b f7d851af 42385cc7
905b06f4 1f6d47ab 1b3a2c12 17d14d15 070c9da5 24734ada 2fe17a95 e600ae9a
4f8b1a66 96661e40 7d3043ec d1023126 5d8ea0d1 81cf23c6 dd3dec9e b3fce204
5b9299bb cca63dee 435a2251 ad0765d4 9d29db2e f5aba161 279aeb5f 6899fe48
7973e36c 1fb13086 d9231b6b 925a8495 4ba0fbca fea844ea 77a9f852 f86915a4
e71bd0ba b9b269c3 9a7a827a 41311ffa 4470140c 8b6509fe 5dbd39e3 ec816066
2d036e13 0e07e233 06a39b18 db0e8efe 64418880 81ac3673 2b4091f6 63690d03
3b486d74 371a20fc 3e214bce 7ed0e797 5ea44453 cd161d32 e8185204 59896571
```

D.1.4 Mô-đun mặc định n kích thước 3072 bit

Giá trị ở cơ số thập lục phân của mô-đun n là:

```
c6046ba6 8beaa061 c468a9a7 4da34d64 21398c73 020837c7 d2a4042b dd9a7628
cab8022e 5bc4246f 75da8d26 03da8021 41c5d112 835e6bdb 57ed799e 28d6fa49
c3d0f5b5 f9776c14 0a901bf7 73ae3113 35d0470e da91b442 dbac621a cdd324e2
a70244d7 cb155adc 4b77dd94 fafe069d 5b5cc494 86e9fe61 c5081190 abb24f54
2d7d21e9 c90453c6 9ac63143 401d6b35 e456ea2f 64ae76f9 2df80328 b48f7962
d5c9b779 b2078496 7d374f02 06b8afbf 678d7f5f 36c3d84e c9e55c28 7ce5c668
17ee05b4 1059168f b5c5e2a3 6bc2f6ce 3b43bd14 56eebdd5 70ffe61e 5a7023a9
04d98f8a 96bfaf55 55a12f81 5561b401 63f3a50e a1e16a36 3f5cddd4 a1db275c
4fc2d650 d51f1e93 f5fd7631 ca45914f f6fe62a0 be55b997 5f6566bb 47e76276
f4e3b2eb 837bf0da 9d824687 042479a3 04147399 2d814a3a 7be7bc3d 06992df6
6c1d7d06 f8c1410e 2bbb573a 0e278e7a daa600f3 2577030e 95b73dd9 96b65f98
4740a485 e27138bd d5f02522 09bcf005 6640a1b3 b1dd97ad 7c187e04 01ba817d
```

D.1.5 Mô-đun mặc định n kích thước 7680 bit

Giá trị ở cơ số thập lục phân của mô-đun n là:

```
b60de1d0 fc4675b4 ff91e1cd 3e667f51 031cb4d0 d3cc9441 dbc56305 eaefbcc3
c4d852ee 8e5cc001 b0d52165 91dfd697 7cfa37ab 5f59b3b3 a20485df 1a73d42b
0a14f632 0f8d743e 7dbe912e 8fe2f7b9 ac883a41 e63503d7 94fcc184 e47e1dbb
5add17ed 19046ac3 e92be97e 4e9f6bd3 f849d16d 012e7c3d 640081a4 a626d304
f36c9d0e 28695216 32fcf527 758031a3 5bbb2b5a d1080a6d 029d3426 281b23ce

1c26a32b ed7a6649 bf5509ef bf9b723b d005d0d8 7eae0feb 657cb40a e4da922e
6ada3034 3e2e378b ec972615 469347f2 cbf3e786 6697e719 451802a9 e1ebc4df
59e74e1c 2303237c ab9b312f 50c6f146 a277d619 9c5dd596 c7ea9fc1 e0a82fde
c24acaff 0d0f531e 5c592014 d16707eb 8eff9ebf 21aef62c 7da8f071 55e48c47
7a56a512 cc2cb047 86a04860 0f99bd62 5c942d0b 0ac0107a 256cd091 decd9a90
9aa03a6f 09cfc722 96998f9b 1d7e2810 66d949ea 77346a64 716794a5 6459206f
d5a0fd3e 9886a4af e6ccd2ac 427a2cce f77d1dc6 639b8012 4dbc993d ea83b7ee
b60f92cd b497928f b157bfd6 f632290d a9b10214 814eea75 dd7fc4ad 342ff86f
4aa53619 0d115e3a 28991b60 ba5dd7ba e925a821 8cf11390 7c2dc96a 57734db8
b0b648c7 eb8b1def fd6e5e27 07646e72 d44e6d2b f52100eb 3c109cc8 1bb0c456
555e6656 ef89afec c60c1f50 3142d44b 14c67f7b 1628563b b501753f 00de58cf
83d1bdb4 6facee26 0c0cd133 f4be288e 19cff9d1 a1f94b02 b948cc2d b3362885
a624e11d 33e850b3 d9952155 be3fba38 433eb7f1 d36f9130 50f3751b 798d58c4
4ce2b955 0dd5660f fef89f13 6705517b 9f97349d 1de2970d af1d49b1 546a1fb3
579d4a52 95deb10c 717450d8 192b97f3 c3f2e961 b8ed3188 3890cc33 eb574d99
65f62b80 63cc4bcd 8d1993bf 5b945644 b8622571 c26c47ab 979b2c28 c51ecd8d
6824e200 0b3d4bf7 c215af3a 0806cad1 4a286f3d a42bf06e 3235d95e 9ca42fdb
87705d34 1c6d76b2 ea77917e ec8e87f0 64bae0e0 f74b6810 e9d91b74 e40b412e
96e27157 eb771b2a 1a2f6f8a c6abf7c1 382a8826 54c05309 b1ba6693 a8392047
bc6f003d c389c944 5f6de2c7 754da965 b13e60a6 66e15703 4e14faf6 93b729aa
d15f554a f13bba12 aa64f9d2 7896499c 8f05cbce cc3992b9 c04bf229 16b15797
2129afbe a3b3a282 8c29b16a 7a5e6f3e ff69e3cd 39c51bbc c45ec1a9 cc254e4b
026ea214 02e93013 b1950838 eb43b2f8 a2514ca4 fa37b302 8050908c beae9237
0bc6a7a1 a9cecd7b 8109b148 1f61ddd2 342570de eb16db12 00d41b6c 400327b7
92bf1ba0 ced0172e 631ae9f8 715f9eab 3c1c8b29 9f0fd186 6503c41e b25f5d11
```

D.1.6 Mô-đun mặc định n kích thước 15360 bit

Giá trị ở cơ số thập lục phân của mô-đun n là:

```
b43ba893 32383bdd 52c9337c 660b52b3 8f1a94f4 1e6c0eaf 9178504e 20a38777
3af3db9f e82c5abb fa5d581d 0167d869 a7c42103 14a778c2 f653f626 4cc1938d
7f3f0171 446cea97 22a8afb2 443c99e5 3c2f02e6 9f00515e 23546cc0 7107ef2c
e5c3eaa4 d1692033 b0c04ef4 d347d095 cdf38214 d81b1872 1b357b58 c61013bf
d1e4ee36 6f3c025b 532b0c81 6f91d4e3 2597514c fcee3a4a 02cb2918 34cea322
999d0059 4fc8f265 aacac767 683df781 0f939c05 fa440989 df631dcf dd76393a
90e82adb 34f27207 9b9870f5 f3f5d439 61b7b421 633baca9 5e89dbc1 97c42de1
ddee0a9f fa819754 0cabd97c 676fe76e 30d52360 be45a9ad edb4b16b 4e746978
0c5ddd67 3aaf680e 1375f9c1 6725547f 707aa3ea 5528e966 86b40837 55c13f02
8d1c1319 03abba1a 442d7e52 1ddf0adc 70fc941e b6b6da7a 25cd7118 f667dca2
5c1d9e59 8889cc6a 96e34c7b 78ca7102 6b19624c 66ff1291 7d664e36 ce1a8126
55536b97 cda457a1 cbfc49e2 b2b43165 b4f2a7a2 8fedf53f 9f1cc688 66132907
036eec2e 21e2c50f 0c85a3b5 a7c85cc6 844130a5 9b76d533 0989c434 acf86764
a6113e97 9320f0c4 3ca79aa3 187d5dcd 23df08a3 6dc4d077 afa71330 0422762c
663bb99a 7f2888a3 2b3b6137 5b3414ce de3737c0 3491dd55 2f60e509 21c66e29
442790ad 79c2a8e6 089fb32a 29023a78 08f4ec84 ee82db4b ce3b0ed0 ffcefada
```

bbc0c9ad 5495f960 87730739 6275a3ec 570f6a48 834b9299 20aefbab c9c1e9ca
45f89763 2947bd8f 1ddffd2a 7e865dea f15ff01c be2f540f 3111d569 87352a9a
de9a13d6 b310ba76 d48528f9 dcfe7ea2 2b70b396 e7059d39 4819112b 9b30786e
e97dcb7a 9cdb771b 4734030a e1ea98eb bd77efd2 e6712fdd e60e20f8 41cd0d21
c3f7a807 587d7dbb 80d5a873 b54e37dc 6897103e 2e236087 0aee1951 14cf0800
b090bf20 8a10b4b3 7f66b3f3 e52524f6 ac5d4e2b 54dcfa47 a9fc9d9b 69a37ce4
1543fd1d dda0dc71 c6386f98 99aec363 738071e4 28c02d43 8c550159 241b3086
9a999dc6 3cdfae38 364ace9d 93aaa91e 1eee30d7 838aa667 3123aed6 108e24cf
a4163507 a1b0081b f0cab8e8 d0aba749 33da6a79 1f4078fc 09216ba7 3c03a443

5b1d63d8 7a8b9fa0 faf26f2c b81508b3 1c969ff4 4fa2006a e3a6d9cf 29d79a20
4e126e89 2ee1c81e ea46314c 27d55b96 b2435255 90ad7cf9 013c9a3e ad64152d
55a5f275 4c35942c 59918799 32fc3b2c 45498f22 69f8ef41 4fca52eb 065f9a0b
7b360bed 15a3dbf7 fc03bf37 37d3ef7b e0730e8b d6ad9148 d25d68e5 9da78b2c
a92b7723 26daee4c 25870929 2d7ebe54 a54b2311 17a5527d dc5c08d8 5a7d030a
cccbd8d2 b00cf20d 1b2d5ee6 826bbb23 cc916ae3 9a566eec 7db11536 3e020567
e9a9e2d1 df8c1055 2d4b7d16 d6781ba2 a2d46f10 4834adac cfdbcaa6 25592eb2
1399cb25 e8e0d2c2 b39ccbae f527a7c2 546a3001 4a7101d6 21430c3b 32f10840
22344b86 9a4eb098 326797c9 e08f7b8b 8377d721 e512298d 63c9db31 9977dd97
1c312c33 bfa3d7fb 7c8dad81 7da0db97 b0ef9b66 97038700 16d2d93c 62da8cce
d8f600be fbd156f5 70f7ee51 1f6091c9 8b3a99df aab901d7 0117c57a 20840f24
b4133826 b6b9090d a4e956ef 908647d0 2e9cf730 538f569c cc918854 46eaef40
c8eca335 8fbf0af2 cc887ed6 f147770c 77fa6e8b a6f47665 5ccdd139 cd9be25c
16b1cf09 5dec9401 3ca27198 34afddfb 12958a6b 0b1dc5ec 11f76e6c 5aeb9a2b
4908c59c 803afd7a 9dda92c5 61838e55 436a57ae ca9a4a17 33c4b8ee d1fad817
2cd54a8c e24613f6 619d04ac 28866e3e 90fbb625 66cde710 5a822992 7348c007
5459d3f9 b7e117a3 2dc747d1 60e63688 640b1ce6 40763a76 61d74565 e0521dc0
a3461457 d7f31907 38cbcfa4 60621b65 5b81097b bc2c5b7a 8c573b3a 6b5a8134
89365ccb 7d238657 a58a6f3d ba964324 d7517993 ac685259 e8a36173 3729594b
557de123 281fb5a3 866febdb 3fdb7e4c cd93e1ad e63d3f06 840dbb5f 9d0650e2
2a97a3ac 9d083dd4 74085809 26b4e6a9 2785018b 38150d2e 948489dc 6462638b
d9c5b3ee 13acc840 febfd9b0 eab51c2a 6c9de9fc 037394a4 33eaf90a 519e98f5
3ff7b0e5 0132dc71 e860e9ce 33ee602e 8f7f3738 5245cc5f d1febdc0 68e12203
fc32cfaf c87e3f9a d3f94783 ff5d0330 e656561f 313c7190 4cd626f7 269348e8
a3837c68 5f9ddcee de3517ff 860fecbc f0461f58 f1f9d02e 29eb92fe e67d236a
546ff957 4746f65e be17f7a8 a0442d5b 565176c0 13767556 520644aa 1ec4925c
63cea832 6dca12ba 2ad5040a 06e81b64 f941854b 576d951f 247f3c4d d6424612
b918f18d 48e7c09d 6406acf6 c97b3f7d 3c4aaafc efe817a2 37088d93 ce436a4a
cef3875f 47b179e2 3dc07b0e e76fcdc9 518d05f6 82a06719 27a7f54c a01e7163
5d37e133 d308b288 e72fd746 d3fd6bdd 75cc0308 eb1e4040 2ea53331 24073131
623907a1 65b43490 e4eedd97 ffdecb8e e0e338a2 b1be8cae f756328f 6d0c114d
ade5340d bd02491b d62ff377 31e4fac7 d7211c9d 925f309f 221dc863 fc5dceec
4d689980 3db404a3 3636c4f5 b8f21e22 691bb39a b204c46f d8c177b6 b1046a23
bc5c22d2 4edf49da f737826f de218fc3 8ef8b8a1 c630c0fd f2e72bc7 c5940102
1fc358a3 81177f45 aa7d1cc9 c3d4de38 2d3e80ec f43e6059 efe5facc f5614119

Phụ lục E
(tham khảo)
Ví dụ về bộ tạo bit ngẫu nhiên bất định

E.1 Ví dụ tung đồng xu điển hình

E.1.1 Tổng quan

Ví dụ về bộ tạo bit ngẫu nhiên bất định này được xem là minh họa cho một số nhưng không phải là tất cả các yêu cầu đối với thiết kế của một bộ tạo bit ngẫu nhiên bất định.

Đối với một số ứng dụng, cần tạo ra các bit ngẫu nhiên bất định mà không cần sử dụng một bộ tạo bit ngẫu nhiên bất định đầy đủ chức năng. Vì việc tung đồng xu thường được coi là mô hình trực quan tiêu chuẩn cho việc tạo bit ngẫu nhiên và dưới các giả thiết khá đơn giản, việc bổ sung một số xử lý sau dựa trên toán học đối với kết quả tung đồng xu sẽ trả về kết quả đầu ra nhị phân hoàn toàn độc lập và không chệch. Tiêu chuẩn này cho phép tạo ra các bit ngẫu nhiên bằng cách sử dụng quá trình tung đồng xu. Trừ khi chỉ cần một số bit ngẫu nhiên tương đối nhỏ, quá trình như vậy sẽ không khả thi. Tuy nhiên, có thể có trường hợp mà phương pháp này là phù hợp; ví dụ: quá trình tạo không thường xuyên các mầm tương đối ngắn cho một bô tạo bit ngẫu nhiên tất định.

Tiêu chuẩn này đề xuất một bộ tạo bit ngẫu nhiên bất định yêu cầu bao gồm các thành phần và chức năng khác nhau có hiệu quả với bộ tạo bit ngẫu nhiên bất định hoạt động cùng với một hoặc nhiều nguồn bất định. Các thành phần này được thiết kế để tạo ra một quá trình không chỉ tạo ra các bit không chệch và độc lập, mà còn chịu được nhiều tình huống có hại khác thông qua việc bao gồm các cơ chế và tương tác an toàn giữa các thành phần. Tuy nhiên, bộ tạo bit ngẫu nhiên dựa trên việc tung đồng xu điển hình được mô tả ở đây không trực tiếp làm theo phương pháp này. Có thể lập luận rằng trong trường hợp bộ tạo bit ngẫu nhiên bất định dựa trên việc tung đồng xu là thích hợp, các mục tiêu được đáp ứng bởi các thành phần bộ tạo bit ngẫu nhiên bất định này rõ ràng là không cần thiết vì môi trường hoặc được thỏa mãn thông qua các bước thủ tục nhất định trong quá trình tung đồng xu. Các bước tung đồng xu được mô tả trong phụ lục này là một bộ tạo bit ngẫu nhiên bất định điển hình. Mục sau đây quy định một quy trình cho bộ tạo bit ngẫu nhiên bất định dựa trên việc tung đồng xu.

E.1.2 Mô tả quá trình cơ bản

Theo tiêu chuẩn này, bộ tạo bit ngẫu nhiên bất định dựa trên việc tung đồng xu cho phép một người liên tục tung đồng xu, gán một mặt của đồng xu cho kết quả "0" và mặt kia của đồng xu cho kết quả "1" và thực hiện quá trình không chệch Peres trên chuỗi đầu ra kết quả (xem E.1.5 để biết mô tả chi tiết về quá trình không chệch Peres). Giả sử rằng kết quả tung đồng xu là độc lập và phân bố đều (nhưng không nhất thiết là không chệch), quá trình không chệch Peres được biết là tạo ra đầu ra không chệch với số bit đầu ra được tạo ra gần bằng với độ bất định Shannon trong chuỗi tung đồng xu.

E.1.3 Mối quan hệ giữa các thành phần bộ tạo bit ngẫu nhiên bất định cơ bản

Bộ tạo bit ngẫu nhiên bất định dựa trên tung đồng xu điển hình được chấp nhận là một bộ tạo bit ngẫu nhiên bất định. Sự tương ứng giữa các tính năng của bộ tạo bit ngẫu nhiên bất định điển hình này và các tính năng của mô hình chức năng bộ tạo bit ngẫu nhiên bất định như sau:

1. Nguồn bất định chính: Một đồng xu. Lưu ý rằng do quá trình không chệch Peres, đồng xu không bắt buộc phải hoàn toàn không chệch, mặc dù giả định rằng mọi độ chệch là cố định.
2. Các đầu vào bổ sung: Một hoặc nhiều đồng xu bổ sung được sử dụng để tăng sự đảm bảo hoặc tin tưởng giữa nhiều bên.
3. Trạng thái bên trong: Các giá trị trong mỗi thanh ghi và biến được sử dụng để thực hiện quá trình không chệch Peres.

4. Hàm chuyển đổi trạng thái bên trong: Hàm sử dụng quá trình không chệch Peres để cập nhật nội dung của mỗi giá trị trung gian để đáp ứng cho mỗi lần tung thêm đồng xu.
5. Hàm tạo đầu ra: Hoạt động thu thập các bit không chệch kết quả từ quá trình không chệch Peres và gửi chúng đến ứng dụng yêu cầu. Nếu sử dụng một quá trình phức tạp hơn sử dụng nhiều đồng xu, hàm tạo đầu ra cũng có thể bao gồm XOR nhiều dòng đầu ra với nhau.
6. Kiểm tra chất lượng: Kiểm tra thống kê trên các mẫu tung đồng xu. Kiểm tra chất lượng cũng có thể bao gồm việc sử dụng một người quan sát để quan sát quá trình tung đồng xu và quá trình không chệch Peres.

E.1.4 Biến thể tùy chọn

Tùy thuộc vào tính sẵn sàng của tài nguyên tính toán, có thể mở rộng bộ tạo bit ngẫu nhiên bất định dựa trên tung đồng xu cơ bản để cung cấp sự tin tưởng giữa một số bên. Ví dụ, nếu một số bên tham gia có một phần trong đầu ra ngẫu nhiên được tạo ra, có thể không chấp nhận cho một bên tham gia có trách nhiệm duy nhất về việc tung đồng xu, đặc biệt nếu những bên tham gia ở các vị trí vật lý khác nhau và không thể quan sát việc tung đồng xu. Quá trình sau tạo ra một chuỗi nhị phân ngẫu nhiên gồm N bit (N là một số nguyên dương bất kỳ) với đặc tính không có bên tham gia nào hoặc liên minh nào có thể ảnh hưởng bất lợi đến kết quả cuối cùng và bất kỳ bên tham gia nào mong muốn một chuỗi ngẫu nhiên có thể tin tưởng vào tính ngẫu nhiên của kết quả cuối cùng. Quá trình này yêu cầu mỗi bên tham gia có quyền truy cập vào quá trình thực thi hàm băm mật mã ISO/IEC được phê duyệt.

1. Mỗi bên tham gia P_i tạo ra một chuỗi N bit x_i và một chuỗi M bit bổ sung y_i sử dụng quá trình tung đồng xu cơ bản. (Xem bình luận phía dưới về giá trị M.)
2. Mỗi bên tham gia P_i tính toán $z_i = H[x_i||y_i]$, trong đó H là hàm băm và "||" biểu diễn phép nối, và gửi z_i đến từng bên tham gia còn lại. Điều này gắn kết bên tham gia P_i với chuỗi ngẫu nhiên x_i.
3. Mỗi bên tham gia P_i gửi x_i và y_i cho từng bên tham gia còn lại.
4. Mỗi bên tham gia kiểm tra giá trị băm của các chuỗi x_i và y_i của từng bên tham gia còn lại, để đảm bảo rằng không có chuỗi ngẫu nhiên x_i của bên tham gia nào bị thay đổi theo yêu cầu của bên x_i.
5. Mỗi bên tham gia tính XOR tất cả các chuỗi x_i. N bit đầu tiên của kết quả là các bit ngẫu nhiên được tạo ra một cách đồng thuận.

Mục đích của M bit bổ sung là ngăn chặn các bên tham gia bằng cách xác định các chuỗi x_i của các bên tham gia khác sử dụng các bảng tra tính toán trước. Nếu không có những giá trị này, một kiểu tấn công như vậy có thể xảy ra nếu N đủ nhỏ. Do đó, M phải đủ lớn để khôi phục một tiền ảnh hàm băm có độ dài $M + N$ bit là không khả thi về mặt tính toán. Lưu ý rằng bất kỳ bên tham gia nào cũng có thể đảm bảm tính ngẫu nhiên của đầu ra cuối cùng mà không cần tin tưởng vào các bên tham gia khác.

E.1.5 Quá trình không chệch Peres

Một cách khác để giải quyết các vấn đề tiền ẩn trong việc cố gắng mô phỏng bộ tạo bit ngẫu nhiên lý tưởng là cố gắng giải quyết các vấn đề một cách trực tiếp. John von Neumann đã khám phá ra một quá trình trích xuất các bit không chệch từ các bit chệch, thậm chí không cần biết về độ chệch. Phương pháp này yêu cầu giả định rằng các bit là độc lập và độ chệch là một hằng số. Giả sử rằng một đồng xu được tung hai lần (trong đó H = mặt ngửa và T = mặt sấp); gán giá trị 1 nếu kết quả là HT và gán giá trị 0 nếu kết quả là TH. Nếu kết quả là HH hoặc TT, thì bỏ qua. Phương pháp đơn giản này dự kiến sẽ mất khoảng 4 lần tung đồng xu để tạo ra một bit không chệch, giả định rằng đồng xu hầu như không bị chệch.

Phương pháp von Neumann đã được thực hiện đệ quy (có nghĩa là đa cấp) bởi Yuval Peres [11] để trích xuất nhiều bit không chệch hơn. Ví dụ, có thể thực hiện một phép trích xuất bit cấp hai; thay vì loại bỏ HH và TT, gán HHTT bằng 1 và TTHH bằng 0. Do đó, HT-HH-TH-TT-HH-TT-TH là kết quả của xâu bit 1000. Có thể giải thích như sau. HT được gán bằng 1; HH bị bỏ qua; TH được gán bằng 0; TT-HH

không bị bỏ qua nhưng được gán bằng 0 khi trích xuất bit cấp 2; TT bị bỏ qua; và TH được gán bằng 0. Điều này sẽ dẫn đến một giới hạn để chỉ có duy nhất một lần xảy ra lỗi trích xuất các bit không chệch là khi xâu kết quả tung đồng xu giống hệt nhau.

Một phương pháp để khởi tạo bộ tạo bit ngẫu nhiên bất định là trung một đồng xu nhiều lần và sử dụng phương pháp không chệch Peres để kết nối đến độ bất định. Tiêu chuẩn này cho phép sử dụng phương pháp đó để thiết lập một giới hạn tối đa về độ khó cần thiết để khởi tạo bộ tạo bit ngẫu nhiên tất định đúng cách.

E.2 Ví dụ điốt nhiễu giả thuyết

E.2.1 Tổng quan

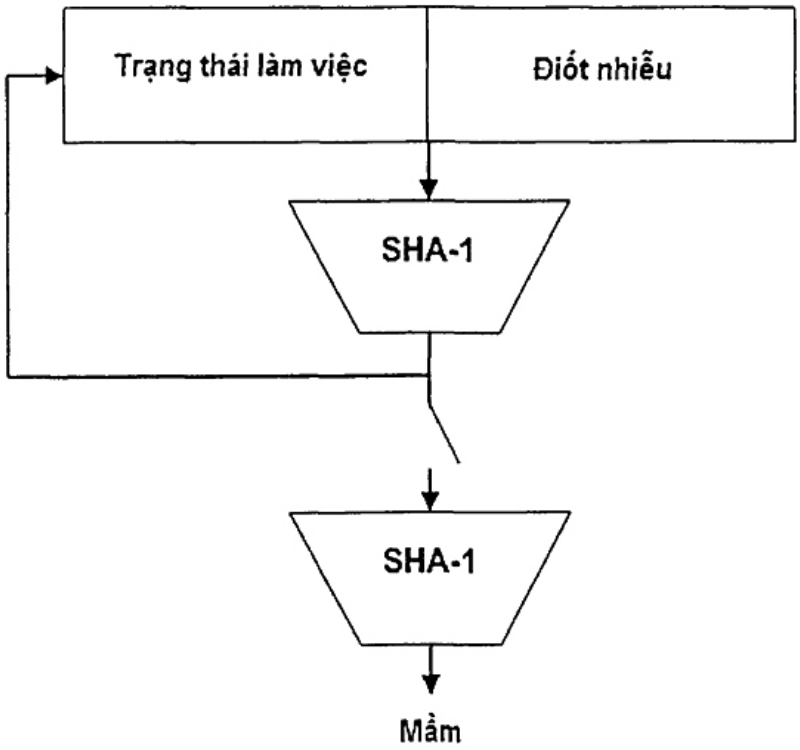
Phụ lục này trình bày một ví dụ lý thuyết về một bộ tạo bit ngẫu nhiên bất định thoả mãn các yêu cầu của tiêu chuẩn này. Mô tả này chỉ nhằm đưa ra một ý tưởng chung về một mẫu bộ tạo bit ngẫu nhiên bất định và không bao gồm tất cả thông tin chi tiết cần thiết để xác định và thực thi đầy đủ một bộ tạo bit ngẫu nhiên bất định đáp ứng tiêu chuẩn này.

E.2.2 Cấu trúc chung

Bộ tạo bit ngẫu nhiên bất định này được thiết kế để tạo ra các đầu ra ngẫu nhiên trong các khối 160 bit bằng cách xử lý đầu ra không xác định từ một điốt nhiễu. Thiết kế thoả mãn yêu cầu về độ mạnh tối thiểu của một bộ tạo bit ngẫu nhiên tất định được phê duyệt trong trường hợp nguồn bất định bị hỏng hoàn toàn (liên quan đến E.2.4 bên dưới). Cấu trúc tổng thể của bộ tạo bit ngẫu nhiên bất định được minh hoạ trong hình E.1. Trong hình vẽ này, "SHA-1" chỉ thuật toán băm an toàn.

Đối với mẫu bộ tạo bit ngẫu nhiên bất định này, việc thực hiện các thành phần chức năng được xác định trong mục 8 như sau:

1. Nguồn bất định chính: Một đi-ốt nhiễu.
2. Nguồn bất định bổ sung tuỳ chọn và các đầu vào khác: Chỉ điều khiển đầu vào; không có nguồn bất định bổ sung.
3. Trạng thái bên trong: Thanh ghi 160 bit cho trạng thái làm việc và giá trị khởi tạo SHA-1 160 bit hiện tại là tham số bí mật.
4. Hàm chuyển đổi trạng thái bên trong: Hàm băm tiêu chuẩn SHA-1 của trạng thái làm việc hiện tại và đầu vào nguồn bất định bổ sung, giá trị khởi tạo là tham số bí mật.
5. Hàm tạo đầu ra: Hàm băm tiêu chuẩn SHA-1 của trạng thái làm việc tạo ra đầu ra 160 bit.
6. Kiểm tra chất lượng: Kiểm tra với câu trả lời đã biết được kết hợp với hoạt động của các thành phần tất định sau khi thay thế các giá trị thay đổi bằng các giá trị đã biết.



Hình E.1 – Ví dụ về bộ tạo bit ngẫu nhiên bất định sử dụng một nguồn nhiễu vật lý để tạo bit

E.2.3 Hoạt động cụ thể

E.2.3.1 Nguồn bất định

Nguồn bất định là một đi-ốt nhiễu tạo ra các giá trị điện áp ngẫu nhiên theo phân bố chuẩn có giá trị trung bình 6.0 và độ lệch chuẩn 1.0 khi nhiệt độ môi trường nằm trong phạm vi thường. Trong mỗi khoảng thời gian lấy mẫu, điện áp đầu ra đi-ốt được số hóa và chuyển đổi thành giá trị bốn bit. Bảng E.1 cho thấy ánh xạ từ điện áp đến các giá trị số hóa cũng như xác suất của mỗi đầu ra.

Dữ liệu đầu vào nguồn bất định cho mỗi lần thay thế trạng thái làm việc được yêu cầu chứa ít nhất 160 bit độ bất định. Độ bất định H_2 của điốt nhiễu là $-log_2\left[\sum_{i=0}^{15} p_i^2\right] \cong 2{,}84067$. Nếu chúng ta căn cứ ước lượng độ bất định dựa trên độ bất định H_2, thì vì mỗi đầu ra điốt nhiễu chứa 2,84067 bit bất định, 160 bit bất định yêu cầu một chuỗi đầu ra gồm ít nhất $\left\lceil \frac{160}{2{,}84067} \right\rceil = 57$ bốn bit từ điốt nhiễu.

Tuy nhiên, min-entropy của điốt là $-log_2(max\{p_i\}) = -log_2(0{,}191462) = 2{,}38487$ (mong đợi nhỏ hơn độ bất định H_2). Vì sử dụng min-entropy làm thước đo, nên sẽ yêu cầu ít nhất $\left\lceil \frac{160}{2{,}38487} \right\rceil = 68$ mẫu từ điốt. Đầu vào này được số hóa và đệm thêm cho phù hợp với đặc điểm kỹ thuật của SHA-1.

Bảng E.1 – Danh mục các giá trị mong đợi trong 1000 mẫu

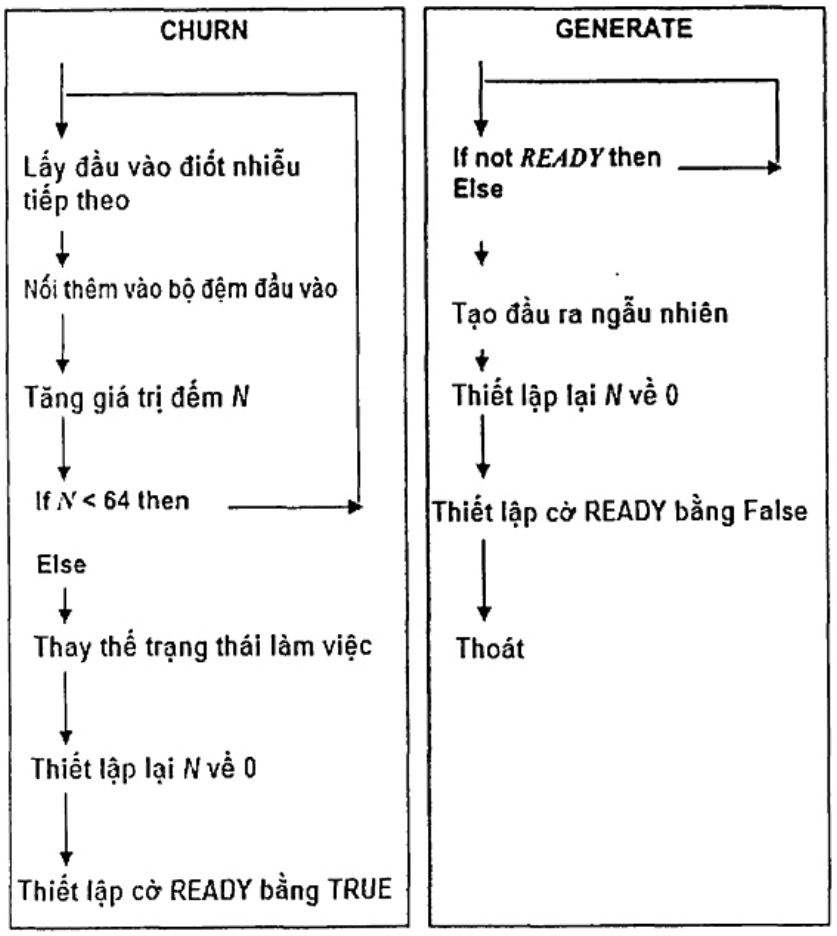
Điện áp lấy mẫu	Xác suất p_i	Đầu ra được số hóa
$-\infty < Z < 2{,}5$	0,000233	0000
$2{,}5 \leq Z < 3$	0,001117	0001
$3 \leq Z < 3{,}5$	0,004860	0010

$3,5 \leq Z < 4$	0,016540	0011
$4 \leq Z < 4,5$	0,044057	0100
$4,5 \leq Z < 5$	0,091848	0101
$5 \leq Z < 5,5$	0,149882	0110
$5,5 \leq Z < 6$	0,191462	0111
$6 \leq Z < 6,5$	0,191462	1000
$6,5 \leq Z < 7$	0,149882	1001
$7 \leq Z < 7,5$	0,091848	1010
$7,5 \leq Z < 8$	0,044057	1011
$8 \leq Z < 8,5$	0,016540	1100
$8,5 \leq Z < 9$	0,004860	1101
$9 \leq Z < 9,5$	0,001117	1110
$9,5 \leq Z < \infty$	0,000233	1111

E.2.3.2 Nhiệm vụ chính

Bộ tạo bit ngẫu nhiên bất định này gồm hai quá trình – một quá trình nền được gọi là "CHURN" thu thập dữ liệu nguồn bất định và thay thế trạng thái bên trong và một quá trình được gọi là "GENERATE" tạo ra đầu ra ngẫu nhiên 80 bit khi được yêu cầu. "CHURN" chạy bất cứ khi nào có sẵn tài nguyên bộ vi xử lý. "GENERATE" chỉ chạy khi được gọi bởi một ứng dụng yêu cầu. Cấu trúc này cho phép bộ tạo bit ngẫu nhiên bất định tiếp tục tích lũy thêm ảnh hưởng từ nguồn bất định theo thời gian, thay vì chỉ dựa vào mức ảnh hưởng của lượng độ bất định tối thiểu theo yêu cầu của tiêu chuẩn này. Hoạt động của "CHURN" và "GENERATE" được minh họa bởi các sơ đồ trong hình E.2.

Như minh họa trong hình E.2, CHURN hoạt động liên tục như một quá trình nền được cho phép bởi hệ thống tổng thể. Cần thực hiện một vòng lặp để thu thập đầu ra bốn bit (xem bảng E.1) từ điốt nhiễu, lưu chúng trong bộ đệm đầu vào 64 đầu ra (256 bit) và thay thế trạng thái làm việc khi bộ đệm này tràn. Ngay khi trạng thái làm việc đầu tiên được thay thế hoàn toàn, cờ "Ready – sẵn sàng" được thiết lập để chỉ ra rằng bộ tạo bit ngẫu nhiên bất định đã sẵn sàng để tạo ra đầu ra ngẫu nhiên, mặc dù CHURN vẫn tiếp tục thu thập các đầu ra nguồn bất định bổ sung và thay thế trạng thái làm việc. Cờ "Ready" được thiết lập trong quá trình khởi tạo bộ tạo bit ngẫu nhiên bất định. Lưu ý rằng dựa vào chương trình hoạt động của các nhiệm vụ hệ thống khác, việc thu thập, lưu bộ đệm và cập nhật dữ liệu này có thể không xảy ra liên tục.



Hình E.2 – Lưu đồ logic cho các nhiệm vụ bộ tạo bit ngẫu nhiên bất định CHURN và GENERATE

Như minh họa trong hình E.2, hoạt động GENERATE nhận các yêu cầu đầu ra ngẫu nhiên từ ứng dụng. Khi ứng dụng yêu cầu, trước tiên GENERATE kiểm tra cờ "Ready" để đảm bảo rằng trạng thái làm việc chứa đủ độ bất định. Nếu cờ "Ready" không được thiết lập, GENERATE đợi cho đến khi CHURN thiết lập cờ cho biết độ bất định đầy đủ trong trạng thái làm việc. Nếu cờ "Ready" được thiết lập, GENERATE sử dụng trạng thái bên trong hiện tại để tạo ra một đầu ra ngẫu nhiên 80 bit. GENERATE hoàn thành bằng sử dụng SHA-1 để băm nội dung của trạng thái làm việc và trính xuất 80 bit có trọng số cao trong kết quả đầu ra hàm băm 160 bit chính là 80 bit đầu ra ngẫu nhiên.

E.2.3.3 Nhiệm vụ hỗ trợ

Ngoài hai nhiệm vụ chính CHURN và GENERATE, bộ tạo bit ngẫu nhiên bất định này cũng có hai quá trình bổ sung để thực hiện các hàm hỗ trợ cần thiết. Đầu tiên là quá trình được gọi là "START" chạy một lần khi bắt đầu một phiên bật nguồn, khởi tạo bộ tạo bit ngẫu nhiên bất định bằng cách thiết lập các biến cần thiết cho trạng thái khởi tạo và nếu cần thiết thì tạo một tham số bí mật khởi tạo. Thứ hai là quá trình được gọi là "SEFTTEST" chạy trong quá trình khởi động nguồn, trong khoảng thời gian 24 giờ trong một phiên hoạt động và bất cứ khi nào người dùng yêu cầu, thực hiện một tập các phép kiểm tra chất lượng đối với bộ tạo bit ngẫu nhiên bất định.

Nhiệm vụ START bắt đầu bằng cách gọi SEFTTEST để thực hiện kiểm tra chất lượng. START thiết lập giá trị V làm khóa bí mật (để khởi động ban đầu) hoặc để lưu trạng thái làm việc nếu bộ tạo bit ngẫu nhiên bất định tiếp tục hoạt động.

Nhiệm vụ SELFTEST thực hiện một tập các phép kiểm tra chất lượng trong khi khởi động nguồn, trong khoảng thời gian 24 giờ trong một phiên hoạt động và bất cứ khi nào người dùng yêu cầu. Thông tin

chi tiết về các phép kiểm tra có trong 8.8 và bao gồm kiểm tra với câu trả lời đã biết đối với các thành phần tất định, các phép kiểm tra thống kê đối với nguồn bất định và các phép kiểm tra thống kê đối với đầu ra ngẫu nhiên của bộ tạo bit ngẫu nhiên bất định.

E.2.3.4 Kiểm tra chất lượng cụ thể

Bộ tạo bit ngẫu nhiên bất định này thực hiện một tập các phép kiểm tra chất lượng trong quá trình khởi động nguồn và tại các khoảng thời gian trong một phiên khởi động. Các phép kiểm tra như sau:

1. Kiểm tra chất lượng đối với các thành phần tất định bao gồm một bài kiểm tra với câu trả lời đã biết tổng thể đối với toàn bộ bộ tạo bit ngẫu nhiên bất định. Điều này yêu cầu thiết lập giá trị khởi tạo SHA-1 bằng một giá trị 160 bit xác định trước và trạng thái làm việc bằng một giá trị 160 bit xác định trước và thay thế các giá trị 64 bốn bit từ điốt nhiễu bằng một giá trị 256 bit xác định trước. Sau đó, bộ tạo bit ngẫu nhiên bất định tạo ra đầu ra 80 bit sử dụng các bước tương tự như hoạt động bình thường. Đầu ra kết quả được so sánh với giá trị tiền xác định và khai báo một lỗi nếu không trùng.
2. Có hai phép kiểm tra chất lượng đối với nguồn bất định này, được thiết kế để có xác suất lỗi loại 1 xấp xỉ bằng 10^{-4} hoặc nhỏ hơn. Phép đầu tiên bao gồm mẫu $N = 1000$ đầu ra, đếm số O_i là số lần xuất hiện của 16 đầu ra có thể và thực hiện phép kiểm tra "sự phù hợp" χ^2 đối với kết quả. Vì quy tắc chung yêu cầu số lượng dự kiến tối thiểu bằng năm trong mỗi danh mục, kết hợp ba trường hợp đầu ra đầu tiên với ba trường hợp đầu ra cuối cùng vào từng danh mục kết hợp, tổng cộng có 12 danh mục. Xác suất đầu ra và số lượng dự kiến E_i có trong bảng E.2. Thống kê $\chi^2_{(11)} = \sum_{i=1}^{12} \frac{(O_i - E)^2}{E_i}$ có phân bố χ^2 với mười một bậc tự do khi nguồn bất định hoạt động chính xác. Nguồn bất định được xác định là lỗi khi kiểm tra chất lượng nếu số liệu thống kê này vượt quá 37,4. Phép kiểm tra thứ hai tìm kiếm hành vi được biết là đã tăng khả năng khi nhiệt độ nằm ngoài phạm vi cho phép. Tình trạng này dẫn đến các khoảng điện áp thường xuyên ở mức thấp của khoảng giá trị. Phép kiểm tra này sử dụng cùng một mẫu 1000 đầu ra và tìm kiếm các lần xuất hiện của hai hoặc nhiều đầu ra 0000 liên tiếp. Nếu một chuỗi như vậy xuất hiện trong mẫu, điều kiện lỗi sẽ được khai báo. Lưu ý rằng phép kiểm tra này bổ sung cho phép kiểm tra đầu tiên; phép kiểm thử đầu tiên tìm kiếm sự mâu thuẫn chung của đầu ra với mô hình thống kê đặc trưng, trong khi phép kiểm tra thứ hai tìm kiếm một loại đầu ra cụ thể đã biết có liên quan đến một điều kiện lỗi đã biết.
3. Mặc dù kiểm tra này có thể được thực hiện như một phép kiểm tra chất lượng tùy chọn, bản thân phép kiểm tra có một quá trình bắt buộc để thực hiện như sau:
   a) Thu thập và nối 250 giá trị 80 bit liên tục (tổng cộng 20000 bit) từ bộ tạo bit ngẫu nhiên bất định.
   b) Thực hiện phép kiểm tra đơn bit, poker, loạt và loạt dài được quy định trong 8.8.5.
   c) Nếu một phép kiểm tra bị lỗi, thì điều kiện lỗi được khai báo.

Ngoài ra, mỗi đầu ra 80 bit ngẫu nhiên được tạo ra khi hoạt động bình thường phải được kiểm tra; nếu nó trùng với bất cứ chuỗi nào trong bốn chuỗi gồm toàn một, toàn không hoặc xen kẽ một và không thì điều kiện lỗi được khai báo.

Bảng E.2 – Danh mục các giá trị mong đợi trong 1000 mẫu

Danh mục	Đầu ra được số hóa	Số lượng dự kiến

1	0000-0010	6,21
2	0011	16,54
3	0100	44,06
4	0101	91,85
5	0110	149,88
6	0111	191,46
7	1000	191,46
8	1001	149,88
9	1010	91,85
10	1011	44,06
11	1100	16,54
12	1101-1111	6,21

E.2.4 Hậu quả của thiết kế không phù hợp

Bộ tạo bit ngẫu nhiên bất định này được thiết kế để thỏa mãn yêu cầu thiết kế không kém an toàn hơn so với một bộ tạo bit ngẫu nhiên tất định được phê duyệt trong trường hợp nguồn bất định hoàn toàn bị lỗi theo cách không thể phát hiện được hoặc bị kiểm soát bởi kẻ tấn công.

E.2.5 Ví dụ sửa đổi

Xem xét ví dụ được thảo luận trong E.2.3.1 đến E.2.3.4 nhưng giả sử rằng điện áp của điốt nhiễu không được số hóa thành các giá trị bốn bit mà thành các giá trị một bit. Tương tự như ví dụ ban đầu, xem các giá trị một bit này là việc thực hiện các biến ngẫu nhiên có giá trị nhị phân độc lập. Không giống như các biến ngẫu nhiên trong ví dụ ban đầu, các biến ngẫu nhiên tương ứng (tùy thuộc vào điốt nhiễu) tồn tại độ chệch nhỏ. Tùy thuộc vào độ chệch, các giá trị được số hóa có thể được sử dụng trực tiếp. Ngoài ra, các cặp không chồng lấn được XOR với nhau hoặc áp dụng thuật toán không chệch von Neumann. Phép kiểm tra đơn bit (với giới hạn không chấp nhận hợp lý) chính là kiểm tra chất lượng đối với đầu ra của điốt nhiễu. Các bit tạo ra được lưu trữ cho đến khi kiểm tra chất lượng đối với nguồn bất định được thông qua. Sau đó, các bit có thể được xuất ra. (Nếu một kiểm tra chất lượng lỗi thì ít nhất các bit ngẫu nhiên được lưu trữ sẽ bị xóa.) Giả sử thêm rằng điốt nhiễu và đầu ra của nó được bảo vệ chống lại kẻ tấn công tiềm năng.

E.3 Ví dụ chuyển động của con trỏ chuột

Một nguồn bất định trên máy tính cá nhân là chuyển động của con trỏ chuột. Các chuyển động của con trỏ chuột có thể được lấy mẫu khá thường xuyên (ví dụ một trăm lần mỗi giây) và một số thuộc tính (ví dụ: vị trí hoặc vận tốc tuyệt đối) có thể được số hóa và tích lũy như đầu vào bất định của bộ tạo bit ngẫu nhiên bất định (hoặc đầu vào bổ sung cho bộ tạo bit ngẫu nhiên tất định). Để làm được điều này, có một yêu cầu để mô hình phân bố xác suất của phép đo chuyển động của con trỏ chuột. Mô hình này sẽ cung cấp một khái niệm chấp nhận được về tỷ lệ độ bất định có thể mong đợi, cũng như một phương tiện để đo lường hoạt động đúng đắn của nguồn. Nguồn này phụ thuộc vào hành vi của người dùng. Do đó, nó có thể thay đổi theo thời gian khi hoạt động của người dùng thay đổi (ví dụ: thu được độ bất định lớn hơn trong khi lướt web trong thời gian nghỉ trưa). Sự biến đổi theo thời gian có thể

được giảm thiểu bằng cách lấy mẫu trong thời gian dài và tích lũy, hoặc bằng một thói quen khởi tạo, nhắc nhở người dùng di chuyển chuột nhiều hơn trong một khoảng thời gian ngắn. Tuy nhiên, nguồn bất định này không hữu ích trong mọi tình huống; đặc biệt, một máy chủ chạy không có mặt con người tại bàn điều khiển sẽ đem lại nhiều sự lựa chọn tốt hơn về độ bất định.

Phần mềm cho phép hệ điều hành giao tiếp với chuột thông qua trình điều khiển con trỏ chuột. Công việc của nó là tìm, điều khiển phần mềm giống như một trình điều khiển chuột trong khi giao tiếp với con trỏ chuột bất cứ cách nào nào nó cần. Trình điều khiển chuột ẩn thông tin chi tiết về cách nó giao tiếp với con chuột và con trỏ chuột trông như thế nào. Giả sử rằng bộ tạo bit ngẫu nhiên bất định giao tiếp với con trỏ chuột thông qua trình điều khiển, thông tin về trình điều khiển vô cùng quan trọng trong việc xác định tỷ lệ độ bất định của nguồn.

Đơn vị cơ bản để phát hiện chuyển động của con trỏ chuột được gọi là "mickey". Thông thường, một mickey là khoảng 1/200 inch tuỳ thuộc vào phần cứng của chuột. Một số phần cứng chuột nhạy hơn các phần cứng khác.

Nếu chuột di chuyển nhỏ hơn một mickey, thì sẽ không được tính là một chuyển động.

Về cơ bản có ba loại sắp xếp vật lý để móc chuột vào một hệ thống:

1. Bus của con trỏ chuột móc vào một số đầu nối đến bus hệ thống, có thể đọc tín hiệu hiện tại từ chuột. Điều này cho phép con trỏ chuột không cần bộ xử lý. Trong một số thiết kế, trình điều khiển chuột gửi yêu cầu đến chuột ít nhất 30 lần mỗi giây. Trong các thiết kế khác, đầu nối chờ đến khi tín hiệu từ chuột thay đổi, sau đó gây ra ngắt để trình điều khiển chuột có thể đọc từ đầu nối.
2. Một con chuột nối tiếp giao tiếp qua một số cổng nối tiếp. Trong quá khứ có thể là cổng RS232, nhưng hiện nay chuột USB được sử dụng phổ biến hơn. Về cơ bản, bộ xử lý của con trỏ chuột phát hiện chuyển động và tạo ra một gói, thường là 3-5 byte xác định số lượng mickey đã di chuyển theo hướng X và Y kể từ gói cuối cùng và trạng thái nút của nó. Chuột có ba nút và nút có bánh xe để cuộn nhanh hơn, bao gồm các byte thừa để gửi trạng thái bổ sung đó.
3. Chuột PS/2 hoạt động tương tự với chuột nối tiếp; nó có bộ xử lý riêng và giao tiếp với bộ điều khiển bàn phím/chuột trên PC bằng cách gửi các gói tin không đồng bộ. Tài liệu đã chỉ ra rằng tỷ lệ mẫu con trỏ chuột có thể thấp hơn 10 mẫu mỗi giây.

Trình điều khiển chuột theo dõi vị trí con trỏ chuột, trạng thái nút và bất kỳ sự kiện nào khác có sẵn. Có thể thoải mái làm bất cứ điều gì để ngắt tín hiệu chuột, ví dụ: cho chuột một lượng quán tính nhất định. Một số trình điều khiển thực hiện điều chỉnh làm xiên các tín hiệu nhận được. Ví dụ: một số trình điều khiển nhất định cho chuột bút chì trên máy tính xách tay sẽ phản hồi người dùng cầm đầu bên phải trong vài giây bằng cách khiến con trỏ di chuyển sang trái khi người dùng nhả chuột. Các trình điều khiển khác cập nhật vị trí thường xuyên đến mức chắc chắn xảy ra một số tăng tốc phần mềm (tức là trình điều khiển đang thêm vào các tín hiệu nhận được từ con chuột). Vấn đề ở đây là nếu các nhà thiết kế bộ tạo bit ngẫu nhiên bất định dựa vào trình điều khiển chuột để truyền các chuyển động của chuột sử dụng làm độ bất định, họ cần phải biết rằng dữ liệu có thể được điều chỉnh theo một cách nào đó và họ cần xây dựng mô hình độ bất định cho phù hợp.

Phụ lục F
(tham khảo)
Các lưu ý về an toàn

F.1 Mô hình tấn công

Xác định mục tiêu của kẻ tấn công là một phần quan trọng trong việc xác định các mô hình tấn công và các mối đe dọa khác nhau mà thiết kế của một bộ tạo bit ngẫu nhiên cố gắng giải quyết. Mô hình hóa bộ tạo bit ngẫu nhiên như là hộp của Kerckhoff, giả thiết rằng kẻ tấn công có đầy đủ thông tin về thiết kế hệ mật và biết mọi thứ ngoại trừ thông tin bí mật chứa trong đó. Kẻ tấn công cố có được một lợi thế bằng một hoặc nhiều cách sau đây:

1. Thu thập thông tin liên quan đến đầu ra bí mật/riêng từ một bộ tạo bit ngẫu nhiên, ví dụ: độ chệch để ưu tiên hoặc đơn giản hóa các nỗ lực đoán giá trị.
2. Thu thập thông tin liên quan đến độ bất định sử dụng trong một bộ tạo bit ngẫu nhiên. Bao gồm thông tin liên quan đến mầm bí mật được sử dụng trong một bộ tạo bit ngẫu nhiên tất định (có thể tiết lộ thông tin về các kết quả đầu ra đượ tạo ra trước đó và/hoặc sắp tới) cũng như khả năng kiểm soát nguồn bất định của bộ tạo bit ngẫu nhiên bất định.
3. Thu thập thông tin liên quan đến phần bí mật của trạng thái bên trong trong một bộ tạo bit ngẫu nhiên.
4. Kẻ tấn công có thể cố gắng điều khiển bộ tạo bit ngẫu nhiên để gây ra các hiệu ứng khiến cho đầu ra dễ dự đoán hơn hoặc làm giảm độ bất định của đầu vào hoặc trạng thái bên trong.
5. Khả năng kháng lại của bộ tạo bit ngẫu nhiên chống lại việc điều khiển phải phù hợp với độ mạnh an toàn mà ứng dụng yêu cầu.

CHÚ THÍCH Không được coi là tấn công nếu kẻ tấn công có thể phân biệt được bộ tạo bit ngẫu nhiên chọn từ không gian mẫu có thay thế hoặc không thay thế, nghĩa là xác định đầu ra lặp hoặc không lặp lại trong căn bậc hai kích thước của không gian mẫu. Giả sử rằng kẻ thông tin biết thông tin này như là một phần của bản đặc tả thiết kế. Như đã đề cập ở phần trước, các tấn công căn bậc hai chủ động dựa vào các va chạm trong trạng thái bên trong có thể là vấn đề đối với các ứng dụng có yêu cầu bảo mật cao nếu không gian trạng thái bên trong nhỏ hơn hai lần độ mạnh an toàn mong muốn.

F.2 Độ an toàn của hàm băm

Độ mạnh an toàn của một hàm băm đối với các bộ tạo bit ngẫu nhiên bằng kích thước khối đầu ra. Nếu không có lỗ hổng do va chạm (ví dụ: hàm băm được sử dụng làm một phần tử trong thiết kế bộ tạo bit ngẫu nhiên) và hàm không có nghịch đảo thì độ mạnh bằng kích thước khối đầu ra. Tuy nhiên, CHÚ THÍCH rằng khi hàm băm được sử dụng làm một phần tử trong một dịch vụ mật mã có khả năng dễ bị va chạm thì độ mạnh của đầu ra hàm băm được đánh giá bằng một nửa kích thước của khối đầu ra dựa vào nghịch lý ngày sinh.

F.3 Lựa chọn thuật toán và kích thước khóa

F.3.1 Tổng quan

Mục này cung cấp hướng dẫn cho việc lựa chọn các thuật toán và kích thước khóa phù hợp. Nhấn mạnh tầm quan trọng của việc có được hệ thống mật mã với các thuật toán và kích thước khóa phù hợp để cung cấp bảo vệ đầy đủ cho 1) vòng đời dự kiến của hệ thống và 2) dữ liệu được bảo vệ bởi hệ thống đó trong vòng đời dự kiến của dữ liệu. Cũng bao gồm sự cần thiết khi lựa chọn bộ tạo bit ngẫu nhiên thích hợp nhằm hỗ trợ các thuật toán mật mã.

F.3.2 Độ mạnh thuật toán tương đương

Các thuật toán mật mã cung cấp các "độ mạnh" an toàn khác nhau tùy thuộc vào thuật toán và kích thước khóa sử dụng. Hai thuật toán được coi là có độ mạnh bằng nhau với kích thước khóa cho trước (X và Y) nếu khối lượng công việc (được tính bằng các phép toán) cần để "phá vỡ thuật toán" hoặc xác định khóa (với kích thước khóa cho trước) xấp xỉ tương đương bằng cách sử dụng tài nguyên đã cho. Độ mạnh của một thuật toán (cũng được gọi là hệ số hoạt động) đối với một kích thước khoá cho trước được định nghĩa là lượng công việc cần thực hiện để thử tất các các khoá đối với một thuật toán đối xứng với kích thước khoá là "X" mà không có các tấn công rút gọn khác (tức là kiểu tấn công hiệu quả nhất để thử tất cả các khoá có thể). Trong trường hợp này, kiểu tấn công tốt nhất được gọi là tấn công vét cạn. Một thuật toán có khoá gồm "Y" bit nhưng có độ mạnh tương đương với một khoá gồm "X" bit của thuật toán đối xứng đó được gọi là cung cấp "X bit an toàn" hoặc cung cấp "độ mạnh X bit". Một thuật toán đối xứng cung cấp độ mạnh X bit mất trung bình $2^{X-1}T$ để tấn công, trong đó T là thời gian cần để thực hiện một phép mã hoá đối với một giá trị rõ và so sánh kết quả với giá trị mã tương ứng.

Đánh giá độ mạnh an toàn của một thuật toán mật mã có thể là không đáng kể. Ví dụ, xem xét TDEA. TDEA sử dụng ba khóa 56 bit ($K1, K2$ và $K3$). Nếu mỗi khóa này được tạo ra một cách độc lập, thì đượ cgọi là ba tùy chọn khóa hoặc ba khóa TDEA (3TDEA). Tuy nhiên, nếu $K1$ và $K2$ được tạo ra một cách độc lập, và $K3$ được đặt bằng $K1$, thì gọi là hai tùy chọn khóa hoặc hai khóa TDEA (2TDEA). Người ta có thể mong đợi rằng 3TDEA sẽ cung cấp $56 \times 3 = 168$ bit độ mạnh. Tuy nhiên, có một tấn công vào 3TDEA làm giảm sức mạnh của công việc có liên quan đến việc vét cạn khóa 112 bit. Đối với 2TDEA, nếu vét cạn là tấn công hiệu quả nhất, thì độ mạnh của 2TDEA sẽ là $56 \times 2 = 112$ bit. Đây chính là trường hợp kẻ tấn công chỉ có một vài cặp rõ và mã tương ứng. Tuy nhiên, nếu kẻ tấn công có thể thu được 2^{40} cặp như vậy, thì 2TDEA có độ mạnh xấp xỉ tương đương một thuật toán 80 bit.

Kích thước khóa tương đương được đề xuất trong mục này dựa trên các đánh giá được thực hiện khi công bố tiêu chuẩn này. Những tiến bộ trong các thuật toán phân tích thừa số nguyên tố, tiến bộ trong tấn công logarit rời rạc nói chung và tấn công logarit rời rạc trên đường cong elliptic và tính toán lượng tử có thể ảnh hưởng đến các giá trị tương đương này trong tương lai. Các tấn công hoặc kỹ thuật mới, cải tiến có thể được phát triển để loại bỏ một số thuật toán hiện tại không còn an toàn. Trong trường hợp tính toán lượng tử, tất cả các kỹ thuật bất đối xứng có thể không còn an toàn nữa. Các đánh giá định kỳ phải được thực hiện để xác định liệu các giá trị tương đương đã nêu có cần được sửa đổi hay không (ví dụ: kích thước khóa phải được tăng lên) hoặc liệu các thuật toán có còn an toàn không.

Khi lựa chọn một thuật toán mật mã mã khối (ví dụ: AES hoặc TDEA), kích thước khối cũng là một yếu tố cần được xem xét, vì độ an toàn được cung cấp phụ thuộc vào kích thước khối, sau khoảng căn bậc hai kích thước khối (ví dụ 2^{32} khối đối với DES và 2^{64} khối đối với AES) được mã hóa, thông tin về bản rõ sẽ bị rò rỉ.

Bảng F.1 cung cấp các hướng dẫn tương đương cho một số thuật toán mật mã và hàm băm.

1. Cột 1 cho biết độ mạnh an toàn (tính bằng bit) được cung cấp bởi thuật toán và kích thước khóa trong một hàng cụ thể.
2. Cột 2 cung cấp các thuật toán khóa đối xứng cung cấp mức độ bảo mật được chỉ định (ở mức tối thiểu).
3. Cột 3 cung cấp các hàm băm tương đương được quy định cho mức độ an toàn cho trước chống lại tấn công va chạm.
4. Cột 4 cho biết kích thước các tham số liên quan đến các tiêu chuẩn sử dụng logarit rời rạc và số học trường hữu hạn, trong đó L là kích thước mô-đun p và N là kích thước nhóm con bậc q. L thường được coi là kích thước khóa cho thuật toán, mặc dù L thực sự là kích thước khóa của khóa công khai và N là kích thước khóa của khóa bí mật.

5. Cột 5 xác định giá trị cho k (kích thước mô-đun n) đối với thuật toán RSA hoặc RW. Giá trị k là tham số an toàn chính và thường được coi là kích thước khóa, mặc dù khóa công khai cũng chứa một số mũ công khai và khóa bí mật chứa (ít nhất ở một dạng nào đó) một số mũ bí mật.
6. Cột 6 xác định giá trị của f (kích thước của n, trong đó n là bậc của điểm cơ sở G) đối với các thuật toán logarit rời rạc sử dụng số học đường cong elliptic. Giá trị f là tham số an toàn chính và thường được coi là kích thước khóa.

Bảng F.1 – Độ mạnh an toàn tương đương

Độ mạnh an toàn (tính bằng bit)	**Các thuật toán khóa đối xứng**	**Các ham băm (va chạm)**	**FFC**	**IFC**	**ECC**
80	TDEA 2 khóa	SHA-1	$L = 1024$ $N = 160$	$k = 1024$	$f = 160$
112	TDEA 3 khóa	SHA-224	$L = 2048$ $N = 224$	$k = 2048$	$f = 224$
128	AES-128	SHA-256	$L = 3072$ $N = 256$	$k = 3072$	$f = 256$
192	AES-192	SHA-384	$L = 7680$ $N = 384$	$k = 7680$	$f = 384$
256	AES-256	SHA-512	$L = 15360$ $N = 512$	$k = 15360$	$f = 512$

F.3.3 Lựa chọn bộ tạo bit ngẫu nhiên tất định phù hợp

Các thuật toán có độ mạnh và kích thước khóa khác nhau có thể được sử dụng cùng nhau vì lý do hiệu suất, tính sẵn sàng hoặc khả năng tương tác, với điều kiện phải cung cấp sự bảo vệ đầy đủ. Nhìn chung, thuật toán và kích thước khóa yếu nhất được sử dụng để cung cấp bảo vệ về mật mã xác định độ mạnh của sự bảo vệ. Các ngoại lệ đối với quy tắc này yêu cầu sự phân tích chuyên sâu. Việc xác định độ mạnh an toàn cung cấp cho thông tin được bảo vệ bao gồm phân tích không chỉ đối với (các) thuật toán và kích thước khóa được sử dụng để mã hóa thông tin, mà còn đối với thuật toán và kích thước khóa bất kỳ liên quan đến việc thiết lập (các) khóa được dùng để bảo vệ thông tin, bao gồm cả khóa được sử dụng bởi các bộ tạo bit ngẫu nhiên và giao thức truyền thông.

Thông thường, một tổ chức sẽ chọn các dịch vụ mật mã cần thiết cho một ứng dụng cụ thể. Sau đó, dựa trên vòng đời bảo mật của dữ liệu và số năm mà hệ thống sẽ được sử dụng theo dự đoán, một bộ thuật toán và kích thước khóa được chọn đủ để đáp ứng các yêu cầu này. Sau đó, tổ chức thiết lập hệ thống quản lý khóa, bao gồm các sản phẩm mật mã cung cấp các dịch vụ theo yêu cầu của ứng dụng.

Các tổ chức mua hệ thống phải xem xét thời gain hoạt động tiềm năng của hệ thống. Các tổ chức nên chọn các thuật toán dự kiến sẽ an toàn trong toàn bộ thời gian tồn tại của hệ thống hoặc đảm bảo rằng thuật toán và kích thước khóa có thể dễ dàng được cập nhật.

F.4 Độ an toàn của bộ tạo bit ngẫu nhiên tất định dựa trên mã khối

Tiêu chuẩn này mô tả hai bộ tạo bit ngẫu nhiên tất định dựa trên mã khối. Một bộ sử dụng mã khối ở chế độ CTR; bộ còn lại sử dụng mã khối ở chế độ OFB. Không có sự khác biệt về độ an toàn thực tế giữa hai bộ tạo bit ngẫu nhiên tất định này ; chế độ CTR đảm bảo rằng chu kỳ ngắn không xảy ra trong một yêu cầu đầu ra đơn lẻ, trong khi đó chế độ OFB đảm bảo rằng chu kỳ ngắn sẽ xảy ra với xác suất cực kỳ thấp. Chế độ OFB đòi hỏi ít giả định hơn đối với mã khối, nhưng sự an toàn của cả hai bộ tạo bit ngẫu nhiên tất định liên quan một cách đơn giản và rõ ràng đến sự an toàn của mã khối trong những ứng dụng của nó.

Đây là sự khác biệt cơ bản giữa các bộ tạo bit ngẫu nhiên tất định này với bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm, trong đó sự an toàn của bộ tạo bit ngẫu nhiên tất định dựa trên các đặc tính giả ngẫu nhiên không tạo nên một phần thông thường của các yêu cầu đối với hàm băm. Một tấn công đối với bộ tạo bit ngẫu nhiên tất định dựa trên hàm băm bất kỳ không nhất thiết thể hiện một điểm yếu trong hàm băm; tuy nhiên, đối với các cấu trúc dựa trên mã khối này, điểm yếu của bộ tạo bit ngẫu nhiên tất định liên quan trực tiếp đến điểm yếu trong mã khối.

Vì không có sự khác biệt về tính an toàn thực tế giữa hai loại bộ tạo bit ngẫu nhiên tất định dựa trên mã khối, lựa chọn giữa hai cấu trúc hoàn toàn là vấn đề thuận tiện và hiệu suất thực thi. Quá trình thực thi sử dụng mã khối ở chế độ OFB, CBC hoặc CFB đủ khối có thể dễ dàng được sử dụng để thực thi cấu trúc bộ tạo bit ngẫu nhiên tất định dựa trên OFB; quá trình thực thi hỗ trợ chế độ bộ đếm có thể sử dụng lại phần cứng hoặc phần mềm để thực thi bộ tạo bit ngẫu nhiên tất định ở chế độ bộ đếm.

Xét về hiệu suất, cấu trúc ở chế độ CTR phù hợp hơn so với cấu trúc xử lý liên lệnh và song song trong khi đó cấu trúc ở chế độ OFB lại yêu cầu hỗ trợ phần cứng ít hơn.

F.5 Nguồn bất định có điều kiện và hàm dẫn xuất

Bộ tạo bit ngẫu nhiên tất định dựa trên mã khối sử dụng một trong hai cách để khởi tạo trạng thái bộ tạo bit ngẫu nhiên tất định trong quá trình khởi tạo và thay mầm mới: với các xâu đầu vào dạng tuỳ ý chứa một lượng độ bất định cụ thể hoặc với các xâu có đủ độ bất định với độ dài được quy định chính xác. Các chuỗi tuỳ ý bắt buộc phải sử dụng một hàm dẫn xuất khi không sử dụng các xâu có đủ độ bất định.

Quá trình thực thi sẽ chọn có cung cấp các bit bất định có điều kiện hay không, hoặc để hỗ trợ hàm dẫn xuất. Đây là quyết định thiết kế hệ thống cấp cao ; nó ảnh hưởng đến loại nguồn bất định được sử dụng, số cổng hoặc kích thước mã nguồn của quá trình thực thi và giao diện mà ứng dụng phải có đối với bộ tạo bit ngẫu nhiên tất định. Ngoài ra, thiết kế số cổng rất thấp có thể sử dụng các nguồn bất định phần cứng điều chỉnh dễ dàng, ví dụ như một bộ dao động vòng được XOR với nhau, thay vì hỗ trợ xử lý phức tạp đối với các xâu đầu vào. Mặt khác, quá trình thực thi bộ tạo bit ngẫu nhiên tất định với mục đích chung có thể cần khả năng xử lý các xâu đầu vào tuỳ ý như đối với các xâu thông tin cá nhân và đầu vào bổ sung; trong trường hợp này thực thi hàm dẫn xuất của mã khối.

Phụ lục G
(tham khảo)
Thảo luận về ước lượng của độ bất định

Việc ước tính số lượng độ bất định được tạo ra bởi một nguồn bất định là một khía cạnh quan trọng trong thiết kế bộ tạo bit ngẫu nhiên bất định đáp ứng tiêu chuẩn này. Điểm bắt đầu cho phép tính này là xác định mô hình thống kê mô tả hoạt động của nguồn bất định. Với mô hình này, lượng độ bất định được tạo ra bởi nguồn bất định có thể được ước tính bằng cách sử dụng các công thức khác nhau.

Phụ lục này tập trung vào hai công thức cụ thể trong đó nguồn bất định tạo ra một trong n đầu ra hoặc chuỗi đầu ra có thể trong một khoảng thời gian với đầu ra có thể thứ i có xác suất là p_i.

Định nghĩa phổ biến nhất của độ bất định là định nghĩa của Shannon, $H = -\sum_{i=1}^{n} p_i \log_2 p_i$, hữu ích trong các bối cảnh lý thuyết thông tin khác nhau. Tuy nhiên, các nhà nghiên cứu mật mã đã nghiên cứu một họ đo độ bất định thay thế được gọi là độ bất định Rényi, được tham số hóa bởi giá trị α, trong đó $0 \leq \alpha < \infty$ (xem [12]). Độ bất định Rényi cấp α đối với phân bố trên được định nghĩa là $H_\alpha = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{n} p_i^\alpha$. Họ này bao gồm cả độ bất định Shannon, vì sử dụng quy tắc l'Hôpital, ta có mặc dù H_1 không xác định, giới hạn của H_α khi α tiệm cận 1 chính là H. Tương tự, cũng dễ dàng thấy giới hạn của H_α khi α tiệm cận ∞ bằng $-\log_2(\max\{p_i\})$, được gọi là độ bất định nhỏ nhất (min-entropy).

Độ bất định Shannon cho biết độ bất định trung bình nhưng không chỉ ra giới hạn dưới của độ bất định, tức là *min-entropy để yêu cầu mức độ an toàn mong muốn đạt được*. Các phép đo H_2 và min-entropy có ưu điểm cụ thể trong phân tích bộ tạo bit ngẫu nhiên. Đối với một phân bố cố định $\{p_i\}, H_\alpha$ là hàm đơn điệu giảm đối với α. Do đó, min-entropy là đại lượng đo chính xác nhất của độ bất định và phục vụ cho việc xác định một ước lượng trường hợp xấu nhất của mẫu độ bất định. Min-entropy cũng đưa ra lời giải thích quan trọng liên quan đến khả năng thành công của kẻ tấn công đang cố đoán giá trị của một xâu. Giả sử xâu được tạo ra với min-entropy là K bit. Điều này có nghĩa là xác suất để đoán ra xâu là 2^{-K}. Vì vậy, nếu kẻ tấn công thực hiện chiến lược đoán tối ưu, xác suất thành công của anh ta chỉ là 2^{-K}. Ngoài ra, nếu kẻ tấn công đó thực hiện theo chiến lược tối ưu để thực hiện 2^w dự đoán (tức là anh ta đoán nhiều nhất 2^w giá trị) xác suất thành công tối đa là 2^{w-K}. Việc tính toán độ bất định có điều kiện dễ dàng đối với H_1 hơn đối với độ bất định nhỏ nhất. Dưới giả định cho các biến ngẫu nhiên có giá trị nhị phân cố định (điển hình là các bộ tạo bit ngẫu nhiên bất định vật lý), độ bất định Shannon H_1 liên quan chặt chẽ đến khối lượng công việc dự kiến cần thiết để đoán các chuỗi bit ngẫu nhiên (đủ dài).

Mặt khác, H_2 *có mối liên hệ trực quan với tỷ lệ lặp lại của một chuỗi đầu ra có phân bố xác suất cho trước*. Ngoài ra, H_2 có thể được tính toán hiệu quả bằng cách sử dụng quy trình thao tác ma trận lặp, cho trước một mô hình Markov hoặc Markov ẩn của nguồn bất định.

Phụ lục H
(tham khảo)
Sự đảm bảo của bộ tạo bit ngẫu nhiên

Một bộ tạo bit ngẫu nhiên khi được triển khai cho mục đích mật mã phải đảm bảo rằng:

- Bộ tạo bit thực sự tạo ra các bit ngẫu nhiên không thể dự đoán trước với xác suất lớn hơn đáng kể 1/2 ;
- Bộ tạo hoạt động chính xác với các giới hạn thiết kế ; và
- Quá trình thực thi hỗ trợ các dịch vụ mật mã dự kiến theo độ nhạy cảm của thông tin.

Một số hình thức kiểm tra quá trình thực thi bộ tạo bit ngẫu nhiên có thể đạt được mức độ bảo đảm cụ thể. Kiểm tra quá trình thực thi bộ tạo bit ngẫu nhiên bao gồm cả kiểm tra xác nhận bởi một phòng thí nghiệm xác minh độc lập (nghĩa là phòng thí nghiệm sẽ tự thực hiện kiểm tra), và kiểm tra vận hành (tức là kiểm tra một bộ tạo bit ngẫu nhiên được tự thực hiện trong quá trình hoạt động).

Độ chính xác của việc thực thi một bộ tạo bit ngẫu nhiên được xác nhận bởi người thực thi (và được chứng minh thông qua tài liệu) hoặc có thể được xác nhận để phù hợp với tiêu chuẩn này bởi phòng thí nghiệm được công nhận. Việc xác nhận như vậy cung cấp một mức độ bảo đảm cao hơn rằng bộ tạo bit ngẫu nhiên được thực thi đúng. Thực hiện kiểm tra trong quá trình kiểm thử tại phòng thí nghiệm nằm ngoài phạm vi của tiêu chauarn này. Kiểm tra hoạt động được thực hiện trong một bộ tạo bit ngẫu nhiên để xác định rằng bộ tạo bit ngẫu nhiên tiếp tục hoạt động như thiết kế và thực thi. Kiểm tra hoạt động được thực hiện bởi một bộ tạo bit ngẫu nhiên bất định và bộ tạo bit ngẫu nhiên tất định được cung cấp trong mục 8 và 9 tương ứng.

Phụ lục I
(tham khảo)
Ranh giới của bộ tạo bit ngẫu nhiên

Các quá trình bộ tạo bit ngẫu nhiên được đóng gói trong một ranh giới bộ tạo bit ngẫu nhiên. Trạng thái và bất kỳ đầu vào nào khác đối với bộ tạo bit ngẫu nhiên sẽ không được xuất ra một cách trực tiếp (tức là không có bất kỳ quá trình xử lý và biến đổi nào của hàm chuyển đổi trạng thái bên trong) nhưng sẽ chỉ tồn tại trong ranh giới bộ tạo bit ngẫu nhiên.

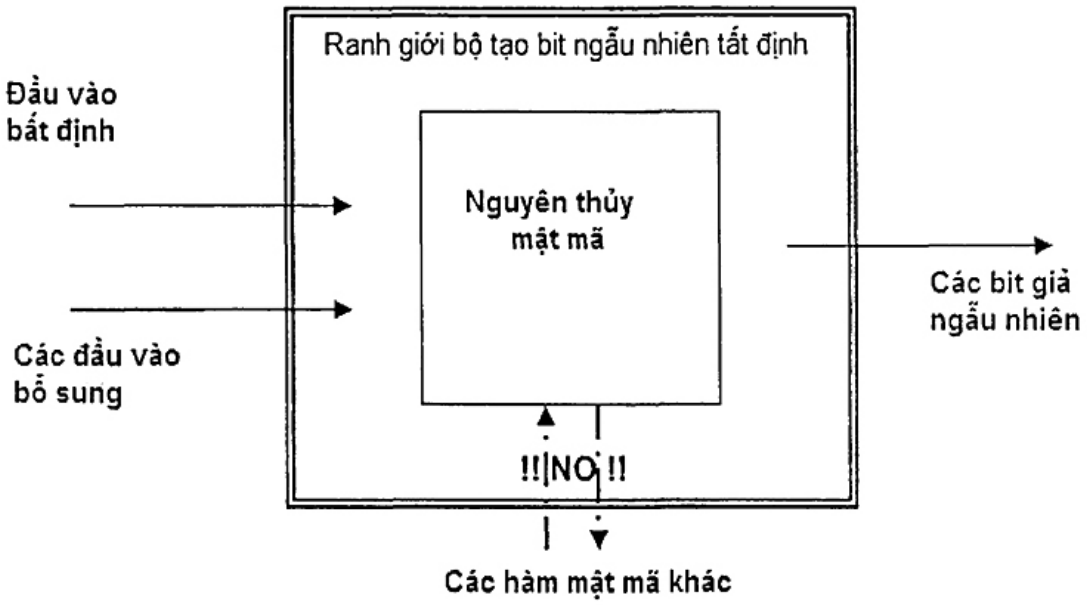
Xem xét ví dụ về từng thiết kế bộ tạo bit ngẫu nhiên tất định trong phụ lục C. Mỗi thiết kế này bao gồm một hoặc nhiều nguyên thủy mật mã (ví dụ: hàm băm). Trong trường hợp này, ranh giới bộ tạo bit ngẫu nhiên tất định được quy định để cấm hoặc cho phép truy cập vào nguyên thủy mật mã (ví dụ: hàm băm) bằng các hàm mật mã khác (ví dụ: quá trình tạo hoặc kiểm tra chữ ký số). Trong hình I.1, tất cả các hàm khác nằm ngoài ranh giới bộ tạo bit ngẫu nhiên tất định. Nguyên thủy mật mã trong ranh giới bộ tạo bit ngẫu nhiên tất định không thể được đánh giá bởi các hàm nằm ngoài ranh giới bộ tạo bit ngẫu nhiên tất định; quan sát !!NO!! trên các mũi tên nét đứt trong hình. Đối với ví dụ này, hàm chữ ký số nằm ngoài ranh giới bộ tạo bit ngẫu nhiên tất định không sử dụng hàm băm có trong ranh giới bộ tạo bit ngẫu nhiên tất định. Trong trường hợp này, một hàm băm riêng biệt sẽ được yêu cầu cho hàm chữ ký số.

Thiết kế trong đó bộ tạo bit ngẫu nhiên là chức năng duy nhất trong ranh giới bộ tạo bit ngẫu nhiên cung cấp mức độ bảo đảm cao hơn so với thiết kế trong đó các hàm khác có thể truy cập đến nguyên thủy mật mã hoặc trạng thái bên trong của bộ tạo bit ngẫu nhiên.

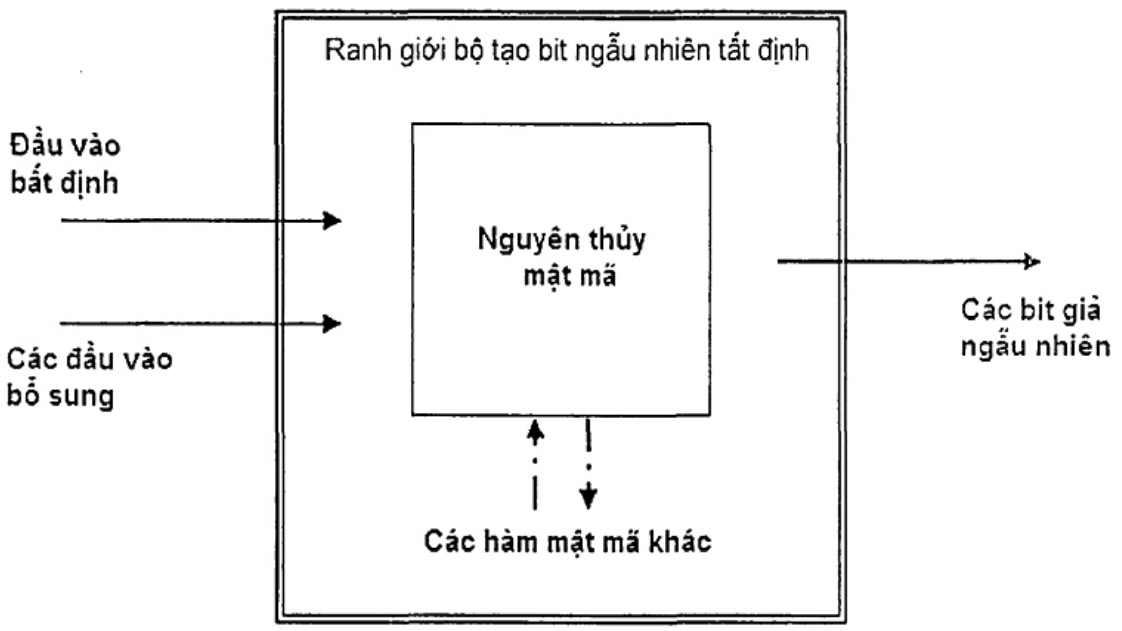
Trong hình I.2, các hàm ngoài các quá trình của bộ tạo bit ngẫu nhiên tất định nằm trong ranh giới của nó. Thiết kế này cung cấp ít sự đảm bảo hơn so với thiết kế trong hình I.1 vì các hàm không phải của bộ tạo bit ngẫu nhiên tất định khác có khả năng truy cập đến trạng thái bên trong. Sử dụng thiết kế này, nguyên thủy mật mã được sử dụng bởi các quá trình của bộ tạo bit ngẫu nhiên tất định có thể được sử dụng bởi các hàm mật mã khác trong ranh giới bộ tạo bit ngẫu nhiên tất định (ví dụ: trong quá trình tạo hoặc kiểm tra chữ ký số); quan sát các mũi tên nét đứt vào và ra từ các nguyên thủy mật mã của bộ tạo bit ngẫu nhiên tất định.

Trong cả hai hình, đầu vào bất định được cung cấp từ bên ngoài ranh giới bộ tạo bit ngẫu nhiên, để thuận tiện chúng ta mô tả như vậy. Thực tế, đầu vào bất định có thể được cung cấp từ bên trong hoặc bên ngoài ranh giới bộ tạo bit ngẫu nhiên.

Các quá trình bộ tạo bit ngẫu nhiên được sử dụng bởi các ứng dụng phải được thực thi trong ranh giới mô-đun mật mã ISO/IEC 19790. Một ranh giới bộ tạo bit ngẫu nhiên được chứa hoàn toàn trong ranh giới mô-đun mật mã hoặc sẽ trùng với ranh giới mô-đun mật mã.



Hình I.1 – Ranh giới bộ tạo bit ngẫu nhiên tất định không chứa các hàm khác



Hình I.2 – Ranh giới bộ tạo bit ngẫu nhiên tất định chứa các hàm khác

Phụ lục J
(Tham khảo)
Lý do thiết kế các bài kiểm tra thống kê

J.1 Tổng quan

Có bốn phép kiểm tra thống kê xác định trong 8.8.5 là các phép kiểm tra chất lượng đối với đầu ra ngẫu nhiên. Bốn phép kiểm tra này được thiết kế để có xác suất loại 1 là 10^{-4}. Phụ lục này trình bày một số lý do cho việc thiết kế các phép kiểm tra thống kê.

J.2 Kiểm tra loạt

Ngưỡng độ dài loạt là 6 trong định nghĩa kiểm tra loạt được áp dụng cho một chuỗi 20000 bit đầu ra ngẫu nhiên. Điều này có thể giải thích như sau. Số lần xuất hiện mong đợi của loạt độ dài i trong chuỗi n bit ngẫu nhiên được tính toán bằng biểu thức $e_i = (n - i + 3)/2^{i+2}$ (xem [8]). Bảng J.1 chỉ ra e_i đối với loạt độ dài $i = 1,2,\ldots,7$ với $n = 20000$.

Bảng J.1 – Số lần xuất hiện mong đợi của loạt độ dài i

Loạt độ dài i	$[e_i]$
1	2500
2	1250
3	625
4	312
5	156
6	78
7	39

Từ bảng J.1, số lần xuất hiện thực tế của loạt độ dài i với $i \geq 6$ có thể nhỏ hơn 100 do đó không có ba chữ số. Ngược lại, những loạt với $i \leq 5$ kết quả là một số có nhiều hơn ba chữ số. Do đó, từ quan điểm thống kê, cần tổng hợp số lần xuất hiện thực tế của loạt độ dài i thành một danh mục sao cho số lượng kết quả gồm ba chữ số, cùng mức độ chính xác với số lần xuất hiện thực tế của loạt độ dài i với $i \leq 5$.

J.3 Kiểm tra loạt dài

Bảng J.2 chỉ ra số lần xuất hiện mong đợi của loạt độ dài i trong 10^4 tập chuỗi ngẫu nhiên 20000 bit và sử dụng biểu thức e_i trong J.2.

Bảng J.2 – Số lần xuất hiện mong đợi của loạt độ dài i trong 10^4 tập chuỗi ngẫu nhiên 20000 bit

Loạt độ dài i	$10^4 e_i$
26	0,74
27	0,37

Khi xem xét các con số được liệt kê trong bảng J.2 được làm tròn lên, có khả năng là có một lần xuất hiện loạt độ dài 26 trong 10^4 tập chuỗi ngẫu nhiên 20000 bit. Ngoài ra, không xuất hiện loạt độ dài 27

trong điều kiện tương tự. Do đó, loạt độ dài 27 là ngưỡng tương ứng với lỗi loại 1 với xác suất là 10^{-4} trong kiểm tra loạt dài.

Phụ lục K

(tham khảo)

Ví dụ các trường hợp đặc biệt cho MQ_DRBG

K.1. Tổng quan

Phụ lục K và các tập tin hỗ trợ của nó cung cấp các trường hợp ví dụ cho 14 thiết lập được liệt kê trong ISO/IEC 18.031: 2011, Bảng C.5. Các tập tin hỗ trợ có sẵn tại URL sau:

http://standards.iso.org/iso/18031/

Trong mỗi thiết lập của 14 được mô tả trong Phụ lục K, chuỗi bit P cung cấp một hệ thống lựa chọn ngẫu nhiên của các phương trình bậc hai đa phương mà tuân theo các quy tắc lựa chọn của C.5.2.5. Kết quả phân chia xếp hạng từ việc kiểm tra các điều kiện xếp hạng được trình bày chi tiết cho mỗi thiết lập.

P được đưa ra theo định dạng mô tả trong C.5.2.4 dưới đây. Mỗi trường hợp ví dụ cũng bao gồm một chuỗi các cặp đầu vào và đầu ra liên tiếp cho hàm **Evaluate_MQ** (...).

K.1.1 Định dạng để biểu diễn các phần tử trường

Mỗi hệ số của hệ thống là một phần của trường nhị phân $GF(2^{field\_size})$ và là một đa thức đơn biến trên trường $GF(2)$ theo mô-đun đa thức tối giản được đưa ra trong Bảng C.6. Mỗi phần tử của trường được xử lý như là một chuỗi xâu bit của các bit *filed_size* bao gồm các hệ số $GF(2)$ của nó được sắp xếp theo mức độ giảm dần. Ví dụ, đa thức $x^3 + x + 1$ trong $GF(2^4)$ được biểu diễn dưới dạng các chuỗi xâu bit 1011.

K.1.2 Định dạng để biểu diễn một phương trình bậc hai đa phương duy nhất

Hệ phương trình bậc hai được sử dụng trong MQ_DRBG hoạt động trên các biến $n = state\_length/field\_size$ và bao hàm các phương trình $(n + m)$ với $m = block\_length/field\_size$. Phương trình bậc hai được viết dưới dạng kết nối các hệ số của nó theo thứ tự từ điển và bằng cách giảm bớt bậc. Do đó hệ số của đơn thức x_1x_1 xuất hiện đầu tiên, tiếp theo là x_1x_2 và vân vân, đến hệ số x_1x_n. Hệ số của đơn thức x_2x_2 xuất hiện tiếp theo, sau đó là x_2x_3 và vân vân, cho đến cuối cùng đạt được hệ số bậc hai là $x_{n-1}x_n$. Sau đó các hệ số tuyến tính xuất hiện, bắt đầu với hệ số của đơn thức x_1 và kết thúc bằng hệ số x_n. Khi $field\_size = 1$, các hệ số tuyến tính được bỏ qua từ đó trường nằm bên dưới là $GF(2)$ và $x_ix_i = x_i$. Chuỗi kết thúc với hệ số không đổi của phương trình bậc hai.

K.1.3 Định dạng để biểu diễn một hệ phương trình bậc hai hoàn chỉnh

Hệ phương trình bậc hai được mã hoá trong chuỗi bit P bao hàm các phương trình bậc hai $n + m$ của nó được ghép nối theo thứ tự tuần tự, bắt đầu với các hệ số của phương trình đầu tiên và kết thúc với phương trình thứ $(n + m)$.

K.1.4 Định dạng để biểu diễn cho các đầu vào và đầu ra

Dữ liệu đầu vào x đến **Evaluate_MQ**(P, x) là một vector của các phần tử trường n và cho dưới dạng là một chuỗi bit được tạo thành bằng cách ghép nối các biểu diễn chuỗi bit của chúng, bắt đầu với $x_1$và kết thúc với x_n. Tương tự, đầu ra $y||z$ là một vector của các phần tử trường $n + m$ được biểu diễn trong cùng một định dạng.

K.1.5 Tóm tắt các trường hợp ví dụ

Bảng K.1 tóm tắt 14 trường hợp ví dụ.

Bảng K.1 - Tóm tắt các trường hợp ví dụ

requested_ strength	block_length			
	112	128	192	256
80	K.2 Trường nhị phân $GF(2)$ $n = 112$ $m = 112$ **min_weight** = 4 *min_rank* ≥ 106	K.4 Trường nhị phân $GF(2^4)$ $n = 32$ $m = 32$ *min_weight* = 5 *min_rank* ≥ 30	K.7 Trường nhị phân $GF(2^6)$ $n = 32$ $m = 32$ *min_weight* = 5 *min_rank* ≥ 30	K.11 Trường nhị phân $GF(2^8)$ $n = 32$ $m = 32$ *min_weight* = 5 *min_rank* ≥ 30
112	K.3 Trường nhị phân $GF(2)$ $n = 120$ $m = 112$ *min_weight* = 4 *min_rank* ≥ 114	K.5 Trường nhị phân $GF(2)$ $n = 128$ $m = 128$ *min_weight* = 4 *min_rank* ≥ 122	K.8 Trường nhị phân $GF(2^4)$ $n = 48$ $m = 48$ *min_weight* = 5 *min_rank* ≥ 44	K.12 Trường nhị phân $GF(2^4)$ $n = 64$ $m = 64$ *min_weight* = 5 *min_rank* ≥ 60
128		K.6 Giống như K5	K.9 Trường nhị phân $GF(2^3)$ $n = 64$ $m = 64$ *min_weight* = 5 *min_rank* ≥ 60	K.13 Giống như K.12
192			K.10 Trường nhị phân $GF(2)$ $n = 200$ $m = 192$ *min_weight* = 4 *min_rank* ≥ 192	K.14 Trường nhị phân $GF(2^2)$ $n = 128$ $m = 128$ *min_weight* = 5 *min_rank* ≥ 124

256				K.15 Trường nhị phân $GF(2)$ $n = 272$ $m = 256$ *min_weight* = 4 *min_rank* ≥ 264

K.2 Ví dụ trường hợp *requested_strength* = 80 và *block_length* = 112

K.2.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit P có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-112-Sec-80-F2.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 177212 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

95d78546df132777af932886a887da96aa9afa46

Các hàng được phân phối như sau:

106: 4561

108: 2213145

110: 58156950

112: 43613144

Tổng: 103987800

K.2.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 00000000000000000000000000001

y = bb8cf180cbc3a6002c19c770ed0d

z = 7847b864cfadf70fb359203e06d8

x = bb8cf180cbc3a6002c19c770ed0d

y = a1e0811b5b7733113ca8e22dd2b1

z = 57d27f7b0fc67aec0d5e8115cd93

x = a1e0811b5b7733113ca8e22dd2b1

y = 634ae5294dbc4cc79ce11cfeb1d7

z = c42c5cc5b5b61396df3fcf7a4e2b

x = 634ae5294dbc4cc79ce11cfeb1d7

y = 36701faea23130a0407a44f5e420

z = bf3ddd3cbb141fcd96cbba66ebb9

x = 36701faea23130a0407a44f5e420

y = 74b5baa1095f61eb6b15d317d5ed

z = 7f4ad5787a0c5451bddcf2aef533

x = 74b5baa1095f61eb6b15d317d5ed

y = 62804addbe9da290c38e9deOfe71

z = 5f1f209b62cce21f75d9d03607a9

x = 62804addbe9da290c38e9deOfe71

y = 7d0892da52eed7facc377af1918f

z = 69d5bef53c03fa33a0273cf44c21

x = 7d0892da52eed7facc377af1918f

y = 8ee43a16842345d4cd182852cdea

z = ed479a677e6c2a3cffbbada0e765

x = 8ee43a16842345d4cd182852cdea

y = 2eb8cc9185445b2bab3f4b504aaf

z = 9407f0fe9393fa335051ac2bf414

x = 2eb8cc9185445b2bab3f4b504aaf

y = 8deblOcb7Obc3818209a576fb5cb

z = 6106cb8aa8e9a7de949a506b2278

K.3 Ví dụ trường hợp *requested_strength* = 112 và *block_length* = 112

K.3.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-112-Sec-112-F2.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 210569 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

ae1c4ea33afc96e3aa421f6456055a7c7ee33989

Các hàng được phân phối như sau:

114: 5239

116: 2551294

118: 66936700

120: 50200265

Tổng: 119693498

K.3.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 00000000000000000000000000000001

y = 46609cda28057a917a08b60a1d969d

z = a06fe3e456a8c24315dfde6088bd

x = 46609cda28057a917a08b60a1d969d

y = 37d12de7b69f2170ba8717e96f0f43

z = 8fb9899c9e2d4ef33056aadf946d

x = 37d12de7b69f2170ba8717e96f0f43

y = 463860297cec60797650c4897563d4

z = 89745528548d7bd3a2c9e5afd3fc

x = 463860297cec60797650c4897563d4

y = 6a4c5b16c156738e9b07c4c2c2818e

z = 5f9f14194e601f48657164f34e34

x = 6a4c5b16c156738e9b07c4c2c2818e

y = 289c50a28bb48a685703eb425597dd

z = c9dae7a3c32a01648a32d91b8728

x = 289c50a28bb48a685703eb425597dd

y = 4d96224af4aeaac54d8472374f645d

z = cf7a6cc73793049241497ee26603

x = 4d96224af4aeaac54d8472374f645d

y = df5ac81223125d967056d5dcdba088

z = 3d9741ec702076fe8473b7181aa9

x = df5ac81223125d967056d5dcdba088

y = 41a1df8cc57c402f520d671464b728

z = 285d6b741e417e417b9f8fa87356

x = 41a1df8cc57c402f520d671464b728

y = 0af3539a48bc07e3afb00d3c529ff5

z = e6d4d36dcc2cca4826b94e76be10

x = 0af3539a48bc07e3afb00d3c529ff5

y = e2f7d8f01d2ae145a643b9351ada76

z = 29bdd54840cf84027f20e48ce195

K.4 Ví dụ trường hợp *requested_strength* = 80 và *block_length* = 128

K.4.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit P có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-128-Sec-80-F16.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 17952 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

d6614e19bd953ca88ff49f016b80f5ac17b7dab1.

Các hàng được phân phối như sau:

30: 520948

32: 7782684

Tổng: 8303632

K.4.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 00000000000000000000000000000001

y = f719e81ed992ca7c793258b5251d0534

z = 66092272f74a85ecaef639d78ed9831f

x = f719e81ed992ca7c793258b5251d0534

y = 37614b89b9bbd6eea4560ecb3bdb8807

z = 96b4c1aeb27aa47fbc7a3b1464343736

x = 37614b89b9bbd6eea4560ecb3bdb8807

y = 136bf7d8fbcbabd37a2baa321a5d94f7

z = 29141359d8099496eaf84ae3d863591a

x = 136bf7d8fbcbabd37a2baa321a5d94f7

y = bc6316205ac244b4fc8dcee70f423874

z = d8005ccefa012118820cf02c9eb4328d

x = bc6316205ac244b4fc8dcee70f423874

y = 64d8adbf03a6418fa549f235e5f84bcd

z = 9c0aad312ef00336d0f055e81f2b3677

x = 64d8adbf03a6418fa549f235e5f84bcd

y = 3ac1c733b68ca734550343d950649d5a

z = 1f07210c4a6d4fd784ee0f9f9789c5ab

x = 3ac1c733b68ca734550343d950649d5a

y = 1a22cbbe771e641373700306718dbf6e

z = ba8064102a7e8d714e92e0dfddfbe607

x = 1a22cbbe771e641373700306718dbf6e

y = fa2eabf2c9794f6b9bac6561409aabOd

z = 7e2bae34daaf284557bbe5ae48e54d26

x = fa2eabf2c9794f6b9bac6561409aabOd

y = 46f6f74d23504a64565b2c35cd0036df

z = c6285e77cbf16150457d03bfc6015ef7

x = 46f6f74d23504a64565b2c35cd0036df

y = 729bc30c32fd7fec1ccb95bc4aabfa27

z = 963bda8ab7dc84ee2dd5a60a9c4392cd

K.5 Ví dụ trường hợp *requested_strength* = 112 và *block_length* = 128

K.5.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit P có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-128-Sec-112-F2.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 264224 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

fcd983e78ddd489a9425be58b8139e04c89fb6c6.

Các hàng được phân phối như sau:

122: 7704

124: 3783524

126: 99303857

128: 74493971

Tổng: 177589056

K.5.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 00000000000000000000000000000001

y = c04f664eb59219b1e6b0d0e0fc5ae660

z = 894f5e21cc208ce73ebb136c0c7b6e47

x = c04f664eb59219b1e6b0d0e0fc5ae660

y = 10da311bd87ba42fd89a17f45b0b0931

z = f3561e3a42a23037d04b7991e44f98d0

x = 10da311bd87ba42fd89a17f45b0b0931

y = c42c14916632d8518f435796c069a381

z = 461593a9673573772cf8f8a93020eada

x = c42c14916632d8518f435796c069a381

y = 325ebb605c4037b6092a7952adedd16d

z = 3f80f69b2f81f012994189125cba6b00

x = 325ebb605c4037b6092a7952adedd16d

y = 596cdd4392413988fa7a15fa7fb5d74b

z = 8ec0f223da49f826f6faf8d25b54b231

x = 596cdd4392413988fa7a15fa7fb5d74b

y = 17790ac47b8112312631c0e3b0066fd0

z = 8d98258ade35f74057a98542c0d7d937

x = 17790ac47b8112312631c0e3b0066fd0

y = d76a00f9e3318091e0f113b48f0cb752

z = 8b9664e30122848541b91743171b4812

x = d76a00f9e3318091e0f113b48f0cb752

y = 4ffffdf6def93bb391d90312a801ece5

z = d17acb75d2f57976df164061716601e0

x = 4ffffdf6def93bb391d90312a801ece5

y = 89b596d08123105f5994679f5e428136

z = 1de72c77e98fa45090197c81e4d2a3a3

x = 89b596d08123105f5994679f5e428136

y = 7294da76d6d2bf9dfb9d2c1d03ca4928

z = 96cea7ffad3bc8be151106cd4c067565

K.6 Ví dụ trường hợp *requested_strength* = 128 và *block_length* = 128

Trường hợp này tương tự như trong K.5.

K.7 Ví dụ trường hợp *requested_strength* = 80 và *block_length* = 192

K.7.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-192-Sec-80-F64.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 26928 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

dad206d21189e0b9aaa5bb60298e0cf3f918ae81.

Các hàng được phân phối như sau:

30: 129895

32: 8173737

Tổng: 8303632

K.7.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 0000000000000000000000000000000000000000000000000000000000000001

y = 9b021d45d52d7637ca3350732d880265bc826a1a7e9969b4

z = f5df581e7eb83f86d7f1101b248b4740662733897f4b4689

x = 9b021d45d52d7637ca3350732d880265bc826a1a7e9969b4

y = 937625307ba580d0763a6272e3ba7d9256536795a0017b7c

z = 338d582a1b8c37812aed0f07f2958101f11ac11ee962a02a

x = 937625307ba580d0763a6272e3ba7d9256536795a0017b7c

y = d9d32553f1299fcd1444030c23f084d1c85fd08a2ef5275b

z = 9bb8f5de5ff6c6aba79ae510429221a6dc3b0ab7ef1b27ca

x = d9d32553f1299fcd1444030c23f084d1c85fd08a2ef5275b

y = 329800d2a8f045893d24c54d6993e67e0e2c34b2956ee269

z = 0a01d2046afc3adbd68e024f1205ca00cee736e53c7aaabe

x = 329800d2a8f045893d24c54d6993e67e0e2c34b2956ee269

y = 723a5476da2e4617f9418545014266ac7179f591451952e9

z = 5cf9cfeefb0d0ebf2985a4cc8e66c7ba0fcf332971a4ebad

x = 723a5476da2e4617f9418545014266ac7179f591451952e9

y = 67786dd966efde83df99af5f6059f4c3f07f84fe5ae1d656

z = f68fa0ae09671a71f780550280204bb9430318733649ea3d

x = 67786dd966efde83df99af5f6059f4c3f07f84fe5ae1d656

y = 03ecc9b12c99215531b56f5a99e5407d5c343ac6b7793c54

z = 34a9a04510415ad411fc14f8046f6538e4df891b1744462d

K.8 Ví dụ trường hợp *requested_strength* = 112 và *block_length* = 192

K.8.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-192-Sec-112-F16.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 58800 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

151f9a4c9b32f6e9236ae071408bbba39b597585.

Các hàng được phân phối như sau:

44: 2

46: 4054618

48: 60538940

Tổng: 64593560

K.8.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 00000000000000000000000000000000000000000000000001

y = 4bc49157a1cfc4e81a00e38073c3c330716846f262fd6bd8

z = 9f9c37ccbf0351e173402c9778721e07b7f92e7ad9c13a0c

x = 4bc49157a1cfc4e81a00e38073c3c330716846f262fd6bd8

y = efd8d72753ca6319ad85e8632a2731082e24851cb496517b

z = 3b6ed30bfac7560230730efa2befd0b63c610ddc84579b47

x = efd8d72753ca6319ad85e8632a2731082e24851cb496517b

y = 6817118a48c2faae74cd95654416f072c44dfd2309bb216d

z = e2dd5934948ac3f16f4dcd5e1343104e8a69fc004e700ae1

x = 6817118a48c2faae74cd95654416f072c44dfd2309bb216d

y = 668a844f1fd85b03eb5ace651179ba6adb6080dda180dfc0

z = 78f8d2f16f0c1cdb96b7eeef95e8d852267d2599bfd0a3a5

x = 668a844f1fd85b03eb5ace651179ba6adb6080dda180dfc0

y = 84f988e9a1335ceaa21e526bd1df380d137a7470ad2167fb

z = d459b9f5ac1ab0622e324994bd1c70038e1e500d6ed21c87

x = 84f988e9a1335ceaa21e526bd1df380d137a7470ad2167fb

y = 5f3263e90c40060d975443ca13dc9388f01bec3c3f3d28ea

z = b427e405b17d64c91a9e8622a896d20c7c75a8ad5139ccbf

x = 5f3263e90c40060d975443ca13dc9388f01bec3c3f3d28ea

y = 7f4a9db932fff536a7f5df253d8e974ae6852866d2558629

z = b23c1c8f7fff065e9b9827b65b6f3d156d4c8bc1343b173c

x = 7f4a9db932fff536a7f5df253d8e974ae6852866d2558629

y = 5e1395441cb914d8ad706f7c9c03ccc73e5780c2e44f5504

z = 284e181d513eb5e9bebde7d78f6c8ed52bec0ffda2a96634

x = 5e1395441cb914d8ad706f7c9c03ccc73e5780c2e44f5504

y = 9a28d4de9d2a5dae64c38a936c672072e90f60f71fa1cbf9

z = e41bb6b73ba2052d035d07f8dc5cb40d15f1eca51fbd6e0e

x = 9a28d4de9d2a5dae64c38a936c672072e90f60f71fa1cbf9

y = 66660d18cf3651da98e89f566bf00355cca6c0baf9a2d1c1

z = 601c8b113003889e0348566729e8e0751f53c294f5dbe000

K.9 Ví dụ trường hợp *requested_strength* = 128 và *block_length* = 192

K.9.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit P có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-192-Sec-128-F8.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 102960 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

b286f9cad81b0b718ead54180239eba7d3b4d14a.

Các hàng được phân phối như sau:

60: 1089

62: 34926475

64: 240656468

Tổng: 275584032

K.9.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 000000000000000000000000000000000000000000000001

y = 762258ce8826f6d2dd46a7c6784733f050929aa23ab60ee8

z = 1155bc183f9c8fa8ed268a4bae7256b7f2c456f3729b5d0b

x = 762258ce8826f6d2dd46a7c6784733f050929aa23ab60ee8

y = 02e827998b27a602323923e9db9c48750316d0e055556622

z = 3127eeb8f998fd23ed59e9993263ee5cd6cec2a221bb6ce5

x = 02e827998b27a602323923e9db9c48750316d0e055556622

y = bd469c2e97afbc180f8b023205e46c060a6bc453ef5898d3

z = c1586f6694edfbc5cf7b269136fdddcb392129e938453cf2

x = bd469c2e97afbc180f8b023205e46c060a6bc453ef5898d3

y = 293f502e6b12185832844d5850f7a08ce4229e567adeea11

z = 7541eb4affc5517ead8430e2701c94d50b461e14e063543d

x = 293f502e6b12185832844d5850f7a08ce4229e567adeea11

y = 15f0f194a952f1c98eb5d4449a3b21d8dfa0b068982b19df

z = 71f515a88870778b7194870180fb7bd963960c44d43fc4f0

x = 15f0f194a952f1c98eb5d4449a3b21d8dfa0b068982b19df

y = bce241c8b0ea940679bad3c259860ff978d7efaad313aed7

z = 0a37faed572b12a1a0fb5167a69bab22d1093a051cbeb725

x = bce241c8b0ea940679bad3c25986Off978d7efaad313aed7

y = fe7bf955dbd9b517d516ce81584985f54bc93229bd0baf97

z = b29254189ef82d3cb673c2c17ded8378a3effbc323e09095

x = fe7bf955dbd9b517d516ce81584985f54bc93229bd0baf97

y = 3b0ab7a9534dc16df8dbc8f921049bf24163009c7a468684

z = c7d85ea5c1adc569c435db47135c20c63ae525b18fe6d5b0

x = 3b0ab7a9534dc16df8dbc8f921049bf24163009c7a468684

y = 421b005096b17b479d04215fdd7262e86460fe7c0e439949

z = bf3f9480e4f1fb02ad22f94e24063349d68600394a9c2ed4

x = 421b005096b17b479d04215fdd7262e86460fe7c0e439949

y = 9cfd31b93ea12da49b9c176710f7830b43803b20fab8bd51

z = 55a2eee9251283d70798708e1d65e7c23a2f4f06d5841d75

K.10 Ví dụ trường hợp *requested_strength* = 192 và *block_length* = 192

K.10.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit P có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-192-Sec-192-F2.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 984949 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

3d71baf6bdf23781111540b36f9f393746902a08.

Các hàng được phân phối như sau:

192: 4

194: 42773

196: 20853635

198: 547456333

200: 410557593

Tổng: 978910338

K.10.2 Đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 000000000000000000000000000000000000000000000001

y = ca6ed87127464695c014665de58f55d8c81bbba26d7dbb34f3

z = bee4dcf9847b33da091bb3a50f6b1a1ff8689bc249426711

x = ca6ed87127464695c014665de58f55d8c81bbba26d7dbb34f3

y = 7cc8ea31334cbda06d7bcf77b54a2b4ef5d27771720c2028e7

z = c80918a5a5d501bafc6df7ca2da3891911a899202c5f6da0

x = 7cc8ea31334cbda06d7bcf77b54a2b4ef5d27771720c2028e7

y = 80fb06345c0079dcc5c5c4f875f7174585e5ff0d63b65b1424

z = 7fd6439d40e5bf3f4590bdc7d4887c0d2988f46e1f0cb58e

x = 80fb06345c0079dcc5c5c4f875f7174585e5ff0d63b65b1424

y = ac2e60f576985e0bc4555e5c1a3b78b3bfe98f6f97aac6967e

z = 1a536213fbcd3d18d07c5bc5177daa29f7389508c2fa0d22

x = ac2e60f576985e0bc4555e5c1a3b78b3bfe98f6f97aac6967e

y = 3eb7d7381b1f10098867525a5ab25513a25b74ef5756212963

z = 1001ccf1d6f3ce825973efb492326804f676790d0d3650f8

x = 3eb7d7381b1f10098867525a5ab25513a25b74ef5756212963

y = b9f5fc0d53a53333e0c3f3eecfcd898ebc4d3228fd1f03420a

z = fff44f677a5585fe668f0275a62ab2cc2b221628ccbe4532

x = b9f5fc0d53a53333e0c3f3eecfcd898ebc4d3228fd1f03420a

y = 9b51e0e702e4b72cfab3f8c19b0a39cc577eb3d24603868913

z = b2a6ea5edce91844f3b479d12521e1f362dcc89addc393c8

x = 9b51e0e702e4b72cfab3f8c19b0a39cc577eb3d24603868913

y = 8117c767d4ea992546e091d42ff05ade6f6da391727af34042

z = 3cb7d9104dd79deecfa0cee874385e726bc5c43154c620b2

x = 8117c767d4ea992546e091d42ff05ade6f6da391727af34042

y = 7e8740424081b4f02360920a1f9123c68ae49246e0aed231e1

z = c138871d42aec2ba6e7bfa619d7999e105e7379d5779de79

x = 7e8740424081b4f02360920a1f9123c68ae49246e0aed231e1

y = 43152f18f70c2993caf5dc64ff26be9e8f823e9cf13a5b44f8

z = 20fcbff61224ccc6deccef04f8409842e6670c63ff50fb25

K.11 Ví dụ trường hợp *requested_strength* = 80 và *block_length* = 256

K.11.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-256-Sec-80-F256.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 35904 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

25ac05d47b2d1dc9ca211330b5db4228c06c0552.

Các hàng được phân phối như sau:

30: 32305

32: 8271327

Tổng: 8303632

K.11.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 0000000000000000000000000000000000000000000000000000000000000001

y = 845ff78e453e980385e2265728bbbc90c792bafa77bd1d942ac2fcfb4c508963

z = bc5c9f41eb6ce6d3f9936c5cb7cac610f55902149afcdace189b610d254c2bff

x = 845ff78e453e980385e2265728bbbc90c792bafa77bd1d942ac2fcfb4c508963

y = 99dc7f54302df0c04b075e7db6c3c10fbe70a18edf9852b7896f31aa18788192

z = 92f38812133684f0c24cd8a1852df8edd0eaa91828011f3e1b5e2bf5def130d9

x = 99dc7f54302df0c04b075e7db6c3c10fbe70a18edf9852b7896f31aa18788192

y = bf6202d53d315b60b8263a77faf610926d5f479e33bcd899713ae123bb0f5769

z = 7f1d0a9e2895c40c03fe7153e4afc4738ea041593e11adc0eed00749cfc22b04

x = bf6202d53d315b60b8263a77faf610926d5f479e33bcd899713ae123bb0f5769

y = 0ce571d57cc82cb8ab65ad61a2d98af5fd32b0a6315e1cb68c43a6da4b5569f0

z = 9689a9dba6ae9e51fd4d31fe1a2d0b6396aadae62f1125d9f050f38ef445c704

x = 0ce571d57cc82cb8ab65ad61a2d98af5fd32b0a6315e1cb68c43a6da4b5569f0

y = 3b1210b596739b1dd0b376ff8db7cc8623892f285f6d3e3608c34a5718318ea1

z = e6f5e3fc89ae09a9658d8518d42f163102f87dc57e779d46b442e11516345ec2

x = 3b1210b596739bldd0b376ff8db7cc8623892f285f6d3e3608c34a5718318ea1

y = 2fd0bc4dfdb228a972df094d412cb8369ea68a8ce3c81d7a431bc1264842895e

z = 2e424cdf55c87e859648db33578b025c2ede28e4c822382cd2784e8320f49436

x = 2fd0bc4dfdb228a972df094d412cb8369ea68a8ce3c81d7a431bc1264842895e

y = f73335a5e4fdc5ebe31adf389c1346a7968361b745ad5b69f781bae0eefc54b0

z = 74d2d21284d3deaca76009f5633c33cbd6808cd3bf239b5aab8969034db1edlf

x = f73335a5e4fdc5ebe31adf389c1346a7968361b745ad5b69f781bae0eefc54b0

y = d64a1e12e8574169dfba54b856415289fafbfae8e7f605af72a15c89b4cf203f

z = 38639991897b5c3f7a0d8adba26abf061e6c4c5532e0514c5029c6c6a20a399d

x = d64a1e12e8574169dfba54b856415289fafbfae8e7f605af72a15c89b4cf203f

y = 011fd95e7497a73cc21e2fd30a6f14a2b3e37e9716e238893e3a81e8a0ff7334

z = 7541b2e6d273efc5d2481b2d9154e92b6c90e73b0fca4e090576a91574c4e9fe

x = 011fd95e7497a73cc21e2fd30a6f14a2b3e37e9716e238893e3a81e8a0ff7334

y = 141bb711f4996dff29382dbb1083e5c30d60d022f65603eb42907a467c934c42

z = 55b876a9a19dad4caeb1c5c4c04efb6e93da1f065c3dee4771d2de484c3180bc

K.12 Ví dụ trường hợp *requested_strength* = 112 và *block_length* = 256

K.12.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-256-Sec-112-F16.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 137280 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

e975ba3b3ab3b87311alab5f62aed2587c20e7de.

Các hàng được phân phối như sau:

60: 23

62: 17286027

64: 258297982

Tổng: 275584032

K.12.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 0000000000000000000000000000000000000000000000000000000000000001

y = 0c44dd5a830cb42a0d78978a66e73343712523adf59e2161b7290d45e211208f

z = bb29008dddfd182c7ad58ed9c986046fde8694cbacb45bc2acf201fc1f37916f

x = 0c44dd5a830cb42a0d78978a66e73343712523adf59e2161b7290d45e211208f

y = f12417b73a69d210afef9ded8488b2790c437eafa19c6e7276bb341b0fe8bcf4

z = d895edf6a3f388968f8c2b283b2724b7cc428ee78bfce9ef88b041c1e83c1aa0

x = f12417b73a69d210afef9ded8488b2790c437eafa19c6e7276bb341b0fe8bcf4

y = 57c56dcb6db4dcaef4c09859e4a64939781ce90ea94b168987d4b47cb24ddb5c

z = 0f734bb269d9678ae4ec77162e0b175da2d4daf7a8f2211920ca1d1804cfe91f

x = 57c56dcb6db4dcaef4c09859e4a64939781ce90ea94b168987d4b47cb24ddb5c

y = d570b0477acf679409ec942280313d6f479fb1060016d11b942d2f46511024be

z = 65eeadbc13e38fabb5045c62ea917404996bab6fdf6f4abf930c2f4a8d221d42

x = d570b0477acf679409ec942280313d6f479fb1060016d11b942d2f46511024be

y = 05198de61c6cf4f1f7f3157c3210579ff4248827538836f96d037a60876386cb

z = 2711aaaa0d37e8d7ecc80ebf0d68abbc4a4b780b0c3b7cb7fba0098328aed060

x = 05198de61c6cf4f1f7f3157c3210579ff4248827538836f96d037a60876386cb

y = 069d436cea13ccc1e50a87d724f9aa96d8d99108a5ae86a9a02a15f6721114cd

z = d1cb31c2d4486d8aa8449d02dc50fae73cf0317311cf2dce9eaac923d978e769

x = 069d436cea13ccc1e50a87d724f9aa96d8d99108a5ae86a9a02a15f6721114cd

y = cbc5434867a88371200f951e4f44dc8c8fb3c4236a13d7aec62787e7f239b042

z = 4df52214ee887e49ec16881425645ae7125e556473b3eaad0902a59e08b81b8e

x = cbc5434867a88371200f951e4f44dc8c8fb3c4236a13d7aec62787e7f239b042

y = 2366a4e171758576d43053482ce20c33cfc3392e57cc498a56b0524cff5e1732

z = 212d896ba57ab438a3fc43d2889a34b2eb953c9cfd98309a8b4f1013d6e34d73

x = 2366a4e171758576d43053482ce20c33cfc3392e57cc498a56b0524cff5e1732

y = 45ba6db0eb61b4983c48e31af8b5e8635fef8ed55519f8020c5630b7fc38640a

z = aa821c12f821809adda4deb3f86590aaa9ecf414bfa3360fd819479b995992b8

x = 45ba6db0eb61b4983c48e31af8b5e8635fef8ed55519f8020c5630b7fc38640a

y = c877e8a6048d573b14f74936c0bafa49428f976c5340c1d93461f6aed92b7dd8

z = d4afc4905c3c846eaee1861f62c4e08499ec9334a0bc7fab9369f5b855e22af1

K.13 Ví dụ trường hợp *requested_strength* = 128 và *block_length* = 256

Trường hợp này tương tự như trong K.12.

K.14 Ví dụ trường hợp *requested_strength* = 192 và *block_length* = 256

K.14.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-256-Sec-192-F4.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 536640 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

d883d1291963672afc1b3015a912e069631e1802.

Các hàng được phân phối như sau:

124: 46726

126: 46570407

128: 130971923

Tổng: 177589056

K.14.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 0000000000000000000000000000000000000000000000000000000000000001

y = 53a4adacc42ce608b30010baa3b411015608ed7b86d42bbf227f9ed4b914c713

z = e090caa1474e6ed7a10b8a83046dfd2febd4ef016e7fb7352ff1b238a16adddd

x = 53a4adacc42ce608b30010baa3b411015608ed7b86d42bbf227f9ed4b914c713

y = 90ad7d2fe3a3d15ee80c4e08e44cbdd577a52861dc2521d89329c4c74123d4c8

z = a282a129823bd2b1c92a147214bd4d33b154a035847191841e3f82721a880147

x = 90ad7d2fe3a3d15ee80c4e08e44cbdd577a52861dc2521d89329c4c74123d4c8

y = 741dcb55103e36a0e3b474ba58061a5e78afc5b272bf52d95173bc58c3a4585c

z = 8e9f174c81a71f198791e913361aa4a9de1b8a1da2b6febf5f128bcf62680529

x = 741dcb55103e36a0e3b474ba58061a5e78afc5b272bf52d95173bc58c3a4585c

y = 0006a1fddcc1b68aa22aed2c9754aecaa001df7736dbafa54706f831df8f80c5

z = 838dd9f0ffb3423e4f1a2e3fa7b1e0be8dab954cdbe9070c6f316c5268092540

x = 0006alfddcc1b68aa22aed2c9754aecaa001df7736dbafa54706f831df8f80c5

y = 9f80f5ee116e5a5adcbbcf443b52aab3e9638b44805cbad947742fe56d162af3

z = 09337d41ff6d44f5000ed138bd027449ba6213e1e4e3638c408f9e29b55e6f33

x = 9f80f5ee116e5a5adcbbcf443b52aab3e9638b44805cbad947742fe56d162af3

y = 7a592bd01147431ff91edb535ab72763c8ec45e674783d92810627632d0c84e7

z = e969c2832eef47894a583772e1fae9a72638a535d17bb1cee4ec385bdc71dc41

x = 7a592bd01147431ff91edb535ab72763c8ec45e674783d92810627632d0c84e7

y = a1c97ffa967f5d17c4b82933b1a1575b30f1e73ab37aeleaeb299eb499fdc733

z = ed85e8650930cb8c294f35df9f440ee811160e3296aceedc1e516bcef25c0713

x = a1c97ffa967f5d17c4b82933b1a1575b30f1e73ab37aeleaeb299eb499fdc733

y = 6e338cb2faeae78bcf733e7bf0310ce3dd276db8a216ebc14b2670fbfda893a4

z = fbc50c9c4c8c81ae7035cd8452f946b40b5c7ba09a84b2ad6139836ea5795a16

x = 6e338cb2faeae78bcf733e7bf0310ce3dd276db8a216ebc14b2670fbfda893a4

y = 17dcfbf43c229c2aedf1c4c989ed5b5518e70a535671a4b5a3d29d84cc10359d

z = 996d18d025e2755888ad72fbcf77f30964df5d90681155e356eb8eaaa3f7fa2

x = 17dcfbf43c229c2aedf1c4c989ed5b5518e70a535671a4b5a3d29d84cc10359d

y = 37a9d647c9b6a76bd38f09ca33a428fc37e1d364a7bb7d1e180ce59859e17b2a

z = 0df041b0576d66818e2976ab65fcfe3a4db403f55a2afd312cd02bf565d042ef

K.15 Ví dụ trường hợp *requested_strength* = 256 và *block_length* = 256

K.15.1 Hệ các phương trình bậc hai đa biến

Chuỗi bit *P* có chứa các hệ số hệ thống được cung cấp ở dạng số trong tập tin "hệ số-BL-256-Sec-256-F2.bin" theo định dạng được mô tả trong K.1.3.

Tập tin chứa 2450514 byte và tổng kiểm tra SHA-1 của nó dưới dạng thập lục phân là:

388278a18bc63cf4ef55d90e3e0b11c3bb7b3414.

Các hàng được phân phối như sau:

264: 18

266: 140810

268: 68733537

270: 1804179033

272: 1353163134

Tổng: 3226216532

K.15.2 Các đầu vào và đầu ra

Các chuỗi bit x, y và z được cung cấp ở dạng số theo định dạng được mô tả trong K.1.4. Các giá trị thập lục phân của chúng là:

x = 0000000000000000000000000000000000000000000000000000000000000001

y = aef21a672f2676bab27761cc24b4c7cd2ce555cbf065f9c371dc81d17effaa0b2b76

z = baf02399ad3f16898659d4d03f7e6c09e08735d945b9840faf190a0477bbe8d8

x = aef21a672f2676bab27761cc24b4c7cd2ce555cbf065f9c371dc81d17effaa0b2b76

y = 0a04d5cb6f5c943b33fb99370493a25dd8b57dfe132c27a4ab4320c866ccdc9aefc3

z = fd7c53d28d8f505108b71aaaeeac3801cd6f4df7f019ff9c6f78d8854da52624

x = 0a04d5cb6f5c943b33fb99370493a25dd8b57dfe132c27a4ab4320c866ccdc9aefc3

y = 45e8272fab0bb4c9186677be21de32a948f7f77eb5f84ecc364f13bebba8adbe6270

z = 943412a41183a3c3d0fd00ed56562a5804316ab8b76d2962d0b948e3998bf9be

x = 45e8272fab0bb4c9186677be21de32a948f7f77eb5f84ecc364f13bebba8adbe6270

y = 8516f4a7043dfa4792e34bdd2bda8a93241b632b055b793fc1cbf090f2578ad99b03

z = 63f116a52e5add65c0c9888a51b792e619851caf330e0d874e2b8f590859cbae

x = 8516f4a7043dfa4792e34bdd2bda8a93241b632b055b793fc1cbf090f2578ad99b03

y = 0c31a2d1d1627f7e2d99baa3b8055f27afe27ebcf9fa0078332bddb50c7e5c054b7b

z = ada0deffc2542c6b9a794fd80aceb6e70b2c0be79ae3223937326fc1f296f5bb

x = 0c31a2d1d1627f7e2d99baa3b8055f27afe27ebcf9fa0078332bddb50c7e5c054b7b

y = bd408c45a2e2bc9ff7574d190c4f745e1b7966f71403be2c28b398f95f762d68754a

z = 66bb3d2c8a4b7976db9eb572909fd8ce28f28406f66b060318734671d55ae8ba

x = bd408c45a2e2bc9ff7574d190c4f745e1b7966f71403be2c28b398f95f762d68754a

y = 5818bf316818c0ef72985fbe7ba2210622973b5f46d15cac02237c9f1a71de845ca4

z = 64626eabbc7fca5895259e71219276eace0c6e720718efdb96cf579ba60cc00d

x = 5818bf316818c0ef72985fbe7ba2210622973b5f46d15cac02237c9f1a71de845ca4

y = be7a197ec3e63af300431b3d8bc99d3c42793ec1077926d547cc87a667a31f41fa3f

z = 307897d339ec79961a5578b144672bf2b98db86cafec9f94730ef4fde5efba98

x = be7a197ec3e63af300431b3d8bc99d3c42793ec1077926d547cc87a667a31f41fa3f

y = c0db254f172a60c67b318b77097bdbc44045534de83c95492c7be40d1354f66deb9b

z = e56d0d242beb150644c9282d05597a0fc4b6341a8ea56cfd534bcdddeb013783

Thư mục tài liệu tham khảo

[1] ANSI X9.82-1-2006 *Random Number Generation Part 1: Overview and Basic Principles*

[2] ANSI X9.82-2 (draft) *Random Number Generation Part 2: Entropy Sources*

[3] ANSI X9.82-3-2007 *Random Number Generation Part 3: Deterministic Random Bit Generators*

[4] AIS 20, version 1. *Functionality classes and evaluation methodology for deterministic random number generators.* Bundesamt für Sicherheit in der Informationstechnik (BSI). 1999. (also available at https://www.bsi.bund.de/cae/servlet/contentblob/478152/publicationFile/30265/ais20e_pdf.pdf).

[5] AIS 31, version 1. *Functionality classes and evaluation methodology for physical random number generators.* Bundesamt für Sicherheit in der Informationstechnik (BSI). 2001. (also available at https://www.bsi.bund.de/cae/servlet/contentblob/478130/publicationFile/30260/ais31e_pdf.pdf; the mathematical-technical reference is available at: https://www.bsi.bund.de/cae/servlet/contentblob/478134/publicationFile/30240/trngk31e_pdf.pdf).

[6] ISO/IEC 11770-1, *Information technology — Security techniques — Key management — Part 1: Framework*

[7] KILLMANN, W., SCHINDLER, W.: *A Design for a Physical RNG with Robust Entropy Estimators.* CHES 2008, Lecture Notes in Computer Science Vol. 5154, Berlin 2008, 146-163.

[8] MENEZES, A.J., van OORSCHOTP.C., VANSTONE, S.A.: *Handbook of Applied Cryptography*, CRC Press, New York, 1997 (pages 169-190).

[9] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-2, May 25, 2001 (with latest change notices).

[10] NIST, Special Publication (SP) 800-90, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, March 2007.

[11] PERES, Y.: *Iterating von Neumann's Procedure for Extracting Random Bits.* The Annals of Statistics, 1992, Vol. 20, No. 1, 590-597.

[12] RÉNYI, A,: *On measures of entropy and information*, Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability, 1960, 547-561.

[13] SCHINDLER, W.: *Efficient online tests for true random number generators.* CHES 2001, Lecture Notes in Computer Science Vol. 2162, Berlin 2001, 103-117.

[14] National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-3, June 2009 (also available at http://www.csrc.nist.gov/publications/fips/index.html).

[15] TCVN 11816-1, *Công nghệ thông tin - Các kỹ thuật an toàn - Hàm băm - Phần 1: Tổng quan.*

[16] TCVN 7817-1, *Công nghệ thông tin - Các kỹ thuật an toàn - Quản lý khóa - Phần 1: Tổng quan*

[17] TCVN 11367-1, *Công nghệ thông tin - Các kỹ thuật an toàn - Thuật toán mật mã - Phần 1: Tổng quan*