DSN 2026 — Practical Experience Report

Exploring Side-Channel Protections in Hardware Implementations of PQC ML-KEM Verification

Davis Ranney | Yashaswini I. Makaram | A. Adam Ding | Yunsi Fei

Northeastern University — Electrical & Computer Engineering - Mathematics

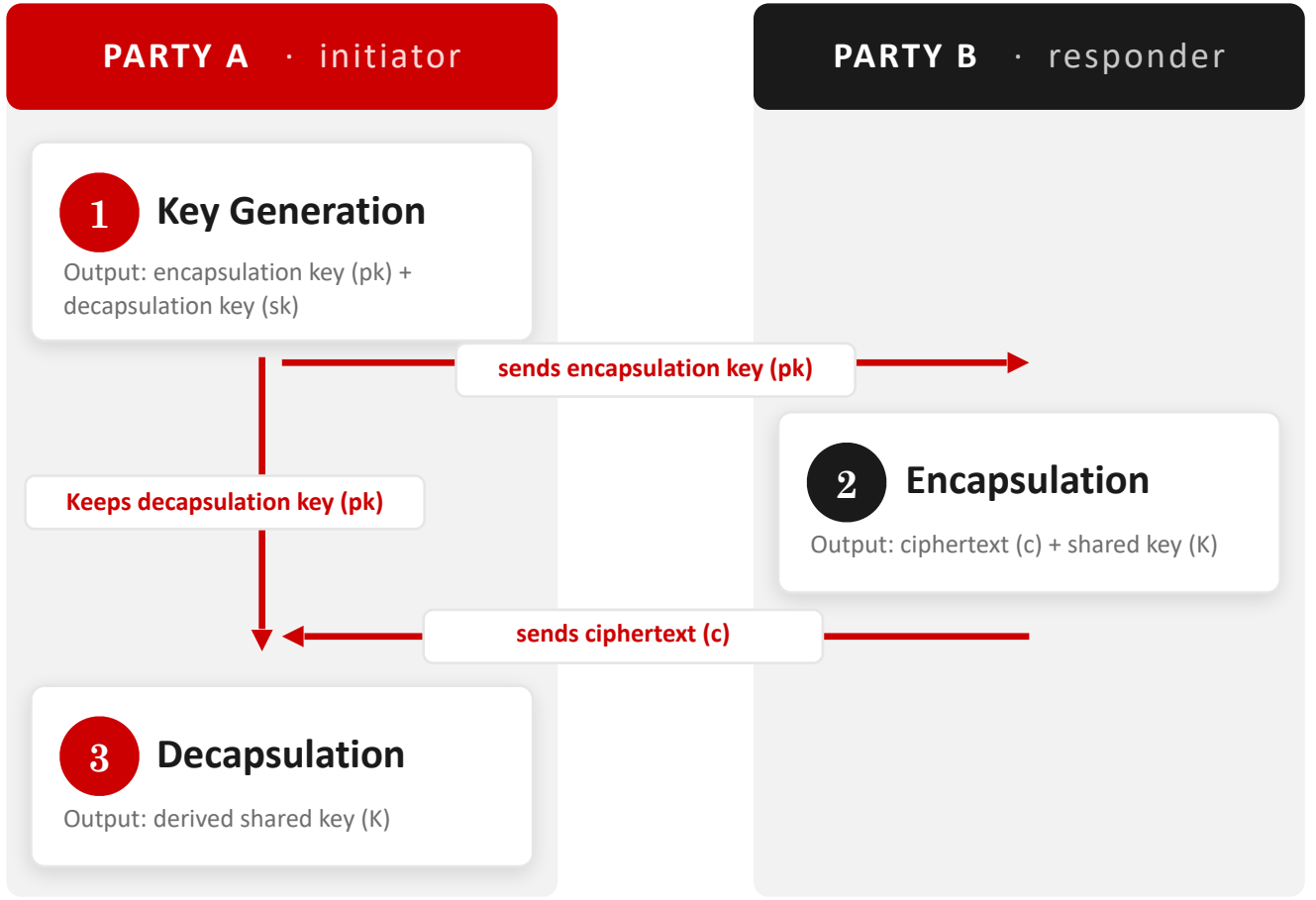
DSN 2026 - Practical Experience Report - This work was supported in part by the National Science Foundation (NSF) under grants SaTC-1929300, CNS-1916762, and industry partners of NSF IUCRC CHEST

Background — The Standard

NIST-standardized post-quantum KEM (FIPS 203, ML-KEM), built on module lattices and learning with errors, based upon CRYSTALS-Kyber

WHERE IT'S USED

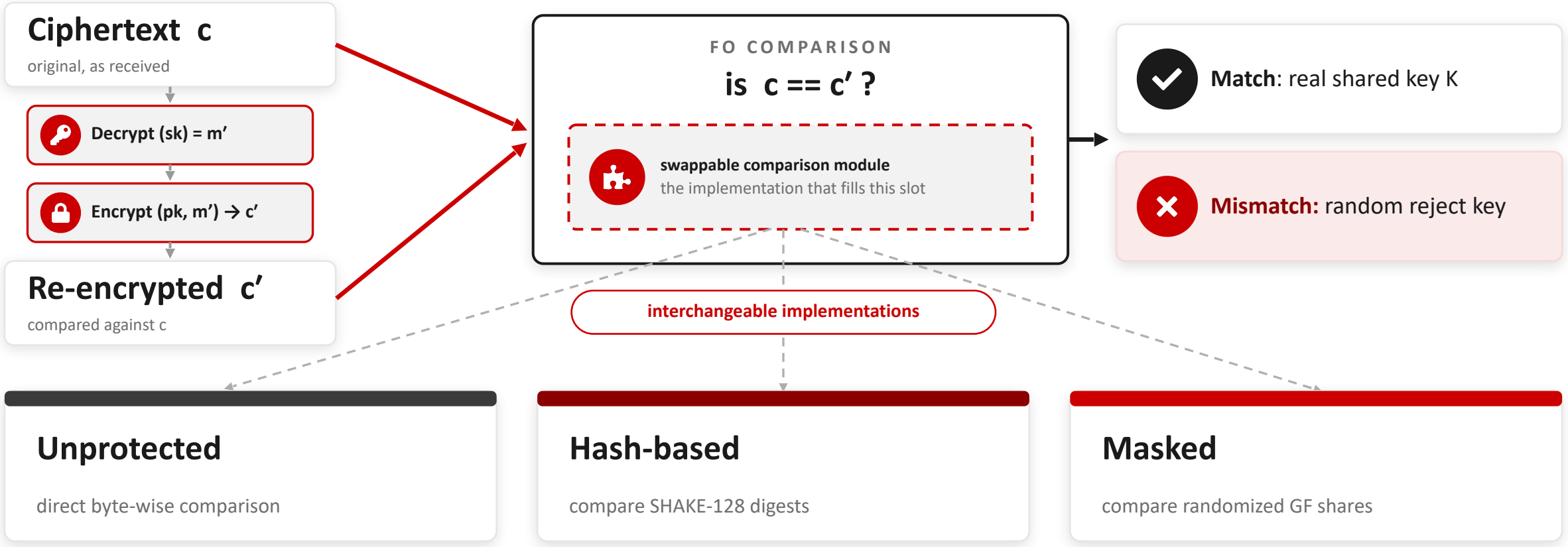
- **TLS 1.3:** hybrid X25519 ML-KEM-768 handshake
- **OpenSSL 3.5+ / BoringSSL:** built-in (NGINX, HAProxy)
- **Web browsers:** Chrome, Edge & Firefox by default
- **Secure messaging:** Signal (ML-KEM Braid), iMessage (PQ3)
- **SSH & VPNs:** Post-quantum session-key agreement



Result: both parties now hold the same secret key K to encrypt the session.

Background — The Leakage Mechanism

FO verification re-encrypts the decrypted message and compares the result with the original ciphertext. This is what makes ML-KEM **IND-CCA2 secure**



Different levels of proposed side channel protection, but **data-dependent power signatures are present and able to be measured by an attacker**

Threat — The Attack Structure

1. Craft ciphertexts
Chosen ciphertexts that differ only slightly from honest ones

2. Measure the compare
Power / EM separates success from failure

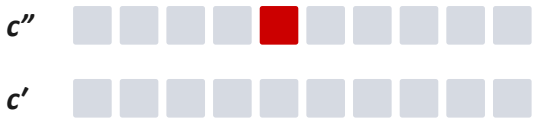
3. Classify success / fail
Each trace gives one inequality to recover the secret in the secret key

4. Solve for the key
Belief-propagation recovers the full ML-KEM-512 key with ~6000-8000 inequalities

WHAT THE POWER TRACE MEASURES

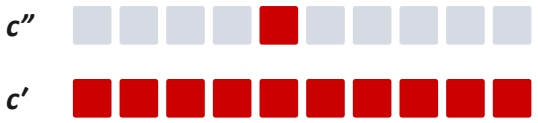
The attacker crafts ***c”*** slightly different from an honest ciphertext:

DECAPSULATION SUCCESS



c’ re-encrypts cleanly
c” and c’ differ in ~1 block
c’ == c

DECAPSULATION FAILURE



c’ unrelated to c”
nearly all blocks differ → HIGH, spread power

The software returns randomized bytes either way *(implicit rejection)*, **but the comparison's power trace separates success from failure**

Our Contribution

BASELINE

Unprotected

Direct byte-wise compare of c and c′ — no obfuscation.

FIRST-ORDER

Hash-based

Hash both ciphertexts with SHAKE-128, then compare digests.

HIGHER-ORDER

Randomized Shares

Split into randomized GF shares; compare in the masked domain.

Focus of this work: all three are implemented on a **SAKURA-G Spartan-6 FPGA** (vs. a Cortex-M4 microcontroller) and measured with power side-channel analysis. The next slides take each in turn: how it works then what we measured

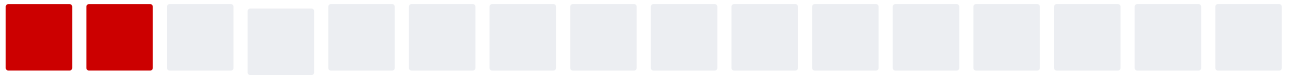
Unprotected — As defined in FIPS 203

A plain equality check of c against c’ with no obfuscation or protection. Useful to characterize noise profiles and compare implementations

- **MCU:** a serial loop over all 768 bytes, one at a time
- **FPGA:** compares a configurable width (32, 128, or 512 bits)
  - Wider lane = fewer cycles = higher throughput

ONE CLOCK CYCLE, MULTIPLE LANE WIDTHS

32-bits / clock



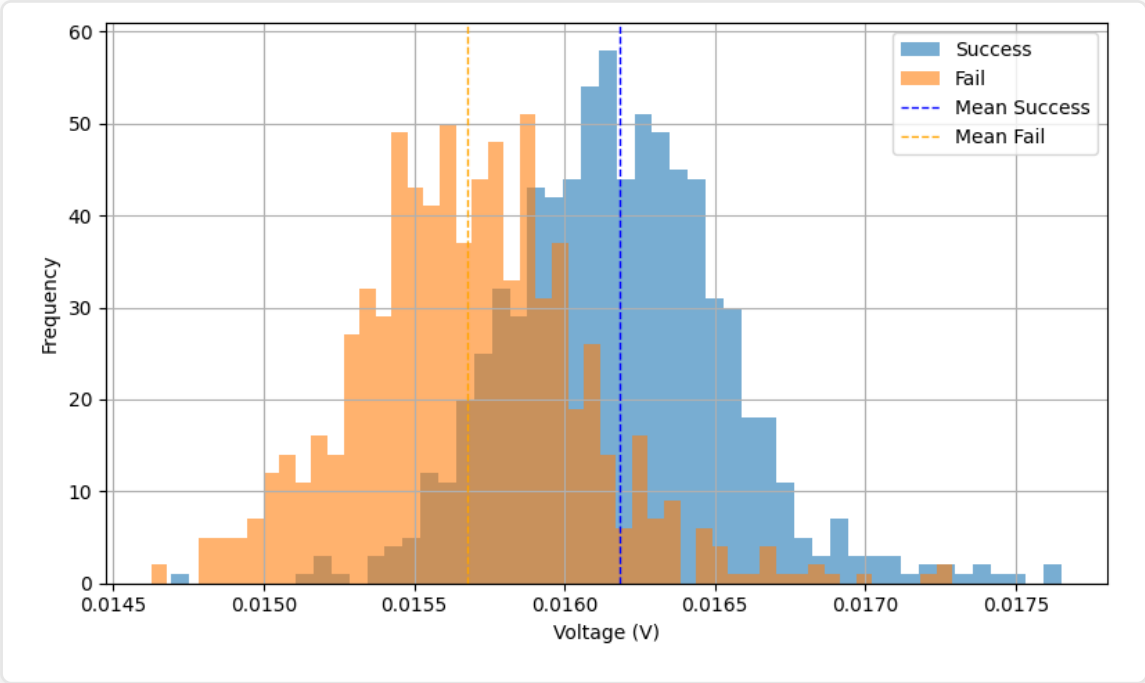
few bits switch together = weaker, noisier leakage

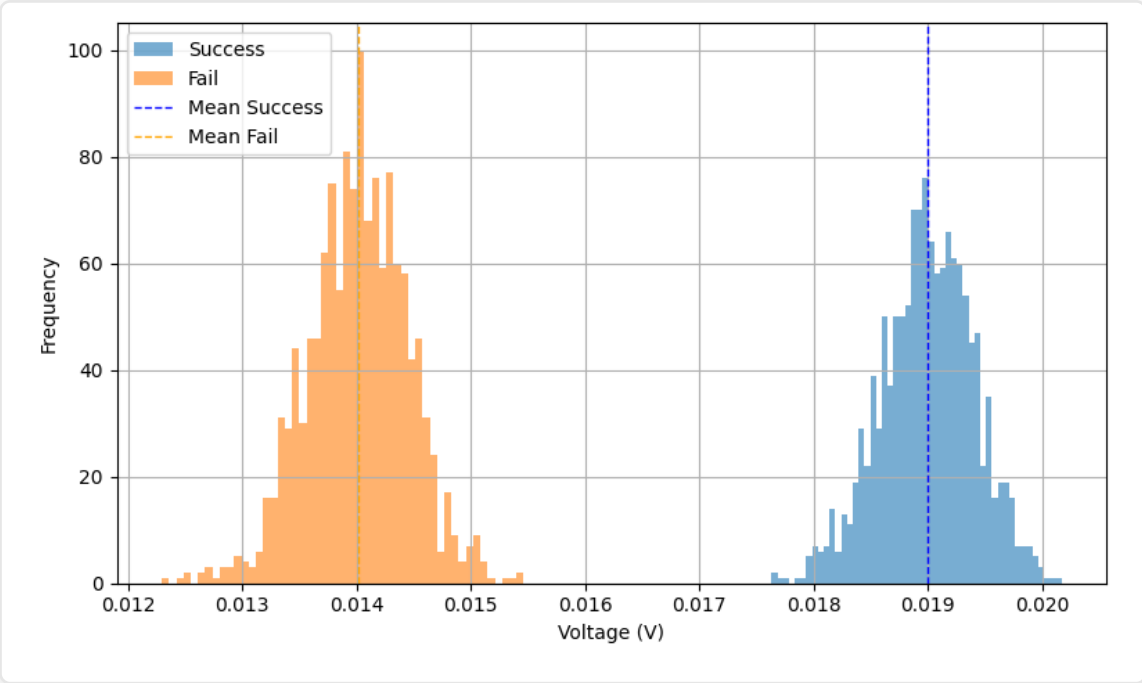
512-bits / clock

More parallel computations = stronger, cleaner leakage

Next — results: the wider the lane, the cleaner the success-vs-failure power gap becomes.

Unprotected — Results





32-bit width, classes overlap, SNR 0.95, 62.9% classification accuracy, below our 80% usable threshold to recover the key

512-bit width, classes fully separate, SNR 8.33, 100% classification accuracy, full-key recovery

	Microcontroller	32-bit FPGA	128-bit FPGA	512-bit FGPA
Throughput (Gb\s)	0.046	1.536	6.144	24.576
Classification Accuracy (%)	94.8	62.9	99.1	100.0

Hash-based — Power Collision Attack

Hash each ciphertext with SHAKE-128 and compare the 128-byte digests instead of raw bytes

- One byte changes will propegate to a completely different digest
- **FPGA:** plane-pipelined Keccak core, 5 absorb cycles over the 768-byte input
- Tested randomized row scheduling to add temporal noise

IT STILL LEAKS

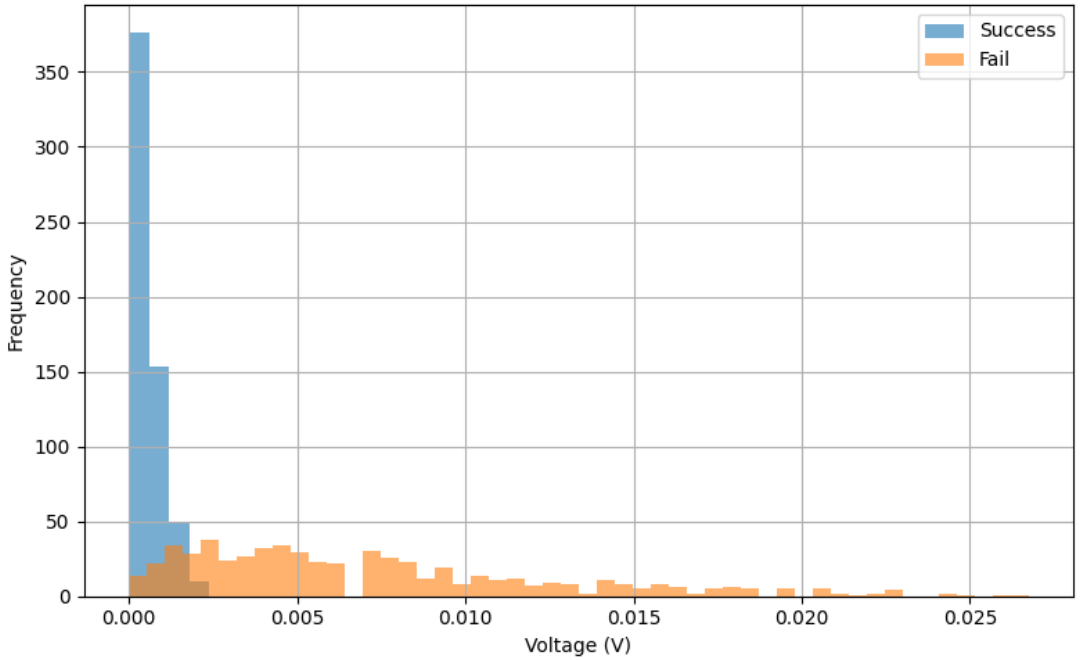
SHAKE-128: 24 permutation rounds

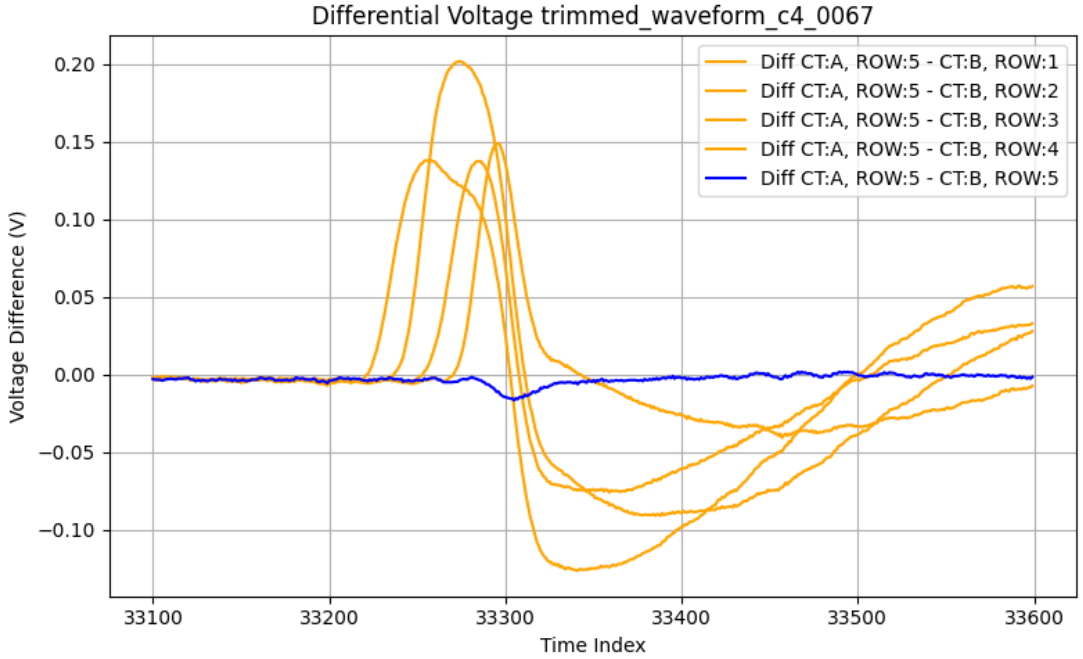
Round 1 (Rho/Pi) still works on the same input bytes

Comparing digests hides the final value, but the early-round power signature still depends on the input, so the leakage survives

Next — results: the first-round leak still classifies success vs failure, and shuffling is fingerprintable

Hash-based — Results





Early-round leakage, 94.7% classification accuracy, targeted the Rho/Pi step of the first Keccak round, failure spreads wide while success clusters near zero

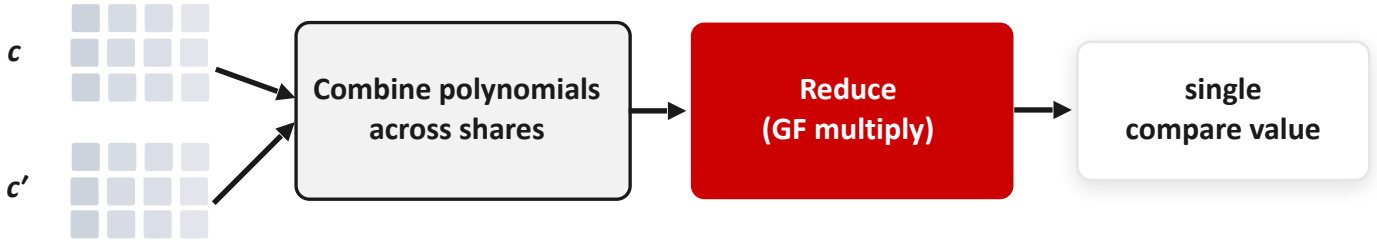
Shuffling undone, 99.9% classification accuracy, Only the matching row (blue) stays flat; the rest spike, so the true execution order is identifiable

Masked Comparison — How it works

Split each ciphertext into n random shares, compare across the shares, and reduce to one match/ mismatch value

- No step reconstruction of the original ciphertexts so secure against t-probing
- **FPGA:** All shares run in parallel, so the leakage is measurable at one time point vs. the microcontroller leaking one bit of key information across many time points

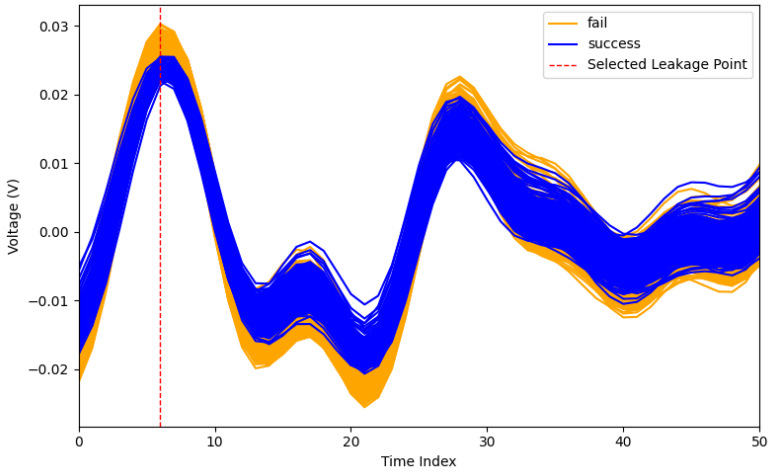
HOW THE MASKED COMPARE RUNS



(n random shares each)

Never rebuilds c or c', t-probing secure in theory

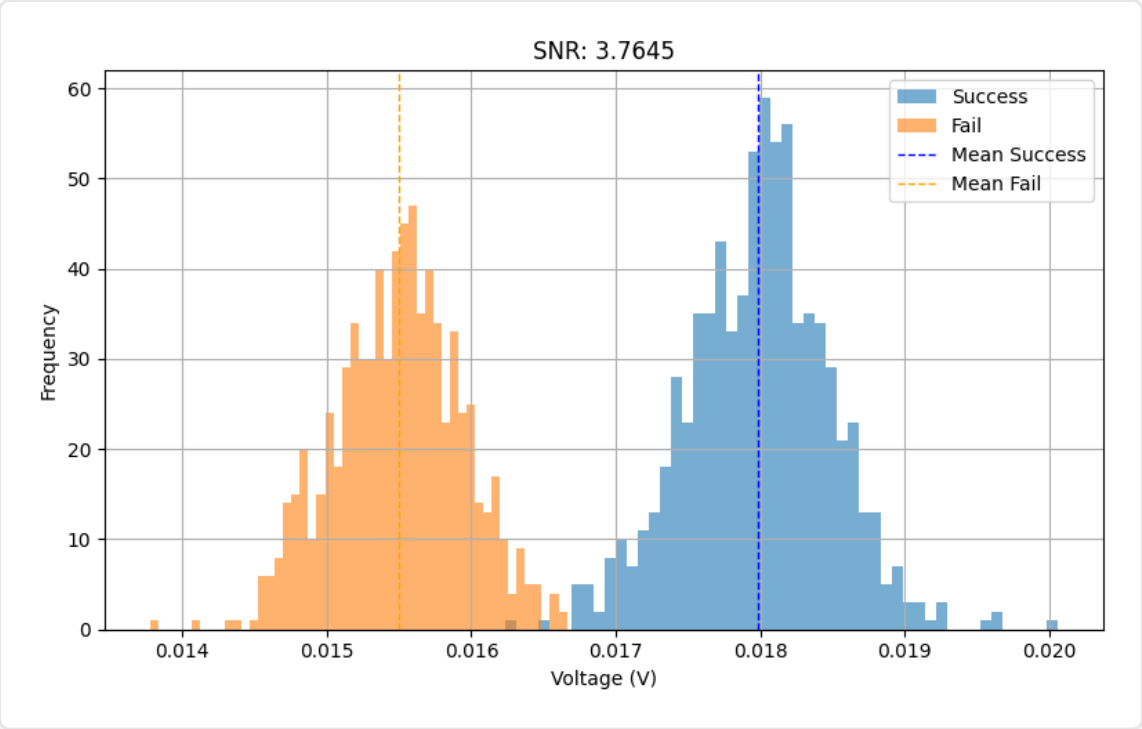
GF-multiply reduction breaks it, leaks in higher orders



FPGA leakge trace

Next — results: even while masked, success and failure are classifiable, and 6 shares leak more than 4

Masked comparison — Results



Masked comparison (Galois-Field reduction): success and failure still separate cleanly (SNR 3.76)

CLASSIFICATION ACCURACY

Microcontroller [prior, ref. 6] *0.001 Gb/s*	**95.0%**
FPGA - 4-Share *0.542 Gb/s*	**97.9%**
FPGA - 6-Share *0.542 Gb/s*	**98.5%**

Conclusion — Securing real deployments

STRONGEST

Limit key reuse, separate auth mechanism

Use ML-KEM keys as ephemeral session secrets and re-key often; handle long-term authentication with a separate primitive. Limited traces are impossible to recover the key with

PROMISING

Mask earlier in the pipeline

Move masked verification before ciphertext compression so protection is distributed across decapsulation, still under study

REQUIRED

Hardware-specific leakage models

Design countermeasures for the parallel execution model; re-evaluate across FPGA generations & process nodes.

NOT ENOUGH

Dummy ops / simple masking

Obscure but do not eliminate leakage — statistical filtering recovers the signal given enough traces, at a throughput cost.

ML-KEM is excellent for establishing ephemeral symmetric session keys. But re-keying and authentication must be handled separately to shrink this attack surface. iMessage and Signal PQXDH already handle keys this way

DISCUSSION

Thank you — questions?

Exploring Side-Channel Protections in Hardware Implementations of PQC ML-KEM Verification

Davis Ranney | Yashaswini I. Makaram | A. Adam Ding | Yunsi Fei — Northeastern University

ranney.d | imakaram.y | a.ding | y.fei *@northeastern.edu*

DSN 2026 - Practical Experience Report - This work was supported in part by the National Science Foundation (NSF) under grants SaTC-1929300, CNS-1916762, and industry partners of NSF IUCRC CHEST