Hash-Based Direct Anonymous Attestation

Liqun Chen[1], Changyu Dong[2], Nada El Kassem[1],
Chris Newton[1], Yalan Wang[1]

[1] University of Surrey
[2] Guangzhou University

PQCrypto, August 2023

Trusted Computing and Attestation

- Trusted computing aims to achieve “zero trust but verify” for a computer system
- A computer system can be a single device or a network with multiple devices
- Verification is based on cryptographic mechanisms, including
  - Authentication
  - Authorization
  - Data confidentiality
  - Data integrity
  - Key management
  - ……
  - **Attestation**

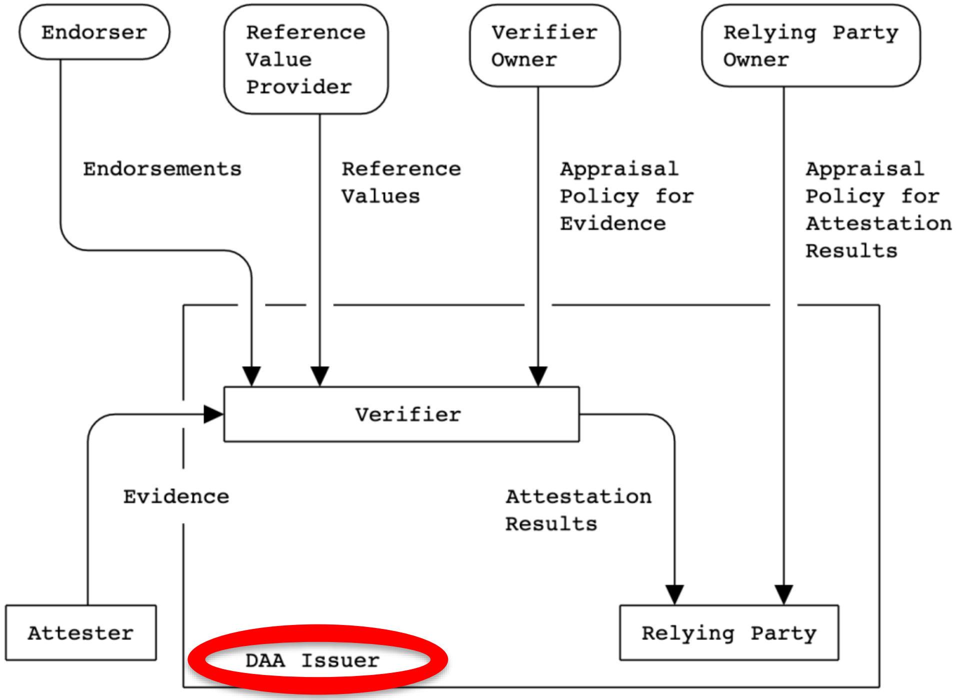
What is an Attestation Service?

```
************   *************    ************    *****************
* Endorser *   * Reference *    * Verifier *    * Relying Party *
************   * Value     *    *  Owner   *    *  Owner        *
     |         * Provider  *    ************    *****************
     |         *************          |                 |
     |                |               |                 |
     |Endorsements    |Reference      |Appraisal        |Appraisal
     |                |Values         |Policy           |Policy for
     |                |               |for              |Attestation
     .-----------.    |               |Evidence         |Results
                 |    |               |                 |
                 |    |               |                 |
                 v    v               v                 |
               .-------------------------.              |
       .------>|         Verifier        |------.       |
       |       '-------------------------'      |       |
       |                                        |       |
       |                             Attestation|       |
       |                             Results    |       |
       | Evidence                               |       |
       |                                        |       |
       |                                        v       v
 .----------.                              .-----------------.
 | Attester |                              | Relying Party   |
 '----------'                              '-----------------'
```

https://datatracker.ietf.org/doc/rfc9334/

- IETF RATS (Remote ATtestation procedureS) architecture
- An attester (prover) provides attestation evidence to a verifier and the evidence is **a digital signature** on the state of the attester's computer system
- Based on endorsements, reference values and evidence appraisal policy, the verifier provides an attestation result to a relying party
- Based on their appraisal policy, the relying party decides whether to accept or reject the result

Direct Anonymous Attestation (DAA)



https://datatracker.ietf.org/doc/html/draft-ietf-rats-daa-03
by Birkholz, Newton, Chen, and Thaler

- Turn a digital signature used for attestation into an anonymous signature, which is a group type of signature but
  - No traceability
  - User-controlled linkability, i.e., two signatures can be configured to show whether they from the same signer or not
- Prover receives a DAA credential from a privacy CA (DAA issuer)
- Given a DAA signature, Prover is anonymous to all entities, including the issuer
- Prover cannot abuse anonymity because of
  - Rogue key revocation
  - User-controlled link revocation

DAA History & State-of-the-Art

- DAA was needed by the Trusted Computing Group (TCG) for Trusted Platform Module (TPM) in 2003
- RSA-based DAA
  - Used in TPM 1.2 version
  - ACM CCS 2004 (This paper is received a test of time award at ACM CCS 2014)
- ECC-based DAA
  - Used in TPM 2.0 version
  - Support many other applications
  - There are many improvements
- Lattice-based DAA
  - Only a small number of schemes
  - Performance is not ideal
  - There is still much room for improvement

Various Signatures from Symmetric Primitives

- Traditional signatures from symmetric primitives
  - Hash-based signatures
    - One-time signatures
    - Few-time signatures
    - Stateful signatures
    - Stateless signatures
  - Picnic-style signatures

- Anonymous signatures
  - Ring signatures
  - Group signatures
  - Enhanced Privacy ID (EPID)

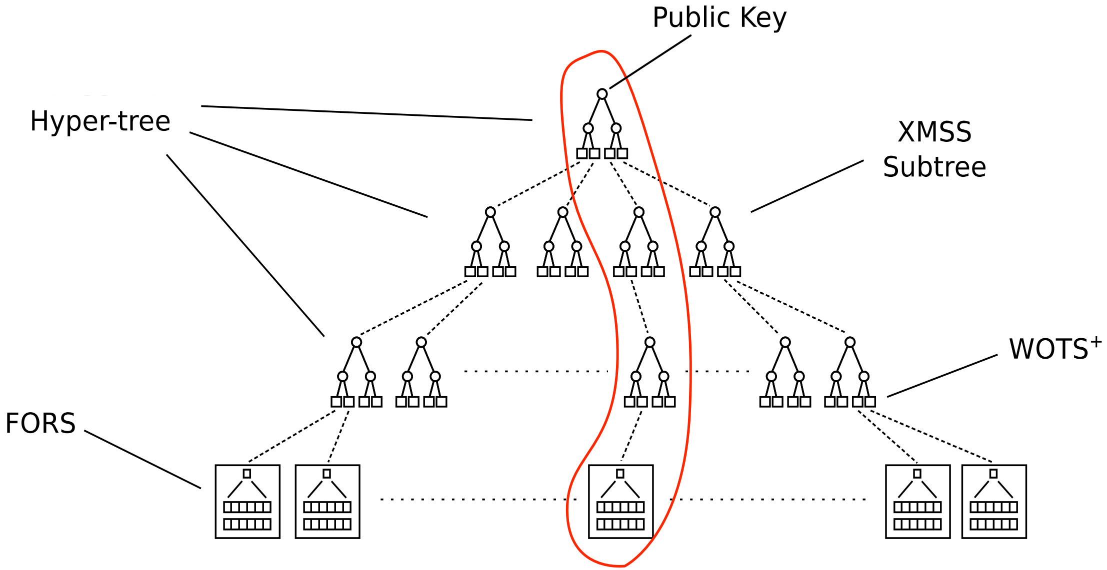
Challenges to DAA from Symmetric Primitives

- A DAA signer is split into two entities
  - A principal signer, TPM – a tamper-resistant root of trust
  - A semi-trusted assistant signer, software in the host computer
- Group size – the level of anonymity
  - The existing anonymous signatures use a Merkle tree, so the group size is small
  - We aim to have a big group size, up to 2^{60}
- Performance
  - Apart from usual performance requirements for digital signatures, we also need to make the TPM's workload as small as possible

Our Design Choices

- Modify SPHINCS+ to use as a DAA credential by
  - Modifying WOTS$^+$
  - Modifying FORS
- Use a Picnic-style of signature to provide Non-Interactive Zero-Knowledge Proof (NIZKP)
  - Mask all sensitive inputs and outputs
  - Use a partial proof for a better performance
- Chain multiple NIZKPs
  - Connect TPM’s NIZKPs with host’s NIZKPs
  - TPM only makes 5 Picnic-style signatures
  - Host proves the whole credential

SPHINCS+



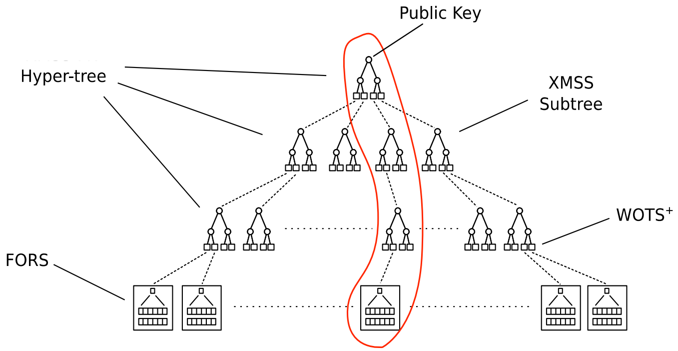
The SPHINCS+ signature scheme:

- A secret signing key is a seed that is used to create a hyper-tree
- The corresponding public verification key is the root value of the tree
- The hyper-tree consists of multiple XMSS-type subtrees
- A message to be signed is arranged as an entry to the tree
- A signature is the authentication path of the message on the tree

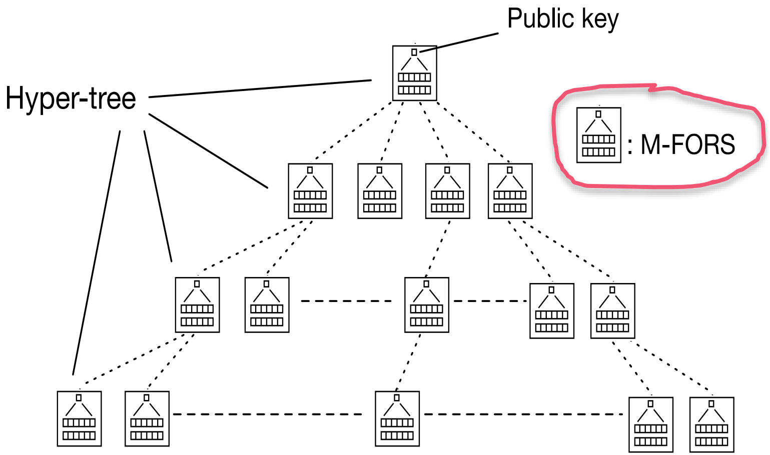
There were two difficulties if directly using SPHINCS+ to generate DAA credentials

- NIZK proof of FORS
  - FORS' top layer hash function is not scalable
  - Our solution: change this layer to a Merkle tree
  - We name the modified FORS by M-FORS
- NIZK proof of WOTS$^+$
  - Hiding the signed message for a WOTS$^+$ is not straightforward and very costly
  - Our solution: replace each subtree with an M-FORS tree

F-SPHINCS+ (Modified SPHINCS+)



SPHINCS+

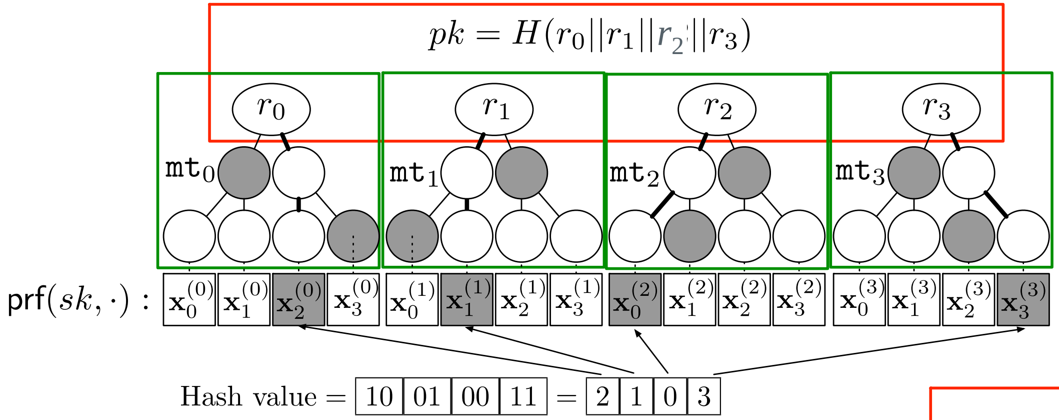


F-SPHINCS+

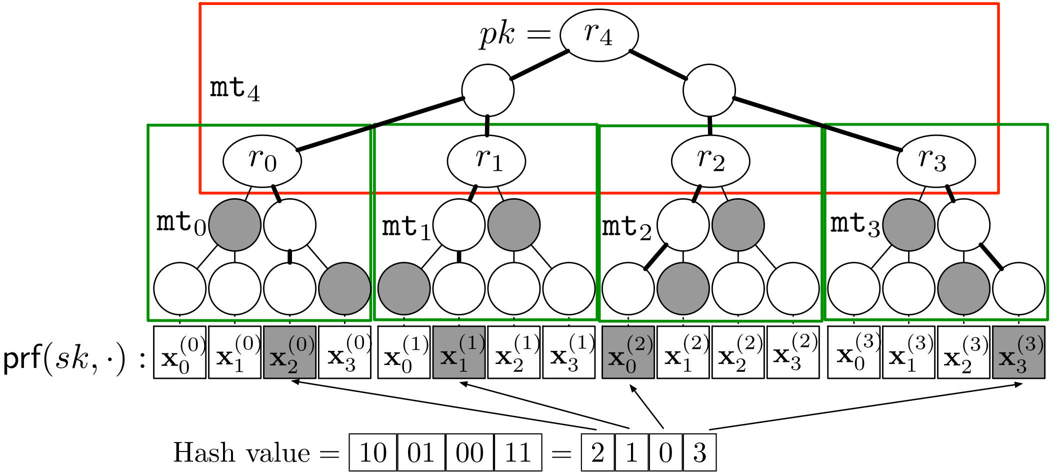
In the F-SPHINCS+ signature scheme:

- A secret signing key is a seed that is used to create a hyper-tree
- The corresponding public verification key is the root value of the tree
- A message to be signed is arranged as an entry to the tree
- A signature is the authentication path of the message on the tree
- The hyper-tree consists of multiple XMSS-type subtrees
- Each subtree is an M-FORS tree

M-FORS (Modified FORS)

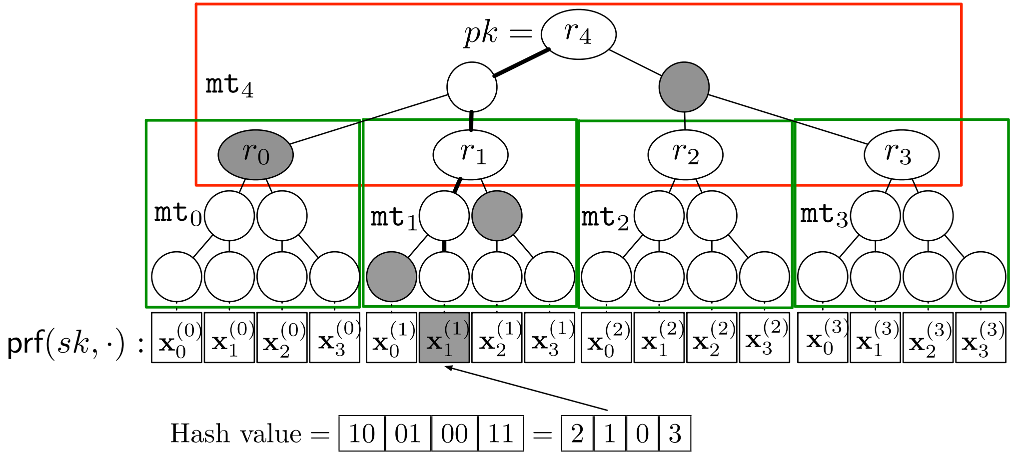


FORS



M-FORS

M-FORS Partial Proof



- A Picnic-style signature requires many runs in MPCitH
- Proving the possession of a DAA credential involves *h*+1 M-FORS signature verifications for an *h*-layer hyper-tree in F-SPHINCS+
- Our more efficient strategy is that in MPCitH, only a partial M-FORS signature is verified, i.e., one block in each run
- The proof of the Merkle tree authentication path guarantees that all the partial proofs are associated with the same tree

Split the Signer Role

Given $gsk_u = (sk_u, gr_u, \mathbf{S})$, rpk, msg, str and bsn, gid = H1(rpk), sid = H1($msg||str$), lid = H1(bsn)

TPM's signing workload

$$\pi_{\mathrm{D_T}} : \mathcal{P}\{(gp,\ sid,\ gid,\ lid,\ slt,\ hk,\ cet_u); (sk_u,\ sst,\ et_u)| \\ slt = F(sk_u,\ lid) \wedge sst = F(sk_u,\ sid) \wedge et_u = F(sk_u,\ gid) \\ \wedge hk = H_1(sst) \wedge cet_u = F(sst,\ et_u)\}$$

Host's signing workload

$$\pi_{\mathrm{D_H}} : \mathcal{P}\{(gp,\ rpk,\ slt,\ com,\ hk,\ cet_u); (et_u,\ sst,\ gr_u,\ \mathbf{S} = \{\sigma_h, \cdots, \sigma_0\})| \\ hk = H_1(sst) \wedge cet_u = F(sst,\ et_u) \wedge mt_u||idx = H_3(et_u||gr_u) \\ \wedge pk_h = \mathsf{recoverPK}(\sigma_h, mt_u, (n, d, k, (h, idx))) \\ \wedge pk_{h-1} = \mathsf{recoverPK}(\sigma_{h-1}, pk_h, (n, d, k, (h-1, \lfloor \frac{idx}{q} \rfloor))) \wedge \cdots \\ \wedge rpk = \mathsf{recoverPK}(\sigma_0, pk_1, (n, d, k, (0, 0))) \\ \wedge com = H_1(sst||pk_h|| \cdots ||rpk)\}$$

Verifier's point of view

$$\pi_{\mathrm{D}} : \mathcal{P}\{(gp,\ rpk,\ gid,\ sid,\ lid,\ slt,\ com); \\ (sk_u, et_u,\ sst,\ gr_u,\ \mathbf{S} = \{\sigma_h, \cdots, \sigma_0\})| \\ slt = F(sk_u,\ lid) \wedge sst = F(sk_u,\ sid) \wedge et_u = F(sk_u,\ gid) \\ \wedge mt_u||idx = H_3(et_u||gr_u) \\ \wedge pk_h = \mathsf{recoverPK}(\sigma_h, mt_u, (n, d, k, (h, idx))) \\ \wedge pk_{h-1} = \mathsf{recoverPK}(\sigma_{h-1}, pk_h, (n, d, k, (h-1, \lfloor \frac{idx}{q} \rfloor))) \wedge \cdots \\ \wedge rpk = \mathsf{recoverPK}(\sigma_0, pk_1, (n, d, k, (0, 0))) \\ \wedge com = H_1(sst||pk_h|| \cdots ||rpk)\}$$

A DAA signature is

$$\Sigma = (str,\ slt,\ com,\ \pi_{\mathrm{D}})$$

Security Proof

- Our proof follows the Universal Composability (UC) model for DAA

- The hash-based DAA scheme supports
  - Correctness
  - Anonymity
  - User-controlled linkability, using basenames
  - Unframeability
    - No adversary can create a signature w.r.t. a basename that links to another signature created by an honest TPM for the same basename
    - When the issuer and all TPMs are honest, no adversary can provide a signature on a message *msg* w.r.t. a basename *bsn* when no TPM signed this (*msg*, *bsn*) pair
    - When the issuer is honest, an adversary can only sign in the name of corrupt TPMs; if *n* TPMs are corrupt, the adversary can create at most *n* unlinkable signatures for the same basename

Conclusions

- We propose the first DAA scheme from symmetric primitives
  - It can support a large group size up to 2^{60}
  - It holds the DAA security properties under the UC model
- It makes use of two building blocks:
  - A hash-based signature as a DAA credential
  - A Picnic-style signature to prove the possession of that credential in a NIZK manner
- Performance is based on these two building blocks
  - If a TPM can support such a Picnic-style signature, a DAA signing requires the workload for 5 ordinary signatures
  - Improving the performance will be possible if either a more efficient stateless hash-based signature scheme than F-SPHINCS+ or an efficient Picnic-style signature scheme is developed
- This work is still in its early stages

Thank you!
Questions?

liqun.chen@surrey.ac.uk