Lamport signatures: Let $f: \mathcal{X} \to \mathcal{Y}$ be a one-way function.

length of message ($m = \{0,1\}^n$)

- Setup$(1^\lambda, 1^n)$: Sample $x_{i,b} \xleftarrow{R} \mathcal{X}$ $\forall i \in [n], b \in \{0,1\}$ and compute $y_{i,b} \leftarrow f(x_{i,b})$ $\forall i \in [2n]$, $b \in \{0,1\}$
  Set

$sk =$

$x_{1,0}$	$x_{2,0}$	$\cdots$	$x_{n,0}$
$x_{1,1}$	$x_{2,1}$	$\cdots$	$x_{n,1}$

$pk =$

$y_{1,0}$	$y_{2,0}$	$\cdots$	$y_{n,0}$
$y_{1,1}$	$y_{2,1}$	$\cdots$	$y_{n,1}$

- Sign(sk, m) [$m \in \{0,1\}^n$]: Output $(x_{1,m_1}, \ldots, x_{n,m_n})$
- Verify(vk, m, σ): Output 1 if $\forall i \in [n]$, $f(x_{i,m_i}) = y_{i,m_i}$ and 0 otherwise

Theorem. If f is one-way, then Lamport signatures are secure one-time signatures (i.e., where adversary can only make 1 signing query).

Proof. Suppose A is a one-time signature adversary. We construct B for f as follows:

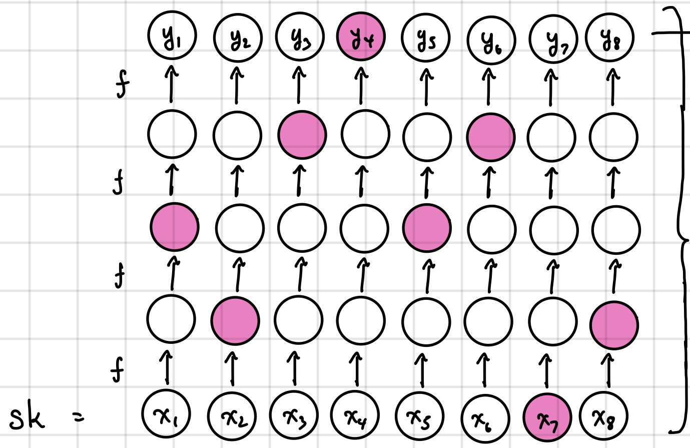
1. Algorithm A receives challenge $y = f(x)$ where $x \xleftarrow{R} \mathcal{X}$ from challenger.
2. Choose $i^* \xleftarrow{R} [n]$, $b^* \xleftarrow{R} \{0,1\}$ to program challenge. Sample $x_{i,b} \xleftarrow{R} \mathcal{X}$, $y_{i,b} \leftarrow f(x_{i,b})$ for $(i,b) \neq (i^*, b^*)$.
   Set $pk = (y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$.
3. Send pk to B. If B makes signing query on $m = m_1, \ldots, m_n$:
   - If $m_{i^*} = b^*$, then abort.
   - Otherwise, reply with $(x_{1,m_1}, \ldots, x_{n,m_n})$.
4. After A outputs a forgery (m^*, σ^*), if $m^*_{i^*} \neq b^*$, then abort. Otherwise, output σ_{i^*}.

By construction, $m^*_{i^*} = b^*$ with probability $1/2n$. Thus if A succeed with probability ε, B succeeds with prob. $\varepsilon/2n$.

Limitations: One-time only [will fix later!]
two signatures allow recovering secret key!

Long public keys, secret keys, and signatures
- Compose with CRHF to get poly(λ)-size parameters (independent of message length)
- Secret key can be derived from PRG (e.g., just λ bits)
- Public key can also be shortened to 2λ bits (special case of Winternitz construction below)

Many combinatoric tricks to reduce signature size:
- Winternitz signatures: use an iterated one-way function $(f: \mathcal{X} \to \mathcal{X}$, $f^{(d)} = \underbrace{f(f(\cdots f(x)\cdots))}_{d \text{ copies}})$



depth d

$pk = H(y_1, \ldots, y_8)$ — public key is single hash!

key length

$P(m)$ maps onto $[d]^n$
↳ give out values corresponding to shaded nodes

we say that $P(m) \leq P(m')$ if each component of $P(m)$ is smaller than $P(m')$
$\Rightarrow$ signature on m can be used to obtain signature on m'
$\Rightarrow$ for security, just need a function P where $P(m) \not\leq P(m')$

Constructing $P(m)$:

Idea: if $m < m'$, then $-m > -m'$.

- View $m \in \{0,1\}^t$ as a number in base d: $s_1, \dots, s_\ell$ $\quad (\ell \sim t/\log d)$
- Compute $d\ell - (s_1 + \cdots + s_\ell)$ and write this in base d: $t_1, \dots, t_{\ell'}$ $\quad (\ell' \sim \log_d d\ell)$
- Output $(s_1, \dots, s_\ell, t_1, \dots, t_{\ell'})$

Suppose $P(m) \le P(m')$ for some $m \neq m'$. This means that $s_1 \le s_1', \dots, s_\ell \le s_\ell'$ (and at least 1 strict). Then, $(s_1 + \cdots + s_\ell) < (s_1' + \cdots + s_\ell')$. Thus, $d\ell - (s_1 + \cdots + s_\ell) > d\ell - (s_1' + \cdots + s_\ell')$ so there is at least one t_i where $t_i > t_i'$, which is a contradiction.

Benefit of Winternitz construction: if messages are $O(\lambda)$ bits and $\log |\mathcal{X}| = O(\lambda)$ bits, then

- Lamport signatures: $|pk| = O(\lambda^2)$ $\quad |\sigma| = O(\lambda^2)$
- Winternitz: $|pk| = O(\lambda)$ $\quad |\sigma| = O(\lambda^2 / \log d)$

Very significant in practice!

using CRHF as OWF

Lamport signatures (with $\lambda = 256$): $|pk| = 16$ KB, $|\sigma| = 8$ KB

Winternitz $(d=2)$: $|pk| = 32$ bytes, $|\sigma| \approx 8.5$ KB

$(d=16)$: $|\sigma| \approx 2.1$ KB

$(d=1024)$: $|\sigma| \approx 0.9$ KB

verification needs more hash evaluations (very fast!)

One-time signatures are very fast (only needs <u>symmetric cryptography</u>)

- Very useful in streaming setting: each packet in stream should be signed, but expensive to do so
  - Instead: include pk for one-time signature in first packet
    sign first packet using standard signature algorithm (public key)
    each packet includes OTS public key for next packet:

$$(m_0, vk_1), \sigma \rightarrow (m_1, vk_2), \sigma_1 \rightarrow (m_2, vk_3), \sigma_2, \dots$$

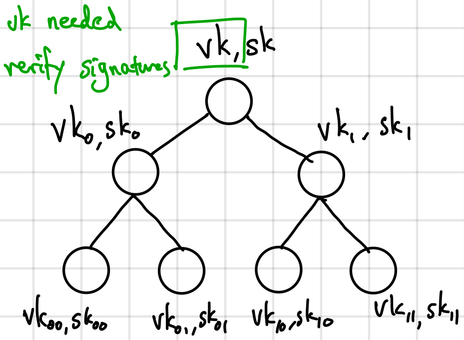
σ: signed using many-time signature; σ_1: signed using secret key for vk_1; σ_2: signed using secret key for vk_2

chaining signatures

Stateful many-time signatures from one-time signatures:

<u>Idea</u>: use a tree of one-time signatures:

only vk needed to verify signatures!



- every node is associated with a key-pair for an OTS scheme
- each signing key used to sign verification keys of its children
- signing key for leaf nodes used to sign messages
- each leaf can only be used to sign <u>one</u> message — need to keep track of which nodes have been used (<u>stateful</u> signature)

<u>Example</u>: Signing message m using (vk_{00}, sk_{00}):

- $\sigma_0 \leftarrow \text{Sign}(sk, vk_0 \| vk_1)$
- $\sigma_{00} \leftarrow \text{Sign}(sk_0, vk_{00} \| vk_{01})$
- $\sigma_m \leftarrow \text{Sign}(sk_{00}, m)$
- Output $(vk_0 \| vk_1, vk_{00} \| vk_{01}, \sigma_0, \sigma_{00}, \sigma_m)$

To verify, check

$\text{Verify}(vk, vk_0 \| vk_1, \sigma_0) = 1$

$\text{Verify}(vk_0, vk_{00} \| vk_{01}, \sigma_{00}) = 1$

$\text{Verify}(vk_{00}, m) = 1$

Only root vk needed here, all other keys included in σ

Security (Intuition): - Keys for internal nodes only used to sign <u>single</u> message (verification keys of children)

- As long as leaf node never reused, then leaves are also only used once
- Security now reduces to one-time security of signature scheme

How to remove state?

- Consider a tree with 2^λ leaves and choose leaf at random for signing
- If we sign $\text{poly}(\lambda)$ messages, there will not be a collision in the leaf with $1-\text{negl}(\lambda)$ probability
- Problem: Signing key is exponential (need to store $O(2^\lambda)$ signing keys)

<u>Solution</u>: Derive signing keys from a PRF!

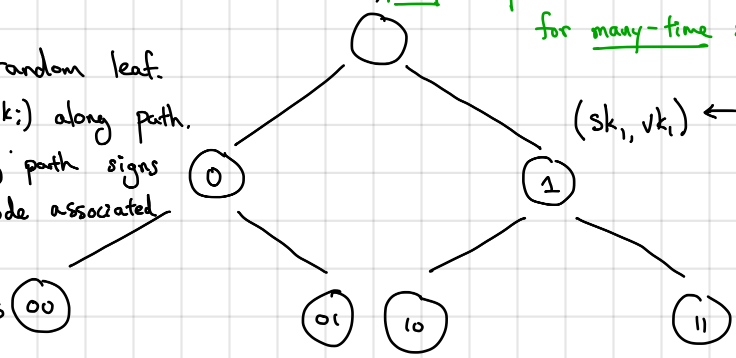
$$(vk_i, sk_i) \leftarrow \text{KeyGen}(1^\lambda ; \text{PRF}(k, i))$$

randomness to key-generation algorithm

part of signing key

node index

$sk, \boxed{vk}$ — public vk for <u>many-time</u> signature



$(sk_1, vk_1) \leftarrow \text{KeyGen}(1^\lambda ; \text{PRF}(k, 1))$

To sign, choose random leaf.
Derive all (sk_i, vk_i) along path.
Each node along path signs verification node associated with children.
Leaf node signs message.
Signature contains complete validation path from root to leaf and signature of leaf on message.
Every internal node still signs only one message.

$(sk_{10}, vk_{10}) \leftarrow \text{KeyGen}(1^\lambda ; \text{PRF}(k, 10))$