NDSS SYMPOSIUM

Looma

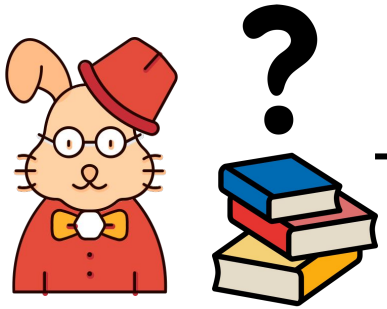
A Low-latency PQTLS Authentication Architecture for Cloud Applications

Xinshu Ma, Michio Honda (University of Edinburgh)

looma

THE UNIVERSITY OF EDINBURGH

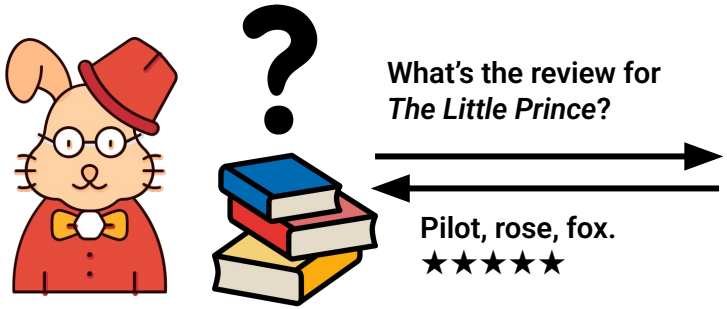
Cloud Applications



What's the review for
The Little Prince?

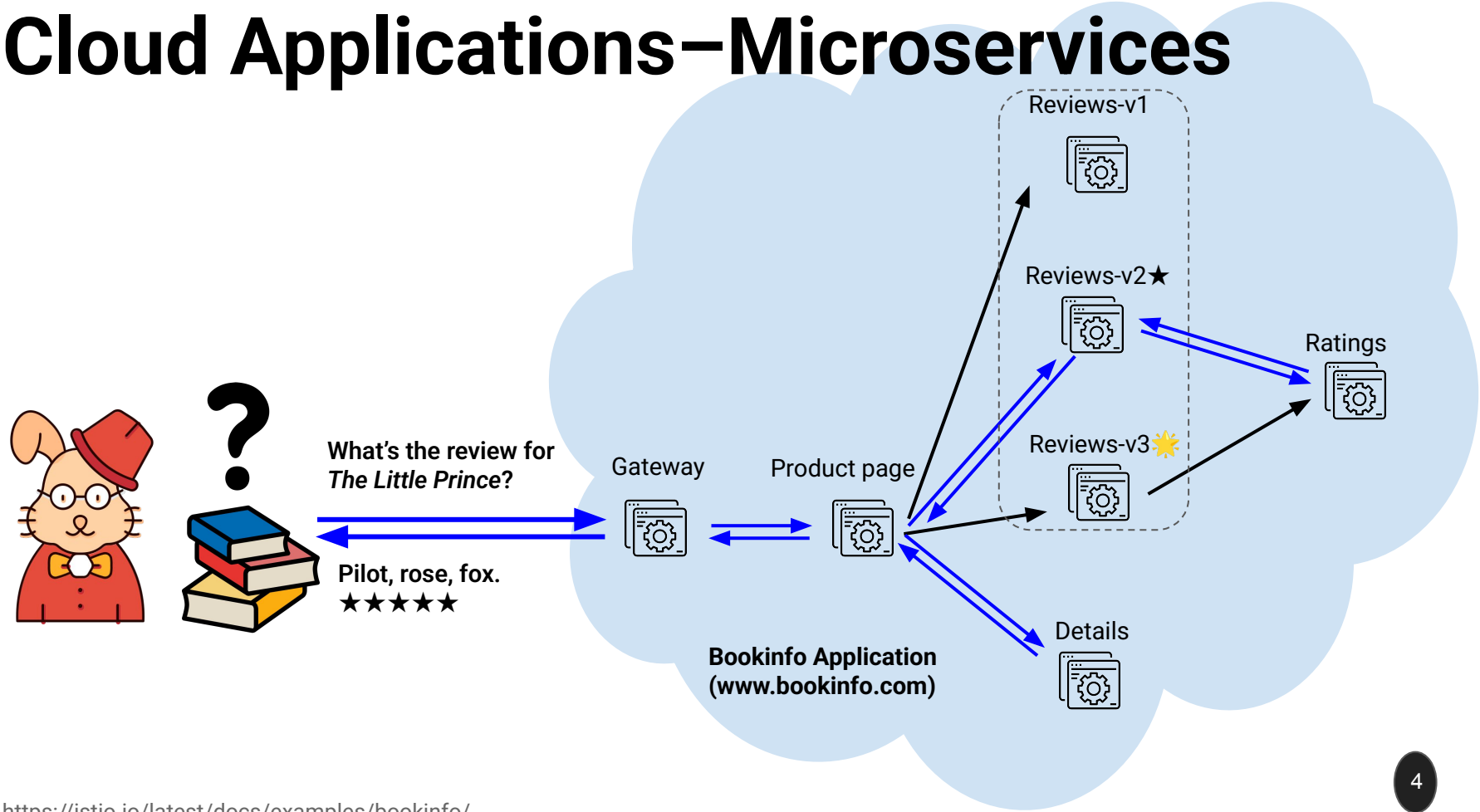
Bookinfo Application
(www.bookinfo.com)

Cloud Applications



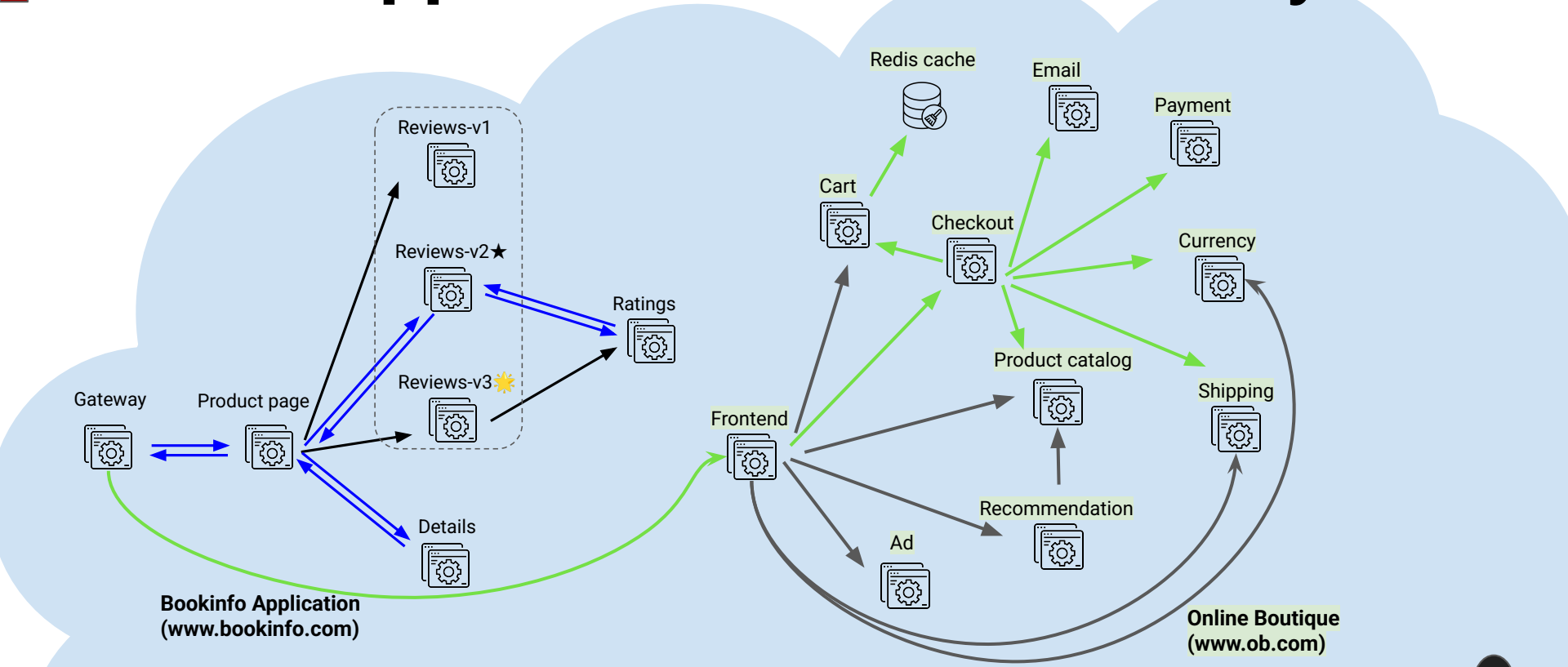
Bookinfo Application
(www.bookinfo.com)

Cloud Applications–Microservices



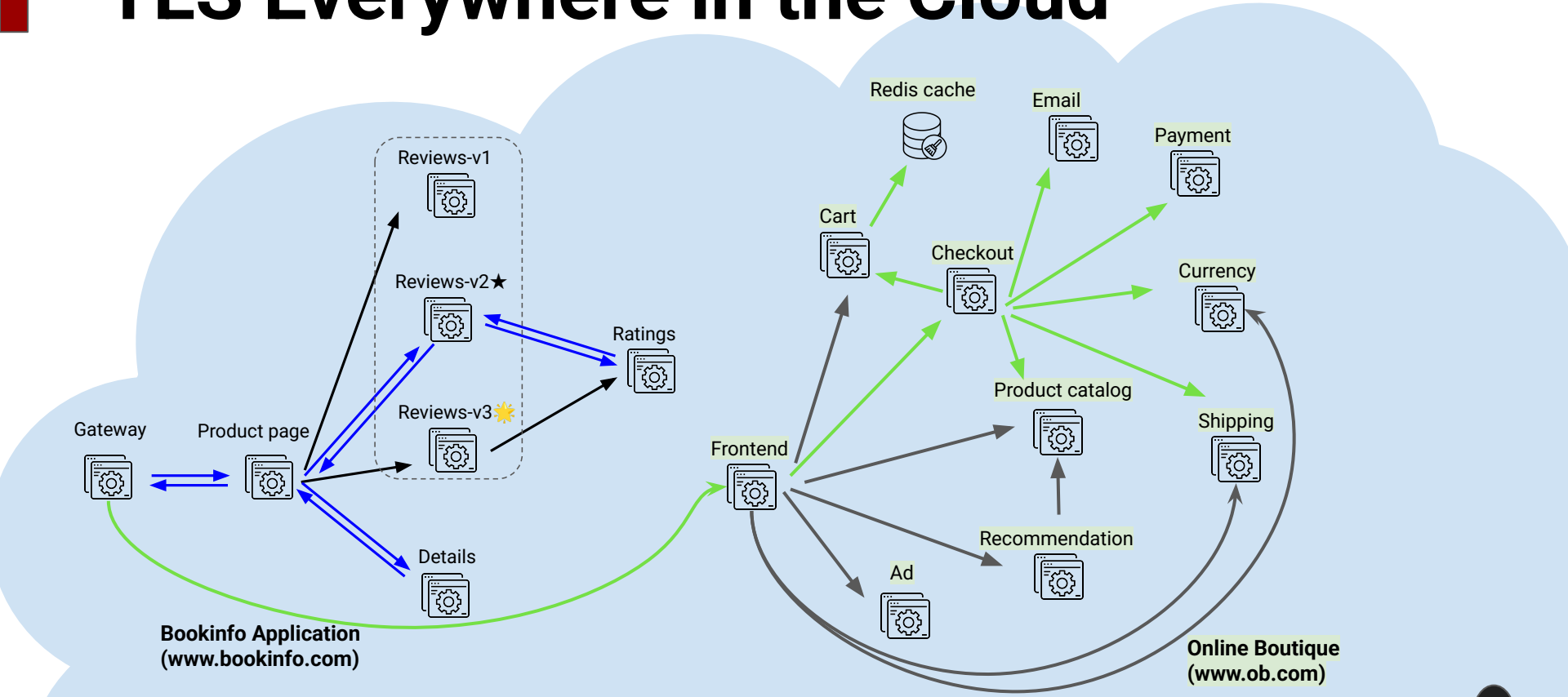
https://istio.io/latest/docs/examples/bookinfo/

Cloud Applications–Multi-Tenancy



https://github.com/GoogleCloudPlatform/microservices-demo

TLS Everywhere in the Cloud



https://github.com/GoogleCloudPlatform/microservices-demo

Why Fast Handshake Matters

Cloud needs fast TLS handshakes

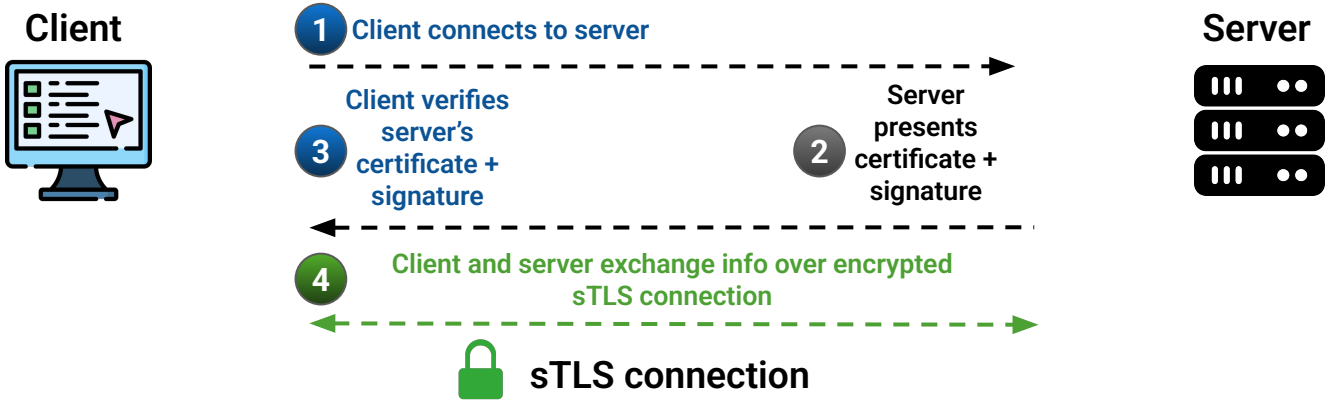
1. Microservices create many short connections → frequent handshakes.

2. With short-lived inter-service flows, handshake overhead can dominate end-to-end time.

3. Cloud systems demand microsecond-scale latency.
   a. Internet RTT: ~20-100 ms
   b. Intra-datacenter RTT: tens of μs (e.g., <50 μs)

Problem 1: mTLS Adds Overhead

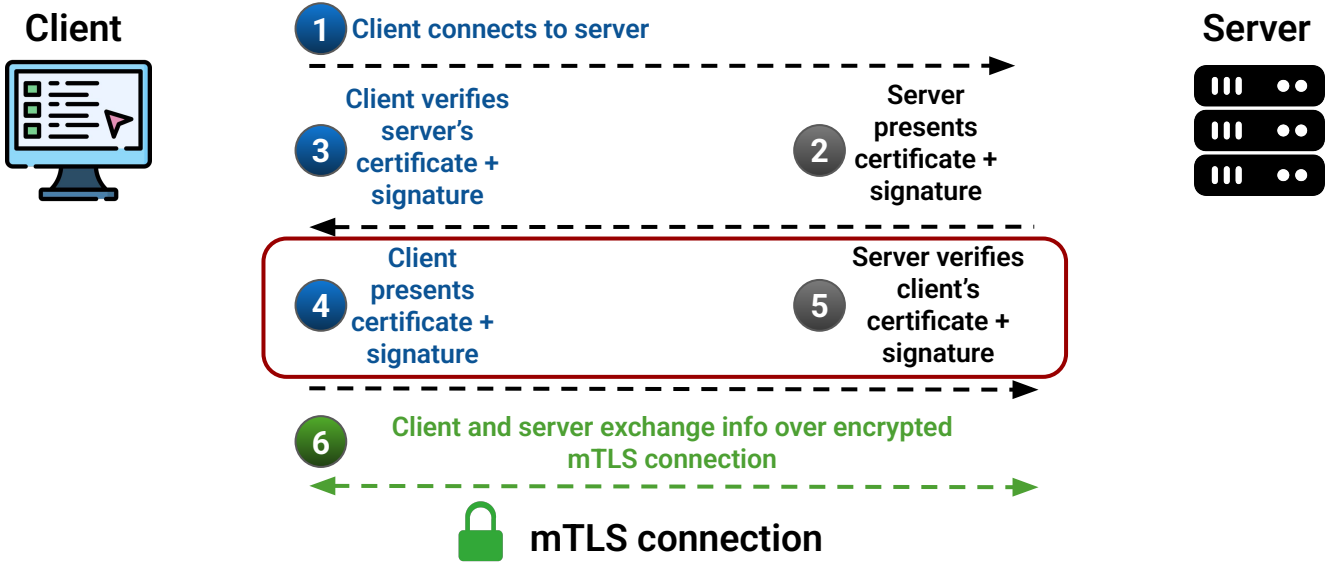
sTLS: client authenticates server



Handshake	Keygen+ex	Verify cert	Sign	Verify
$sTLS_c$	●	●	○	●
$sTLS_s$	●	○	●	○

Problem 1: mTLS Adds Overhead

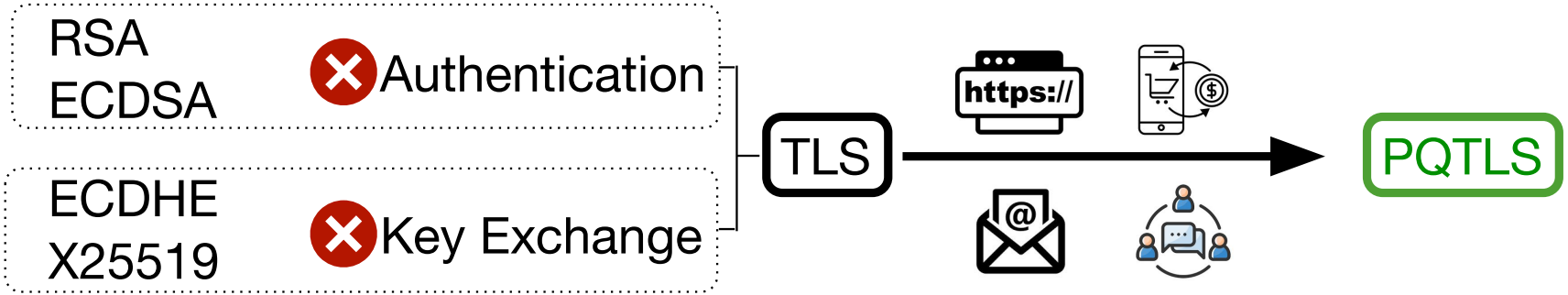
mTLS: mutual authentication



Handshake	Keygen+ex	Verify cert	Sign	Verify
$sTLS_c$	●	●	○	●
$sTLS_s$	●	○	●	○
$mTLS_{c,s}$	●	●	●	●

Problem 2: PQTLS Amplifies the Pain

The crypto shift is coming: TLS → Post-Quantum TLS



Problem 2: PQTLS Amplifies the Pain

Example: Handshake crypto overhead breakdown

Operation	Algorithm	Latency (µs)	PQ	Library
Keygen+ex	ECDHE[a]	45.20	○	OpenSSL
	KEM[b]	74.70	●	OQS
Verify cert	RSA-2048	17.22	○	OpenSSL
	Dilithium-2	26.27	●	OQS
	Falcon-512	30.7	●	OQS
Sign	ECDSA	14.92	○	OpenSSL
	RSA-2048	200.37	○	OpenSSL
	Dilithium-2	51.74	●	OQS
	Falcon-512	149.2	●	OQS
Verify	ECDSA	40.18	○	OpenSSL
	RSA-2048	15.32	○	OpenSSL
	Dilithium-2	21.27	●	OQS
	Falcon-512	28.3	●	OQS

PQ

PQ schemes have cheap verify but expensive sign.

Existing Solutions

Mostly targets key exchange, not authentication.

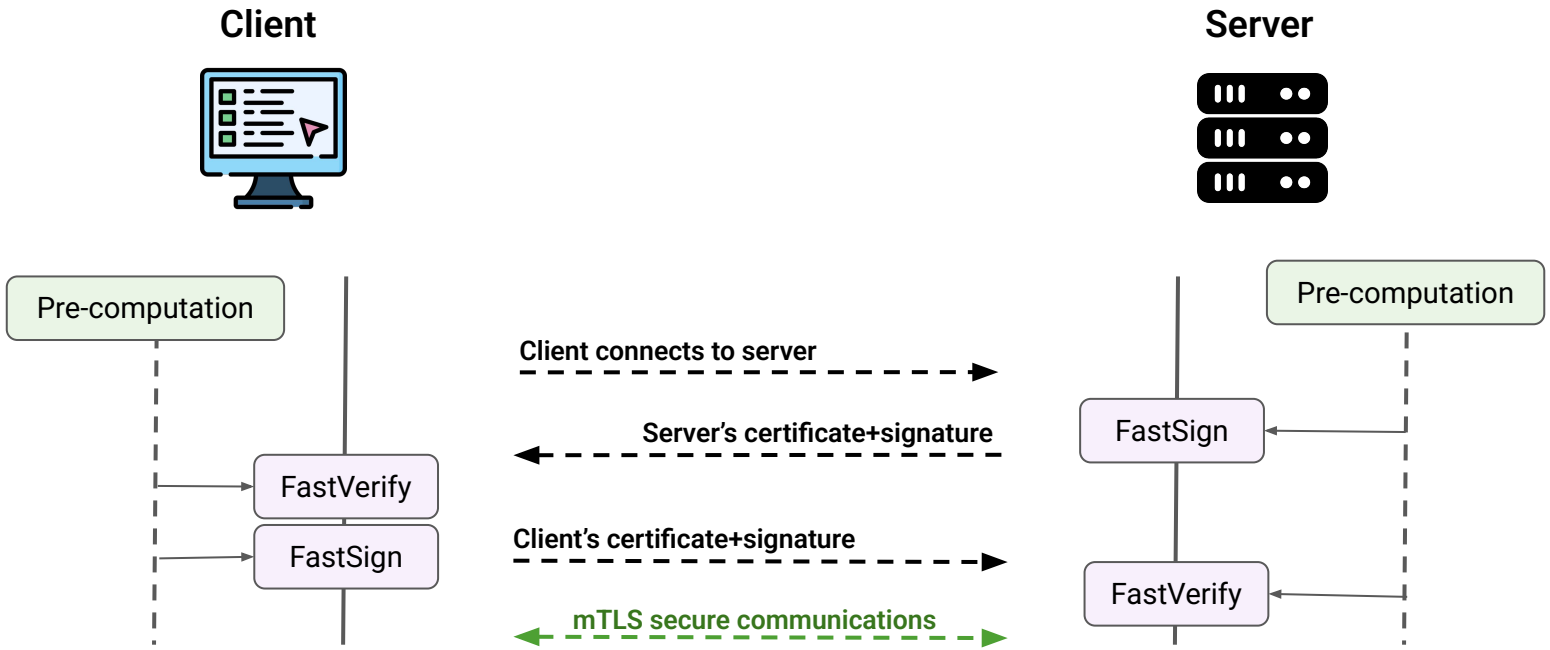
A. Offloading TLS handshake
   a. SmartNICs [Asia-Pacific'20]
   b. GPUs [NSDI'11]
   c. Specialized key server [SIGCOMM'24]

B. Network protocol enhancement
   a. zTLS [WWW'23]
   b. KEMTLS [CCS'20]

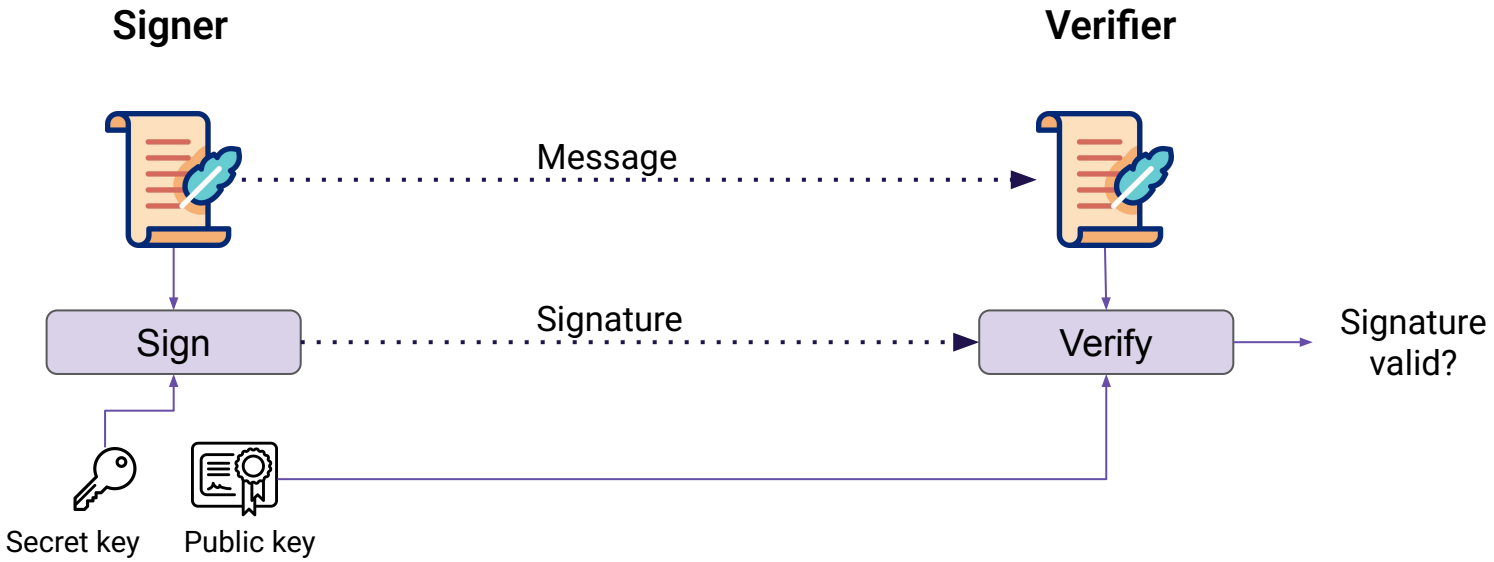
C. Cryptographic operation optimization
   Focus on key exchange acceleration [S&P'22, S&P'24]

Looma: Decoupling Expensive Auth

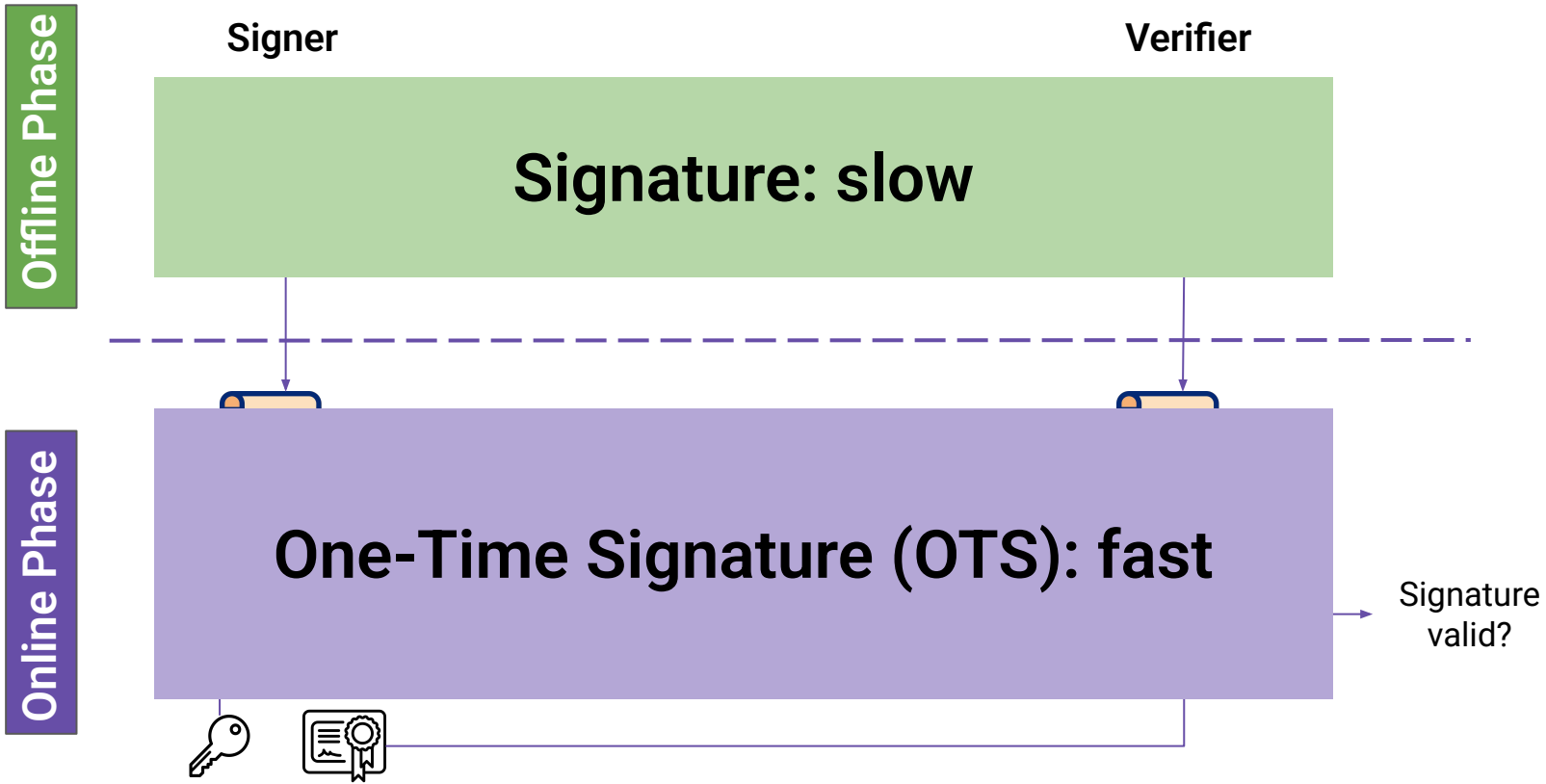
Do expensive work early; keep the online path fast



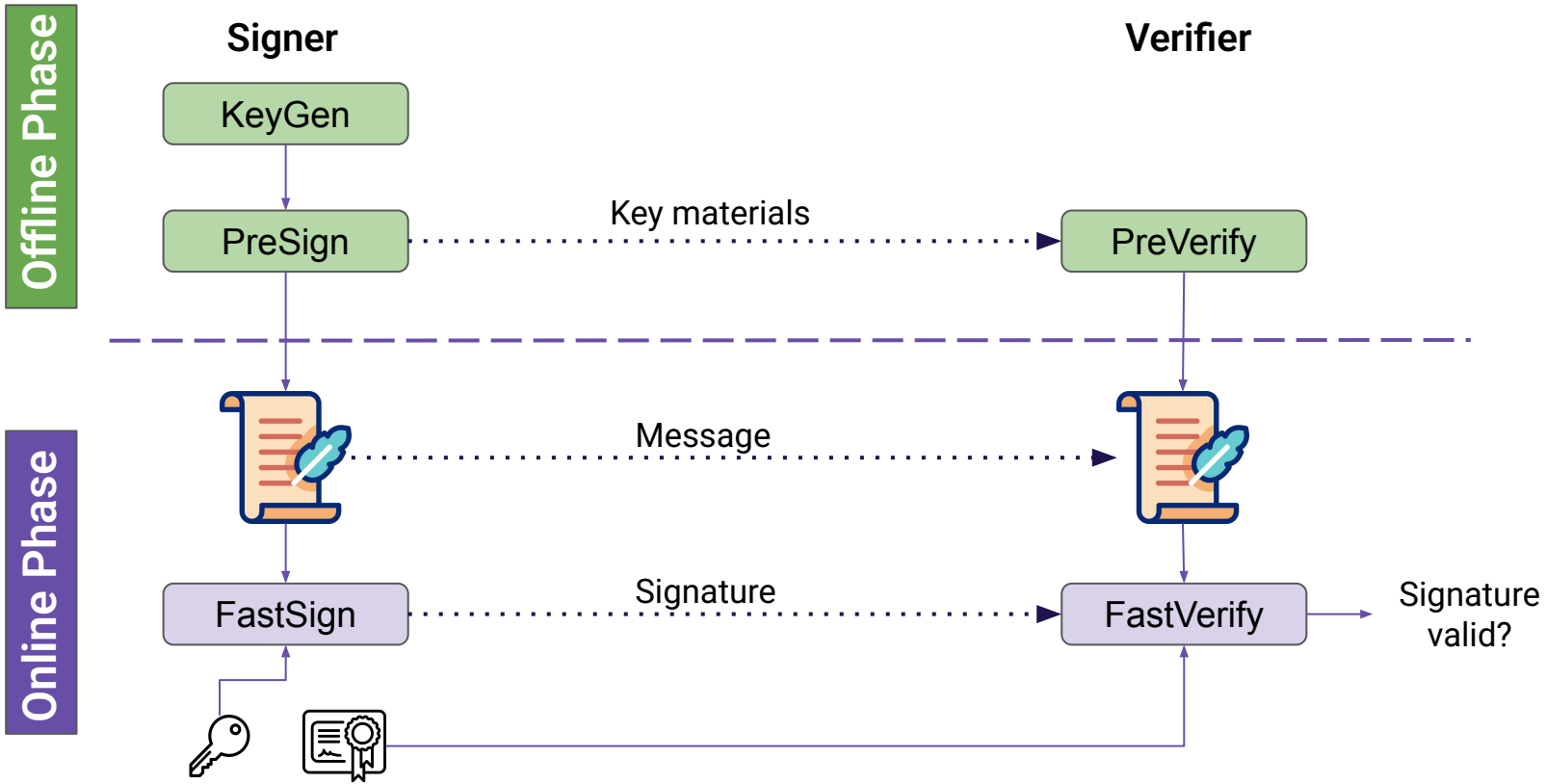
Normal Signature Workflow



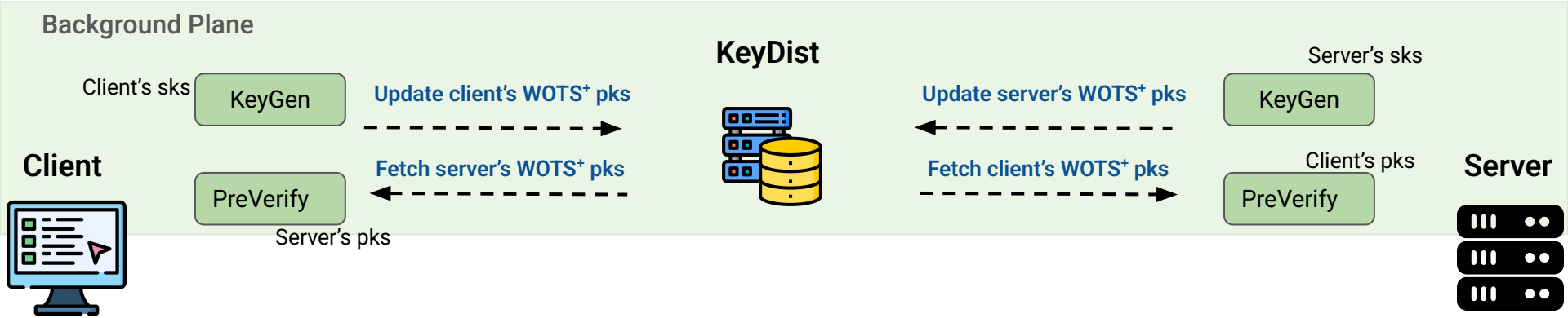
Online/Offline Signature Paradigm



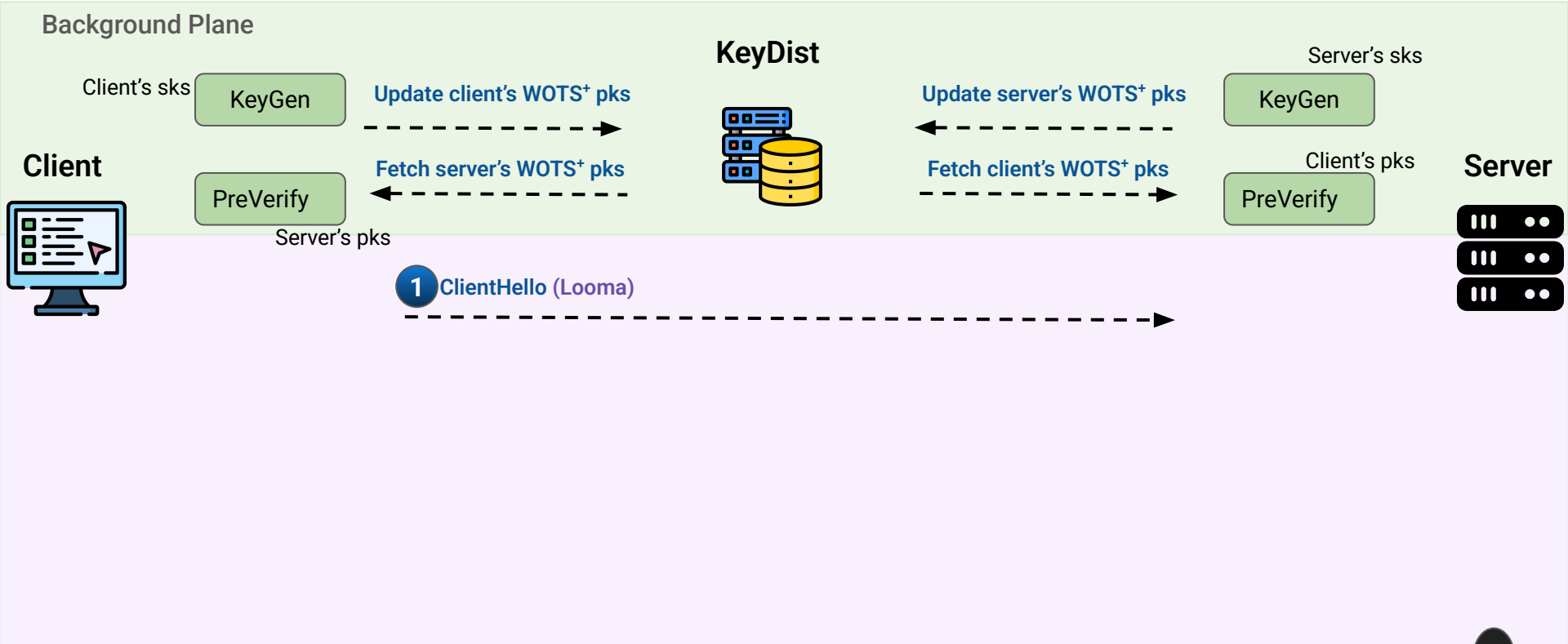
Online/Offline Signature Paradigm



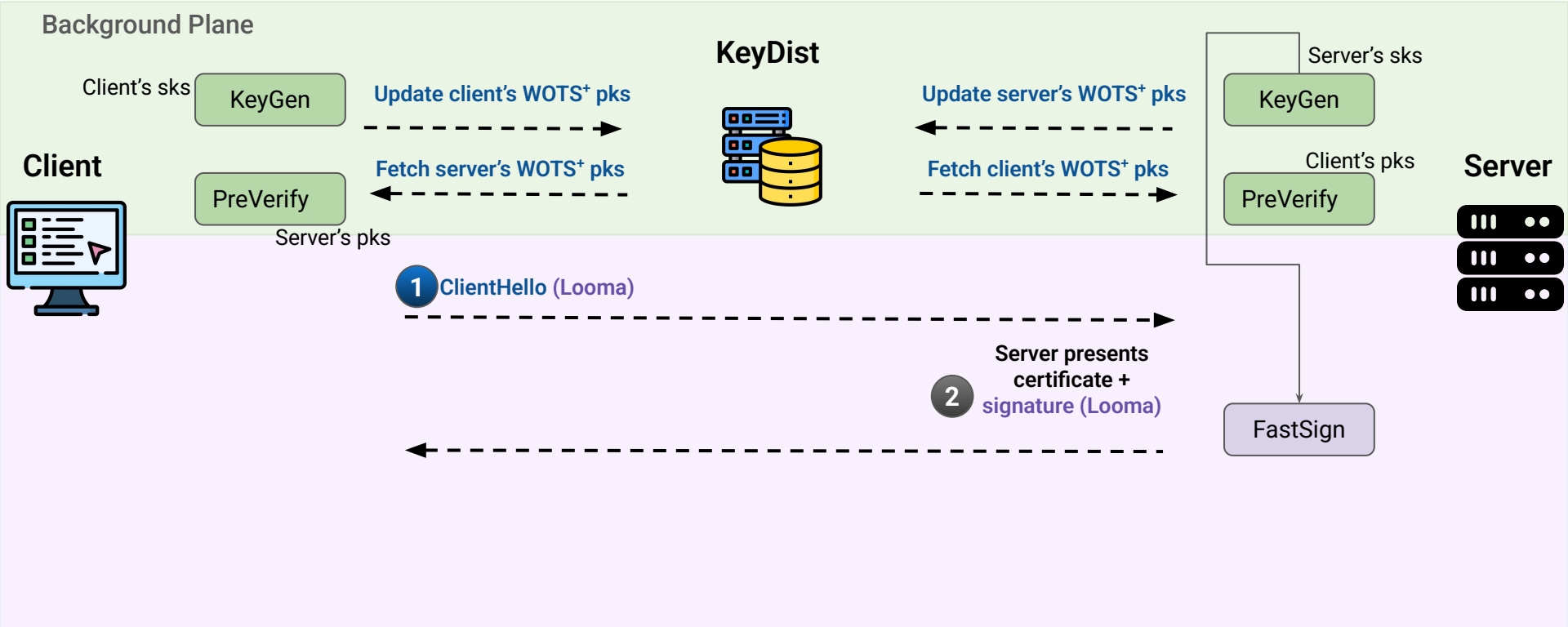
Looma: KeyDist + Offline Tasks



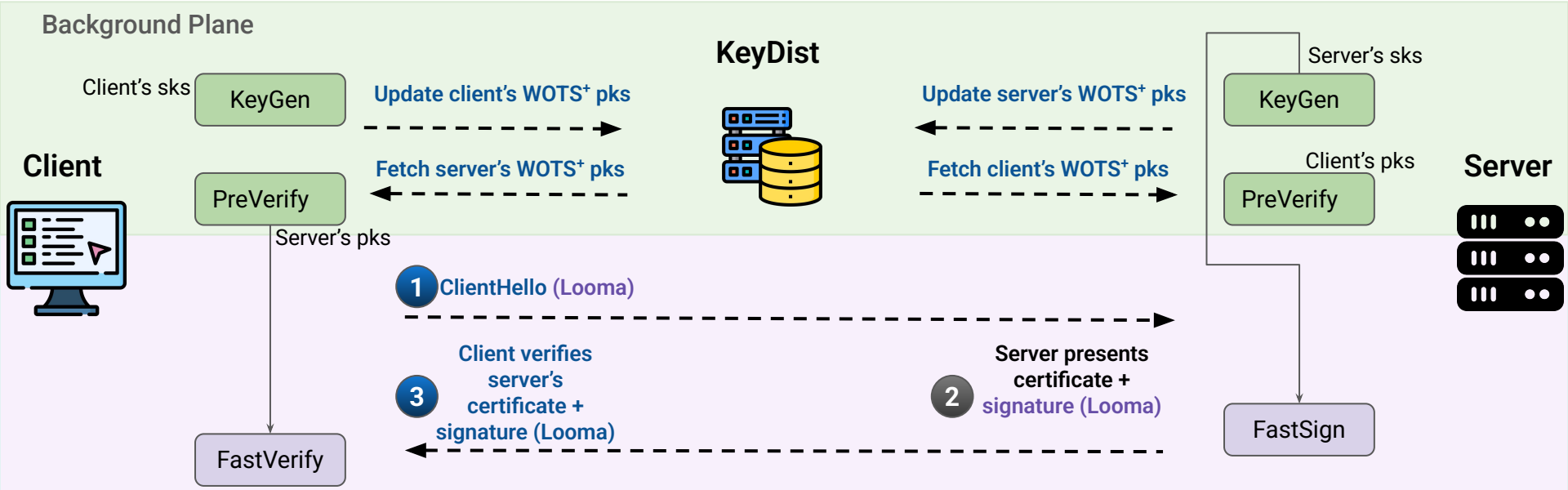
Looma: Client Starts PQTLS Handshake



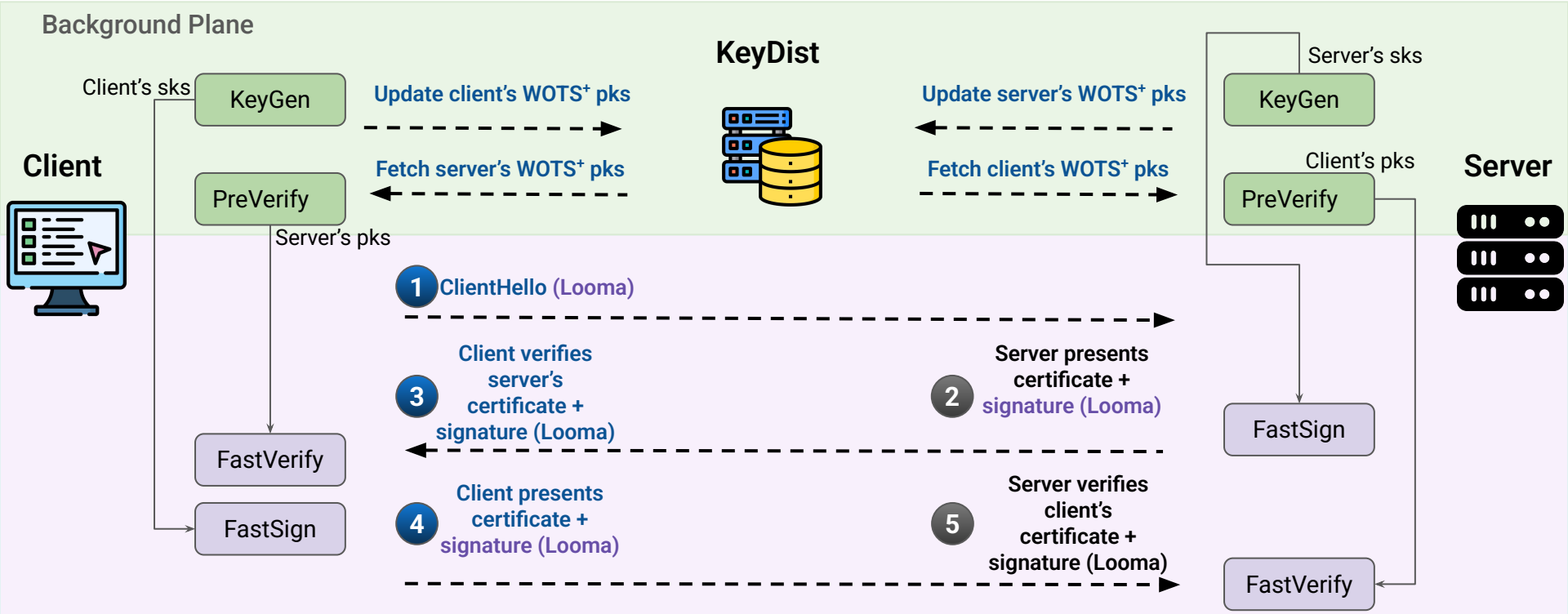
Looma: Server Signs



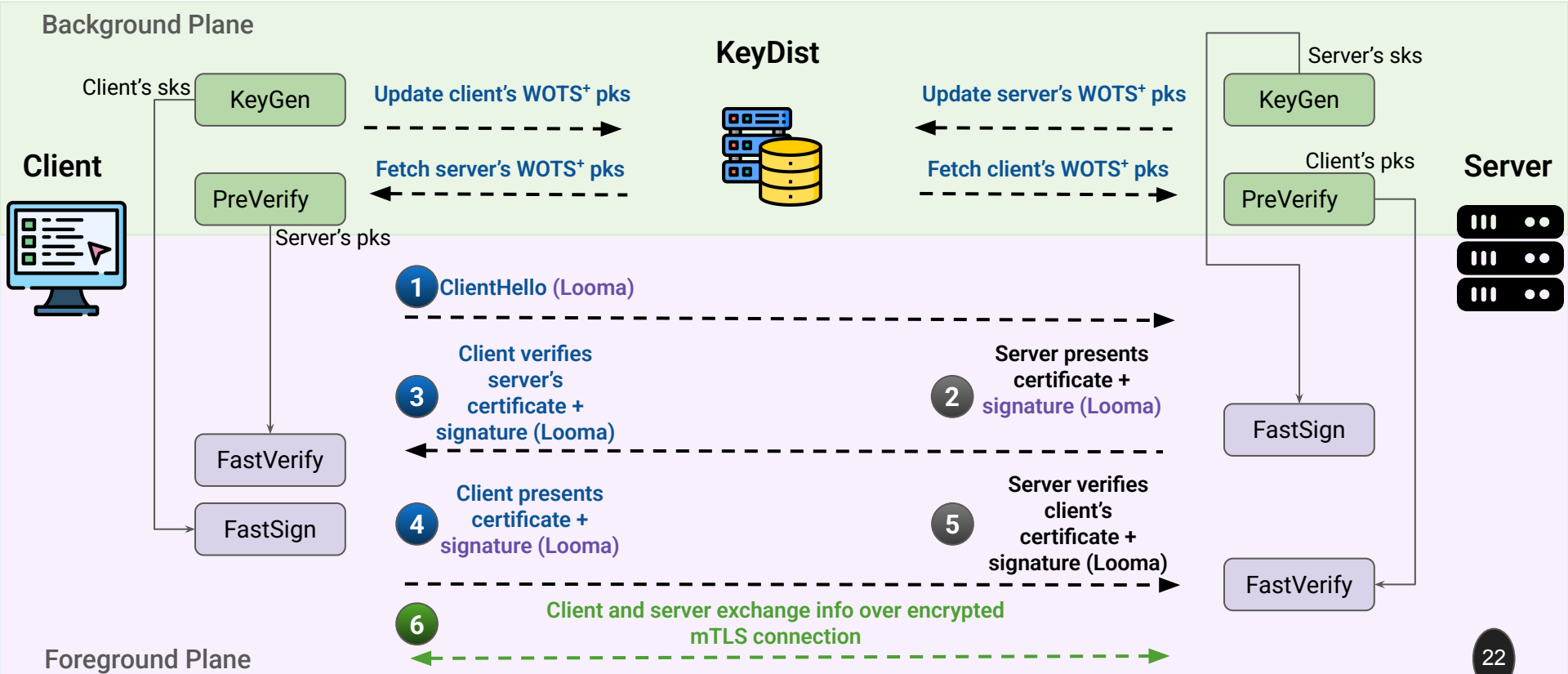
Looma: Client Verifies



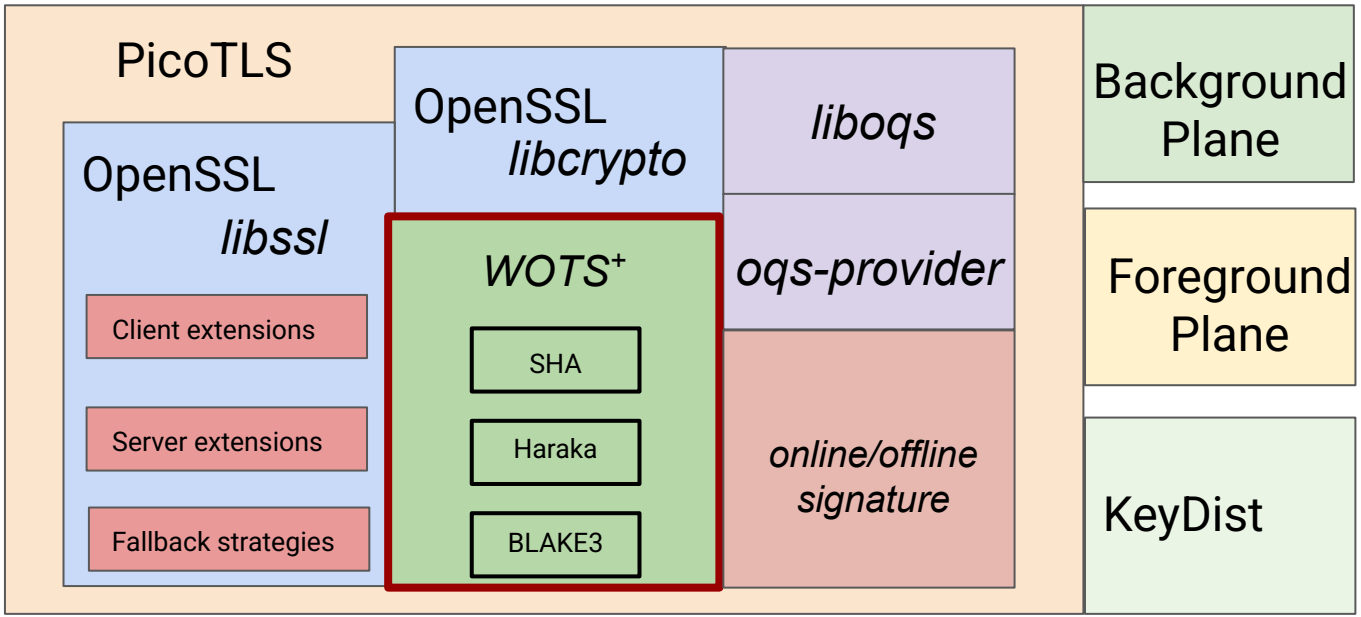
Looma: Server Authenticates Client



Looma: mTLS Completed



Looma: Implementation



Instantiation: Dilithium-2 + WOTS$^+$

Looma: Implementation Optimizations

TABLE IV: Signature size and per-operation latency of WOTS$^+$ signature across three modern hash families.

w	ℓ	Sig/PK/SK size (B)	Key-gen (µs)				Sign (µs)	Verify (µs)			
			SHA256	BLAKE3	Haraka	Haraka8x		SHA256	BLAKE3	Haraka	Haraka8x
2	265	8,480	35.57	47.23	11.90	7.18	0.62	20.45	23.12	6.43	4.60
4	133	4,256	51.17	68.60	15.26	10.10	0.38	26.82	36.33	8.64	6.43
8	90	2,880	79.42	106.49	22.94	15.42	0.29	72.86	95.33	19.93	13.98
16	67	2,144	128.01	169.33	35.73	24.61	0.25	58.89	83.77	17.16	13.50
32	55	1,760	213.03	290.00	60.45	43.09	0.24	198.58	271.58	52.60	38.63
64	45	1,440	353.73	475.80	99.21	71.07	0.21	308.46	434.51	97.51	70.12

- ➔ WOTS$^+$ sign optimization
  - ◆ Caching intermediate hashes further speeds up FastSign
- ➔ Choose a fast hash
  - ◆ Eight-way SIMD-accelerated Haraka (Haraka8x)
- ➔ Pick a good Winternitz Parameter (w)
  - ◆ w = 4

Looma Performance (One-to-One)

TABLE VI: TLS handshake latency summary (µs).

Signature Algorithm	sTLS				mTLS			
	Mean	St. Dev.	P50	P99	Mean	St. Dev.	P50	P99
RSA-2048	842.4	206.3	900	1,427	1,095.4	179.2	1,112	1,656
ECDSA-256	422.0	38.9	416	513	462.6	38.6	449	555
ED25519	417.6	47.4	411	511	429.3	57.0	432	534
Dilithium-2	499.9	106.2	485	837	560.5	112.4	548	902
Falcon-512	778.9	117.6	801	902	969.9	56.6	963	1,066
SPHINCS$^+$ SHA256-128f-simple	13,358.5	3,611.7	12,725	18,894	35,289.7	1,264.3	35,471	36,099
Looma	**354.9**	**42.3**	**348**	**447**	**363.6**	**46.2**	**363**	**464**

34% **48%**

Not only improves the median, but also cuts the tail significantly.

Looma Microbenchmarks: (Sign/Verify)

TABLE VII: Sign and verification latency (in μs).

	I2		A2	
Algorithm	Sign	Verify	Sign	Verify
RSA-2048	261.1	19.3	200.7	15.4
ECDSA-256	21.1	62.6	14.9	40.8
Ed25519	27.6	85.2	18.7	54.4
Dilithium-2	61.4	24.4	51.4	21.3
Falcon-512	181.1	34.3	149.2	28.3
SPHINCS$^+$-SHA2-128f	6866.2	524.9	5611.6	426.3
Looma	**0.39**	**6.83**	**0.32**	**6.19**

Looma Performance (Many-to-One)

Concurrent setting

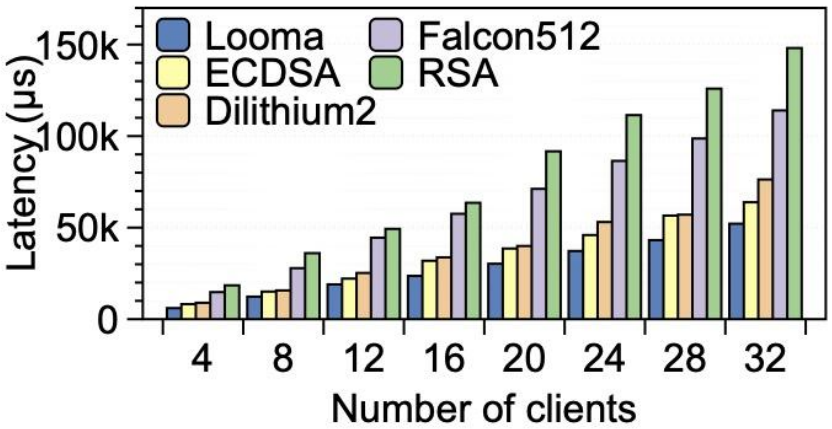


Fig3. mTLS handshake latency p99

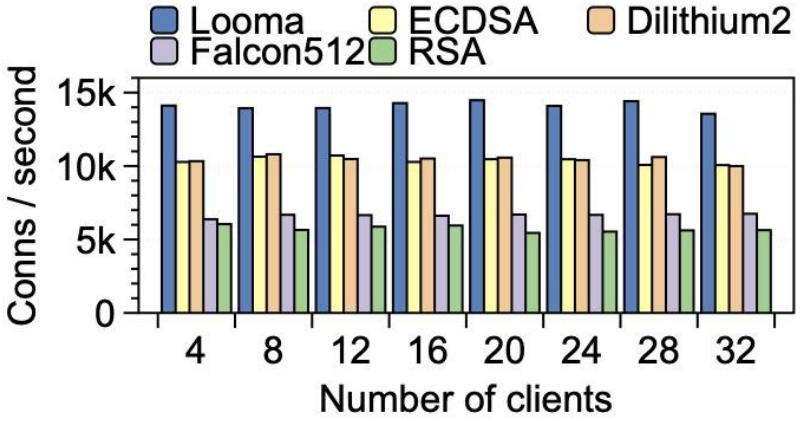
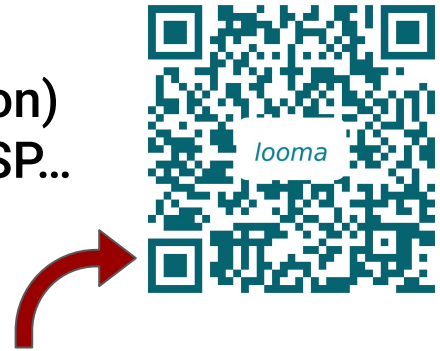


Fig4. mTLS handshake throughput

Takeaways

- Q: Can we make PQ authentication fast enough for cloud mTLS?
- Online/offline signature can work well in the cloud.
- Looma is a general architecture
  - Applies to various PQ signatures (Dil-2, Falcon)
  - Applies to TLS-like protocols: kTLS, QUIC, PSP…



Check out our paper and code!