ETSI TR 103 692 V1.1.1 (2021-11)

CYBER;
State management for stateful authentication mechanisms

Reference
DTR/CYBER-QSC-0016

Keywords
digital signature, Quantum Safe Cryptography

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Implementations of Stateful Hash-Based Signature (S-HBS) schemes require exceptional care to ensure they are done securely. Existing specifications are complex and prioritize the interoperability of implementations at the cost of certain security and operational considerations. An implementor of a S-HBS scheme, using only the existing specifications, is likely to experience unforeseen security vulnerabilities and operational problems due to the under-specification of the state object and its management.

State management is not only about ensuring state is not reused. There are operational considerations of state as well, such as the capabilities of the physical systems the algorithms are run on, and the inherent suitability of S-HBS solutions for specific applications.

1 Scope

The present document is limited to discussion of the characteristics of the state object, the reuse of the state index, and of architectural and operational considerations for deploying stateful hash-based signatures. First, it discusses characteristics of the state object for S-HBS schemes and identifies potential security vulnerabilities and operational problems associated with its management. Second, it gives guidance on mitigating the issues identified. And third, it helps a prospective implementor determine if a S-HBS solution is suitable for their given application; examples of suitable and non-suitable applications are given.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] IRTF RFC 8391: "XMSS: eXtended Merkle Signature Scheme", 2018.

[i.2] IRTF RFC 8554: "Leighton-Micali Hash-Based Signatures", 2019.

[i.3] D. A. Cooper, D. C. Apon, Q. H. Dang, M. S. Davidson, Morris J. Dworkin and Carl A. Miller. "Recommendation for Stateful Hash-Based Signature Schemes", NISTIR 8240, SP 800-208.

NOTE: Available at https://csrc.nist.gov/publications/detail/sp/800-208/final.

[i.4] P. Kampanakis and S. Fluhrer: "LMS vs XMSS", IACR ePrint Archive 2016/085, 2017.

[i.5] L. Lamport: "Constructing digital signatures from a one way function", Technical Report SRI-CSL-98. SRI International Computer Science Laboratory, 1979.

[i.6] J.-P. Aumasson, D.J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kolbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe and B. Westerbaan: "SPHINCS+ Submission to the NIST post-quantum project, v.3". October 1, 2020 .

NOTE: Available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[i.7] D. J. Bernstein, J. Buchmann and E. Damen, (eds.): "Post-Quantum Cryptography", Springer-Verlag Berlin Heidelberg, 2009.

[i.8] J. Buchmann, E. Dahmen, and M. Schneider: "Merkle tree traversal revisited", LNCS vol. 5299, pages 63-78, 2008.

[i.9] M. Jakobsson, F. T. Leighton, S. Micali, and M. Szydlo: "Fractal Merkle Tree Representation and Traversal". LNCS vol. 2612, pages 314-326, 2003.

[i.10] A. Genêt, M. J. Kannwischer, H. Pelletier, and A. McLauchlan: "Practical Fault Injection Attacks on SPHINCS", IACR ePrint Archive 2018/674, 2018.

[i.11] D. McGrew, P. Kampanakis, S. Fluhrer, S. Gazdag, D. Butin, and J. Buchmann: "State Management for Hash-Based Signatures", IACR ePrint Archive 2016/357, 2016.

[i.12] J. Katz: "Analysis of a Proposed Hash-Based Signature Standard". Security Standardisation Research: Third International Conference, SSR 2016. LNCS, vol. 10074, pp. 261-273. Springer, 2016.

[i.13] S. Fluhrer: "Further Analysis of a Proposed Hash-Based Signature Standard", IACR ePrint Archive 2017/533, 2017.

[i.14] E. Eaton: "Leighton-Micali Hash-Based Signatures in the Quantum Random-Oracle Model", IACR ePrint Archive 2017/607, 2017.

[i.15] A. Hülsing, J. Rijneveld, F. Song: "Mitigating Multi-Target Attacks in Hash-based Signatures", IACR ePrint Archive 2015/1256, 2015.

[i.16] F. Campos, T. Kohlstadt, S. Reith, and M. Stoettinger: "LMS vs XMSS: Comparison of Stateful Hash-Based Signature Schemes on ARM Cortex-M4", IACR ePrint Archive 2020/470, 2020.

[i.17] NIST: "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process", December 2016.

NOTE: Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[i.18] National Institute of Standards and Technology (2001): "Security Requirements for Cryptographic Modules" (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-2, Change Notice 2 December 03, 2002.

NOTE: Available at https://doi.org/10.6028/NIST.FIPS.140-2.

[i.19] National Institute of Standards and Technology (2019): "Security Requirements for Cryptographic Modules" (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-3.

NOTE: Available at https://doi.org/10.6028/NIST.FIPS.140-3.

[i.20] P. Kampanakis, P. Panburana, M. Curcio, C. Shroff, and M. Alam: "Post-Quantum LMS and SPHINCS+ Hash-Based Signatures for UEFI Secure Boot", IACR ePrint Archive 2021/041, 2021.

[i.21] A. Hülsing, C. Busold, and J. Buchmann: "Forward Secure Signatures on Smart Cards", IACR ePrint Archive 2018/924, 2018.

[i.22] M. J. Kannwischer, A. Genêt, D. Butin, J. Krämer, and J. Buchmann: "Differential Power Analysis of XMSS and SPHINCS", IACR ePrint Archive 2018/673, 2018.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

asymmetric cryptography: cryptographic system that utilizes a pair of keys, a private key known only to one entity, and a public key which can be openly distributed without loss of security

authentication path: ordered collection of sibling nodes along a leaf-to-root path in a Merkle tree used to compute the root node of the Merkle tree

binary hash tree: tree structure created by iteratively pairing and hashing leaf nodes

cryptographic hash function: function that maps a bit string of arbitrary length to a fixed length bit string (*message digest* or *digest* for short), and that fulfils some specific security properties

height: number of nodes in a leaf-to-root path in a Merkle tree, inclusive. Equivalently, number of levels in a Merkle tree

hyper-tree: hierarchical structure of binary trees, divided into layers

intermediate signature: signature on a Merkle tree root node within a hyper-tree

layer: index of the vertical position of a Merkle tree in a hyper-tree structure

level: vertical index within a Merkle tree

NOTE: The level of a node is given by the number of nodes along a path from the leaf-level to the given node, non-inclusive.

Merkle tree: binary hash tree used as a component of a Hash-Based Signature (HBS) scheme

message digest/digest: fixed-length output of a cryptographic hash function over a variable length input

node (root, leaf, internal, child, sibling, parent): octet string, serving as a minimal component of a binary hash tree

octet string: ordered sequence of octets/bytes consisting of 8 bits each

private key: key in an asymmetric cryptographic scheme that is kept secret

public key: key in an asymmetric cryptographic scheme that can be made public without loss of security

public key cryptography: See asymmetric cryptography.

security level: measure of the strength of a cryptographic algorithm

NOTE: If 2^n operations are required to break the cryptographic algorithm/scheme/method, then the security level is *n*. Sometimes also referred to as *bit-strength*.

stale state: index corresponding to an OTS key pair which cannot, or can no longer, be used to securely sign a message

NOTE: An index can become stale for a variety of reasons, such as having already been used to sign a message, or as a security mechanism after a system restart.

state index: non-negative integer representing the position of the next unused one-time signature signing key within a S-HBS scheme instance

state object: collection of data related to a stateful hash-based signature scheme instance that is required to compute or verify signatures from that instance

3.2 Symbols

For the purposes of the present document, the following symbols apply:

A || B	The concatenation of binary strings A followed by B.
$H(\,)$	A cryptographic hash function.
N_i^j	The i^{th} node on the j^{th} level of a binary hash tree.
L_i	The i^{th} leaf node of a binary hash tree.
h	The height of a binary hash tree.
w	The Winternitz parameter.
$\mathcal{L}$	The number of layers in a hyper-tree hierarchy.
$\mathcal{H}$	The total height of a hyper-tree hierarchy.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
BDS	Buchmann-Dahmen-Schneider

CA	Certificate Authority
CFRG	Crypto Forum Research Group
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
FTS	Few-time Signature
HBS	Hash-Based Signature
HS-HBS	Hierarchical Stateful Hash-Based Signature
HSM	Hardware Security Module
HSS	Hierarchical Signature System
IRTF	Internet Research Task Force
LM-OTS	Leighton-Micali One-Time Signature
LMS	Leighton-Micali Signature
MTS	Many-Time Signature
NIST	National Institute of Standards and Technology
OTS	One-Time Signature
PKI	Public Key Infrastructure
PQC	Post-Quantum Cryptography
RFC	Request For Comments
RSA	Rivest-Shamir-Adleman
S-HBS	Stateful Hash-Based Signature
SP	Special Publication
TLS	Transport Layer Security
US	United States
VM	Virtual Machine
W-OTS	Winternitz One-Time Signature
W-OTS+	Winternitz One-Time Signature Plus
XMSS	eXtended Merkle Signature Scheme
XMSS-MT	Multi-tree eXtended Merkle Signature Scheme

4 Background

4.1 Hash-based Signatures

4.1.1 Introduction

The theoretical security of a cryptographic signature scheme is typically derived from the difficulty of solving an instance of some underlying mathematical problem, such as the Discrete Logarithm Problem or the Integer Factorization Problem. Hash-Based Signature (HBS) schemes are atypical in this sense as their theoretical security is based on the security of an underlying hash function.

Although the security of hash functions can themselves be modelled after mathematical problems, making a distinction between these two cases is useful. If an HBS implementation uses a hash function which is believed to be resistant to attacks from quantum-capable adversaries, then the resulting signature scheme will also be (believed to be) quantum-resistant. Further, if the underlying hash function is eventually broken, or if a different hash function is desired for any reason, the scheme can be repaired by switching out and replacing the hash function. With signature schemes such as RSA or ECDSA, a break in the underlying mathematical problem is not generally fixable.

NOTE: Although their theoretical security is based on that of their underlying hash functions, the specific security requirements of the underlying hash functions are not the same for all HBS schemes. Different HBS schemes sometimes require different security properties of the hash function they employ. For this reason, the present document does not explicitly discuss hash function security properties, but instead uses the term "cryptographic hash function" to imply a hash function with the relevant security properties for the HBS scheme under consideration.

The issues of state management are particular to a specific class of HBS schemes, namely, *stateful hash-based signature schemes*. Such schemes are built from a variety of cryptographic algorithms and mathematical techniques. The following clauses describe the main categories of HBS schemes, including the components that comprise such schemes and their related mathematical concepts. Clause 5 and thereafter specifically discuss stateful hash-based signature schemes, the management of state, and applications for stateful hash-based signature schemes.

4.1.2 One-time signature schemes

4.1.2.1 General

A One-Time Signature (OTS) scheme is a cryptographic signature scheme where each instance is secure if at most one message is signed with the signing key of that instance. Re-using a one-time signing key to sign multiple messages greatly degrades the security of the scheme instance, allowing forgeries to be feasibly computed. The reduction in security is because a one-time signature leaks partial information of the private signing materials of the scheme.

EXAMPLE: Signatures from the original hash-based OTS scheme by Lamport [i.5] revealed an expected 50 % of the private key. Provably, an attacker with 50 % of the private key cannot feasibly forge a signature. However, if two signatures are computed under the same instance of the scheme, 75 % of the private key is expected to be revealed, and making forgeries becomes feasible.

In some OTS schemes, a signature directly reveals components of the private key, and in other schemes a signature reveals sensitive information derived from the private key. In either case, by seeing multiple signatures computed under the same OTS signing key, an attacker likely has acquired enough sensitive information to feasibly forge signatures.

OTS schemes are used as building blocks for the more general schemes described in clause 4.1.4.

The present document is primarily concerned with the HBS schemes specified in IRTF RFC 8391 [i.1] and IRTF RFC 8554 [i.2], and the corresponding profiles described in NIST SP 800-208 [i.3]. IRTF RFC 8391 and IRTF RFC 8554 define the OTS schemes WOTS+ and LM-OTS respectively. As both of these OTS schemes are based on the Winternitz OTS (W-OTS) scheme [i.7], the following clause gives a brief overview of the W-OTS scheme.

4.1.2.2 The Winternitz OTS

In a digital signature scheme, the public key is required to verify signatures that were generated using the corresponding private key. The Winternitz one-time signature scheme, and variants thereof, have the property that their public key can be recomputed from a valid signature by running the signature verification algorithm. At first, this property does not seem particularly useful as the verifier does not have any way of knowing that the recovered public key is the authentic public key of the signer, unless the verifier is also supplied with a copy of the signer's public key. In a stand-alone OTS scheme, it is not generally practical to pre-distribute the public key in this way. However, the W-OTS scheme is used as a building block for the more general signature schemes discussed throughout clause 4.1.4. In the context of these more general schemes, this property of W-OTS signatures can be used effectively. The property allows the sender to not include their OTS public key with the signature, thereby reducing communication overhead. This process is described further in clause 4.1.4.

In the original W-OTS scheme [i.7], a public key is a collection of n-byte octet strings. IRTF RFC 8391 [i.1] and IRTF RFC 8554 [i.2] specify OTS schemes that are variants of the original W-OTS design. The WOTS+ scheme specified in IRTF RFC 8391 has public keys that are also a collection of n-byte octet strings. When used as a component of the XMSS or XMSS-MT signature schemes (clauses 4.1.4, 4.2 and 4.3) the WOTS+ public keys are compressed into a single n-byte octet string by using structures known as L-trees. Public keys in the LM-OTS scheme specified in IRTF RFC 8554 are similar except that they are compressed into a single n-byte octet string by concatenating together each public key component and computing a hash value of the resulting concatenation.

Throughout the rest of the present document, OTS public keys are implicitly assumed to be n-bytes, where n is the output length of the employed hash function, in bytes.

4.1.3 Few-time signature schemes

A Few-Time Signature (FTS) scheme is a cryptographic signature scheme where each instance is secure if some, but not too many, messages are signed with the signing key of that instance. FTS schemes can be thought of as a slight generalization of OTS schemes.

The core difference between an OTS scheme and an FTS scheme is that FTS signatures leak a smaller fraction of private information relative to OTS signatures. In this way, an attacker needs to see a few signatures before they have enough information to compute a forgery. The number of signatures that can be computed securely in an FTS scheme is dependent on the parameters selected and the specifics of the signature scheme itself.

FTS schemes are used in stateless HBS schemes (clause 4.3) such as the SPHINCS+ NIST PQC Round 3 Alternate Candidate [i.6]. Guidance on FTS schemes is outside the scope of the present document.

4.1.4 Many-time signature schemes

4.1.4.1 Introduction

A many-time signature (MTS) scheme is a cryptographic signature scheme where each instance is capable of signing many, but not unlimited messages under the signing key of that instance. MTS schemes are constructed from instances of one- or few-time schemes by using binary hash tree structures (clause 4.1.4.2). The security of MTS schemes is largely based on the security of the underlying OTS or FTS schemes and the hash function used to construct the binary hash tree, as discussed throughout clause 4.1.4.

MTS schemes are sometimes called *full HBS schemes*; the present document uses the two terms interchangeably.

Clauses 4.1.4.2 to 4.1.4.5 describe the generic components of an MTS scheme.

4.1.4.2 Binary hash trees

Let H be a hash function with n-byte outputs. A *binary hash tree* is a data structure built from iterative invocations of H on an ordered collection of n-byte octet strings, called *leaf nodes*, as described below.

The present document assumes a binary hash tree has 2^h leaf nodes, where the exponent, h, is the *height of the tree*. Each leaf node is indexed on the 0^{th} *level* of the tree. The tree is constructed via an iterative process on the leaf nodes. There is a single node at the topmost level of the tree, called the *root node*. Nodes that are neither leaves nor the root are called *internal nodes*. Nodes used to compute nodes on the level directly above them are called *child nodes*, where those pairs of child nodes are called *sibling nodes*. A node computed directly from two sibling nodes is called the *parent node* of the two child nodes.

Level 1 of the hash tree is constructed by concatenating pairs of sibling leaf nodes and applying H to each of the resulting $2n$-byte octet strings. The outputs of these computations are the nodes of the first level, where the ordering is maintained; there are 2^{h-1} nodes on the first level. When used as a component of an HBS scheme, additional data can be included in the hash computations. The process is iterated to exhaustion. The final output of H is the root node.

Nodes in Figure 1 are indexed with a superscript to denote the level of the node, and a subscript to denote the position of the node on that level, where indexing is done from left to right starting from 0, except for the leaf nodes which are notated differently for readability-the leaf nodes can alternatively be indexed as $L_i = N_i^0$. Level 0 is also called the *leaf level*.

EXAMPLE 1: In Figure 1, $N_0^1 = H(L_0 \,||\, L_1)$, $N_1^1 = H(L_2 \,||\, L_3)$, $N_0^2 = H(N_0^1 \,||\, N_1^1)$, and so forth.

The *height of a node* is the level the node is indexed on within the tree. Equivalently, it is the number of nodes along a path from that node to the leaf level, excluding the node being measured from. Therefore, the root node has height h, leaf nodes have height 0, and internal nodes have heights between 1 and $h - 1$ inclusively.

NOTE: Because the number of leaf nodes is assumed to be a power of 2, all paths from a given node to the leaf level will be of the same length.

EXAMPLE 2: In Figure 1, the height of node N_0^2 is 2 because all paths from it to the leaf level contain two nodes. One such path is N_0^1, L_1. Similarly, the height of the root node N_0^3 is 3.

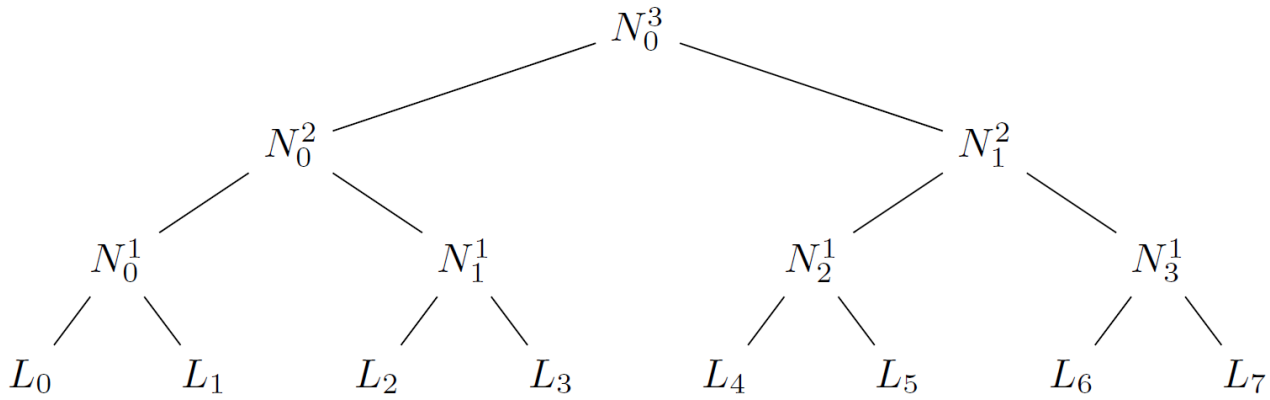

Figure 1: A height 3 binary hash tree

Assuming a cryptographic hash function is used, the root node is determined by the leaf nodes. If even a single bit is flipped in one of the leaf nodes, the resulting root will, with very high probability, be computationally independent of the authentic root. That is, the bit-wise exclusive OR of the two root nodes will be computationally indistinguishable from a uniformly random string of the same length. This is one of the properties of binary hash trees that makes them so well-suited for constructing many-time HBS schemes. This property is discussed further in clause 4.1.4.4.

A binary hash tree used in an MTS scheme is called a *Merkle tree.*

Any tree node is itself the root node of a *sub-tree* embedded in the full tree structure. For a given node N, the corresponding sub-tree has the same height as N. If two sub-trees have no nodes in common, they are said to be *disjoint*.

EXAMPLE 3: In Figure 1, the sub-tree corresponding to node N_1^2 is the height 2 tree constructed from leaf nodes L_4, L_5, L_6, and L_7. The sub-tree corresponding to N_0^1 is the height 1 tree constructed from L_0 and L_1. The sub-trees corresponding to N_1^2 and N_0^1 are disjoint.

A common algorithm for computing a Merkle tree is the TreeHash algorithm. Pseudocode for the TreeHash algorithm is given below. The pseudocode is presented as Algorithm 2.1 in [i.7], from which it is taken directly, except H is used here instead of g as the cryptographic hash function, and h for the tree height, for consistency. As stated above, the output of the TreeHash algorithm is technically the Merkle tree root node. However, to compute the root, each node in the full tree is calculated. By using a different height input, it is not difficult to see how TreeHash can be used to compute any desired tree node.

Further, Leafcalc(j) is simply a sub-routine which calculates the j^{th} leaf node of the Merkle tree by generating the j^{th} OTS key pair, and Stack.push() and Stack.pop() are the typical push and pop operations for a data stack.

TreeHash

Input: Height $h \geq 2$
Output: Root of the Merkle tree
1. **for** $j = 0, \ldots, 2h-1$ **do**
   a) Compute the j^{th} leaf: $Node1 \leftarrow$ Leafcalc(j)
   b) **While** $Node1$ has the same height as the top node on Stack **do**
      i. Pop the top node from the stack: $Node2 \leftarrow$ Stack.pop()
      ii. Compute their parent node: $Node1 \leftarrow H(Node2 \,||\, Node1)$
   c) Push the parent node on the stack: Stack.push$(Node1)$
2. Let R be the single node stored on the stack: $R \leftarrow$ Stack.pop()
3. **Return** R

Algorithm 1: TreeHash

4.1.4.3 Authentication paths

An *authentication path* in a binary hash tree, corresponding to node N, is the ordered collection of the siblings of the nodes on the path starting from N and ending at the root node.