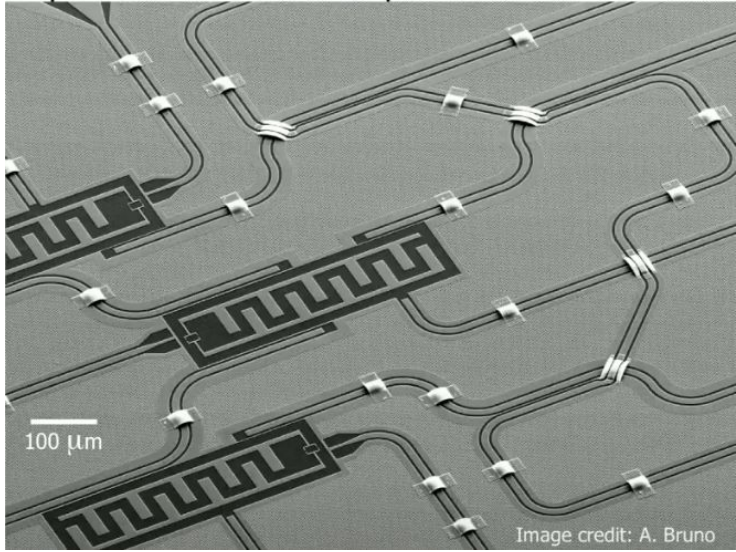


The Road to a Post-Quantum Internet

dr. Bas Westerbaan, Cloudflare Research

CWI symposium Post-Quantum Cryptography IV,Den Haag, 15 nov. 2022

About Cloudflare

We run a global network spanning 275 cities in over 100 countries.

Started of as a CDN and DDoS mitigation company, we now offer many more services, including

- 1.1.1.1, fast public DNS resolver
- Workers, serverless compute
- SASE, to protect corporate networks

We serve nearly 20% of all websites and process 39 million HTTP requests per second.

Building a better Internet

Cloudflare cares deeply about a private, secure and fast Internet, helping design, and adopt, among others:

- TLS 1.3 and QUIC
- DNS-over-HTTPS
- Private relay/OHTTP
- Encrypted ClientHello

And, the topic of this talk:

- Migrating to Post-Quantum Cryptography.

There is not *one* post-quantum migration.

1. Key agreement 🤝

   Communication can be recorded today and decrypted in the future. We need to upgrade as soon as possible.

2. Signatures 

   Less urgent: need to be replaced before the arrival of cryptographically-relevant quantum computers.

Key agreement 🤝

Urgent, and seemingly straightforward.

Seemingly straightforward

- TLS 1.3, the modern protocol behind HTTPS, is designed for secure and gradual upgrades of its underlying cryptography.
- Should then be as simple as upgrading clients and servers, adding support for a new X25519+Kyber hybrid key agreement.

How does Kyber compare to X25519 in performance?

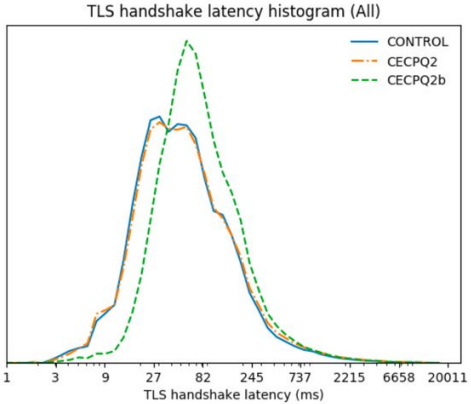
		Size keyshares(in bytes)		Ops/sec (higher is better)	
Algorithm	PQ	**Client**	**Server**	**Client**	**Server**
Kyber512	✅	800	768	50,000	100,000
Kyber768	✅	1,184	1,088	31,000	70,000
X25519	❌	32	32	17,000	17,000

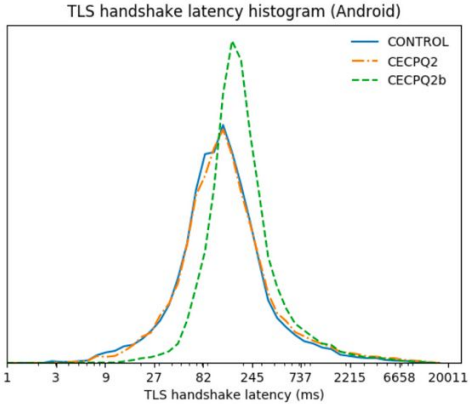
Kyber is very light on the CPU, but has much larger keyshares compared to X25519.

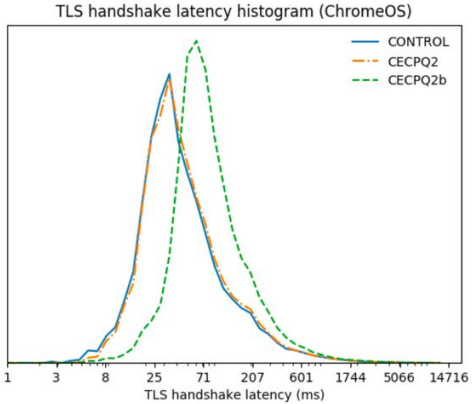
Are these big keyshares a problem?

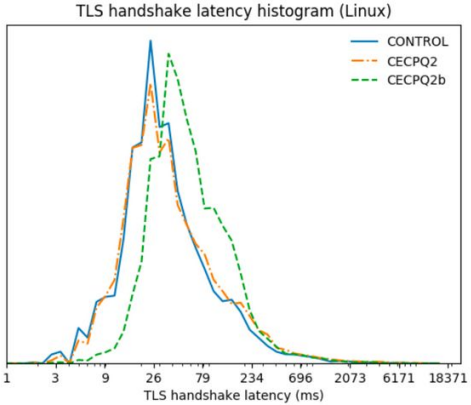
In 2019 we ran an experiment with Google, comparing TLS handshake times between

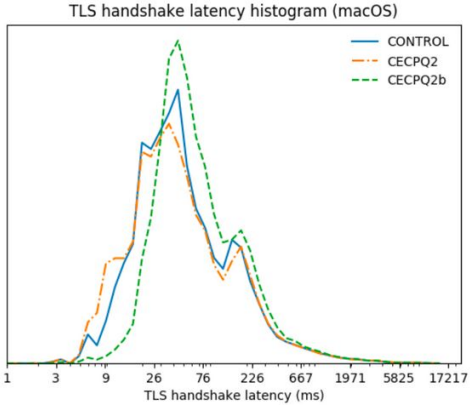
- X25519 (control)
- NTRU-HRSS, which has similar keyshare sizes as Kyber768, but is more demanding on the CPU.
- (SIKE, small keyshares, very demanding on the CPU and recently completely broken.)

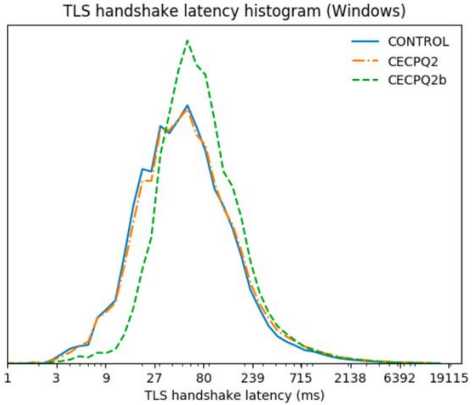












Control is X25519. CECPQ2 is X25519+NTRU-HRSS, and CECPQ2b is X25519+SIKE.

Looks great, but ...

Fragmented ClientHello

- Kyber768 is big enough to fragment the *ClientHello*, the first message sent by the client.
- In our 2019 experiment, we saw some misbehaving middleboxes dropping fragmented *ClientHello*s.
- This is harder to get right for QUIC, and we expect issues with stateless QUIC loadbalancers.
- Kyber512 will also fragment when combined with long SNI or Encrypted ClientHello (ECH).

Moving forward

Last month we deployed Kyber512 and Kyber768 hybrids worldwide, so early adopters can test for these issues.

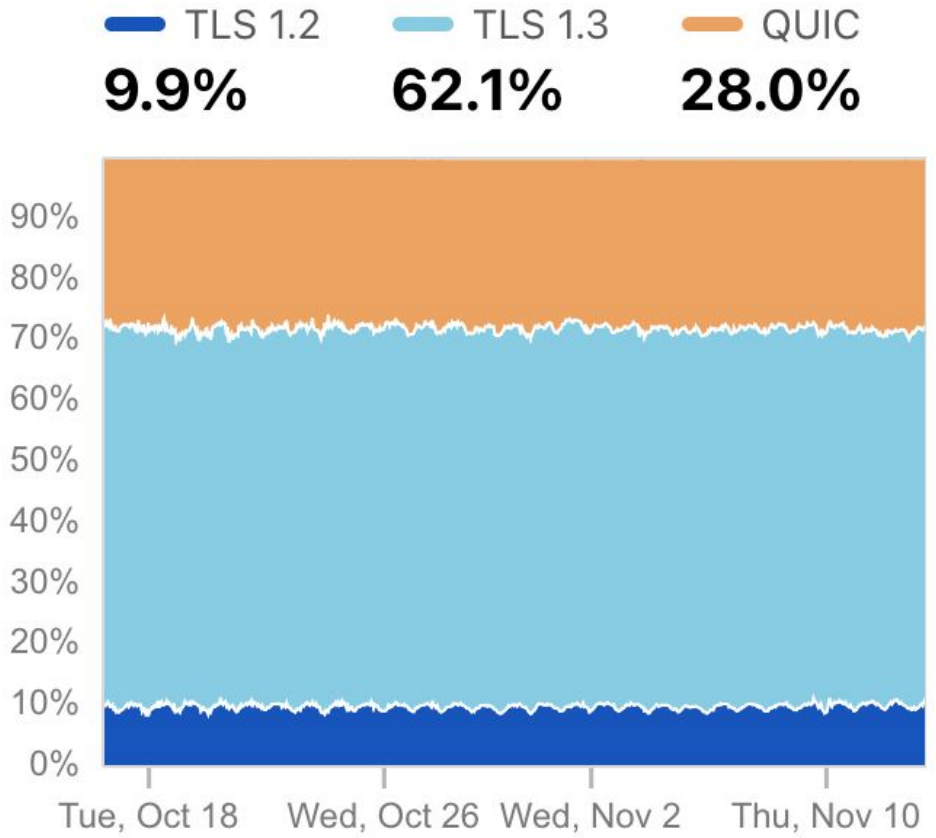
Firefox is expected to start sending these keyshares later this month.

If the fragmented ClientHello is problematic, we will try out various workarounds.

On a positive note

Early versions of TLS 1.3 were completely undeployable because of... middlebox issues.

After several iterations of testing and adding workarounds, the final version of TLS 1.3 is a success, used by 90% of our visitors.



Cloudflare Radar

Key agreement 🤝

Urgent and seemingly straightforward to deploy, but could very well be delayed by requiring workarounds for misbehaving middleboxes.

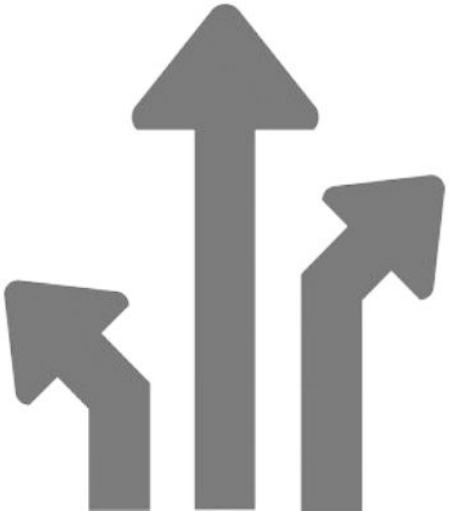
That’s why we’re deploying early.

Signatures 

Less urgent, but more challenging.

#1, many more parties involved:

Cryptography library developers, browsers, certificate authorities, HSM manufacturers, CT logs, and every server admin that cobbled together a PKI script.



#2, there is no all-round great PQ signature

		Size (bytes)		CPU time (lower is better)	
	PQ	Public key	Signature	Signing	Verification
Ed25519	❌	32	64	1 (baseline)	1 (baseline)
RSA-2048	❌	256	256	70	0.3
Dilithium2	✅	1,312	2,420	4.8	0.5
Falcon512	✅	897	666	8*	0.5
SPHINCS⁺128s	✅	32	7,856	8,000	2.8
SPHINCS⁺128f	✅	32	17,088	550	7

Online signing — Falcon's Achilles' heel

- For fast signing, Falcon requires a floating-point unit (FPU).
- We do not have enough experience running cryptography securely (constant-time) on the FPU.
- On commodity hardware, Falcon should not be used when signature creation can be timed, eg. TLS handshake.
- Not a problem for signature verification.

#3, there are many signatures on the Web

- Root on intermediate
- Intermediate on leaf
- Leaf on handshake
- Two SCTs for Certificate Transparency
- An OCSP staple

Typically 6 signatures
and 2 public keys
when visiting a website.

(And we're not even counting DNSSEC.)

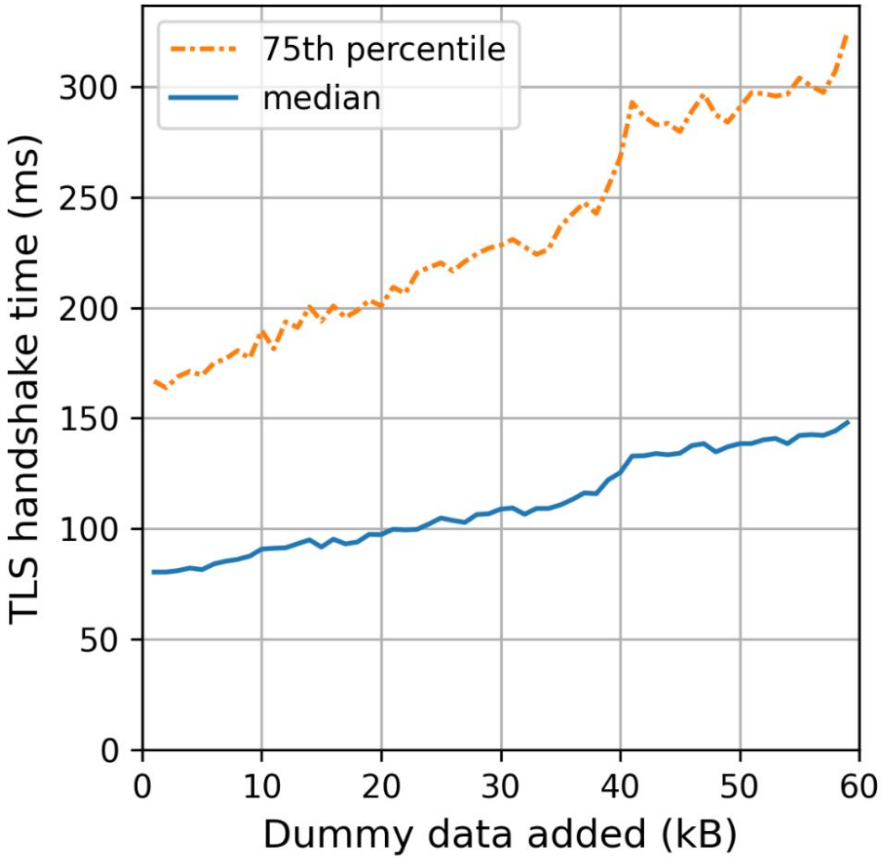
Using only Dilithium2

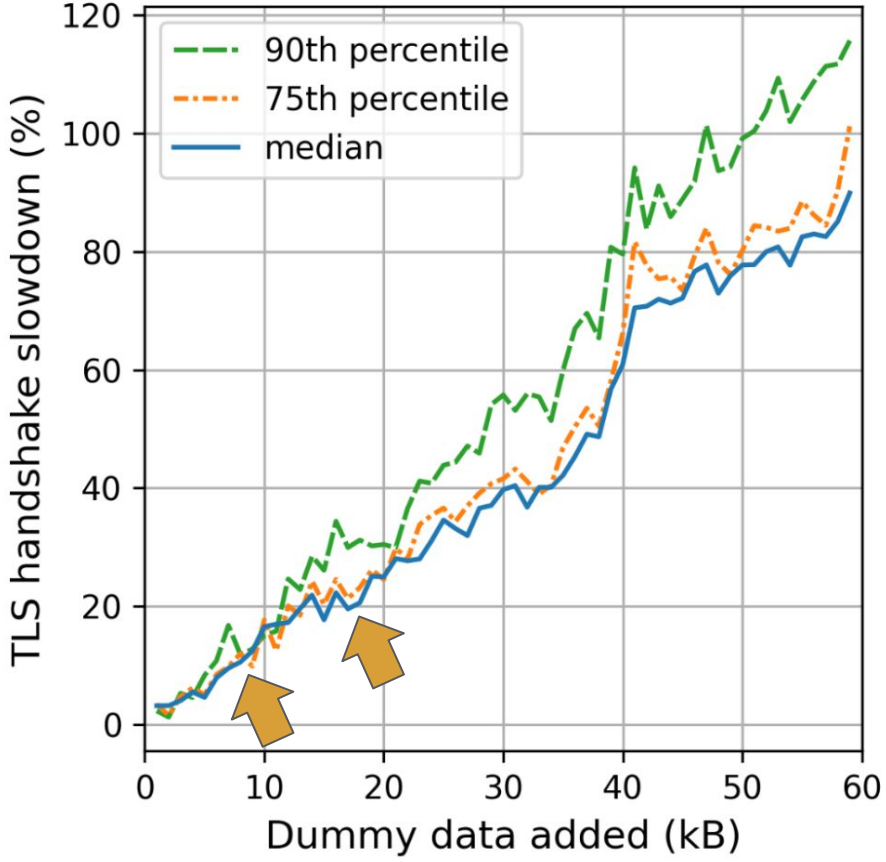
+17,144 bytes

Using Dilithium2 for the TLS handshake and Falcon for the rest

+7,959 bytes

Is that too much? We had a look...



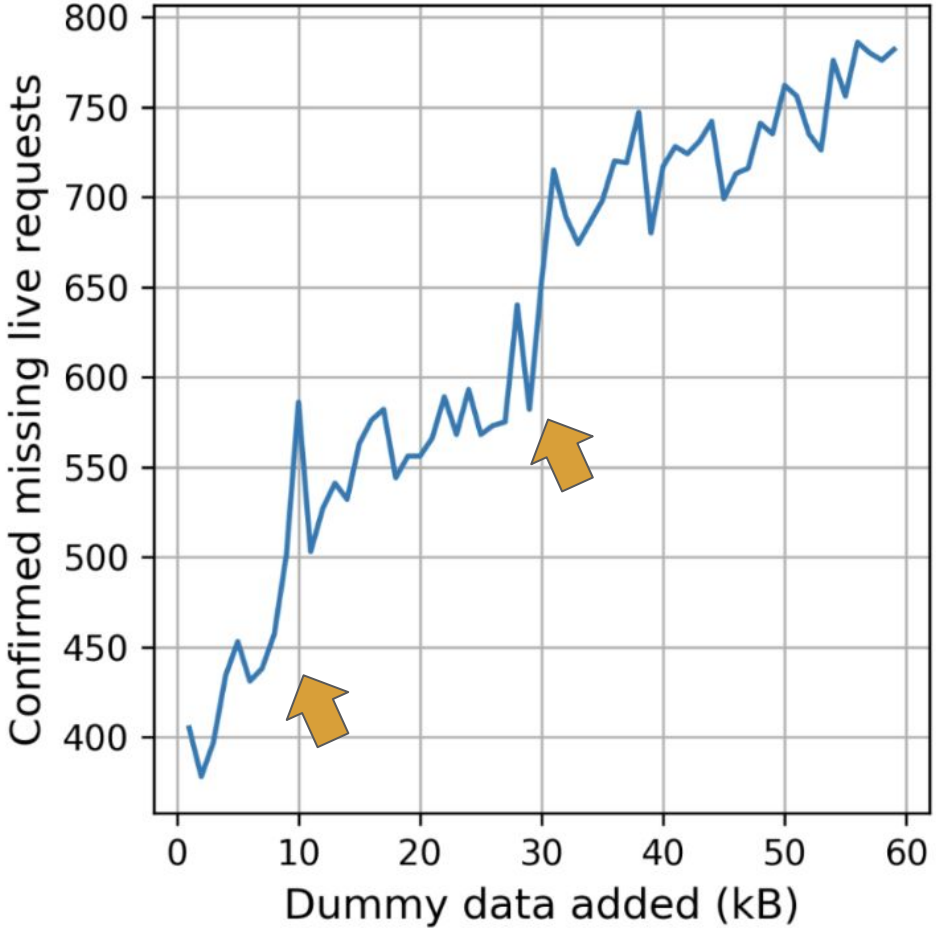


blog.cloudflare.com/sizing-up-post-quantum-signatures

And, of course…

Middleboxes

Bump in missing requests suggests some middleboxes do not like certificate chains longer than 10kB and 30kB.



Not great, not terrible

It probably won’t break the Web, but the performance impact could well delay adoption.

NIST signature on-ramp

NIST took notice and is calling for new signature schemes to be submitted. We're hoping for two different schemes:

- Unbalanced. Scheme that has small signatures ~100 bytes, fast verification, paying with a larger public key 10–100kB and slow signing. (eg. UOV, MAYO)
- Balanced. Scheme that optimises signature + public key size with fast signing and verification. (eg. HAWK, MAYO)

In the meantime

There are small and larger changes to the possible to the protocols to reduce the number of signatures.

- Leave out intermediate certificates.
- Use KEM instead of handshake signature.
- There seems to be momentum to overhaul the Web PKI in more fundamental ways.

Unclear whether any of these will make it.

Signatures 

Less urgent, but path is unclear. Real risk we will start migrating too late.

That’s not all: the Internet isn’t just TLS

There is much more cryptography out there with their own unique challenges.

- DNSSEC with its harder size constraints
- Research into post-quantum privacy enhancing techniques, eg. anonymous credentials, is in the early stages.

Thank you, questions?

References

- Follow along on the IETF PQC e-maillist
- Check out our research team and Cloudflare's blog, eg.:
  - 2019 TLS experiment with Google
  - Sizing-up Post-Quantum Signatures
  - Deploying Kyber worldwide
- Reach out: ask-research@cloudflare.com

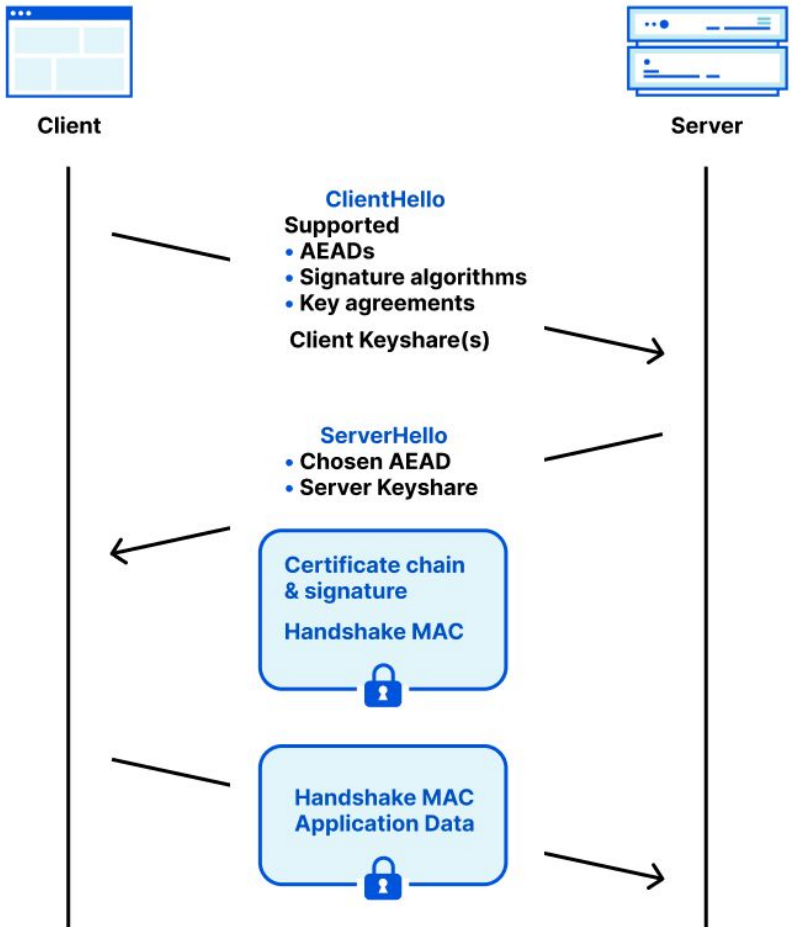
Backup slides

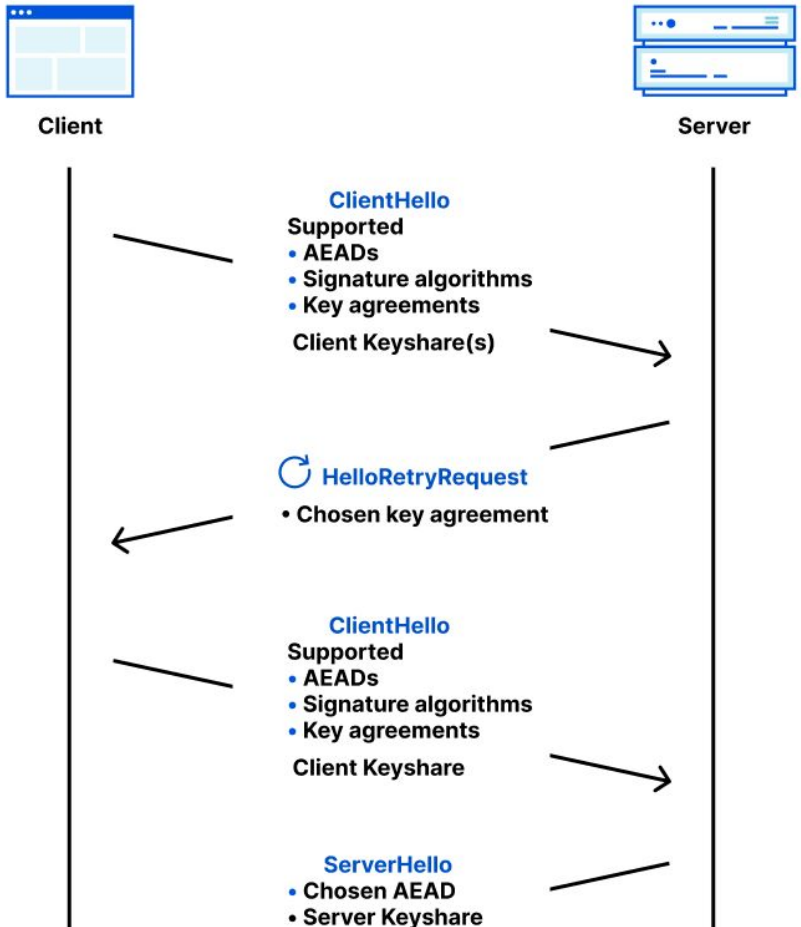
```c
static inline int64_t
fpr_rint(fpr x)
{
	/*
	 * We do not want to use llrint() since it might be not
	 * constant-time.
	 *
	 * Suppose that x >= 0. If x >= 2^52, then it is already an
	 * integer. Otherwise, if x < 2^52, then computing x+2^52 will
	 * yield a value that will be rounded to the nearest integer
	 * with exactly the right rules (round-to-nearest-even).
	 *
	 * In order to have constant-time processing, we must do the
	 * computation for both x >= 0 and x < 0 cases, and use a
	 * cast to an integer to access the sign and select the proper
	 * value. Such casts also allow us to find out if |x| < 2^52.
	 */
	int64_t sx, tx, rp, rn, m;
	uint32_t ub;

	sx = (int64_t)(x.v - 1.0);
	tx = (int64_t)x.v;
	rp = (int64_t)(x.v + 4503599627370496.0) - 4503599627370496;
	rn = (int64_t)(x.v - 4503599627370496.0) + 4503599627370496;

	/*
	 * If tx >= 2^52 or tx < -2^52, then result is tx.
	 * Otherwise, if sx >= 0, then result is rp.
	 * Otherwise, result is rn. We use the fact that when x is
	 * close to 0 (|x| <= 0.25) then both rp and rn are correct;
	 * and if x is not close to 0, then trunc(x-1.0) yields the
	 * appropriate sign.
	 */

	/*
	 * Clamp rp to zero if tx < 0.
	 * Clamp rn to zero if tx >= 0.
	 */
	m = sx >> 63;
	rn &= m;
	rp &= ~m;

	/*
	 * Get the 12 upper bits of tx; if they are not all zeros or
	 * all ones, then tx >= 2^52 or tx < -2^52, and we clamp both
	 * rp and rn to zero. Otherwise, we clamp tx to zero.
	 */
	ub = (uint32_t)((uint64_t)tx >> 52);
	m = -(int64_t)((((ub + 1) & 0xFFF) - 2) >> 31);
	rp &= m;
	rn &= m;
	tx &= ~m;

	/*
	 * Only one of tx, rn or rp (at most) can be non-zero at this
	 * point.
	 */
	return tx | rn | rp;
}
```

This function from Falcon as submitted to round 3 is not constant-time on ARMv7 as claimed.

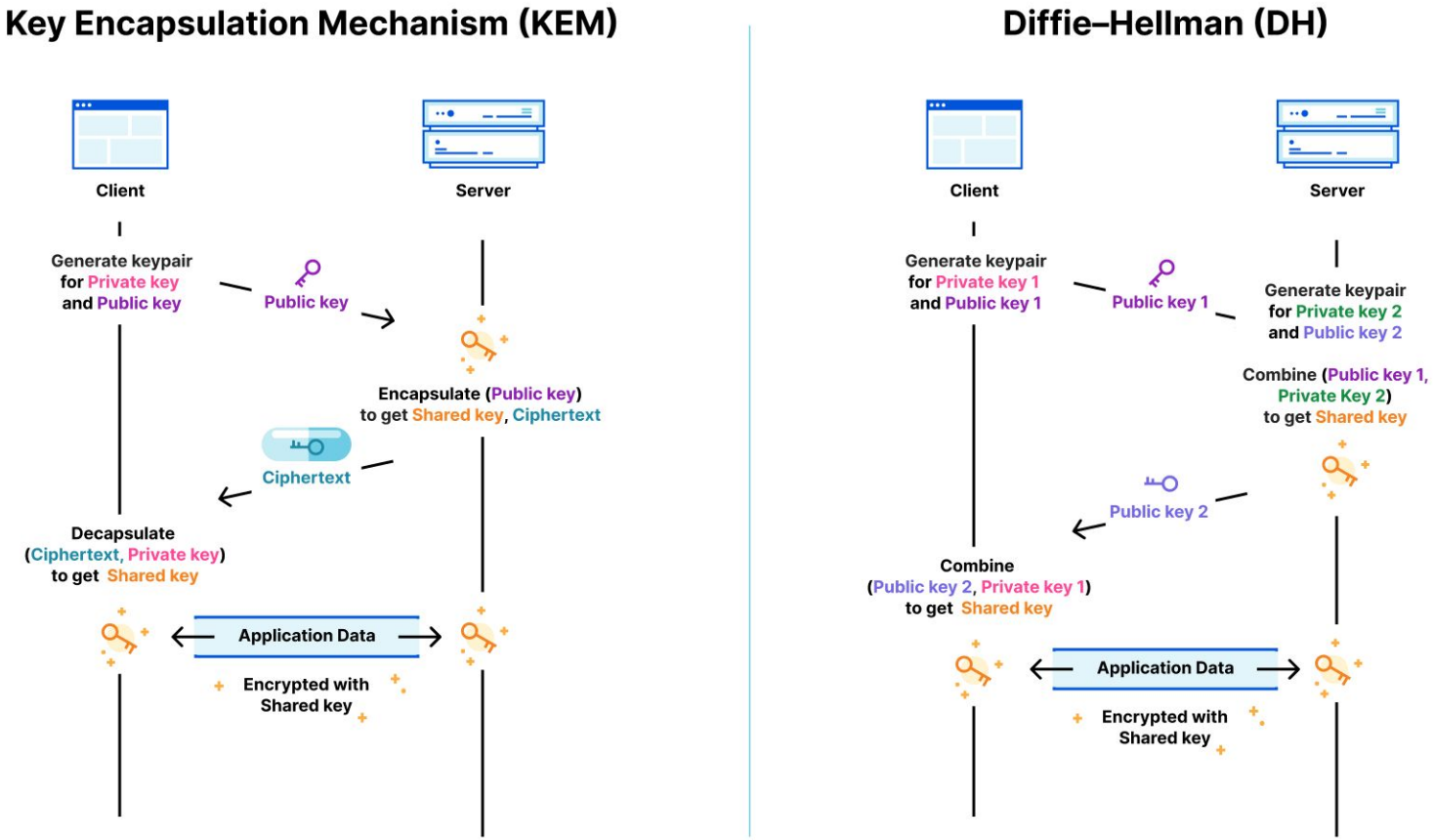
Can you spot the error?

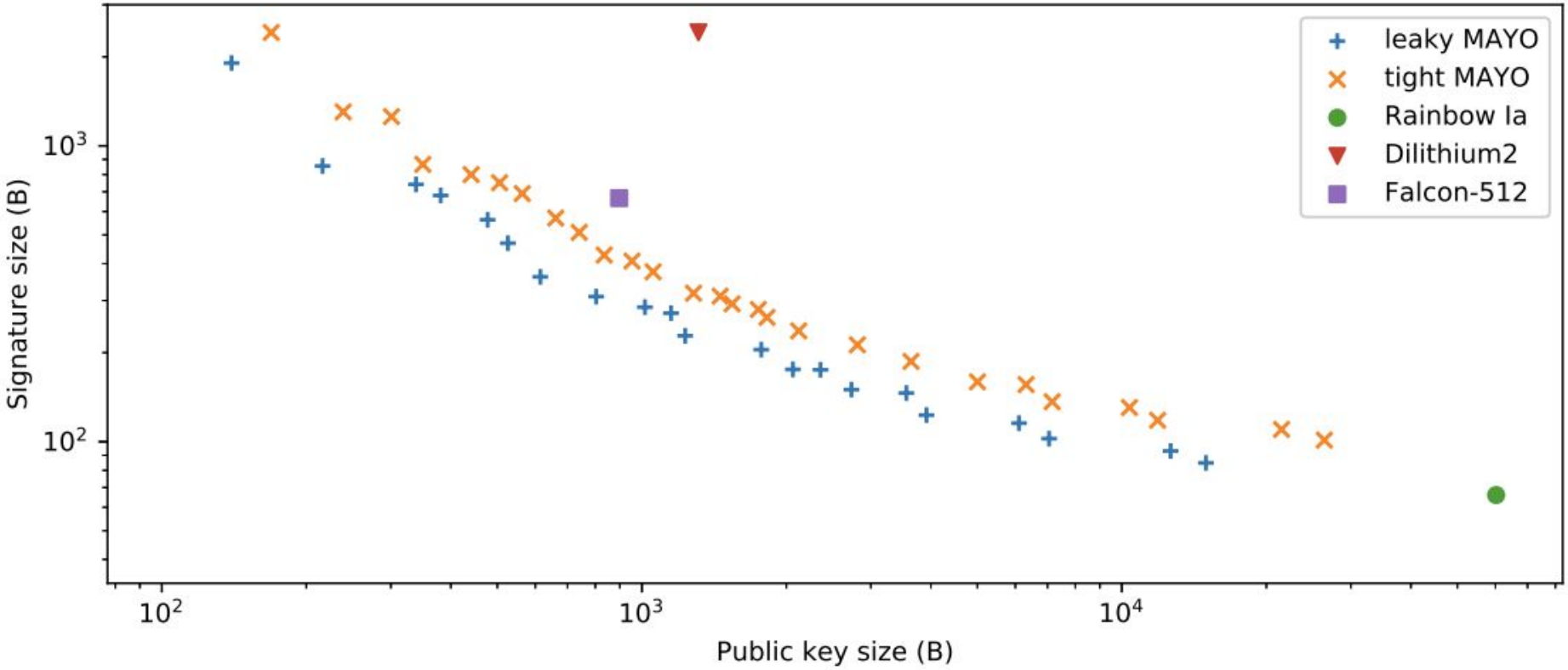
TLS 1.3 handshake





KEM versus Diffie–Hellman




leaky MAYO
tight MAYO
Rainbow Ia
Dilithium2
Falcon-512
10^3
10^2
Signature size (B)
10^2
10^3
10^4
Public key size (B)