PQSHIELD

Efficient Threshold ML-DSA

ACM CCS '25

Sofia Celi (Brave Research & Univ Bristol),
Rafael del Pino, Thomas Espitau,
Guilhem Niot, Thomas Prest (PQShield)

Problem & Motivation

- Real deployments require distributed trust
- **ML-DSA** (FIPS 204) signatures lack an efficient threshold version (TSS)
  - Prior proposals have too many rounds, trust assumptions too strong
- Main difficulty: distributing its **rejection sampling mechanism**

Our contributions

1. **First Efficient ML-DSA TSS:** < 1s signing, ≤6 parties, no trusted setup
2. Distributed KeyGen or A Posteriori Key Sharing
3. Open implementation (Golang)

Technical Approach

Secret sharing with short shares

Signing

Local per-party rejection sampling + Global size rejection

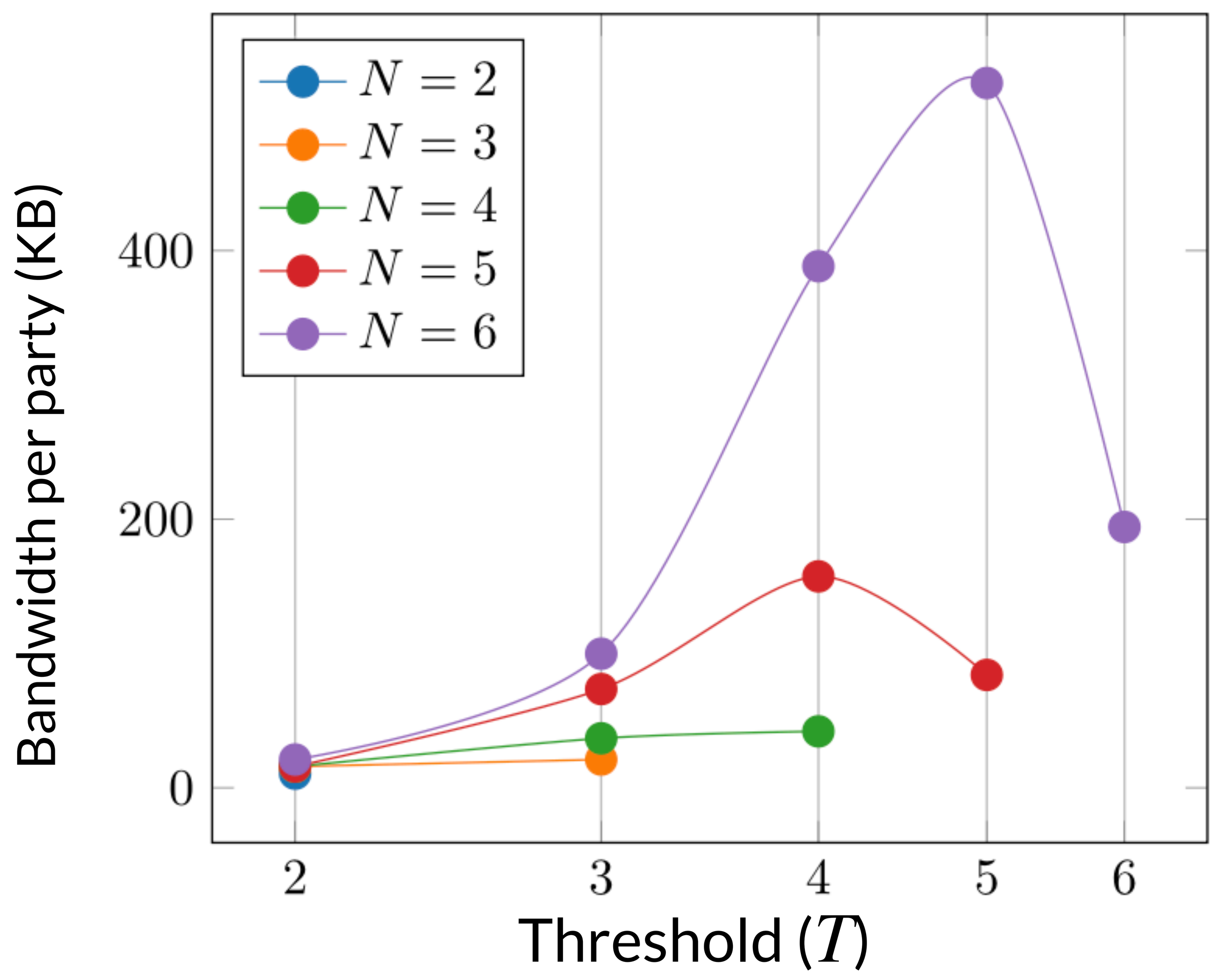
Optimizations

- Rejection sampling with **Unbalanced Hyperballs** for improved success rate
- Optimal share reconstruction using max-flow problem resolution
- Parallel protocol repetitions

Key Features

1. **Backward compatible:** Drop-in integration with ML-DSA verifiers.
2. **Flexible Key Gen:** Both distributed and a posteriori keygen supported.
3. **High Performance:** Signing in < 1s, even in distributed WAN setting.
4. **Provably Secure:** As secure as the original ML-DSA.

Performance Evaluation



WAN experiments (one signing attempt)
L = London, S = Seoul, T = Taipei, V = Virginia

(T,N)	Locations	Signing (ms)
(2,6)	T - S	27
(2,6)	T - V	620
(4,6)	T - V - L - L	750
(6,6)	T - V - L - L - S - S	659

Comparison with other works

Scheme	Parties	Round	Comm (MB)	Security
Our work	≤ 6	6	0.021 to 1.05	Dishonest Majority
Bienstock et al.	Unlimited	96	>1.2	Honest Majority
		24	>2.3	
Trilithium	2	60	234	Trusted Party

+ our approach has lower **computation costs.**

Further information:
Source code and artifacts available at
https://github.com/GuilhemN/threshold-ml-dsa