Merkle Tree Ladder Mode: Reducing The Size Impact Of NIST PQC Signature Algorithms In Practice

Dr. Burt Kaliski Jr. (joint work with Andrew Fregly, Joseph Harvey, Swapneel Sheth)

Senior Vice President and Chief Technology Officer, Verisign

Video courtesy of RSA Conference

Introduction

NIST PQC Project has resulted in a remarkable variety of post-quantum algorithms

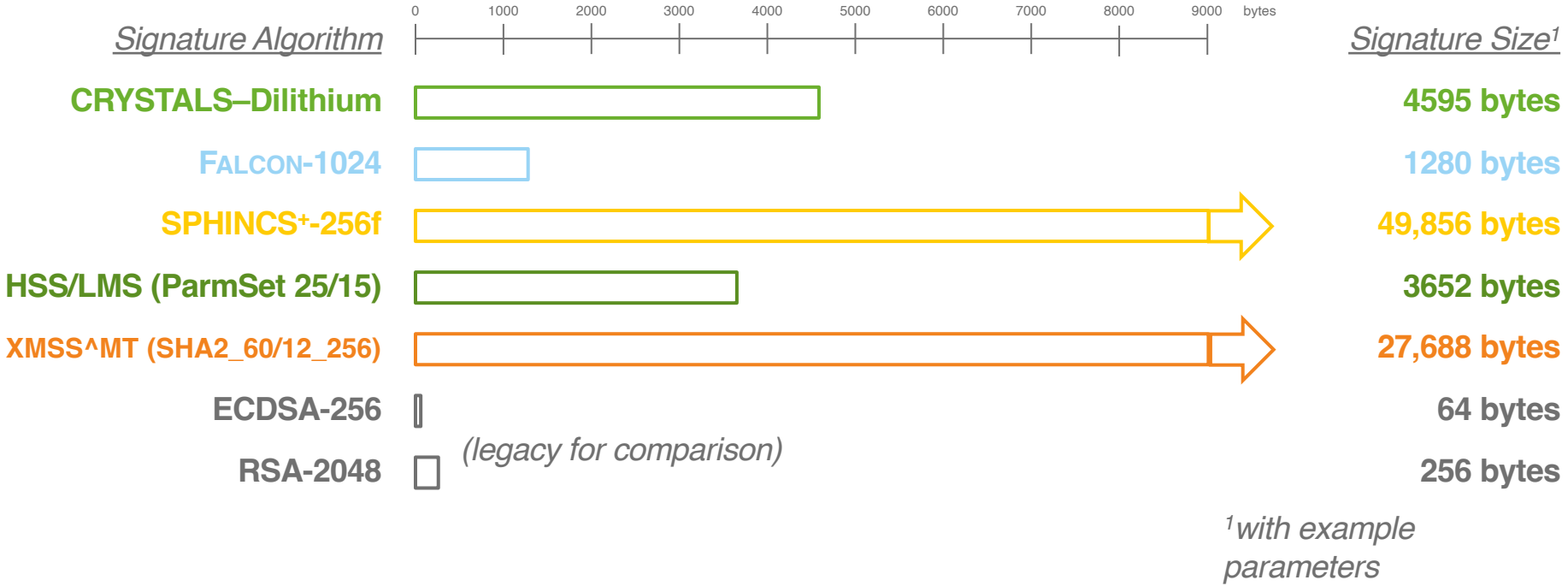
… but sizes of selected signature algorithms are large!

Large size may impact legacy applications with size constraints (e.g., DNSSEC)

How to reduce impact in practice?

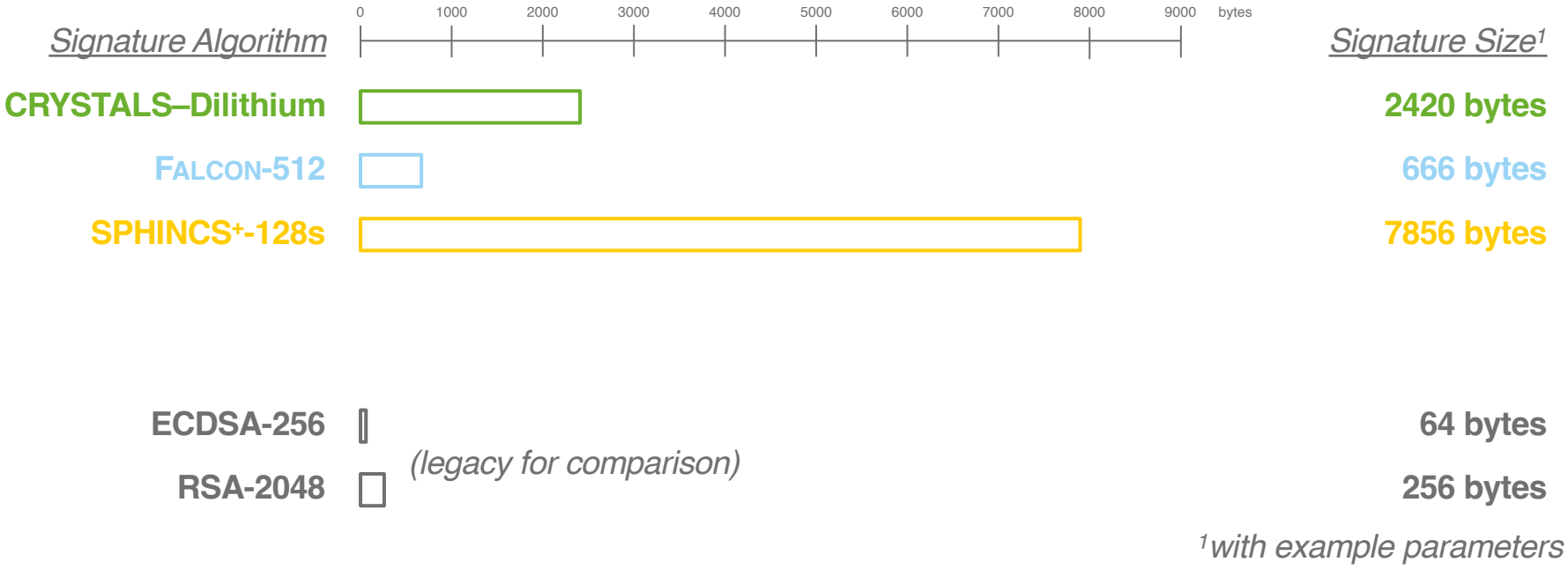
NIST PQC Signature Algorithms: Large Examples

Level V Parameters (256-bit security)



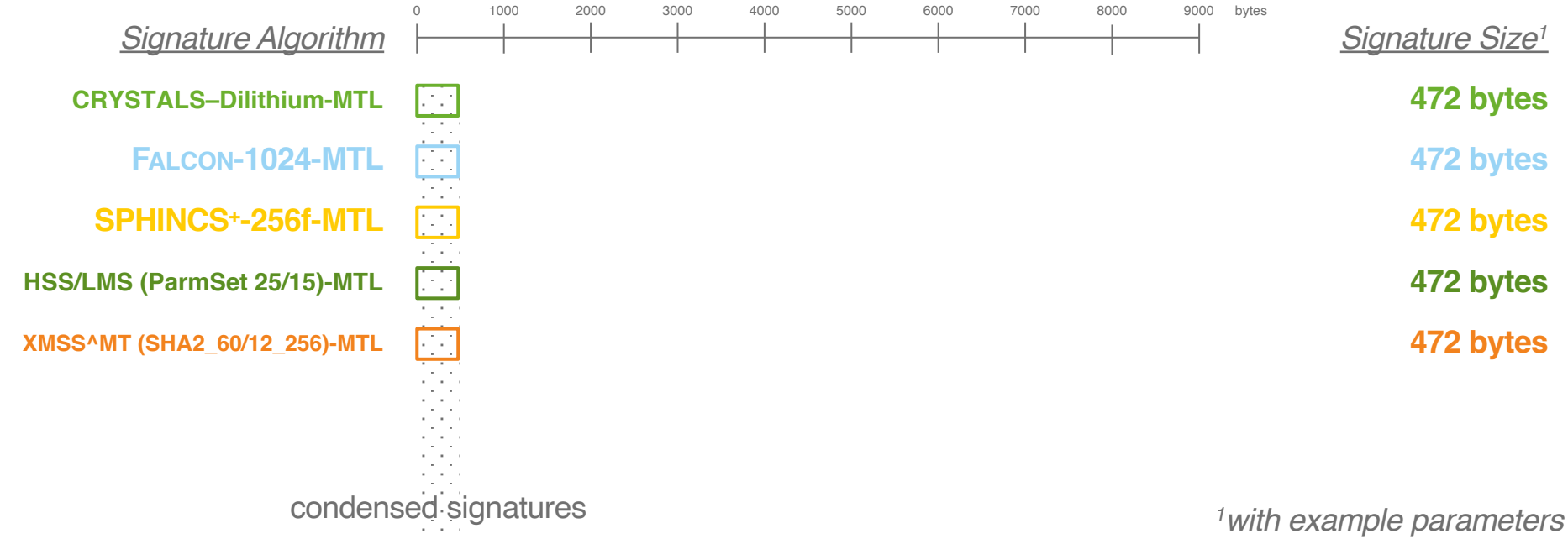
NIST PQC Signature Algorithms: Small Examples

Level I or II Parameters (128-bit+ security)



Merkle Tree Ladder Mode

Much Shorter… But How?

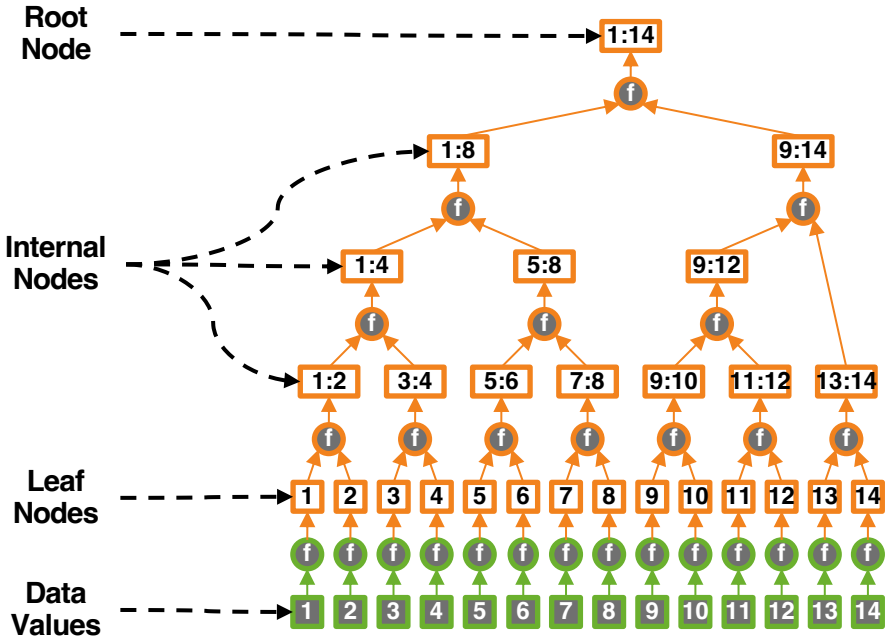


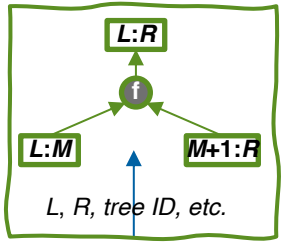
[1] *with example parameters*

Merkle Tree Ladders

Merkle Tree

Root Node Recursively Authenticates Data Values

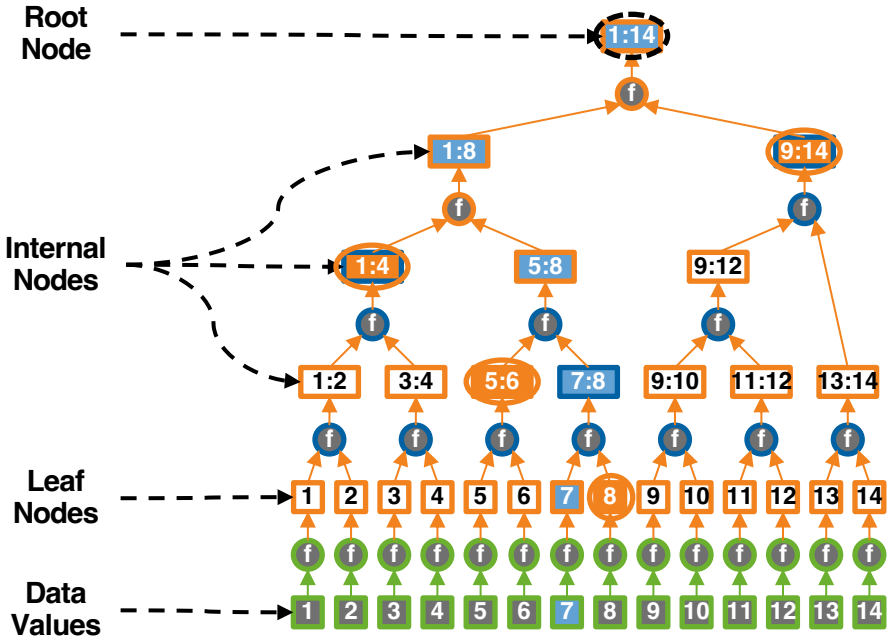




- *Merkle (1979)*
- *Not limited to power of 2*

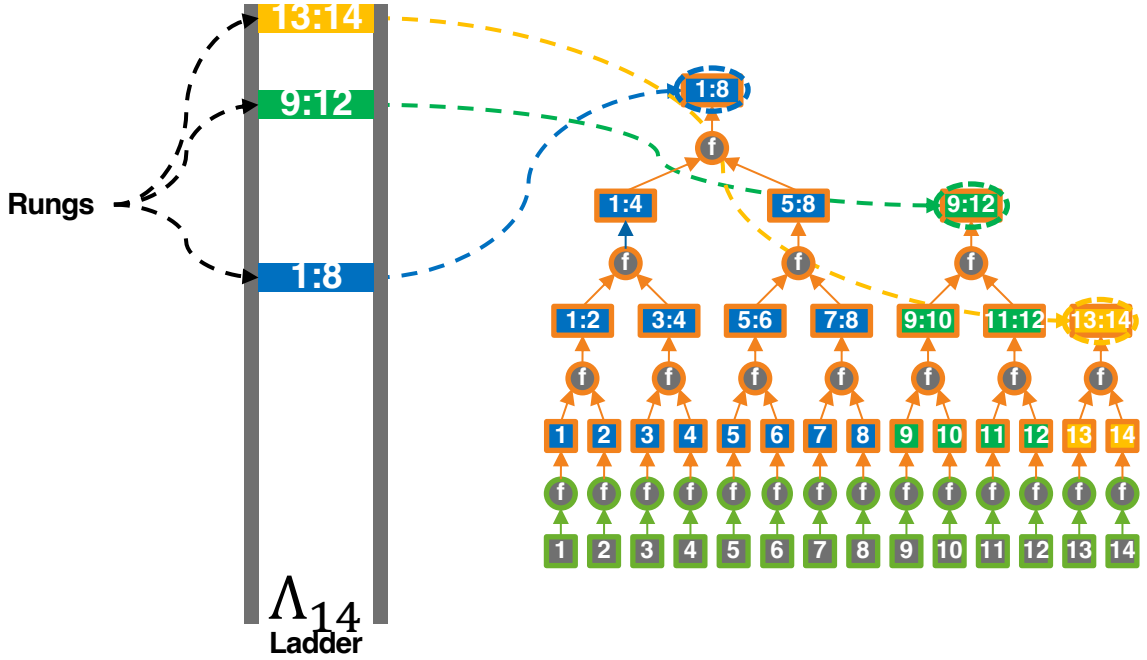
Authentication Path

Verify Data Value by Re-Hashing with Sibling Nodes



Merkle Tree Ladder

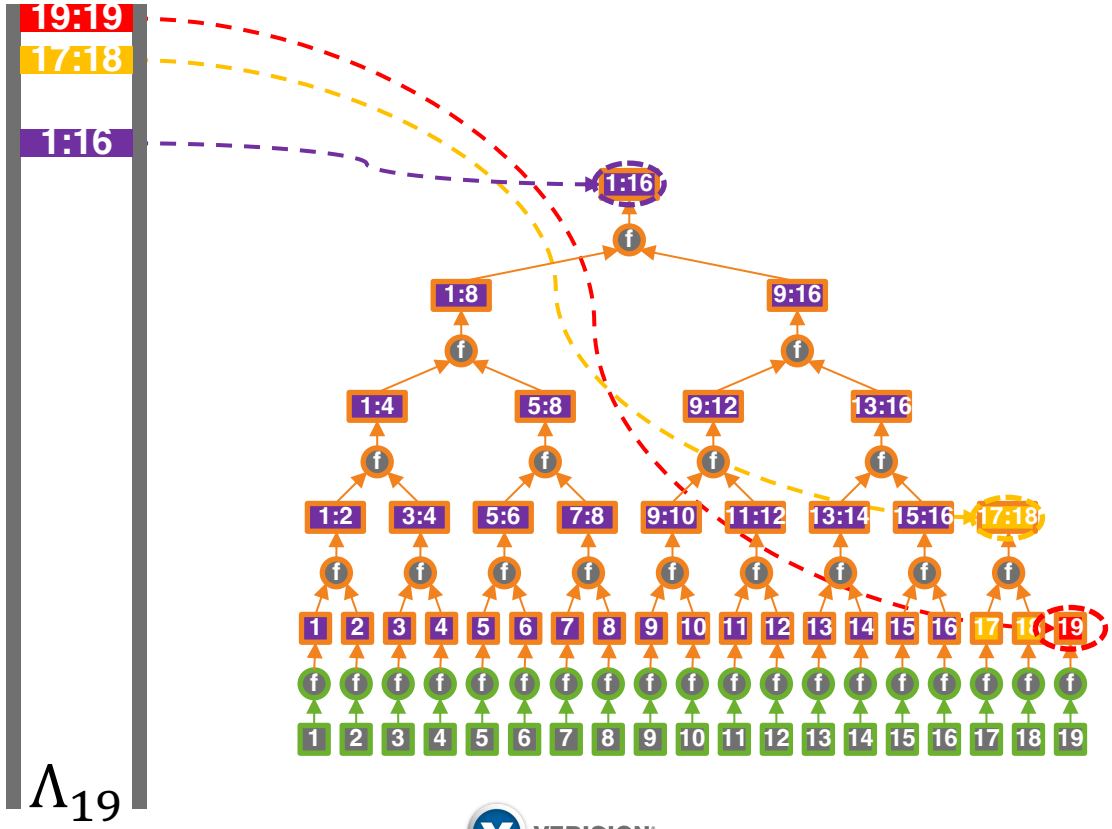
"Rungs" Collectively Authenticate All Data Values



- *"Binary" rung strategy shown; others possible*
- *Also called "binary numeral tree" (Champine 2021); "history trees" (Crosby-Wallach 2009), "Merkle Mountain Ranges" (Todd 2012)*

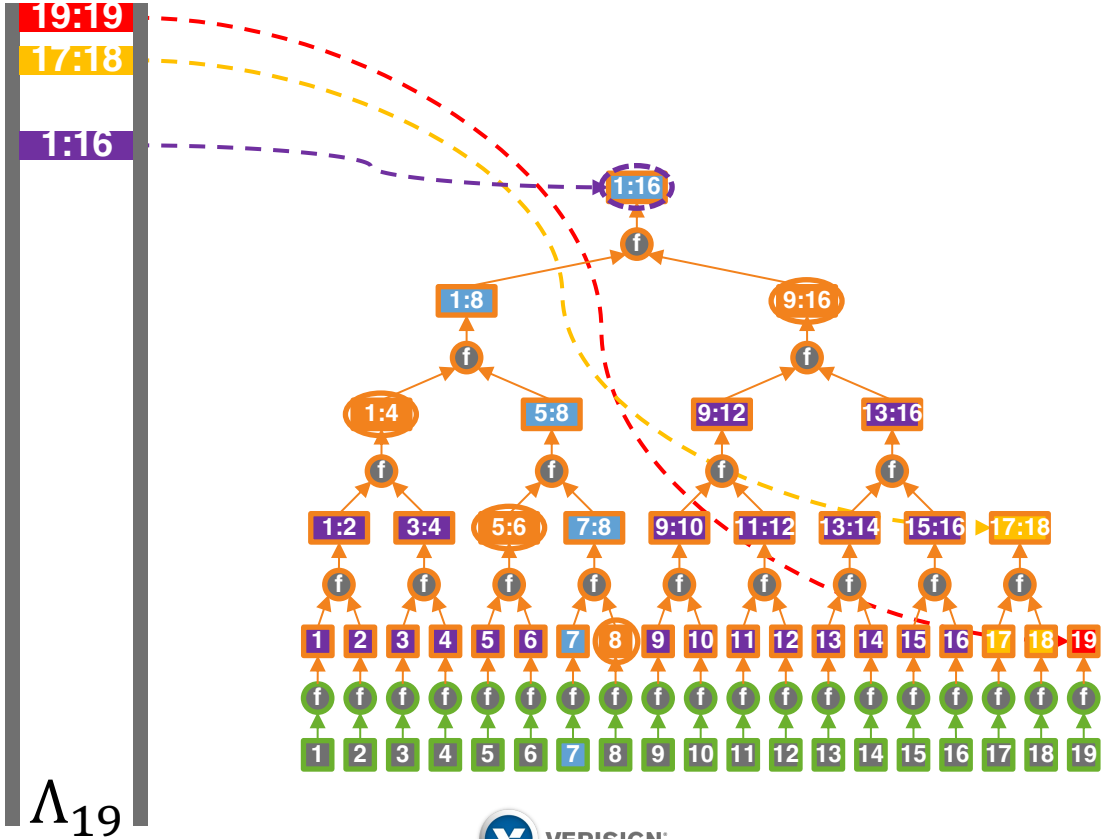
Ladder Evolution

Rungs Updated As New Data Values are Added



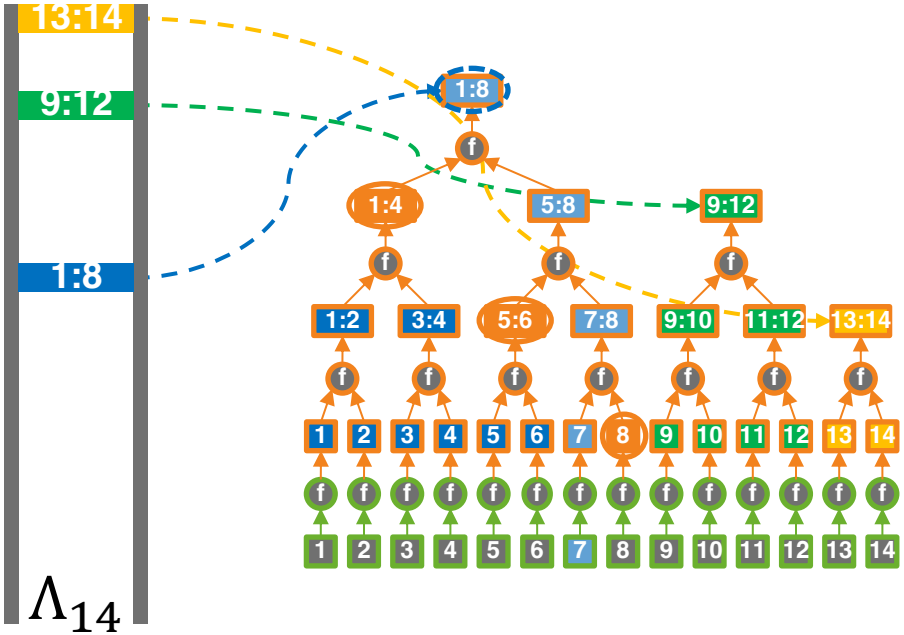
Authentication Path to Ladder

Verify Data Value by Re-Hashing with Sibling Nodes



Backward Compatibility

Authentication Path Can Be Verified Using Old Ladder

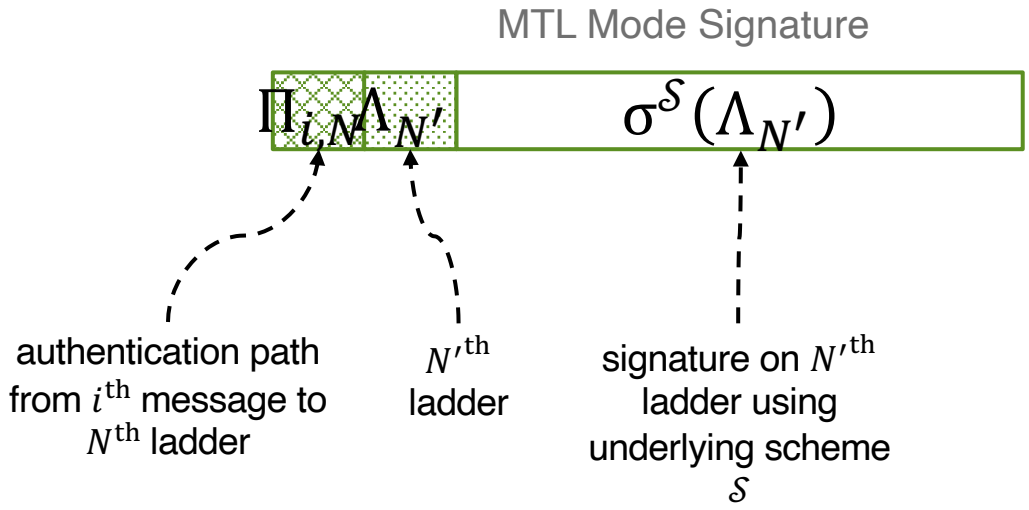


Analogous to "old-accumulator compatibility" (Reyzin-Yakoubov 2016)

Merkle Tree Ladder Mode

MTL Mode of Operation

Authentication Path + Ladder + Signature on Ladder



Signature Generation
produces initial signatures with
$i = N' = N$

Signature Verification
accepts signatures with
$i \leq N' \leq N$
verifiable via backward compatibility
(MTL mode is malleable)

MTL Mode: Evolve Trees, Sign Ladders

$\mathcal{S}$- MTL : *Stateful Transformation of Scheme $\mathcal{S}$*

Key Pair Generation

1. Generate $\mathcal{S}$ key pair
2. Add random series identifier to key pair
3. Add empty tree to private key

Signature Generation

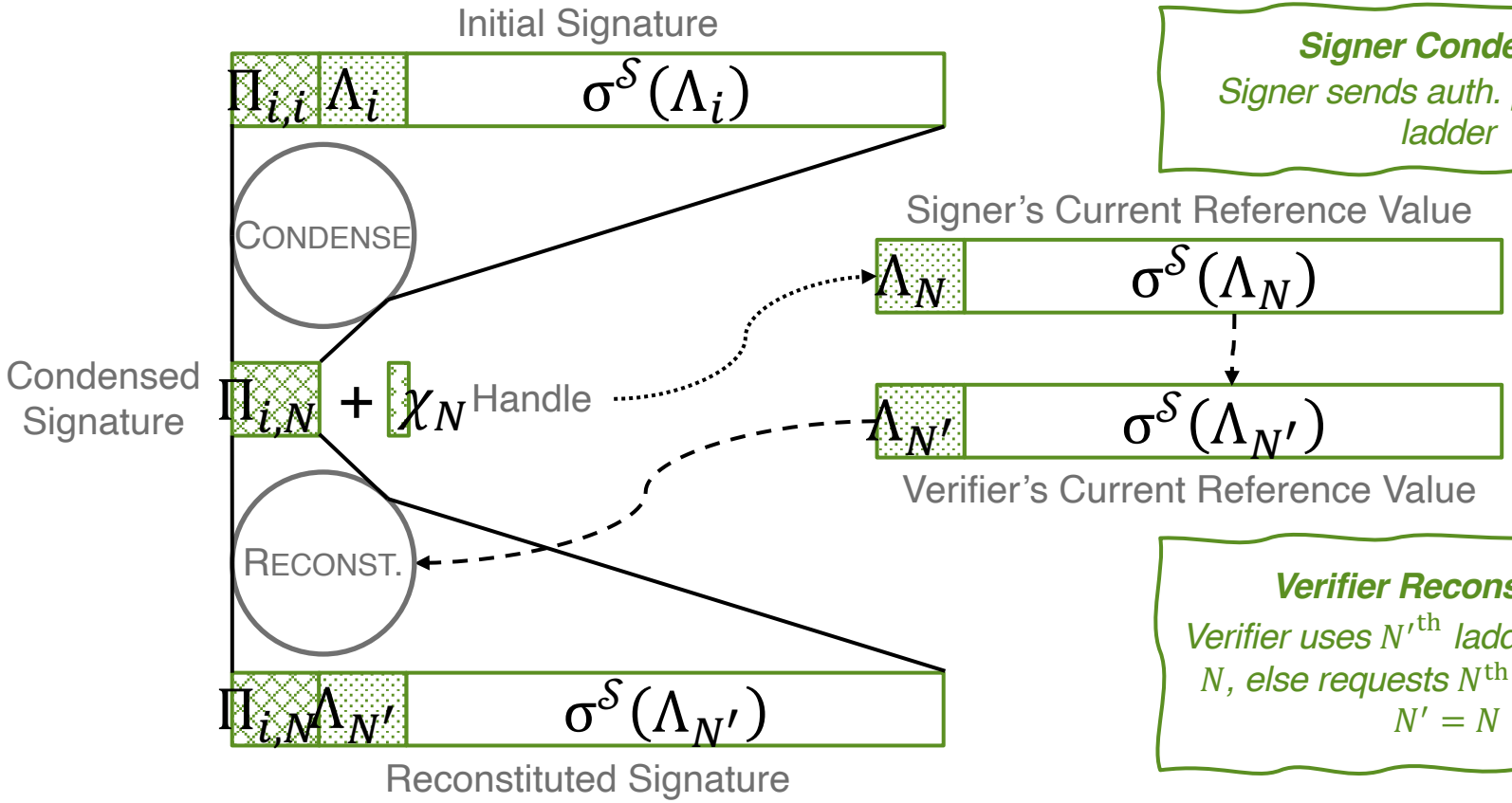
1. Hash message with randomizer → data value
2. Add data value to tree → new ladder
3. Sign ladder with $\mathcal{S}$
4. Form signature from randomizer + auth. path to ladder + ladder + $\mathcal{S}$ sig.

Signature Verification

1. Parse signature
2. Verify $\mathcal{S}$ sig. on ladder
3. Verify auth. path to ladder
4. Verify randomized hash on message

Multi-Target Design
All hash inputs include series identifier + message / node index

Condensing and Reconstituting MTL Mode Sigs.



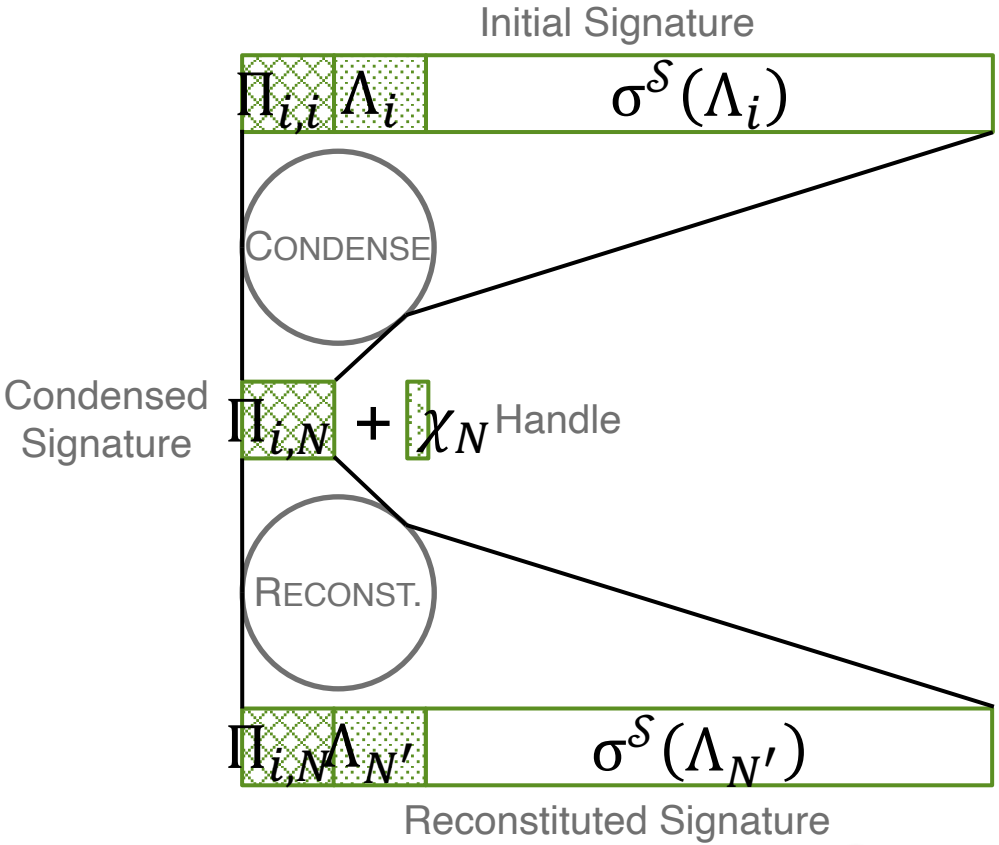
Signer Condenses

Signer sends auth. path to N^{th} ladder

Verifier Reconstitutes

Verifier uses N'^{th} ladder if $i \leq N' \leq N$, else requests N^{th} ladder, sets $N' = N$

Condensing and Reconstituting MTL Mode Sigs.



Anyone can condense *given access to signatures*

Condensation & reconstitution only involve public information → *don't impact security of MTL mode*

Anyone can reconstitute *given access to reference values*

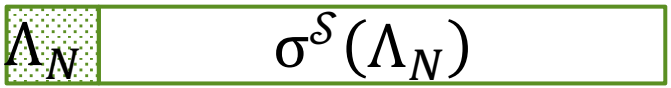
MTL Mode Signatures: Example Sizes (in bytes)

$N \approx 10{,}000$, *256-Bit Hash Output, 32-Bit Indexes*

Full Signature

Condensed Signature

Reference Value



- Ladders ≤ 14 hash values
- Authentication paths ≤ 13 hash values
- Full signature (also incl. series identifier, randomizer, i, N, N' and data value):

$$13{\times}32 + 14{\times}32 + 16 + 32 + 4 + 4 + 4 + 32 = 956 + \text{size of } \mathcal{S} \text{ sig.}$$

- Condensed signature (also incl. series identifier, randomizer, i, N'):

$$13{\times}32 + 16 + 32 + 4 + 4 = 472$$

- Reference value (also incl. N):

$$14{\times}32 + 4 = 452 + \text{size of } \mathcal{S} \text{ sig.}$$

Security Proofs

Security Against Existential Forgery

Proof Sketch

Forgery Cases

- $\mathcal{S}$ forgery
- auth. path second preimage

  (multi-target / multi-function)
- randomized hash second preimage

  (nonce extended-target)

Signature Verification

1. Parse signature
2. Verify $\mathcal{S}$ sig. on ladder
3. Verify auth. path to ladder
4. Verify randomized hash on message

- *Follows XMSS-T definitions (Hülsing et al. 2016)*

Multi-Target Design

All hash inputs include series identifier + message / node index

Random Oracle Model Bounds

Against Classical Adversaries

$$\mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}\text{-MTL}) \leq \mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}) + \frac{(q+1)}{2^{\ell}} + \frac{q}{2^{\ell_c}}$$

where

- q is number of hash queries
- ℓ is size of hash output in bits
- ℓ_c is size of randomizer in bits
- randomized hash & hash function families for tree leaves & internal nodes are modeled as random oracles

Motivated by security proofs for LMS (Katz 2016; Fluhrer 2017)

MTL Mode: Evolve Trees, Sign Ladders

$\mathcal{S}$-MTL : *Stateful Transformation of Scheme $\mathcal{S}$*

Hash Function Instantiation

- "program" pseudorandom function oracle to embed second preimage

Signature Generation

1. Hash message with randomizer → data value
2. Add data value to tree → new ladder
3. Sign ladder with $\mathcal{S}$
4. Form signature from randomizer + auth. path to ladder + ladder + $\mathcal{S}$ sig.

Motivated by XMSS-T (Hülsing et al. 2016) and SPHINCS⁺ (2022) designs

Mostly Standard Model Bounds

Against Classical Adversaries

$$\mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}\text{-MTLr}) \leq$$
$$\mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}) + \mathrm{InSec}^{\text{MM-SPR}}(H'_{\text{leaf}}) +$$
$$\mathrm{InSec}^{\text{MM-SPR}}(H'_{\text{int}}) + \frac{(q+1)}{2^{\ell}} + \frac{q}{2^{\ell_c}}$$

where

- q is number of hash queries
- ℓ is size of hash output in bits
- ℓ_c is size of randomizer in bits
- H'_{leaf}, H'_{int} are hash function families for tree leaves & internal nodes
- randomized hash & pseudorandom functions are modeled as random oracles

Mostly Standard Model Bounds
Against Quantum Adversaries

$$\mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}\text{-}\,\mathrm{MTLr}) \leq$$
$$\mathrm{InSec}^{\text{EU-CMA}}(\mathcal{S}) + \mathrm{InSec}^{\text{MM-SPR}}(H'_{\text{leaf}}) +$$
$$\mathrm{InSec}^{\text{MM-SPR}}(H'_{\text{int}}) + \frac{8(q+q_s+2)^2}{2^{\ell}} + 3q_s\sqrt{\frac{q+q_s+1}{2^{\ell_c}}}$$

where terminology is as above plus

- q_s is number of signature queries

Based on (Grilo et al. 2021) improvement to (Bos et al. 2020) for hash-and-sign with nonce

Bit Security of MTL Mode

Incremental to Underlying Signature Scheme

Size of Hash Output ℓ	Size of Randomizer ℓ_c	Number of Signatures q_s	Classical Security (bits)	Quantum Security (bits)
256	256	2^{20}	253	125
		2^{60}		
	259	2^{64}		
128	128	2^{20}	125	61
		2^{60}		

Double-Signing Impact in MTL Mode

- **Chosen State Attacks:** Suppose adversary can control MTL mode state, get signatures on multiple messages with same index i
- If $\mathcal{S}$ is LMS, XMSS: one-time private key material revealed → security *compromised*
- If $\mathcal{S}$ is Dilithium, FALCON, SPHINCS$^+$: multiple hash targets with same index → security gradually *degraded*
- **Reimagine the State:** MTL mode uses state to bring benefits (condensability, etc.) without the usual risk

Practical Impact

Ladder Endurance: General Model

How Many Signatures Until Next Reference Value?

N_0 initial messages

α new messages signed per “iteration”

ρ messages randomly selected to verify per iteration — with current condensed sig. $\Pi_{i,N}$

Verifier initially has reference value with ladder Λ_{N_0}

How many messages until verifier needs new reference value, i.e., $i > N_0$?

Ladder Endurance: General Model, cont'd

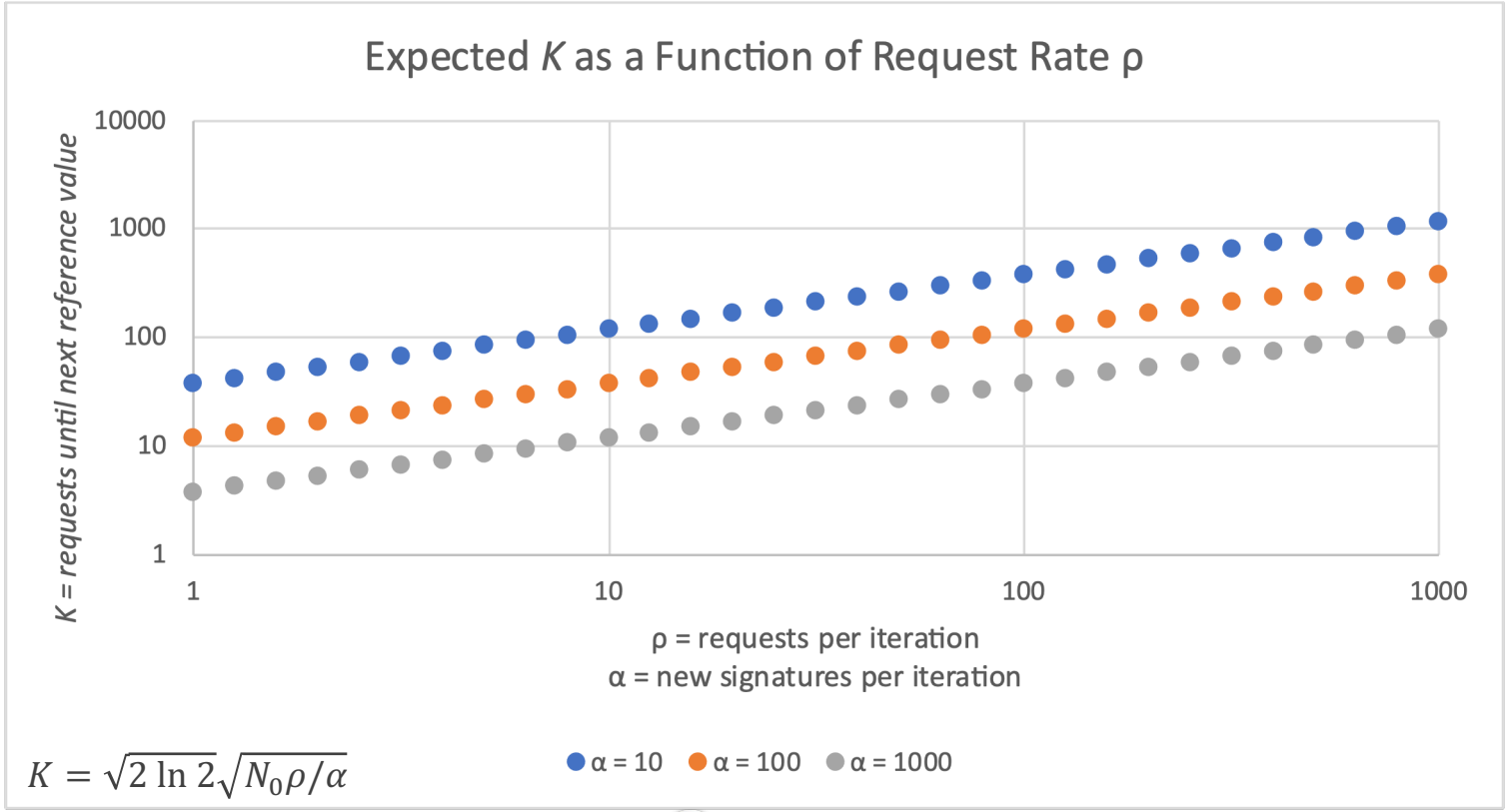
How Many Signatures Until Next Reference Value?

- $\text{Prob}[\kappa \text{ "good" iterations}] \approx \prod_{t=1}^{\kappa} \left(\frac{N_0}{N_0+t\alpha}\right)^{\rho} \approx \exp\left(-\frac{\kappa^2 \alpha\rho}{2N_0}\right)$

- Probability $\approx 1/2$ when $\kappa \approx \sqrt{2 \ln 2}\sqrt{N_0/\alpha\rho}$
- Endurance: $\sqrt{2 \ln 2}\sqrt{N_0/\alpha\rho}$ iterations → $K = \sqrt{2 \ln 2}\sqrt{N_0\rho/\alpha}$ messages

Expected Ladder Endurance

w/Request Rate ρ, Update Rate α; Assume $N_0 = 10{,}000$



$$K = \sqrt{2 \ln 2}\sqrt{N_0 \rho / \alpha}$$

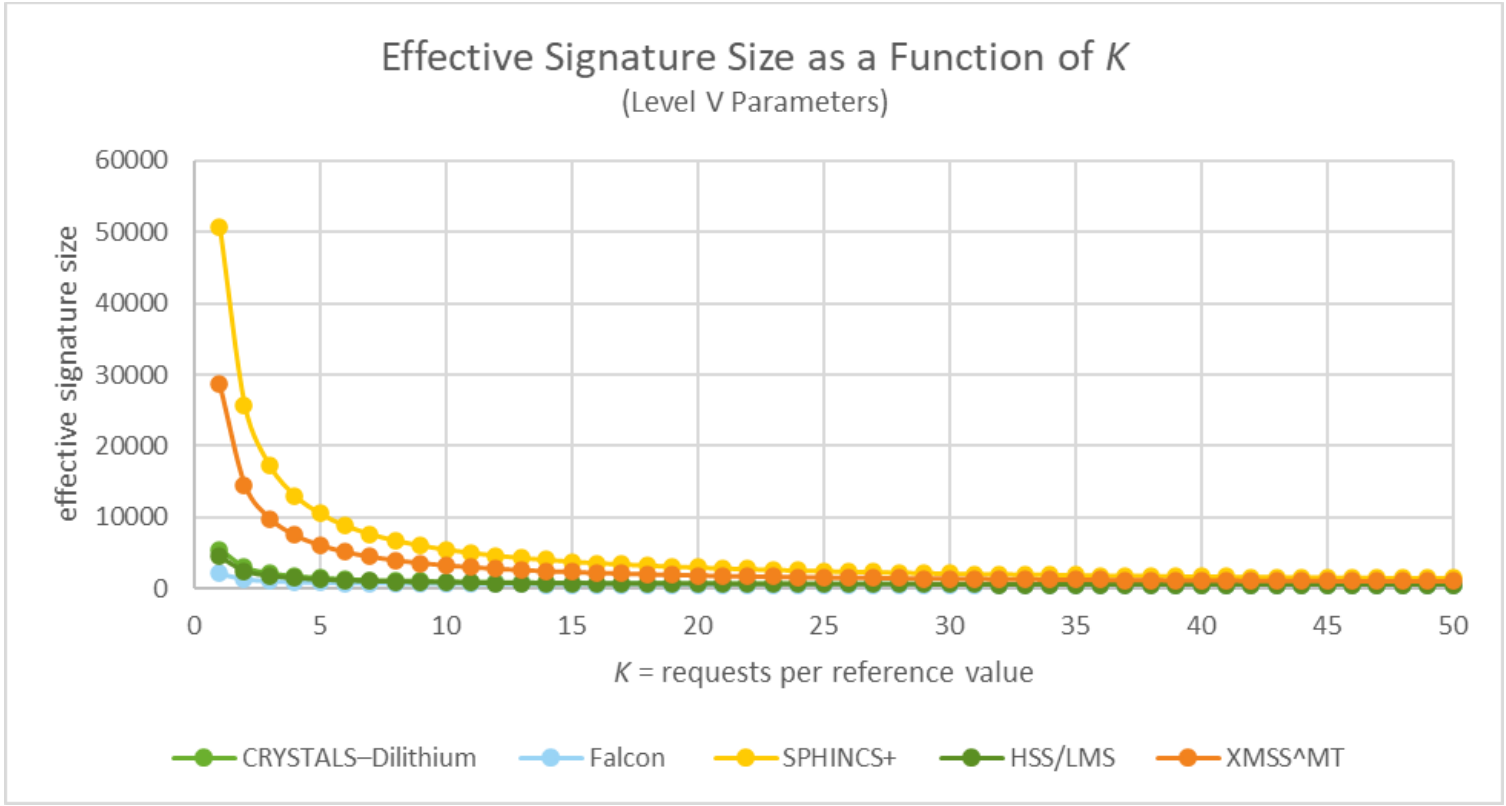
Effective Signature Size for Large Examples

MTL Mode Parameterized for ~10,000-Message Series

Signature Algorithm (NIST Level V)	Underlying Signature Size (bytes)	Condensed MTL Mode Signature Size $\lvert\varsigma\rvert$	Reference Value Size $\lvert\upsilon\rvert$	Effective MTL Mode Signature Size ($K = 10$)
CRYSTALS-Dilithium	4595	472	5047	976.7
FALCON-1024-MTL	1280	472	1732	645.2
SPHINCS⁺-256f-MTL	49,856	472	50,308	5502.8
HSS/LMS (ParmSet 25/15)-MTL	3652	472	4104	882.4
XMSS^MT (SHA2_60/12_256)-MTL	27,688	472	28,140	3286

Effective Signature Size

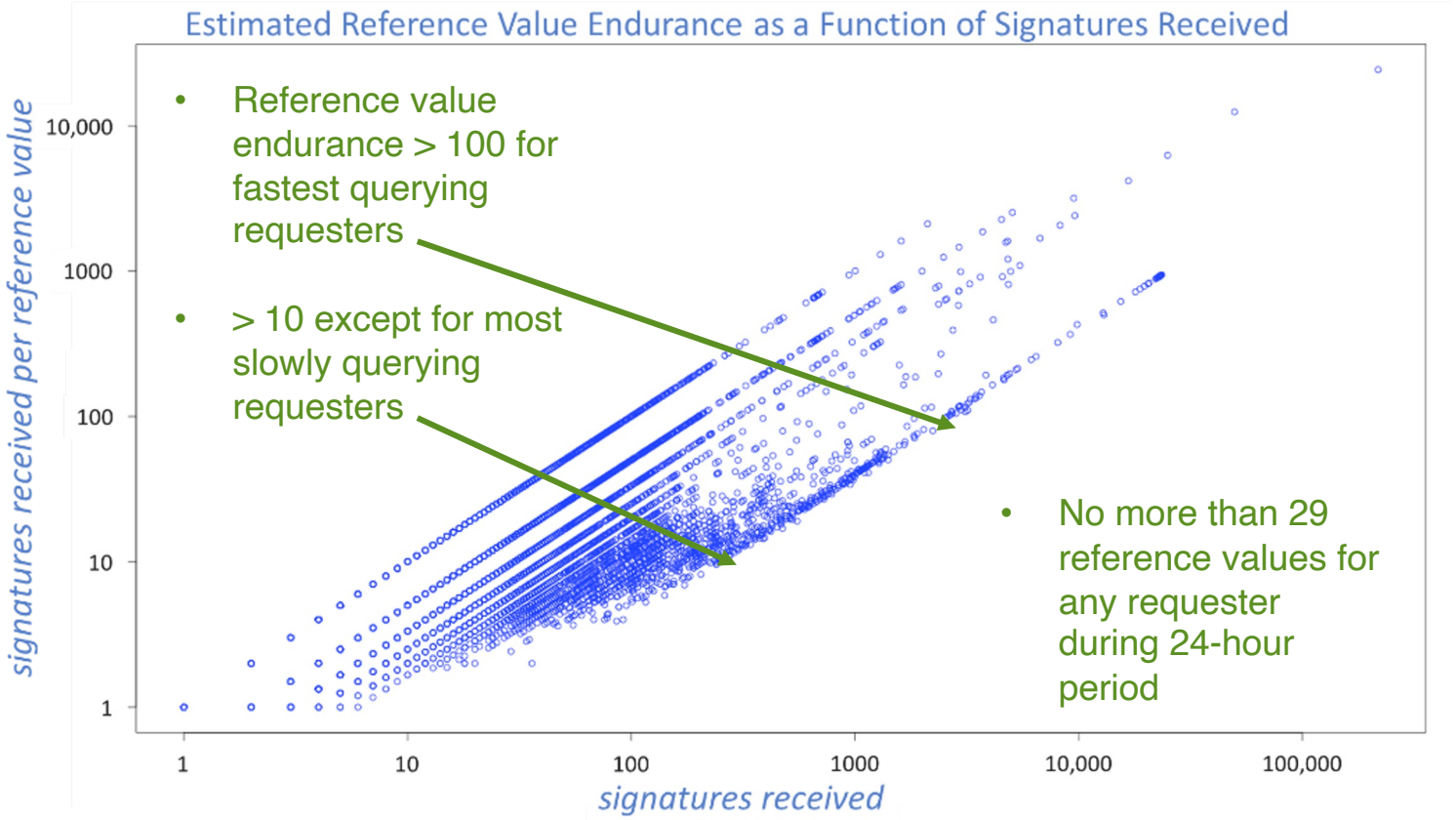
Converging to Condensed Signature Size as K Increases



DNSSEC Use Case

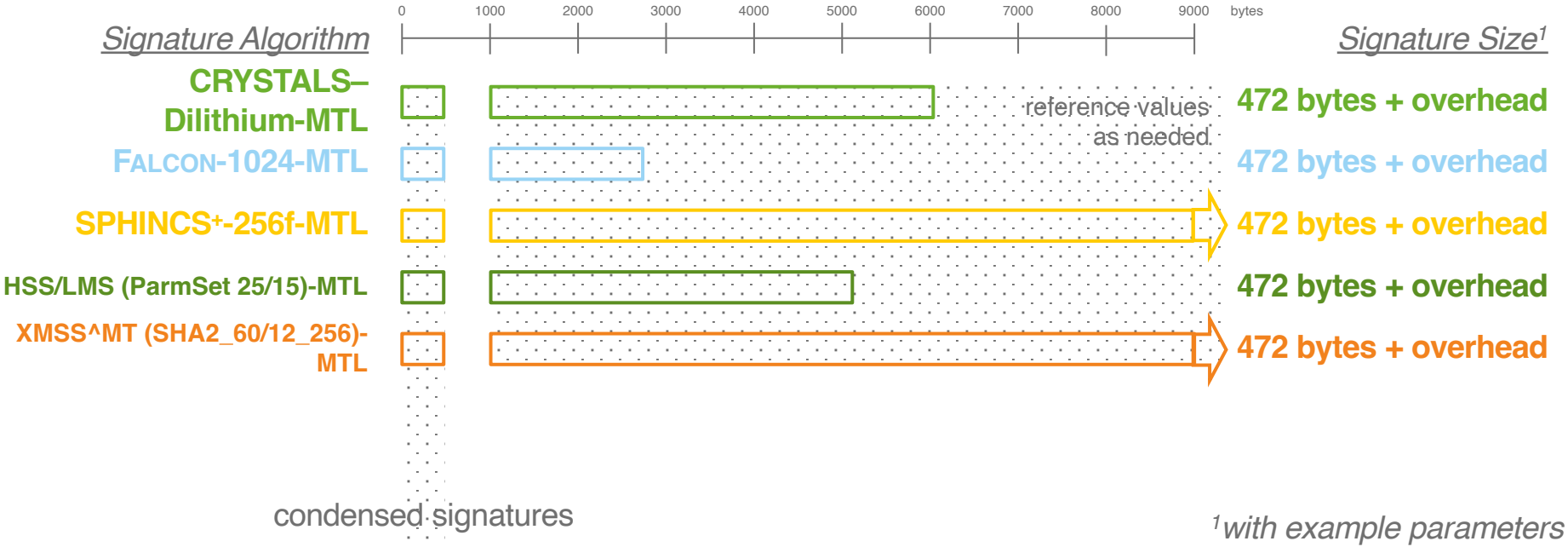
- DNSSEC signatures authenticate DNS records
  - Typically precomputed by signers, at top levels of DNS
  - Requested by verifiers based on interest (~ our model)
- We analyzed one of DNS-OARC's Day In The Life (DITL) data sets from 2015 to see what would have happened if MTL mode had been applied
- Measurement: How often does DNSSEC "signature inception" reach new maximum for a given verifier?
  - Proxy for requesting new reference values

DNSSEC Results



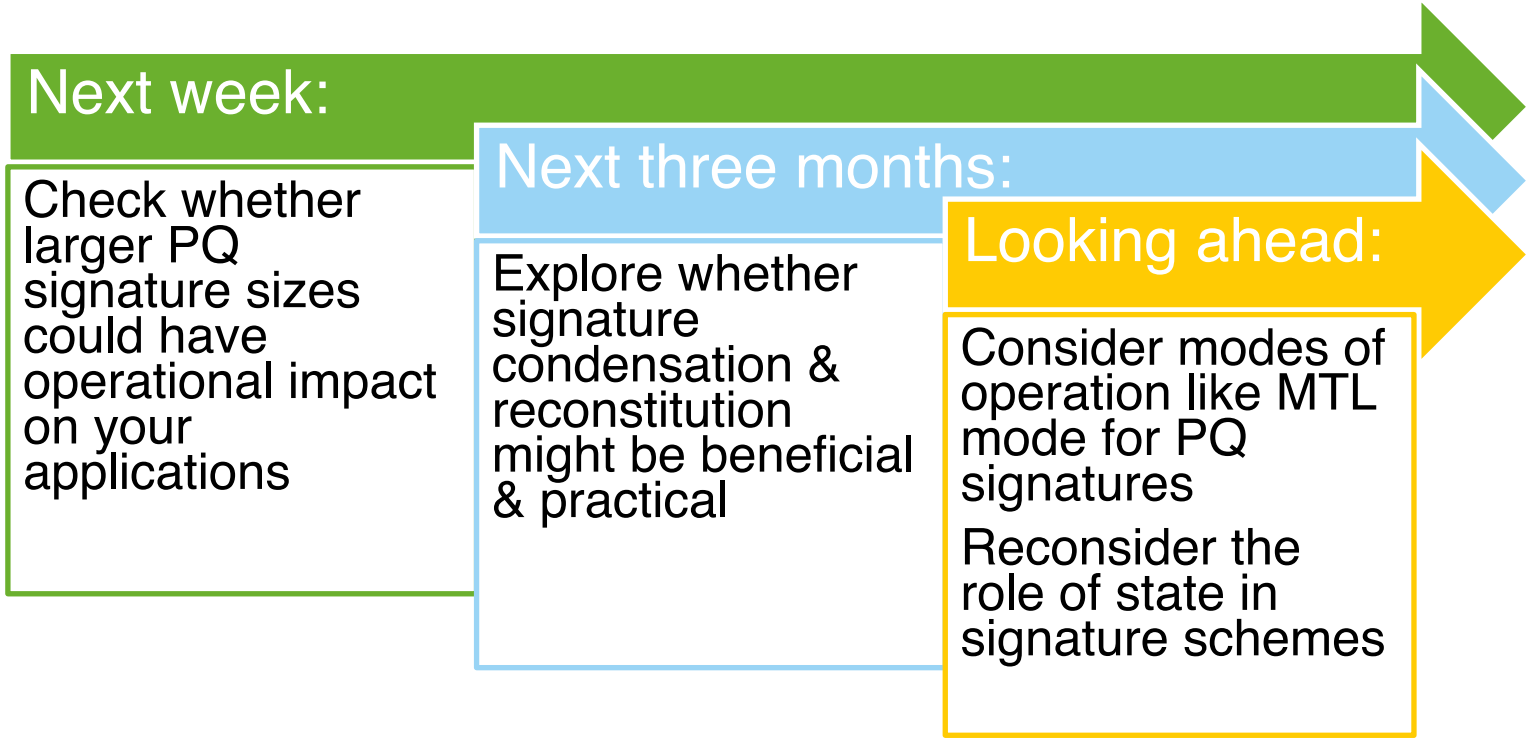
Summary: Reducing Effective Size Impact

Send Condensed Signatures, Look Up Reference Values



[1] *with example parameters*

Applying MTL Mode



Conclusion

Merkle Tree Ladder mode transforms any signature scheme into a stateful, *condensable* signature scheme

Condensation can reduce size impact in message-series-signing scenarios, e.g., DNSSEC

Security bounds are comparable to those for other hash-based signature schemes

Reimagine the state: MTL mode uses state to bring benefits without the usual risk

VERISIGN®