Breaking Category Five SPHINCS$^+$ with SHA-256

Ray Perlner[1], John Kelsey[1,2], and David Cooper[1]

1. NIST, 2. COSIC/KU Leuven

Summary of Result

- SPHINCS^+ is a stateless hash-based signature selected for standardization by NIST
- We present a forgery attack that reduces classical security by 40 bits
  - For submitted parameter sets:
    - That target Category 5
    - While using SHA-256
- Our attack builds on a previous attack by Antonov on the DM-SPR property of SHA-256 (a security assumption for SPHINCS^+)
- The SPHINCS^+ team has proposed a tweak which defeats our attack by using SHA-512 instead of SHA-256 (where necessary)

Outline

- Hash-Based Signatures: One-time, Multi-Use, Stateless
- SPHINCS$^+$ Basic Design, including WOTS$^+$
- DM-SPR Property and Antonov's Attack
- Using Antonov's Attack to Forge WOTS$^+$ (This Paper)
- Optimizations (This Paper)
- The SPHINCS$^+$ Tweak
- Conclusion

Hash-Based One-Time Signature (OTS)

- Most basic hash-based signature (Lamport 1979)
  - For bit b_i
    - Generate a secret S_{i0} for signing 0 and another S_{i1} for signing 1
  - Public key is

$$H(S_{00})|\ H(S_{00})|\ \ldots\ H(S_{(n-1)0})|\ H(S_{(n-1)1})$$

  - Can securely sign one n-bit digest
    - Signature is:

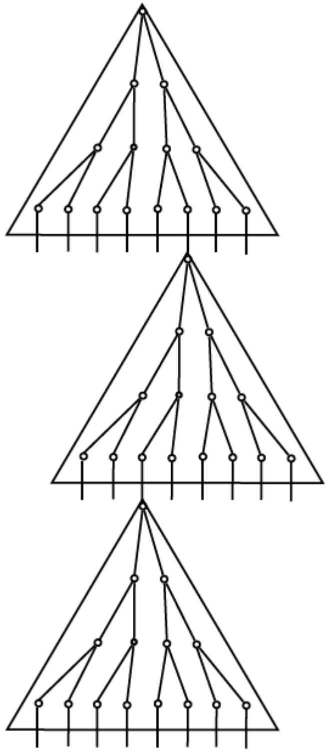
$$S_{0b_0}|\ \ldots\ |S_{(n-1)b_{(n-1)}}$$

- More advanced variants (e.g. WOTS+ – discussed later)
  - Reduce signature size using hash chains, etc.
  - Reduce PK size by using public key hash instead
    - We'll call the thing that gets hashed the *Public Key Preimage*

Multi-Use Hash-Based Signature

- Hash many OTS public keys together in a Merkle Tree
  - Only increases signature size logarithmically
  - But all OTS keys need to be precomputed
- Can get rid of precomputation by having OTS leaves of top tree sign roots of trees generated on the fly

- Hypertree:



Stateless Hash-Based Signature

- Create a hypertree by having multiple layers of Merkle trees with leaves from one signing the root of the next
- If the hypertree has enough leaves, the leaf can be chosen randomly with little risk of using the same leaf twice
  - Can make hypertree a lot smaller by using a few-time signature to sign the message
- Generate OTS keypairs pseudorandomly from seed and hypertree location so that each upper leaf always signs the same Merkle-Tree root

SPHINCS$^+$

Basic Design

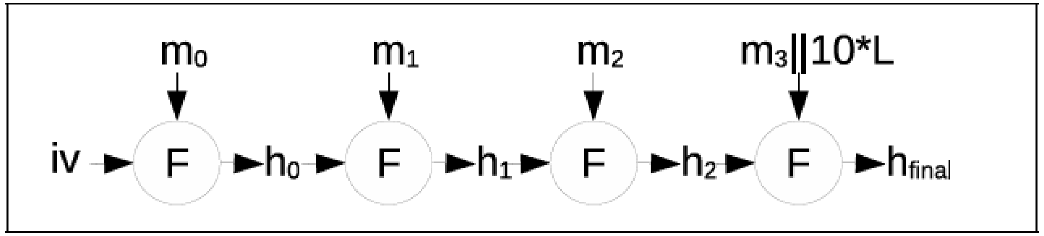
- Merkle roots are signed by WOTS$^+$
  - The Focus of our attack
- (Randomized) message digest is signed by FORS
  - FORS root is also signed by WOTS$^+$
- Hypertree path to FORS key is determined by extended message digest

SPHINCS+

Prefixes and *Distinct Function Multitarget Preimage Resistance* (DM-SPR)

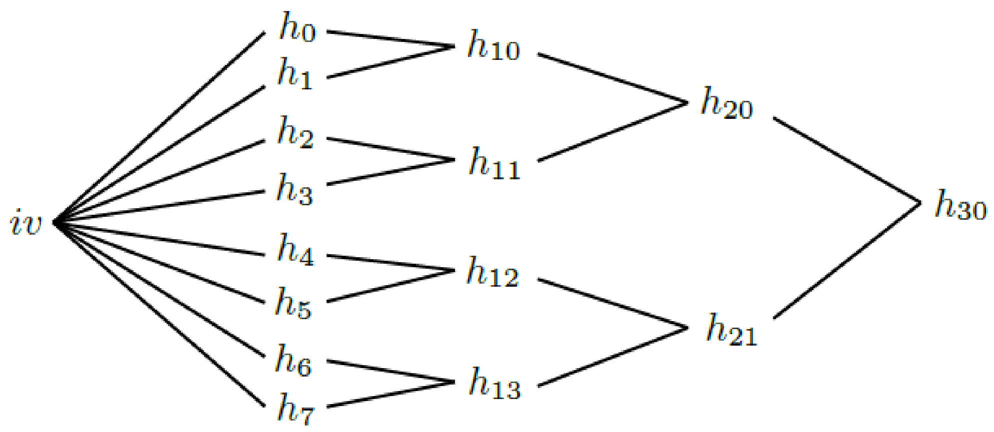
- Many places in hypertree where a preimage can create a forgery:
  - Hashes in Merkle Trees
  - Hash Chains in $WOTS^+$
  - Hash trees in FORS
  - FORS public key hash
  - $WOTS^+$ public key hash (Our attack here)
- New targets are revealed with every honest signature
- To avoid a 1 out of t multi-target preimage attack:
  - Make sure hash input at each hypertree location has a distinct prefix
  - Formalized as a tweakable hash function with DM-SPR property

Merkle-Damgård (SHA-256) Construction



- Hash an arbitrary-sized message using a fixed-input-length compression function, F
- Break message into B-bit blocks and repeatedly use F to produce an n-bit chaining value
- For SHA-256: B=512, n=256
- MD hash is proven collision resistant if F is
- But, MD doesn't always get more than $n/2$ bits of security for other properties:
  - Multicollisions for multi-block messages [Joux 2004]
  - Long message preimage attack [Dean 1999], [KS 2004]
  - Herding attack [KK 2005]
- What about DM-SPR?

Herding Attack



- Create many messages
  - With distinct fixed prefixes
  - That hash to the same value
- Build “Diamond Structure”
  - Distinct prefixes result in distinct internal states (h_1-h_7)
  - Use collision search on compression function to find message blocks that collide resulting in fewer distinct states (h_{10}-h_{13})
  - After adding logarithmically many (Above, 3) blocks to prefix, all messages hash to same chaining value (h_{30})

Antonov's Attack on SHA-256 DM-SPR [Antonov 2022]

- Collect t target hashes with different prefixes
- Find preimage with the same prefix for 1 of them
  - Use Herding to reach same state from all prefixes at the penultimate block
  - Use Multi-Target preimage search on compression function to find a block to append and reach a target
- Longest hash input in SPHINCS $^+$ is WOTS$^+$ public key hash
- That's still pretty short (34 blocks)
  - To balance cost of herding, multi-target preimage search, use some compression-function 3-collisions
    - Let t be $2^{10}3^{23} \approx 2^{46}$ instead of 2^{33}
    - 3-Collision search cost: $1.5 \cdot 3^{23} \cdot 2^{170.7} \approx 2^{208}$
    - Multi-Target preimage cost: $2^{256}/2^{46} \approx 2^{210}$

What's Left to Do?

- Antonov's attack lets us create a validly-signed $WOTS^+$ public key preimage
- But we need to know the corresponding private key to forge a $SPHINCS^+$ signature
  - This involves knowing preimages of parts of $WOTS^+$ public key
  - For validity, prefix must match hypertree location
  - But hypertree location depends which target we reached
  - No way to force correct prefix for all targets
- Or at least part of it…
  - As long as we can sign more than one possible digest with our $WOTS^+$ key
  - Can graft a forged Merkle-Tree root to the hypertree for less than 2^{256} work!

Our Attack: Outline

- Find a preimage of some $WOTS^+$ public key with enough private key info to sign **some** digests
- Brute-force search for a valid Merkle/FORS tree whose root has signable digest
- Sign the tree root with the attacked $WOTS^+$ key
- To forge a signature, try message randomization strings until the hypertree address is a descendent address of the tree root

WOTS+ Signature

- Write digest as base-w (16) number
- Append a base-w checksum
  - (960 – <sum of digits>)
- Sign each digit d_i of digest plus checksum by:
  - Hash $sk_{i,0}$ (with prefix) d_i times
  - Put the result in the signature
- Note: The signature of 0xF is just pk_i

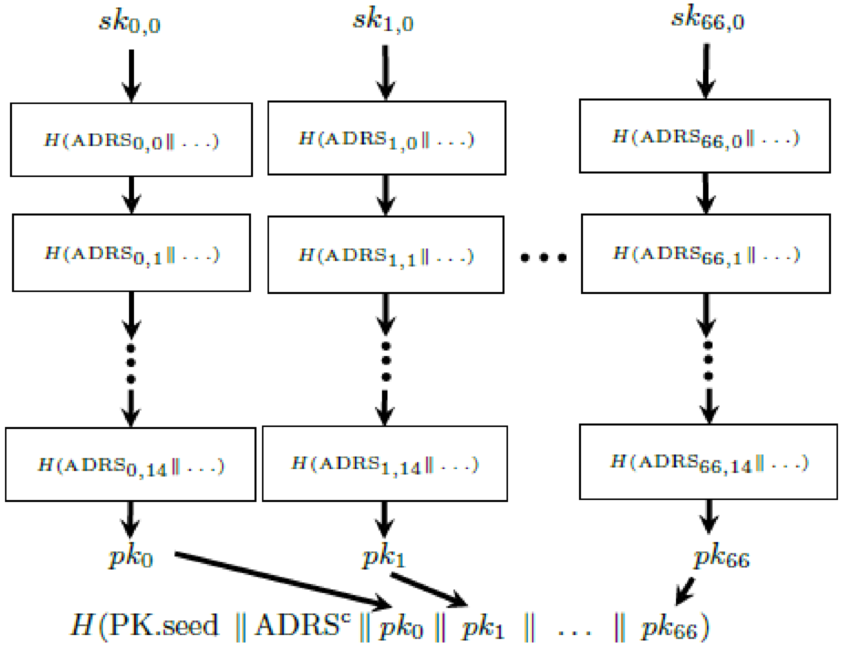


Fig. 2. A WOTS+ public key.

Finding a Merkle/FORS Root We Can Sign

- Aim to sign a digest like:

  xxxxxxxx xxxxxxxx xxxxxxxF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

- Modify Antonov's multi-target preimage search to find a WOTS+ public key that can sign this
  - Treat the part that signs xxxx… as prefix – so we know $sk_{i,0}$ for this part
  - Use the last block of the prefix and the part that signs FFFF… for herding and multitarget preimage search
  - Target the SHA-256 state immediately before the first block that signs checksum
  - The part that signs the checksum will come from the target honest signature
- Can forge a signature on any Merkle/FORS root of the above form as long as checksum works out

Making Sure the Checksum Works Out

- For a digest like:

  xxxxxxxx xxxxxxxx xxxxxxxF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

  - Checksum is $960 - 41 \cdot 15 - \sum x$
- We can increment, but not decrement, digits of honest checksum
  - Increment a digit by hashing (with prefix) sk_{i,d_i}
- Can choose targets with unusually small checksums
- Need $\sum x$ to be small enough with high enough probability

Batched Multicollision Search

- Best parameterization of our attack involves finding lots of 4-way collisions with distinct prefixes
- It is cheaper to search for lots of collisions at once
  - Finding a single 4-way collision costs $\sim 2^{192}$
  - Finding t 4-way collisions costs $\sim 2^{192} t^{1/4}$
  - (Ignoring prefixes and memory access costs)
- To get good memory access costs, use parallel collision search techniques
- To avoid wasting time colliding already-used prefixes
  - Compute collisions in smaller batches of size αt
- More detail in paper

Attack Complexity

Table 1. Summary of Our Results on SPHINCS$^+$ Category Five Parameters

Parameter Set	Cost				Reference
	Herd	Link	Signable	Total	
SPHINCS$^+$-256f	$2^{214.8}$	$2^{216.4}$	$2^{215.7}$	$\approx 2^{217.4}$	Section 4.3
SPHINCS$^+$-256s	$2^{214.8}$	$2^{216.4}$	$2^{215.7}$	$\approx 2^{217.4}$	Section 4.3

SPHINCS$^+$ Tweak [Hülsing 2022]

- In response to Antonov's attack on DM-SPR the SPHINCS$^+$ team issued a tweak to the SPHINCS$^+$ specification
  - Replaced SHA-256 with SHA-512, for hashing multi-block inputs in Category 3 and 5 parameters
  - Still some use of SHA-256, but doesn't seem exploitable

Conclusion

- Our attack shows that some submitted parameter sets of SPHINCS$^+$ are not as strong as claimed
- The problem is not the security proof for the SPHINCS$^+$ construction, but how its tweakable hash functions are instantiated
- Lesson: need to be very careful trying to get more than 128 bits of security from SHA-256

- On the upside:
  - SPHINCS$^+$'s proposed tweak seems to address these issues
  - SHA-256 on fixed-length inputs pretty reliably gets 128 bits of security, so it's unlikely this sort of oversight leads to a practical break

Thank You!

References

[Antonov 2022] Antonov, S.: *Round 3 official comment: SPHINCS+* (2022), https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/FVItvyRea28/m/mGaRi5iZBwAJ

[Hülsing 2022] Hülsing, A.: *Round 3 official comment: SPHINCS+* (2022), https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Ca4zQeyObOY

[Joux 2004] A. Joux, *Multicollisions in iterated hash functions. Application to cascaded constructions*, in ed. by M.K. Franklin. CRYPTO'04. Lecture Notes in Computer Science, vol. 3152 (Springer, 2004), pp. 306–316, https://www.iacr.org/archive/crypto2004/31520306/multicollisions.pdf

[Dean 1999] R.D. Dean, *Formal Aspects of Mobile Code Security*. Ph.D. thesis, Princeton University (January 1999)

[KS 2004] J. Kelsey, B. Schneier, *Second preimages on n-bit hash functions for much less than 2^n work*, in ed. by R. Cramer, Advances in Cryptology—EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494 (Springer, 2005), pp. 474–490, https://eprint.iacr.org/2004/304.pdf

[KK 2005] J. Kelsey, T. Kohno, *Herding hash functions and the nostradamus attack*, in ed. by S. Vaudenay. *EUROCRYPT*. Lecture Notes in Computer Science, vol. 4004 (Springer, 2006), pp. 183–200, https://eprint.iacr.org/2005/281