HASH-BASED SIGNATURES

4. LAMPORT: PROBLEMS AND SOLUTIONS

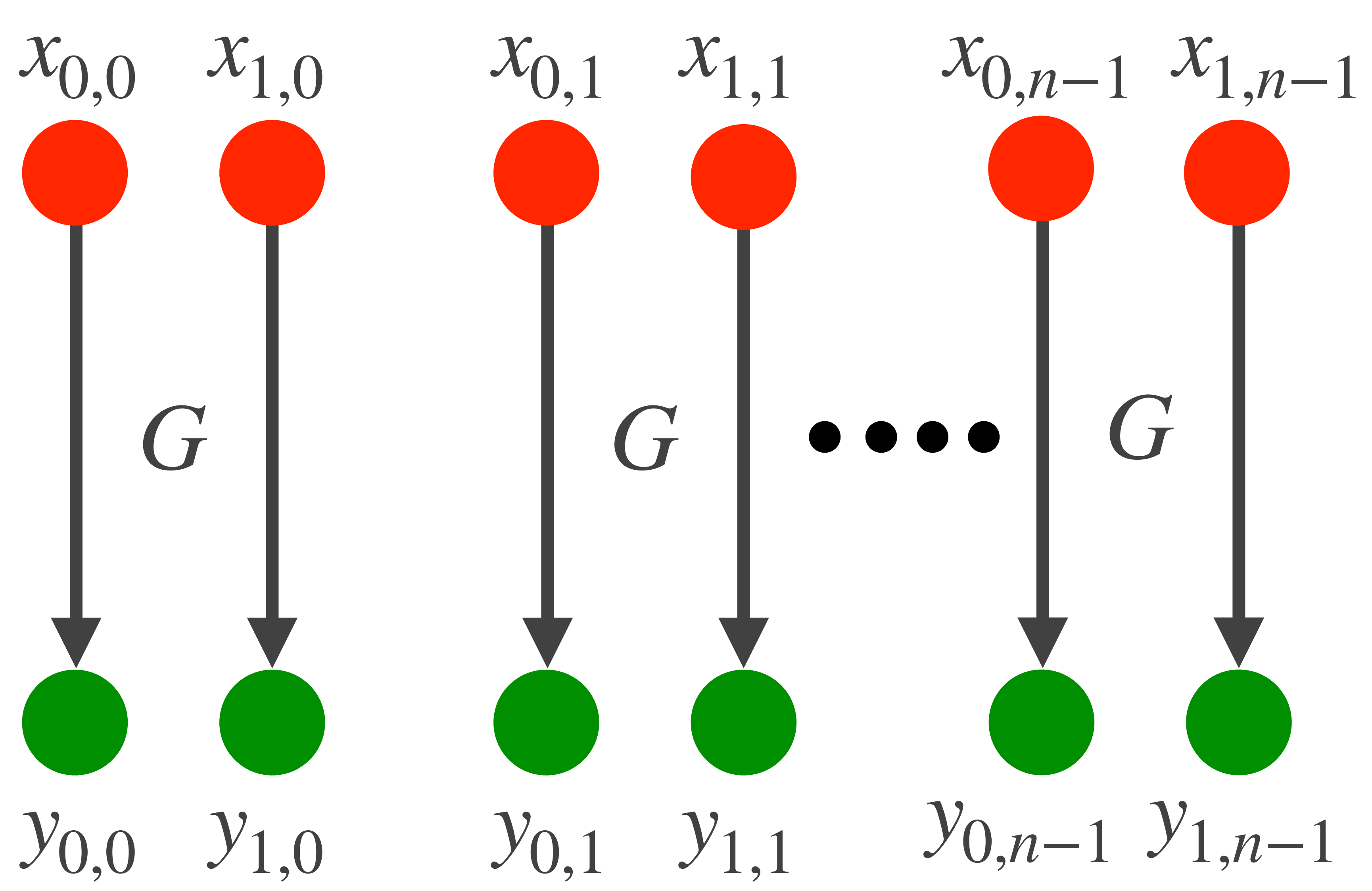
Alfred Menezes
cryptography101.ca

Outline

1. Lamport drawbacks: Large public keys, private keys, and signatures
2. Winternitz OTS
3. Merkle trees
4. Hypertrees

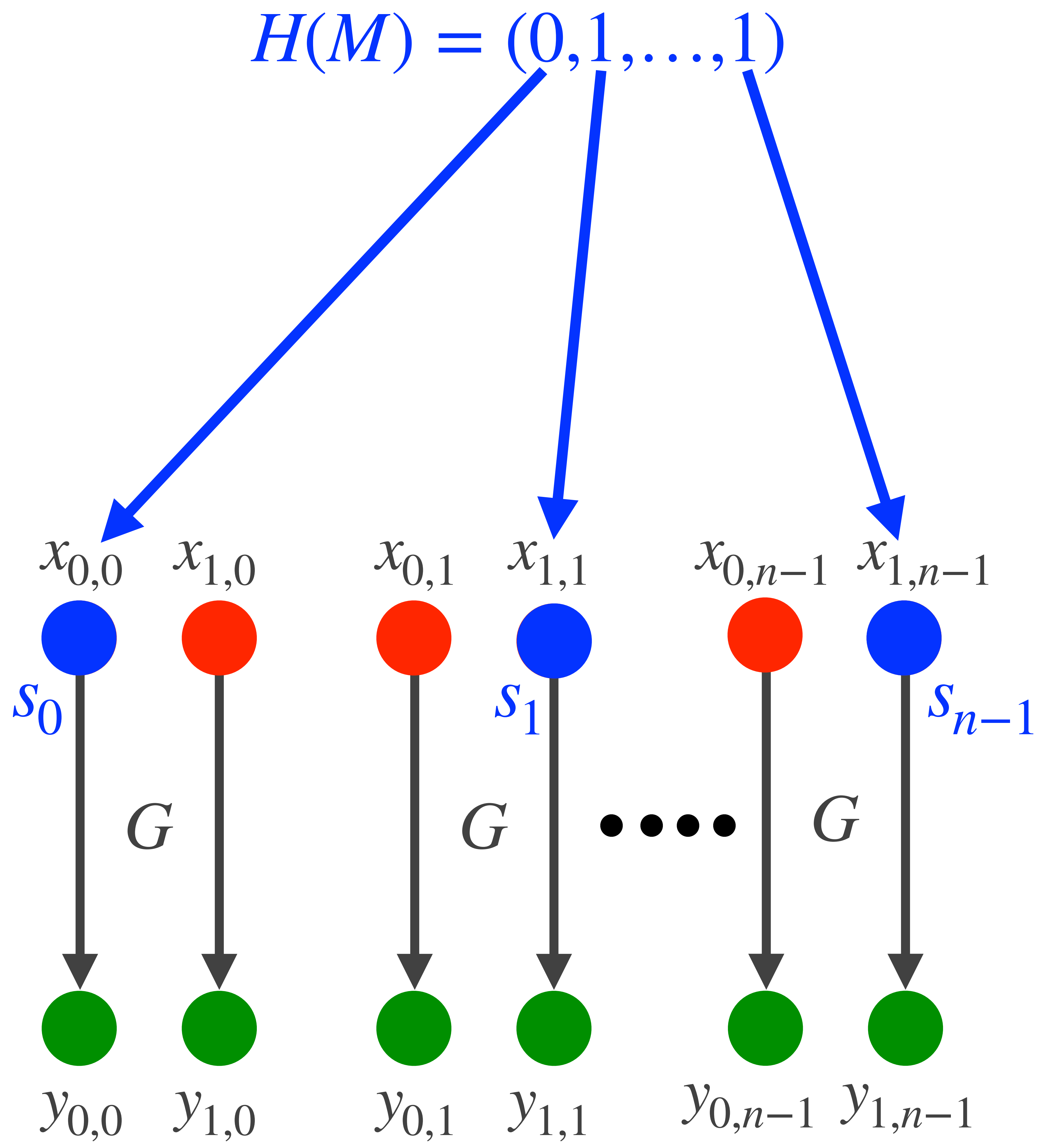
Lamport OTS: key generation

- $G : \{0,1\}^n \to \{0,1\}^n$ is a PR hash function.
- $H : \{0,1\}^* \to \{0,1\}^n$ is a CR hash function.
- **Key generation**: Alice does the following:
  - Select $x_{i,j} \in_R \{0,1\}^n$ for $i = 0,1$, $j = 0,1,\ldots,n-1$
  - Compute $y_{i,j} = G(x_{i,j})$.
  - Alice's **private key** is $X = (x_{i,j})$.
  - Alice's **public key** is $Y = (y_{i,j})$.



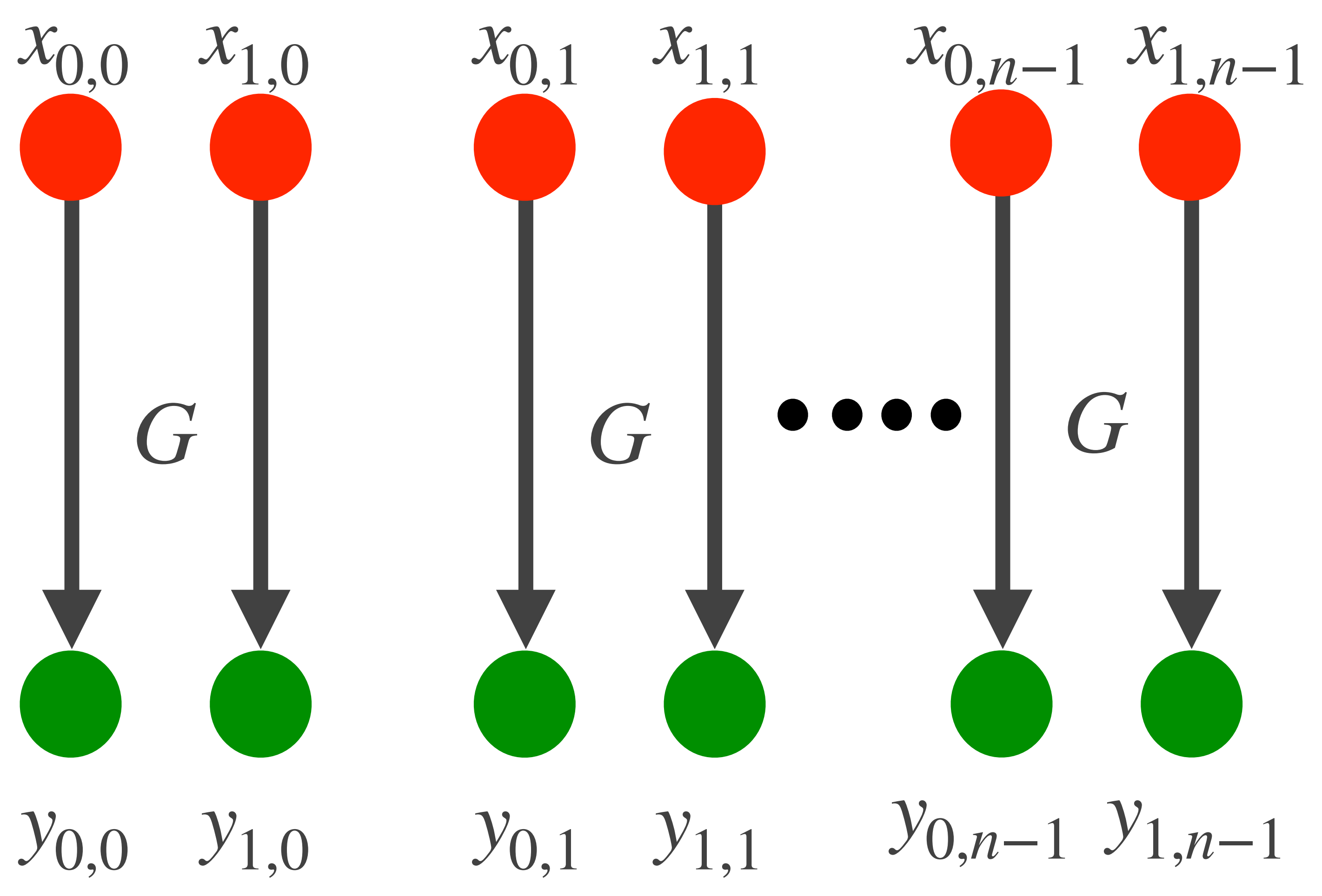
Lamport OTS: signature generation/verification

- **Signature generation**: To sign $M \in \{0,1\}^*$, Alice computes $h = H(M) = (h_0, h_1, \ldots, h_{n-1})$. Her signature on M is $S = (s_0, s_1, \ldots, s_{n-1})$, where $s_j = x_{h_j,j}$.
- **Signature verification**: To verify (M, S), anyone who possesses an authentic copy of Y can compute $h = H(M)$ and check that $y_{h_j,j} = G(s_j)$ for $j = 0,1,\ldots,n-1$.



Problem #1: Large private keys

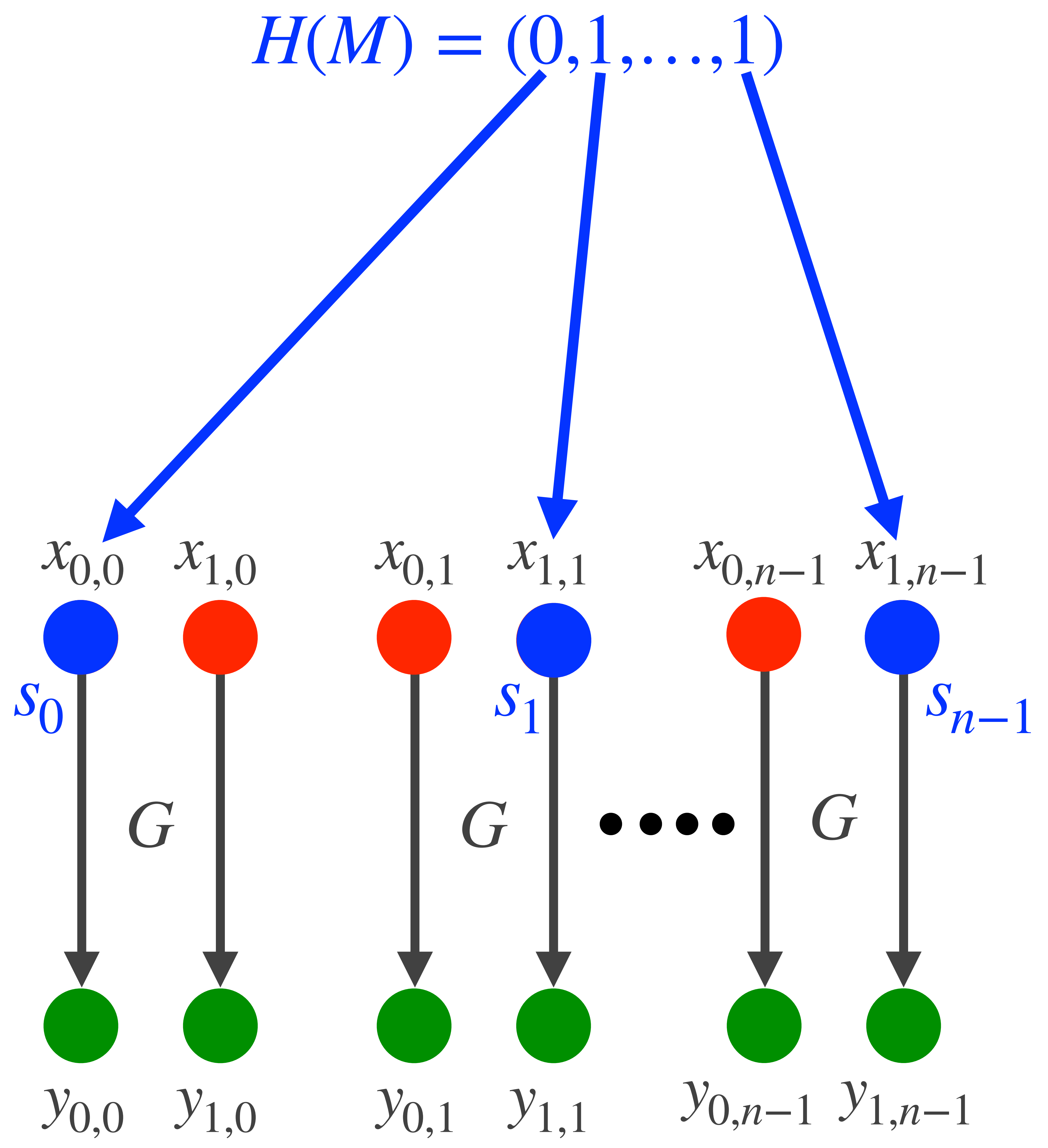
- Private keys: $2n^2$ bits, Public keys: $2n^2$ bits, Signature size: n^2 bits.
- For $n = 256$, these are 16,384, 16,384, and 8,192 bytes.
- **Solution**: The private key components $x_{i,j}$ are generated from a randomly-generated n-bit seed and a counter using a pseudorandom function: $x_{i,j} = \text{prf}(\text{SEED}, i, j)$.



- The seed is securely stored, and used to generate the $x_{i,j}$ when needed.
- The private key size is reduced to n bits, or 32 bytes when $n = 256$.

Problem #2: Large public keys and signatures

- Public keys are $2n^2$ bits in size, signatures are n^2 bits in size.
- The large sizes are because the bits of $H(M)$ are signed one bit at a time.
- **Solution**: (Winternitz, 1979) Sign the bits of $H(M)$, one *w-bit block* at a time using *hash chains*.



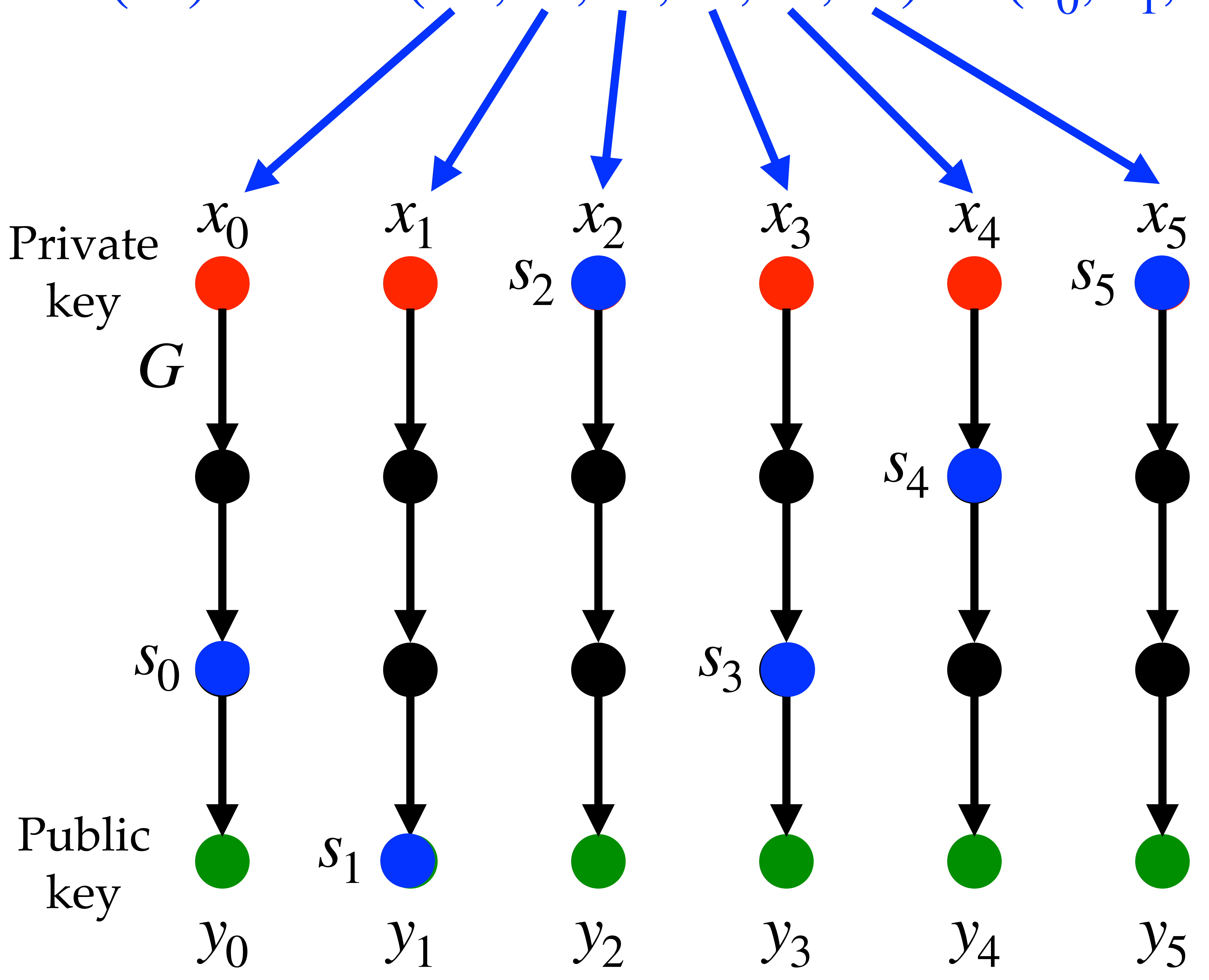
Hash chains

- $G : \{0,1\}^n \to \{0,1\}^n$ is a preimage-resistant hash function.
- Let $x_0 \in_R \{0,1\}^n$.
- Define $x_i = G(x_{i-1})$ for $i \geq 1$, so $x_i = G^i(x_0)$.
- The sequence $x_0, x_1, x_2, \ldots, x_m$ is called a **hash chain of length** m.

- Given any x_i, it's easy to compute x_j for all $j > i$.
- However, computing x_j for any $j < i$ is infeasible since G is preimage resistant.

Winternitz OTS: Toy example (1)

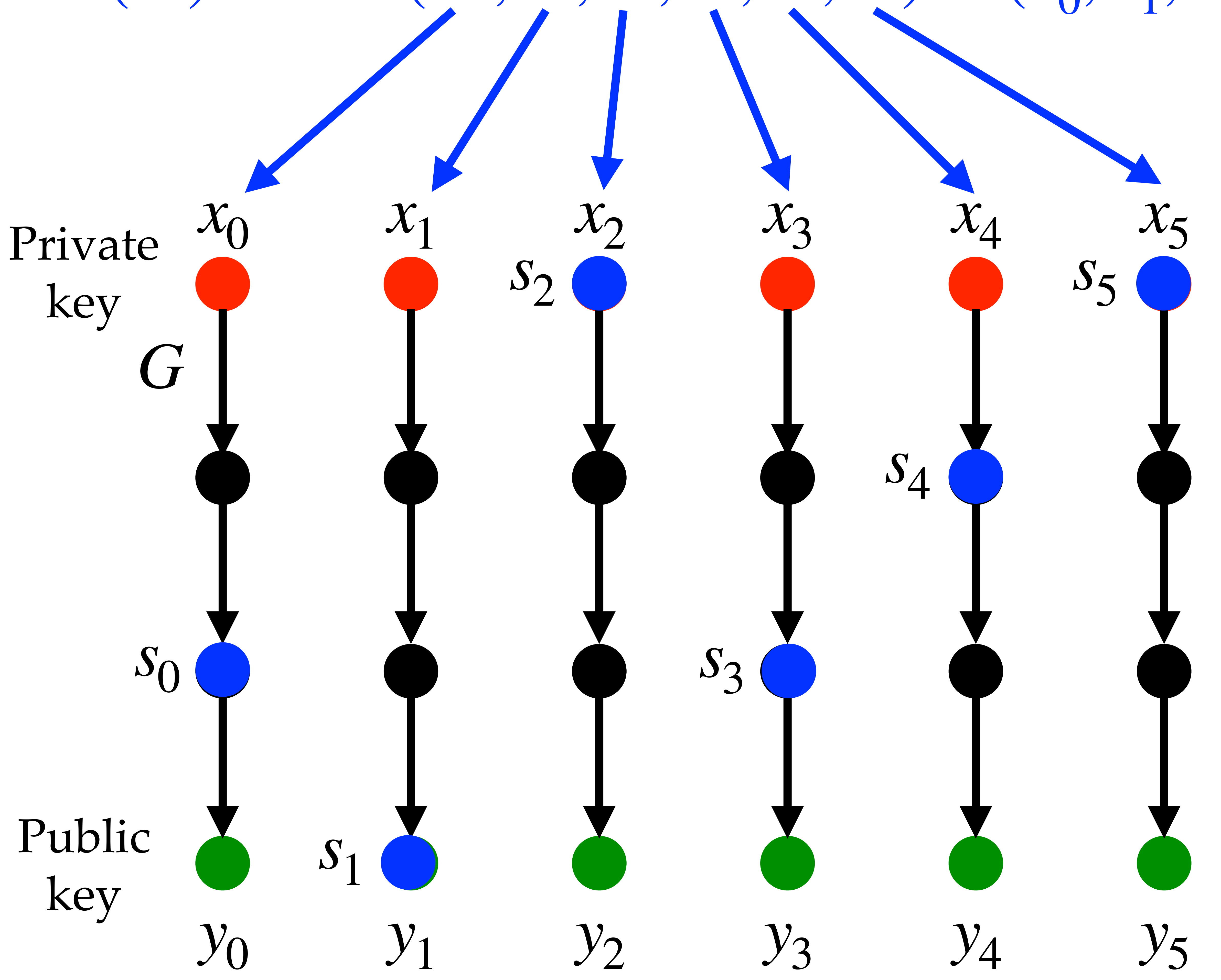
$H(M) = h = (10,11,00,10,01,00) = (h_0, h_1, h_2, h_3, h_4, h_5) = (2,3,0,2,1,0)$



- $G : \{0,1\}^{256} \to \{0,1\}^{256}$.
- $H = \{0,1\}^* \to \{0,1\}^{12}$ (so $n = 12$).
- **Winternitz parameter** is $w = 2$.
- **Hash chain length** is $2^w - 1 = 3$.
- **Private key**: $X = (x_0, x_1, \ldots, x_5)$.
- **Public key**: $Y = (y_0, y_1, \ldots, y_5)$.
- **Signature**: $S = (s_0, s_1, \ldots, s_5)$.
- **Verification**: Compute $H(M)$ and check that $G(s_0) = y_0$, $s_1 = y_1$, $G^3(s_2) = y_2$, $G(s_3) = y_3$, $G^2(s_4) = y_4$, $G^3(s_5) = y_5$.

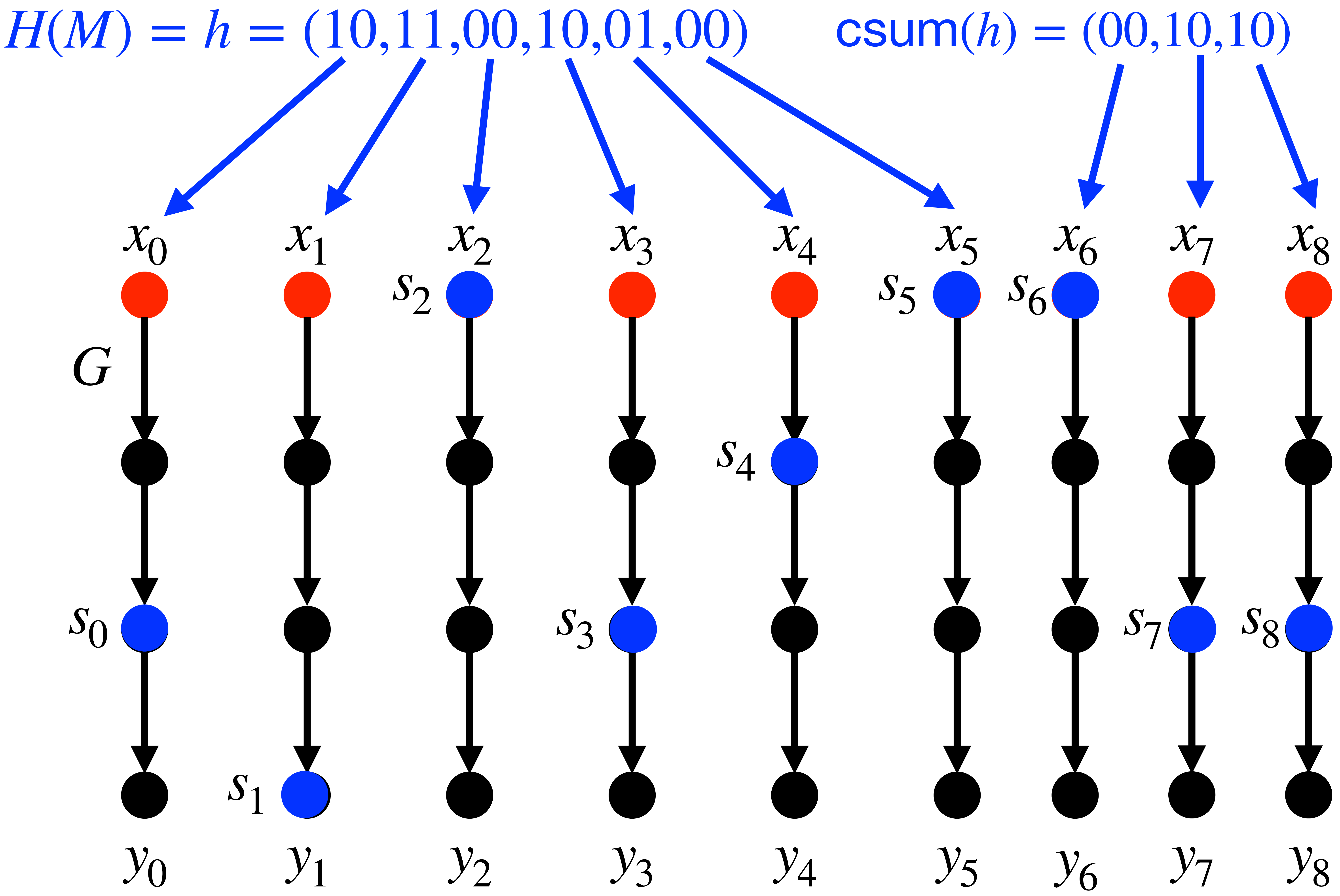
Winternitz OTS: Toy example (2)

$H(M) = h = (10,11,00,10,01,00) = (h_0, h_1, h_2, h_3, h_4, h_5) = (2,3,0,2,1,0)$



- Given only the public key Y, the adversary is only able to forge signatures on messages with $H(M) = (11,11,11,11,11,11)$.
- A signed message (M, S) reveals about half of each hash chain. So, the private key should only be used once (OTS).
- **Security concern:** Given (M, S), a forger can compute the signature on *all* messages M^* for which $h_i^* \geq h_i$ for all $0 \leq i \leq 5$.

Winternitz OTS: Toy example (3)



- **Solution**: Add hash chains for a checksum:

$$\text{csum}(h) = \sum_{i=0}^{\ell_1 - 1} (2^w - 1 - h_i).$$

- Let M^* be a message for which $h_i^* \geq h_i$ for all $i \in [0,5]$ and $h_i^* > h_i$ for at least one $i \in [0,5]$.
- Then $\text{csum}(h^*) < \text{csum}(h)$, whence $h_i^* < h_i$ for at least one index $i \in \{6,7,8\}$. Thus, the forger cannot produce the valid signature for M^*.

Winternitz OTS

- $G : \{0,1\}^n \to \{0,1\}^n$ is preimage resistant, $H : \{0,1\}^* \to \{0,1\}^n$ is collision resistant.
- **Parameters**: w (divisor of n), $\ell_1 = n/w$ (# of w-bit blocks in h), $\ell_2 = \lfloor \log_2(\ell_1(2^w - 1))/w \rfloor + 1$ (# of w-bit blocks in csum(h)), $\ell = \ell_1 + \ell_2$.
- **Key generation**: Alice selects $x_i \in_R \{0,1\}^n$ and computes $y_i = G^{2^w-1}(x_i)$, $0 \le i \le \ell - 1$. Her **private key** is $X = (x_0, \ldots, x_{\ell-1})$; **public key** is $Y = (y_0, \ldots, y_{\ell-1})$.
- **Signature generation**: Compute $h = H(M) = (h_0, \ldots, h_{\ell_1-1})$ and csum(h) $= (h_{\ell_1}, \ldots, h_{\ell-1})$, where each h_i is a w-bit block.
  Alice's signature on M is $S = (s_0, \ldots, s_{\ell-1})$ where $s_i = G^{h_i}(x_i)$ for $0 \le i \le \ell - 1$.
- **Signature verification**: Compute $h = H(M) = (h_0, \ldots, h_{\ell_1-1})$ and csum(h) $= (h_{\ell_1}, \ldots, h_{\ell-1})$, and verify that $G^{2^w-1-h_i}(s_i) = y_i$ for $0 \le i \le \ell - 1$.

Winternitz OTS: Public key and signature sizes

- Suppose that $n = 256$ and $w = 8$ (so a hash chain has length 255).
- Then $\ell_1 = 32$, $\ell_2 = 2$, and $\ell = \ell_1 + \ell_2 = 34$.
- Public key size: 1,088 bytes (vs. 16,384 bytes for Lamport).
- Signature size: 1,088 bytes (vs. 8,192 bytes for Lamport).
- Private key size: 32 bytes (using a secret seed and prf).

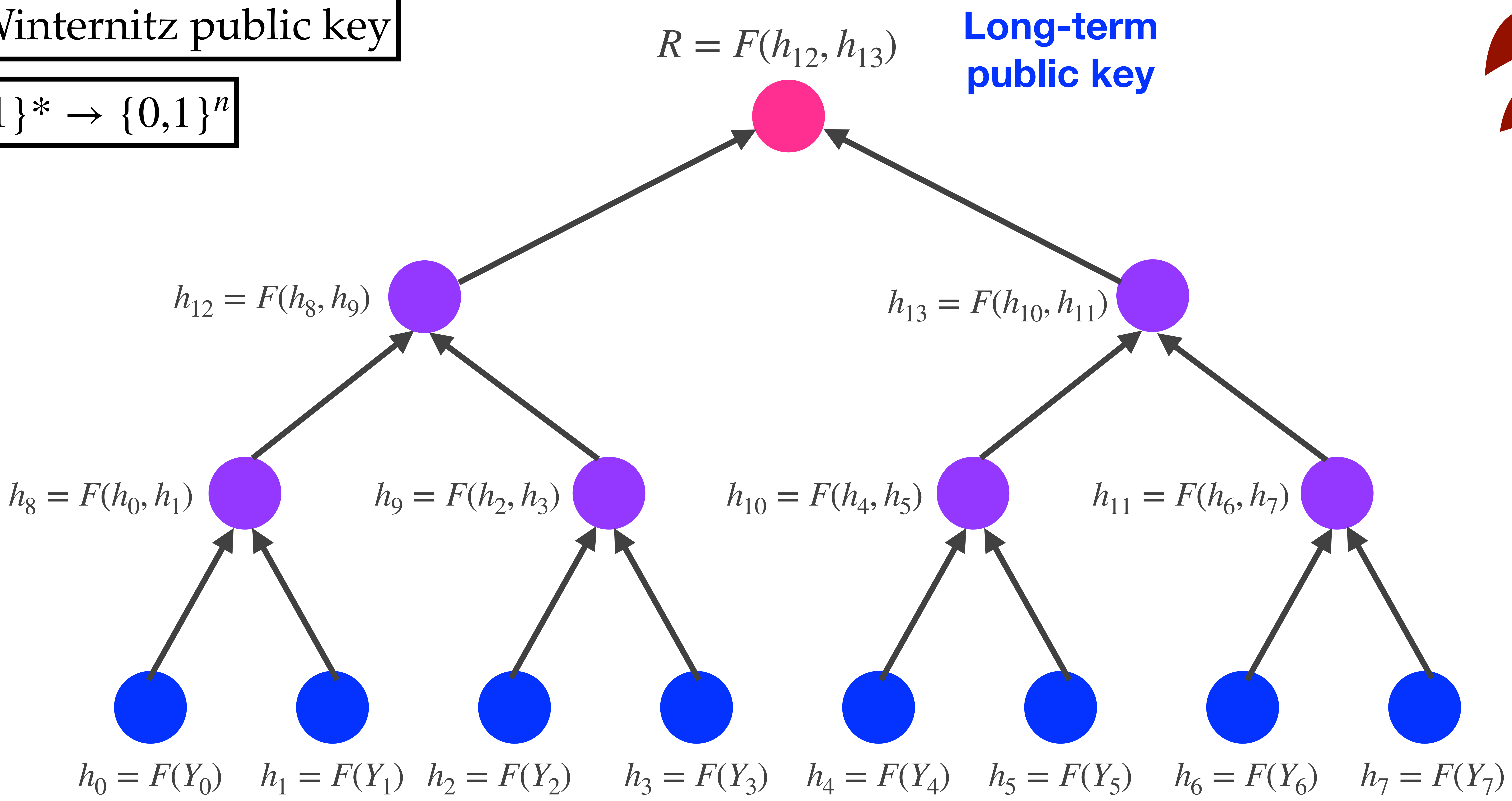
Problem #3: A private key can only be used once

- **Solution:** Generate many Winternitz OTS key pairs, and use them one at a time.
- **Problem**: Need a large number of OTS keys if you expect to sign a large number of messages. The total size of all the OTS public keys is very large.
- **Solution**: Use a **Merkle tree** to authenticate the OTS public keys.
- **Main idea**: Build a complete binary tree whose leaves are assigned the hashes of the OTS public keys. Each internal node's value is obtained by hashing the concatenation of the values of its two children. The user's <u>long-term public key</u> is the value assigned to the root of the tree. <u>Authentication paths</u> to the root are used to confirm the authenticity of an individual OTS public key.

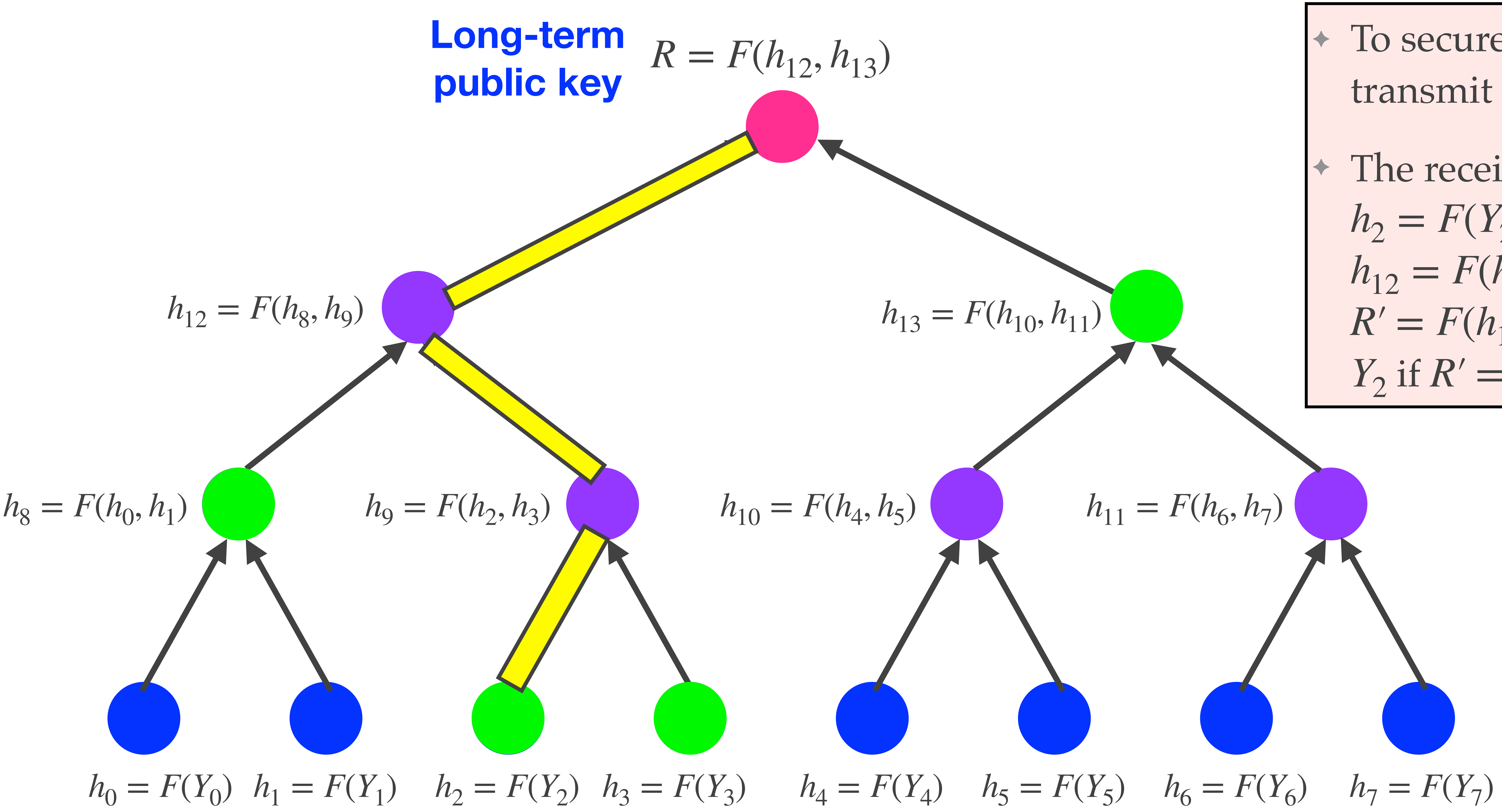
Merkle trees

$Y_i =$ Winternitz public key

$F : \{0,1\}^* \to \{0,1\}^n$



Merkle trees: Authentication paths



- To securely send Y_2, transmit Y_2, h_3, h_8, h_{13}.
- The receiver computes $h_2 = F(Y_2)$, $h_9 = F(h_2, h_3)$, $h_{12} = F(h_8, h_9)$, and $R' = F(h_{12}, h_{13})$, and accepts Y_2 if $R' = R$.

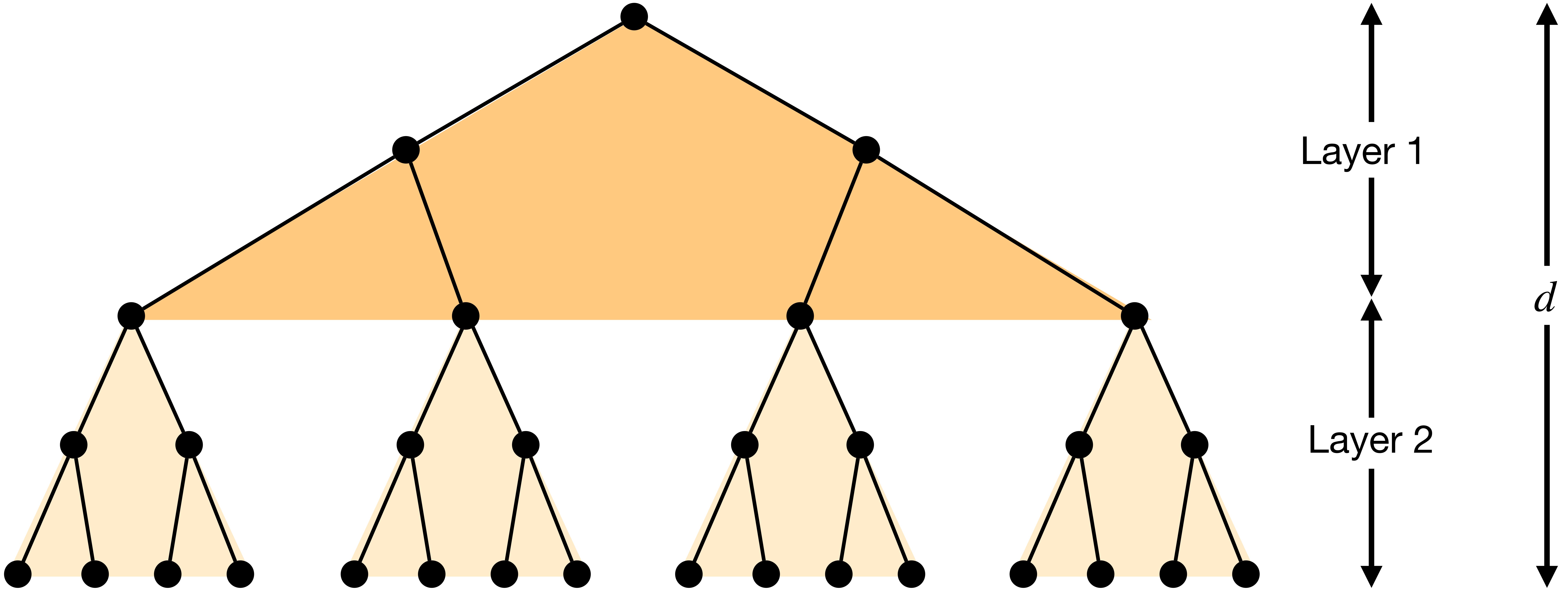
Example: Winternitz OTS + Merle trees

- **Winternitz parameters**: $n = 256, w = 8, \ell = 34$.
- **Merkle tree**: Height 20 (so 2^{20} Winternitz OTS key pairs).
- **Long-term public key size**: 32 bytes.
- **Private key size**: 32 bytes (using a secret seed and prf).
- **Signature size**: 1,088 + 1,088 (Y_i) + 20 × 32 (auth. path) = 2,816 bytes.

Problem #4: Constructing a large Merkle tree is slow

- To support a large number of signatures, the Merkle tree must have large height d.
- For example, to support 2^{40} signatures, the Merkle has height 40.
- However, key generation is very slow since all 2^d OTS key pairs need to be generated in order to determine the root R (the long-term public key) of the tree.
- **Solution**: The Merkle tree is divided into smaller trees to form a *hypertree*, also called a *multi-tree*.

Hypertrees (1)



Of course, more than two layers can be used.

Hypertrees (2)

The leaves of the Layer 1 Merkle tree are used to sign the roots of the Layer 2 Merkle trees.

The leaves of the Layer 2 Merkle trees are used to sign messages.

Using a hypertree

LMS signature scheme

- The **Leighton-Micali signature scheme (LMS)** uses the Winternitz OTS with Merkle trees (and hypertrees), and some additional optimizations to enhance security.
- LMS will be presented in video V5.

