Analysis of a Proposed Hash-Based Signature Standard

Jonathan Katz

Motivation and background

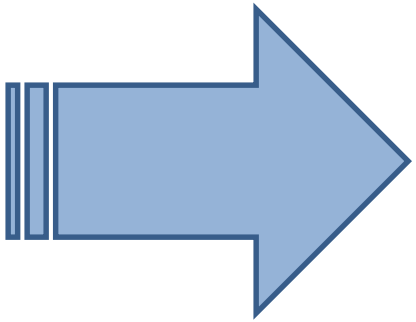
- Recent interest in standardization of "post-quantum" public-key primitives
- For signature schemes, several proposals based on cryptographic hash functions

- We study the *concrete security* of two versions of an Internet Draft by McGrew and Curcio
  - …in the random-oracle model

McGrew-Curcio proposals (10,000-ft view)

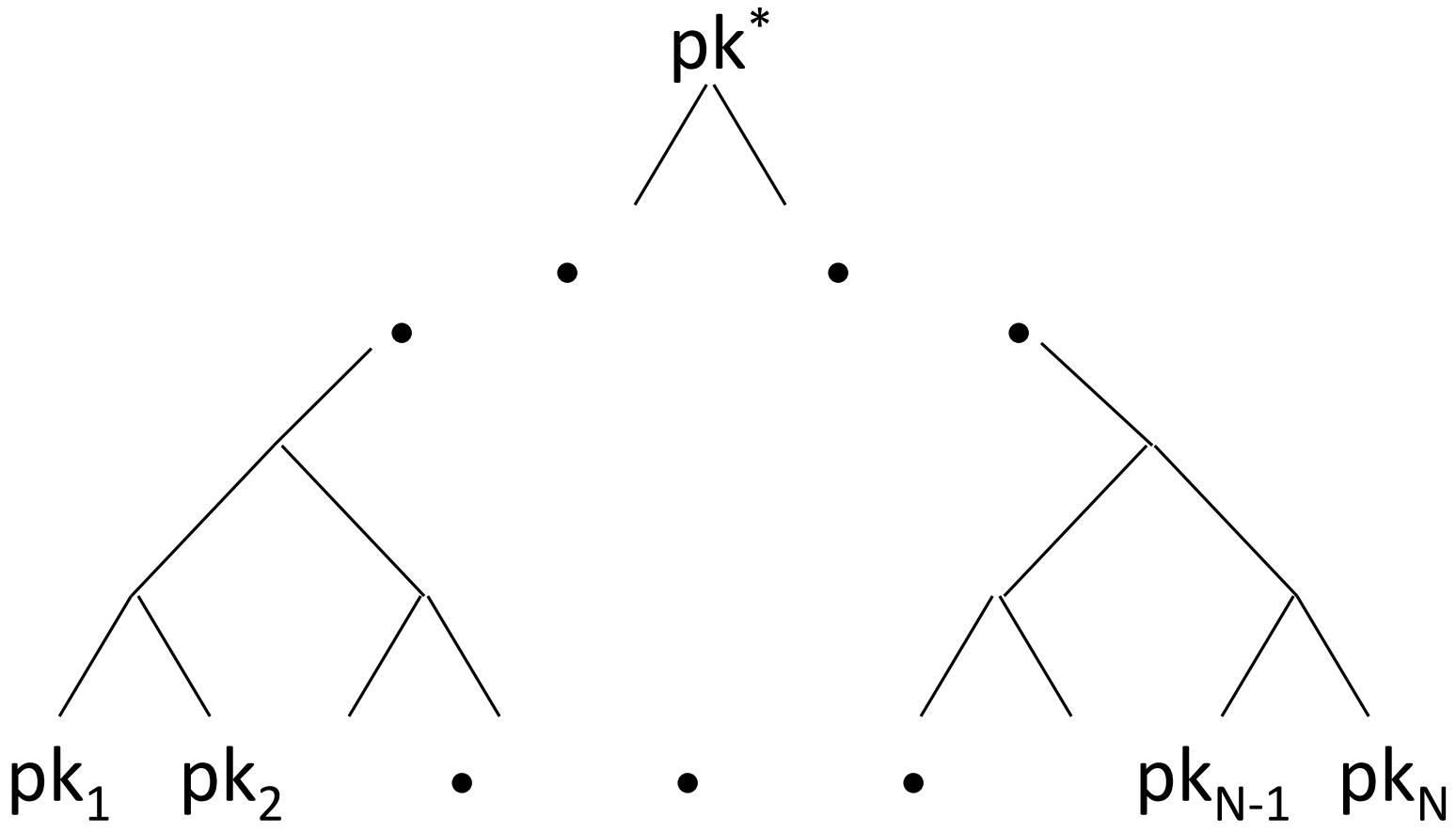
Merkle tree

1-time signature scheme [LDWM]



(stateful) many-time signature scheme

McGrew-Curcio proposals (10,000-ft view)



Key observation

- The scheme is composed of *multiple* instances of the 1-time scheme

  $\Rightarrow$ Concrete security of the scheme (even in a single-user setting) depends on concrete security of the 1-time scheme in the *multi-user setting*

Multi-user security

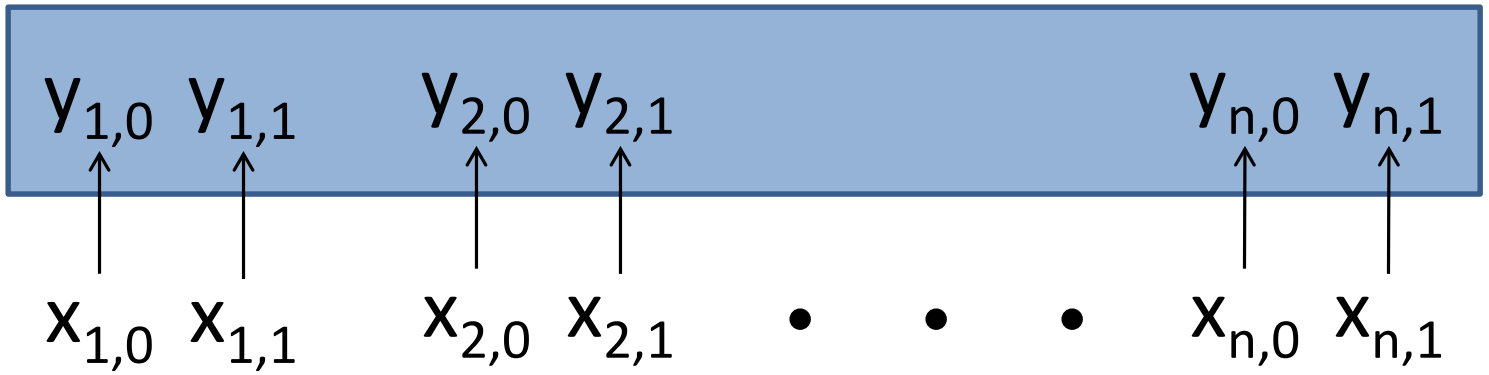
- [Bellare, Boldyreva, Micali],
  [Galbraith, Malone-Lee, Smart]
- Attacker given N (independent) public keys
  - Succeeds if it can forge a signature with respect to *any* of them
- If attacker can succeed with probability $\leq \varepsilon$ when attacking one scheme, can succeed with probability $\leq N \cdot \varepsilon$ when attacking N schemes
  - Is a tighter reduction possible?

Our results

- An initial version of the McGrew-Curcio draft (v02, 2014) has only a “loose” reduction
  - Because the 1-time scheme used has only a loose reduction in the multi-user setting
- An updated version of the McGrew-Curcio draft (v04, 2016) has a tight reduction
  - Even in the multi-user setting

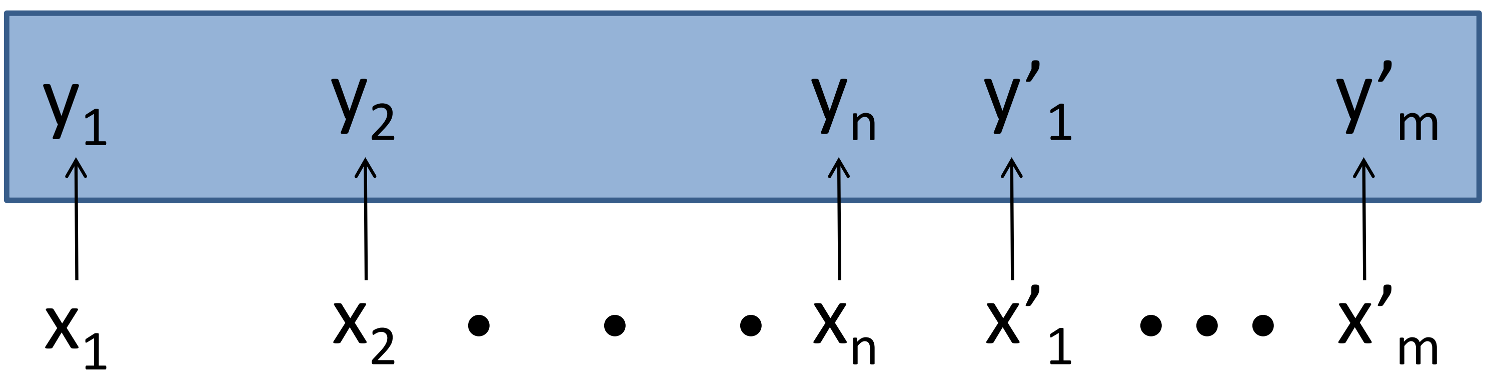
The LDWM 1-time scheme (v02)

Lamport's scheme



Sign(01…1) = $x_{1,0}$, $x_{2,1}$, …, $x_{n,1}$

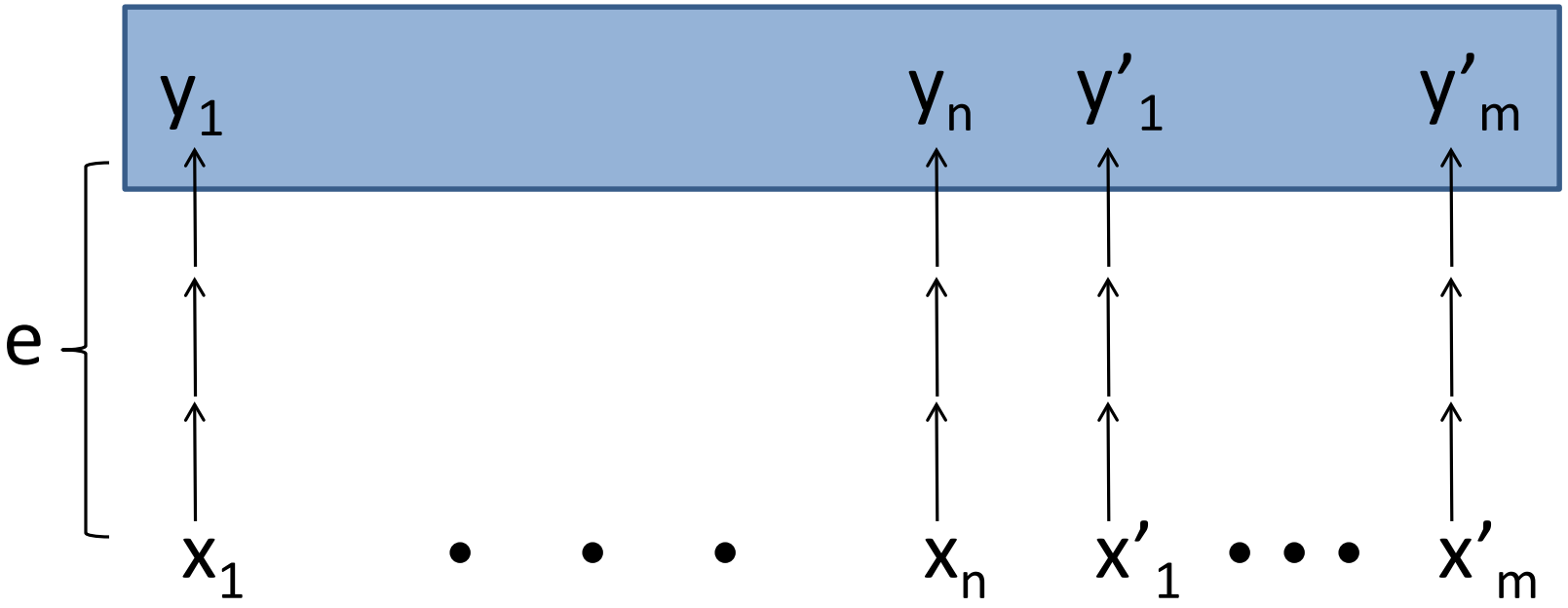
Improvement I



Sign(01…1) Sign(checksum(01…1))

Signature length n + log n

Improvement II



Public key/signatures compressed by log e;
signing/verification time increases by O(e)

“Trivial” improvements

- Sign H(M) rather than M
- Set pk = H($y_1 \ldots y_m$) instead of $y_1 \ldots y_m$

Security analysis?

- Let q be the number of H-queries made by the attacker, and t be the output length of H
- Forging a signature given pk_1, …, pk_N
  - Find M, M' with H(M)
    - Success probability O(
  - Compute $y^*_1=H^e(x^*_1)$, …, $y^*_Q=H^e(x^*_Q)$ and find j, i_1, …, i_m such that p
    - Success probabi
  - Find x^* such that
    - Success probabi

Would like to avoid birthday attack, also

Loose security in the multi-user setting!

Note…

- Security of the many-time scheme (even in the single-user setting) cannot be better than multi-user security of the 1-time scheme

The LDWM 1-time scheme (v04)

Key ideas

- Use *domain separation* so every invocation of H is on a distinct domain [Leighton, Micali]

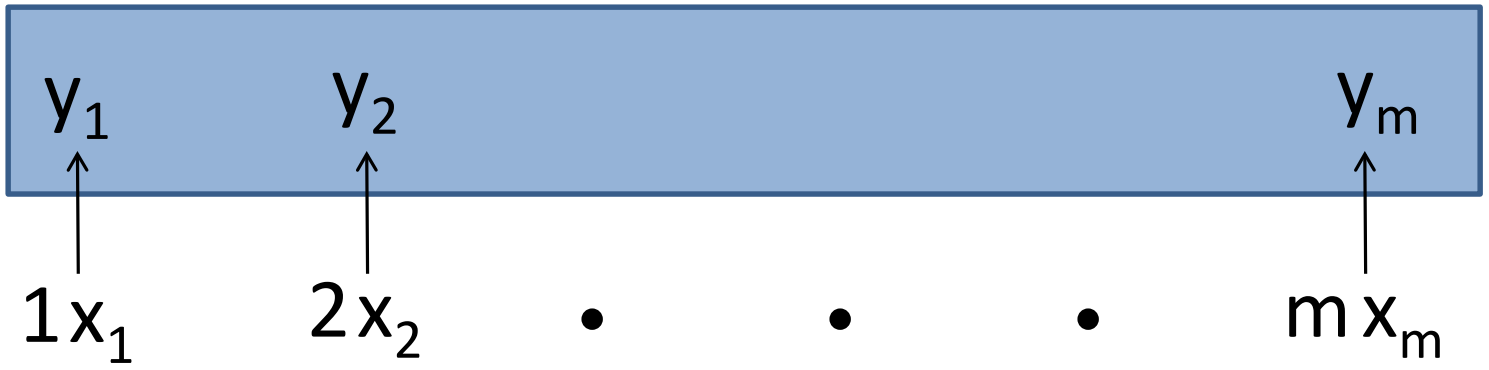
  $\Rightarrow$ Each H-query of the attacker can be "charged" to *at most one step* of key generation/signing

- Per-key identifier/diversification factor to ensure domain separation *for different keys*

  $\Rightarrow$ Each H-query of the attacker can be "charged" to $\leq$ 1 step of key generation *for at most one public key*

- Use "salted" hash to prevent birthday attack

Domain separation



Identifier/diversification factor

- When keys are generated by *multiple* users
  - Identifiers can be based on users' identities
  - Can also incorporate random values unlikely to repeat across (honest) users
- When multiple keys are generated by *one* user
  - Identifier can be based on identity
  - Diversification factor can be based on sequence numbers to ensure distinctness

Security theorem

- As long as identifiers/diversification factors are distinct across all keys, attacker's success probability is at most $3q/2^t$
  - Regardless of the number of keys!
- Proof by case analysis and probabilistic arguments treating H as a random oracle

The many-time scheme (v04)

Key generation (high level)

- Generate N keys for the 1-time scheme
  - Using a distinct diversification factor each time
- Construct a Merkle tree over those N keys
  - Ensuring domain separation at each node
  - Ensures that each H-query of the attacker can be "charged" to *at most one node* of the tree

Security theorem

- Attacker's success probability is at most $3q/2^t$
  - Holds for multi-user setting as well

Summary

- Signature scheme in an initial version of the McGrew-Curcio draft does *not* admit a tight security reduction
  - Since the underlying 1-time signature does not admit a tight reduction in the multi-user setting
- Modified scheme in a later version of the draft *does* admit a tight security reduction to the underlying hash function
  - Even in the multi-user setting

Questions?