SPHINCS-alpha

Kaiyi Zhang, Hongrui Cui, Yu Yu

Shanghai Jiao Tong University

2nd Oxford PQ Cryptography Summit

SPHINCS-alpha

- Hash-based Signature
- Improved SPHINCS+
  - Improve One-Time Signature
  - Retune parameters

- The signature size and signing time are slightly better, but verification time increased.

	SPHINCS$^+$				SPHINCS-α				Relative Change			
Param.	KeyGen	Sign	Verify	Size	KeyGen	Sign	Verify	Size	KeyGen	Sign	Verify	Size
128f	1143558	26872236	2204802	17088	1036602	26635716	2028186	16720	−9.35%	−0.88%	−8.01%	−2.15%
192f	1662498	45405504	3003534	35664	2199276	45218790	1744038	34896	32.29%	−0.41%	−41.93%	−2.15%
256f	4327632	92059542	2967642	49856	4286574	91335474	3175290	49312	−0.95%	−0.79%	7.00%	−1.09%
128s	72597852	551233638	846486	7856	51421086	537033762	2689650	6880	−29.17%	−2.58%	217.74%	−12.42%
192s	105310692	1022229270	1201230	16224	78050718	988899534	3845970	14568	−25.89%	−3.26%	220.17%	−10.21%
256s	69033492	918473904	1701324	29792	52048332	764352612	6005448	27232	−24.60%	−16.78%	252.99%	−8.59%

Hash-based Signature

- Construct digital signature from Hash functions ONLY
  - No Lattice, Code, MQ, RSA, ECC …
- Advantage:
  - Conservative Assumption
  - Post-Quantum
- Disadvantage :
  - Large Signature Size/Signing time
  - Can not construct PKE/KEM

Lamport's One-Time Signature

Let H be a cryptographic hash function, $y_{i,j} = H(x_{i,j})$

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{n,1} \end{bmatrix}, pk = \begin{bmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{n,1} \end{bmatrix}$$

sign(sk,m):

- $m = [m_1, m_2, \dots, m_n], m_i \in \{0,1\}$
- $\sigma = \left[x_{1,m_1}, x_{2,m_2}, \dots, x_{n,m_n}\right]$

verify(pk, m, σ):

- $m = [m_1, m_2, \dots, m_n]$
- Check if $H(\sigma_i) = y_{i,m_i}$

Lamport's One-Time Signature

Let H be a cryptographic hash function, $y_{i,j} = H(x_{i,j})$

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{n,1} \end{bmatrix}, pk = \begin{bmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{n,1} \end{bmatrix}$$

$n = 3, m = 010, \sigma = [x_{1,0}, x_{2,1}, x_{3,0}]$; Check if $H(\sigma_i) = y_{i,m_i}$

sign(sk,m):

- $m = [m_1, m_2, \ldots, m_n], m_i \in \{0,1\}$
- $\sigma = [x_{1,m_1}, x_{2,m_2}, \ldots, x_{n,m_n}]$

verify(pk, m, σ):

- $m = [m_1, m_2, \ldots, m_n]$
- Check if $H(\sigma_i) = y_{i,m_i}$

Lamport's One-Time Signature

Let H be a cryptographic hash function, $y_{i,j} = H(x_{i,j})$

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{n,1} \end{bmatrix}, pk = \begin{bmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{n,1} \end{bmatrix}$$

$n = 3, m = 010, \sigma = [x_{1,0}, x_{2,1}, x_{3,0}];$

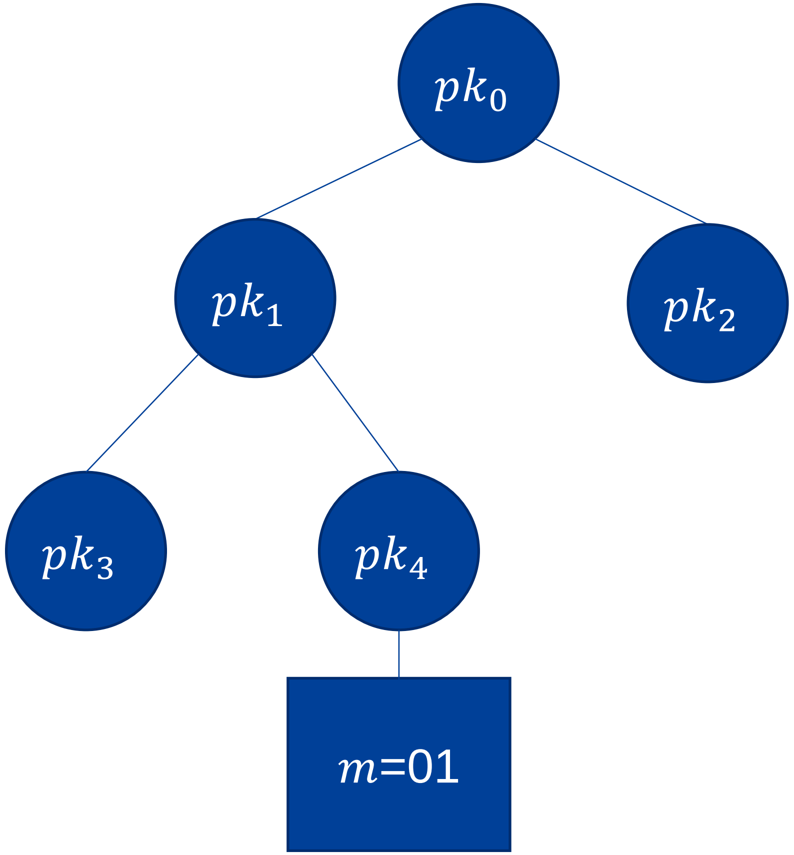
$n = 3, m' = 101, \sigma = [x_{1,1}, x_{2,0}, x_{3,1}];$

$n = 3, m^* = 111, \sigma = [x_{1,1}, x_{2,1}, x_{3,1}];$

Tree-based Signature

- OTS ⇒ SIG
- Each node of the tree is an OTS
- Generate nodes when you need it ($\text{node}_i = \text{PRF}(k, i)$)
- Parent node authenticates two children
- Leaf node authenticates the message

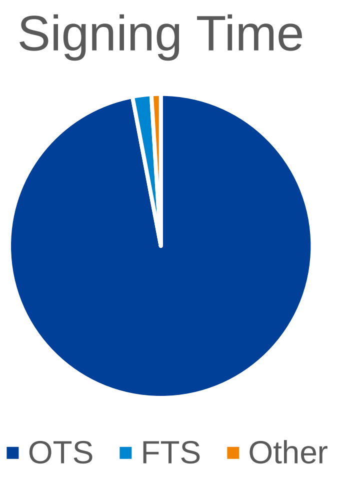
- $\text{sign}(sk_0, H(pk_1 || pk_2))$
- $\text{sign}(sk_1, H(pk_3 || pk_4))$
- …
- $\text{sign}(sk_4, m)$

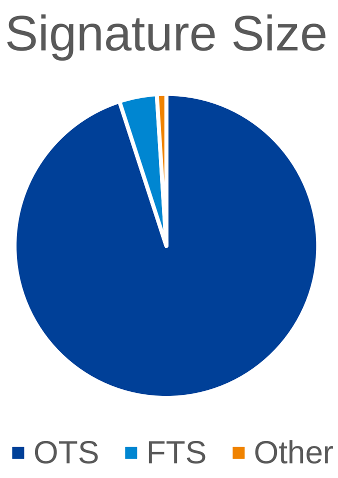


SPHINCS+

- Recently, NIST approved SPHINCS+ as a PQC standard

- OTS dominates the cost of SPHINCS+.





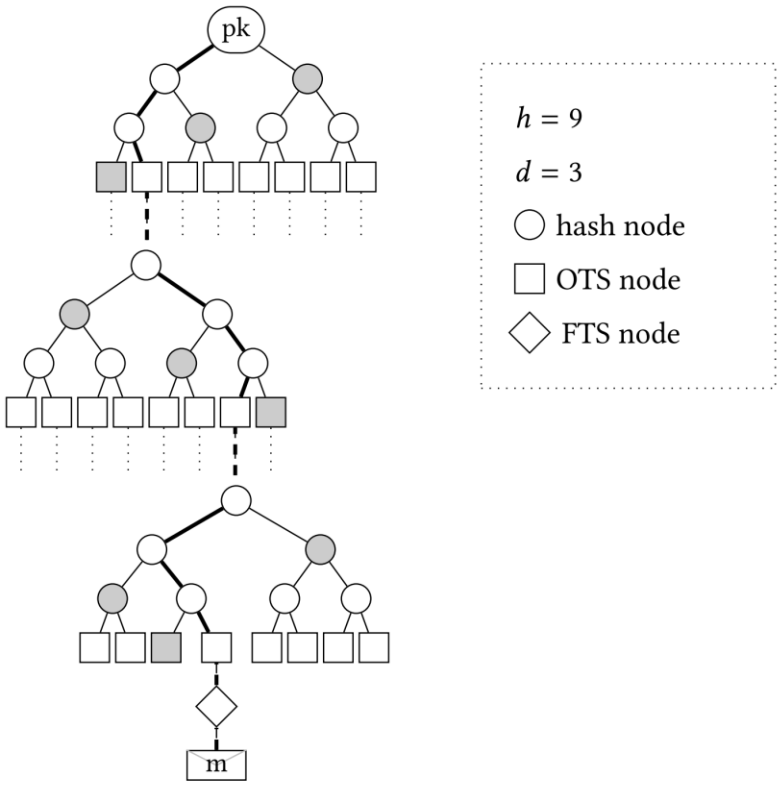


Figure 1: An illustration of a (small) SPHINCS structure.

Lamport's One-Time Signature

Let H be a cryptographic hash function, $y_{i,j} = H(x_{i,j})$

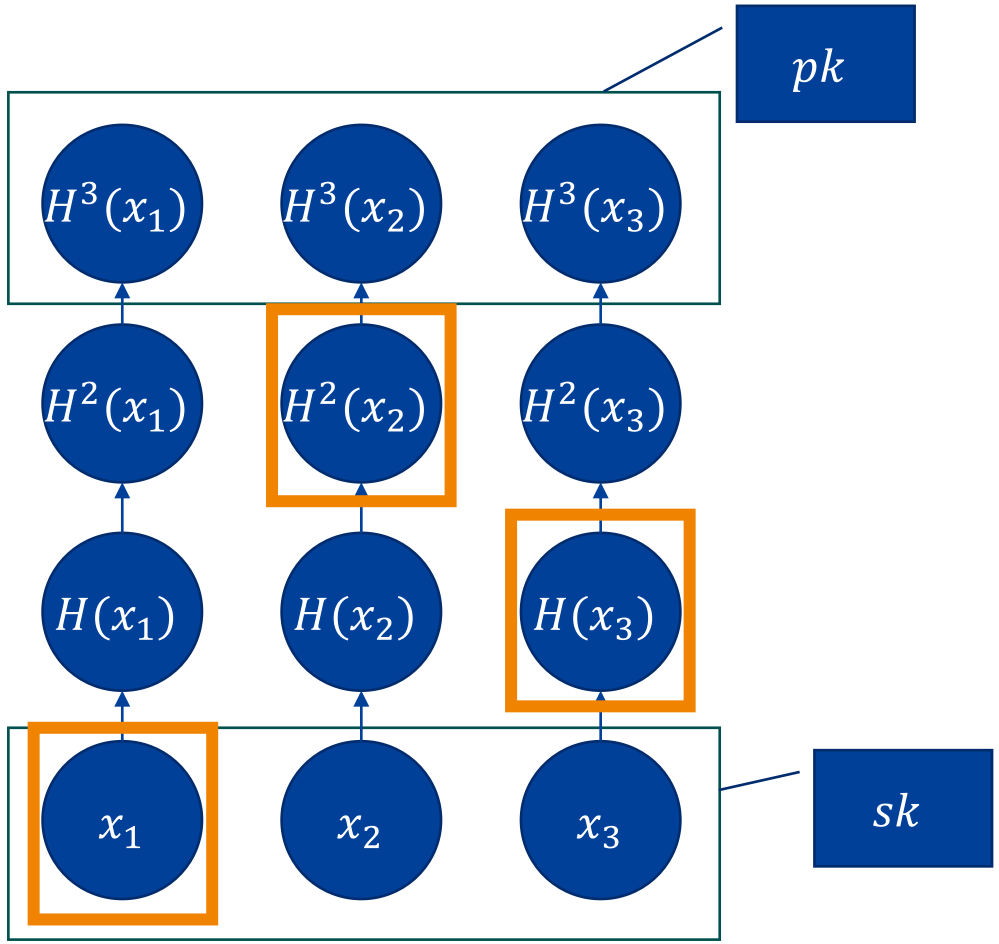
$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{n,1} \end{bmatrix}, pk = \begin{bmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{n,1} \end{bmatrix}$$

$n = 3, m = 010, \sigma = [x_{1,0}, x_{2,1}, x_{3,0}]$; Check if $H(\sigma_i) = y_{i,m_i}$

- One hash value (λ bit) to encode only 1 bit message

(Naïve) Winternitz One-Time Signature (WOTS)

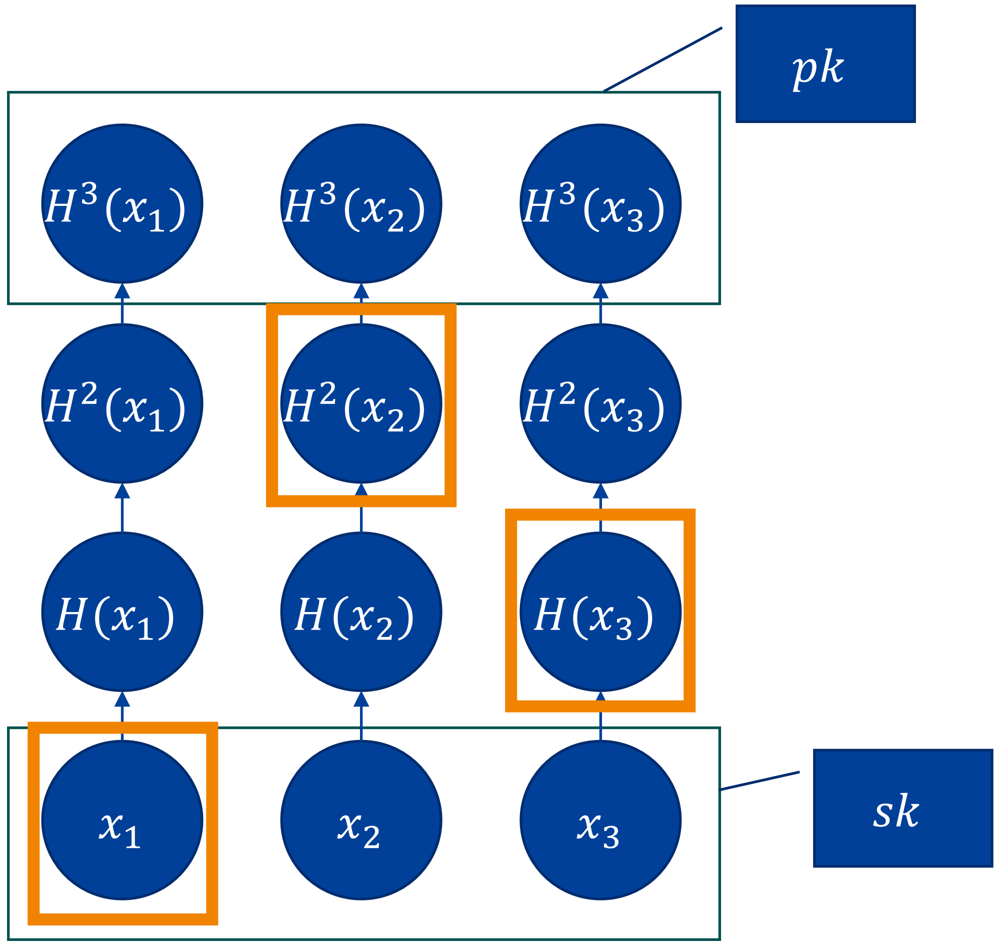
- $sk = [\dots, x_i, \dots]$
- $pk = [\dots, H^{w-1}(x_i), \dots]$
- sign(sk,m):
  - m=$[m_1, \dots, m_l]$, base-w number
  - $\sigma = [\dots, H^{m_i}(x_i), \dots]$
- verify(pk, m, σ):
  - $H^{w-1-m_i}(\sigma_i) = H^{w-1}(x_i) = pk_i$
- $m = [0,2,1]$
- $\sigma = [x, H^2(x_2), H(x_3)]$



(Naïve) Winternitz One-Time Signature (WOTS)

- sk =
- pk =
- sign
  - m
  - $\sigma = [\dots, H \quad (x_i), \dots]$
- verify(pk, m, σ):
  - $H^{w-1-m_i}(\sigma_i) = H^{w-1}(x_i) = pk_i$
- $m = [0,2,1]$
- $\sigma = [x, H^2(x_2), H(x_3)]$

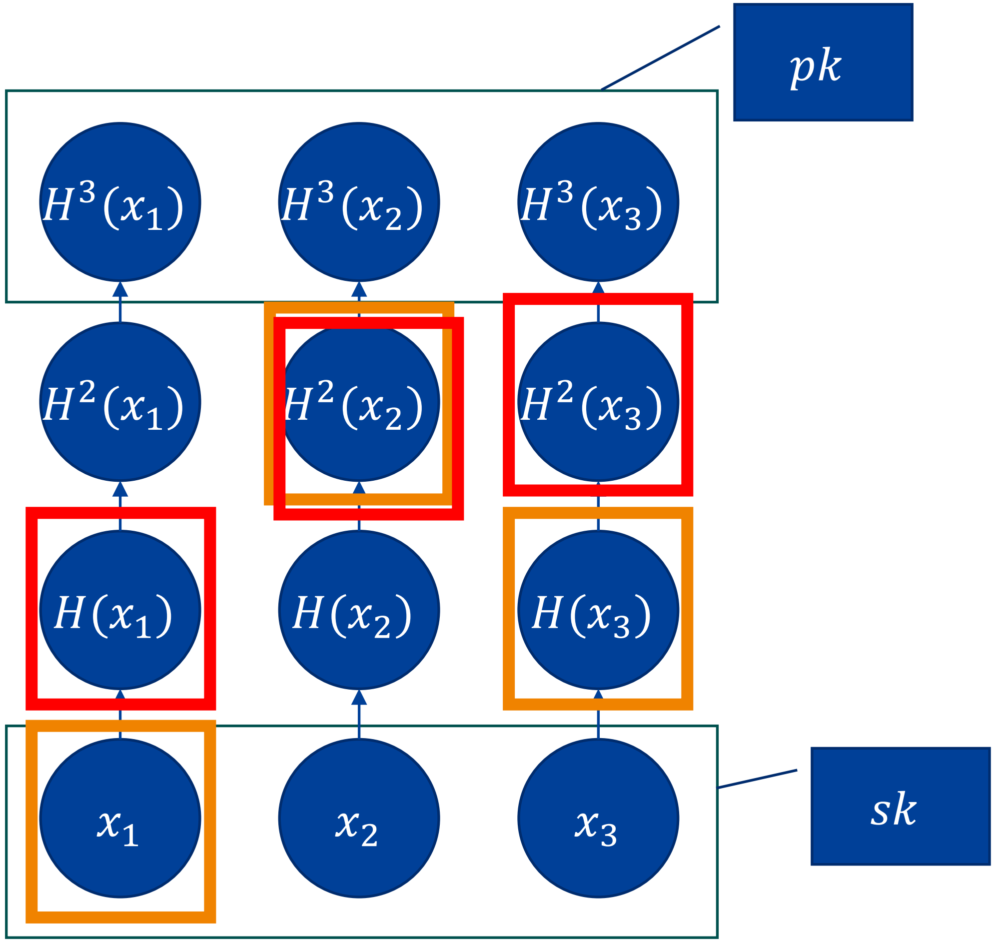
So we can encode $\log_2 w$ bit message on a length-w chain with only one hash value?



(Naïve) Winternitz One-Time Signature (WOTS)

- NOT Secure!
- Adversary can forge any m' such that $m' \geq m$ i.e. $\forall i, m'_i \geq m_i$

- $m = [0,2,1]$
- $\sigma = [x, H^2(x_2), H(x_3)]$
- $m' = [1,2,2]$
- $\sigma = [x, H^2(x_2), H^2(x_3)]$



Encoding of WOTS

- Encode messages such that for each pair $m \neq m'$, neither $\text{encode}(m) \leq \text{encode}(m')$ nor $\text{encode}(m) \geq \text{encode}(m')$
- Simple Solution:
  - $\text{encode}(m) = m||\bar{m}$, where $\bar{m}_i = w - 1 - m_i$
  - Sign the encoded message
- Proof
  - Suppose there are $m \neq m'$ such that $\text{encode}(m) \leq \text{encode}(m')$
  - Then $m \leq m'$ and $\bar{m} \leq \bar{m}'$
  - Then $m = m'$,
  - Contradiction to $m \neq m'$

Encoding of WOTS

- Encode messages such that for each pair $m \neq m'$, neither $\text{encode}(m) \leq \text{encode}(m')$ nor $\text{encode}(m) \geq \text{encode}(m')$
- Better Solution:
  - $\text{encode}(m) = m||c, \text{where } c = \sum \bar{m}_i$
  - Sign the encoded message
- Proof
  - Suppose there are $m \neq m'$ such that $\text{encode}(m) \leq \text{encode}(m')$
  - Then $m \leq m'$ and $c \leq c'$
  - There is an index i such that $m_i < m_i{}'$ because $m \neq m'$ and $m \leq m'$
  - But $c = \sum \bar{m}_i > \sum \bar{m}'_i = c'$
  - Contradiction to $c \leq c'$

Encoding of WOTS

- Encode messages such that for each pair $m \neq m'$, neither $\text{encode}(m) \leq \text{encode}(m')$ nor $\text{encode}(m) \geq \text{encode}(m')$
- Better Solution:
  - $\text{encode}(m) = m||c$ where $c = \sum \bar{m}$
  - Sign the encod
- Proof
  - Suppose there are $m) \leq \text{encode}(m')$
  - Then $m \leq m'$ and
  - There is an index i such that $m_i < m_i$ because $m \neq m'$ and $m \leq m'$
  - But $c = \sum \bar{m}_i > \sum \bar{m}_i = c'$
  - Contradiction

Checksum encoding is used in SPHINCS+ and XMSS

Constant-sum WOTS

- Encode messages such that for each pair $m \neq m'$, neither $\text{encode}(m) \leq \text{encode}(m')$ nor $\text{encode}(m) \geq \text{encode}(m')$
- Our Solution:
    - $\text{encode}(m) \mapsto \text{C}$, where each $v \in C$ is constant-sum i.e. $\sum_i v_i = s = \lfloor l(w-1)/2 \rfloor$
    - Sign the encoded message
- Proof
    - Suppose there are $m \neq m'$ such that $\text{encode}(m) \leq \text{encode}(m')$
    - There must be an index j such that $v_j < v_j'$
    - However, $\sum_i v_i = s = \sum_i v_i'$
    - Thus $\sum_{i \neq j} v_i + v_j = \sum_{i \neq j} v_i' + v_j'$
    - Therefore $\sum_{i \neq j} v_i > \sum_{i \neq j} v_i'$
    - There must exist an index k such that $v_k > v_k'$
    - Contradiction to $v \leq v'$

Constant-sum WOTS

- The concept of Constant-sum WOTS is not new [Vaudenay 1992]
- But we don't know how to efficiently encode them.

We have to use a monotone code for the same reason. The set of the words $b_1 b_2 \ldots b_s$ such that :

$$b_1 + b_2 + \ldots + b_s = \lfloor s\frac{\beta - 1}{2} \rfloor$$

is still a good one. However, it is still difficult to exhibit an efficient coding scheme. To get rid of this difficulty, we will not use this algorithm for signatures, but for interactive proofs.

Counting

- $D_{l,s} = \left|\{v \in [w]^l : \sum_i v_i \ = s\}\right|$
- w : length of each chain
- ℓ : the number of chains
- s : the sum of all chains

$$D_{\ell,s} = \sum_{i=0}^{w-1} D_{\ell-1,s-i}$$

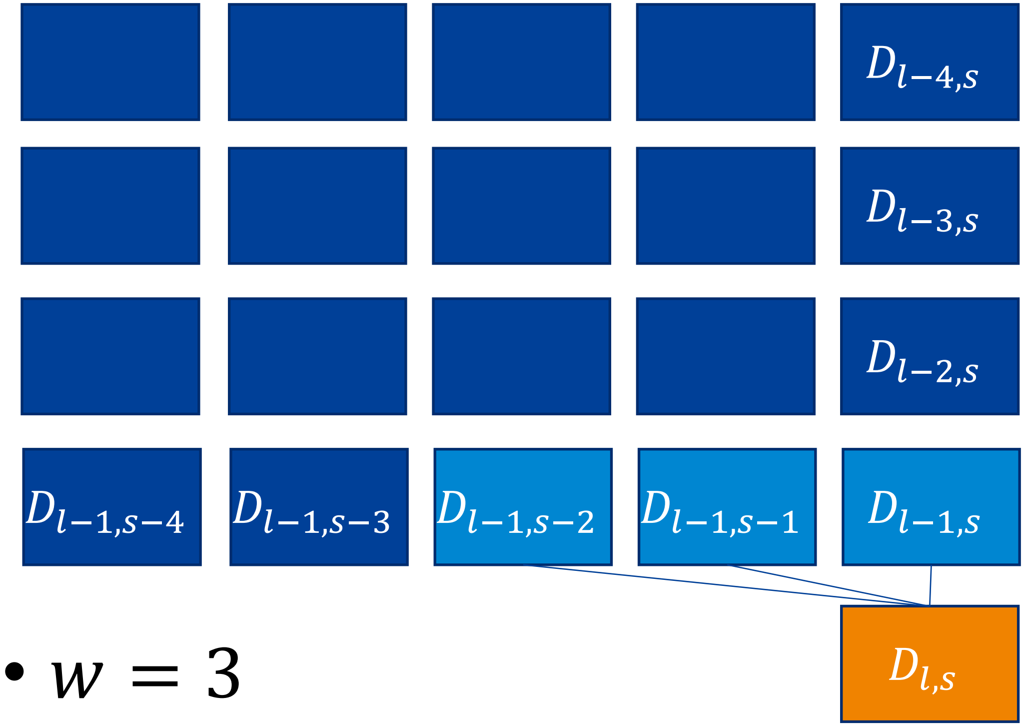
- Init

$$D_{\ell,s} = \begin{cases} 1, & \ell = 1, 0 \le s \le w-1 \\ 0, & \ell \ge 2, n < 0 \end{cases}$$

- Final Result

$$D_{\ell,\frac{(w-1)\cdot\ell}{2}}$$

Encoding

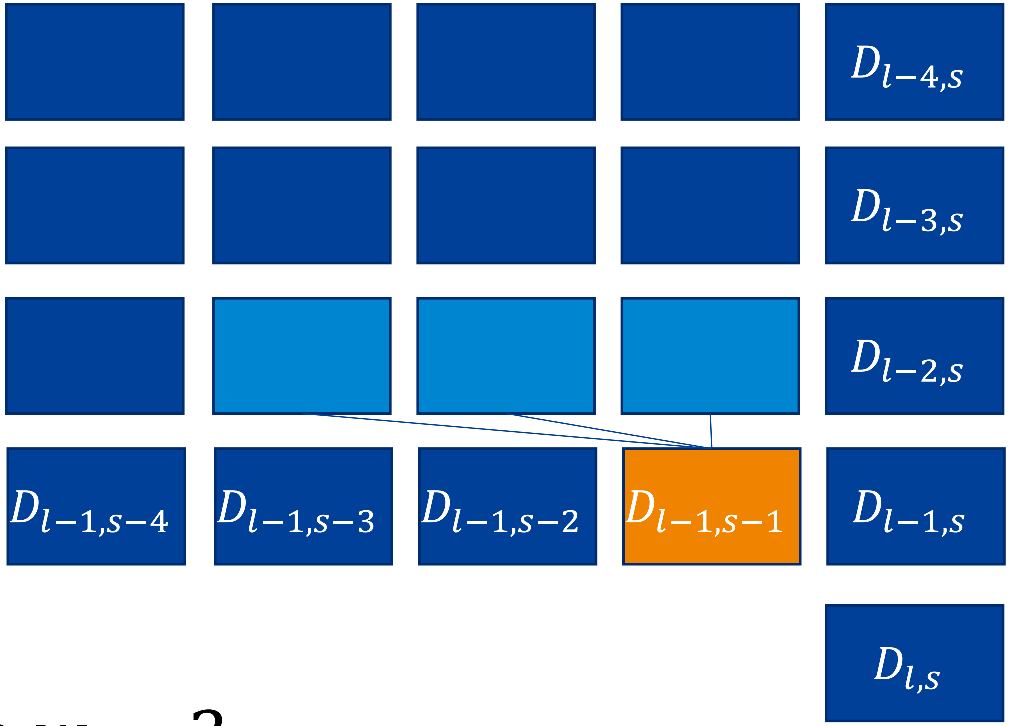


- $w = 3$
- $D_{l-1,s} \leq x < D_{l-1,s} + D_{l-1,s-1}$

Algorithm 1: Encode

```
Function Encode(x)
    Let v be an array of size l;
    s := ⌊l(w − 1)/2⌋;
    for i := l − 1 … 0 do
        for j := 0 … min(w − 1, s) do
            if x ≥ D_{i,s−j} then
                x := x − D_{i,s−j};
            else
                v_i := j;
                break;
        s := s − v_i;
    return v;
```

Encoding



- $w = 3$

Algorithm 1: Encode

Function Encode(x)

Let v be an array of size l;

$s := \lfloor l(w-1)/2 \rfloor$;

for $i := l-1 \ldots 0$ **do**

  for $j := 0 \ldots \min(w-1, s)$ **do**

    if $x \geq D_{i,s-j}$ **then**

      $x := x - D_{i,s-j}$;

    else

      $v_i := j$;

      break;

  $s := s - v_i$;

return v;

Comparison

- Reduce ~2% signature size and hash function calls
- Stable Computing Time
  - The number of hash function calls is fixed

	128-bit		192-bit		256-bit	
w	WOTS$^+$	CS	WOTS$^+$	CS	WOTS$^+$	CS
8	46	45	67	66	90	88
16	35	34	51	50	67	66
24	31	30	45	44	59	58
32	28	27	42	40	55	53
40	27	26	39	38	52	50
48	25	25	37	36	48	48

Results

- Improve SPHINCS+ by replacing their OTS with Constant-sum Winternitz OTS
- And we retune the parameter of SPHINCS+
- The signature size and signing time are slightly better, but verification time increased.

	SPHINCS$^+$				SPHINCS-α				Relative Change			
Param.	KeyGen	Sign	Verify	Size	KeyGen	Sign	Verify	Size	KeyGen	Sign	Verify	Size
128f	1143558	26872236	2204802	17088	1036602	26635716	2028186	16720	−9.35%	−0.88%	−8.01%	−2.15%
192f	1662498	45405504	3003534	35664	2199276	45218790	1744038	34896	32.29%	−0.41%	−41.93%	−2.15%
256f	4327632	92059542	2967642	49856	4286574	91335474	3175290	49312	−0.95%	−0.79%	7.00%	−1.09%
128s	72597852	551233638	846486	7856	51421086	537033762	2689650	6880	−29.17%	−2.58%	217.74%	−12.42%
192s	105310692	1022229270	1201230	16224	78050718	988899534	3845970	14568	−25.89%	−3.26%	220.17%	−10.21%
256s	69033492	918473904	1701324	29792	52048332	764352612	6005448	27232	−24.60%	−16.78%	252.99%	−8.59%

Thank You!

Q & A