More efficient KEMTLS with pre-distributed keys

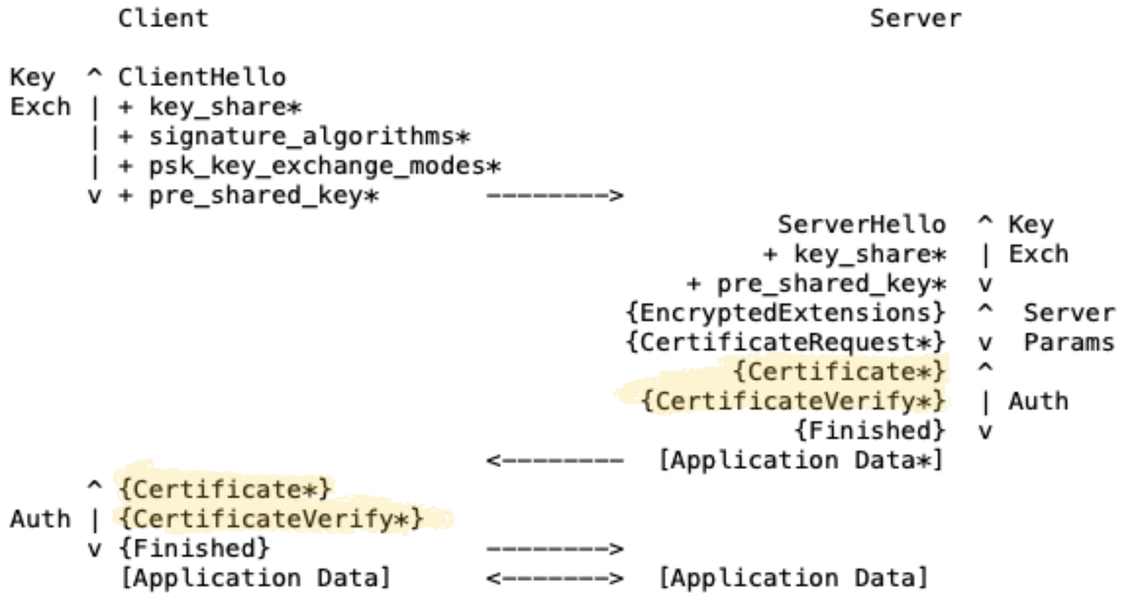
ESORICS 2021

Peter Schwabe, Douglas Stebila and Thom Wiggers

TLS 1.3

RFC 8446

- **Signed** Key Exchange
  - Ephemeral (EC)DH key exchange
  - RSA or EC signatures



The Quantums

(Everyone already knows this by now, right?)

- Shor's algorithm (1994)
  - RSA, Elliptic curves completely broken by quantum computers
- We need post-quantum cryptography
  - NIST is running a PQC standardization project for **Key Exchange Mechanisms (KEMs)** and **Signature algorithms**
- We need to to think about how move to post-quantum algorithms now: standardizing e.g. TLS or certificates takes *years*.

Most PQ Signature algorithms are big and/or slow and/or require hw support

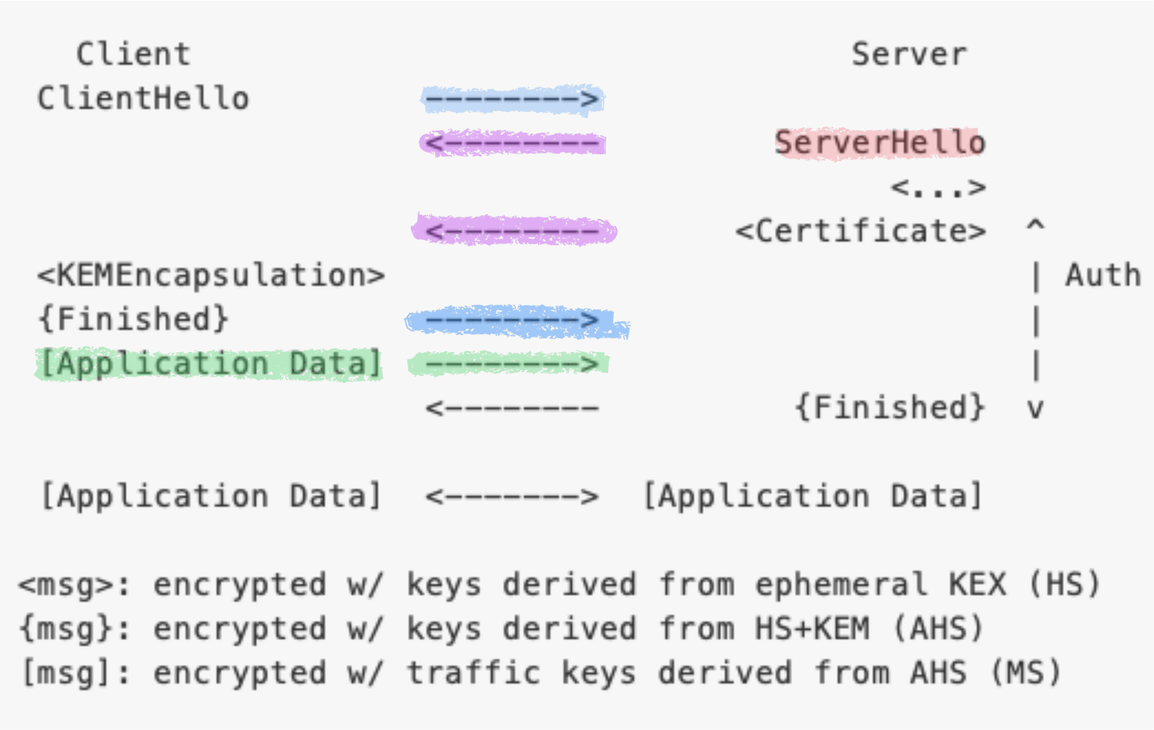
Use KEMs for authentication instead

KEMTLS: ACM CCS 2020

KEMTLS

Us, ACM CCS 2020

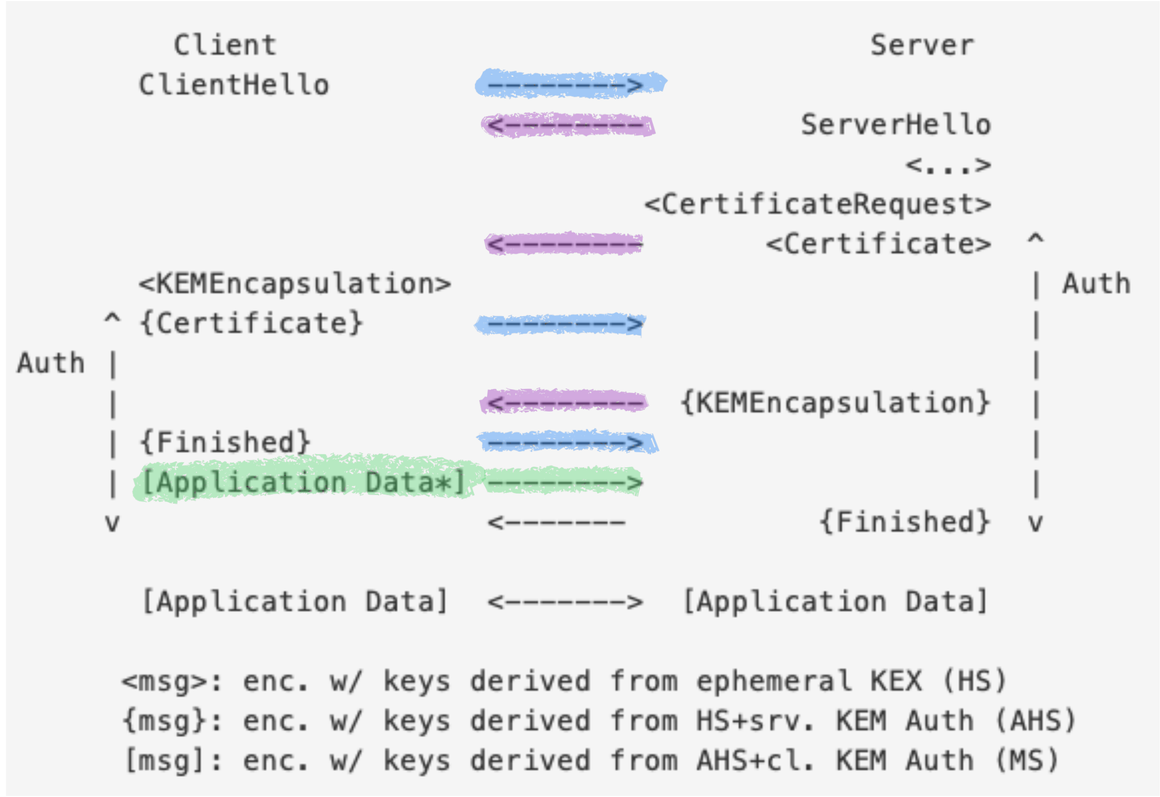
- Use KEMs for handshake authentication
- Avoid extra round-trip by allowing client to send data first, instead of server
- First round-trip is *implicitly authenticated*



KEMTLS

What we left on the table

- Client authentication is terrible (full extra RTT)



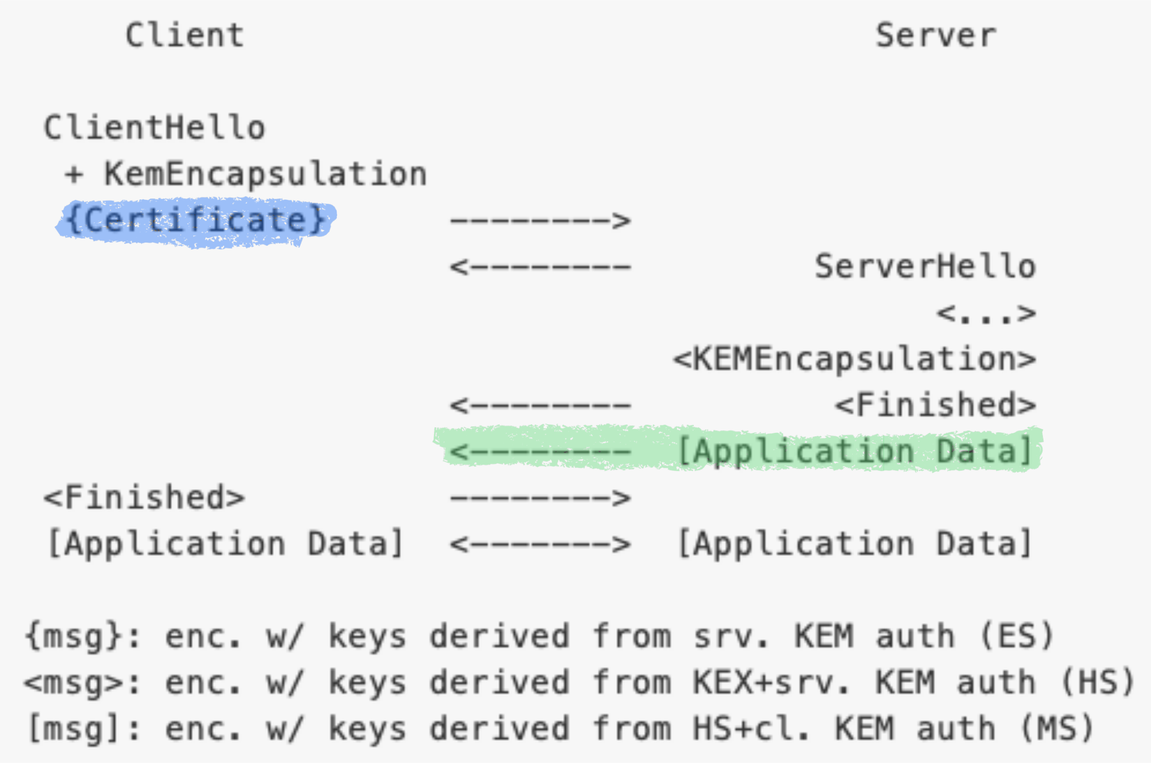
- *What if the client already has the server's long-term public key?*
  - e.g. through caching or pre-installed

KEMTLS-PDK

KEMTLS-PDK

Pre-Distributed Keys

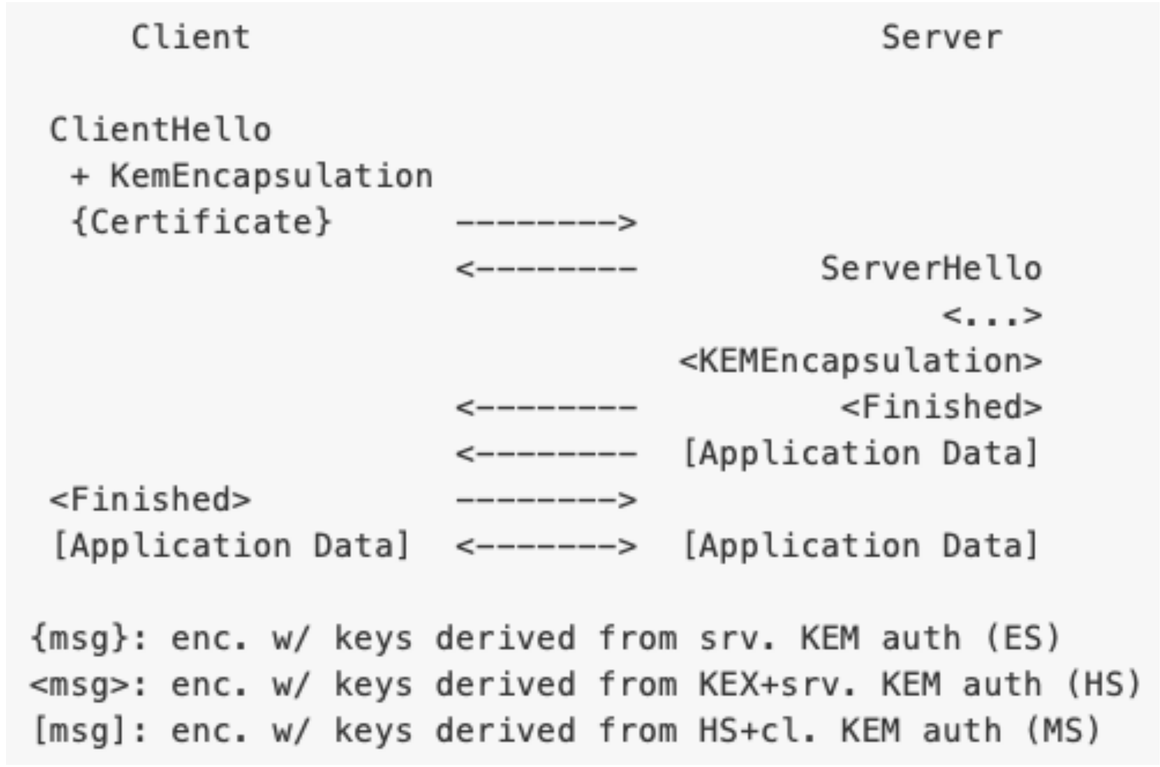
- Resumption-style mechanism that avoids the downsides of symmetric-key TLS PSK
- Given server's long-term key, client can send ciphertext in ClientHello
- Also allow to send client certificate in ClientHello
- Get a **1-RTT**, TLS 1.3-shape handshake *without implicit authentication*



KEMTLS-PDK

Pre-Distributed Keys

- Certificate message can’t be forward-secure and possibly can be replayed
- Potential impact seems lower than similar 0-RTT data concerns
- We use PSK/0-RTT “EarlySecret” keys from TLS 1.3 key schedule to also communicate this similarity.



More Efficient KEMTLS with Pre-Distributed Keys

Peter Schwabe, Douglas Stebila and Thom Wiggers

- https://ia.cr/2021/779
- If you already have the server's public key, you can do smart things
  - Save loads of bytes on the wire compared to KEMTLS
  - Achieve full explicit authentication in 1RTT
  - Also do client authentication in 1RTT if the client knows to authenticate
- We discuss the implementation and the analysis in the extended presentation.
- Security proof in the full online version

Analysis

Implementation

- Extended original implementation of KEMTLS in Rustls
  - Actually, a new, much improved implementation of KEMTLS in Rustls
- Fast (AVX2-accellerated) post-quantum implementation via Open-Quantum-Safe's liboqs
- We hacked Rainbow’s AVX2 code into liboqs for fair comparision

- All software available via https://wggrs.nl/p/kemtlspdk/

Table 1: Summary of performance characteristics of KEMTLS, signed-KEM TLS 1.3 with cached server certificate, and KEMTLS-PDK

	KEMTLS	Cached TLS	KEMTLS-PDK
Unilaterally authenticated			
Round trips until client receives response data	3	3	3
Size (bytes) of public key crypto objects transmitted:			
• Minimum PQ	932	499	561
• Module-LWE/Module-SIS (Kyber, Dilithium)	5,556	3,988	2,336
• NTRU-based (NTRU, Falcon)	3,486	2,088	2,144
Mutually authenticated			
Round trips until client receives response data	4	3	3
Size (bytes) of public key crypto objects transmitted:			
• Minimum PQ	1,431	2,152	1,060
• MLWE/MSIS	9,554	10,140	6,324
• NTRU	5,574	4,365	4,185

Unilaterally authenticated		31.1 ms RTT, 1000 Mbps		
		Client sent req.	Client recv. resp.	Server expl. auth.
KEMTLS	Minimum	75.2	115.9	115.9
	MLWE/MSIS	63.2	**94.7**	94.6
	NTRU + Falcon	63.1	**94.7**	94.6
Cached TLS	TLS 1.3 X25519 + RSA	66.3	**97.5**	66.2
	Minimum	70.0	**101.2**	69.9
	MLWE/MSIS	63.9	**95.1**	63.8
	NTRU + Falcon	64.8	**96.1**	64.7
PDK	Minimum	66.1	**97.3**	66.1
	Kyber	63.0	**94.3**	63.0
	NTRU	63.0	**94.3**	62.9
	SABER	63.1	**94.3**	63.0

SIKE + Rainbow

Kyber/Dilithium

SIKE + McEliece (eph.) (auth)

Still requires Signatures in certificates

Only HS sigs, no certificates

1.5 KEM on the wire (2× ct, 1× pk)

Mutually authenticated		31.1 ms RTT, 1000 Mbps		
		Client sent req.	Client recv. resp.	Server expl. auth.
KEMTLS	Minimum	129.6	160.8	122.9
	MLWE/MSIS	95.1	**126.5**	95.0
	NTRU	95.0	**126.4**	94.8
Cached TLS	TLS 1.3	68.6	100.1	66.0
	Minimum	71.0	**102.6**	69.9
	MLWE/MSIS	64.4	**96.1**	63.8
	NTRU	66.1	**98.0**	64.7
PDK	Minimum	84.9	116.0	84.9
	Kyber	63.5	**94.8**	63.4
	NTRU	63.6	**94.9**	63.5
	SABER	63.6	**94.9**	63.5

More Efficient KEMTLS with Pre-Distributed Keys

Peter Schwabe, Douglas Stebila and Thom Wiggers

- Paper with full security proof: https://ia.cr/2021/779
- Implementation: https://wggrs.nl/p/kemtlspdk/

- Thanks for watching