

TCVN

TIÊU CHUẨN QUỐC GIA

DỰ THẢO

TCVN XXXXX:2026

Xuất bản lần 1

**CÔNG NGHỆ THÔNG TIN – CÁC KỸ THUẬT AN TOÀN –
THUẬT TOÁN CHỮ KÝ SỐ HẬU LƯỢNG TỬ HB-PQSignv1**

*Information technology – Security techniques –
Post-quantum digital signature algorithm HB-PQSignv1*

HÀ NỘI - 2026

Mục lục

Lời nói đầu	5
1. Phạm vi áp dụng	6
2. Tài liệu viện dẫn	6
3. Thuật ngữ và định nghĩa.....	6
4. Ký hiệu và chữ viết tắt.....	8
5. Tổng quan	12
6. Mô tả thuật toán chữ ký số HB-PQSignv1	14
6.1. Các hàm phụ trợ.....	14
6.1.1. Các hàm băm và hàm giả ngẫu nhiên.....	14
6.1.2. Các hàm thành viên	15
6.1.3. Hàm chuyển đổi các mảng, chuỗi byte và số nguyên.....	15
6.1.4. Hàm chuỗi.....	17
6.2. Các thành phần chính.....	17
6.2.1. Lược đồ chữ ký WOTS + ...	17
6.2.2. Lược đồ chữ ký XMSS	20
6.2.3. Siêu cây trong HB-PQSignv1	23
6.2.4. Lược đồ chữ ký FORS	26
6.3. Các thuật toán chữ ký số HB-PQSignv1	29
6.3.1. Thuật toán sinh khóa HB-PQSignv1	29
6.3.2. Thuật toán sinh chữ ký HB-PQSignv1	31
6.3.3. Thuật toán xác minh chữ ký HB-PQSignv1	32
7. Khuyến cáo về các trường hợp sử dụng	32
7.1. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng nguyên bản.....	33
7.1.1. Sinh chữ ký HB-PQSignv1 dạng nguyên bản.....	33
7.1.2. Xác minh chữ ký HB-PQSignv1 dạng nguyên bản	34
7.2. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng băm trước.....	34
7.2.1. Sinh chữ ký HB-PQSignv1 dạng băm trước	34
7.2.2. Xác minh chữ ký HB-PQSignv1 dạng băm trước	35
7.3. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng ký bó.....	35
7.3.1. Sinh chữ ký HB-PQSignv1 dạng ký bó	35
7.3.2. Xác minh chữ ký HB-PQSignv1 dạng ký bó.....	36
8. Quy định về tham số và thuật toán mật mã	37
8.1. Quy định bộ tham số mật mã	37
8.2. Quy định thuật toán mật mã	38

8.2.1.	Bộ thuật toán mật mã	38
8.2.2.	Triển khai cho các hàm băm và giả ngẫu nhiên	38
9.	Một số yêu cầu bổ sung	39
Phụ lục A (quy định) Vec tơ kiểm tra cho HB-PQSignv1		40
Thư mục tài liệu tham khảo		57

Lời nói đầu

Tiêu chuẩn TCVN XXXXX:2026 **THUẬT TOÁN CHỮ KÝ SỐ HẬU LƯỢNG TỬ HB-PQSignv1** do Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ biên soạn, Ban Cơ yếu Chính phủ đề nghị, Ủy ban Tiêu chuẩn Đo lường Chất lượng Quốc gia thẩm định, Bộ Khoa học và Công nghệ công bố.

Tiêu chuẩn TCVN XXXXX:2026 **THUẬT TOÁN CHỮ KÝ SỐ HẬU LƯỢNG TỬ HB-PQSignv1** được xây dựng dựa trên tiêu chuẩn FIPS 205, **Stateless Hash-Based Digital Signature Standard** của Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) với một số bổ sung, sửa đổi.

Công nghệ thông tin – Các kỹ thuật an toàn – Thuật toán chữ ký số hậu lượng tử HB-PQSignv1

Information technology – Security techniques – Post-quantum digital signature algorithm HB-PQSignv1

1. Phạm vi áp dụng

Tiêu chuẩn này quy định thuật toán chữ ký số hậu lượng tử dựa trên hàm băm HB-PQSignv1, gồm thuật toán sinh khóa, sinh chữ ký, xác minh chữ ký, các bộ tham số và ba dạng sử dụng (nguyên bản, băm trước, ký bó). Thuật toán HB-PQSignv1 được xây dựng dựa trên khung cấu trúc SPHINCS+. Thuật toán này được sử dụng để đảm bảo tính xác thực và toàn vẹn dữ liệu trong các ứng dụng mật mã và bảo mật thông tin. Tiêu chuẩn này áp dụng đối với các tổ chức, cá nhân thiết kế, phát triển, đánh giá và sử dụng hệ thống chữ ký số đảm bảo tính an toàn lâu dài trước các tấn công bằng máy tính lượng tử tiềm tàng.

Phạm vi ứng dụng cụ thể của HB-PQSignv1 bao gồm:

- Tích hợp trong thiết bị lưu trữ khóa chuyên dụng (HSM).
- Ký mã nguồn và cập nhật phần mềm an toàn.
- Xác thực trong hệ thống VPN Gateway sử dụng mạng riêng.

2. Tài liệu viện dẫn

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

- TCVN 12853:2020 Công nghệ thông tin – Các kỹ thuật an toàn – Bộ tạo bit ngẫu nhiên.
- TCVN 11816-3:2017 Công nghệ thông tin – Các kỹ thuật an toàn – Hàm băm – Phần 3: Hàm băm chuyên dụng.

3. Thuật ngữ và định nghĩa

Trong tiêu chuẩn này sử dụng các thuật ngữ và định nghĩa sau:

3.1.

Chuỗi byte

Mảng các số nguyên mà trong đó mỗi số nguyên thuộc tập $\{0, \dots, 255\}$

3.2.

Chuỗi băm

Dãy các giá trị được tạo ra bằng cách lặp đi lặp lại phép băm (hash) một giá trị ban đầu. Trong phạm vi của tài liệu này, thuật ngữ chuỗi băm sẽ được sử dụng để định nghĩa các thuật toán thành phần của lược đồ chữ ký WOTS+.

3.3.**Dữ liệu ký**

Dữ liệu hoặc thông điệp mà trên đó một chữ ký số được tạo để cung cấp tính xác thực và toàn vẹn.

3.4.**Khóa riêng**

Khóa mật mã mà được sử dụng trong một thuật toán mật mã phi đối xứng (hay thuật toán mật mã khóa công khai). Khóa riêng được liên kết một cách duy nhất với chủ sở hữu khóa và được giữ một cách bí mật. Trong phạm vi của lược đồ chữ ký số, khóa riêng được sử dụng trong thuật toán sinh chữ ký.

3.5.**Khóa công khai**

Khóa mật mã mà được sử dụng trong một thuật toán mật mã phi đối xứng (hay thuật toán mật mã khóa công khai). Khóa công khai được liên kết với chủ sở hữu khóa và có thể được công khai. Trong phạm vi của lược đồ chữ ký số, khóa công khai được sử dụng với mục đích chính là xác minh một chữ ký số mà đã được sinh ra bằng cách sử dụng khóa riêng tương ứng.

3.6.**Cặp khóa**

Khóa công khai và khóa riêng tương ứng.

3.7.**Chủ sở hữu**

Chủ sở hữu của một cặp khóa là thực thể được cấp phép để sử dụng khóa riêng của cặp khóa đó.

3.8.**Quá trình sinh chữ ký**

Quá trình sử dụng một thuật toán chữ ký và một khóa riêng để tạo một chữ ký số trên dữ liệu.

3.9.**Quá trình xác minh chữ ký**

Quá trình sử dụng một thuật toán chữ ký và một khóa công khai để xác minh tính hợp lệ của một chữ ký số trên dữ liệu.

3.10.**Bên xác minh**

Thực thể xác minh tính hợp lệ của một chữ ký số bằng cách sử dụng khóa công khai.

3.11.**Cây băm Merkle**

Cấu trúc cây nhị phân được định nghĩa dựa trên một **hàm băm**, trong đó mỗi nút của cây hoặc là nút lá (không có nút con), hoặc là nút trong (với hai nút con). Mỗi nút lá của cây băm Merkle chứa một giá trị có độ dài bit bằng với độ dài đầu ra **hàm băm**, và mỗi nút trong chứa một giá trị được tính bằng cách sử dụng **hàm băm** để băm hai nút con của nó.

3.12.**Siêu cây**

Cấu trúc cây gồm nhiều tầng cây XMSS, trong đó mỗi cây XMSS đều là một cây băm Merkle.

3.13.**Đường dẫn xác thực**

Cấu trúc dữ liệu dùng để khôi phục nút gốc của cây băm Merkle. Trong đó, nút gốc của một cây băm Merkle là nút duy nhất không phải là nút con của bất cứ nút nào của cây đó.

3.14.**Địa chỉ**

Cấu trúc dữ liệu được biểu diễn dưới dạng 32-byte giúp xác định vị trí **trong kiến trúc siêu cây của thuật toán chữ ký HB-PQSignv1**. Trong tài liệu này, có 07 kiểu địa chỉ được sử dụng gồm:

- Địa chỉ băm WOTS⁺ (có kiểu WOTS_HASH): giúp xác định chuỗi băm theo lược đồ WOTS⁺ được tính.
- Địa chỉ nén khóa công khai WOTS⁺ (có kiểu WOTS_PK): giúp xác định khóa công khai WOTS⁺ được nén.
- Địa chỉ cây băm (có kiểu TREE): giúp xác định nút bên trong cây XMSS được tính.
- Địa chỉ cây FORS (có kiểu FORS_TREE): giúp xác định nút bên trong cây FORS được tính.
- Địa chỉ nén các nút gốc của tập cây FORS (có kiểu FORS_ROOTS): giúp xác định khóa công khai của tập cây FORS được nén.
- Địa chỉ sinh khóa WOTS⁺ (có kiểu WOTS_PRF): giúp xác định giá trị khóa bí mật WOTS⁺ được sinh.
- Địa chỉ sinh khóa FORS (có kiểu FORS_PRF): giúp xác định giá trị khóa bí mật FORS được sinh.

4. Ký hiệu và chữ viết tắt

FORS	Forest of Random Subsets (Chữ ký số một vài lần dựa trên tổ hợp các tập con ngẫu nhiên)
PQC	Post-Quantum Cryptography (Mật mã hậu lượng tử)
PRF	Pseudorandom Function (Hàm giả ngẫu nhiên)
SHA	Secure Hash Algorithm (Thuật toán hàm băm an toàn)
WOTS ⁺	Winternitz One-Time Signature Plus (Chữ ký một lần dạng Winternitz mở rộng)
XMSS	eXtended Merkle Signature Scheme (Lược đồ chữ ký Merkle mở rộng)
SLH-DSA	StateLess Hash-based Digital Signature Algorithm (Thuật toán chữ ký số không trạng thái dựa trên hàm băm)
HB-PQSignv1	Hash-Based Post-Quantum Signature version 1 (Thuật toán chữ ký số hậu lượng tử dựa trên hàm băm phiên bản 1)
$X \parallel Y$	Phép nối hai chuỗi X và Y . Nếu X là một chuỗi có độ dài ℓ_x , và Y là một chuỗi có độ dài ℓ_y , khi đó $Z = X \parallel Y$ là một chuỗi có độ dài $\ell_x + \ell_y$ sao cho $Z[i] = \begin{cases} X[i], & \text{nếu } 0 \leq i < \ell_x \\ Y[i - \ell_x], & \text{nếu } \ell_x \leq i < \ell_x + \ell_y \end{cases}$
$X[i:j]$	Một chuỗi con của X . Nếu X là một chuỗi có độ dài ℓ_x , $0 \leq i < j \leq \ell_x$, và $Y = X[i:j]$ thì Y là

một chuỗi có độ dài $j - i$ sao cho $Y[k] = X[i + k]$ với $0 \leq k < j - i$.

$\text{Trunc}_\ell(X)$	Một hàm chặt cụt mà đưa ra ℓ byte tận cùng bên trái của chuỗi byte X . Nếu $Y = \text{Trunc}_\ell(X)$, thì Y là một chuỗi byte có độ dài ℓ sao cho $Y[i] = X[i]$ với $0 \leq i < \ell$ (nghĩa là, $Y = X[0:\ell]$).
$ X $	Độ dài (theo byte) của chuỗi byte X .
$\lfloor x \rfloor$	Hàm trả về số nguyên nhỏ nhất không bé hơn x .
$\lceil x \rceil$	Hàm trả về số nguyên lớn nhất không vượt quá x .
$x \bmod y$	Phần dư của số nguyên x khi chia cho số nguyên dương y .
$x \cdot y$	Tích của hai số x và y .
x^y	Lũy thừa bậc y của x .
$\log_2 x$	Logarit cơ số 2 của x .
$\lg_w$	Logarit cơ số 2 của w , nghĩa là $\log_2(w)$
0b	Phần tiền tố đặt trước một số được biểu diễn dưới dạng nhị phân.
0x	Phần tiền tố đặt trước một số được biểu diễn dưới dạng hexa.
$x \gg y$	Phép dịch phải y vị trí đối với x (nghĩa là, $x \gg y = \lfloor x/2^y \rfloor$).
$x \ll y$	Phép dịch trái y vị trí đối với x (nghĩa là, $x \ll y = x \cdot 2^y$).
$x \oplus y$	Phép XOR theo bit của x và y . Ví dụ, $115 \oplus 1 = 114$ (vì $115 \oplus 1 = 0b01110011 \oplus 0b00000001 = 0b01110010 = 114$).
n	Tham số an toàn được sử dụng trong thuật toán chữ ký HB-PQSignv1.
$\mathbb{B}$	Ký hiệu tập tất cả các byte đơn lẻ.
$\mathbb{B}^*$	Ký hiệu tập các chuỗi byte có độ dài bất kỳ.
$\mathbb{B}^n$	Ký hiệu tập các chuỗi byte có độ dài n .
$s \leftarrow x$	Ký hiệu thể hiện biến s được gán bằng x .
$s \xleftarrow{\$} \mathbb{B}^n$	Ký hiệu thể hiện biến s được gán bằng một chuỗi byte có độ dài n được chọn ngẫu nhiên. Một giá trị ngẫu nhiên mới được sinh cho mỗi lần phép gán này được thực hiện.
$\perp$	Ký hiệu thể hiện sự thất bại hoặc thiếu đầu ra từ một thuật toán.
idx	Ký hiệu biểu diễn một chỉ số.
idx_{tree}	Ký hiệu biểu diễn một chỉ số cây (bấm) trong một tầng của siêu cây hoặc một rừng cây FORS.
idx_{leaf}	Ký hiệu biểu diễn một chỉ số lá trong một cây bấm.
AUTH	Ký hiệu biểu diễn đường dẫn xác thực theo một cây bấm.
ADRS	Chuỗi 32-byte biểu diễn địa chỉ của một thành phần trong kiến trúc siêu cây của

thuật toán chữ ký HB-PQSignv1. Tùy thuộc vào từng kiểu địa chỉ, các thuộc tính cụ thể sẽ được quy định:

- Địa chỉ băm WOTS⁺ (có kiểu WOTS_HASH): 4-byte đầu biểu diễn địa chỉ tầng LayerAddress, 12-byte tiếp theo biểu diễn địa chỉ cây TreeAddress, 4-byte tiếp theo biểu diễn kiểu WOTS_HASH (tức là biểu diễn số 0), 4-byte tiếp theo biểu diễn địa chỉ cặp khóa WOTS⁺ KeyPairAddress, 4-byte tiếp theo biểu diễn địa chỉ chuỗi băm ChainAddress, 4-byte cuối biểu diễn địa chỉ băm HashAddress.
- Địa chỉ nén khóa công khai WOTS⁺ (có kiểu WOTS_PK): Tương tự như địa chỉ băm WOTS⁺, nhưng các byte thứ 17 đến 20 biểu diễn kiểu WOTS_PK (tức là biểu diễn số 1) và 8-byte cuối biểu diễn số 0.
- Địa chỉ cây băm (có kiểu TREE): 16-byte đầu tương tự địa chỉ băm WOTS⁺, 4-byte tiếp theo biểu diễn kiểu TREE (tức là biểu diễn số 2), 4-byte tiếp theo biểu diễn số 0, 4-byte tiếp theo biểu diễn độ cao TreeHeight của nút bên trong cây, 4-byte biểu diễn chỉ số TreeIndex của nút bên trong cây có độ cao TreeHeight.
- Địa chỉ cây FORS (có kiểu FORS_TREE): 4-byte đầu biểu diễn địa chỉ tầng LayerAddress = 0, 12-byte tiếp theo biểu diễn địa chỉ cây TreeAddress, 4-byte tiếp theo biểu diễn kiểu FORS_TREE (tức là biểu diễn số 3), 4-byte tiếp theo biểu diễn địa chỉ cặp khóa WOTS⁺ KeyPairAddress để ký khóa công khai của tập cây FORS, 4-byte tiếp theo biểu diễn độ cao TreeHeight của nút bên trong tập cây FORS, 4-byte biểu diễn chỉ số TreeIndex của nút bên trong tập cây FORS có độ cao TreeHeight.
- Địa chỉ nén các nút gốc của tập cây FORS (có kiểu FORS_ROOTS): Tương tự như địa chỉ cây FORS, nhưng các byte thứ 17 đến 20 biểu diễn kiểu FORS_ROOTS (tức là biểu diễn số 4) và 8-byte cuối biểu diễn số 0.
- Địa chỉ sinh khóa WOTS⁺ (có kiểu WOTS_PRF): Tương tự như địa chỉ băm WOTS⁺, nhưng các byte thứ 17 đến 20 biểu diễn kiểu WOTS_PRF (tức là biểu diễn số 5) và 4-byte cuối biểu diễn HashAddress = 0.
- Địa chỉ sinh khóa FORS (có kiểu FORS_PRF): Tương tự như địa chỉ cây FORS, nhưng các byte thứ 17 đến 20 biểu diễn kiểu FORS_PRF (tức là biểu diễn số 6) và các byte thứ 25 đến 28 biểu diễn TreeHeight = 0.

- h* Độ cao của siêu cây được sử dụng trong HB-PQSignv1.
- d* Số tầng của siêu cây được sử dụng trong HB-PQSignv1.
- h'* Độ cao của mỗi cây băm thành phần của siêu cây được sử dụng trong HB-PQSignv1.
- w* Tham số Winternitz trong lược đồ chữ ký thành phần WOTS⁺ của HB-PQSignv1.
- SK Ký hiệu khóa riêng được sử dụng trong HB-PQSignv1, mà bao gồm các thành phần:
- SK.seed – mầm bí mật độ dài *n*-byte
 - SK.prf – giá trị giả ngẫu nhiên bí mật độ dài *n*-byte
 - PK.seed – mầm công khai độ dài *n*-byte
 - PK.root – giá trị băm gốc công khai độ dài *n*-byte

PK	Ký hiệu khóa công khai được sử dụng trong HB-PQSignv1, mà bao gồm các thành phần: - PK.seed – mầm công khai độ dài n-byte - PK.root – giá trị băm gốc công khai độ dài n-byte Lưu ý rằng, trong HB-PQSignv1 khóa công khai PK là một phần của khóa riêng tương ứng.
PH	Hàm được sử dụng trong HB-PQSignv1 theo dạng “băm trước” (xem Mục 7.2) hoặc “ký bó” (xem Mục 7.3) để băm trước nội dung thông điệp hoặc thực hiện các tính toán liên quan đến cây băm, nhằm tạo thông điệp đầu vào cho các thuật toán sinh chữ ký và xác minh chữ ký của HB-PQSignv1.
N	Tham số được sử dụng trong HB-PQSignv1 theo dạng “ký bó” (xem Mục 7.3) để xác định số lượng thông điệp được ký bó đồng thời.
k	Ký hiệu thể hiện số lượng cây con trong một rừng cây FORS được sử dụng trong HB-PQSignv1 .
a	Ký hiệu độ cao của mỗi cây con trong một rừng cây FORS được sử dụng trong HB-PQSignv1 .
M	Ký hiệu thông điệp được ký bởi HB-PQSignv1 hoặc các lược đồ thành phần.
len_1	Độ dài của mảng biểu diễn thông điệp ký trong lược đồ WOTS ⁺ theo cơ số w , mà được tính theo công thức $len_1 = \left\lceil \frac{8 \cdot n}{lg_w} \right\rceil$.
len_2	Độ dài của mảng biểu diễn giá trị tổng kiểm tra tương ứng với thông điệp ký trong lược đồ WOTS ⁺ theo cơ số w , mà được tính theo công thức
$len_2 = \left\lceil \frac{\log_2(len_1 \cdot (w - 1))}{lg_w} \right\rceil + 1.$	
len	Độ dài chuỗi biểu diễn thông điệp ký và giá trị tổng kiểm tra tương ứng trong lược đồ WOTS ⁺ theo cơ số w , mà được tính theo công thức
$len = len_1 + len_2$	
SIG _{WOTS⁺}	Ký hiệu biểu diễn chữ ký số WOTS ⁺ .
SIG _{XMSS}	Ký hiệu biểu diễn chữ ký số XMSS.
SIG _{HT}	Ký hiệu biểu diễn chữ ký số siêu cây.
SIG _{FORS}	Ký hiệu biểu diễn chữ ký số FORS.
$s.getX()$	Hàm này được sử dụng khi cấu trúc dữ liệu s có thuộc tính X là duy nhất, và trả về giá trị của thuộc tính X trong s .
$s.getX(i)$	Hàm này được sử dụng khi cấu trúc dữ liệu s chứa nhiều thuộc tính X , và trả về giá trị của bản thể hiện thứ i của X trong s .
$s.setX(Y)$	Hàm gán thành phần X trong s thành giá trị Y .

5. Tổng quan

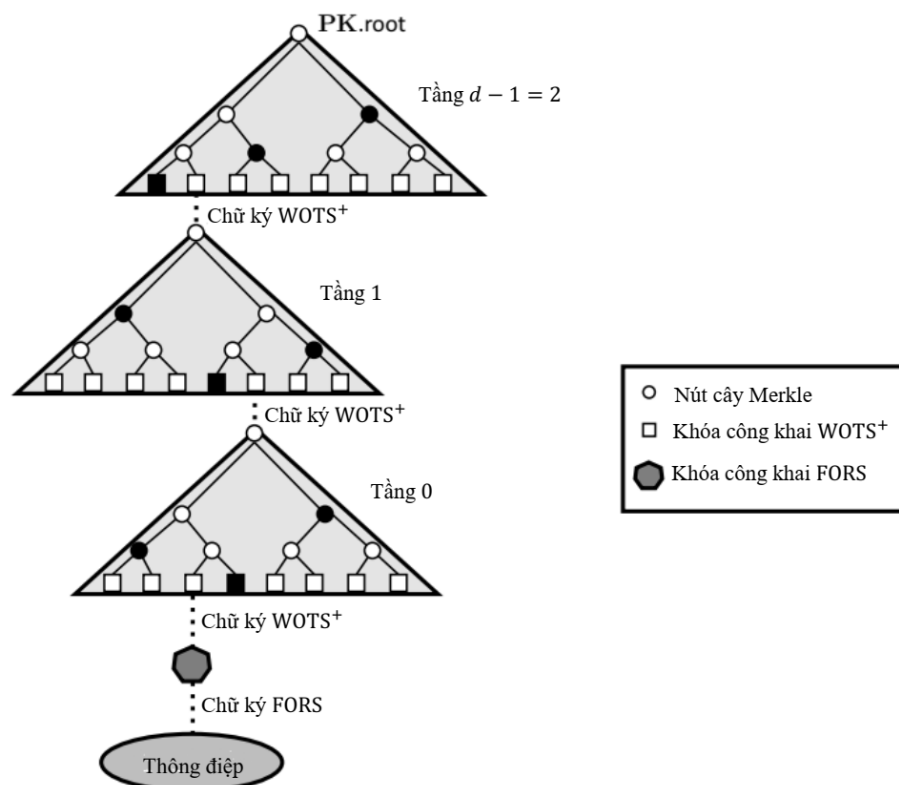
HB-PQSignv1 là một **thuật toán chữ ký số** không trạng thái dựa trên hàm băm theo khung cấu trúc SPHINCS+. Trên thực tế, khung cấu trúc này đã được sử dụng để xây dựng chuẩn chữ ký số hậu lượng tử SLH-DSA của Mỹ [7] và ứng viên chuẩn hóa chữ ký số hậu lượng tử Hypericum của Nga [6]. Mặc dù có chung khung cấu trúc với những **thuật toán** chữ ký kể trên, nhưng HB-PQSignv1 vẫn có những nét riêng mà được thể hiện trong bảng sau.

	SLH-DSA của Mỹ [7]	Hypericum của Nga [6]	HB-PQSignv1
Nguyên lý thiết kế	Khung cấu trúc SPHINCS+	Khung cấu trúc SPHINCS+ theo hướng SPHINCS+C	Khung cấu trúc SPHINCS+
Mô hình an toàn	Mô hình QROM cho tính chất an toàn EUF-CMA với bên đối kháng có năng lực tính toán lượng tử và truy vấn ký cổ điển	Mô hình QROM cho tính chất an toàn EUF-CMA với bên đối kháng có năng lực tính toán lượng tử và truy vấn ký cổ điển	Mô hình QROM cho tính chất an toàn EUF-CMA với bên đối kháng có năng lực tính toán lượng tử và truy vấn ký cổ điển
Hàm băm	Có 02 phiên bản dựa trên: - SHA2 - SHAKE	Chỉ sử dụng hàm băm Streebog	Hiện sử dụng SHA3 (theo tiêu chuẩn TCVN)
Phương pháp chứng minh và lựa chọn tham số an toàn	- Vẫn dựa trên các giả thiết phỏng đoán - Bỏ qua một vài hệ số nhỏ trong đánh giá ngưỡng an toàn. - Ứng với số truy vấn ký tối đa 2^{64} - Có 06 bộ tham số ứng với $n \in \{16, 24, 32\}$; với $n = 32$ cho độ an toàn cỡ 127 bit lượng tử.	- Vẫn dựa trên các giả thiết phỏng đoán. - Có tính đến các hệ số nhỏ trong đánh giá ngưỡng an toàn. - Chỉ có bộ tham số ứng với $n = 256$ cho độ an toàn cỡ 120 bit lượng tử.	- Dựa trên các giả thiết có chứng minh quy dẫn, nhưng kích thước tham số lớn hơn khi so sánh cùng mức an toàn. - Có tính đến các hệ số nhỏ trong đánh giá ngưỡng an toàn. - 3 mức giới hạn số truy vấn ký tối đa. - Có 18 bộ tham số ứng với $n \in \{24, 32\}$; với $n = 32$ cho độ an toàn tối đa cỡ 113 bit lượng tử.
Các dạng sử dụng	- Dạng nguyên bản - Dạng băm trước	- Dạng nguyên bản	- Dạng nguyên bản - Dạng băm trước - Dạng ký bó

Về mặt hình dung, HB-PQSignv1 được xây dựng bằng cách sử dụng các lược đồ chữ ký số dựa trên hàm băm khác làm thành phần: (1) lược đồ chữ ký một vài lần dựa trên tổ hợp các tập con ngẫu nhiên, mà được ký hiệu là FORS, và (2) một lược đồ chữ ký nhiều lần, tức lược đồ XMSS. Nhắc lại rằng, XMSS được xây dựng bằng cách sử dụng lược đồ chữ ký số một lần dựa trên hàm băm, cụ thể là WOTS⁺ làm thành phần chính.

Về mặt khái niệm, một cặp khóa HB-PQSignv1 bao gồm một lượng rất lớn các cặp khóa FORS. Lược đồ chữ ký một vài lần FORS cho phép mỗi cặp khóa được sử dụng để ký an toàn cho một lượng nhỏ

thông điệp. Một chữ ký HB-PQSignv1 được tạo bằng cách tính một giá trị băm (ngẫu nhiên hóa) của thông điệp, tiếp theo sử dụng một phần của giá trị băm nhận được để lựa chọn một khóa FORS theo cách giả ngẫu nhiên, và cuối cùng ký phần còn lại của giá trị băm nhận được với khóa đó. Một chữ ký HB-PQSignv1 bao gồm chữ ký FORS và thông tin để xác thực khóa công khai FORS. Thông tin xác thực được tạo bằng cách sử dụng các chữ ký XMSS.



Hình 1. Minh họa chữ ký HB-PQSignv1 với $d = 3$ tầng.

XMSS là một lược đồ chữ ký nhiều lần mà được tạo bằng cách kết hợp các chữ ký một lần WOTS⁺ và các cây băm Merkle. Một khóa XMSS bao gồm $2^{h'}$ khóa WOTS⁺ và có thể ký $2^{h'}$ thông điệp. Các khóa công khai WOTS⁺ được tạo thành một cây Merkle, và gốc của cây là khóa công khai XMSS. (Cây băm Merkle được hình thành từ các khóa WOTS⁺ cũng được coi là một cây XMSS.) Một chữ ký XMSS bao gồm một chữ ký WOTS⁺ và một đường dẫn xác thực (mà là một phần của cây băm Merkle) cho khóa công khai WOTS⁺. Trong **Hình 1**, các tam giác biểu diễn các cây XMSS, các hình vuông biểu thị khóa công khai WOTS⁺, và các hình tròn biểu diễn các nút bên trong của cây băm. Bên trong một cây XMSS, **tập hợp các hình tròn và hình vuông được tô đen** biểu diễn đường dẫn xác thực cho khóa công khai WOTS⁺ cần thiết để xác minh chữ ký.

Thông tin xác thực cho một khóa công khai FORS là một *chữ ký siêu cây (hypertree signature)*, như được mô tả trong **Hình 1**. Cây bao gồm d tầng mà trong đó tầng cao nhất (tức tầng $d - 1$) bao gồm một cây XMSS duy nhất, tầng ngay phía dưới (tức tầng $d - 2$) bao gồm $2^{h'}$ cây XMSS, và tầng thấp nhất (tức tầng 0) bao gồm $2^{(d-1)h'}$ cây XMSS. Khóa công khai trong khóa XMSS ở mỗi tầng 0 đến $d - 2$ được ký bởi một khóa (bí mật) XMSS ở tầng ngay phía trên. Các khóa XMSS ở tầng 0 bao hàm tổng cộng $2^{dh'} = 2^h$ khóa WOTS⁺, mà được sử dụng để ký 2^h khóa công khai FORS trong cặp khóa HB-PQSignv1. Dãy gồm d chữ ký XMSS cần thiết để xác thực một khóa FORS mà bắt đầu với khóa công khai của khóa XMSS tại tầng $d - 1$ được xem là một chữ ký siêu cây.

Một khóa công khai HB-PQSignv1 (**Hình 7**) bao gồm hai chuỗi thành phần có độ dài n -byte: (1) **PK.root**, mà là khóa công khai của khóa XMSS tại tầng $d - 1$, và (2) **PK.seed**, mà được sử dụng để giúp phân biệt miền giữa các cặp khóa HB-PQSignv1 khác nhau. Một khóa riêng (bí mật) HB-PQSignv1 (**Hình 8**) bao gồm một mầm **SK.seed** có độ dài n -byte mà được sử dụng để sinh giả ngẫu nhiên tất cả các giá trị bí mật cho các khóa WOTS⁺ và FORS, cùng một khóa **SK.prf** có độ dài n -byte mà được sử dụng trong quá trình tính giá trị băm (ngẫu nhiên hóa) của thông điệp. Một khóa riêng HB-PQSignv1 cũng bao gồm các bản sao của **PK.root** và **PK.seed**, vì những giá trị này đều cần thiết trong cả quá trình sinh và xác minh chữ ký.

Lược đồ chữ ký một lần WOTS⁺ được đặc tả trong Mục 6.2.1, và lược đồ chữ ký nhiều lần XMSS được đặc tả trong Mục 6.2.2. Mục 6.2.3 đặc tả quá trình sinh và xác minh của các chữ ký siêu cây. Lược đồ chữ ký một vài lần FORS được đặc tả trong Mục 6.2.4. Cuối cùng, Mục 6.3 đặc tả các hàm sinh khóa, hàm sinh chữ ký và hàm xác minh chữ ký. Vì các lược đồ WOTS⁺, XMSS, siêu cây và FORS được đề cập trong tiêu chuẩn này không được nhắm đến để sử dụng như các lược đồ chữ ký đứng độc lập, do vậy chỉ các thành phần của chúng có liên quan đến HB-PQSignv1 mới được mô tả rõ ràng. **Cụ thể, tiêu chuẩn này không quy định hàm sinh cặp khóa cho các lược đồ thành phần, và hàm xác minh chữ ký chỉ được quy định đối với chữ ký siêu cây.**

Khi được sử dụng **trong HB-PQSignv1**, các chữ ký WOTS⁺, XMSS, và FORS được xác minh ẩn bằng cách sử dụng các hàm để sinh các khóa công khai từ các thông điệp và chữ ký (xem Mục 6.2.1.3, 6.2.2.3, và 6.2.4.4). Khi xác minh một chữ ký HB-PQSignv1, giá trị băm (ngẫu nhiên hóa) của thông điệp và chữ ký FORS được sử dụng để tính ra một khóa công khai FORS ứng viên. Khóa công khai này và chữ ký WOTS⁺ từ khóa XMSS ở tầng 0 được sử dụng để tính ra một khóa công khai WOTS⁺ ứng viên, mà sau đó được sử dụng cùng với đường dẫn xác thực tương ứng để tính ra một khóa XMSS ứng viên ở tầng 0. Khóa công khai ứng viên ở tầng 0 được sử dụng cùng với chữ ký XMSS ở tầng 1 để tính khóa công khai XMSS ứng viên ở tầng 1. Quá trình này được lặp lại cho đến khi một khóa công khai XMSS ứng viên ở tầng $d - 1$ được tính. Quá trình xác minh chữ ký HB-PQSignv1 thành công nếu khóa công khai XMSS ứng viên ở tầng $d - 1$ trùng với khóa công khai HB-PQSignv1 gốc là **PK.root**.

6. Mô tả thuật toán chữ ký số HB-PQSignv1

6.1. Các hàm phụ trợ

6.1.1. Các hàm băm và hàm giả ngẫu nhiên

Đặc tả của HB-PQSignv1 quy định sử dụng 06 hàm cơ sở – **PRF_{msg}**, **H_{msg}**, **PRF**, **T_ℓ**, **H**, **F** – mà đều được thiết lập bằng cách sử dụng các hàm băm với **độ dài đầu ra cố định**:

- **PRF_{msg}**(**SK.prf**, *opt_rand*, *M*) là một hàm giả ngẫu nhiên từ $\mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^* \rightarrow \mathbb{B}^n$ mà sinh thành phần ngẫu nhiên *R* để tính giá trị băm (ngẫu nhiên hóa) của thông điệp ký.
- **H_{msg}**(*R*, **PK.seed**, **PK.root**, *M*) là một hàm băm từ $\mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^* \rightarrow \mathbb{B}^m$ mà được sử dụng để tính giá trị băm cho thông điệp ký.
- **PRF**(**PK.seed**, **SK.seed**, **ADRS**) là một hàm giả ngẫu nhiên từ $\mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^{32} \rightarrow \mathbb{B}^n$ mà được sử dụng để sinh các giá trị bí mật trong các khóa riêng FORS và WOTS⁺.
- **T_ℓ**(**PK.seed**, **ADRS**, *M_ℓ*) là một hàm băm từ $\mathbb{B}^n \times \mathbb{B}^{32} \times \mathbb{B}^{\ell n} \rightarrow \mathbb{B}^n$ mà được sử dụng để ánh xạ một thông điệp ℓn -byte thành một thông điệp n -byte.

- $H(\text{PK.seed}, \text{ADRS}, M_2)$ là một trường hợp đặc biệt của T_ℓ mà lấy đầu vào là thông điệp $2n$ -byte, nghĩa là một hàm băm từ $\mathbb{B}^n \times \mathbb{B}^{32} \times \mathbb{B}^{2n} \rightarrow \mathbb{B}^n$.
- $F(\text{PK.seed}, \text{ADRS}, M_1)$ là một hàm băm từ $\mathbb{B}^n \times \mathbb{B}^{32} \times \mathbb{B}^n \rightarrow \mathbb{B}^n$ mà lấy đầu vào là một thông điệp n -byte và trả về một đầu ra n -byte.

Ngoài ra, đối với các dạng băm trước và ký bó (trong các **Thuật toán 22**, **Thuật toán 23**, **Thuật toán 24**, **Thuật toán 25**), còn sử dụng thêm hàm: $\text{PH}(\text{sid}, \text{TW}, M)$ là một hàm băm từ $\mathbb{B}^{2n} \times \mathbb{B}^* \times \mathbb{B}^* \rightarrow \mathbb{B}^{64}$. Các thể hiện cụ thể cho những hàm này sẽ được quy định trong Mục 8.2.2.

6.1.2. Các hàm thành viên

Các thuật toán trong tài liệu này yêu cầu sử dụng các *hàm thành viên (member function)* để cập nhật địa chỉ **ADRS** được sử dụng như đầu vào cho các hàm phụ trợ **PRF**, T_ℓ , **H**, **F**. Trong các hàm dưới đây, với X là thuộc tính của **ADRS** và một giá trị Y , thì ký hiệu **ADRS.getX()** và **ADRS.setX(Y)** tuân theo đặc tả ở Mục 4.

Hàm thành viên	Ký hiệu mở rộng
ADRS.setLayerAddress (l)	$\text{ADRS} \leftarrow \text{toByte}(l, 4) \parallel \text{ADRS}[4: 32]$
ADRS.setTreeAddress (t)	$\text{ADRS} \leftarrow \text{ADRS}[0: 4] \parallel \text{toByte}(t, 12) \parallel \text{ADRS}[16: 32]$
ADRS.setTypeAndClear (Y)	$\text{ADRS} \leftarrow \text{ADRS}[0: 16] \parallel \text{toByte}(Y, 4) \parallel \text{toByte}(0, 12)$
ADRS.setKeyPairAddress (i)	$\text{ADRS} \leftarrow \text{ADRS}[0: 20] \parallel \text{toByte}(i, 4) \parallel \text{ADRS}[24: 32]$
ADRS.setChainAddress (i) ADRS.setTreeHeight (i)	$\text{ADRS} \leftarrow \text{ADRS}[0: 24] \parallel \text{toByte}(i, 4) \parallel \text{ADRS}[28: 32]$
ADRS.setHashAddress (i) ADRS.setTreeIndex (i)	$\text{ADRS} \leftarrow \text{ADRS}[0: 28] \parallel \text{toByte}(i, 4)$
$i \leftarrow \text{ADRS.getKeyPairAddress}()$	$i \leftarrow \text{toInt}(\text{ADRS}[20: 24], 4)$
$i \leftarrow \text{ADRS.getTreeIndex}()$	$i \leftarrow \text{toInt}(\text{ADRS}[28: 32], 4)$

6.1.3. Hàm chuyển đổi các mảng, chuỗi byte và số nguyên

Nếu X là một chuỗi byte có độ dài n , thì $X[i]$ với $i \in \{0, \dots, n-1\}$ sẽ tham chiếu đến byte thứ i trong chuỗi X . Nếu X là một mảng gồm m chuỗi n -byte, thì $X[i]$ với $i \in \{0, \dots, m-1\}$ sẽ tham chiếu đến chuỗi n -byte thứ i trong X , và X sẽ tham chiếu đến chuỗi $(m \cdot n)$ -byte $X[0] \parallel X[1] \parallel \dots \parallel X[m-1]$.

Một chuỗi byte có thể được phân tích như biểu diễn dạng big-endian của một số nguyên. Trong những trường hợp đó, một chuỗi byte X có độ dài n sẽ được chuyển thành số nguyên

$$X[0] \cdot 256^{n-1} + X[1] \cdot 256^{n-2} + \dots + X[n-2] \cdot 256 + X[n-1].$$

Thuật toán 1 mô tả hàm chuyển một chuỗi byte X có độ dài n thành một số nguyên theo cách kể trên.

Thuật toán 1 $\text{toInt}(X, n)$ – Chuyển một chuỗi byte thành một số nguyên
Đầu vào: Chuỗi byte X có độ dài n .
Đầu ra: Giá trị nguyên của X .
1: $total \leftarrow 0$

```
2: for  $i$  from 0 to  $n - 1$  do
3:    $total \leftarrow 256 \cdot total + X[i]$ 
4: end for
5: return  $total$ 
```

Tương tự, một số nguyên x có thể được chuyển thành một chuỗi byte có độ dài n bằng cách tìm các hệ số $x_0, x_1, \dots, x_{n-2}, x_{n-1} \in \{0, \dots, 255\}$ sao cho

$$x = x_0 \cdot 256^{n-1} + x_1 \cdot 256^{n-2} + \dots + x_{n-2} \cdot 256 + x_{n-1},$$

và sau đó gán chuỗi byte kết quả thành $x_0 \parallel x_1 \parallel \dots \parallel x_{n-2} \parallel x_{n-1}$. **Thuật toán 2** mô tả hàm chuyển đổi một số nguyên thành một chuỗi byte có độ dài n theo cách kể trên.

Thuật toán 2 toByte(x, n) – Chuyển một số nguyên thành một chuỗi byte

Đầu vào: số nguyên x , độ dài n .
Đầu ra: chuỗi byte có độ dài n để biểu diễn dạng big-endian của x theo byte.

```
1:  $total \leftarrow x$ 
2: for  $i$  from 0 to  $n - 1$  do
3:    $S[n - 1 - i] \leftarrow total \bmod 256$ 
4:    $total \leftarrow total \gg 8$ 
5: end for
6: return  $S$ 
```

Đối với các lược đồ WOTS⁺ và FORS, các thông điệp ký cần được phân tách thành một dãy các chuỗi b -bit, mà ở đó mỗi chuỗi b -bit biểu diễn một số nguyên từ 0 đến $2^b - 1$.¹ Điều này là tương đương với việc tìm biểu diễn theo cơ số 2^b cho thông điệp. Hàm base_2 ^{b} (**Thuật toán 3**) lấy đầu vào là một chuỗi byte X , một độ dài bit b , và một độ dài đầu ra out_len để đưa ra một mảng các số nguyên theo cơ số 2^b mà biểu diễn $(out_len \cdot b)$ -bit đầu tiên của X nếu các byte riêng lẻ của X được biểu diễn như các chuỗi 8-bit theo dạng big-endian. X phải có độ dài tối thiểu là $\lceil out_len \cdot b/8 \rceil$ theo byte.

Thuật toán 3 base_2 ^{b} (X, b, out_len) – Tính biểu diễn cơ số 2^b của X

Đầu vào: Chuỗi byte X có độ dài tối thiểu $\lceil out_len \cdot b/8 \rceil$, số nguyên b , độ dài đầu ra out_len .
Đầu ra: Mảng gồm out_len số nguyên thuộc miền $[0, \dots, 2^b - 1]$

```
1:  $in \leftarrow 0$ 
2:  $bits \leftarrow 0$ 
3:  $total \leftarrow 0$ 
4: for  $out$  from 0 to  $out\_len - 1$  do
5:   while  $bits < b$  do
6:      $total \leftarrow (total \ll 8) + X[in]$ 
```

¹ b sẽ là giá trị $\log_2 w$ khi hàm base_2 ^{b} được sử dụng trong lược đồ WOTS⁺, và b sẽ là giá trị của a khi hàm base_2 ^{b} được sử dụng trong lược đồ FORS.

```

7:       $in \leftarrow in + 1$ 
8:       $bits \leftarrow bits + 8$ 
9:  end while
10:    $bits \leftarrow bits - b$ 
11:    $baseb[out] \leftarrow (total \gg bits) \bmod 2^b$ 
12: end for
13: return  $baseb$ 

```

Lưu ý rằng, trong **Thuật toán 3**, vì giá trị của biến $bits$ không bao giờ vượt quá $b + 7$, nên một số nguyên không dấu $(b + 7)$ -bit là đủ để lưu trữ biến $total$ (nghĩa là, $total$ có thể được lưu trữ như một số nguyên không dấu 32-bit).

6.1.4. Hàm chuỗi

Hàm chuỗi chain là một thành phần quan trọng để tính các chuỗi băm cho thành phần lược đồ chữ ký $WOTS^+$ của **HB-PQSignv1**. Hàm này nhận đầu vào là một chuỗi n -byte X cùng các số nguyên s và i (ở đó $i + s < w$) để đưa ra kết quả sau khi lặp lại s lần hàm băm F kể từ chỉ số i .² Hàm chuỗi cũng yêu cầu các đầu vào khác là **PK.seed**, mà là một phần của khóa công khai **HB-PQSignv1**, và một địa chỉ **ADRS**. Địa chỉ này phải có kiểu **WOTS_HASH**, và địa chỉ tầng, địa chỉ cây, địa chỉ cặp khóa, và địa chỉ chuỗi phải được đặt tương ứng với địa chỉ của chuỗi băm đang được tính.

Thuật toán 4 $chain(X, i, s, \mathbf{PK.seed}, \mathbf{ADRS})$ – Hàm chuỗi được sử dụng trong $WOTS^+$

Đầu vào: Chuỗi đầu vào X , chỉ số bắt đầu i , số bước lặp s , mầm công khai **PK.seed**, địa chỉ **ADRS**.

Đầu ra: Giá trị đầu ra của F được lặp lại s lần trên đầu vào X .

```

1:  $tmp \leftarrow X$ 
2: for  $j$  from  $i$  to  $i + s - 1$  do
3:    $\mathbf{ADRS.setHashAddress}(j)$ 
4:    $tmp \leftarrow F(\mathbf{PK.seed}, \mathbf{ADRS}, tmp)$ 
5: end for
6: return  $tmp$ 

```

6.2. Các thành phần chính

6.2.1. Lược đồ chữ ký $WOTS^+$

Mục này quy định thành phần $WOTS^+$ của thuật toán chữ ký **HB-PQSignv1**. Trong đó, thành phần này được đặc tả thông qua các *thuật toán sinh khóa công khai $WOTS^+$* (Mục 6.2.1.1), *thuật toán sinh chữ ký $WOTS^+$* (Mục 6.2.1.2) và *thuật toán tính toán khóa công khai từ chữ ký $WOTS^+$* (Mục 6.2.1.3). Ở đây, lược đồ $WOTS^+$ sử dụng hai tham số. Đó là tham số an toàn n (theo byte) và tham số w (hay tham số Winternitz). Hai tham số trên được sử dụng để tính các giá trị bổ sung len_1 , len_2 và len (xem Mục 4).

6.2.1.1. Sinh khóa công khai $WOTS^+$

Một khóa riêng $WOTS^+$ bao gồm len giá trị bí mật có độ dài n -byte. Trong **HB-PQSignv1**, những giá trị này được sinh từ một mầm **SK.seed** có độ dài n -byte bằng cách sử dụng hàm giả ngẫu nhiên **PRF**. Các

² Giá trị ban đầu của chuỗi băm có chỉ số bằng 0.

chuỗi băm có độ dài w sau đó được tạo từ các giá trị bí mật bằng cách sử dụng một hàm chuỗi, và các giá trị nhận được sau cùng ở mỗi chuỗi băm đều là các giá trị công khai. Khóa công khai $WOTS^+$ được tính như giá trị băm của những giá trị công khai này.

Hàm `wots_pkGen` (xem **Thuật toán 5**) sinh các khóa công khai $WOTS^+$. Hàm này lấy **SK**.seed và **PK**.seed từ khóa riêng HB-PQSignv1 cùng một địa chỉ **ADRS** làm đầu vào. Địa chỉ này phải có kiểu `WOTS_HASH`, và địa chỉ tầng, địa chỉ cây, và địa chỉ cặp khóa phải được đặt tương ứng với địa chỉ của khóa công khai $WOTS^+$ cần được tính.

Thuật toán 5 `wots_pkGen(SK.seed, PK.seed, ADRS)` – Sinh khóa công khai $WOTS^+$

Đầu vào: Mầm bí mật **SK**.seed, mầm công khai **PK**.seed, địa chỉ **ADRS**.

Đầu ra: Khóa công khai $WOTS^+$ pk_{WOTS^+} .

```

1: skADRS  $\leftarrow$  ADRS
2: skADRS.setTypeAndClear(WOTS_PRF)
3: skADRS.setKeyPairAddress(ADRS.getKeyPairAddress())
4: for  $i$  from 0 to  $len - 1$  do
5:   skADRS.setChainAddress( $i$ )
6:    $sk \leftarrow$  PRF(PK.seed, SK.seed, skADRS)
7:   ADRS.setChainAddress( $i$ )
8:    $tmp[i] \leftarrow$  chain( $sk$ , 0,  $w - 1$ , PK.seed, ADRS)
9: end for
10: wotspkADRS  $\leftarrow$  ADRS
11: wotspkADRS.setTypeAndClear(WOTS_PK)
12: wotspkADRS.setKeyPairAddress(ADRS.getKeyPairAddress())
13:  $pk_{WOTS^+} \leftarrow$  Tlen(PK.seed, wotspkADRS,  $tmp$ )
14: return  $pk_{WOTS^+}$ 

```

6.2.1.2. Sinh chữ ký $WOTS^+$

Một chữ ký $WOTS^+$ là một mảng gồm len chuỗi byte có độ dài n , như được biểu diễn trong **Hình 2**. Để tạo một chữ ký, thông điệp $8n$ -bit trước tiên được chuyển thành một mảng gồm len_1 số nguyên theo cơ số w . Một giá trị tổng kiểm tra sau đó được tính cho chuỗi này, và giá trị tổng kiểm tra sẽ được chuyển thành một mảng gồm len_2 số nguyên theo cơ số w . Chữ ký bao gồm các phần tử thích hợp từ các chuỗi băm tương ứng với từng số nguyên trong các mảng thông điệp và tổng kiểm tra.

SIG _{WOTS⁺} [0]	n -byte
...	
SIG _{WOTS⁺} [$len - 1$]	n -byte

Hình 2. Định dạng dữ liệu cho chữ ký $WOTS^+$

Quá trình sinh chữ ký $WOTS^+$ được thực hiện thông qua hàm `wots_sign` (xem **Thuật toán 6**). Ngoài thông

điệp cần ký có độ dài n -byte, hàm `wots_sign` còn lấy các đầu vào khác là **SK**.seed và **PK**.seed từ khóa riêng HB-PQSignv1 cùng một địa chỉ **ADRS**. Địa chỉ này phải có kiểu `WOTS_HASH`, và địa chỉ tầng, địa chỉ cây, và địa chỉ cặp khóa phải được đặt tương ứng với địa chỉ của khóa công khai $WOTS^+$ mà được sử dụng để ký thông điệp.

Thuật toán 6 <code>wots_sign(M, SK.seed, PK.seed, ADRS)</code> – Sinh một chữ ký $WOTS^+$ trên một thông điệp M có độ dài n -byte
Đầu vào: Thông điệp M, mầm bí mật SK.seed, mầm công khai PK.seed, địa chỉ ADRS. Đầu ra: Chữ ký $WOTS^+$ SIG_{WOTS^+}. <pre> 1: $csum \leftarrow 0$ 2: $msg \leftarrow \text{base_2}^b(M, lg_w, len_1)$ 3: for i from 0 to $len_1 - 1$ do 4: $csum \leftarrow csum + w - 1 - msg[i]$ 5: end for 6: $csum \leftarrow csum \ll \left(\left(8 - ((len_2 \cdot lg_w) \bmod 8) \right) \bmod 8 \right)$ 7: $msg \leftarrow msg \parallel \text{base_2}^b \left(\text{toByte} \left(csum, \left\lceil \frac{len_2 \cdot lg_w}{8} \right\rceil \right), lg_w, len_2 \right)$ 8: $skADRS \leftarrow \text{ADRS}$ 9: $skADRS.setTypeAndClear(WOTS_PRF)$ 10: $skADRS.setKeyPairAddress(ADRS.getKeyPairAddress())$ 11: for i from 0 to $len - 1$ do 12: $skADRS.setChainAddress(i)$ 13: $sk \leftarrow PRF(PK.seed, SK.seed, skADRS)$ 14: $ADRS.setChainAddress(i)$ 15: $SIG_{WOTS^+}[i] \leftarrow \text{chain}(sk, 0, msg[i], PK.seed, ADRS)$ 16: end for 17: return SIG_{WOTS^+} </pre>

6.2.1.3. Tính toán khóa công khai $WOTS^+$ từ chữ ký

Việc xác minh một chữ ký $WOTS^+$ bao gồm tính toán một giá trị khóa công khai từ một thông điệp và một chữ ký. Quá trình xác minh thành công nếu giá trị khóa công khai đúng được tính ra. Tính “đúng” được xác định bằng cách sử dụng giá trị khóa công khai nhận được cùng với thông tin khác để tính một giá trị **PK**.root ứng viên và sau đó so sánh giá trị đó với giá trị khóa công khai gốc từ khóa công khai HB-PQSignv1. Mục này mô tả thuật toán `wots_pkFromSig` (**Thuật toán 7**), một hàm mà tính khóa công khai $WOTS^+$ ứng viên từ một chữ ký $WOTS^+$ và thông điệp tương ứng.

Ngoài một thông điệp M có độ dài n -byte và một chữ ký SIG_{WOTS^+} có độ dài $(len \cdot n)$ -byte, mà được biểu diễn thành một mảng gồm len chuỗi có độ dài n -byte, hàm `wots_pkFromSig` nhận **PK**.seed từ khóa công khai HB-PQSignv1 và một địa chỉ **ADRS** làm đầu vào. Địa chỉ này phải có kiểu `WOTS_HASH`, và địa chỉ tầng, địa chỉ cây, và địa chỉ cặp khóa phải được đặt tương ứng với địa chỉ của khóa công khai $WOTS^+$ mà được sử dụng để ký thông điệp.

Thuật toán 7 $\text{wots_pkFromSig}(\text{SIG}_{\text{WOTS}^+}, M, \text{PK. seed}, \text{ADRS})$ – Tính một khóa công khai WOTS^+ từ một thông điệp và chữ ký WOTS^+

Đầu vào: Chữ ký WOTS^+ $\text{SIG}_{\text{WOTS}^+}$, thông điệp M , mầm công khai PK. seed , địa chỉ ADRS .

Đầu ra: Khóa công khai WOTS^+ pk_{WOTS^+} được dẫn xuất từ $\text{SIG}_{\text{WOTS}^+}$.

```

1:  $csum \leftarrow 0$ 
2:  $msg \leftarrow \text{base\_2}^b(M, \lg_w, len_1)$ 
3: for  $i$  from 0 to  $len_1 - 1$  do
4:    $csum \leftarrow csum + w - 1 - msg[i]$ 
5: end for
6:  $csum \leftarrow csum \ll \left( \left( 8 - ((len_2 \cdot \lg_w) \bmod 8) \right) \bmod 8 \right)$ 
7:  $msg \leftarrow msg \parallel \text{base\_2}^b\left(\text{toByte}\left(csum, \left\lceil \frac{len_2 \cdot \lg_w}{8} \right\rceil\right), \lg_w, len_2\right)$ 
8: for  $i$  from 0 to  $len - 1$  do
9:    $\text{ADRS.setChainAddress}(i)$ 
10:   $tmp[i] \leftarrow \text{chain}(\text{SIG}_{\text{WOTS}^+}[i], msg[i], w - 1 - msg[i], \text{PK. seed}, \text{ADRS})$ 
11: end for
12:  $\text{wotspkADRS} \leftarrow \text{ADRS}$ 
13:  $\text{wotspkADRS.setTypeAndClear}(\text{WOTS\_PK})$ 
14:  $\text{wotspkADRS.setKeyPairAddress}(\text{ADRS.getKeyPairAddress}())$ 
15:  $pk_{\text{WOTS}^+} \leftarrow \text{T}_{len}(\text{PK. seed}, \text{wotspkADRS}, tmp)$ 
16: return  $pk_{\text{WOTS}^+}$ 

```

6.2.2. Lược đồ chữ ký XMSS

XMSS mở rộng lược đồ chữ ký WOTS^+ thành một lược đồ chữ ký mà có thể ký nhiều lần. Một cây Merkle có độ cao h' được sử dụng để cho phép $2^{h'}$ khóa công khai WOTS^+ được xác thực bằng một khóa công khai XMSS duy nhất với độ dài n -byte, mà là giá trị băm gốc của cây Merkle. Vì mỗi khóa WOTS^+ có thể được sử dụng để ký một thông điệp, nên khóa XMSS có thể được dùng để ký $2^{h'}$ thông điệp.

$\text{SIG}_{\text{WOTS}^+}$	$(len \cdot n)$ -byte
$\text{AUTH}_{\text{XMSS}}[0]$	n -byte
...	
$\text{AUTH}_{\text{XMSS}}[h' - 1]$	n -byte

Hình 3. Định dạng dữ liệu của chữ ký XMSS

Một chữ ký XMSS là một chuỗi có độ dài $((h' + len) \cdot n)$ -byte và chứa một chữ ký WOTS^+ và một đường dẫn xác thực (xem **Hình 3**). Đường dẫn xác thực là một dãy các nút từ cây Merkle – mỗi nút sẽ tương

ứng với từng mức (theo độ cao) của cây, ngoại trừ nút gốc – mà cho phép người xác minh tính nút gốc của cây Merkle khi được dùng kết hợp với khóa công khai WOTS⁺ (vốn có thể được tính từ chữ ký WOTS⁺).

6.2.2.1. Tính một nút cây băm Merkle

Hàm `xmss_node` (**Thuật toán 8**) tính **một nút** bên trong một cây XMSS. Hàm này nhận các đầu vào sau:

- Các mầm **SK**.seed và **PK**.seed từ khóa riêng HB-PQSignv1;
- Một chỉ số nút mục tiêu i , mà là chỉ số của nút đang được tính;
- Một độ cao nút mục tiêu z , mà là độ cao bên trong cây Merkle của nút đang được tính.
- Một địa chỉ, mà phải có các trường địa chỉ tầng và địa chỉ cây được đặt tương ứng với cây XMSS mà bên trong đó nút đang được tính.

Lưu ý rằng, các tham số đầu vào z (độ cao của nút mục tiêu) và i (chỉ số của nút mục tiêu) phải thỏa mãn điều kiện $z \leq h'$ và $i < 2^{(h'-z)}$.

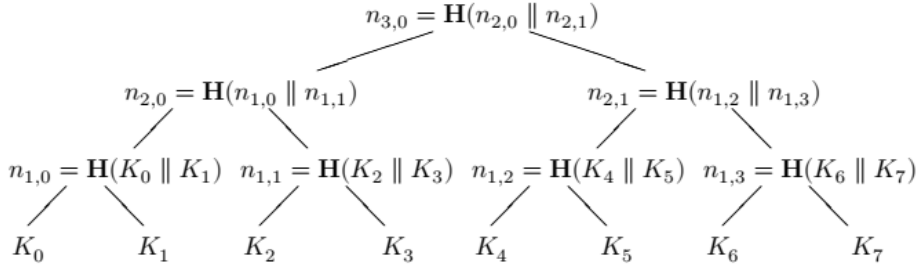
Thuật toán 8 <code>xmss_node(SK.seed, i, z, PK.seed, ADRS)</code> – Tính nút của cây băm Merkle
Đầu vào: Mầm bí mật SK.seed, chỉ số nút mục tiêu i, độ cao nút mục tiêu z, mầm công khai PK.seed, địa chỉ ADRS. Đầu ra: chuỗi <i>node</i> có độ dài n-byte là giá trị nút thứ i tại độ cao z của cây băm Merkle. <pre> 1: if $z = 0$ then 2: <code>ADRS.setTypeAndClear(WOTS_HASH)</code> 3: <code>ADRS.setKeyPairAddress(i)</code> 4: $node \leftarrow \text{wots_pkGen}(\text{SK.seed}, \text{PK.seed}, \text{ADRS})$ 5: else 6: $lnode \leftarrow \text{xmss_node}(\text{SK.seed}, 2i, z - 1, \text{PK.seed}, \text{ADRS})$ 7: $rnode \leftarrow \text{xmss_node}(\text{SK.seed}, 2i + 1, z - 1, \text{PK.seed}, \text{ADRS})$ 8: <code>ADRS.setTypeAndClear(TREE)</code> 9: <code>ADRS.setTreeHeight(z)</code> 10: <code>ADRS.setTreeIndex(i)</code> 11: $node \leftarrow H(\text{PK.seed}, \text{ADRS}, lnode \parallel rnode)$ 12: end if 13: return $node$ </pre>

6.2.2.2. Sinh một chữ ký XMSS

Hàm `xmss_sign` (**Thuật toán 9**) tạo một chữ ký XMSS trên một thông điệp³ M có độ dài n -byte bằng cách tạo một đường dẫn xác thực (dòng 1 đến 4) và ký M với khóa WOTS⁺ phù hợp (dòng 5 đến 7). Ngoài M , `xmss_sign` còn lấy các đầu vào khác là mầm **SK**.seed và **PK**.seed từ khóa riêng HB-PQSignv1, một địa chỉ **ADRS**, và một

³ Trong HB-PQSignv1, thông điệp M mà được ký bằng cách sử dụng XMSS là một khóa công khai XMSS hoặc một khóa công khai FORS.

chỉ số idx . Địa chỉ **ADRS** phải có địa chỉ tầng và địa chỉ cây tương thích với khóa XMSS mà đang được sử dụng để ký thông điệp, và chỉ số idx phải là chỉ số của khóa WOTS⁺ bên trong cây XMSS mà sẽ được sử dụng để ký thông điệp.



Hình 4. Cây Merkle với 8 nút lá.

Đường dẫn xác thực bao gồm các nút anh em (sibling node) của mỗi nút mà nằm trên đường dẫn từ khóa công khai WOTS⁺ được sử dụng đến nút gốc. Ví dụ, trong **Hình 4**, nếu thông điệp được ký với K_2 , thì K_2 , $n_{1,1}$ và $n_{2,0}$ là các nút trên đường dẫn nối đến nút gốc, và do vậy đường dẫn xác thực bao gồm K_3 , $n_{1,0}$ và $n_{2,1}$. Trong dòng 2 của **Thuật toán 9**, $\lfloor idx/2^j \rfloor$ là chỉ số của nút trên đường dẫn nối đến nút gốc, và $\lfloor idx/2^j \rfloor \oplus 1$ là chỉ số của nút nằm trên đường dẫn xác thực. Dòng 3 tính giá trị của nút nằm trên đường dẫn xác thực.

Thuật toán 9 $xmss_sign(M, \mathbf{SK}.seed, idx, \mathbf{PK}.seed, \mathbf{ADRS})$ - Sinh một chữ ký XMSS

Đầu vào: Thông điệp M có độ dài n -byte, mầm bí mật $\mathbf{SK}.seed$, chỉ số idx , mầm công khai $\mathbf{PK}.seed$, địa chỉ **ADRS**.

Đầu ra: chữ ký XMSS có dạng $\mathbf{SIG}_{XMSS} = (\mathbf{SIG}_{WOTS^+} \parallel \mathbf{AUTH}_{XMSS})$

1: **for** j **from** 0 **to** $h' - 1$ **do**

2: $k \leftarrow \lfloor idx/2^j \rfloor \oplus 1$

3: $\mathbf{AUTH}_{XMSS}[j] \leftarrow xmss_node(\mathbf{SK}.seed, k, j, \mathbf{PK}.seed, \mathbf{ADRS})$

4: **end for**

5: **ADRS**.setTypeAndClear(WOTS_HASH)

6: **ADRS**.setKeyPairAddress(idx)

7: $\mathbf{SIG}_{WOTS^+} \leftarrow wots_sign(M, \mathbf{SK}.seed, \mathbf{PK}.seed, \mathbf{ADRS})$

8: $\mathbf{SIG}_{XMSS} \leftarrow \mathbf{SIG}_{WOTS^+} \parallel \mathbf{AUTH}_{XMSS}$

9: **return** $\mathbf{SIG}_{XMSS}$

6.2.2.3. Tính khóa công khai XMSS từ một chữ ký

Xác minh một chữ ký XMSS bao gồm việc tính giá trị khóa công khai từ một thông điệp và một chữ ký số. Quá trình xác minh thành công nếu giá trị khóa công khai đúng được tính ra. Tính “đúng” được xác định bằng cách sử dụng giá trị khóa công khai nhận được cùng với thông tin khác để tính một giá trị $\mathbf{PK}.root$ ứng viên và sau đó so sánh giá trị đó với giá trị khóa công khai gốc từ khóa công khai HB-PQSignv1. Mục này mô tả hàm $xmss_pkFromSig$ (**Thuật toán 10**), mà dùng để tính một khóa công khai XMSS ứng viên từ một chữ ký XMSS và thông điệp tương ứng.

Ngoài thông điệp M có độ dài n -byte và một chữ ký $\mathbf{SIG}_{XMSS}$ có độ dài $((len + h') \cdot n)$ -byte, hàm $xmss_pkFromSig$ còn nhận các đầu vào khác là mầm $\mathbf{PK}.seed$ từ khóa công khai HB-PQSignv1, một địa

chỉ **ADRS**, và một chỉ số idx . Địa chỉ **ADRS** phải được gán cho địa chỉ tầng và địa chỉ cây của khóa XMSS đã được sử dụng để ký thông điệp, chỉ số idx phải là chỉ số của khóa WOTS⁺ bên trong cây XMSS mà đã được sử dụng để ký thông điệp.

Thuật toán 10 $xmss_pkFromSig(idx, \mathbf{SIG}_{XMSS}, M, \mathbf{PK.seed}, \mathbf{ADRS})$ – Tính khóa công khai XMSS từ thông điệp và một chữ ký XMSS

Đầu vào: Chỉ số idx , chữ ký XMSS $\mathbf{SIG}_{XMSS} = (\mathbf{SIG}_{WOTS^+} \parallel \mathbf{AUTH}_{XMSS})$, thông điệp M có độ dài n -byte, mầm công khai $\mathbf{PK.seed}$, địa chỉ **ADRS**.

Đầu ra: giá trị nút gốc $node[0]$ có độ dài n -byte.

```

1: ADRS.setTypeAndClear(WOTS_HASH)
2: ADRS.setKeyPairAddress( $idx$ )
3:  $\mathbf{SIG}_{WOTS^+} \leftarrow \mathbf{SIG}_{XMSS}.getWOTSSignature()$ 
4:  $\mathbf{AUTH}_{XMSS} \leftarrow \mathbf{SIG}_{XMSS}.getXMSSAUTH()$ 
5:  $node[0] \leftarrow wots\_pkFromSig(\mathbf{SIG}_{WOTS^+}, M, \mathbf{PK.seed}, \mathbf{ADRS})$ 
6: ADRS.setTypeAndClear(TREE)
7: ADRS.setTreeIndex( $idx$ )
8: for  $k$  from 0 to  $h' - 1$  do
9:   ADRS.setTreeHeight( $k + 1$ )
10:  if  $(\lfloor idx/2^k \rfloor == 0 \bmod 2)$  then
11:    ADRS.setTreeIndex(ADRS.getTreeIndex()/2)
12:     $node[1] \leftarrow \mathbf{H}(\mathbf{PK.seed}, \mathbf{ADRS}, node[0] \parallel \mathbf{AUTH}_{XMSS}[k])$ 
13:  else
14:    ADRS.setTreeIndex( $(\mathbf{ADRS}.getTreeIndex() - 1)/2$ )
15:     $node[1] \leftarrow \mathbf{H}(\mathbf{PK.seed}, \mathbf{ADRS}, \mathbf{AUTH}_{XMSS}[k] \parallel node[0])$ 
16:  end if
17:   $node[0] \leftarrow node[1]$ 
18: end for
19: return  $node[0]$ 

```

6.2.3. Siêu cây trong HB-PQSignv1

HB-PQSignv1 yêu cầu một lượng lớn khóa WOTS⁺ để ký các khóa công khai FORS. Tuy nhiên nếu chỉ có một khóa XMSS duy nhất cho tất cả khóa WOTS⁺ thì có thể ảnh hưởng đáng kể đến hiệu năng của **HB-PQSignv1** (đặc biệt là thuật toán sinh chữ ký), do vậy một siêu cây sẽ được sử dụng để ký các khóa FORS. Như được mô tả trong **Hình 1**, một siêu cây về bản chất là một cây gồm nhiều cây con có dạng XMSS. Các khóa XMSS ở tầng thấp nhất của siêu cây được sử dụng để ký các khóa công khai FORS (xem Mục 6.2.4), và các khóa XMSS ở các tầng khác sẽ được sử dụng để ký khóa công khai XMSS ở tầng ngay phía dưới.

Siêu cây có d tầng cây XMSS, trong đó mỗi cây XMSS đều là một cây băm Merkle có độ cao h' , do vậy tổng độ cao của siêu cây là $h = d \cdot h'$. Tầng trên cùng (tức tầng $d - 1$) chứa một cây XMSS duy nhất, và khóa công khai của cặp khóa XMSS này (nghĩa là giá trị băm gốc của cây Merkle) là khóa công khai của siêu cây (tức là

giá trị khóa công khai gốc **PK.root**). Tầng ngay phía dưới (tức tầng $d - 2$) có tất cả $2^{h'}$ cây XMSS và khóa công khai của từng cặp khóa XMSS này được ký bởi một trong $2^{h'}$ khóa WOTS⁺ mà là thành phần khóa XMSS ở tầng trên cùng. Tầng thấp nhất có $2^{(d-1) \cdot h'} = 2^{h-h'}$ cây XMSS, giúp cung cấp 2^h khóa WOTS⁺ để ký các khóa FORS.

6.2.3.1. Sinh chữ ký siêu cây

Một chữ ký siêu cây có độ dài $((h + d \cdot \text{len}) \cdot n)$ -byte và chứa một dãy gồm d chữ ký XMSS, mà bắt đầu với một chữ ký được sinh bằng cách sử dụng một khóa XMSS ở tầng thấp nhất và kết thúc với một chữ ký được sinh bằng cách sử dụng khóa XMSS ở tầng trên cùng (xem **Hình 5**).

Chữ ký XMSS SIG _{XMSS} (tầng 0)	$((h' + \text{len}) \cdot n)$ -byte
Chữ ký XMSS SIG _{XMSS} (tầng 1)	$((h' + \text{len}) \cdot n)$ -byte
...	
Chữ ký XMSS SIG _{XMSS} (tầng $d - 1$)	$((h' + \text{len}) \cdot n)$ -byte

Hình 5. Định dạng dữ liệu của một chữ ký siêu cây

Ngoài thông điệp M có độ dài n -byte,⁴ hàm `ht_sign` (Thuật toán 11) nhận đầu vào là các mầm **SK.seed** và **PK.seed** từ khóa riêng HB-PQSignv1, chỉ số của cây XMSS tại tầng thấp nhất mà sẽ ký thông điệp có chỉ số idx_{tree} , và chỉ số của khóa WOTS⁺ bên trong cây XMSS mà sẽ ký thông điệp có chỉ số idx_{leaf} .

Thuật toán 11 `ht_sign( $M, \text{SK.seed}, \text{PK.seed}, \text{idx}_{\text{tree}}, \text{idx}_{\text{leaf}}$ )` – Sinh một chữ ký siêu cây

Đầu vào: Thông điệp M , mầm bí mật **SK.seed**, mầm công khai **PK.seed**, chỉ số cây idx_{tree} , chỉ số lá idx_{leaf} .

Đầu ra: chữ ký siêu cây **SIG**_{HT}

1: **ADRS** $\leftarrow \text{toByte}(0,32)$

2: **ADRS.setTreeAddress**(idx_{tree})

3: **SIG**_{tmp} $\leftarrow \text{xmss_sign}(M, \text{SK.seed}, \text{idx}_{\text{leaf}}, \text{PK.seed}, \text{ADRS})$

4: **SIG**_{HT} $\leftarrow \text{SIG}_{\text{tmp}}$

5: **root** $\leftarrow \text{xmss_pkFromSig}(\text{idx}_{\text{leaf}}, \text{SIG}_{\text{tmp}}, M, \text{PK.seed}, \text{ADRS})$

6: **for** j **from** 1 **to** $d - 1$ **do**

7: $\text{idx}_{\text{leaf}} \leftarrow \text{idx}_{\text{tree}} \bmod 2^{h'}$

8: $\text{idx}_{\text{tree}} \leftarrow \text{idx}_{\text{tree}} \gg h'$

9: **ADRS.setLayerAddress**(j)

10: **ADRS.setTreeAddress**(idx_{tree})

11: **SIG**_{tmp} $\leftarrow \text{xmss_sign}(\text{root}, \text{SK.seed}, \text{idx}_{\text{leaf}}, \text{PK.seed}, \text{ADRS})$

12: **SIG**_{HT} $\leftarrow \text{SIG}_{\text{HT}} \parallel \text{SIG}_{\text{tmp}}$

13: **if** ($j < d - 1$) **then**

14: **root** $\leftarrow \text{xmss_pkFromSig}(\text{idx}_{\text{leaf}}, \text{SIG}_{\text{tmp}}, \text{root}, \text{PK.seed}, \text{ADRS})$

⁴ Trong HB-PQSignv1, thông điệp M mà được yêu cầu ký bởi `ht_sign` là khóa công khai FORS.

```

15: end if
16: end for
17: return  $\text{SIG}_{HT}$ 

```

6.2.3.2. Xác minh chữ ký siêu cây

Quá trình xác minh chữ ký siêu cây hoạt động bằng cách gọi d lần đến hàm `xmss_pkFromSig` (**Thuật toán 10**) và so sánh kết quả với khóa công khai của siêu cây. Ngoài thông điệp M có độ dài n -byte và chữ ký SIG_{HT} , hàm `ht_verify` (**Thuật toán 12**) nhận các đầu vào khác là mầm PK.seed và giá trị gốc PK.root từ khóa công khai HB-PQSignv1, chỉ số của cây XMSS tại tầng thấp nhất mà đã ký thông điệp có chỉ số idx_{tree} , và chỉ số của khóa WOTS^+ bên trong cây XMSS mà đã ký thông điệp có chỉ số idx_{leaf} .

Ở mỗi tầng, hoặc thông điệp M hoặc khóa công khai được tính ra của cặp khóa XMSS ở tầng dưới sẽ được cung cấp cùng với chữ ký XMSS phù hợp đến hàm `xmss_pkFromSig` để nhận được khóa công khai XMSS ở tầng đang xét. Nếu khóa công khai XMSS được tính ra của cây ở tầng trên cùng trùng với khóa công khai của siêu cây, tức giá trị PK.root , thì quá trình xác minh được xem là thành công.

Thuật toán 12 `ht_verify( $M, \text{SIG}_{HT}, \text{PK.seed}, \text{idx}_{tree}, \text{idx}_{leaf}, \text{PK.root}$ )` – Xác minh một chữ ký siêu cây

Đầu vào: Thông điệp M , chữ ký SIG_{HT} , mầm công khai PK.seed , chỉ số cây idx_{tree} , chỉ số lá idx_{leaf} , khóa công khai siêu cây PK.root

Đầu ra: giá trị bool biểu thị quá trình thành công hay thất bại

```

1:  $\text{ADRS} \leftarrow \text{toByte}(0,32)$ 
2:  $\text{ADRS.setTreeAddress}(\text{idx}_{tree})$ 
3:  $\text{SIG}_{tmp} \leftarrow \text{SIG}_{HT}.\text{getXMSSSignature}(0)$ 
4:  $node \leftarrow \text{xmss\_pkFromSig}(\text{idx}_{leaf}, \text{SIG}_{tmp}, M, \text{PK.seed}, \text{ADRS})$ 
5: for  $j$  from 1 to  $d - 1$  do
6:    $\text{idx}_{leaf} \leftarrow \text{idx}_{tree} \bmod 2^{h'}$ 
7:    $\text{idx}_{tree} \leftarrow \text{idx}_{tree} \gg h'$ 
8:    $\text{ADRS.setLayerAddress}(j)$ 
9:    $\text{ADRS.setTreeAddress}(\text{idx}_{tree})$ 
10:   $\text{SIG}_{tmp} \leftarrow \text{SIG}_{HT}.\text{getXMSSSignature}(j)$ 
11:   $node \leftarrow \text{xmss\_pkFromSig}(\text{idx}_{leaf}, \text{SIG}_{tmp}, node, \text{PK.seed}, \text{ADRS})$ 
12: end for
13: if ( $node = \text{PK.root}$ ) then
14:   return true
15: else
16:   return false
17: end if

```

6.2.4. Lược đồ chữ ký FORS

FORS là một lược đồ chữ ký một vài lần mà được sử dụng để ký lên giá trị băm của *thông điệp ký trong HB-PQSignv1*. Không giống WOTS⁺, một khóa FORS có thể được sử dụng nhiều hơn một lần để tạo chữ ký, tuy nhiên độ an toàn của nó sẽ dần bị suy giảm khi số lượng chữ ký được tạo ra tăng lên.

FORS sử dụng hai tham số: k và $t = 2^a$. Khóa riêng FORS bao gồm k tập, mà mỗi tập chứa t chuỗi n -byte. Tất cả những chuỗi này đều được sinh giả ngẫu nhiên từ mầm **SK**. seed. Mỗi tập trong số k tập kể trên sẽ được tạo thành một cây Merkle, và các nút gốc của các cây sẽ được băm cùng nhau để hình thành khóa công khai FORS. Một chữ ký trên $(k \cdot a)$ -bit giá trị băm thông điệp sẽ bao gồm k phần tử từ khóa riêng, **trong đó mỗi phần tử** sẽ được chọn từ một tập riêng biệt dựa trên các a -bit của giá trị băm thông điệp, cùng với các đường dẫn xác thực từ mỗi phần tử đến gốc của cây Merkle tương ứng (xem **Hình 6**).

Giá trị khóa riêng (cây thứ 0)	n -byte
$AUTH_{FORS}^0$ (cây thứ 0)	$(a \cdot n)$ -byte
...	
Giá trị khóa riêng (cây thứ $k - 1$)	n -byte
$AUTH_{FORS}^{k-1}$ (cây thứ $k - 1$)	$(a \cdot n)$ -byte

Hình 6. Định dạng dữ liệu chữ ký FORS

6.2.4.1. Sinh các giá trị bí mật FORS

Hàm `fors_skGen` (**Thuật toán 13**) sinh ra các chuỗi n -byte của các khóa riêng FORS. Hàm này nhận đầu vào là **SK**. seed, **PK**. seed từ khóa riêng HB-PQSignv1, một địa chỉ **ADRS**, và một chỉ số idx . Địa chỉ **ADRS** phải có kiểu `FORS_TREE`, và địa chỉ cây và địa chỉ cặp khóa phải được gán tương ứng với chỉ số của khóa WOTS⁺ bên trong cây XMSS mà ký khóa FORS. Địa chỉ tầng phải được gán bằng 0. Chỉ số idx là chỉ số của giá trị bí mật FORS bên trong các tập cây FORS.

Thuật toán 13 `fors_skGen(SK. seed, PK. seed, ADRS, idx)` – Sinh một giá trị khóa riêng FORS

Đầu vào: Mầm bí mật **SK**. seed, mầm công khai **PK**. seed, địa chỉ **ADRS**, chỉ số khóa bí mật idx .

Đầu ra: giá trị khóa riêng FORS có độ dài n -byte.

1: `skADRS` ← **ADRS**

2: `skADRS.setTypeAndClear(FORS_PRF)`

3: `skADRS.setKeyPairAddress(ADRS.getKeyPairAddress())`

4: `skADRS.setTreeIndex(idx)`

5: **return** `PRF(PK. seed, SK. seed, skADRS)`

6.2.4.2. Tính nút FORS

Hàm `fors_node` (**Thuật toán 14**) tính các nút của một cây Merkle. Thuật toán này là giống như hàm `xmss_node`, với khác biệt là các nút lá là các giá trị băm của các giá trị bí mật FORS thay vì các khóa công khai WOTS⁺.

Hàm `fors_node` nhận đầu vào là **SK**. seed và **PK**. seed từ khóa riêng HB-PQSignv1; một chỉ số nút mục tiêu i ,

mà là chỉ số của nút đang được tính; một độ cao nút mục tiêu z , mà là độ cao bên trong cây Merkle của nút đang được tính; và một địa chỉ **ADRS**. Địa chỉ **ADRS** **phải có địa** chỉ tầng được gán bằng 0 (vì cây XMSS mà ký một khóa FORS luôn nằm ở tầng 0), địa chỉ cây được gán tương ứng với cây XMSS mà ký khóa FORS, kiểu địa chỉ phải được đặt là FORS_TREE, và địa chỉ cặp khóa được gán bằng chỉ số của khóa WOTS⁺ bên trong cây XMSS mà ký khóa FORS. Chỉ số và độ cao nút mục tiêu phải thỏa mãn $z \leq a$ và $i < k \cdot 2^{(a-z)}$.

Thuật toán 14 `fors_node(SK.seed, i, z, PK.seed, ADRS)` – *Tính một nút của cây con Merkle trong FORS*

Đầu vào: mầm bí mật **SK.seed**, chỉ số nút mục tiêu i , độ cao nút mục tiêu z , mầm công khai **PK.seed**, địa chỉ **ADRS**.

Đầu ra: chuỗi *node* có độ dài n -byte là giá trị nút thứ i tại độ cao z của cây con Merkle.

```

1: if  $z = 0$  then
2:    $sk \leftarrow \text{fors\_skGen}(\text{SK.seed}, \text{PK.seed}, \text{ADRS}, i)$ 
3:   ADRS.setTreeHeight(0)
4:   ADRS.setTreeIndex( $i$ )
5:    $node \leftarrow \mathbf{F}(\text{PK.seed}, \text{ADRS}, sk)$ 
6: else
7:    $lnode \leftarrow \text{fors\_node}(\text{SK.seed}, 2i, z - 1, \text{PK.seed}, \text{ADRS})$ 
8:    $rnode \leftarrow \text{fors\_node}(\text{SK.seed}, 2i + 1, z - 1, \text{PK.seed}, \text{ADRS})$ 
9:   ADRS.setTreeHeight( $z$ )
10:  ADRS.setTreeIndex( $i$ )
11:   $node \leftarrow \mathbf{H}(\text{PK.seed}, \text{ADRS}, lnode \parallel rnode)$ 
12: end if
13: return  $node$ 

```

6.2.4.3. Sinh một chữ ký FORS

Hàm `fors_sign` (**Thuật toán 15**) ký một giá trị băm thông điệp md có độ dài $(k \cdot a)$ -bit.⁵ Ngoài giá trị băm thông điệp, hàm `fors_sign` còn nhận đầu vào khác là **SK.seed** và **PK.seed** từ khóa riêng HB-PQSignv1 và một địa chỉ **ADRS**. Địa chỉ này phải có địa chỉ tầng bằng 0 (vì cây XMSS mà ký một khóa FORS luôn nằm ở tầng 0), địa chỉ cây được gán tương ứng với cây XMSS mà ký khóa FORS, kiểu địa chỉ phải được đặt là FORS_TREE, và địa chỉ cặp khóa được gán bằng chỉ số của khóa WOTS⁺ bên trong cây XMSS mà ký khóa FORS.

Hàm `fors_sign` phân tách $(k \cdot a)$ -bit của md thành k chuỗi có độ dài a -bit (dòng 2), mỗi chuỗi trong số đó biểu diễn một số nguyên trong khoảng 0 đến $t - 1$. Mỗi số nguyên này được sử dụng để chọn một giá trị bí mật từ một trong k tập (hay k cây FORS) trong dòng 4. Với mỗi giá trị bí mật được lựa chọn, một đường dẫn xác thực được tính và thêm vào chữ ký (**các dòng 5 đến 9**).

Thuật toán 15 `fors_sign(md, SK.seed, PK.seed, ADRS)` – *Sinh một chữ ký FORS*

Đầu vào: Giá trị băm thông điệp md , mầm bí mật **SK.seed**, mầm công khai **PK.seed**, địa chỉ **ADRS**.

⁵ **Để thuận tiện**, hàm `fors_sign` nhận đầu vào là một giá trị băm thông điệp có độ dài $\left\lceil \frac{k \cdot a}{8} \right\rceil$ -byte và trích xuất $(k \cdot a)$ -bit đầu tiên để ký.

Đầu ra: Chữ ký FORS SIG_{FORS}

```

1:  $\text{SIG}_{\text{FORS}} \leftarrow \text{NULL}$ 
2:  $\text{ind} \leftarrow \text{base\_2}^b(\text{md}, a, k)$ 
3: for  $i$  from 0 to  $k - 1$  do
4:    $\text{SIG}_{\text{FORS}} \leftarrow \text{SIG}_{\text{FORS}} \parallel \text{fors\_skGen}(\text{SK.seed}, \text{PK.seed}, \text{ADRS}, i \cdot 2^a + \text{ind}[i])$ 
5:   for  $j$  from 0 to  $a - 1$  do
6:      $s \leftarrow \lfloor \text{ind}[i] / 2^j \rfloor \oplus 1$ 
7:      $\text{AUTH}_{\text{FORS}}^i[j] \leftarrow \text{fors\_node}(\text{SK.seed}, i \cdot 2^{a-j} + s, j, \text{PK.seed}, \text{ADRS})$ 
8:   end for
9:    $\text{SIG}_{\text{FORS}} \leftarrow \text{SIG}_{\text{FORS}} \parallel \text{AUTH}_{\text{FORS}}^i$ 
10: end for
11: return  $\text{SIG}_{\text{FORS}}$ 

```

6.2.4.4. Tính một khóa công khai FORS từ một chữ ký

Việc xác minh một chữ ký FORS bao gồm quá trình tính một giá trị khóa công khai từ giá trị băm thông điệp và chữ ký. Quá trình xác minh thành công nếu giá trị khóa công khai đúng được tính ra. Tính “đúng” được xác định bằng cách xác minh chữ ký siêu cây trên giá trị khóa công khai được tính ra thông qua khóa công khai HB-PQSignv1. Mục này mô tả hàm `fors_pkFromSig` (**Thuật toán 16**) dùng để tính một khóa công khai FORS ứng viên từ chữ ký FORS và giá trị băm thông điệp tương ứng.

Ngoài giá trị băm thông điệp md và một chữ ký SIG_{FORS} có độ dài $(k \cdot (a + 1) \cdot n)$ -byte, hàm `fors_pkFromSig` còn nhận các đầu vào khác là PK.seed từ khóa công khai HB-PQSignv1 và một địa chỉ ADRS . Địa chỉ này phải có địa chỉ tầng bằng 0 (vì cây XMSS mà ký một khóa FORS luôn nằm ở tầng 0), địa chỉ cây được gán tương ứng với cây XMSS mà ký khóa FORS, kiểu địa chỉ phải được đặt là `FORS_TREE`, và địa chỉ cặp khóa được gán bằng chỉ số của khóa WOTS^+ bên trong cây XMSS mà ký khóa FORS.

Thuật toán 16 `fors_pkFromSig( $\text{SIG}_{\text{FORS}}$ ,  $\text{md}$ ,  $\text{PK.seed}$ ,  $\text{ADRS}$ )` – Tính một khóa công khai FORS từ thông điệp và một chữ ký FORS

Đầu vào: chữ ký FORS SIG_{FORS} , giá trị băm thông điệp md , mầm công khai PK.seed , địa chỉ ADRS .

Đầu ra: khóa công khai FORS pk_{FORS} .

```

1:  $\text{ind} \leftarrow \text{base\_2}^b(\text{md}, a, k)$ 
2: for  $i$  from 0 to  $k - 1$  do
3:    $\text{sk} \leftarrow \text{SIG}_{\text{FORS}}.\text{getSK}(i)$ 
4:    $\text{ADRS}.\text{setTreeHeight}(0)$ 
5:    $\text{ADRS}.\text{setTreeIndex}(i \cdot 2^a + \text{ind}[i])$ 
6:    $\text{node}[0] \leftarrow \text{F}(\text{PK.seed}, \text{ADRS}, \text{sk})$ 
7:    $\text{AUTH}_{\text{FORS}}^i \leftarrow \text{SIG}_{\text{FORS}}.\text{getAUTH}(i)$ 
8:   for  $j$  from 0 to  $a - 1$  do
9:      $\text{ADRS}.\text{setTreeHeight}(j + 1)$ 
10:    if  $(\lfloor \text{ind}[i] / 2^j \rfloor \bmod 2 == 0)$  then

```

```

11:      ADRS.setTreeIndex(ADRS.getTreeIndex()/2)
12:       $node[1] \leftarrow H(\mathbf{PK}.seed, \mathbf{ADRS}, node[0] \parallel AUTH_{FORS}^i[j])$ 
13:  else
14:      ADRS.setTreeIndex((ADRS.getTreeIndex() - 1)/2)
15:       $node[1] \leftarrow H(\mathbf{PK}.seed, \mathbf{ADRS}, AUTH_{FORS}^i[j] \parallel node[0])$ 
16:  end if
17:       $node[0] \leftarrow node[1]$ 
18:  end for
19:   $root[i] \leftarrow node[0]$ 
20: end for
21: forspkADRS  $\leftarrow$  ADRS
22: forspkADRS.setTypeAndClear(FORS_ROOTS)
23: forspkADRS.setKeyPairAddress(ADRS.getKeyPairAddress())
24:  $pk_{FORS} \leftarrow T_k(\mathbf{PK}.seed, \text{forSpkADRS}, root)$ 
25: return  $pk_{FORS}$ .

```

6.3. Các thuật toán chữ ký số HB-PQSignv1

HB-PQSignv1 được hình thành từ sự kết hợp giữa siêu cây và các khóa FORS. Khóa riêng HB-PQSignv1 chứa một giá trị mầm bí mật và một khóa PRF bí mật. Khóa công khai bao gồm một mầm công khai $\mathbf{PK}.seed$ và giá trị nút gốc của siêu cây. Một chữ ký HB-PQSignv1 được tạo bằng cách băm thông điệp, sau đó sử dụng một phần giá trị băm nhận được để lựa chọn khóa FORS, phần còn lại của giá trị băm thông điệp ký với khóa FORS được lựa chọn, và tiếp theo sinh một chữ ký siêu cây cho khóa (công khai) FORS được lựa chọn. Các tham số cho HB-PQSignv1 được quy định tuân theo các thành phần WOTS⁺, XMSS, siêu cây, và FORS (xem Mục 8).

HB-PQSignv1 sử dụng một tham số bổ sung m , mà là kích thước theo byte của giá trị băm thông điệp. Tham số này được tính theo công thức:

$$m = \left\lceil \frac{h - h'}{8} \right\rceil + \left\lceil \frac{h'}{8} \right\rceil + \left\lceil \frac{k \cdot a}{8} \right\rceil$$

HB-PQSignv1 sử dụng h -bit (thuộc $\left\lceil \frac{h-h'}{8} \right\rceil + \left\lceil \frac{h'}{8} \right\rceil$ -byte cuối) của giá trị băm thông điệp để lựa chọn một khóa FORS. Trong đó $(h - h')$ -bit để lựa chọn một cây XMSS tại tầng thấp nhất, h' -bit để lựa chọn một khóa WOTS⁺ và khóa FORS tương ứng từ cây XMSS đó. Phần được ký bởi khóa FORS được chọn chính là $(k \cdot a)$ -bit (thuộc $\left\lceil \frac{k \cdot a}{8} \right\rceil$ -byte đầu tiên bên trái) của giá trị băm thông điệp. Do **chỉ có $h + k \cdot a$ bit** được sử dụng, việc triển khai được đơn giản hóa bằng cách trích xuất các bit cần thiết từ một giá trị băm có kích thước lớn hơn đôi chút.

Mục này mô tả các thuật toán cho việc sinh khóa, sinh chữ ký và xác minh chữ ký trong HB-PQSignv1.

6.3.1. Thuật toán sinh khóa HB-PQSignv1

Các khóa công khai HB-PQSignv1 bao gồm hai thành phần (xem **Hình 7**). Đầu tiên là một mầm công khai $\mathbf{PK}.seed$ có kích thước n -byte, mà được sử dụng trong các lần gọi hàm băm để cung cấp sự phân tách miền

giữa các cặp khóa HB-PQSignv1 khác nhau. Giá trị thứ hai là khóa công khai siêu cây **PK.root** (nghĩa là giá trị nút gốc của cây XMSS ở tầng cao nhất) có độ dài n -byte. **PK.seed** phải được sinh bằng thuật toán sinh bit ngẫu nhiên được chuẩn hóa, mà ở đó phiên bản cụ thể của bộ sinh số giả ngẫu nhiên hỗ trợ mức độ an toàn tối thiểu $8n$ -bit.

PK.seed	n -byte
PK.root	n -byte

Hình 7. Khóa công khai HB-PQSignv1

SK.seed	n -byte
SK.prf	n -byte
PK.seed	n -byte
PK.root	n -byte

Hình 8. Khóa riêng HB-PQSignv1

Khóa riêng HB-PQSignv1 bao gồm hai giá trị bí mật, ngẫu nhiên có độ dài n -byte (xem **Hình 8**). **SK.seed** được sử dụng để sinh tất cả các thành phần khóa riêng FORS và WOTS⁺. **SK.prf** được sử dụng để sinh một giá trị giúp ngẫu nhiên hóa phép băm thông điệp trong HB-PQSignv1. Khóa riêng cũng bao gồm một bản sao khóa công khai. Cả **SK.seed** và **SK.prf** đều phải được sinh bằng thuật toán sinh bit ngẫu nhiên được chuẩn hóa, mà ở đó phiên bản cụ thể của bộ sinh số giả ngẫu nhiên hỗ trợ mức độ an toàn tối thiểu $8n$ -bit. Quá trình sinh cặp khóa HB-PQSignv1 được thể hiện thông qua **Thuật toán 17**.

Thuật toán 17 hbpqs_keygen() – Sinh một cặp khóa HB-PQSignv1

Đầu vào: (không có đầu vào)

Đầu ra: cặp khóa HB-PQSignv1 (SK, PK)

```

1: SK.seed  $\xleftarrow{\$}$   $\mathbb{B}^n$ 
2: SK.prf  $\xleftarrow{\$}$   $\mathbb{B}^n$ 
3: PK.seed  $\xleftarrow{\$}$   $\mathbb{B}^n$ 
4: if ((SK.seed == NULL)  $\vee$  (SK.prf == NULL)  $\vee$  (PK.seed == NULL)) then
5:   return  $\perp$ 
6: end if
7: ADRS  $\leftarrow$  toByte(0,32)
8: ADRS.setLayerAddress( $d - 1$ )
9: PK.root  $\leftarrow$  xmss_node(SK.seed, 0,  $h'$ , PK.seed, ADRS)
10: return cặp khóa ((SK.seed, SK.prf, PK.seed, PK.root), (PK.seed, PK.root))

```

6.3.2. Thuật toán sinh chữ ký HB-PQSignv1

Một chữ ký HB-PQSignv1 bao gồm một chuỗi ngẫu nhiên hóa, một chữ ký FORS, và một chữ ký siêu cây, như được chỉ ra trong **Hình 9**.

Giá trị ngẫu nhiên R	n -byte
Chữ ký FORS SIG_{FORS}	$(k \cdot (1 + a) \cdot n)$ -byte
Chữ ký siêu cây SIG_{HT}	$((h + d \cdot len) \cdot n)$ -byte

Hình 9. Định dạng chữ ký HB-PQSignv1

Việc sinh một chữ ký HB-PQSignv1 được thực hiện thông qua hàm `hbpqs_sign` (**Thuật toán 18**) dưới đây. Đầu vào của hàm này là thông điệp M , khóa riêng SK và một giá trị ngẫu nhiên bổ sung $addrnd$. Trong đó, $addrnd$ phải là một giá trị ngẫu nhiên n -byte và phải được sinh một cách lý tưởng bởi một bộ sinh bit ngẫu nhiên được chuẩn hóa.

Thuật toán 18 `hbpqs_sign(M, SK, addrnd)` – Sinh một chữ ký HB-PQSignv1

Đầu vào: Thông điệp M , khóa riêng $SK = (SK.seed, SK.prf, PK.seed, PK.root)$ và một giá trị ngẫu nhiên bổ sung $addrnd$.

Đầu ra: Chữ ký HB-PQSignv1 SIG .

- 1: $ADRS \leftarrow \text{toByte}(0, 32)$
- 2: $opt_rand \leftarrow addrnd$
- 3: $R \leftarrow \text{PRF}_{msg}(SK.prf, opt_rand, M)$
- 4: $SIG \leftarrow R$
- 5: $digest \leftarrow H_{msg}(R, PK.seed, PK.root, M)$
- 6: $md \leftarrow digest \left[0 : \left\lceil \frac{k \cdot a}{8} \right\rceil \right]$
- 7: $tmp_idx_{tree} \leftarrow digest \left[\left\lceil \frac{k \cdot a}{8} \right\rceil : \left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil \right]$
- 8: $tmp_idx_{leaf} \leftarrow digest \left[\left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil : \left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil + \left\lceil \frac{h}{8d} \right\rceil \right]$
- 9: $idx_{tree} \leftarrow \text{toInt}(tmp_idx_{tree}, \left\lceil \frac{h-h/d}{8} \right\rceil) \bmod 2^{h-h/d}$
- 10: $idx_{leaf} \leftarrow \text{toInt}(tmp_idx_{leaf}, \left\lceil \frac{h}{8d} \right\rceil) \bmod 2^{h/d}$
- 11: $ADRS.setTreeAddress(idx_{tree})$
- 12: $ADRS.setTypeAndClear(FORS_TREE)$
- 13: $ADRS.setKeyPairAddress(idx_{leaf})$
- 14: $SIG_{FORS} \leftarrow \text{fors_sign}(md, SK.seed, PK.seed, ADRS)$
- 15: $SIG \leftarrow SIG \parallel SIG_{FORS}$
- 16: $pk_{FORS} \leftarrow \text{fors_pkFromSig}(SIG_{FORS}, md, PK.seed, ADRS)$
- 17: $SIG_{HT} \leftarrow \text{ht_sign}(pk_{FORS}, SK.seed, PK.seed, idx_{tree}, idx_{leaf})$
- 18: $SIG \leftarrow SIG \parallel SIG_{HT}$

19: **return** SIG

6.3.3. Thuật toán xác minh chữ ký HB-PQSignv1

Giống như thuật toán sinh chữ ký, thuật toán xác minh chữ ký HB-PQSignv1 (**Thuật toán 19**) bắt đầu bằng cách tính một giá trị băm thông điệp (dòng 8); và sau đó trích xuất thành phần được ký md (dòng 9), chỉ số idx_{tree} (các dòng 10 và 12), và idx_{leaf} (các dòng 11 và 13) từ giá trị băm. Một khóa công khai FORS ứng viên sau đó được tính (dòng 17), và chữ ký trên khóa FORS được xác minh (dòng 18). Nếu quá trình xác minh chữ ký này thành công, thì khóa công khai FORS chính xác đã được tính, và chữ ký SIG trên thông điệp M là hợp lệ.

Thuật toán 19 hbpqs_verify(M , SIG, PK) – Xác minh một chữ ký HB-PQSignv1

Đầu vào: Thông điệp M , chữ ký SIG, khóa công khai PK = (PK.seed, PK.root)

Đầu ra: giá trị bool biểu thị quá trình thành công hay thất bại

```

1: if |SIG|  $\neq (1 + k \cdot (1 + a) + h + d \cdot len) \cdot n$  then
2:   return false
3: end if
4: ADRS  $\leftarrow$  toByte(0,32)
5: R  $\leftarrow$  SIG.getR()
6: SIGFORS  $\leftarrow$  SIG.getFORSSignature()
7: SIGHT  $\leftarrow$  SIG.getHTSignature()
8: digest  $\leftarrow$  Hmsg(R, PK.seed, PK.root, M)
9: md  $\leftarrow$  digest  $\left[0: \left\lceil \frac{k \cdot a}{8} \right\rceil \right]$ 
10: tmp_idxtree  $\leftarrow$  digest  $\left[ \left\lceil \frac{k \cdot a}{8} \right\rceil : \left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil \right]$ 
11: tmp_idxleaf  $\leftarrow$  digest  $\left[ \left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil : \left\lceil \frac{k \cdot a}{8} \right\rceil + \left\lceil \frac{h-h/d}{8} \right\rceil + \left\lceil \frac{h}{8d} \right\rceil \right]$ 
12: idxtree  $\leftarrow$  toInt(tmp_idxtree,  $\left\lceil \frac{h-h/d}{8} \right\rceil$ ) mod  $2^{h-h/d}$ 
13: idxleaf  $\leftarrow$  toInt(tmp_idxleaf,  $\left\lceil \frac{h}{8d} \right\rceil$ ) mod  $2^{h/d}$ 
14: ADRS.setTreeAddress(idxtree)
15: ADRS.setTypeAndClear(FORS_TREE)
16: ADRS.setKeyPairAddress(idxleaf)
17: pkFORS  $\leftarrow$  fors_pkFromSig(SIGFORS, md, PK.seed, ADRS)
18: return ht_verify(pkFORS, SIGHT, PK.seed, idxtree, idxleaf, PK.root)

```

7. Khuyến cáo về các trường hợp sử dụng

Mục này trình bày các trường hợp sử dụng đối với **thuật toán chữ ký số HB-PQSignv1** mà có thể được triển khai trong môi trường thực tế. Trong đó, các thuật toán sinh chữ ký và xác minh chữ ký ứng với từng trường hợp sử dụng sẽ được đặc tả dựa trên thuật toán sinh chữ ký và xác minh chữ ký của HB-PQSignv1 (mà đã được đặc tả trong Mục 6.3).

Trong phạm vi của tiêu chuẩn này, có 03 trường hợp sử dụng của **thuật toán chữ ký số HB-PQSignv1** được khuyến cáo sử dụng:

- HB-PQSignv1 dạng “nguyên bản”, nhìn chung, đây là phiên bản sẽ thường được sử dụng trong các tình huống thực tế. Ở dạng này, HB-PQSignv1 sẽ được sử dụng để ký và xác minh trực tiếp lên nội dung của thông điệp cần ký.
- HB-PQSignv1 dạng “băm trước”, là một cách sử dụng khác của HB-PQSignv1. Ở dạng này, HB-PQSignv1 sẽ được sử dụng để ký và xác minh trực tiếp lên giá trị băm nội dung của thông điệp cần ký. **Cách sử dụng này phù hợp để triển khai HB-PQSignv1 trong một số mô-đun mật mã có bộ nhớ bị hạn chế.**
- HB-PQSignv1 dạng “ký bó”, là cách sử dụng HB-PQSignv1 giúp cung cấp chữ ký cho nhiều thông điệp cùng lúc với chi phí tính toán tương đương với một lần ký thông thường. Dạng sử dụng này của HB-PQSignv1 phù hợp để triển khai trong các yêu cầu hiệu năng ký cao.

Trong cả ba dạng kể trên, các cặp khóa được sinh ra tuân thủ đúng theo thuật toán sinh khóa của HB-PQSignv1. Điểm khác biệt của những trường hợp sử dụng này chính là ở đặc tả của các thuật toán sinh chữ ký và xác minh chữ ký. Trong các Mục 7.1, 7.2 và 7.3 dưới đây, tiêu chuẩn này sẽ trình bày các thuật toán sinh chữ ký và xác minh chữ ký ứng với từng trường hợp sử dụng HB-PQSignv1 theo dạng nguyên bản, băm trước và ký bó. Lưu ý rằng, trong các thuật toán đó có sử dụng một chuỗi ngữ cảnh ctx (với độ dài tối đa 255-byte), mà do từng ứng dụng cụ thể quy định. (Theo mặc định chuỗi ctx là rỗng).

7.1. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng nguyên bản

7.1.1. Sinh chữ ký HB-PQSignv1 dạng nguyên bản

Trong phiên bản nguyên bản, nội dung dữ liệu ký được nối thêm vào **đăng trước bởi 1-byte** biểu diễn chuỗi phân tách miền sử dụng, 1-byte biểu diễn độ dài của chuỗi ctx , và chuỗi ctx . Chuỗi phân tách miền sử dụng (mang giá trị 0 đối với thuật toán sinh chữ ký nguyên bản) được thêm vào để tránh các chữ ký theo các dạng sử dụng khác được xác minh một cách hợp lệ giống như các chữ ký nguyên bản và ngược lại. Trong trường hợp mặc định mà ở đó chuỗi ctx là rỗng, hàm ký nguyên bản đơn giản thực hiện thêm hai byte 0 vào trước nội dung cần ký, sau đó truyền kết quả đó cho hàm `hbpqs_sign` cùng với khóa bí mật, và một giá trị ngẫu nhiên $addrnd$ có độ dài n -byte.

Thuật toán 20 $\text{pure_hbpqs_sign}(M, ctx, SK)$ – Sinh một chữ ký HB-PQSignv1 dạng nguyên bản
Đầu vào: Thông điệp M, chuỗi ctx, khóa riêng SK. Đầu ra: chữ ký HB-PQSignv1 dạng nguyên bản SIG. <pre> 1: if $ctx > 255$ then 2: return $\perp$ 3: end if 4: $addrnd \xleftarrow{\\$} \mathbb{B}^n$ 5: if $(addrnd == \text{NULL})$ then 6: return $\perp$ 7: end if 8: $M' \leftarrow \text{toByte}(0,1) \parallel \text{toByte}(ctx , 1) \parallel ctx \parallel M$ </pre>

```

9: SIG  $\leftarrow$  hbpqs_sign( $M'$ , SK,  $addrnd$ )
10: return SIG

```

7.1.2. Xác minh chữ ký HB-PQSignv1 dạng nguyên bản

Thuật toán 21 đặc tả hàm xác minh chữ ký HB-PQSignv1 theo dạng nguyên bản (pure_hbpqs_verify). Thuật toán này xây dựng M' theo cách tương tự như **Thuật toán 20**, và chuyển thông điệp M' nhận được cho hàm xác minh hbpqs_verify **theo HB-PQSignv1** để đưa ra kết quả xác minh.

Thuật toán 21 pure_hbpqs_verify(M , SIG, ctx , PK) – Xác minh một chữ ký HB-PQSignv1 dạng nguyên bản

Đầu vào: Thông điệp M , chữ ký SIG, chuỗi ctx , khóa công khai PK.

Đầu ra: giá trị bool biểu thị quá trình thành công hay thất bại.

```

1: if  $|ctx| > 255$  then
2:   return false
3: end if
4:  $M' \leftarrow \text{toByte}(0,1) \parallel \text{toByte}(|ctx|, 1) \parallel ctx \parallel M$ 
5: return hbpqs_verify( $M'$ , SIG, PK)

```

7.2. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng băm trước

7.2.1. Sinh chữ ký HB-PQSignv1 dạng băm trước

Theo dạng băm trước, thông điệp đầu vào cho hàm sinh chữ ký hbpqs_sign của HB-PQSignv1 là kết quả của việc áp dụng hàm băm lên nội dung dữ liệu ký, sau đó được nối thêm vào đằng trước bởi 1-byte biểu diễn chuỗi phân tách miền sử dụng, 1-byte biểu diễn độ dài của chuỗi ctx , chuỗi ctx , và một định danh chữ ký sid. Chuỗi phân tách miền sử dụng mang giá trị 1 đối với thuật toán sinh chữ **ký dạng băm trước** HB-PQSignv1 dạng băm trước, và định danh chữ ký sid là một giá trị được sinh ngẫu nhiên.

Thuật toán 22 hash_hbpqs_sign(M , ctx , SK) – Sinh một chữ ký HB-PQSignv1 dạng băm trước

Đầu vào: Thông điệp M , chuỗi ctx , khóa riêng SK.

Đầu ra: Chữ ký HB-PQSignv1 dạng băm trước SIG

```

1: if  $|ctx| > 255$  then
2:   Return  $\perp$ 
3: end if
4:  $addrnd \xleftarrow{\$} \mathbb{B}^n$ 
5: if ( $addrnd == \text{NULL}$ ) then
6:   return  $\perp$ 
7: end if
8:  $sid \xleftarrow{\$} \mathbb{B}^{2n}$ 
9:  $h_M \leftarrow \text{PH}(sid, \text{toByte}(0,1), M)$ 
10:  $M' \leftarrow \text{toByte}(1,1) \parallel \text{toByte}(|ctx|, 1) \parallel ctx \parallel sid \parallel h_M$ 
11:  $\text{SIG}_{\text{hbpqs}} \leftarrow \text{hbpqs\_sign}(M', SK, addrnd)$ 

```

12: **return** $SIG = (sid, SIG_{hbpqs})$

7.2.2. Xác minh chữ ký HB-PQSignv1 dạng băm trước

Thuật toán 23 đặc tả hàm xác minh chữ ký HB-PQSignv1 theo dạng băm trước (hash_hbpqs_verify). Thuật toán này xây dựng M' theo cách tương tự như **Thuật toán 22**, và chuyển thông điệp M' nhận được cho hàm xác minh hbpqs_verify **theo HB-PQSignv1** để đưa ra kết quả xác minh.

Thuật toán 23 hash_hbpqs_verify(M, SIG, ctx, PK) – Xác minh một chữ ký HB-PQSignv1 dạng băm trước

Đầu vào: Thông điệp M , chữ ký $SIG = (sid, SIG_{hbpqs})$, chuỗi ctx , khóa công khai PK .

Đầu ra: giá trị bool biểu thị quá trình thành công hay thất bại.

```

1: if  $|ctx| > 255$  then
2:   return false
3: end if
4:  $h_M \leftarrow PH(sid, toByte(0,1), M)$ 
5:  $M' \leftarrow toByte(1,1) \parallel toByte(|ctx|, 1) \parallel ctx \parallel sid \parallel h_M$ 
6: return hbpqs_verify( $M', SIG_{hbpqs}, PK$ )

```

7.3. Sử dụng thuật toán chữ ký HB-PQSignv1 dạng ký bó

7.3.1. Sinh chữ ký HB-PQSignv1 dạng ký bó

Thuật toán 24 đặc tả hàm sinh chữ ký HB-PQSignv1 dạng ký bó (batch_hbpqs_sign). Ở đó, một danh sách thông điệp cần ký, chuỗi ctx và khóa riêng SK sẽ được sử dụng như đầu vào của thuật toán này. Các thông điệp đầu vào sẽ được sử dụng để tạo một cây băm Merkle, giá trị gốc của cây này được nối thêm vào đằng trước bởi 1-byte biểu diễn chuỗi phân tách miền sử dụng, 1-byte biểu diễn độ dài của chuỗi ctx , chuỗi ctx , một định danh chữ ký sid ; và được thêm vào đằng sau bởi 2-byte biểu diễn số lượng thông điệp trong mỗi lần ký bó. Kết quả này được đưa vào hàm sinh chữ ký hbpqs_sign như thông điệp đầu vào. Lưu ý rằng, chuỗi phân tách miền sử dụng mang giá trị 2 đối với **thuật toán sinh chữ k HB-PQSignv1 dạng ký bó; định danh chữ ký sid là một giá trị được sinh ngẫu nhiên; và giá trị N phải là lũy thừa của 2 thỏa mãn điều kiện $1 \leq N \leq 2^8$.**

Thuật toán 24 batch_hbpqs_sign($M_0, M_1, \dots, M_{N-1}, ctx, SK$) – Sinh một chữ ký HB-PQSignv1 dạng ký bó

Đầu vào: Danh sách thông điệp $M_0, M_1, \dots, M_{N-1}$, chuỗi ctx , khóa riêng SK .

Đầu ra: Danh sách chữ ký HB-PQSignv1 dạng ký bó ($BSIG_{M_0}, BSIG_{M_1}, \dots, BSIG_{M_{N-1}}$).

```

1: if  $|ctx| > 255$  then
2:   return  $\perp$ 
3: end if
4:  $addrnd \xleftarrow{\$} \mathbb{B}^n$ 
5: if ( $addrnd == \text{NULL}$ ) then
6:   return  $\perp$ 

```

```

7: end if
8:  $\text{sid} \xleftarrow{\$} \mathbb{B}^{2n}$ 
9: for  $l$  from 0 to  $N - 1$  do
10:    $\text{Tree}[0, l] \leftarrow \text{PH}(\text{sid}, \text{toByte}(0, 1) \parallel \text{toByte}(l, 1), M_l)$ 
11: end for
12:  $\text{high} \leftarrow \log_2(N)$ 
13: for  $i$  from 0 to  $\text{high} - 1$  do
14:   for  $j$  from 0 to  $2^{\text{high}-i-1} - 1$  do
15:      $\text{lnode} \leftarrow \text{Tree}[i, 2j]$ 
16:      $\text{rnode} \leftarrow \text{Tree}[i, 2j + 1]$ 
17:      $\text{Tree}[i + 1, j] \leftarrow \text{PH}(\text{sid}, \text{toByte}(i + 1, 1) \parallel \text{toByte}(j, 1), \text{lnode} \parallel \text{rnode})$ 
18:   end for
19: end for
20:  $\text{Root}_{\text{Batch}} \leftarrow \text{Tree}[\text{high}, 0]$ 
21:  $M' \leftarrow \text{toByte}(2, 1) \parallel \text{toByte}(|\text{ctx}|, 1) \parallel \text{ctx} \parallel \text{sid} \parallel \text{Root}_{\text{Batch}} \parallel \text{toByte}(N, 2)$ 
22:  $\text{SIG}_{\text{hbpqs}} \leftarrow \text{hbpqs\_sign}(M', \text{SK}, \text{addrnd})$ 
23: for  $l$  from 0 to  $N - 1$  do
24:    $\text{AUTH} \leftarrow []$ 
25:   for  $i$  from 0 to  $\text{high} - 1$  do
26:      $j \leftarrow \lfloor l/2^i \rfloor$ 
27:     if  $(j == 0 \bmod 2)$  then
28:        $\text{AUTH}[i] \leftarrow \text{Tree}[i, j + 1]$ 
29:     else
30:        $\text{AUTH}[i] \leftarrow \text{Tree}[i, j - 1]$ 
31:     end if
32:   end for
33:  $\text{BSIG}_{M_l} \leftarrow (\text{sid}, N, l, \text{SIG}_{\text{hbpqs}}, \text{AUTH})$ 
34: end for
35: return  $(\text{BSIG}_{M_0}, \text{BSIG}_{M_1}, \dots, \text{BSIG}_{M_{N-1}})$ 

```

7.3.2. Xác minh chữ ký HB-PQSignv1 dạng ký bó

Thuật toán 25 đặc tả hàm xác minh chữ ký HB-PQSignv1 theo dạng ký bó (batch_hbpqs_verify). Thuật toán này xây dựng M' bằng cách khôi phục giá trị băm gốc từ thông điệp kiểm tra và thành phần đường dẫn xác thực của chữ ký, và sau đó được ghép nối theo **cách tương tự như Thuật toán 24**. Sau đó, thông điệp M' được chuyển cho hàm xác minh hbpqs_verify **theo HB-PQSignv1** để đưa ra kết quả xác minh.

Thuật toán 25 $\text{batch_hbpqs_verify}(M, \text{BSIG}, \text{ctx}, \text{PK})$ – Xác minh một chữ ký HB-PQSignv1 dạng ký bó

Đầu vào: Thông điệp M , chữ ký $\text{BSIG} = (\text{sid}, N, l, \text{SIG}_{\text{hbpqs}}, \text{AUTH})$, chuỗi ctx , khóa công khai PK .

Đầu ra: giá trị bool biểu thị quá trình thành công hay thất bại.

```

1: if  $|\text{ctx}| > 255$  then
2:   return false
3: end if
4:  $\text{node} \leftarrow \text{PH}(\text{sid}, \text{toByte}(0, 1) \parallel \text{toByte}(l, 1), M)$ 
5:  $\text{high} \leftarrow \log_2(N)$ 
6: for  $i$  from 0 to  $\text{high} - 1$  do
7:    $j \leftarrow \lfloor l/2^i \rfloor$ 
8:   if  $(j == 0 \bmod 2)$  then
9:      $\text{node} \leftarrow \text{PH}(\text{sid}, \text{toByte}(i + 1, 1) \parallel \text{toByte}(\lfloor j/2 \rfloor, 1), \text{node} \parallel \text{AUTH}[i])$ 
10:  else
11:     $\text{node} \leftarrow \text{PH}(\text{sid}, \text{toByte}(i + 1, 1) \parallel \text{toByte}(\lfloor j/2 \rfloor, 1), \text{AUTH}[i] \parallel \text{node})$ 
12:  end if
13: end for
14:  $M' \leftarrow \text{toByte}(2, 1) \parallel \text{toByte}(|\text{ctx}|, 1) \parallel \text{ctx} \parallel \text{sid} \parallel \text{node} \parallel \text{toByte}(N, 2)$ 
15: return  $\text{hbpqs\_verify}(M', \text{SIG}_{\text{hbpqs}}, \text{PK})$ 

```

8. Quy định về tham số và thuật toán mật mã

Mục này sẽ quy định bộ tham số và thuật toán mật mã được triển khai trong **thuật toán chữ ký số HB-PQSignv1**.

8.1. Quy định bộ tham số mật mã

Trong bảng dưới đây, tiêu chuẩn đặc tả các bộ 8-tham số HB-PQSignv1 với các thành phần $(n, h, d, h', a, k, w, m)$ được sử dụng trong **thuật toán chữ ký số HB-PQSignv1** theo các mức an toàn tối thiểu 128-bit cổ điển (tương ứng 64-bit lượng tử) và 192-bit cổ điển (tương ứng 96-bit lượng tử). Các bộ tham số này được đặt tên **theo quy tắc sau**: một tiền tố “HB” được đặt ở đầu, tiếp theo là mức an toàn cổ điển tối thiểu, **$\log_2$ của số lần ký tối đa** và một **chữ cái** thể hiện mức độ ưu tiên (cụ thể ‘f’ thể hiện bộ tham số ưu tiên tốc độ sinh chữ ký, ‘s’ thể hiện bộ tham số ưu tiên kích thước chữ ký nhỏ và ‘b’ thể hiện bộ tham số cân bằng giữa hai yếu tố trên). Ngoài ra, các ký hiệu bitsec, qbitsec và q_s tương ứng thể hiện mức an toàn cổ điển, mức an toàn lượng tử và số lượng truy vấn ký tối đa đối với từng bộ tham số.

	n	h	d	h'	a	k	w	m	q_s	bitsec	qbitsec
HB-128-20f	24	22	11	2	6	56	16	46	2^{20}	166	80
HB-128-20s	24	24	3	8	11	20	16	31	2^{20}	167	81
HB-128-20b	24	24	6	4	7	37	16	37	2^{20}	166	80
HB-128-48f	24	50	25	2	6	56	16	49	2^{48}	138	66
HB-128-48s	24	48	6	8	13	18	16	36	2^{48}	139	67

HB-128-48b	24	52	13	4	8	30	16	37	2^{48}	139	67
HB-128-54f	24	56	28	2	9	27	16	39	2^{54}	133	64
HB-128-54s	24	56	7	8	13	17	16	35	2^{54}	133	64
HB-128-54b	24	56	14	4	9	27	16	39	2^{54}	133	64
HB-192-20f	32	22	11	2	7	60	16	57	2^{20}	230	112
HB-192-20s	32	24	3	8	11	27	16	41	2^{20}	231	113
HB-192-20b	32	20	4	5	8	53	16	56	2^{20}	230	112
HB-192-48f	32	50	25	2	7	60	16	60	2^{48}	202	98
HB-192-48s	32	48	6	8	12	28	16	48	2^{48}	203	99
HB-192-48b	32	48	12	4	8	53	16	60	2^{48}	202	98
HB-192-54f	32	57	19	3	10	32	16	48	2^{54}	197	96
HB-192-54s	32	56	7	8	14	21	16	44	2^{54}	197	96
HB-192-54b	32	56	14	4	11	29	16	48	2^{54}	197	96

8.2. Quy định thuật toán mật mã

Trong mục này, tiêu chuẩn đưa ra quy định về các bộ thuật toán mật mã được sử dụng. Sau đó, tiêu chuẩn cũng đặc tả chi tiết cách thiết lập đối với các hàm băm và giả ngẫu nhiên cho HB-PQSignv1.

8.2.1. Bộ thuật toán mật mã

Mục này liệt kê các tham số và thuật toán mật mã sử dụng trong tiêu chuẩn này cho thuật toán chữ ký HB-PQSignv1.

Thuật toán	Chú giải
Thuật toán hàm băm	
SHA3-256	Hàm băm cấu trúc Sponge có độ dài đầu ra 256-bit theo tiêu chuẩn TCVN 11816-3:2017 Công nghệ thông tin – Các kỹ thuật an toàn – Hàm băm – Phần 3: Hàm băm chuyên dụng.
SHA3-512	Hàm băm cấu trúc Sponge có độ dài đầu ra 512-bit theo tiêu chuẩn TCVN 11816-3:2017 Công nghệ thông tin – Các kỹ thuật an toàn – Hàm băm – Phần 3: Hàm băm chuyên dụng.
Bộ sinh số ngẫu nhiên/giả ngẫu nhiên	
HMAC_DRBG	Bộ sinh số ngẫu nhiên/giả ngẫu nhiên HMAC_DRBG theo tiêu chuẩn TCVN 12853:2020 Công nghệ thông tin - Các kỹ thuật an toàn – Bộ tạo bit ngẫu nhiên.
HDRBG	Bộ sinh số ngẫu nhiên/giả ngẫu nhiên HDRBG theo tiêu chuẩn TCVN 12853:2020 Công nghệ thông tin - Các kỹ thuật an toàn – Bộ tạo bit ngẫu nhiên.

8.2.2. Triển khai cho các hàm băm và giả ngẫu nhiên

Đối với các hàm được sử dụng chung cho HB-PQSignv1, tiêu chuẩn quy định triển khai các hàm này theo thiết lập sau:

$$H_{msg}(R, \mathbf{PK}.seed, \mathbf{PK}.root, M) = \text{Trunc}_m(\text{SHA3-512}(R \parallel \mathbf{PK}.seed \parallel \mathbf{PK}.root \parallel M))$$

$$\text{PRF}(\mathbf{PK}.seed, \mathbf{SK}.seed, \mathbf{ADRS}) = \text{Trunc}_n(\text{SHA3-256}(\mathbf{SK}.seed \parallel \mathbf{PK}.seed \parallel \mathbf{ADRS}))$$

$$\mathbf{PRF}_{msg}(\mathbf{SK.prf}, opt_rand, M) = \text{Trunc}_n(\text{SHA3-256}(\mathbf{SK.prf} \parallel opt_rand \parallel M))$$

$$\mathbf{F}(\mathbf{PK.seed}, \mathbf{ADRS}, M_1) = \text{Trunc}_n(\text{SHA3-256}(\mathbf{PK.seed} \parallel \mathbf{ADRS} \parallel M_1))$$

$$\mathbf{H}(\mathbf{PK.seed}, \mathbf{ADRS}, M_2) = \text{Trunc}_n(\text{SHA3-256}(\mathbf{PK.seed} \parallel \mathbf{ADRS} \parallel M_2))$$

$$\mathbf{T}_\ell(\mathbf{PK.seed}, \mathbf{ADRS}, M_\ell) = \text{Trunc}_n(\text{SHA3-256}(\mathbf{PK.seed} \parallel \mathbf{ADRS} \parallel M_\ell))$$

Ngoài ra, đối với các dạng băm trước và ký bó, hàm **PH** (với các đầu vào là định danh chữ ký *sid*, thành phần điều chỉnh *TW* và thông điệp đầu vào *M**) trong các **Thuật toán 22**, **Thuật toán 23**, **Thuật toán 24**, **Thuật toán 25** được thiết lập bởi hàm băm SHA3-512 như sau:

$$\mathbf{PH}(\text{sid}, \text{TW}, M^*) = \text{SHA3-512}(\text{sid} \parallel \text{TW} \parallel M^*)$$

Lưu ý: Trên đây là triển khai của các hàm $\mathbf{PRF}_{msg}$, $\mathbf{H}_{msg}$, **PRF**, $\mathbf{T}_\ell$, **H**, **F** và **PH** theo thể hiện của họ hàm SHA3.

9. Một số yêu cầu bổ sung

Dưới đây là một số yêu cầu bổ sung để ngăn chặn một số nguy cơ tấn công khi triển khai **thuật toán chữ ký số HB-PQSignv1**.

Yêu cầu 1: Các khóa riêng của mỗi người dùng cần được lưu trữ trong các thiết bị bảo mật an toàn.

Yêu cầu 2: Các dữ liệu được sử dụng nội bộ bởi thuật toán sinh khóa và thuật toán sinh chữ ký trong các bước tính toán trung gian cần được xóa một cách an toàn để tránh kẻ tấn công có thể thu thập thông tin về khóa riêng và **từ đó làm** suy yếu độ an toàn của **thuật toán chữ ký số HB-PQSignv1**.

Yêu cầu 3: Các phép lấy ngẫu nhiên được sử dụng trong các thuật toán trong tiêu chuẩn này phải được thực hiện bằng bộ sinh số giả ngẫu nhiên/giả ngẫu nhiên (được quy định trong Mục 8.2.1) và phải có mức an toàn tối thiểu bằng với độ dài đầu ra của chuỗi cần sinh.

Yêu cầu 4: Đối với các ứng dụng đồng thời triển khai các dạng khác nhau của **thuật toán chữ ký số HB-PQSignv1**, các cặp khóa cho mỗi dạng sử dụng phải phân biệt.

Yêu cầu 5: Việc triển khai HB-PQSignv1 trên các thiết bị chuyên dụng cần đảm bảo kháng được các tấn công gây lỗi và tấn công kênh kề.

Phụ lục A
(quy định)
Vector kiểm tra cho HB-PQSignv1

Phần phụ lục này trình bày kết quả vector kiểm tra đối với quá trình ký HB-PQSignv1 bởi hàm hbpqs_sign. Do vector kiểm tra có kích thước lớn, vì vậy chỉ các bộ tham số HB-128-48f, HB-192-20f được trình bày chi tiết; đối với các bộ tham số còn lại, vector kiểm tra được trình bày dưới dạng tóm tắt.

A.1. Bộ tham số HB-128-20f

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-20f	24	22	11	2	6	56	16	46	96	48	23424

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	DFA06CFEE0671D492E3D9C39FC7480231EA900DDD91D0A35
SK.prf	E414D4E8EAC463004E7B8D26EE54005C807BC833DCA4DDFA
PK.seed	326CE873302DA3651D5AE46F041C092DF4C318E3F7E99ED4
Tính PK.root	
PK.root	6193F2B8594D91DC91543AFD39D8EA06192D102C2BEFCC26

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	82E538F57AD742C5AC1A18CA6329A2D39832658A4C149960BA41B3D16AD95CEA1B2CBA1C2C2 249EA83D3F1DA2AB9796E674F803B4218F5A087AD51C488528A9C74ACEC9C0FFD5D5AA45D0D 71D7BF9A3F1EDC566AE4612EF1557F3B11A9768DFAF04BAE7CCBFF7AB65E6FCE2DD45FBA37A 5880AFDCAEFD64C42CFC748F1880C87
addrnd	1E6518D9BE91965799A0104572CDE5DFEE0788BDDA2BF8D7
Chữ ký (23424 bytes)	5DEE959E17722C21103AD2EC3542FFC7561ED40F99088592 553006A3CE7C44FD280B7413422CB04020EACD599861960A 65E6AD2CB3A2AF24D568E6310439D87A5AA81ECAADE77535 ... 994653E0C3030F7FB597FC5311CD5CF7AC4940A8D6BFBE43 2113886229C6C80525156766CAB50BC05224C21AFEEDCB7C

A.2. Bộ tham số HB-128-20s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-20s	24	24	3	8	11	20	16	31	96	48	10032

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	CE5853AED55C9E47CD8FE13F1881508D24B0989A9FD7D892
SK.prf	C169D71B2D3E36F77EE8875BE8293FC2B8C951956DE948C1
PK.seed	8BF3ABE5898D65C5FC83D4CBF7AFA484EECAD3B4C7266A70
Tính PK.root	
PK.root	6CD5E4161EEE3329AD7AB79A5B0A92504B95312ADAD47726

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	E43E269AF58BA9973CC65A4369B7CB8DB70E06BBDFA5E51F3A7930278D198C1A757854769D 67EA9771BCD2653B941F03E57075EDF1CA3F0E2F888CEEF5C80DE9FA36470C225B849399E74
---------------------------	---

	CF827E2973CC25222897F45DAA143D5F19300D5D5FBE7E0FE840927CE10FE3C685364CB02DA6965F541767B4CC85D04955AED6FA62
addrnd	6F947C80900E780C88779136D1DA92178DAD5EB2FF25C15F
Chữ ký (10032 bytes)	71FD73EE5A4D3D8AB1B48191F3259F4A6155D3B507CC6503 1D830C15474D4F067DA303609EA218D63C37352B9F17F55F 6B707E1EDC0D769B442E477ED7A52A5AC99034A03EEDA9AC ... 082E459F8B42D9A4CAFEC2E01E5EF2C903105365A8D45D2E 17072EE6E5D2524AA59FCBFAC57E5117D48CA7B4AF43440E

A.3. Bộ tham số HB-128-20b

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-20b	24	24	6	4	7	37	16	37	96	48	15048

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	1E96F85A7D951BB22886C82DB0D17FA362C16CED529C0E1F
SK.prf	1AB2F29DCE58A179AB00C0AA0F1DE758E563803CEC0E6064
PK.seed	7E3FC9525A28528037CBD51908FB5D91099F2870E66A1E8B
Tính PK.root	
PK.root	0296A5D947B10DE529322DC6616E03BD697C6798CD947C63

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	77F23871275214DCBFECDA CF7BA98BBD1E8EF30FF0A4E1B508A767436984647453185546ADB567494F419D5D43B408C226371DA64CB52E67CD1E71EF49D26EE4B0C3843F21C02EA616EA233CDA94452A793BE08C75854D63DAEF0A9E509DB0C017B63FA549A1659A5EFE93B34A47FC610C82B2BB94082FD7BAB1304C25F8E999
addrnd	82FD1E24ACBEBE5600B295D0702C01256B57BA2F0658DA87
Chữ ký (15048 bytes)	9E9F4A90408F2435E1CCDA51826ABA0F24823484134F1530 0AECDD5D66D3CD33AFBE57B4D067B23A3A8E95B2F64FD312E 60473D3EE2CF3CC995B82CB6E1EB21DD33F5F9CEA1E7FB5A ... 524FE1E48E242747FF79068BF4EF40F6E3B86C44781CF1AB 0AF11F8BB5FB575634B222371FAD100BFD12013AD41C8540

A.4. Bộ tham số HB-128-48f

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-48f	24	50	25	2	6	56	16	49	96	48	41232

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	A1B479F91FEF14F45B0067F5BC10761CF16AAC4C2B13970A
SK.prf	F05EA18CCBEE3E074E141A0D0DDA2382B49150BA437257DA
PK.seed	4BE43CCD55C0EECA68AA528917826B52C80435D52C10D94C
Tính PK.root	
PK.root	205D8361B9FB368C81597D1BAF70A1839F2CE447BF952BBC

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	F44921A6021BA79070089150C85E481AC46644DB00490CAC59D0A33D70A5DC45E19E75A7081D2C4CCB41281A9E5CC7E8179D18788A6FFF7735A3592A34BE5623D2132D0FD0DF2967B0E7EC
---------------------------	--

	C1867C43C1FF39D80DCDED3C51E8BE93E359C05E516CA825F469E080C5030B9CAF9AD2C16908C2084437D1EA088647D7D68BAD3A6F
addrnd	8B0F757B8C281ACF8550DC898A8CA04902CA656A30E6F630

Quá trình ký:

Giai đoạn băm	
opt_rand = addrnd	8B0F757B8C281ACF8550DC898A8CA04902CA656A30E6F630
R	AACCA394DE1854CA1343CB9D34BB1FF1B5B96D7D67BBCC48
md	3E63C966595113B8583FCDC88F074175AA1DE1FFD42B576C815C08B41B888BDB55B40B10D6C8955C6FC8
idx_{tree}	64CC3A1687DE
idx_{leaf}	2
Sinh chữ ký FORS	
indices[1]	F
secret ₁	A0856C47DC36D4A3929CF6CF1B62FF7E6CBE7AD6B29E8C26
AUTH _{FORS} ¹ [1]	7F181B13ADB77188C7E62AEDCD97914ABBA57B3336EB8590
AUTH _{FORS} ¹ [2]	7072A0D7966E0F3A58ED7F87A1C745A3B752B4DE4A5A8052
AUTH _{FORS} ¹ [3]	BCDB481E8BFF31D553062DDAC0FD82B2908A24DB87C9C658
AUTH _{FORS} ¹ [4]	BA9C4FA34123E9312ABB01EBC8332A16A4449094B1174B89
AUTH _{FORS} ¹ [5]	3BA1251D0586234356CBC4D8D6AFDC13E5C274C53228288B
AUTH _{FORS} ¹ [6]	73A56F837A1D9FED95A4F4C78EA6D90DD0EA9935180C4434
root ₁	59251979EEFE5AA8705980C4D3784F39889614F659BA26BD
indices[2]	26
secret ₂	71C1FD1529F18709FB83A256A0E810EF8DEBA6F87FF651F7
AUTH _{FORS} ² [1]	2A6A21A11517F04A31328888B568476940E299587CC34FBA
AUTH _{FORS} ² [2]	CD83EECB9D7486F69F13F5F7FF34F42C13A6FC975D4B3A44
AUTH _{FORS} ² [3]	F9788B6B8F91FE90C6C1977DDAAC3E2EEFBF5D246AA24D94
AUTH _{FORS} ² [4]	BD65A8748A033FFEC77C754A07D34CBA5E579A2A91FDDA04
AUTH _{FORS} ² [5]	F3E0F2565E381F5B404BBB1649DE58390B9C618E7BCCC6EB
AUTH _{FORS} ² [6]	0CBEF25CCB619E8F68AA0BEFC6BBA0A7548E95024904A517
root ₂	F2129057EC51648F3D7137C731BB1B5316BFBF654783FE2F
...	
indices[56]	8
secret ₅₆	D1AA38029031A3F8E66BAB2D6052CDEEAEEA1218165B4AF0
AUTH _{FORS} ⁵⁶ [1]	900F9896529B61A61D51D21E92D87D94EE40B4E47686A6EF
AUTH _{FORS} ⁵⁶ [2]	1910342BE78A6F74505DB2B5524146284E3FDF8D59AE9E35
AUTH _{FORS} ⁵⁶ [3]	81D91F002016F5203E56B6058A00C4EE7DD9AF82C1CB4637
AUTH _{FORS} ⁵⁶ [4]	197688B28C653DBD2D812B658DB61476426648A0DB077009
AUTH _{FORS} ⁵⁶ [5]	1174927E9407E780270BE6A69020E58F3BEB0F9A0D14CE1D
AUTH _{FORS} ⁵⁶ [6]	B25B8BB2DF8AF30E0E837D7D0B6ABB5DB01B327CF797DC3A
root ₅₆	58AD1EC03DC2626EA8EA6F18D6E3B3502D69C62305DD566C
Khóa công khai FORS	
pk_{FORS}	705841D6163C0C19BB0AD39254878B17EA9E1A7BF067C949
Sinh chữ ký Siêu cây:	
Tầng 0	
$SIG_{WOTS^+}^0$	04B8477413EFE6DDF76407961E1519B1CC7016058AD7FC07 2365FCC8BB3DA8C9B8B2F4C5C53DC673C01C3BE53AD7BFCE F69E2A8125F5383A723D5F4C865CB1E87E715552073A597D 25B6464EEB432ED1C874635E7A74508CA323EF654AF6CA8D 48964CDA23CEE7B0B7CAF1F4801A721F9276EB3A0F24FF53 0A70F90EEF51662AE5F17298960A7AA37103FADFEEAFC290 A5E72D1787E04E9F933464CF76428EBDA4D6CFDDB93C892 E04C2F3C1FD3ED8F9D15E2C08C1C06234787D1EB1CA5F9DC

	97E64B30D1FC75B8367FCECB5268C3CB365F138352A02994 E14925F9E10E5FAD2CA4AA20B336B8A0E3E4827586B63189 2B26F83713C0FAB62823EBE20EB0307AE8ED685B89D8AAAD 890EEFA6B9881C9074278AA5449C78B64CFFC9D1AFE8C6B1 A8867ABE26DD17980F968599D2FFB603A02F029D4DD5BD25 64214BDEC9D2C99DB91AAB28D8CD837FC1B1324FE6813F83 9FAD8BD41D8599C5346F1DB65A96DA508D12D30775F2A22F E2DACAF230C8C091EBF876FF600F1FF86407B58596837C7B 6B4A1350592F97E48FA0FF354A67FA6048A232D288A6CD73 41D14E441217FCD3307EBF928A667FADA63B397E24C02643 1505D6B0ABCCFA94F73A2C12EA363D126A8D6905A21DF3E5 832A4EF62D6B144C65F1DF8DF5F0F11765E5ECD6EAD42697 0E27D2BFC559E727D550EBBD5D9E85EBAFEF4BFB0F46B52A DED7353CB88D0A86FB75E6C24A4577AF0115F6E92D055934 87D945F5A0DBDA0AB47601925AED263CF2A4FE8C354ECFF7 E8F331E0BAE45FDE4A0C4B0D2A4BFA5746D249DA030747E3 7A5545F3AD903135637001D2F2B72D6EA6576E67425CDCEA 8309036E67579772A0B8EF0431E9BE82E5D49FB80B9BA136 F70374DEBCC8868511161353FF1C0B41DEE9177701B20E79 5A42AE6069DC9A7A877EEF86437392BC90A26A2745164094 6F33FDFB53EF5F9351A2C7E07C748BDB7003263FC92D4C8F E75115021C8D310EB71797F4E622714DBFC385036765376F 90ED94FAA531232B19AE99F6586F3534873560092A19D781 1BF96BAFF2DBDE97F05A83826277E528DD14C6D0201CF1C2 ABA392F91ED44E4E032698321639D501EAD04A0986A32708 CD661C3AF5025BEC30DA848F3B7E536D0F4459CD9FC2AD4E 3AC18EC56D366340A60198D6DD60C43CB8BBA01B4B4042DB 4C5E5914BD59A9639D0C3CC41A9798C5F2125F97D41BA052 8F8D8D2EBA212784A66C18322432D83FB2D6D0767BF2F7BA E70CE9B20192676F793FDF77F363B3669202CF3F8B1D734B 100D58D67E88053B79A0F9DEB4DD76EA86FA4EACF303B41E A5619829F6903C34E6F452FDE66FD8CC1641AE924AF939A5 CD6CB81B5FE0B9CC1928467FB19946042452DC31BED02D76 661463CE6D01F8608CCAB88CDE7A20A539BAAAF9CC5DFD04 DEF1CC66A9AE2177D212635147EC581DEFCE478090BE8743 9E0F2C4C77A124D8561D31F695085E950AFDBDAFCD57864F E40A1D0E09E9E0F5D2C034C391A059DC726289235226B272 220561363BA831B1174EC5DFE87A9E6613A6E1D2B33C3665 04D50D5B36DD391C37F24D312C50615BCF50451F10D34704 FA01191BB7358E37F74F50A27C1F5E90E8E19B4098F749A1 6FC0990D05FEBD85FF69F4F960333689D127342B65C376B0 FBA64CC71B05FEF97AFE366541ACE32BA5EF1708D50ED992 BDE266D46D22813EE994192553B8A018FA24C0C71FD7B2AF
$pk_{\text{WOTS}^+}^0$	F170017912EEAA2954C5B4B02379C93051AABF2EB30FB68E
$\text{AUTH}_{\text{XMSS}}^0[1]$	3F851C5A2E26C5847BEE52C9107A21D834E7954C014D9F64
$\text{AUTH}_{\text{XMSS}}^0[2]$	5B6C6F33B97E31A1450856C531079FFBFA1525CBDB649EF4
root_0	B5582DC457E80DD22858C1300863D99AB86CB7161150D2C5
Tầng 1	
$\text{SIG}_{\text{WOTS}^+}^1$	FAC69E74B332953210647FAAD959046B6951B68A14CF930F 893648E93DE7DB7EA9F2FBF959680C68D694F000661B9A28 5323098E3BDE7759288BC9AC9780FC3B0917FF4C50B1B116 B4276D5E57F730837FC99674BBB075751C50434A090EF429 31F44A0E7099ACCC33E1D975B81C1B2DD4A7D73278B016A9 45B702C344C6A1E7CFDB344C2A4977A99974A10BD9D3E2E6 139D05B0FDCCAB65224688A09EEB5DEC4F3F36164BDA8152 BCAC711E65E9003C05B91FE957D47F62E8F68BAB35B8E4FF B68BA697A9F49531B0F52D79B39E313DE43479636CBFE506 7A88B25B113138DC36D4BEEB1445F62B40657B3D67A8DD07 32B6B8995593D0F3542C859D6D70316AE3C58D37788AFADF 015F0A51BC938AEC18B8AE7880A58A05E3DB9B7CE4FF8C0

	4FB92AA31438957E80C80C1B49B04F655210D97634FC1D85 D43AC656261EC4DEF373060C5631B9E13E90B767CC7AA476 3E8F5E9C1BEA1E729FFFC1097153D90943736CEC9F4181B3 24F796B28A05313BBE6285FA501EA3AB88636B9F6B62128F B4F370F3FA229EE161ACD2C2F573935DD68D03B281C986FA C422ACDFB374AB7DA00A0B4B3CB126535A2CE8C1E5FAF675 91C7CBE36304D9B54B286777BD44452738C80D674BAD4556 B4F34D6165ABB6AEA01F3EEE4070AC027B95180DDDF2154 CCF066C8A7E3D655C0D9A01EE61071D74EA45F365D7E1F41 8B2AA56D870A70251FA312BAAB77F2FD8126BC0AEDFE18CE C308C3E66643DD2666149307EDF9F8DBAE918A784F875850 70A4C22441E7EEF84AA9FEE783E536897B178F8620F0254E DAF2F220CFC93051303ADE5CEA9FC1806D0559D1FEA90963 018F96F8210F96CA303C690DB39EDEB1FF4A50AB770D0675 50B1FD050520BBFE73907CE6C91AD38823A25D25CDA388BF 57DFBBA36124121F97F1EE622BF08AC02B0A26FD37F6900 F39A948B2A57CD6BB390AC15899D82AE9F09B10194EF9486 0127C3F63D9242DD2C011BDF190F7142066BE097E90483FA EE964C27BD2CACF44FAADD5A581902667B966F089184E006 E4FED4A8BC3A17C2324E31B5C27B649EA1024B142F024F22 EA10B17B2E4AE8AD4AF8938A5C2062895DAE150994AD7014 9DC37B5AD3C5183C22A47594FC55BADC111F1F938E3B5CF7 E0A7F500FA0E3C77CB6F9C490B9F97599B88FDFACAAC7B15 6E4FF330C652C6A59033AC9ABAB3BCC58B9C6C0E7BD35674 4559E1F935E42FD8B1A6A7D378CE63C45EAB69ADF75D8E8A 90ED316A6C6B843AE3D9763EC386AB5DD0FD564BB145A869 B606BD9E8B734BC1BBDBE7EB38B7A087C2628F80E158C85C 6FEEDB9444DAE755CF6565CEDD13D17BE9698AE68A661B5F 457A7B25BB2EDA9A0809729528972BB665806590D43170FB CA13802D3B8A33C148382F16AED10C58909B411976257439 3ADA6AA551410412EF4E3F0AF67F15C688AB15BF2F270494 3BBF3E8AF53D4A8750600274D40754CEAE3B6666DF27285 7904E5226260F29956E146DB898DCF9C034BB0B47F1BCF0B DC475694CDF9E99BB66A025CAB065D586BCFE74A796E4A0B 8886041ABC16202C9C4DDA4112D4A6F34843297BF7821DEE B413A6329D0E7CB6CAD2134F7420867AEFB608A3CF8C6B77 1254E629DE5DAC8C692E3DE6BDD12091027B318C865F8745 22431A3A751FADDA95F76BA8C8FEBDBEDA47138891CC7DC8 D96F4A04739C826D71A5068C26DF3430E6681305837C6606
$pk_{WOTS^+}^1$	FBF2F5C39B8BDC22B867A0564D52981F8BA47A5B23D6AD4E
$AUTH_{XMSS}^1[1]$	17FF4FF1C763AEFCE2830A05C69B5EDE53C3CDDCD3C25AAD
$AUTH_{XMSS}^1[2]$	CCFB57DBBBF70DC4360FC566C368F8C70B1DD4E0EF141142
$root_1$	E02FEA7F7B1A3159B9650C290E0A95D549983BE4026ECF88
...	
Tầng 24	
$SIG_{WOTS^+}^{24}$	CE555C1849BE0A591F85F6852F68769C049B2EE36B4AB1D3 A7BE75E7178C6F1F93C63757578FF44C0D9BB21C51D0019E B4C0681A2FA25C7305F00A808535BD3243436C4E2359C6F2 D5B70CAD25882D04D7B55F1B9457BB9E4C355C7B8367AE8C 21B3954080F8FA368A39E85B78F59D4950B406D8B6555BD1 092BA65A9F55B7FB97C0D4C19B26D9CD5E9C19A467B609CE C09AFBB10950FE07199A04C4DBB86ED0DFA4C01280F6F702 EB1D96A792039FBFC7188D539C66C3B9E1FBE6376D9BB233 D7B4A6DD65289870938BA7A1EF20CDB6F0687F3877E3D56F 53B760857B7732CC00D246A076F6E6D67193C1A004834870 21A2AA640DE9563EA394D8479EC99A7CEBEAD8B5220CF178 32A794C8D5C36AAA9BD578FEFBD5F483803B6C586D9674B5 15B90420FA9E261B66F60773254DF6E28A960EE22B320AF2 E28E65AFB90EA57896B8552472DF221F72225C7751C143AA 6CFC3701B1770E47A878B43E9CF3043DFE93D4B5420EC0D3

	C3BB939F236E30B13DF7EC8EC7694CCEC8B79CCD785954B3 8D794DC35F84155D05166879A7C2C4D3666AC0DBF98E56DD 8D98C72D309E68C1162C1221A92F7D060E5B1BD634229FDF C3360AE40F57E3A3129EA559B60605B28B7E176D4C0A6398 8D3C43EAD81F3A40EF592C5BBA79BCA905A3A3F933395E4F ECA89612E268C0A3D6AFB9D05C835B5BCB3BEB235D91A7AD 4677F38610D1DD4E31E60DBB6452545A606E432F6CCE7C21 A308AD5BD5E156AB966768A146894D40C4265A4FB34B9983 6BA8DE68F2CAC5F9332CF8BA92D406CBB34BFC5218B2B4BF A83524F83A9274DB82B7C7D07C64EC0337231BA691CB41C4 C876DA7AFA4EDDAB76AB1CE88BE9E8891CB79C03631D778A 476BBEEEF07A1740A89CD38AF51FFD7E1205C76A32E8D38A 225F1F2665010A4D31E3191883FE5AACD810DBA9264D1B9C C59CDA303577715E19DC31FB92D3A1186333DEE15ACB59C9 9A773B906F6463F4A52E273026A6AF6591D1BCADEA487C35 063E78E8D921685EAB7C1E07E5AA042CED8ACD18FE28D07E 1E6600222E270A0DFB7174E7CDD61CC7FCD9184E9EC02F0 6407186724B80D9BC2A7F3B0F41E60390135CEE486E221D8 9AD07D93BC2BDE922D64F5204FAA95FBC5BAAF0DBB474BCF 82004B0193683CF3FD41F5F86DE6A749C04DFBB188B0030B AD9D83AF54BC5B62C854F21227B29C251B9F3466ECC1B965 595B851872A9612FB16F29E8E692ED61D95D433DF0D34CB1 0B343559D51CCF88392349DC229B0CA2722147E6B6C3F035 A4E66167CE1B7633238C5833FBA2B420E4104BEAE8CA2174 7B54E60899C4A2DFAE4C52E7C51ABA6A1435FA97B84A3F1C 1F846D8DB004F1EB2A0568F39B4919FC12F647D50E720A4D 3FDB8358D57BA0DC788AAAF6D6E11ED5297219BF1E6117E6 36BB4BB3F07499F76DE535A232351DC2B10FA681F780F7E7 40E67B338E263AA80579B84C794B79620106787577593F2E DD130CCB25B6BBC3355208785AE5AADF2C6D526C23344562 22825E2204C34810E5368A0626839C116863F35E0E31C5A4 E7E8CA44A9B7B902F30310AE48479BF36D6F08D1415ACFE9 6FEC4AE9A6776C2B6C1C5F00E0B86D75D2FFF886F6DE9801 484539BA78C8647AC095F2D84D433B6D52F75E497CC977FF D62D6E8B9E41B39ABD0336093842B117E41D3459573F0815 6B384AF599D30F458BCD93668EA223E78E25DC220B9D968A
pk_{WOTS+}^{24}	C948380D14F197D7BA908B0951AB3D5845B23595FB133623
$AUTH_{XMSS}^{24}[1]$	B9EB6AA3E3EB81C1D900D02182D33B967F0E893FCD2331C6
$AUTH_{XMSS}^{24}[2]$	9A07C43874086220148A65C8BDC306F4D48D11D3D9CD87A0
$root_{24}$	205D8361B9FB368C81597D1BAF70A1839F2CE447BF952BBC

A.5. Bộ tham số HB-128-48s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-48s	24	48	6	8	13	18	16	36	96	48	14568

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	D98BA561C444E6709AC4B01ACF193DC87019C8E4D5D39B03
SK.prf	73C754827F2189BEAFA9E95998720A9B74F6C95EA7D6AB22
PK.seed	4E00728B71F1D16E4485D0E65CA2F60BEE9444DCFFB1B5A9
Tính PK.root	
PK.root	5F342BCDDABEEEC294D16912F419B789BB52FD36F728F48F

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	BA368E4597CE0FAD82E76047EF4CEF3F9604D68C85FA4336537691783F54B7643D11824551A 4381663D9354E906341CFB9593ECC743B3E0F34D49856832E7BC8D15C1504B9E655EA636E20 A94257579989CB6B611122EFC716F9CB9526B006F05F53BF9334BD1096DC558C218037BBDFD 017D12ED0BE8E7EDFA52606634D33B5
addrnd	FD82B4F7A209EAEF288B6AE104CCA4965C8A71994CCE380B
Chữ ký (14568 bytes)	A9404631A24B9C81A3E3EDC2AF00A5F63975AA42D3B701FB B07C6A94D0F2AA225E223D4AD7BFD7BE269C13087D17BECC BBA850C2C6425ADF71D1D5399EDBD58BBA9FFA3744EC0E87 ... 051BD2BB70E4987F0887D7F3B45AD0037641A84C05E43BFF 1C71589C74E13C4099C24089A769CE1233CC1D076D896DD8

A.6. Bộ tham số HB-128-48b

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-48b	24	52	13	4	8	30	16	37	96	48	23664

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	492961471E757AE1D3611184C124E8C8F97A2CEEBA82F363
SK.prf	6EC7D7EEF04DF5F66AE30EFC1482B553CA548A2A6F390493
PK.seed	982931D3C1D3BDFFE53F3806153C99477FFC9E14B97AABDC
Tính PK.root	
PK.root	FE848216BCB7C9416E18CBA5D12082326B8D7F25F9A36C7F

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	8EF8B0E3D8F5E7EB907CB229909F48BF85A2C8A83F0B19A169342C5913FEF7E90C6A3BB06F4 3B9CF96DE51A063DC5CFE7110E0BB354AB2F970479E8223AC1C4FFD73F02853C51264F99E19 6086028DB1AA53478D2389BEF951D23107A8CE2098063FA74CFBBFB53810C7AD8038E4C9CEC 353EAEAD843E29441B9215DC2250E00
addrnd	51E863398ADA189140D2E42BB3408677C49283E9FB4FF60A
Chữ ký (23664 bytes)	D024CFF4A48907C1F7B99613D042BFB404D4E9B854B810C5 4311898D44DFAA2BE93E1488D89E22D983CD68395B4C3C20 82B1CE12E82550FD2A56147A75ABBA9D7352A74E0876B10C ... 624824F1ED81DCA44FD4FFDBF87B6FEDC7DF2DD56B552AB1 44B36C22223E487B346FE6E007EE5086F865156E44481C11

A.7. Bộ tham số HB-128-54f

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-54f	24	56	28	2	9	27	16	39	96	48	42120

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	F4F8EBF1ED5B7E4CF7BAAF6F5107BC8D306B4B69ED01195F
SK.prf	5B7C6DA02393160F02957EED5BA5059452030A93338CB2B9
PK.seed	5E02591419648CB731C842DC3A7B84F1084230D1A79080D7
Tính PK.root	
PK.root	1A9A441279626B3F8D8B6C422D29285BA5E64860A9DB45E2

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	199F16E9B965F274EC57728A6C9974BC4E0289D6E8911E28B08FC0BCB1CDDDAEC45975DF422 A0680CEB36D2BFF500ED90673F3E40AD8D9A19E3DE3E6613BD25E0878DE9959ADCF06BEDD50 D8DFA0FC3D390E22EAC46A62722D60FF28A778B5416DA6EE0335473CCC3FF0E12040FC43E3C AF7B662C873AF2C1CE9CF6538C73E56
addrnd	E8EA77885E629467978280C0B9D0244ED96A2E7DC5495886
Chữ ký (42120 bytes)	DB0A06D57A992F84D91F4CF793613048773A0D22D1E643DD DA3742818BCDCB8E47D1891B90B6CA199B6B007878368DB1 A692557BDC82ECC91EC88C7AAD4085977E0ADD56622447DD ... A480B336D2DEF5459549B206802DBEEF10CD32A461180996 2C2ED1867DC38BE25E1A1F030A93C1E5B9A325C895DD6117

A.8. Bộ tham số HB-128-54s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-54s	24	56	7	8	13	17	16	35	96	48	15648

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	30613998DB1B2A131DF96F5B2FB1FB9635A65ED4AE01F1D5
SK.prf	262409680B516437E7C0066826F2EF8E6975A293BED4B234
PK.seed	EE718BDA7B4BCBDD6A99DC8D5560D414A7DFDAC15D3FF59A
Tính PK.root	
PK.root	7B094C32ED7B601E10633858BBD51B582ADEF41EDFB5B3EE

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	35CE98B664C69EC7225F1536FDC43EBC1B8901D084FB8E1F3573D38BC15A603F1CEEC338F8A 8B6C7258761ACD5CB0FF3948614D9EA9EA937D2AD7D6D26EBF91DA71147A18E791445723519 2315BA9AD67C549F3A317DD851EE61AF5994DF5785C09C348DC0865D0835B8A8EE03838BCB8 4DEE32C66D8814ADC09E8B4FAF69E17
addrnd	5A6D95054D953080065814F4D0C617F2F8B9D3A3354BAA69
Chữ ký (15648 bytes)	ABF7B57A792A05FE9E796E86B292AE91F09B1AE679E4E788 140E62007380C7DA52EB597B56DD89E4F88B2C60F598E81A C133A700485762A7C374B04C5ECF233498BB515D0FB232B0 ... 1C5FFCDF93C79FE00C21EC19D0E7F80C0483826302A97564 E747AE59A7FC20A92D6A8EEFC610BD198AA1CA3EA2FCC952

A.9. Bộ tham số HB-128-54b

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-128-54b	24	56	14	4	9	27	16	39	96	48	24984

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	0720DF854739C3BCC8C05BDB16B1657E96919D199AA860CE
SK.prf	B6E3844F1AA498F02F7E92F55C0DA7B3A285946CA683F091
PK.seed	EEE8B65CDDA2B9879787BED3F94CFBCC72D9FB84E99FEF5D
Tính PK.root	
PK.root	6AF2AA51E6E6814B86BE7F19A5FE7ADCB266EE7AD5AE2621

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	56BB99ED6E0EA612E9189D447515A003951B51B5D3C71DDFFC83F610DA62569B83C89D7C021 DE4A683136E4E733F58E95785C63F906504A08D987D9B4AF2BDABC3EEEEF8B418DE1F16BA57 73197A50420278BBA50479BE249D2EA4B705BE82ED8EAAE4A09523BD407678E6217E4B1BB06 62C4ABB88E9AD99B5D4FD14D699312E
addrnd	B8FCA0D73D7ABCD87BF8F351BB6F4CDAE44AB8A95B8644C1
Chữ ký (24984 bytes)	B2915363AF2FB953C9DC2F3B271F9323455B773E89EAC340 E43E8A0AD76259C8275BF092C213A276B651B14A8F6B9B38 DD856475697436A7F8D80DB3F8586F6150722AB76B35B625 ... 0D245907B2B4A3E7C54D07D948F1C9D3E503599D5941DF56 CD8529EE545D48125D8B4F8C95917FAAD30B54EB18D13A0A

A.10. Bộ tham số HB-192-20f

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-20f	32	22	11	2	7	60	16	57	128	64	39680

Kết quả sinh khóa:

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	37A8AB9F2A5CDDFCDCB64C5A28A7A4A1E296FE0D001609492D42787687DD1571
SK.prf	74F5E6B1D3586FB67C6E5D8D26E64B4B59729CB239C16A8F019E447D1AA697DB
PK.seed	F933FD5EA65FB222AD5190232E7529A56959B1CFFC4CDBE15F6AAD044F2AFFD2
Tính PK.root	
PK.root	D27CEA8128EC8345D8B30D8FA065AB2889D79BAB6D6B5A2F95F40D4B0D9C6ACE

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	2B3D9F52CA3A69EFF626DE610FB9FCBD4EBCB50AC86A8A07CE28380BA4B9031FB9937AD87DE A6EB287D7C501E8250EBC8A86E375837E991934142ED95E38E8B3A5A8DB3738B391C0C4566F 0DBBA6B423482A431266641D4B41E6FCAE4AF017E79A3FFF95FB697946EEB39B2F6986D8DC6 0E9FB003E08F494F601DC0172B30624
addrnd	B8A11FABB06379E82501A1D3DD28C0E93C6F6A112662C7AA1E7ADCFB4BFFD754

Quá trình ký:

Giai đoạn băm	
opt_rand = addrnd	B8A11FABB06379E82501A1D3DD28C0E93C6F6A112662C7AA1E7ADCFB4BFFD754
R	970A37776F80B06F4FDCFC5C6125732E62EB778E904D584C0D2B91FF11D30F02A
md	FFA5DA2F6FE27EEC7CD032DE177790373C8A344929C4860C54A8C665728E17A904F28493952 0A2FDEEDDA7466D8CBFF1453C28FF46
idx _{tree}	A2A04
idx _{leaf}	3
Sinh chữ ký FORS	
indices[1]	7F
secret ₁	1C6FEE5325389746573282215155707235AEE70331D1480DC28C004BCE2F0A7A
AUTH _{FORS} ¹ [1]	9377D1EA92BF774D8EE444D93FF83633F978A8F771BCD29F303AC4FC262B8744
AUTH _{FORS} ¹ [2]	25CCDD2BC13F25843D4866BB3342BBBFCC2B8977C1C69CE68B347BBAF987B8B7
AUTH _{FORS} ¹ [3]	AB981A31C2E800AA47488185FB395A60139558E08433A45DF0D98FA383C379D8
AUTH _{FORS} ¹ [4]	85C2CEA6428F31C6AFCEF79399CE841A9C90AF1C292659CFB6232D35C0D7B06B
AUTH _{FORS} ¹ [5]	37D39D55F48ED04864D44F0E1A25D753BD55D0F334679492A4A7B7E1761CA676
AUTH _{FORS} ¹ [6]	78A21DA03C18B22398B1E28BFA4014BF588353DEF5D3EE33E7648DD960D3DF8
AUTH _{FORS} ¹ [7]	3D8CB2589561EA058701A73E3A37B13F20423C79BFAF61974A109F04166CD3DD
root ₁	DC0F65CFDA9733315F88E3037BAD6B5F888813F3B9C3B1288514CA6D10D38913
indices[2]	69

secret_2	6AF4FA8841D9E6E4CA3542658B55F256074A38F33423A471C6FFFEF61C4C8F74E
$\text{AUTH}_{\text{FORS}}^2[1]$	03A3010A64218833EBC9406761562D1515187FC3C10B723A93D3D9ADC211AA24
$\text{AUTH}_{\text{FORS}}^2[2]$	01CE0D52C4460EFC56A873F88CBD0591DE95894F1ABB30D004BCE37B121C3952
$\text{AUTH}_{\text{FORS}}^2[3]$	5BE6DA45E8C136B4810F99D869CD2F408CD60797DB002E710BEBF977BF98171E
$\text{AUTH}_{\text{FORS}}^2[4]$	1C0E05FCC1E8C2DE53098EB65EEA926E9D42C95F8E90DFE0ACA099D05F1CB3A9
$\text{AUTH}_{\text{FORS}}^2[5]$	D7E9AF3CC24FC9B95B1D817150C59A3C78174C4931E1D4EA4A4263EB273BEF4A
$\text{AUTH}_{\text{FORS}}^2[6]$	A3AF626A0DDF1254BC104FA93EBE27ADD9E8F4E37E040F0A6722A02EDDB7B5C3
$\text{AUTH}_{\text{FORS}}^2[7]$	0BDF7371D38F4F921B26EFE910ECF5C43D4006B322B24A13EC010238EE8BB5EA
root_2	1025EE3284DDEA8189AADB31F17B438EB23FA9BA8F6DB20B113EB1A5A2488742
...	
$\text{indices}[60]$	74
secret_{60}	833888C98A38D7B315EEA664479342C55B78FAEA579FBE8311E5D5421C17BE78
$\text{AUTH}_{\text{FORS}}^{60}[1]$	03ED073212A1ACB3AEF4E769FC5FA067715DC71066032DEF67BD3F09FC9D5338
$\text{AUTH}_{\text{FORS}}^{60}[2]$	D30158A5E4DF455E39B73CCC7E498BF522827EA0F6769809C18071EC671A3288
$\text{AUTH}_{\text{FORS}}^{60}[3]$	930AB2FE6F861D29D22E385DD12F32C0D5D6700944F36CD3F5D9BCDF2EDFA11B
$\text{AUTH}_{\text{FORS}}^{60}[4]$	8196DFC2F546B5CDA0E7F85172DFFAAB50913FD434C8435E1CBD2E9618268859
$\text{AUTH}_{\text{FORS}}^{60}[5]$	FCDF49A9990C19CE4AE37874D231C83A98D77BA64A183A41C235CD24B5475B99
$\text{AUTH}_{\text{FORS}}^{60}[6]$	D14D9A0E820AF0B64BD331C242A60A79503A3D4CEE0C069BB73192F184471DF1
$\text{AUTH}_{\text{FORS}}^{60}[7]$	01432B78C4EA4B1814082894EB6A9053D197DCF5F2BC4EE595D553E133114FFD
root_{60}	EA08A139BC35B7AE0D74D2353CE9815165D138E2E948BDC02C1AC41B6129E0CE
Khóa công khai FORS	
$p_{\text{K}_{\text{FORS}}}$	A5481066EF9C15504D1B6D01A358757995E909815565A145E8BCCD5DA0408BD6
Sinh chữ ký Siêu cây:	
Tầng 0	
$\text{SIG}_{\text{WOTS}^+}^0$	7E8AC490DEEBD6E3C111CEE09BA50D817FB1FA9244E8FE850770ADB44E477229 8EEB21E28D2AD3E0A4D5E59204EB3612D6EC6A4C5B81508F4526C8C387B5D59C 12AB21D07A04D178CC67755A6EA17D827A4A021EB2B3B1F3D34ACE81E0A24DB0 AECA2E51D2F53BFF3EC6657367E5E6AC6F893488CC5829861C900D590073F7E88 F2D5B3ADF7CC6B2F96EE03F3707F13CE1997FE4FC9C6D02F3469C9F39A4CA7EC 6B4DBBFD21E1B858C6693AE01610F544DD4D90AA545DED02C1B83FFC1628A827 061E545623385884742EAD72C3BB9D6E920C211A7055F5B2E73C9C0A59A67034 C3B6F41F39966C1014DBBE64B6CDFD1F51EB3D755CD53BD4D37712125A41CEB6 A9FF042D7A1E229EC38A3362C26E038E95043F5FED68C713C21A08C3C6A96E52 550CBB9B41F15A6BC1047FFAA510AFB3F2B2C3EA6101C35DF17266D17ACA0B0B 87970176DBDF20F44BAC65958BF74AB72B2D5BF4357981DFDD31A20853B2B13D EC181C908E58889CF6E81B0E3F22A6FB69C0945E2EBF64E2A71C3D8B8A9BD624 469AFC2C2EF83D5F508DC69AF17C3DC34323C22105A48C872D42AC4CA4363D6B 1AFC276838CE907344082B8EF920C36F50003D1C35178AFC303A5D1B7432D10E 22B5DCCEF5218B322F13CC748A0133D35E0DE70A275AB75E5100592A58A3027F 1965476CEF1FDCE27D2848D86403B10FCFAFB66D8F19B0662399FDEDBBD7B787 430602F5E7526CB397216FC35D0131D398B1ADDBD901567EFCC7DBD681A6ACF CA0DABF32F7FE70F6518F57DA93EA4270B880F4281DB110DAE72E5FA9CA684BC D8B45C2FA0FB63DA32FE4ED223C29495919EDA0EF72D83C17455EF8B8D73D319 D5943897F2836591E56C94BF608B208BDF9C12BD97287CF844F273C4C05212 07C1434409355E91393B6CB19ADDF99F48A238BB476C2F5E5236322A0E474F47 6E92C71088AB9617776CE37F7C6DF7F6BEC971102D78381EC75F0A0EFB225C6D 5E2844C29E8BF87CB4392AEB1A3E5A042E48E97BA8A218E1A16B6C537B8CB499 2B81295819F6C46D74BDAAAC1D6139CB23D540D9432363F3A800A55301A8A181A C5DF97F552DCAD28D5FE5082F8913B89EF41A9D44327756589E886EA0C242391 812DEE85BA4D037EA48F6279719558E311BFBA928572DFEE49BB0AE3338D8D5F 01255B8BCCF98911CF6E411D178AEACA4EF2C86777C6C6C8F5C7AF63E655ABE4 F23B9359BB5514661E792D1EA3DACE7A3BA0847B180E30BF6EB1E3027056629C 1B0EA2694398906CD1A0F4EA3A0DB2B7AAD812231D3230A3982744D8BD071085 D25EEA2E8F89656CDC39D88433A7DC25E0DCDADBDC0AE4AD8BDB21EAA4B7C9A0 56A2D93C8C9CE9785A0CA949D2A280615624BE986D907AB2FBEB8C8952D11B1E3 45D6E3B0FCAD8A74B75C0D400419F117C4AC94BCC19A402E63B66F04B6223CEB 48DAB6087BF1F9BEF6118ABF5CA5DEF82165415A50C1FEC44243C19B3C9346EA

	D2DBF8559414185AC7DC23BADE6F62C8E682B20779C4BC665BC72E2E83ED3A79 D1FFD80B0F712BB0DA343F94C579D060F9BF2549459B19C7293FC3364F74751F 9E6B74B3699FBFA570C4C37CEA4F3B74EC5CCE108DEAD1AECAB2A1829F2FB4E 384DA9D563770EB22430FC413CECA7B1EBDF70A3F118793BCCF2C49A508E1A65 84A50E2B6B538C1B1F44FCDD8BCCD117D3F3724D8FF0FBE4624D2852C443CE7 6531C6116939B638663B6634E41B2EF7D8AD3BABDBB46F7A3FA7E00FC43B2D69 453247A1D996522BCE5FD64AF5B432DEDD844C29FE619820E2D0750809A931F6 56FD196EDE933C2D5DB893C89993F7FBC72E4F1FAFF65B2C1887943E838BC61C 25AD605DE68BDE5FD23FB5EEE5956C2C7BD12AEB9908A52957F016152E37E4A9 15F357CBA24700927B963EDBA6D9594EFE8F03B90F0D49FD6FF803A7FB419E16 45FF4FC508CD14A9E7FE196C96505E53367F85065EF342B6485F257AE8823A6C 79D908FF3D2BE06369BECAB9071229D7B1617C3DC01CA6745FE4C81F5F2DF33F 26843DBA5DF0D9373179383BE6B9886D5D46FDCD970A596551E36AF224B8F7A2 DFB44216FF98C92498D8B7694F2DB094696E4BCD669E5CE038A1F52B788AE966 84F6BFB33641CFEDF4DD146359630EB3CD179002A60B4B203883F4161038BA8B 3B3FA205C1B0357A6B12E729B3B3FACDFA651AA9AEFAAFD07A688ADCD6CA1A7A 0E708FC8A67F063203165B3D79CCEF86659BA30CF6A38013A4BBEF3FDD2539EC D33591023E4B91C976E30A2D3D52827805A0F70D113358135101B9CD5705943E B9738DB57F74604E329AB6311296920643CF76E52220EF4DD3BD37EE12B5CF47 5B86F1993B0B25F7E25BBD52649610A8A86611CEF175333D77ACACD36357D714 3122726DB6FE50C088C8075DE9E0CC9DE22D564537BF1235347D935A920F9890 AA0429DC794354A39ABF80B448B85F979344B702E8164975FCFF39ADB3ADDE78 07276B0B77F60664A7F237064E6ABB2455DA8747DF90808D91F95B6E50A60D0B C7FE03E317B1C833040768D2BDB1DE1965CA8D30518CE86F0A1071A3D654860A B23ADF4ACB735E383A7329756369CD33B593CD485B0777625D05EF937E268FEC 577EE9D5BBA3236C2A18260EF7D53B33C7CD396740EE5C7A35A04033274AEFEA 72DDBF22404AD094C525B7F22DD4CB639B55FD8CD85EC52156FAC4FD14A7C47E 2C0CF661A8F8B970F8E020FBB162574294B432CF1409EF9EFF46690E0FAB07E E67BEDBF6F95C7D333E3A328E27E17B23B49B00CE6846535C02A1D53F9D3FF5A A0EA2ABE3EF72609F1779C55ADB04C15E753ADAAFF7AFEF5E0B2436CF8BD5720 2B44CAECEAAEE6061F0E9FE74F38F92E062E6233C13222664689F2EA0712CEF3 A64A64B15AAC5397468773511F1D3DD240208EE2E23D6CD739BD5E9737F4FC8E A91526941A8008AFE184B6FB4BA61AB826D3F45AE133CB3A0705930A33BABF39 7063AFFDE15B717038866FBE3D707DE1B8798F325E391EA197EE75CBF1DC7269
$pk_{WOTS^+}^0$	904595CD4A4980EB7D1AC4439D13FF6EF13D09E9FB7E39C85830602DA665D537
$AUTH_{XMSS}^0[1]$	06F726E8DBEC289925FEA350B1006DC5E598DCB075043959E3424EF642D223F0
$AUTH_{XMSS}^0[2]$	C470CB5FA2A4FF03803E0E7C19D5AF6B3D628E42E5C7AD16AD10246B8A1FA213
$root_0$	4E8DF3F34DDC48AE68C6BCA2D37DE2A06FFC2426ADE8F581A3FA02BD24A903FD
Tầng 1	
$SIG_{WOTS^+}^1$	DE4154851D29D22EC4AA078096AB038F4D9991A04C92EE9A2B16C4AD42462383 F8A6AEC97529154FB283E06E5408FDD8E84FF20D9525030C8FA49EB02CB57C77 BAC695576DDB2F6F1C703857033C4CD3E437D10778CF5F953E94D5A76CB15FF8 9755AB29D6AC512B3E4CAAF12E49B5A1CF77A97CF086A3E2519839E590FFCECF 6144C1B80DBF82BB5A5B54209AAB131C32AABED6C5DE10FB42A1069B5745D981 F0395AED19D5471FA9880DED3A85B217ADBB142CEB4AD8E51ECF8D9D981E897D 9682E4A15ACAA6151AB2781A7D4633B0D5D9FC564E1FA251EF57BA453B6A9384 0EFFAA30ADC70DAAF125372CCA097E9FA29617CBFDE9F15F86BAAFA17D04BD62 0C2E84A7B5229F3AFD32AFEEC6602D3AFEDF034C98B5554A078B98B14956672E 98F6BDE3983FB133A5CC682416FD112C09FF7DA17066DA79DAB455727854821A 4B41B3B5DEF3A2CC4C9EB765DC3FED52F59EA2DC23B102685463007D7E483F08 7F704EBFADC0BFE2A0651945E5076561DEFB1AE95E741902740434BB6BB4FB07 C63C9B4AAF4CF5C3CD3D029EA696473443D321AAC6AF7AF72DB2CD4C05FEBB19 F71C72B0B3D54CD34516AF869629D6BC686ACD0E08DE024128ECC063539519E0 E48DD50B443D699F0F565E0C11152941B9522A9E9C78E56932A93BA5E1DA0F5B 2EB1157D6145F0CC8EED08A5AA850D0E3C317F7408C5BE4FB52344FC8B0D927D 9783A4E4048CAF3072F2B9049CE8BA91E39B149A41FA2170C451CC415F9CC837 7A4A28C79855A3EB1C33020A8AD377C605F959E84B57AF79404FD73FE11F5B2A 741E04007FFFE6B6724600987E3669CEBBA835EC6620FB47067C84FBC3EDFB B0B56F08D61F70C1C35159CA4CBA9B83163A703FFBB3FC2FA095051D94F51A87 4230FB2FBC5B28DBAD7B529D2CD9E71D268914ACD61DC2825EB610BFA37EF66E

	4FB7173248A49E03427028288BC2332A5DA5BB6A92AAD98AE786807CA13466AD 3B0180A8B50E131BF71954E357758777F9A69786EF2700226EA0F188DD77A530 5E913B3F58BD1B32BBEC6C794A60673E91A5CD49DD3335C88A231A1BA8330FB3 15E7C130CB295FEB90D5CCFFD68A165269D635E380DFDDFBAC2AC9EE94AADF74 BC6249EEDBB17081E8569262AB5E19256B99F9D38502B1CA3F4F065241BED4CA 40E6C00CE13DACCD1626A70FF2832F878436F96B1FCE35F1351E30AC7BFA2D82 B533A6856EAC625DEF5D56A6F3F72BF4A3793CFFCCFE20BEB7612E4E16901365 5EC64F6BB3DF306E1ACFC56CB587C6E5ED1E47BA5445399092A8A7EBDAC26C05 B5DBA0498A62953E960CE9664327BFA4BC6B58349EC5C9B93BDF15C975BACCFE AE06023518F82E6870B671662F307BF06BF4594AE53E19F5BE98C77933E44814 686DA853112FD87EAB761982A5F7ED5AEDEAC66A4C2ADF78FC762D4A854BD57C E577B23FE169064219F408BE8DBFD73D95218A170240373F4921793CD01D20C8 1AA0480DB037FF13A0302028DB0F5E874B99ABB8CB413759CF1EBE2A23768A71 477CEE23354DA04C0D2DFE94384C718791A1F8D38E289F5DA517BA45D0F2C121 46A108050B1FDA88F3FBB0D3ECBDA4038F11B0DAD1C4F3973BCF635C6F17CCB0C B3C68097CA279FAAC92CFFC6CCCAA84010DDDA6760BAAC8115269BC8212E653 4ECD3DAC3BF7534B8C57BFEAAB3181DF6DF8A8AA84474ED564E5F391269C590E C8C762871899B559984C8012549B8CE658F6080B40772DF17C9A5A49246931D0 C78662FD86EE6E182D719B8F2C113546D4C3687C2DE04923617E229B3190428B BCBB2892AC47F22A21A02D544C1DFCF98A35E72D4016D258E823EA9AB1FC70E8 030515716860E42E28102C15733307D69E4C86855F7AB265FF930174CFBA1BB4 B20F57C2B91C07CCE778548F0F218EC9C1D3CC49F127353274F64BECA161AF8E EB75DFC64A5AB4C1B74E5A8BEA6BC0FC68F9A57B51E7F29633698A77D3E2C8E6 8A862E53C2F637B18413E7C958F5C57D9199C80EEABF5C972019802536F725F7 425F3E889255816D923E5ECEBE54EDD820ABADFE23A621B50F347C17FCCA5117 13C6B9923DF1DEAE47DA497DE639105C09F556E032F069BDA0BB1C5235AFCC81 55919E9262E61F43BBEC967175E562F223D4C78DC27A26560A546251150A0F33 4EF728E17BDEE52ED9DF5DEE3804817B5F0F53A783BC1EE882DEAD8B6F4B045E CAA3402C4C4EE0D82F1C9091E0ADE1590B48118EF38C78171E2023C3D20CEA13 A13675A856F5D116C816AC87C60E3E438E12E8657EFA0A39CA9001FA22F655AF 82EEF63B07626BEECAB91D73039ACBC83F892294CFD23D3862CA30EDE3AB9993 A1A3A7FB70AFBC8385F2795B923D0D8086776D3CC7306A613357C0C2E5359AE9 4068D4CBC2BA305179AE2C304E8804CEF391EDD07166419C6C99A7D6C806FBEB B7B9BB810932C686E50B938654D59F612155E32154E03F41FE488812DE566D1A 8462AA7D57E3897AB0DF58715A70335455D8185071973CECDB80DBFAC1E2935 7E89341F6E655D074B705C64CD4E8A0497F47D7E74CCEB8B461E69C80050697A B85039CC03BFD4FBE5960385058AD3AD59A1AC14E63F5D243EA9585EE13A3FC4 B48FF1A8FCA06E1198D907C90B1D626E3136D0A0CA97FDFB0A14F1CC0AC05A76 D0262371A9B90C1D66AD3C1442A86E6CDC805EBAB6AE0B58E5BFB781FFA3C16B 9F50A138C71F907C9A55B9EDD45A0F577D79661852CC3ABECD63C8019FDA425A 6C6687EE7AFADD32BB0CC8ACE1C5468A9751AF5E7CF1F853D81E63F5578C393A 58E90571D708851BD556DBB59B70ECF9DB54AE38049F46BC924B62506B3E80E5 00C1F307FF2C8F77A215A4EB16C73CFFCD39CCA9909467AB57421C66143CB71F D69114A2F33207D03306F1F1269710D3861700D622A76CBB565D7B3CC12988DD A6A9E0B2A196A5784BEE13AD997FCE9D2629279F15B115F29BDA8235002579D2 8D9730B004635C281210C825B5ADBA3CCF3B04EC78F1A271DC83CC962A5F0B1B
$pk_{\text{WOTS}^+}^1$	C55BC113717D2492423AFF5E9EDC07ECD666AE6754939C15D8095F0B0BB06D71
$\text{AUTH}_{\text{XMSS}}^1[1]$	A066BBF9FEEDD827D31521EC0D1122F639E97E69C9B29BB58B2B1AF3DBD0EA4
$\text{AUTH}_{\text{XMSS}}^1[2]$	C9F92A92AEC3499D51FD09D3F125FC4AFC7D43E102C00AD29CA50029AE132635
root_1	9E1A7D6E4D93E5B77BA7671DE6B2A3526E69983C136B8014508227F20952C710
...	
Tầng 10	
$\text{SIG}_{\text{WOTS}^+}^{10}$	1F5C3DB08BB7FBF8B3136226BA61B97A8FFE9D745427365A5AF10A91F3F3EBC6 1910337FD317870CBC045B3DA216B703F3005974844E6FA6148E25BFC628C0CE F5C6F930B5ED079FE537DB59B9CCB7023CCEAE0DBFBD655BC63F0D852BD14E0E ED6E8CC3A79876989C1E89A16C472499550AADBA07C0452B42BAD8C99E761EB5 45A336F459BD3422B92069A431AB2FA41B72464AE9424875D7E3D6333F6CF6B7 EBC3F261FA080B9D32D37507600728E1D91F26DB4DABE38504C94085B2549BE5 22AE4E4358F6AA8CE44F1FB22AA6A682119D500E60D77960448D33D4AC332C65 ED725F7579F5DC77B49FE3DF98EE2222998A2F536E2D4E9508591465445EF228

	6747F66B9C7505515D8B3D1BE9419B99695AD070145850FEB7400C3F48F9AA40 6EE70EB15AA154FC409C01BC43BC8CF2442004579058F5DF90CA70BCC7C3D0EF 6CA52A15016733FA0B51C0B87CCBA81C9DE157392E68B1F9671900AAD43EDFED 799E42FE814D1CDCF029390A4E1A2C083A54BAA549433685DEA39EE07DE3A2F5 329F6B9B16BBB904924F17B21802864AC7C31691525C34CCAA9BADB7D64E089A DF2C87DE73B3629FBF9FCE2A9F219B46BAD222EA0E832DB3EBF8FD83EDF84381 3E34AB32722315078CD58481C3021009F5F838DAC0755B66D501AD49059E301B 93EF58D7B780DB5C74AFB0A786991576BA420F2AC2F0379ADC0AEB2C652184DE D910EB3DE86308D0237E215AE34D98594DF2BBFCE47E857651D0B78593A7DB9A 45F85273FFCA98C6BA2F37528D14170A06E12E66DC819FEB7B43930632F12994 0C62EB0AA8AFE970E6AD2F47F8596080B37C1BD683BFCDA45BEEEE09AAFAE357A 97D5FFA26D19B95D5672CC14941BEBB8984671A74BFF64101EE946497CFAED2E 6766DADD9C044F0068978BDADC071DFB377C08C8C702C417C685BBE47AFDEA5D 087233D6ACBDD154EE3ED3AD002EF4190646E99B2C2B2298621F561DAB55C07 CFF165D4948D709F5B8732369B029306916BFA24F31BF3138316A9B8162AD83B 1ABB0EFE5CF96738A13DA628613E0D0E9465883EE8D958F44F2866562580D333 BE95F633B47EC397276FA1FCAC52C256B88DD8A64106FE20C1A1A4B039A0B98 E7A3171DC897856BAC9BCB68EB2EEC8B5EAB3694B582C3D36CA39859743933AC 8A9BED0A30C4655A6AA57519D5A37E883571FCFE3C0C059C89B486CE8958DDC0 2D1111C9765A43887709AF00BBA817AD6517E2FFC8D7A3569C52E7440B01D813 360B9C7BBD7A60B626CC0451B0E25C4DE93C2843A152A92807074E0BF39FCE7B AEC107D801F2B07EB1434D354E1862E6047A660C760814939416223314B011FA D0EF3B91DBD82FDD40820283046522F90CB45BCFA594F3065F0765EE616F4A8A EBDA44B6D34B02DEB5969DA8D4E798DF563D5D7B4F67575C378DF7EABE931E0F 3F78E94D62F13741B7D72AF34FF03C3658A232B3DF4828AEAB2F830740E3E300 F797C57D137E33238BF4A58F402F8E5BF91F52907A098DB6C1FEAA6FBA3AC345 D9152424A9793DDE2E6B401FB30ECAC3D87CBAF15DF880E680AF4EB087B07D9A A550019F431FB54397DDE52132082984C5AAFCAE8603720EEA360B562116AC23 6575D74A9BDE641BBC94C71F7C092735AE1149C0441707F530584D216C2D9BE8 D644DCBFB46BE135376F36AA54F7A6368FD1849E002970BFC61D428EA65D0192 454E9378DB7DA110F11346308ED33A4AD292A29A1F10AD5C1B244F63E6879CE4 F3A609297E55EF2BF55A5139128F9AB16E8D2EA6AFC8FD332B8F1D50DC1A3B63 40E4A520C6ECA4C63C3C236B6B55D77454098EDBE84E70F4BAB06679BBE0D299 C3934DBFCF2A2C3E9950397FF8426C39D924D9F7FC78565B68381164017A765A 3F7A24D97BAD952883E3A580B19D8F6E8480995A78594EDBDB7E5080B513BB0B 85DA0FC3084D1AB59EE24CBE76FE0E3BA6A002A58DF7AA220626378A24113695 0F9DF3E5FA1D766BD61CB5B07AB838BD14017DBF9749457993B84733D2258E61 4D2CEE6ABC1A2776FC2F2BED3F4D11042BA47B817D342261E1F0DF44910D3B82 176C26B3104DCBE820EE1B2AE36664A16066B236520CD5C7E5C4C168124D9C17 326880416B29895984DC1C7070056E29525CF487DA149EDCAADA628C0D26F244 414888E91893DB75D079A96BCCFBEEED5AC2C5A176980568A9EF29F92AF4F1F67 5041D6B8C4AE30961732E0919EF89C0926A388ED2524E1E09B93C39F8332E44D 82CFEE5A6206C2DA92D3F99B13827AEE197DACFB3EF760DF95FB3EC35A5042C5 16901F7C7FBE2E4C2300723D612C981CEE357CAFD67090855B1AE6ACCD8ABBF1 5FE2B6E2F0AD664C4538EB9C7C62578589E36DD8406D2F5421FBAA05D63882D7 184847621DD091880954EC85B2A72BC0BE89DCDCB998E4FC3CE63A944F0CF1F1 FEAE581C5D90BAB4783ADF845B828D38A246E54C077D967DB842EFCDD1844D962 4868E7CCCE50A777D6485C2112A079AF57CAE4D2DF654CDA884FAB3309150881 A8AF8C3694B69451DCDE7A624BDAC30637D7EC0DAE0165AD6523E2FDC0795C7C 9841B4DABA2803646DC7D948893D063EAE5AA912DFF3E6B40E10945D91023CA1 5E1C545B82EB15E2CEB4B0031121CC7F7D33BB5C8C8399891FB5C0B61E458455 990FA59386B507A290B36E5064A144BB961666B2F76495E8543585C9328F3290 78E13F3B58867B528F6473800AB121F22F315E5B1E4F8A2F2F2833426E5BCC22 9C3B309B20F80B5057479C6FD1B833F9B668187D426F6EBADE268D23089F2575 47548244494ABFC64B1B5E9F9FEEB91C06D1008E6D92E35F9F8BE8D7F8617D26 418C34F7FB36D3A175EE899591D1BB72A0BC71A8F6F5D0B8CC97CBE3426AD633 A742DD18067A32F078F5DC2846E04DC31F5A21E21E87041C08AA5304DDE61C85 349F3DC27C61044E9EA9D4701979E9F2CC77F331B6A8C3017D7AEA5D9FC06E08 0BFF280DA54F2389A8A22425C17275530B19656AC2ADEEF5D1963CDE2BC722B6
$pk_{\text{WOTS}^+}^{10}$	D455C6DF1436807F284031CF4E3D62400F8E4B3CF2CC6CAF48788E86605F4AB7
$\text{AUTH}_{\text{XMSS}}^{10}[1]$	2416365888752A002C4327FC0C0A099480BC98637F63D1362E828CB408CEF674

AUTH ¹⁰ _{XMSS} [2]	89A63230C8E619614F606498010E03CF3F4E9A4F38470BB48F8E26B36EAC4059
root ₁₀	D27CEA8128EC8345D8B30D8FA065AB2889D79BAB6D6B5A2F95F40D4B0D9C6ACE

A.11. Bộ tham số HB-192-20s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-20s	32	24	3	8	11	27	16	41	128	64	17600

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	7BE99B24F5E1312962736C82AAB30152FA97110D31959D135EC98D5451EC4C96
SK.prf	412F077D96296ED3F4042ABB019A5292C0191D5DC3429A925BF96DC97CFA8E56
PK.seed	308F03658B327FF399FCA398CB88E3151EC7BE173C360E18F42A3C9CED4AB7CF
Tính PK.root	
PK.root	2917D834F3E76A88759704C4A71A696DB08076FE8459C92A0CD220BE5147724C

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	2A2247020BABA6ECA7CB63BE1F4D568730A30711CDA7814461B432E9515A612A9014A74DB5E 2D9A2377398F1923FCD4418A45B78D5764A442E31BA12CBF8560EAE35B62C0F14E2A3DBC822 BBB2C4386BBCE1BB1410AC4AF4173916EDA1A86FB2484154A892DB86CE4BEE25ADCF54C5C45 E752F5940BB9A13D07986717E1702A5
addrnd	C7892888772593F0F205ABA1FB2539C27043A99266311B077B74C476B84B3620
Chữ ký (17600 bytes)	7D4B7194BD62B886FCF04787A7270FB10FE269924EB8481BADABA2E1EB35864C 20397AD8B81492B4EC5683D71D1BC0B1CADE1BB79515833113BD5A673CA45235 3CE043631A9D0F577A004403A78376B9D7D751D8414D666DAF155F70EE122150 ... 1854D3CF7D79014822F0FC15BB6EBF52CF6A392C7C57A009A5DACF73F637367F 30A8836B9D9D90BA48B6055B70970D751C37EDA838AA961EFA7ACE4C80604224

A.12. Bộ tham số HB-192-20b

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-20b	32	20	4	5	8	53	16	56	128	64	24512

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	8AE381FD34F680A75E7074A35153629C4EA771B82AA5171A8F323CBB7EFE7A2F
SK.prf	5F41E6CF8FD1ECBA1CC6C1B65C3577AE5B876A0F7E3FA94A7841A74964250D00
PK.seed	6DA2763AD2438F463237A0206CD369A4C211BAC0BD7EDDF1FDA041675AF46CC2
Tính PK.root	
PK.root	927CC1879334CE4055E2DB3FC9321C1009CD4068DA7822FD628215717096FEC7

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	E395DD50AA8B48A1198FE7C8A70395715075C8D77220FDF9725870584D4E70456198A7FEE3 BCE5649F2CE2C4DCBAFA47D874DEB8A3A9EED44D19BE89BA034958DB4694AA3189E21E2388C D3CE269529776D6CB8C69E1DFCD71AC0FEF51DEC989B7909188D34BC5C5C50E332C9A88417F 3C0D4A1DCE8139F3005B1F31D2CFBF8
addrnd	F2DFD799AB61213C5D801FEE3EAF946D9105361B5F07D3861E402D8C276EFE16
Chữ ký (24512 bytes)	B2B2927C8E69B25F34F1D58AD8A6126348A56CF6A2DB363AFEB2EF49E1CCE689 3FADF9F64F44F7053AC0D7BE53A80BA553FE33863AF90E6827F7642775A03868 D9EF9EE19043442CECA7FD0DCBEBA883C9964410FD18B1433B91FCC93F029A67 ... 834CEC19B29F7CD232542C09D384572D832A22A930F7762FB43ED6028FEF80A8

	8AA56EFC8AE959A32489A2D7AA7D410CEB8E1878039181336C0CB619F6AA2EF3
--	--

A.13. Bộ tham số HB-192-48f

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-48f	32	50	25	2	7	60	16	60	128	64	70592

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	68A13073B8B6DA56E29C6BF4BFFF974A6B48E45A7E93B08E16EA52EF121EAD5
SK.prf	435C67CC3D62934A1FF2750849A9F777C12777433E2CE0ABA99D1DD726D7E73A
PK.seed	53D138BD343FA42A60B26E9557E9C23BBA9332896F95E9D61FC2CCFABD353D59
Tính PK.root	
PK.root	CA95979BFE5944511BDED67C4A9AD8A765BC027C47C265DFA8FF0A4C467B28FD

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	19837339DF069CDB1B9F09B560C4DDF5286D150E1C71D7090DAF53E9E10692D1EFA98B01642032581CC48DAFAE8DA7D6353EF7BD3E31882B53B299EE68F4027D9D73153F98D34425C71A85AE0091AC93483793CE8CDDA8AD1E2C3D292957939A3224BD96CC1FC85D50E62B2FD07CF26D0D474533CA8DA1CD3213D6E0690E9523
addrnd	694EC576343A7413F2CED3EC9E16524FB33EDD55224A0A97DB00A2E7E6D40222
Chữ ký (70592 bytes)	A6DA03B44849F0A7F4628FDD3F85FC70CFC875CF36481036CBA2914EEACF364EE7C3F0CD15ECFE8D0E28787B4C2F73924F2ED69C21C5530016CA985769844CF3312E755F6CB8325266315690DE1146A34C32D38AD83691D0D5E611510AD2528D ... EA489CD1A58DBC000A3CA26E2EB1E24F929FBC55A1CE4F00A75482605A99E17D7DF863B2CC0399592326AA8305B35691B92280A68C5D00073851E19E84B22ECA

A.14. Bộ tham số HB-192-48s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-48s	32	48	6	8	12	28	16	48	128	64	26080

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	24E602F9EE419474DF6B8755721F67F405C751B31D2FEE32FEA4475929D0185C
SK.prf	85A45ADFD0FCB0773BAF2353D46B6E6200EF1864B8277E797AB84401DA1310713
PK.seed	A092C3E45625D36DCAC07BC40E307EF18F0A94811C8C2C9F9E887E4870226700
Tính PK.root	
PK.root	1C72C4AE60038088121A735C80DEAE8681C2316F4A9882B7B618285F6C5A7693

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vectơ kiểm tra.

Thông điệp (128 bytes)	B05DD2358A31DF646876ED81D6D95FFF1F7D85CA2F6E95BD2B240E0726206F10797267B7FC62188DB57A32B51693D69D3649378432CBD1CF3608CB5CD704406CA2E30FC68627CAC70482DD9108E14D87D17BB7A9783E33758A162AAE48865CDC7E1BB63EA28FC850C0DAD7272B889BDD3D6B6308B8F03CFB4A539A7B9BFECB5D
addrnd	E21CFBE3B49C824B3F394F766F716FA3AFDBDC982F2ECD57E9A76A00DF3D30B3
Chữ ký (26080 bytes)	9BF5AEF233D1B75981692EEF5762D343763C0FFFEF23BA4C6791E630381CC6F0584ECE3CA588F796ECBEA56A7DE7C4E190D6F16D158FC66F4796DB5BF005867CBC18857BEC00F08286E549341FF91E0A35A5426290A7E27D93783C9ED944D02C ... 26B9D6AEAF4BE8679DA11A36000D5D8B8F24AE545267C7DD63A993E036B6B8613529A9CDA51158ABF05BDD2B2FD10F224407F6EF6D2768586B23BA87929D2AA7

A.15. Bộ tham số HB-192-48bBộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-48b	32	48	12	4	8	53	16	60	128	64	42560

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	9800F14573220A4257F4452698FF07C10F23493DD13B1D306C183B83203FBB83
SK.prf	CB0AE35CFE114512FC626A2CD0F2C62A038E194356E04631241BA5BC460D1B0
PK.seed	DD1A2BBCD7AC31503581D4BDA88355B6DCF4477699C28B387B53A22C15868CD2
Tính PK.root	
PK.root	80D6D4376D53A08813352EBE140A70D0A5B14D62FAC02C57B76ECC99F90F71A7

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	545D03F98CAC15562178EAE8E1EDD0213B76742602A203051F6BF99B6D1606B0D34B9DF20E7 71794FA5A6FCD583EC30C37B136BAFE96E78C7D88EE0A0B4215AA1C86B86F18247FBE8AB936 FA60060812A90A70DEF54306594224B9F3F75149A04AB31C3527C2F54A741B790FEF0AD7BCB BC2F04EEC32F40F635D2A9CAD399B25
addrnd	E9FB753E483AAC92E776EA2DF6CED3476515251971FAE55AC1858676C9D49BED
Chữ ký (42560 bytes)	B11FE72433632FD26DAF19AA17A17650E011496EC3920AC6431071C2742B686B 73E0B06EAFD8BBD8B0011D0D742CCBFC91C473A1FEBC0DE1900B8692243C3F463 10C373A03D1501675E39253085D1114FF72F969F578AA691AFE5C8872526F01D ... CA556EA3E55126D6117A91A4EA3A3E64DDFD5C0E67CC03762A40A30202393D75 1F71C7FA09DE0771167C6A56DE2289C08475AAD513E76930BC48579974CEEAE

A.16. Bộ tham số HB-192-54fBộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-54f	32	57	19	3	10	32	16	48	128	64	53856

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	E18BC3A774EDB85EDC5C352BCA3CFE537B4850103F4BC0DE88AA8B19325B05F7
SK.prf	E835EFCC940B2E68833359AECADF8C3958CDD28CB83B7C8366370EB50BE61842
PK.seed	699510CCD303CA327C7DC9A40F181B05C918824FBEEBE8D9FF639BAB568BE668
Tính PK.root	
PK.root	F5F93A78B496F17DD6BCCD7643513B8EE05C351688E6CF7D92DC54F9163B7B2B

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	16F72D3014393A012E1ACA7A47D1B367BCD7D15783F3670C9A56B55FCC0159F0943EB5F1C7A 375FAA2CB469DA082DB1646D4B73204FB20BDC7E2BA3C99911086B7E3C578943D1678812B13 1EDA3E092D00F940E28EAD5FBF77D9B288A09B804C76FF71B6DC4A1BD9F7A91D35FF364B03A 9F5DE512D01659DADBDB52D188F1ADF
addrnd	7AF5F22A1877C2D19DA4216C54F4DD1A92F5CAA1124CC776B51AEA2E46351DB6
Chữ ký (53856 bytes)	C534058EFF9D62284B1130BCC985289DE009B98F176B55D150443CD192AAE537 4DC97043E7CFCD7F0494611B534C3424B01EFE92CBE5A08792165C7542BAAD6E 630AB3A23DDBF8D1C5E247171AAD1F9B49AA43E314A68B15EC208463CD457A3A ... EA77C2A40BF5A35B441609345830CA561CBFDDE742C021577C29AEB38FFB4E0 6CEB970154327CEF4B676CFD3EB7191810F082E73612F0312EB6271CBE3CB417

A.17. Bộ tham số HB-192-54s

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-54s	32	56	7	8	14	21	16	44	128	64	26912

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	C51FB48E000A097A73DBB502365784D01500CD8F3DBB63C71D9C3880D5BA2AA7
SK.prf	7A320CBD04B33EC104D84C26806FD56E816DF8498404C879103364DDE92993B6
PK.seed	B0CB7D44AC6EF4A54CD302EAF91D11ECCDD3715C381EEC1DCE0282C4DCA04F6
Tính PK.root	
PK.root	888CC367FB94E838952207EFAF14E026C54FE62784F14045EA85FCA389F91CB4

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	33CCEDB81B6B8C831453093DDD121B1383E98E105ACAA4741F1654A5D6F7DFCC7C6B0E035B73396D964DC8B400EED2EA3481A7E0A646AB5BEC6D28E9E3AEEB528BFD04C081490342D48BEA570393CE651C113DC946040DDDCBAABD2F87667BB3C990EFAB330E532357F0D70953B0C3423F56F6F182378D4F026BE29997CD150
addrnd	4C71E8697FC23F6576B7D46DA5A4310AE6E1F2EB40ABCBF13CDA53B0684C25AA
Chữ ký (26912 bytes)	31A0E339ABC33A89569AEAAC3577FD3FCE377D45CF277E6C6805FED41ED6DF5840737052DA6E9B2BA60679D97420E213F99289E06ABA7810D8E0AB2B56867D319306A4FFE718EF9CB492A7A189789837B45902BACB8C3F9E83C5CBB5241B5C8E...AB7AF3B04A045F91941AC0467B816541A8D3B18C1263028AFA3978DBE84EA7D81A130CB89C6B33BDA750E08349FF6A7353F8279DAD6A7A3A1D6AE8F99CBF47E5

A.18. Bộ tham số HB-192-54b

Bộ tham số:

	n	h	d	h'	a	k	w	m	sk_size	pk_size	sig_size
HB-192-54b	32	56	14	4	11	29	16	48	128	64	42976

Kết quả sinh khóa

Các giá trị sinh bằng bộ sinh số ngẫu nhiên	
SK.seed	9FBCAE390EE385AE6152464FD5B5B0AA70EFB79B1ED5DC8895AD28B911E00B16
SK.prf	45853E6B7CD89B2475D792869DE05FADAF8BDAA4F0E7E89D1268316B5B64738E
PK.seed	4A9A3689AF0B27ECD1BD90F121AD5F487DE5DB9DC7EA23C9227E8F8DA766C0B7
Tính PK.root	
PK.root	0ECD89880627259281F2944D4F27C54B32DBF7936FC96FE0755FE42FD96223F8

Kết quả sinh chữ ký: Thông điệp ký (độ dài 128 byte) và giá trị addrnd được sinh bằng bộ sinh số ngẫu nhiên được sử dụng trong quá trình sinh vector kiểm tra.

Thông điệp (128 bytes)	D5FE1F7F6B65CE81CE8FD5E0897472AF501C8E6E81C2388AC74E541354E12487B46728153BE5455AF8C8DDA99C4E59C9A4A6F28167043B5341DE94DB56FE97EF04E4D1331AD93502C36EA310CB255FE58EE12C8529BB1539ED2C3C1EF0A1B91A834515DA26CFC2A0C38465862A014F5A37EAD9C0FB5941922DF8BDFE50512FB8...4DF0858CC7775B722D383D7082CD1757CBB35524CAEFA7449CD7ABF917579F09BD96A0CE5F0058704ED3F30A1B0A5E312F54CF6C3272A5D0C4DFB900A6A828B
addrnd	D5FE1F7F6B65CE81CE8FD5E0897472AF501C8E6E81C2388AC74E541354E12487
Chữ ký (42976 bytes)	B46728153BE5455AF8C8DDA99C4E59C9A4A6F28167043B5341DE94DB56FE97EF04E4D1331AD93502C36EA310CB255FE58EE12C8529BB1539ED2C3C1EF0A1B91A834515DA26CFC2A0C38465862A014F5A37EAD9C0FB5941922DF8BDFE50512FB8...4DF0858CC7775B722D383D7082CD1757CBB35524CAEFA7449CD7ABF917579F09BD96A0CE5F0058704ED3F30A1B0A5E312F54CF6C3272A5D0C4DFB900A6A828B

Thư mục tài liệu tham khảo

- [1]. TCVN 12853:2020: **Công nghệ thông tin** – Các kỹ thuật an toàn – Bộ tạo bit ngẫu nhiên.
- [2]. TCVN 11816-3:2017: Công nghệ thông tin – Các kỹ thuật an toàn – Hàm băm – Phần 3: Hàm băm chuyên dụng.
- [3]. Bernstein, D.J., et al. "The SPHINCS+ signature framework". in Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019.
- [4]. Hülsing, Andreas, and Mikhail Kudinov. "Recovering the tight security proof of SPHINCS+". International Conference on the Theory and Application of Cryptology and Information Security. Cham: Springer Nature Switzerland, 2022.
- [5]. **Hülsing, Kudinov, Mikhail**, et al. "SPHINCS+ C: Compressing SPHINCS+ with (almost) no cost". Cryptology ePrint Archive (2022).
- [6]. Turchenko, Oleg, and Sergey Grebnev. "Hypericum: a post-quantum digital signature for standardization in Russia." Journal of Computer Virology and Hacking Techniques 20.3 (2024): 455-483.
- [7]. FIPS 205. "Stateless Hash-Based Digital **Signature Standard**". Federal Information Processing Standards Publication, 8/2024.