HASH-BASED SIGNATURES

5. THE LEIGHTON-MICALI SIGNATURE SCHEME

Alfred Menezes
cryptography101.ca

Outline

1. Introduction
2. LM-OTS description
3. LMS description
4. HSS
5. XMSS
6. Stateful signature schemes

Introduction

- The Leighton-Micali signature scheme (LMS) is a hash-based signature scheme standardized by the IETF (RFC 8554) in 2019.
- LMS is fully described in RFC 8554.
    - **LM-OTS**: One-time signature scheme
    - **LMS:** LM-OTS + Merkle trees
    - **HSS** (Hierarchical signature scheme): LMS + hypertrees.
- LMS was also adopted as a NIST standard (SP 800-208) in 2020.
    - SP 800-208 points to RFC 8554, and added some parameter sets.
- **XMSS** (eXtendable Merkle Signature Scheme) is another hash-based signature scheme, standardized by the IETF (RFC 8391) in 2018, and by NIST in 2020 (SP 800-28).

LM-OTS parameters

- An n-bit hash function $H : \{0,1\}^* \to \{0,1\}^n$.
  - For concreteness, we'll take $n = 256$ and $H = \text{SHA256}$.
- **Winternitz parameter** w, a divisor of n.
  - SP 800-208 permits $w \in \{1,2,4,8\}$.
- $\ell_1 = n/w$ (number of w-bit blocks in a hash value).
- $\ell_2 = \lfloor \log_2(\ell_1(2^w - 1))/w \rfloor + 1$ (number of w-bit blocks in a checksum).
  - The **checksum** $\text{csum}(h)$ of a hash value $h = (h_0, h_1, \ldots, h_{\ell_1 - 1})$, where each h_i is a w-bit block, is
$$\text{csum}(h) = \sum_{i=0}^{\ell_1 - 1} (2^w - 1 - h_i).$$
- $\ell = \ell_1 + \ell_2$.
  - $\ell = 265, 133, 67, 34$ for $w = 1, 2, 4, 8$ (and $n = 256$).

LM-OTS is essentially the Winternitz-OTS with some modifications.

LM-OTS modifications (1)

- The **public key** is $K = H(y_0, y_1, \ldots, y_{\ell-1})$ instead of $Y = (y_0, y_1, \ldots, y_{\ell-1})$.
- A **single hash function** $H : \{0,1\}^* \to \{0,1\}^n$ is used, and also shared with LMS and HSS.
- A **unique prefix** is appended before hashing. The prefix includes:
  - A randomly-selected 32-byte identifier I of the Merkle tree associated with the LM-OTS instance.
  - A 4-byte number q that identifies the tree's leaf associated with the LM-OTS instance.
  - The index i of the w-bit block h_i within the message hash/checksum, which determines which hash chain to use
  - The position j within a Winternitz hash chain.
  - A two byte constant, 0x8080 (for computing K) or 0x8181 (for hashing a message).

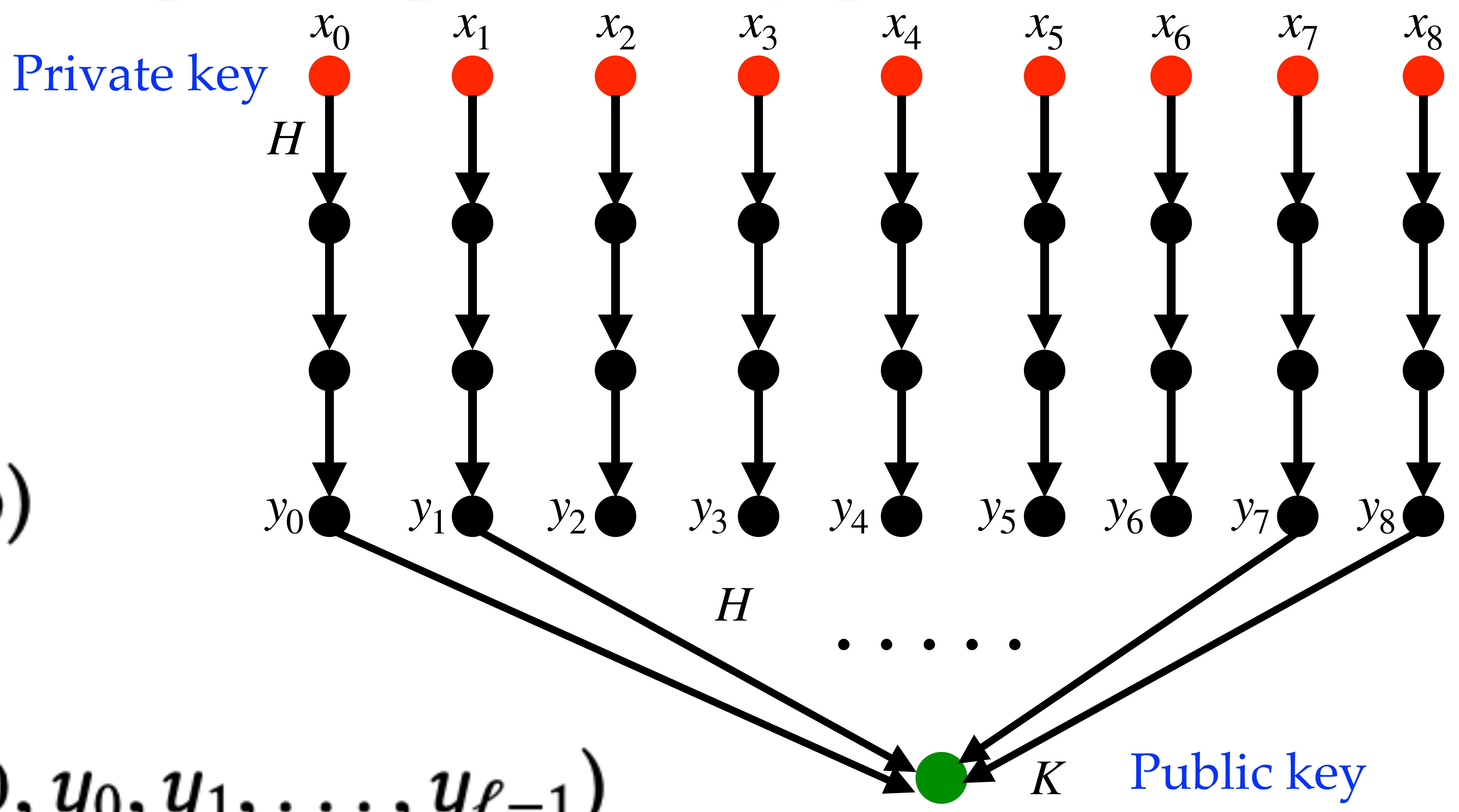
LM-OTS modifications (2)

- A randomly-generated 256-bit string C is appended to the message M before hashing: $h = H(C, M)$.
  - C is called the *message randomizer*.
  - C is included with the signature, since it's needed for signature verification.
- This eliminates the need for collision resistance for H (since C is randomly chosen by the signer each time a message is signed).

LM-OTS key generation

Key generation. Alice generates a public-private key pair as follows:

for $i = 0$ **to** $\ell - 1$ **do**
  Select $x_i \in_R \{0,1\}^n$
  $\mathrm{tmp} \leftarrow x_i$
  for $j = 0$ **to** $2^w - 2$ **do**
    $\mathrm{tmp} \leftarrow H(I, q, i, j, \mathrm{tmp})$
  $y_i \leftarrow \mathrm{tmp}$
Compute $K = H(I, q, \mathrm{0x8080}, y_0, y_1, \ldots, y_{\ell-1})$
Alice's private key is $X = (x_0, x_1, \ldots, x_{\ell-1})$; her public key is (I, q, K)



LM-OTS signature generation

Signature generation. To sign a message $M \in \{0,1\}^*$, Alice does the following:

Select $C \in_R \{0,1\}^n$.

Compute $h = H(I, q, \texttt{0x8181}, C, M)$; the w-bit blocks of h are $(h_0, h_1, \ldots, h_{\ell_1 - 1})$.

Compute $\mathrm{csum}(h)$ and obtain the w-bit blocks $(h_{\ell_1}, h_{\ell_1+1}, \ldots, h_{\ell-1})$ of $\tilde{c}(h)$.

for $i = 0$ **to** $\ell - 1$ **do**

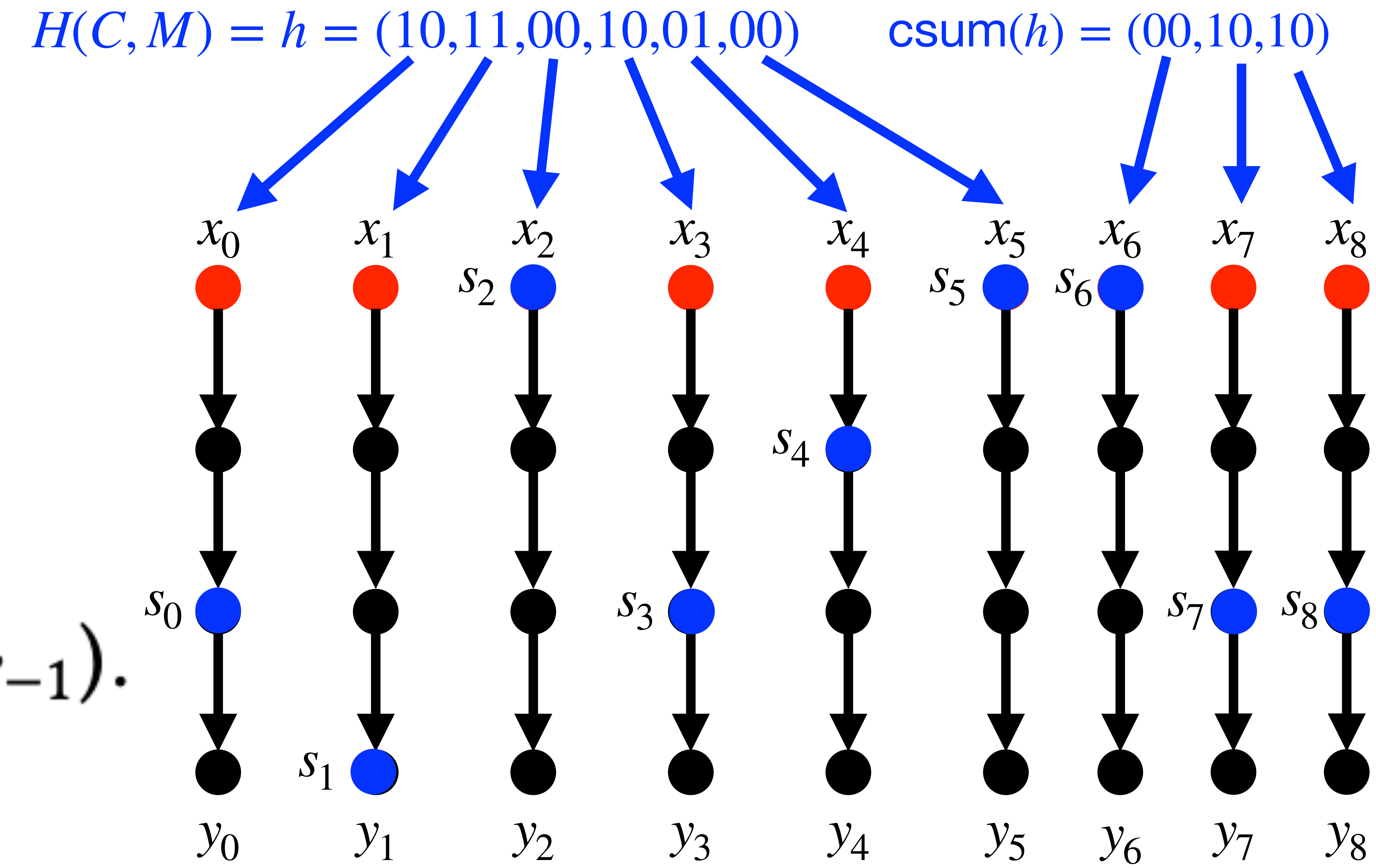
  $\mathrm{tmp} \leftarrow x_i$

  for $j = 0$ **to** $h_i - 1$ **do**

    $\mathrm{tmp} \leftarrow H(I, q, i, j, \mathrm{tmp})$

  $s_i \leftarrow \mathrm{tmp}$

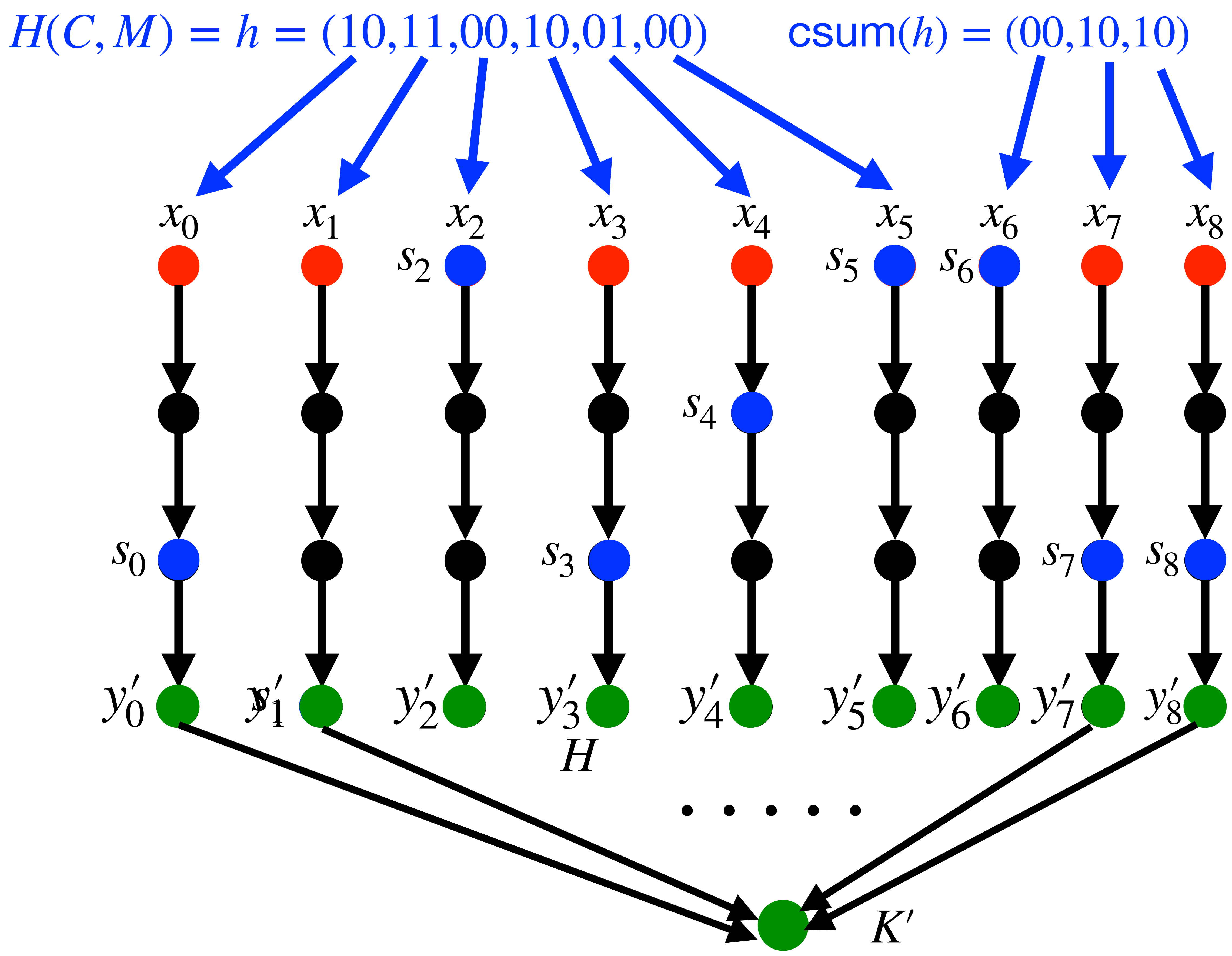
Alice's signature on M is $(C, s_0, s_1, \ldots, s_{\ell-1})$.



LM-OTS signature verification

Signature verification. To verify Alice's signature $(C, s_0, \ldots, s_{\ell-1})$ on M, Bob does:

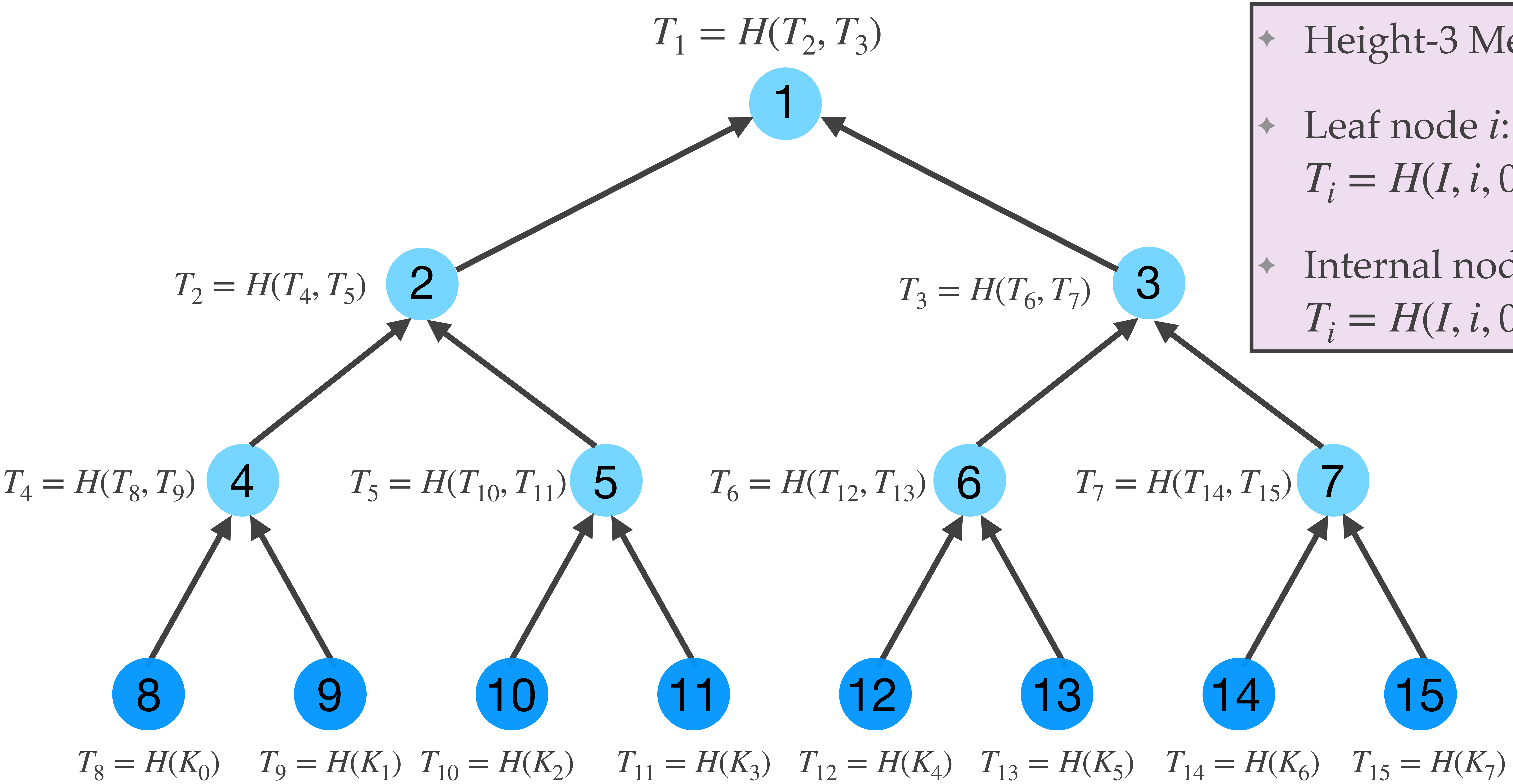
Compute $h = H(I, q, 0x8181, C, M)$; the w-bit blocks of h are $(h_0, h_1, \ldots, h_{\ell_1 - 1})$.
Compute $\mathrm{csum}(h)$ and obtain the w-bit blocks $(h_{\ell_1}, h_{\ell_1+1}, \ldots, h_{\ell-1})$ of $\tilde{c}(h)$.
for $i = 0$ **to** $\ell - 1$ **do**
    $\mathrm{tmp} \leftarrow s_i$
    for $j = h_i$ **to** $2^w - 2$ **do**
        $\mathrm{tmp} \leftarrow H(I, q, i, j, \mathrm{tmp})$
    $y'_i \leftarrow \mathrm{tmp}$
Compute $K' = H(I, q, 0x8080, y'_0, y'_1, \ldots, y'_{\ell-1})$.
if $K' = K$ **then**
    return Accept
else
    return Reject



LMS

- LMS uses a height-d Merkle tree, whose leaves are associated with LM-OTS public keys $K_0, K_1, \ldots, K_{2^d-1}$.
- SP 800-208 permits heights $d \in \{5,10,15,20,25\}$.
- Private keys are generated from a 256-bit random secret SEED:
  - The ith component of private key associated with a leaf numbered q is $H(I, q, i, \text{0xff}, \text{SEED})$.

LMS tree



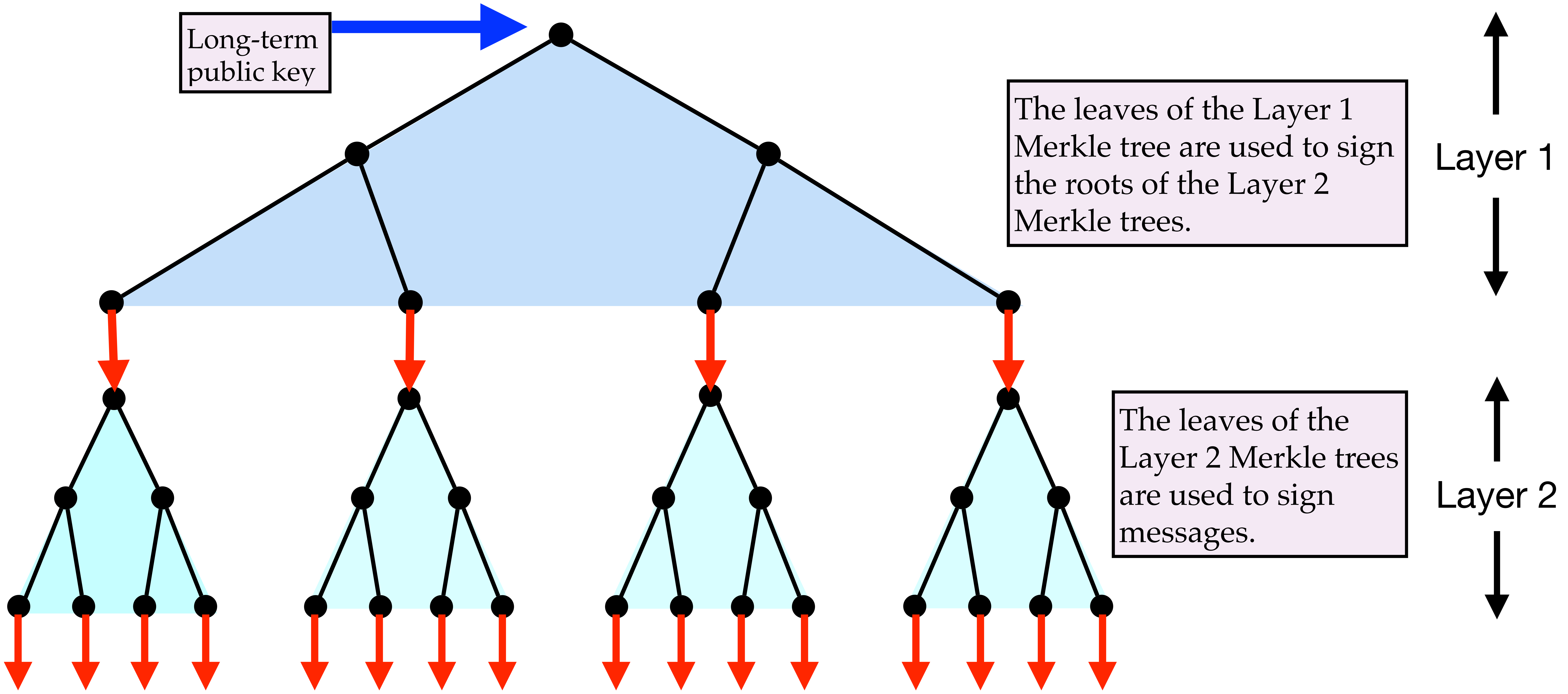
- Height-3 Merkle tree.
- Leaf node i:
  $T_i = H(I, i, 0x8282, K_{i-2^d})$
- Internal node i:
  $T_i = H(I, i, 0x8383, T_{2i}, T_{2i+1})$

LMS security

- (Katz, 2016) LMS has been proven secure in the random oracle model, wherein the hash function H is treated as a public, randomly-selected function.
- (Eaton, 2018) LMS has also been proven secure in the quantum random oracle model, wherein adversaries can query H in superposition.

HSS

- **Hierarchical Signature Scheme (HSS)** is the hypertree version of LMS.
- SP 800-208 permits hypertrees with as many as 8 layers.



XMSS

- **XMSS** closely resembles LMS.
- The main difference is its use of "**bitmasks**", which are pseudorandom bit strings that are XORed with inputs to the hash function.
- The bitmasks enable security proofs for XMSS in the "standard model", i.e., without needing to model the hash function as a random oracle.
- The hypertree version of XMSS is called **XMSS**^{MT}.
- XMSS and XMSS^{MT} are fully specified in IETF RFC 8391, and additional parameter sets are defined in NIST's SP 800-208.

Stateful signature schemes

- LMS and XMSS are **stateful** signature schemes.
- A signer must be very careful not to reuse any of their OTS private keys associated with their long-term public key.
- To enforce this requirement, the signer could keep track of a counter or key index that increments after each message is signed.
    - This type of data that changes over time is called **state**.
- However, maintaining state securely in practice can be challenging.
    - e.g., if a system crashes and is restored to an earlier state.

