Controller of Certifying Authorities
Ministry of Electronics & Information Technology
Government of India

G20 भारत 2023 INDIA

IEEE COMPUTER SOCIETY Bangalore Chapter

IAS IEEE INDUSTRY APPLICATIONS SOCIETY Linking Research to Practice Bangalore Chapter

4TH INTERNATIONAL CONFERENCE ON

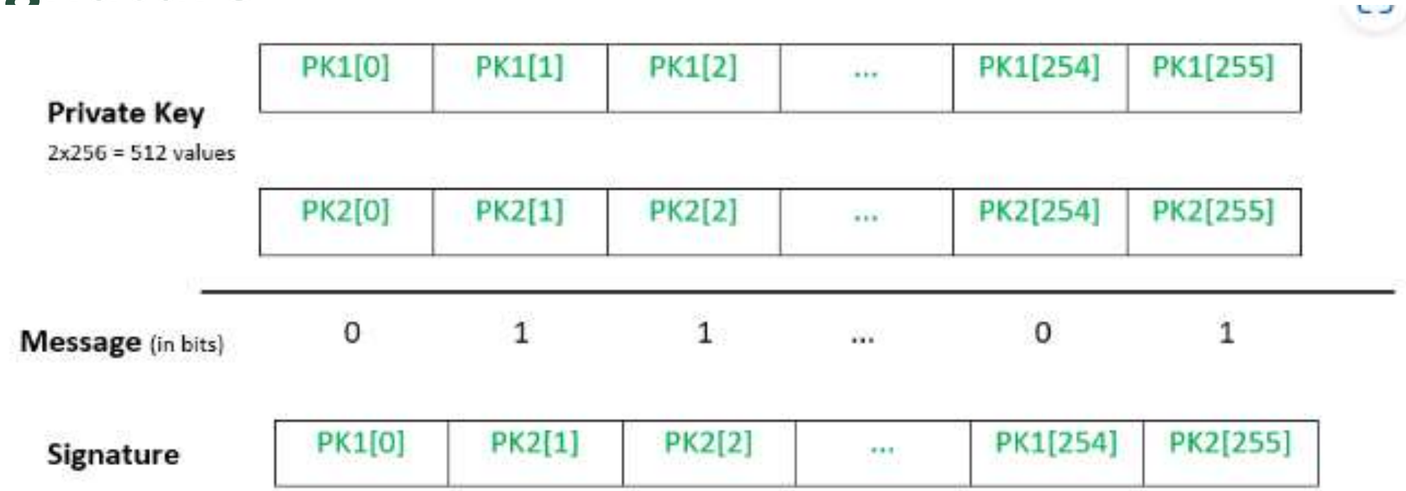
PUBLIC KEY INFRASTRUCTURE AND ITS APPLICATIONS (PKIA 2023)

SEPTEMBER 8-9TH, 2023

Hash Based Digital signatures- A tutorial Review

P.V.Ananda Mohan
Formerly with CDAC, Bangalore

Lamport signature



- Two sets of 256 number of random 256 bit words S0 and S1. These form the **private key** of size: 512x256 bits.
- Hash both using SHA 256 to get two sets: H0 and H1 to get the **public key** of size: 512x256 bits.
- Hash your message to get a 256 bit hash (bits h0 to h255)
- If bit h(0) is 1 select from S1,0 ; else from S0,0 to get W0.
- If bit h(1) is 1 select fromS1,1 ; else from S0,1 to get W1 .
- Continue for all 256 bits of the message hash.
- Your hash is the obtained 256 number of 256 bit concatenated words: W0 ||W1||.....||W255.
- Hence Signature size is 256x256 bits = 8192 bytes.

Winternitz One Time Signature (WOTS)

•Signature size and Memory can be reduced by considering blocks of bits in the hash
•Block size is called **Winternitz parameter.** If w = 256, block size = log(w) = 8
• Private Key generation: Choose 32 random numbers each of 256 bits.
•Hash each of these **256 times** to get 32 number of 256 bit words. This forms the **Public Key.**
•Hash message to get 256 bit Hash. Consider as 32 bytes N(0)N(1)…N(31).
•Hash N(i) (256-N(i)) times. Repeat for each N(i) and concatenate all resulting 256 bit words. This forms the signature.
•Verification:
Verifier will hash the message to obtain N(0) N(1). ..N(31).
•Receiver hashes the received 256 bit words N(i) times . The result shall be same as the word in the public key corresponding to this block.
•Otherwise fails.

Winternitz One Time Signature

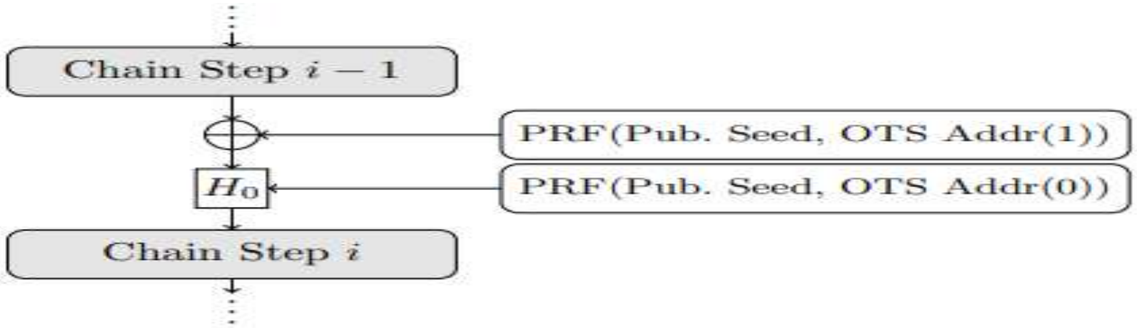
- Limitation: knowing the signature for a block m_i allows one to forge for any $m'_i > m_i$
- Solution: introduce a inverse check sum.
- Add all 32 number of (2^w-1-m_i) values
- Sum can be written a *l* additional blocks:

$$l_1 = \left\lceil \frac{n}{\log(w)} \right\rceil, l_2 = \left\lfloor \frac{\log(l_1(w-1)}{\log(w)} \right\rfloor + 1, l = l_1 + l_2$$

- Example: w=2, n = 256,l_1=128, l_2=5; *l* =133
- Thus 133 *Winternitz chains* will be required to do repeated hashings to get public key from private key and similarly for signatures.
- Size of w is trade-off between signature size and speed.

WOTS+

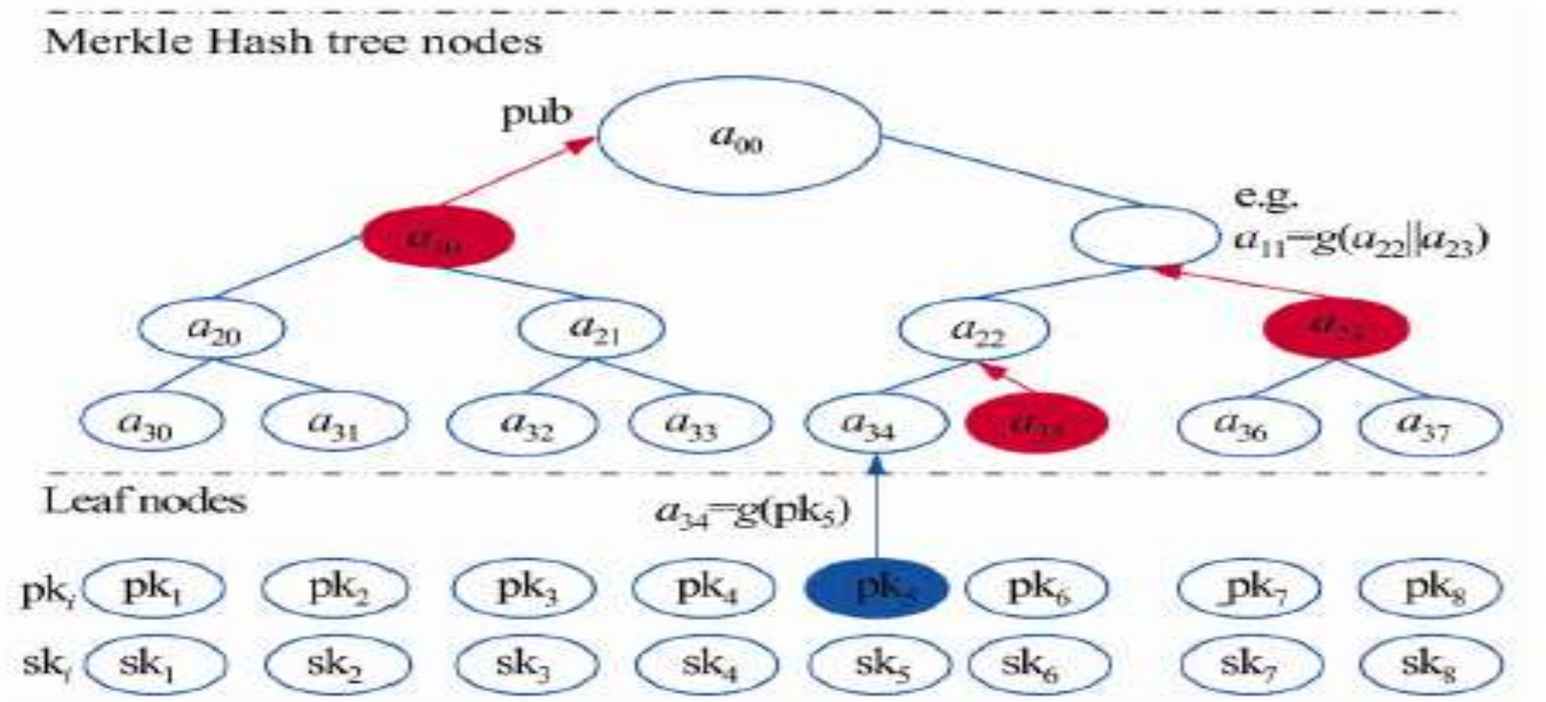
- Improvement over WOTS
- Uses a chain function instead of just hashing
- Needs Random bit masks ‘’r’’ to be generated for each step. *H* is a hash function,



- Limitation: Key size will be increased because of additional masks.
- Solution: generate them from a SEED all needed masks
- Used in XMSS (Extended Merkle signature system)
- One Pubic key, private key pair can be used only once for signing one message.

Extended Merkle Signature System

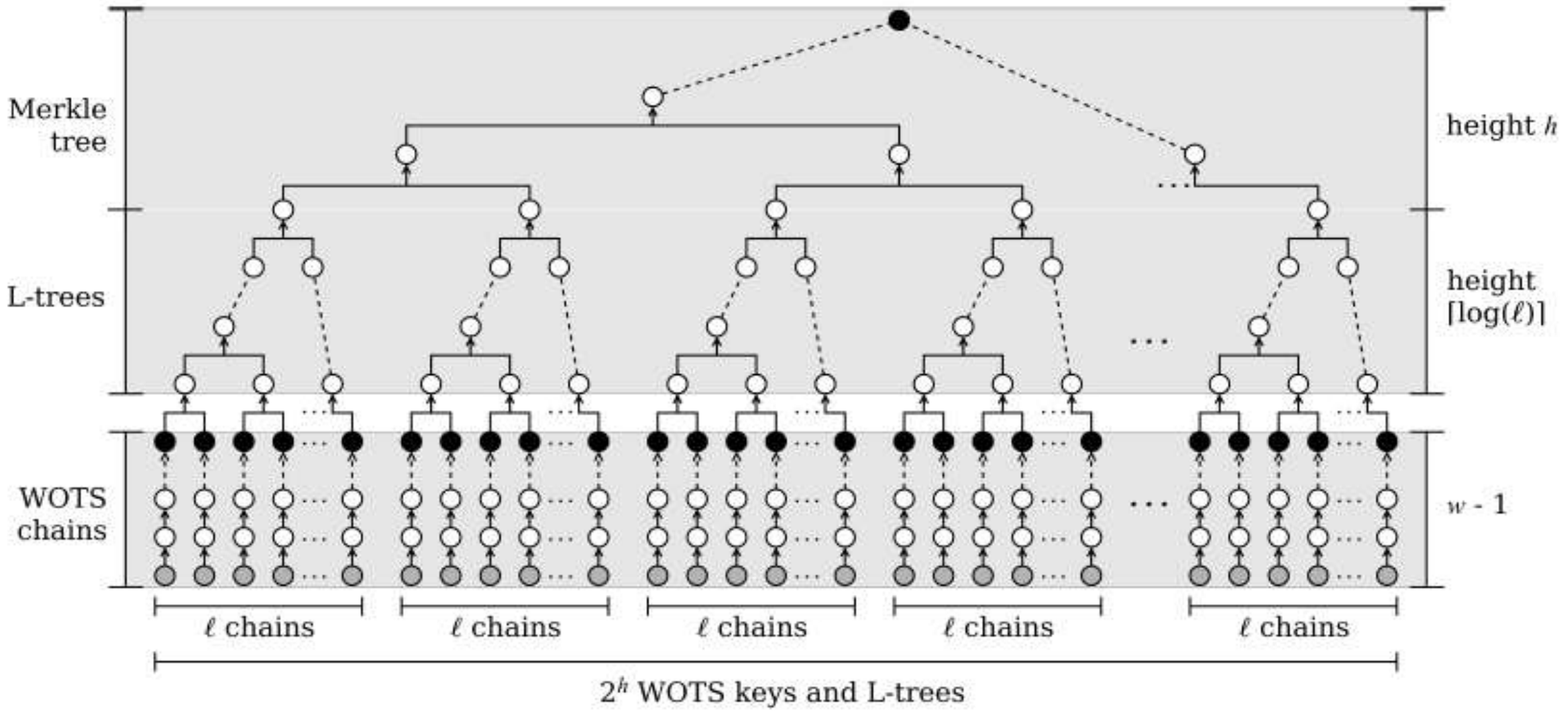
- *How to sign several messages? Use Merkle tree*



- *Verification path* $\{a_{35}, a_{23}, a_{10}\}$. .

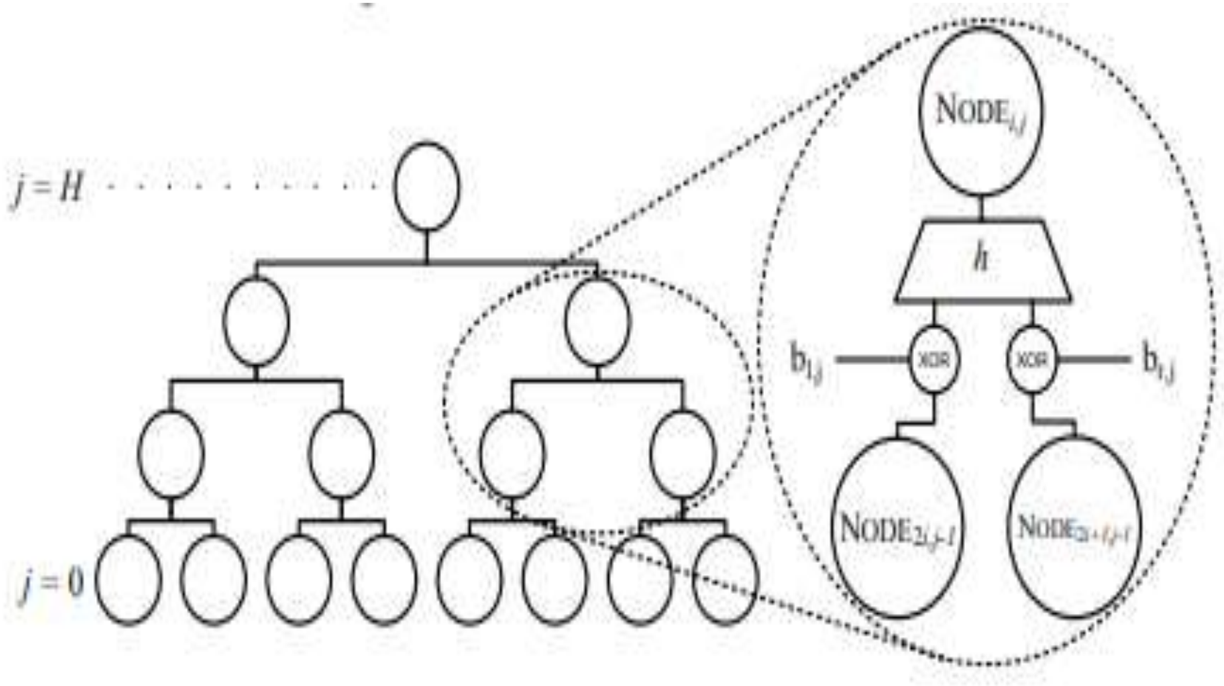
XMSS (RFC 8391)

- Uses Merkle tree and L-tree also. Can sign 2^h messages.
- Winternitz parameter *w* can be 4 or 16. Trade off: large *w* leads to smaller signature but more computation time.
- Black nodes are WOTS+ public keys
- Size of w is trade-off between signature size and speed.



L-tree and XMSS tree computation

•There are 2^h L-trees.
•L-tree is used to compress the WOTS+ public key ‘’len’’ number of *n* bytes each to n bytes. The levels needed are log (*len*).
•Generation of masks, additions (XOR), Hashing in each level of the tree is from a part of the public key seed 32 bytes using a PRF. Note PRF (pseudo random function) uses an additional input in addtion to actual input.

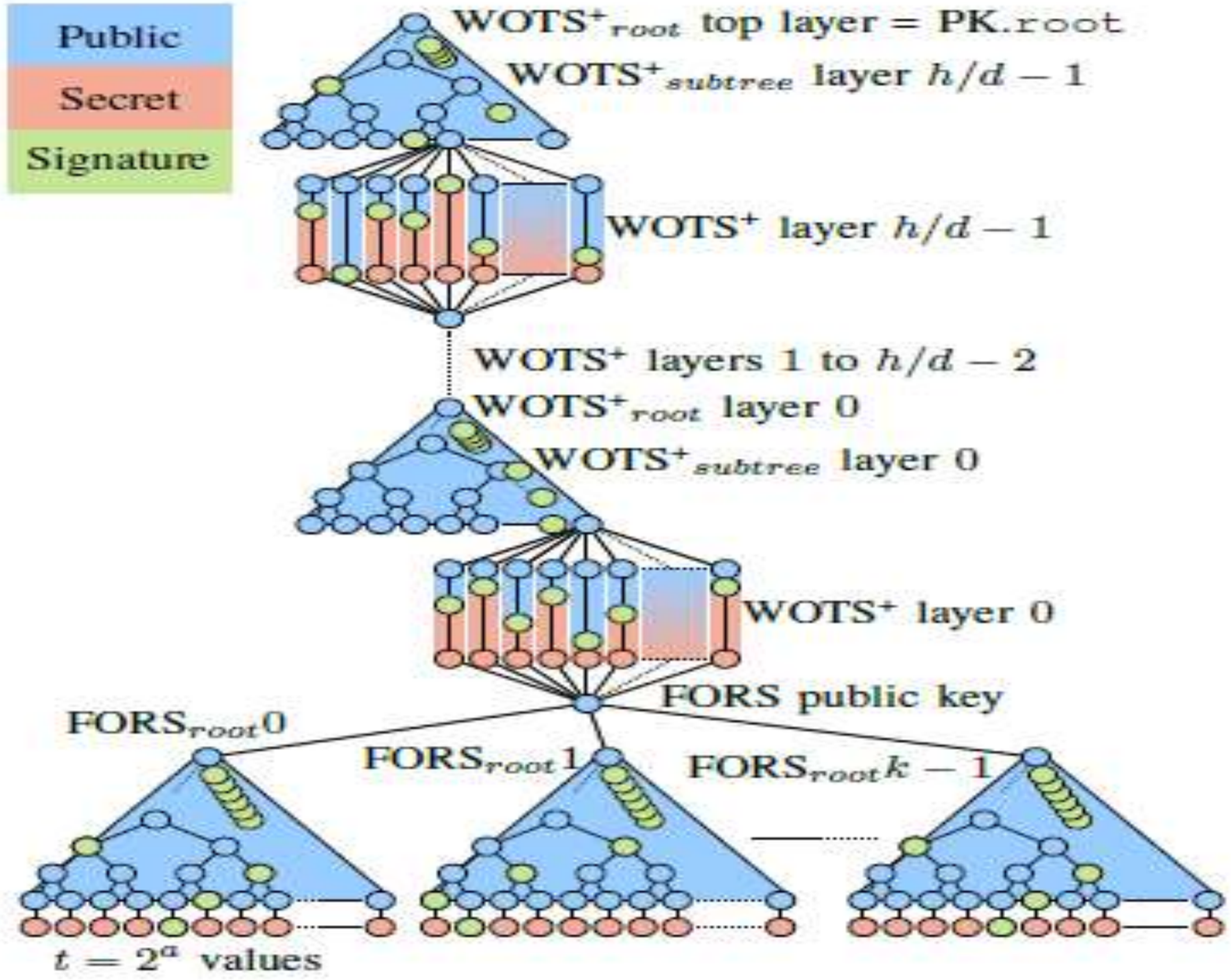


SPHINCS+

- **(Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures)**
- Stateless: meaning no need to keep track of earlier usage/history
- Number of messages that can be signed = 2^{60} for each private key.
- Uses WOTS+, FORS (Forest of random sets) and variety of hash functions (SHA 256, Haraka, SHAKE, BLAKE and CHACHA stream cipher).
- Uses hierarchy of several trees in hypertree
- Several number of trees in FORS. (k trees)
- WOTS+ uses three options for w = 4, 16,256
- Security parameters available 128, 192 and 256 bit
- Resistance to multi-target attacks
- Uses tweakable hash functions, PRFs

SPHINCS+

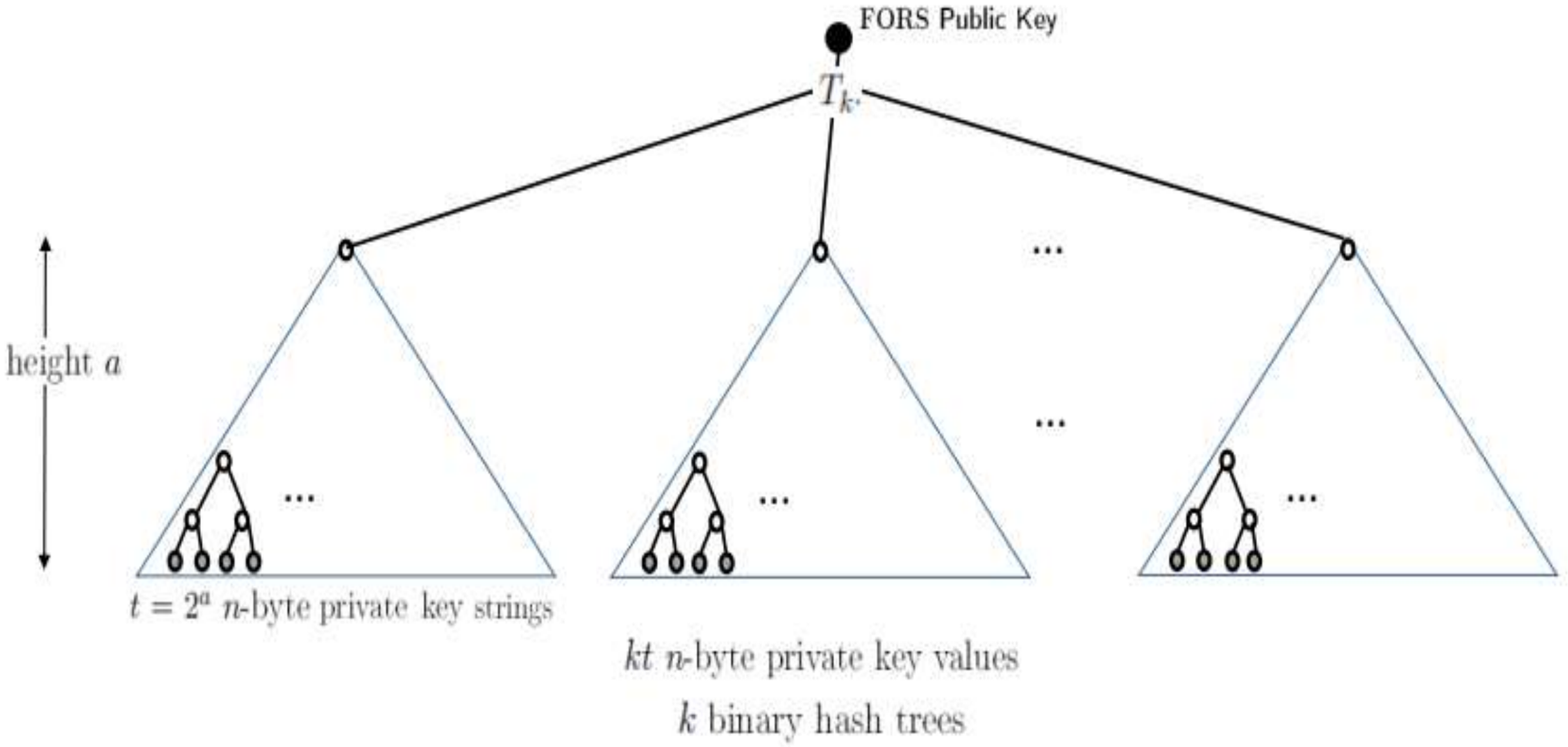
- Lowest layer is FORS
- Next WOTS+ layer signs the FORS root Public key.
- Intermediate trees sign the root of the tree below.
- There are enough leaves in the Merkle tree.
- It is statistically improbable to select same leaf for different signatures.
- Lowest layer signs FORS keys whereas each upper layer signs the public key of the immediate layer below.
- All trees have equal height.
- Total height = h; number of layers = d, height of XMSS tree is h' = h/d.



Parameters used in SPHINCS+

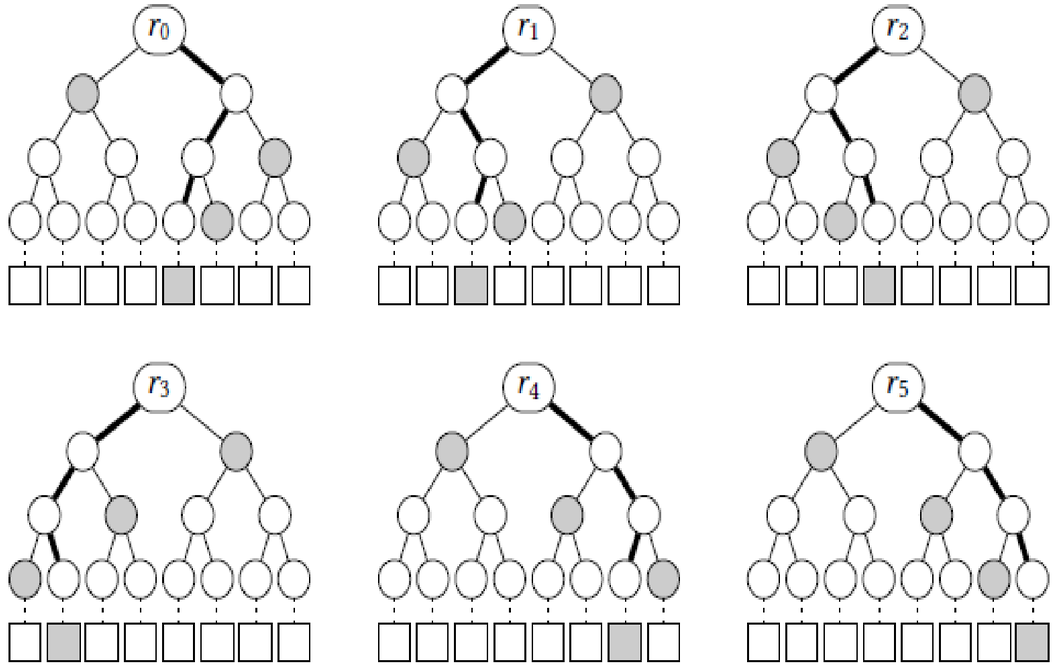
n	security parameter in bytes
h	height
d	number of layers of the hypertree
k	number of trees in FORS
t	number of leaves of a FORS tree

FORS (Forest of Random Sets) tree



- FORS is used to sign at random leaves

FORS illustration



- Randomizer R = PRF(SK.prf,optrand,M)
- MD‖idx = H_{msg}(R,PK.seed, pk.root. M)
- MD is fed to FORS where idx is the leaf index.
- Advantage is that the attacker cannot select an index freely.

- Message of k × a bits k = 6, a = 3 : **100 010 011 001 110 111.**

SPHINCS+ hypertree

- All leaf nodes of all intermediate trees are deterministically generated WOTS+ public keys that do not depend on any of the trees below it. This means that the complete **hypertree is purely virtual: it never needs to be computed in full.**
- **During key generation only , top-most sub- tree is computed .**
- Hypertree is not used to sign messages but is used to sign public keys of FORS instances.
- Similar to XMSS^{MT}
- It is a certification tree of XMSS instances
- SPHINCS+ generates a FORS instance and adds randomness to it for signing the original message. Then signs the FORS public key with an instance of the hypertree to get the SPHINCS+ signature.

Recommended parameter sets in SPHINCS+

	n	h	d	$\log(t)$	k	w	bitsec	sec level	sig bytes
SPHINCS$^+$-128s	16	63	7	12	14	16	133	**1**	7 856
SPHINCS$^+$-128f	16	66	22	6	33	16	128	**1**	17 088
SPHINCS$^+$-192s	24	63	7	14	17	16	193	**3**	16 224
SPHINCS$^+$-192f	24	66	22	8	33	16	194	**3**	35 664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	**5**	29 792
SPHINCS$^+$-256f	32	68	17	9	35	16	255	**5**	49 856

- Example last entry: n=32, d=17,h=68,a=log (t)=9,k=35,
- we have signature = ($k \times n \times (a+1)$) +n + (h+($len \times d$)) $\times n$
- = (35x32x10)+32+ (68+(67x17))x32 = 11200+39648 = 49856 bytes

Comparison of Post-Quantum Round 3 Digital Signatures

Method	Public key size	Private key size	Signature size	Security level
Crystals Dilithium 2 (Lattice)	1,312	2,528	2,420	1 (128-bit)
Crystals Dilithium 3	1,952	4,000	3,293	3 (192-bit)
Crystals Dilithium 5	2,592	4,864	4,595	5 (256-bit)
Falcon 512 (Lattice)	897	1,281	690	1 (128-bit)
Falcon 1024	1,793	2,305	1,330	5 (256-bit)
Sphincs SHA256-128f	32	64	17,088	1 (128-bit)
Sphincs SHA256-192f	48	96	35,664	3 (192-bit)
Sphincs SHA256-256f	64	128	49,856	5 (256-bit)

CONCLUSION

- NIST statement for selecting SPHINCS+ in spite of large signature size and complexity
- **'If NIST's confidence in better performing signature algorithms is shaken by new analysis, SPHINCS+ could provide an immediately available algorithm for standardization at the end of the third round''.**
- Hash based signatures are robust.
- Even though these are very intensive in hash computations, efficient implementations of SHA2, SHAKE. SHA3 are available.
- More importantly, tomorrow if current hash functions are threatened, we can substitute with another improved/better hash function.

Controller of Certifying Authorities
Ministry of Electronics & Information Technology
Government of India

G20 भारत 2023 INDIA

THANK YOU

भगत संगणन विकास केंद्र
सीडैक CDAC
॥ज्ञानादेव तु कैवल्यम्॥
CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING

IEEE COMPUTER SOCIETY
Bangalore Chapter

IAS IEEE INDUSTRY APPLICATIONS SOCIETY
Linking Research to Practice
Bangalore Chapter