Merkle Tree Ladder Mode: Reducing the Size Impact of NIST PQC Signature Algorithms in Practice

Andrew Fregly, Joseph Harvey, Burton S. Kaliski Jr. and Swapneel Sheth
Verisign Labs

4th NIST PQC Standardization Conference

November 29 – December 1, 2022

Introduction

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

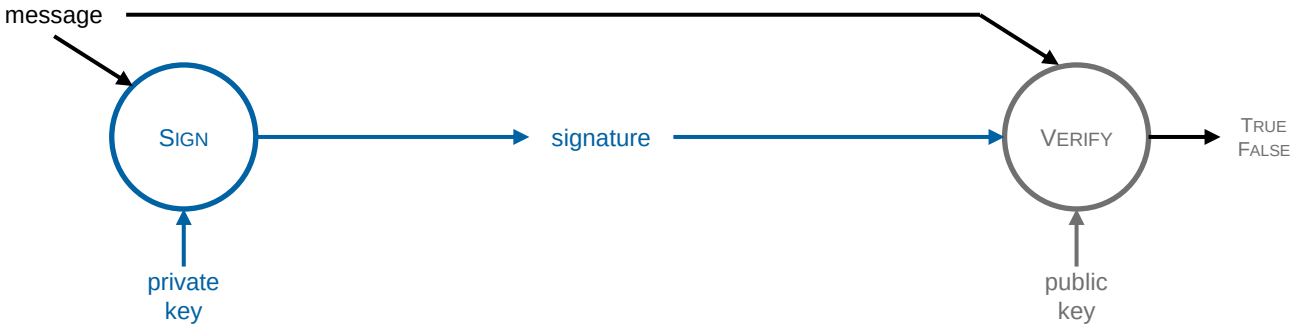
… but sizes of selected signature algorithms are large!

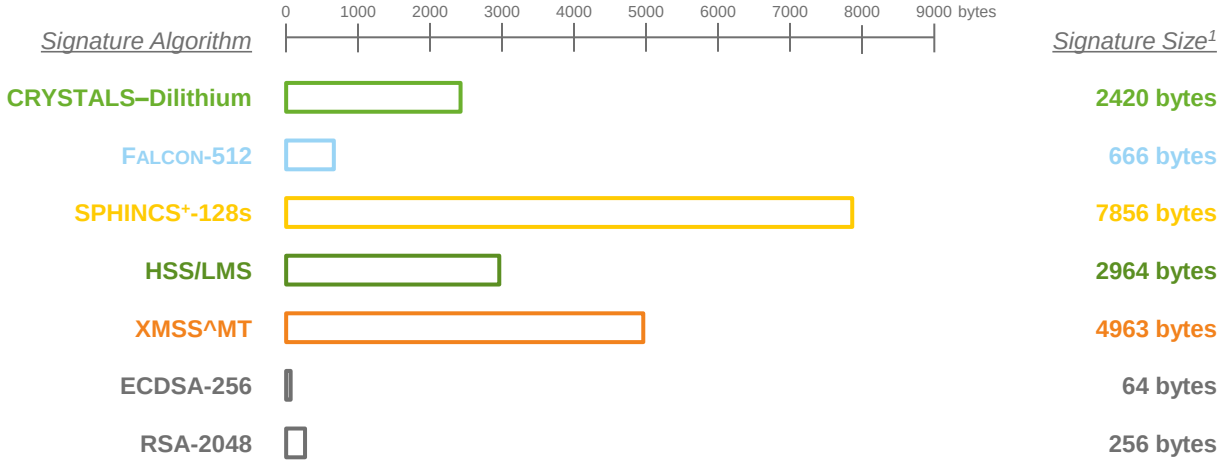
Large size may impact legacy applications with size constraints (e.g., DNSSEC)

How to reduce impact in practice?

NIST PQC Signature Algorithms

666- to 7856-Byte Minimum Signature Sizes with Example Parameters

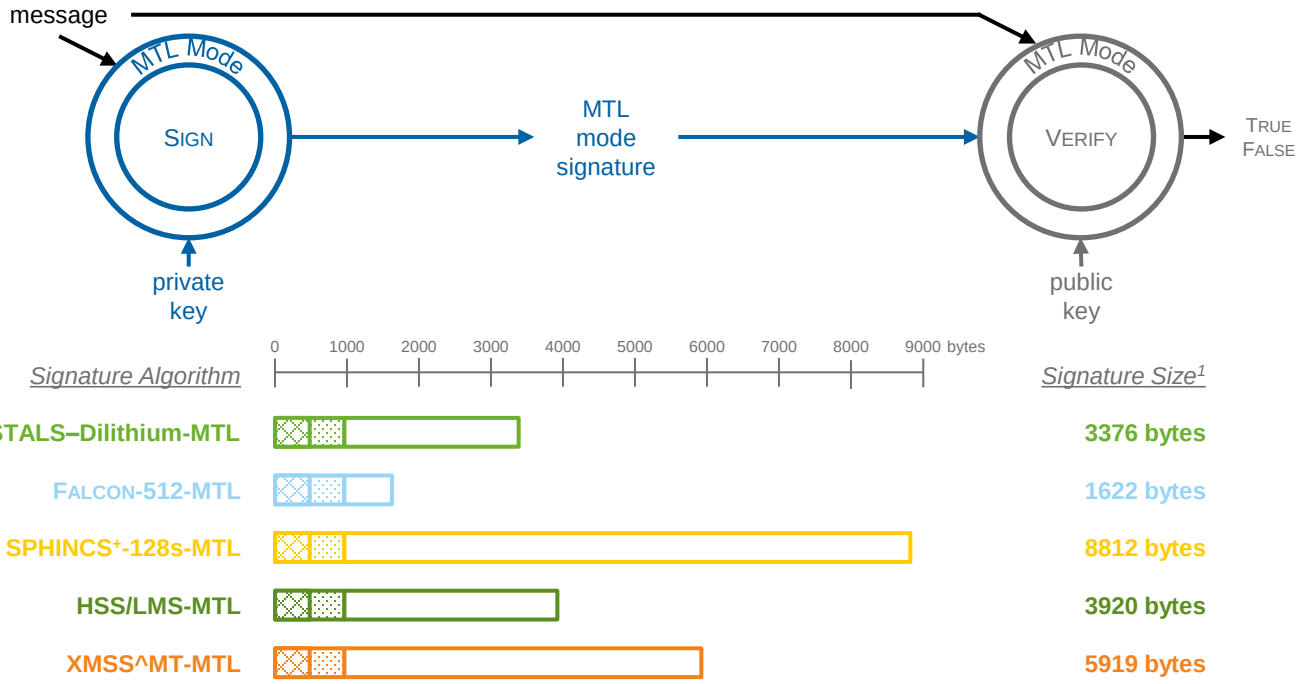




[1]*with example parameters*

Merkle Tree Ladder Mode Signatures

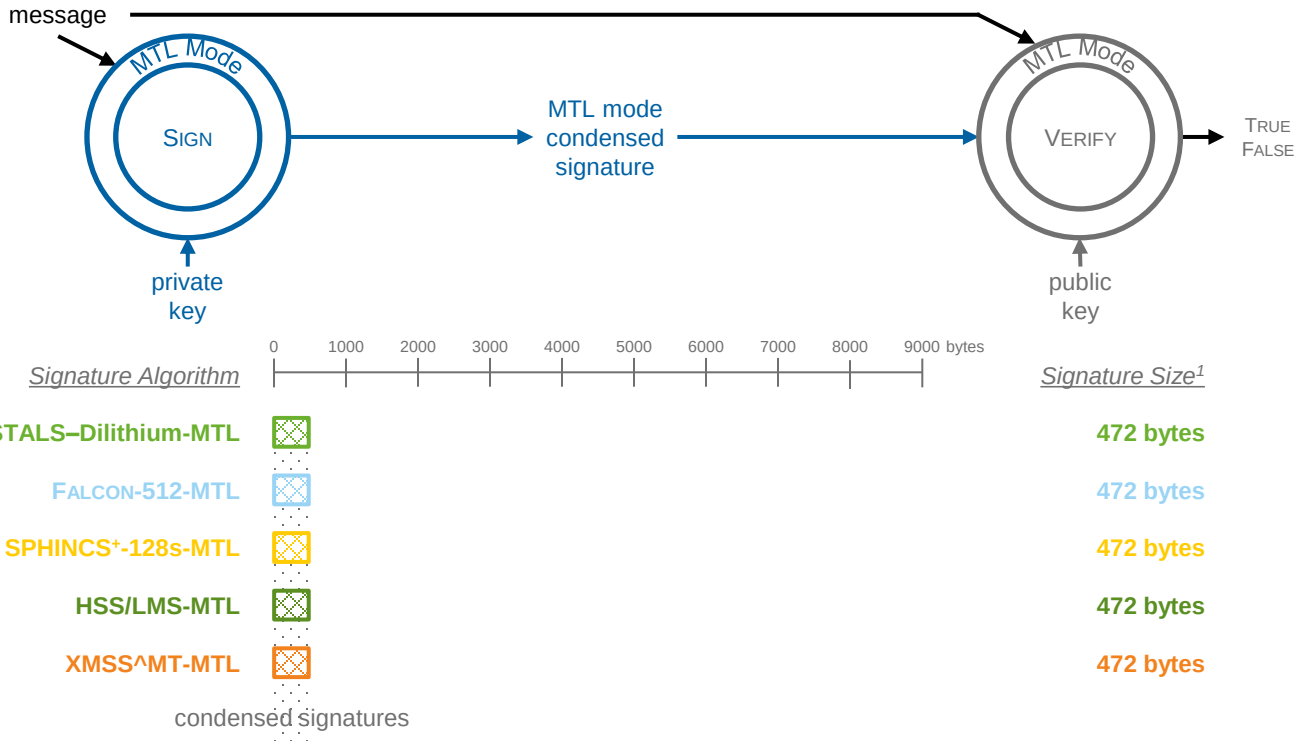
Scheme Transformation… Size Gets Worse Before It Gets Better



[1] *with example parameters*

Merkle Tree Ladder Mode *Condensed* Signatures

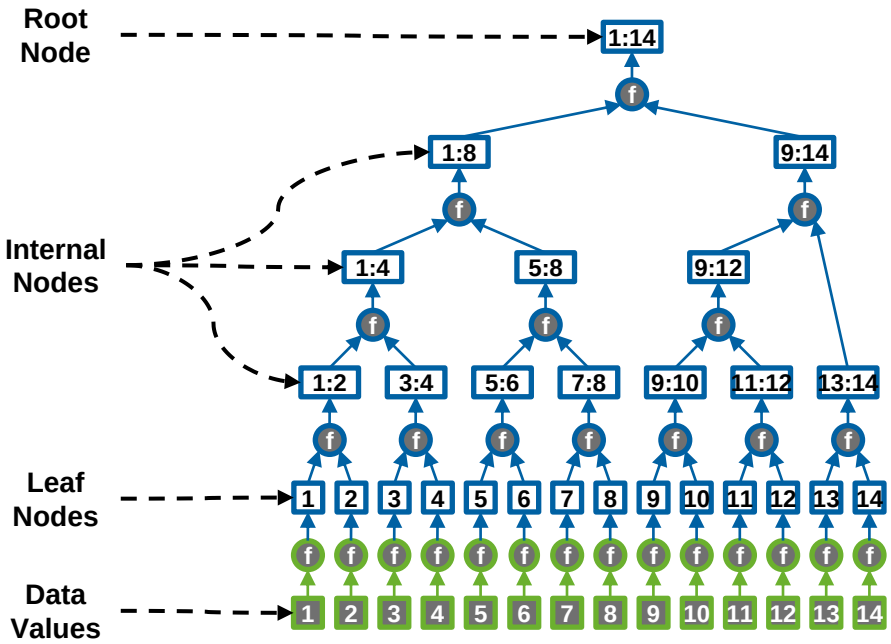
Much Shorter Than Most Underlying PQC Signature Algorithms… But How?

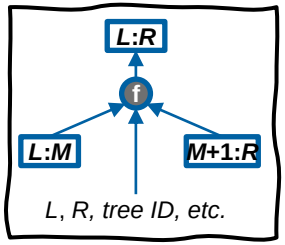


[1]*with example parameters*

Merkle Tree

Root Node Recursively Authenticates Data Values

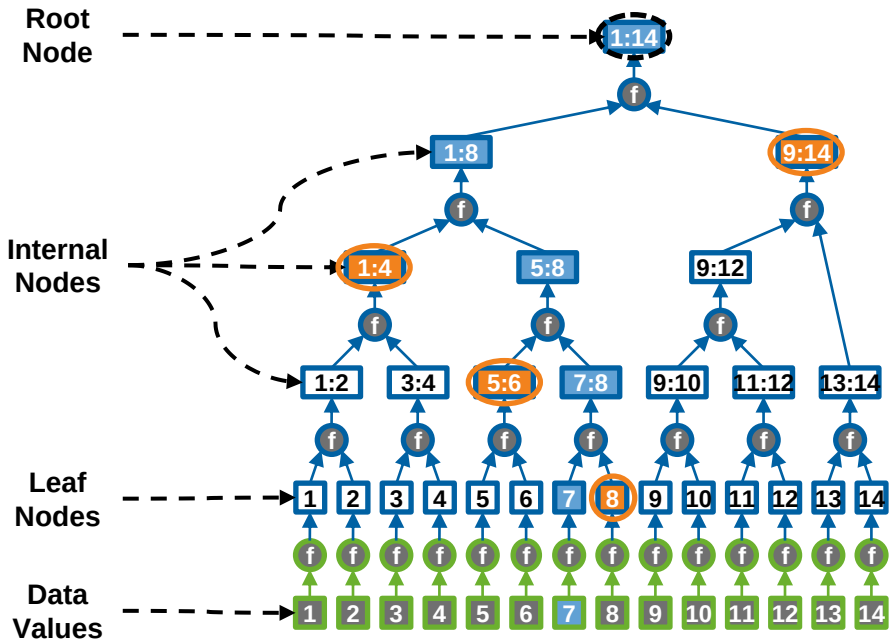




- *Merkle (1979)*
- *Certificate Transparency-like variant*
- *Not limited to power of 2*

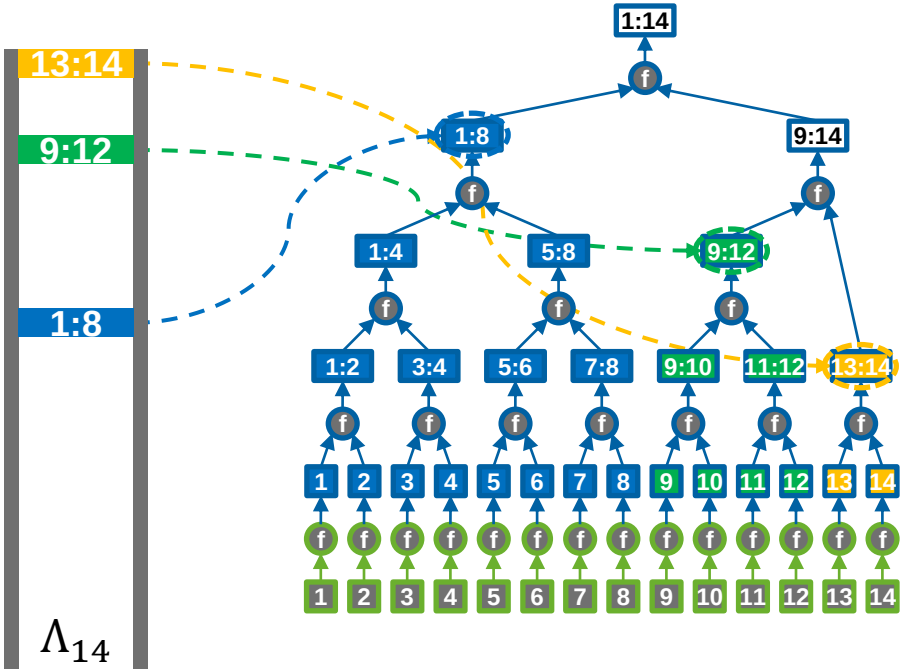
Authentication Path

Verify Data Value by Re-Hashing with Sibling Nodes En Route to Root Node



Merkle Tree Ladder

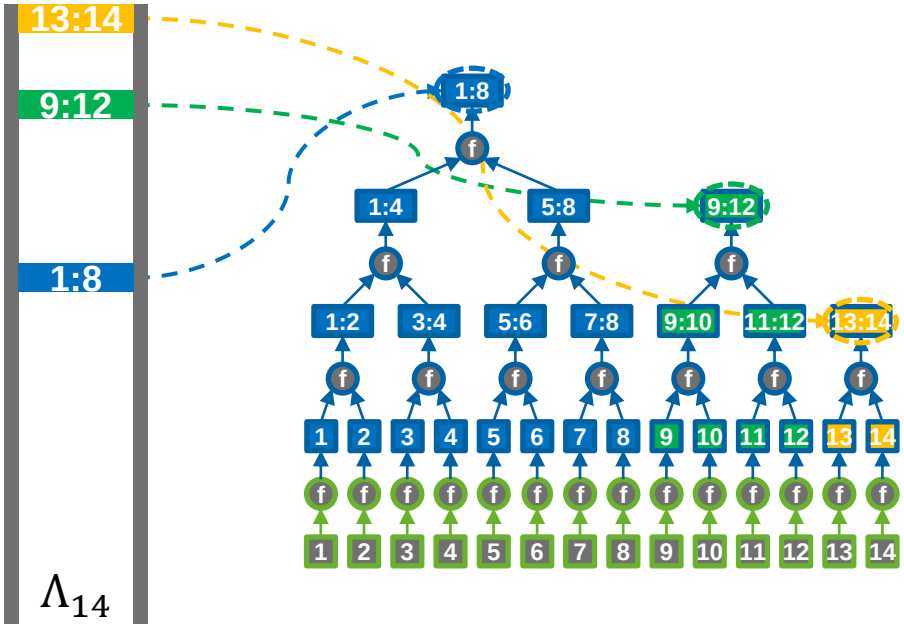
Rungs Collectively Authenticate All Data Values



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

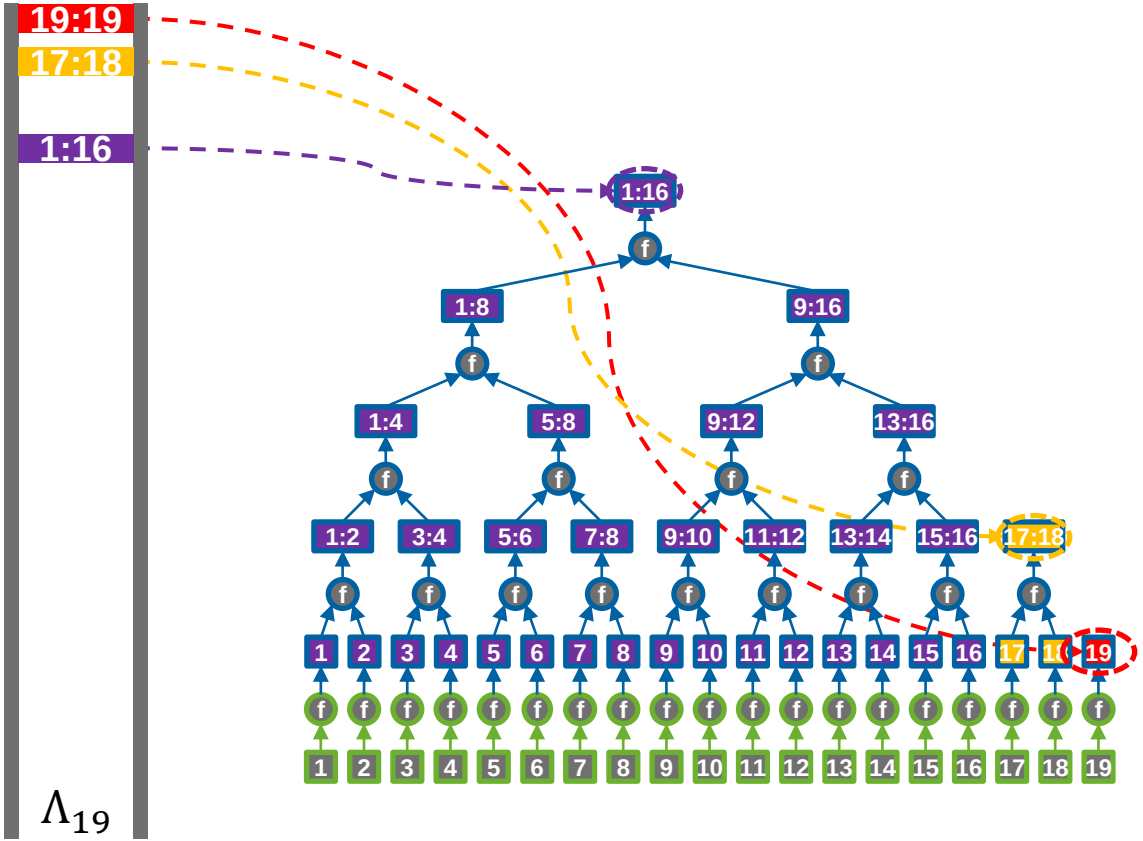
Merkle Tree Ladder

Rungs Collectively Authenticate Data Values… Separate Root Not Necessary



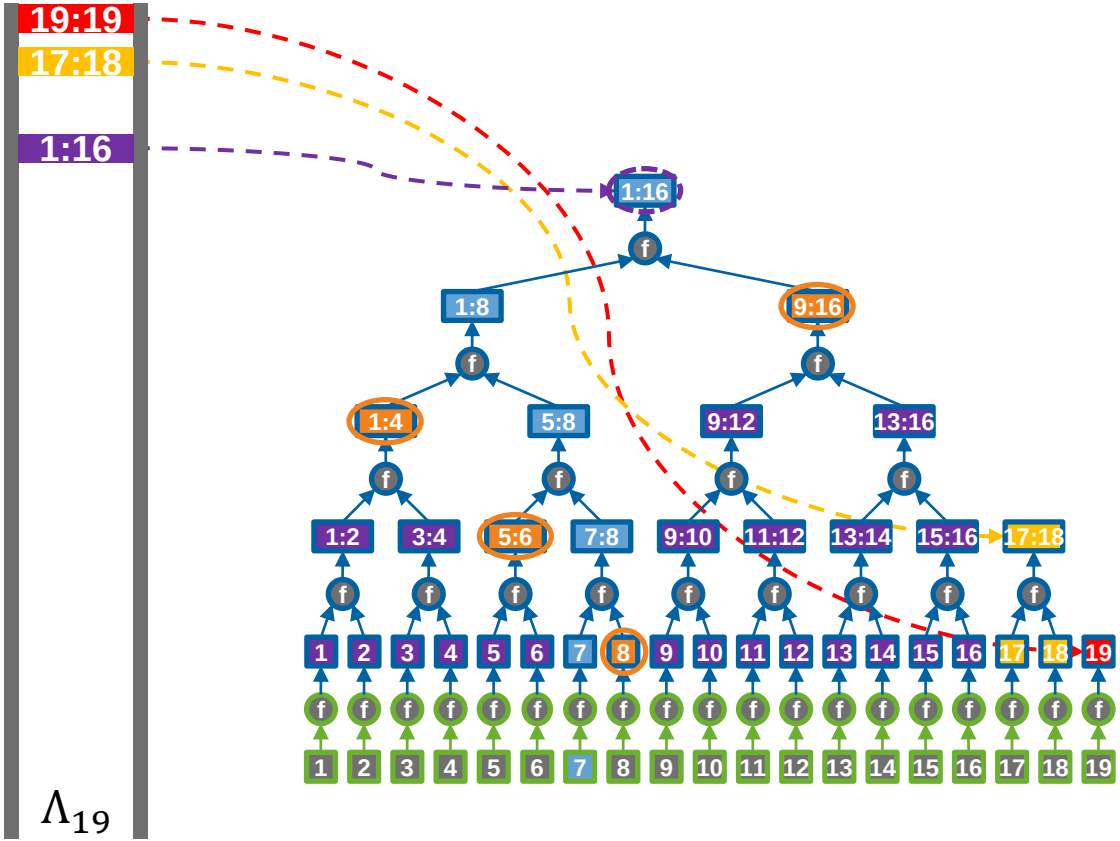
Ladder Evolution

Rungs Updated As New Data Values are Added



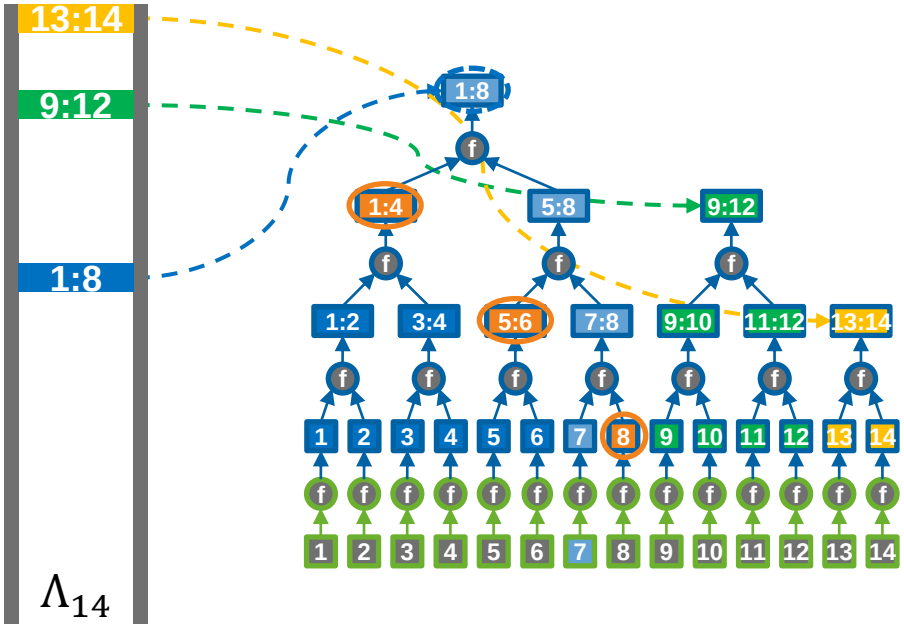
Authentication Path to Ladder

Verify Data Value by Re-Hashing with Sibling Nodes En Route to Rung Node



Backward Compatibility

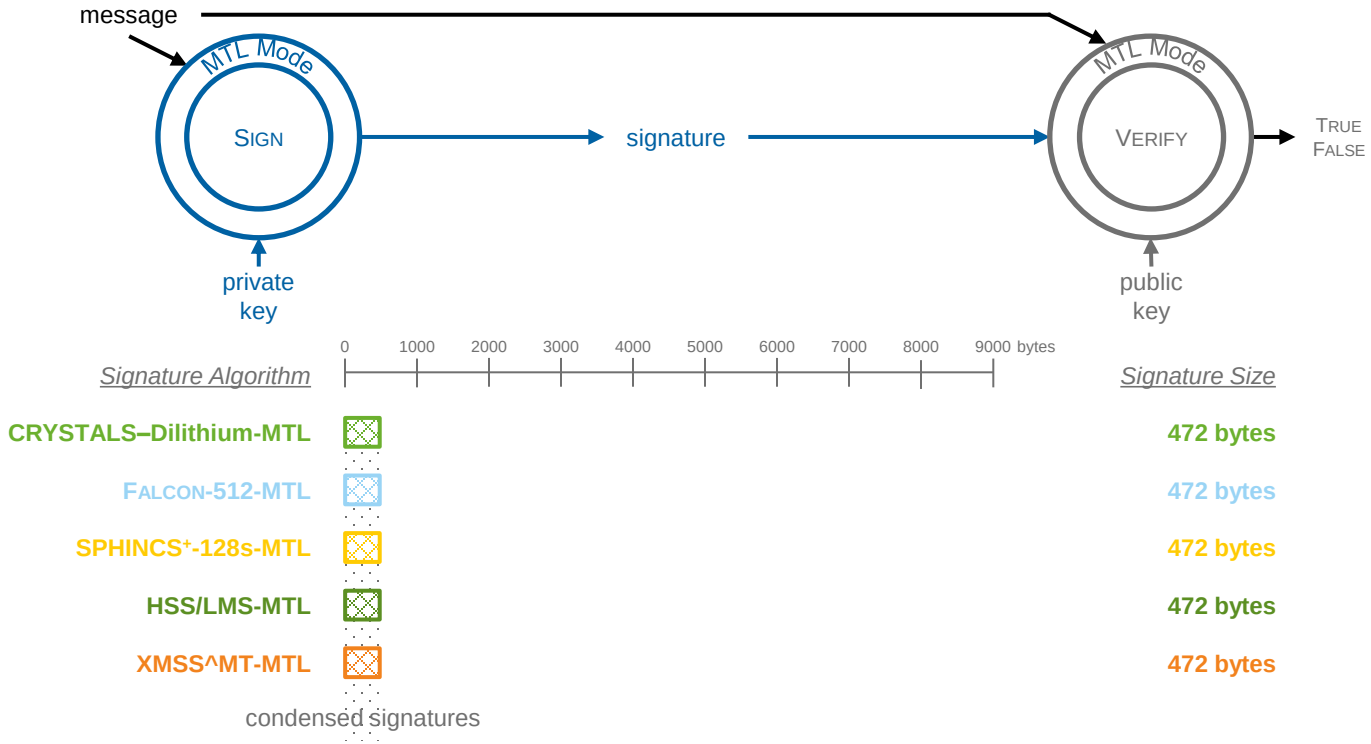
Authentication Path to New Ladder Can Be Verified Using Old Ladder



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

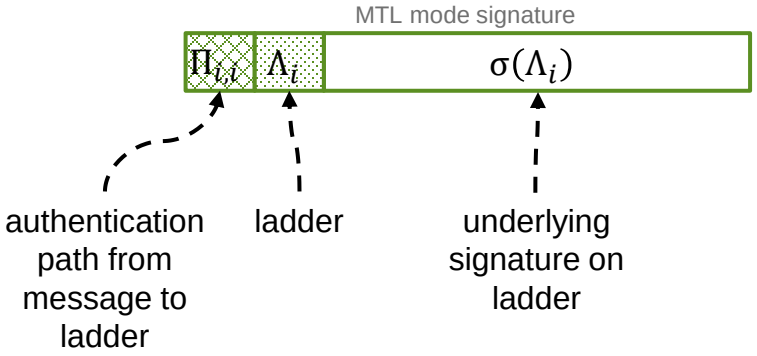
Merkle Tree Ladder Mode *Condensed* Signatures, Revisited

Much Shorter Than Underlying Signature Algorithms… But How?



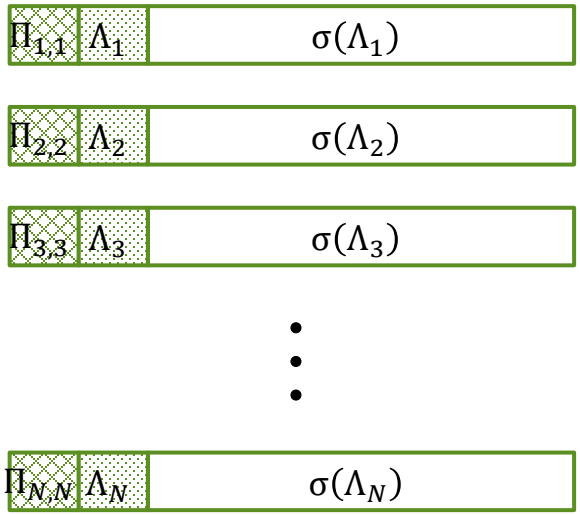
MTL Mode Signature

Idea: Authenticate Message with Ladder; Sign Ladder with Underlying Scheme



MTL Mode Signature *Series*

Multiple Messages Signed in Succession… Ladder Evolves

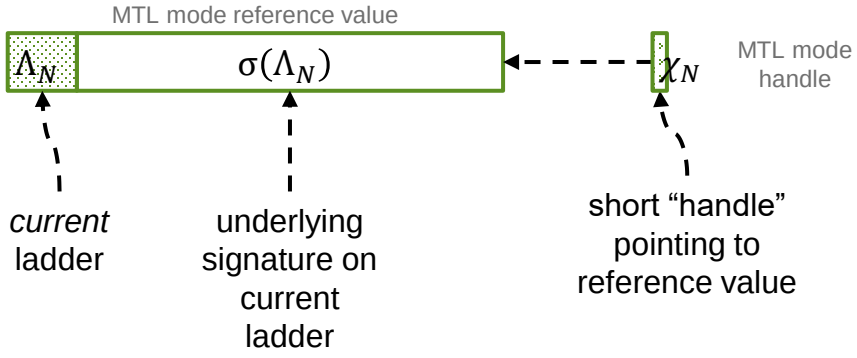


MTL Mode Condensed Signature

Just Authentication Path… Sent in Place of MTL Mode Signature

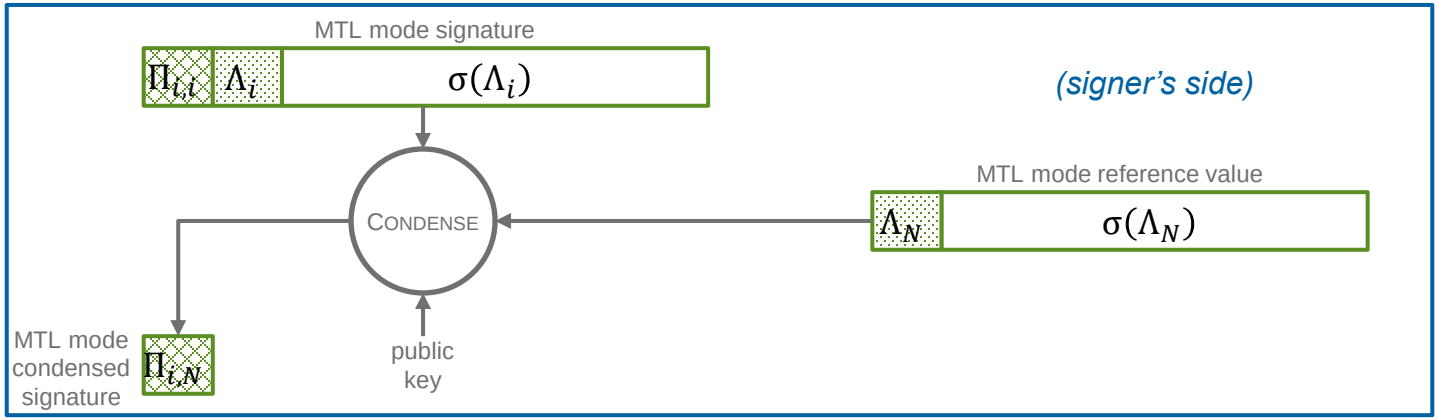
MTL Mode Reference Value

Ladder + Underlying Signature on Ladder… Looked Up When Needed



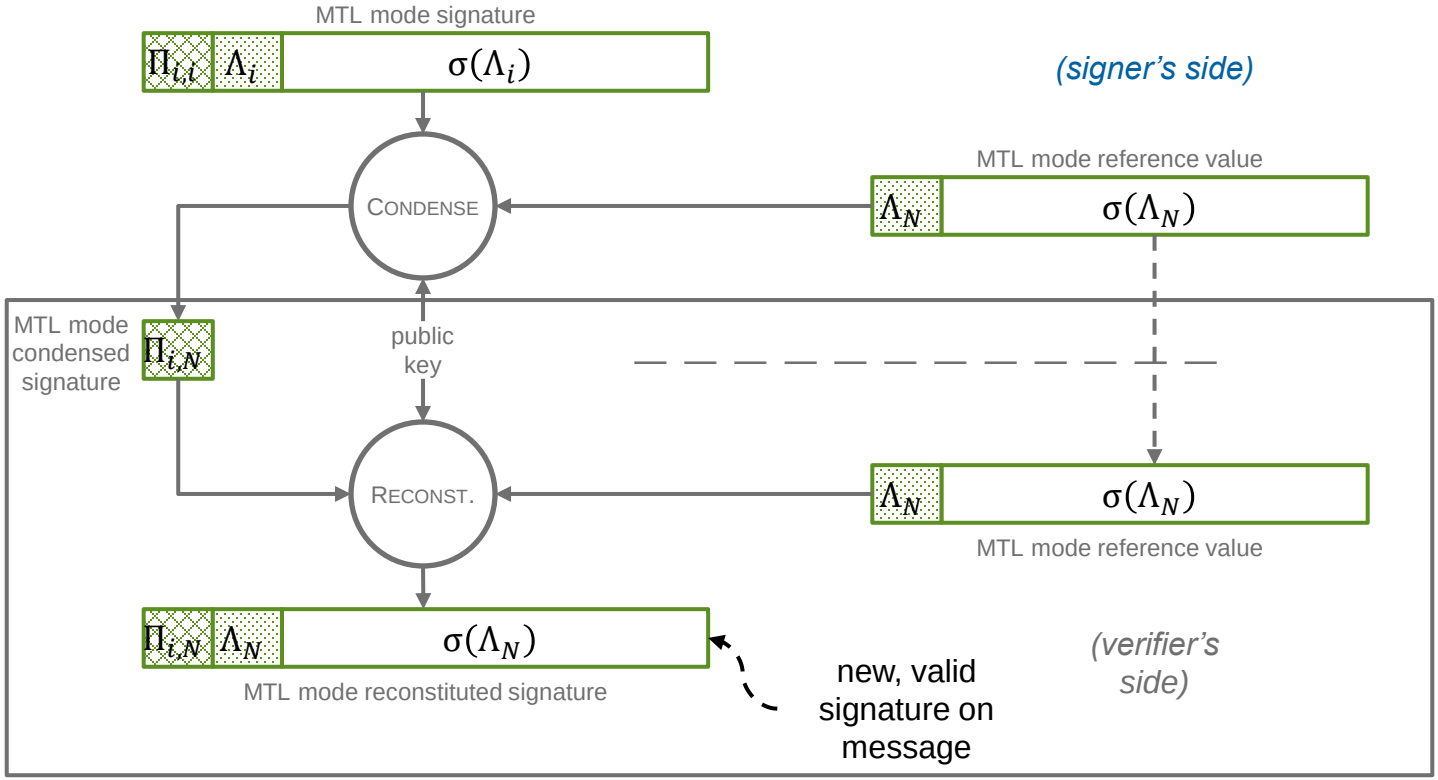
“Condensing” MTL Mode Signatures

Anyone with Access to Signature Series Can Do It… Not Just Signer



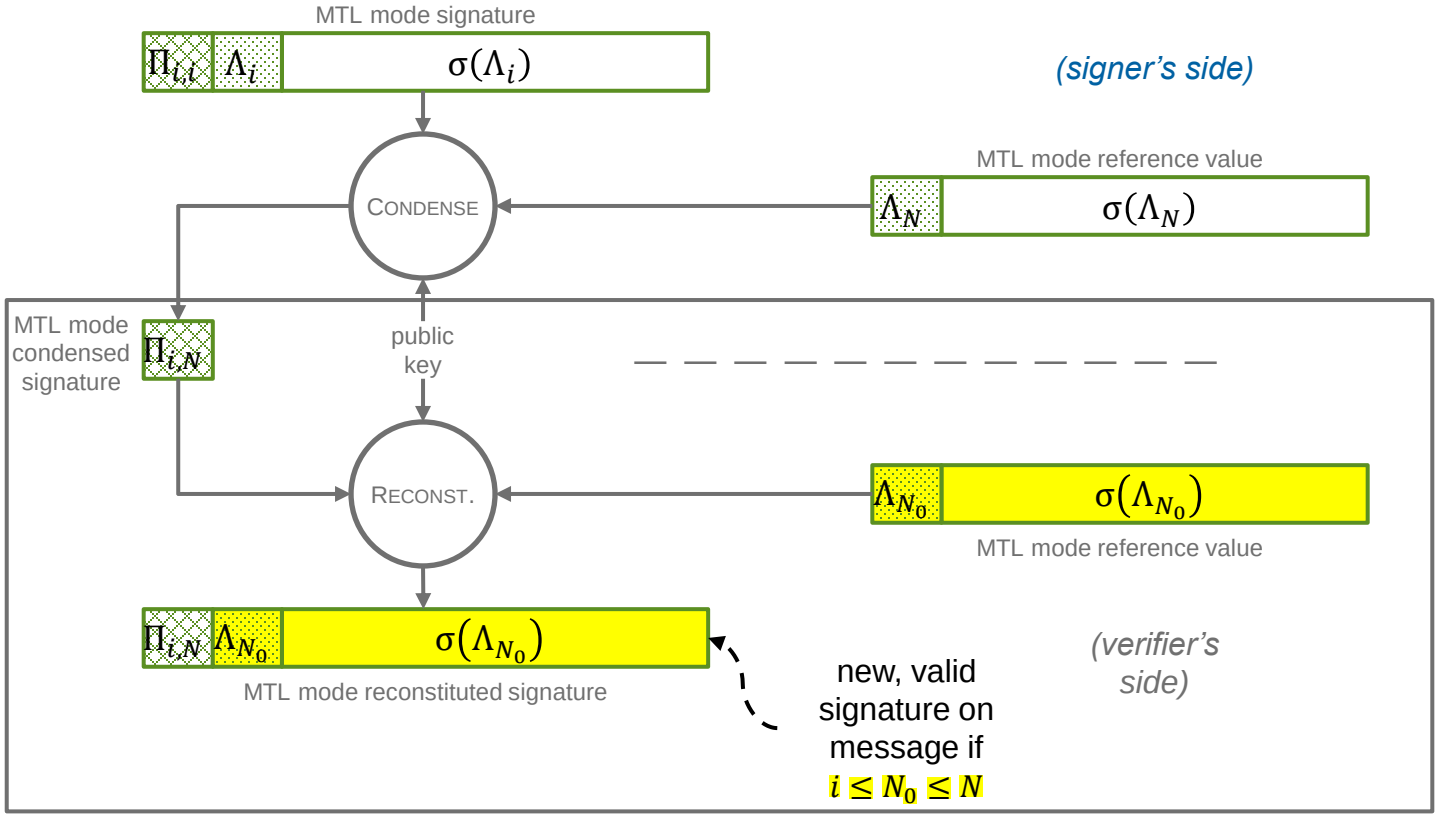
“Reconstituting” MTL Mode Signatures

Anyone who Can Request Reference Values Can Do It… Not Just Verifier



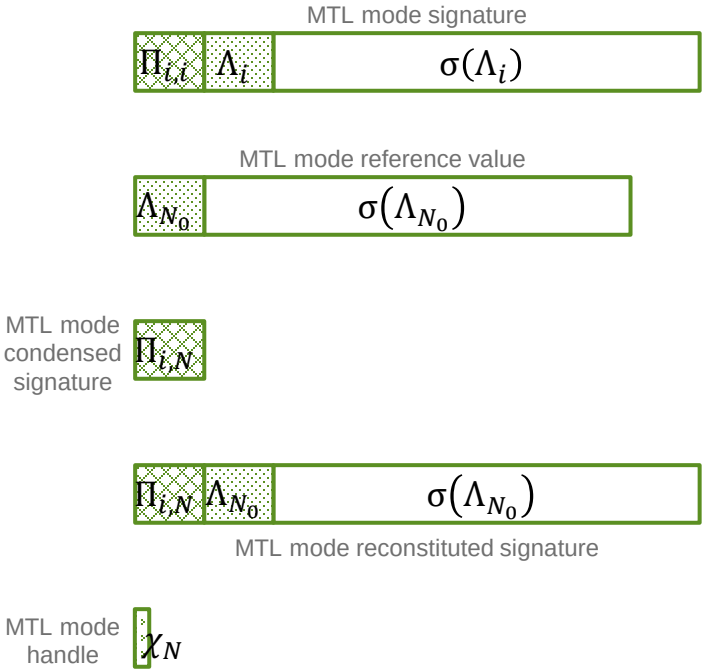
Reconstitution with Backward Compatibility

New Condensed Signature Can Be Verified Using Old Reference Value



MTL Mode Signature: More Details

Example Parameterization with $N_0 = 10{,}000$, 256-Bit Hash, 32-Bit Tree Indexes



- With $N_0 = 10{,}000$, ladders incl. up to 14 hash values, auth. paths up to 13
- Reference value size (also incl. N_0):

$$14 \times 32 + 4 = 452 + \text{size of } \sigma(\Lambda_{N_0})$$

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

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

- MTL mode signature size (also incl. series identifier, randomizer, i, N, N_0 and data value corresponding to message — to reconstruct tree from MTL mode signatures only):

$$13 \times 32 + 14 \times 32 + 16 + 32 + 4 + 4 + 4 + 32 = 956 + \text{size of } \sigma(\Lambda_{N_0})$$

- MTL mode is *stateful*, *malleable*, can be modeled as *tagged* signature scheme
- MTL mode is naturally *quantum-safe*: based on hash functions only

Ladder Endurance: Initial Model

How Many Messages Until Verifier Needs New Reference Value?

N_0 initial messages

1 new message signed per “iteration”

1 message 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: Initial Model, cont'd

How Many Messages Until Verifier Needs New Reference Value?

$$\text{Prob}[\kappa \text{ "good" iterations}] \approx \prod_{t=1}^{\kappa} \frac{N_0}{N_0 + t}$$

$$\approx \exp\left(-\frac{\kappa^2}{2N_0}\right)$$

- Probability $\approx 1/2$ when $\kappa \approx \sqrt{2 \ln 2}\sqrt{N_0}$
  - analysis similar to Birthday Paradox
- $\sqrt{2 \ln 2}\sqrt{N_0}$ iterations → $K = \sqrt{2 \ln 2}\sqrt{N_0}$ messages

Ladder Endurance: General Model

How Many Messages Until Verifier Needs New 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 Messages Until Verifier Needs New Reference Value?

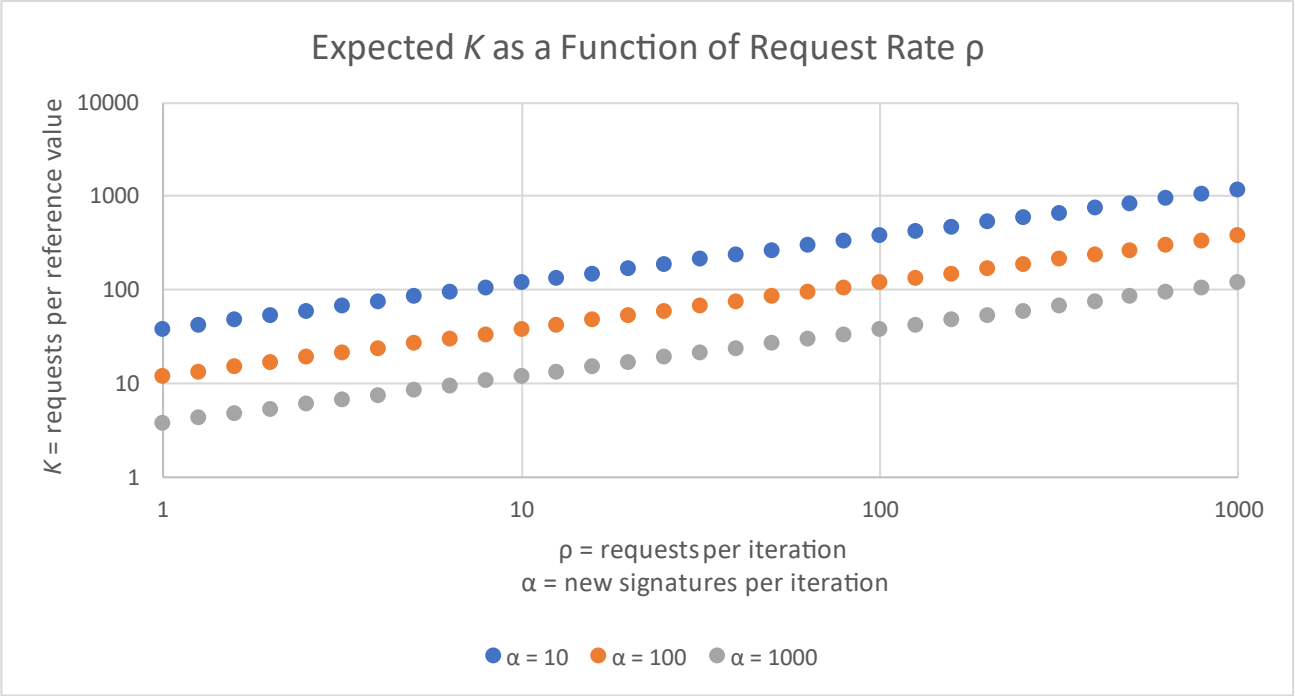
$$\text{Prob}[\kappa \text{ "good" iterations}] \approx \prod_{t=1}^{\kappa} \left(\frac{N_0}{N_0 + t\alpha} \right)^{\rho}$$

$$\approx \exp\left(-\frac{\kappa^2 \alpha\rho}{2N_0} \right)$$

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

Expected Ladder Endurance

As Function of Request Rate ρ, Update Rate α; Assume $N_0 = 10{,}000$



Effective Signature Size

Average Number of Bytes that Signer Sends Per Message of Interest to Verifier

K = number of messages of interest

K' = number of reference values received

$|\varsigma|$ = size of condensed signature

$|\upsilon|$ = size of reference value

- Effective size including "overhead" of reference values:

$$\phi(K, K') = |\varsigma| + \frac{K'}{K}|\upsilon|$$

- $K' = 1$ in our model — K is number of messages until *next* reference value

Effective Signature Size for Various Algorithms in MTL Mode

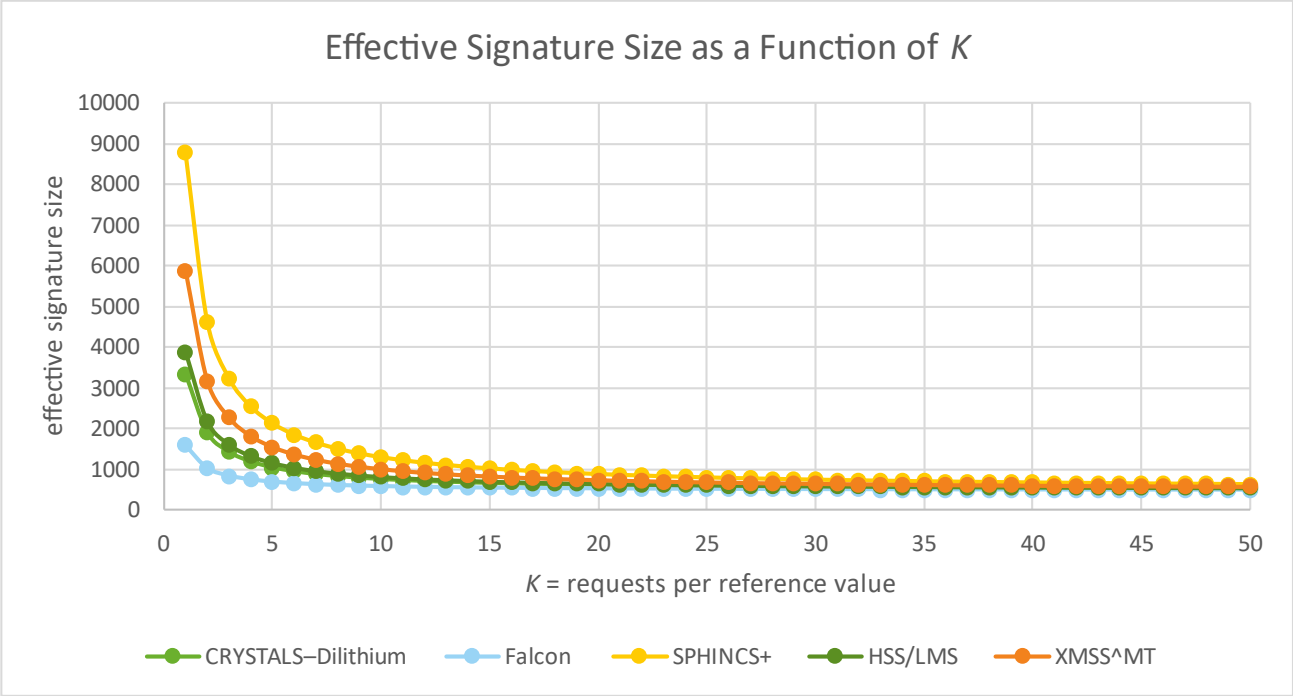
Mode Parameterized for ~10,000-Message Series; Smaller Size if Shorter

Signature Algorithm (in MTL Mode)	Underlying Signature Size	Condensed MTL Mode Signature Size $\lvert\varsigma\rvert$	Reference Value Size $\lvert\upsilon\rvert$	Effective MTL Mode Signature Size ($K = 10$)
CRYSTALS–Dilithium	2420	472	2872	759.2
FALCON-512-MTL	666	472	1118	583.8
SPHINCS⁺-128s-MTL	7856	472	8308	1302.8
HSS/LMS-MTL	2964	472	3416	813.6
XMSS^MT-MTL	4963	472	5415	1013.5

$$\phi(K,1) = |\varsigma| + \frac{1}{K}|\upsilon|$$

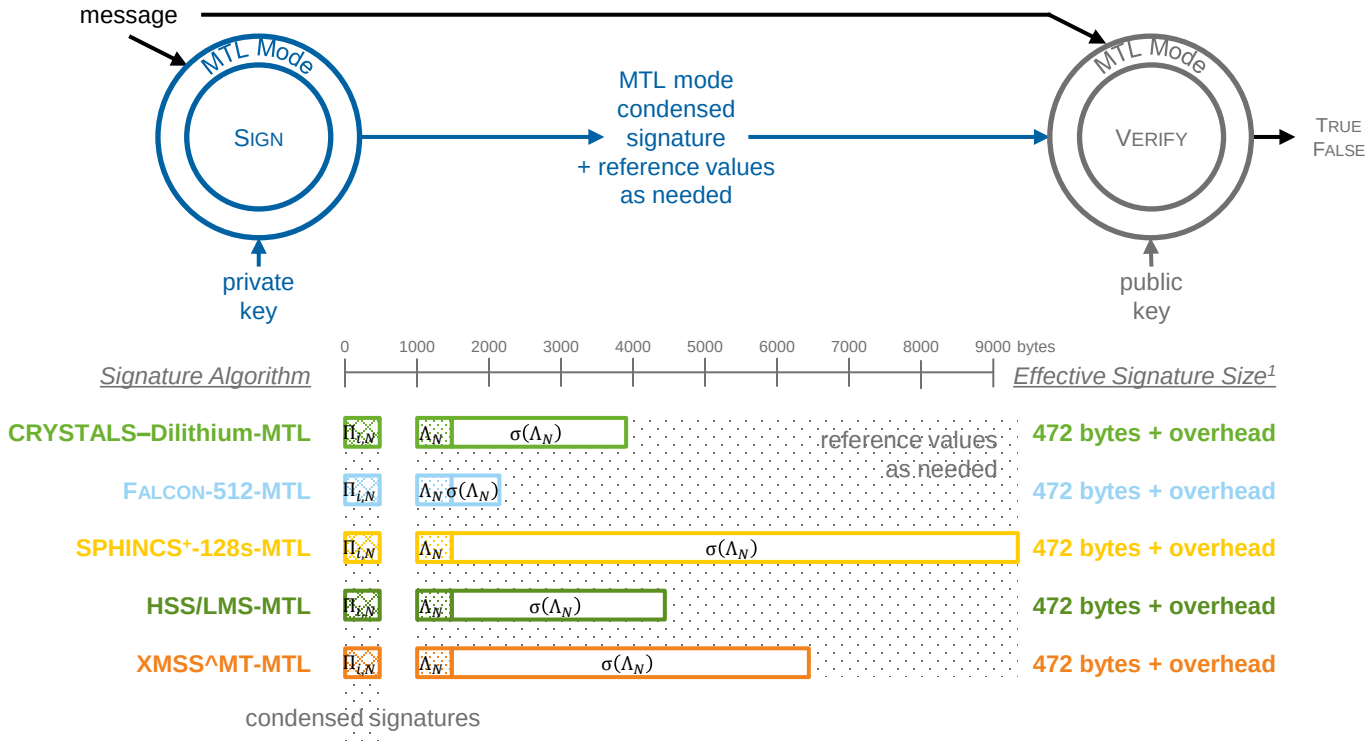
Effective Signature Size for Various Algorithms in MTL Mode

Converging to Condensed Signature Size as K Increases



Summary: Reducing Effective Size Impact with MTL Mode

Send Condensed Signatures, Look Up Reference Values As Needed



[1]*with example parameters*

Conclusion

MTL mode provides a way to reduce size impact of signature algorithms

Hash-based construction — conservative design supporting diversity of algorithm choices

Targets “message series” + reference value lookup models which are common in practice

New direction for PQC signature algorithms: Modes of operation

powered by

VERISIGN®