Migrating Protocols to the Post-Quantum Setting

The Case of WireGuard

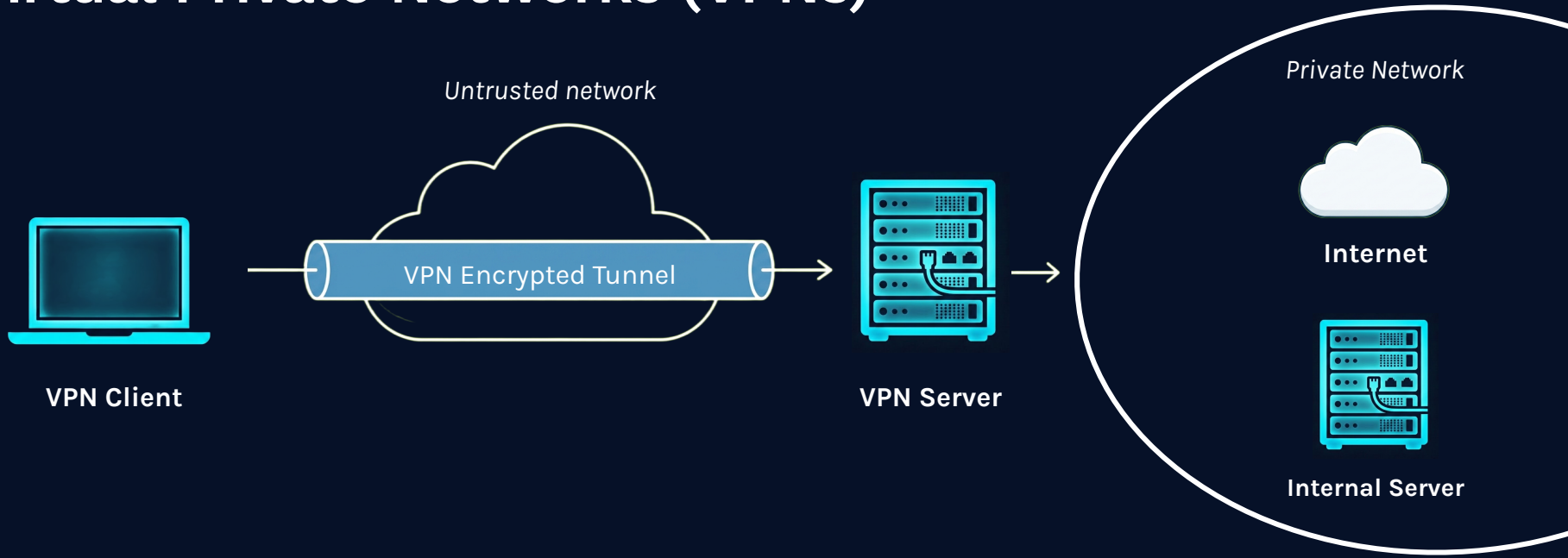
Keitaro Hashimoto[1], Shuichi Katsumata[1,2], **Guilhem Niot**[2], Thom Wiggers[2]
[1]AIST, [2]PQShield

Based on our S&P' 26 paper

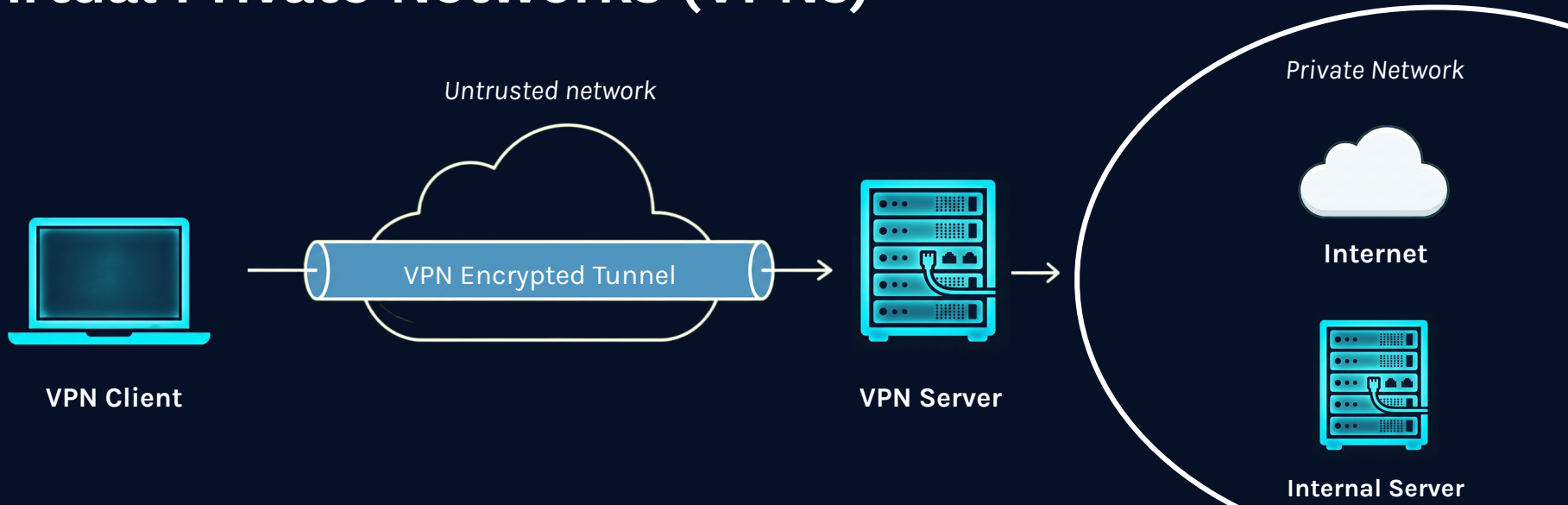
Roadmap

1. Introduction to VPNs and WireGuard
2. Going Post-Quantum: Challenges and The Size Blocker
3. Reinforced KEM
4. Beyond: Formal Verification and Uncovered UKS attack

Virtual Private Networks (VPNs)



Virtual Private Networks (VPNs)



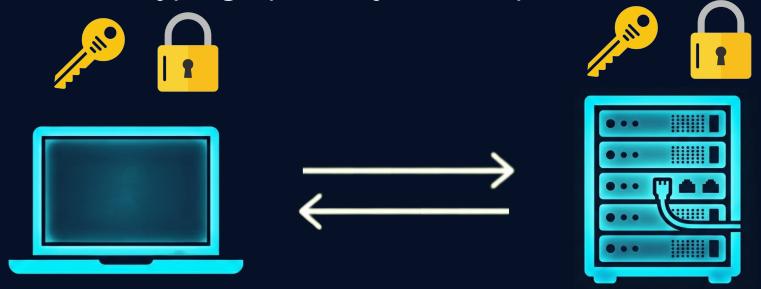
Confidentiality: Data in transit are encrypted so that they are unreadable by the network.

Authentication: Only authorized clients can connect to the server and access the private network.

Virtual Private Networks (VPNs)

Phase 1: Setup

The client and server execute a protocol using their static cryptographic keys to setup a channel.



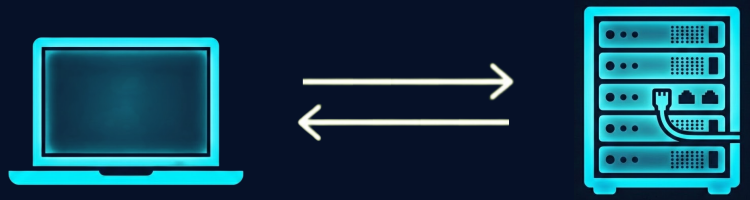
Outputs a shared symmetric key.

*****

Phase 2: Communication

Uses cheaper symmetric cryptography to encrypt traffic with shared symmetric key.

*****



WireGuard: The Why's

- VPN proposed by Donenfeld [NDSS:Don17], based on Diffie-Hellman (DH) Key Exchange.
- "Cryptographically opinionated": no negotiation mechanism, only a small set of modern cryptographic primitives.
  - Allows for very fast single-round setup.
  - Unique security features
- Integrated in the Linux kernel.

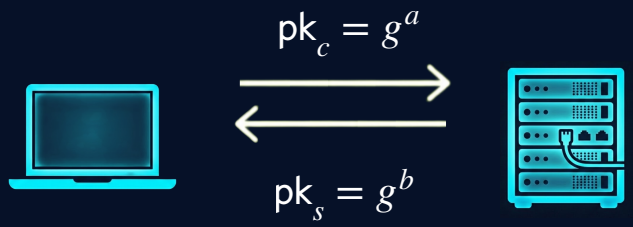
Post-Quantum WireGuard?

- **Phase 1:** Single-round setup heavily relies on classical Diffie-Hellman, broken by a quantum computer.
  Only fallback is if one uses an optional pre-shared key (PSK) per peer, mixed into the key schedule, but limited security. We should rather rely on public-key cryptography.
- **Phase 2:** already quantum safe (symmetric cryptography)!

WireGuard: Single Round-Trip Setup

Diffie-Hellman Key Agreement

Samples a.



Samples b.

Both can compute g^{ab}.

WireGuard: Single Round-Trip Setup

Assume prior exchange of static keys between client and server.

msg1: Client to Server

pk_e

msg2: Server to Client

pk'_e

Ephemeral X25519 keys are sampled by each party and the resulting Diffie-Hellman secrets are mixed into the final key to enhance security.

Key Indistinguishability: Output key is pseudo-random even if static keys leak.

WireGuard: Single Round-Trip Setup

Assume prior exchange of static keys between client and server.

msg1: Client to Server

pk_e	time

AEAD(k_2, transcript || now): confirms knowledge of secret k_2 (derived from client's static and ephemeral secrets, and server's static key), for authentication + prevent replay attacks.

msg2: Server to Client

pk'_e	empty

AEAD(k_3, transcript): confirms knowledge of k_3 (derived from all client's and server's static and ephemeral secrets).

Key Indistinguishability: Output key is pseudo-random even if static keys leak.

Entity Authentication: No party can impersonate another entity, and message origin is verified.

WireGuard: Single Round-Trip Setup

Assume prior exchange of static keys between client and server.

msg1: Client to Server

pk_e	time	static

AEAD(k_1, static pk client): Sends client's public key using k_1, derived from ephemeral key and server's static key. Can only be decrypted by server.

msg2: Server to Client

pk'_e	empty

Key Indistinguishability: Output key is pseudo-random even if static keys leak.

Entity Authentication: No party can impersonate another entity, and message origin is verified.

Identity Hiding: Transcript alone does not reveal the identity of the communicating parties

WireGuard: Single Round-Trip Setup

Assume prior exchange of static keys between client and server.

msg1: Client to Server

pk_e	time	static	m1 | m2

Add MACs of the exchange transcript + server's static key. Avoids performing costly asymmetric crypto operations if the client does not know the server (e.g. internet-wide scanner). Additional challenge mechanism in m2 if heavy load.

msg2: Server to Client

pk'_e	empty	m1 | m2

Key Indistinguishability: Output key is pseudo-random even if static keys leak.

Entity Authentication: No party can impersonate another entity, and message origin is verified.

Identity Hiding: Transcript alone does not reveal the identity of the communicating parties

DoS mitigation: Drop illegitimate messages early

Two Key Chains

WireGuard maintains two chains to derive keys from shared secrets and from the transcript.

Key Chain

The key chain absorbs each shared Diffie-Hellman secret as it becomes available.

The AEAD keys are derived from this chain and ensure knowledge of the original X25519 secret keys. The final session secret also.

Hash Chain

The hash chain absorbs elements separately.

Ensures that the two parties agree on their views.

One must select carefully what element is absorbed in the chains, and in which order to provide all the WireGuard security properties.

Constraints To Preserve An Attractive Protocol

One UDP packet: stateless protocol

Two messages that must fit in one UDP packet: no fragmentation, no state management, and DoS protection.

Kernel code

Code should fit in the kernel for max performance. So, few dynamic allocs, and small stack.

DoS protection

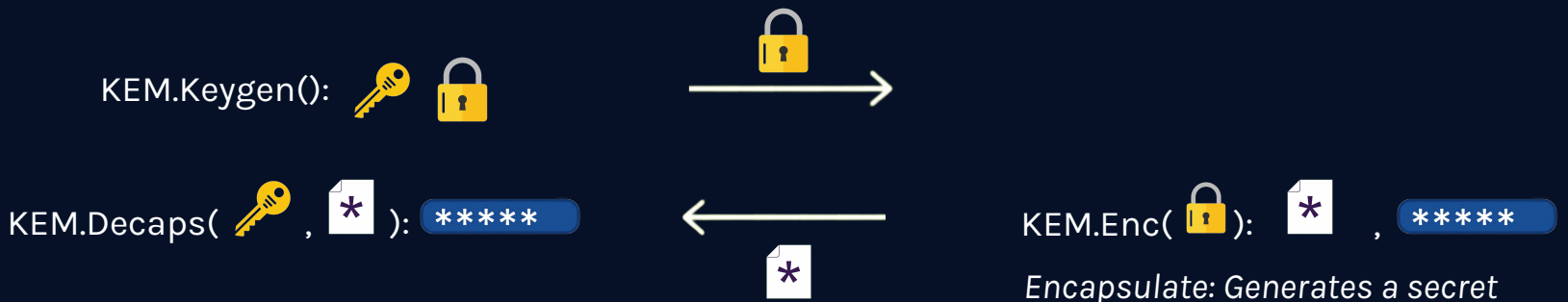
MACs m1, m2 verifiable with no heavy public-key crypto. msg1 authenticated.

Post-Quantum WireGuard?

- We don’t have Diffie-Hellman from post-quantum assumptions.
- Let’s replace DH with KEMs! Hülsing et al. (https://eprint.iacr.org/2020/379)

Post-Quantum WireGuard?

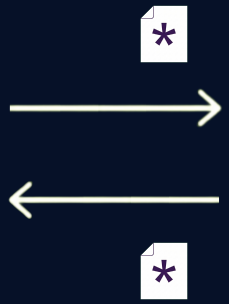
- We don’t have Diffie-Hellman from post-quantum assumptions.
- Let’s replace DH with KEMs! Hülsing et al. (https://eprint.iacr.org/2020/379)



Post-Quantum WireGuard?

- Let’s replace DH with KEMs! Hülsing et al. (https://eprint.iacr.org/2020/379)

Encapsulate to Server’s static key.



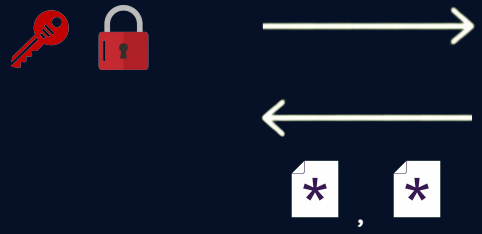
Encapsulate to Client’s static key.

Post-Quantum WireGuard?

- Let's replace DH with KEMs! Hülsing et al. (https://eprint.iacr.org/2020/379)

Encapsulate to Server's static key.

+ Generate ephemeral KEM key



Encapsulate to Client's static key.

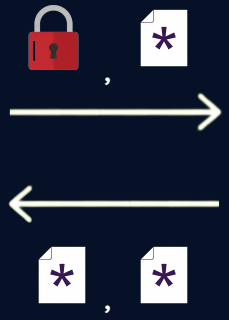
+ Encapsulate to Client's ephemeral key.

Post-Quantum WireGuard?

- **Let's replace DH with KEMs! Hülsing et al. (https://eprint.iacr.org/2020/379)**

Encapsulate to Server's static key.

+ Generate ephemeral KEM key



Encapsulate to Client's static key.

+ Encapsulate to Client's ephemeral key.

But KEM has asymmetric security properties, msg1 does not authenticate the client. Recovered by using pre-shared key (PSK) early in the key chain to guarantee connected client is legit.

Post-Quantum WireGuard: The Key and Hash Chains?

The key and hash chains must be updated to safely absorb the KEM shared secrets.

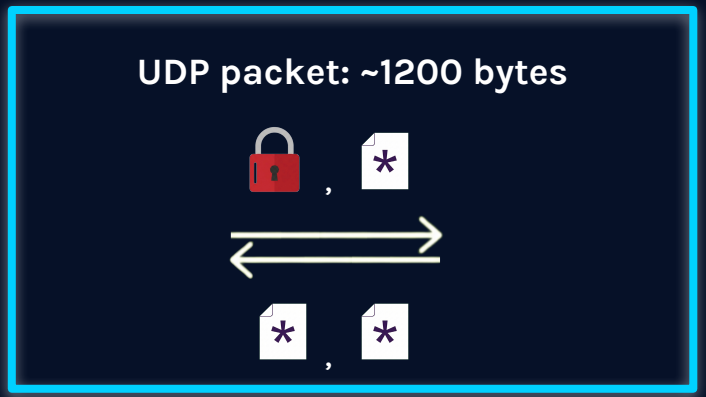
In Hülsing et al.'s work, the key chain only absorbs the client's ephemeral key and the ciphertext it sends to the server. Its long term key and the server's ciphertext to the client are never absorbed.

→ **We will see that this weakens the protocol's security.**

Post-Quantum WireGuard: The Load of PQ Primitives

To offer its nice properties (simplicity, efficiency, DoS protection), WireGuard builds on UDP.

Ensures no fragmentation. **The standardised PQ KEM does not fit!**



ML-KEM-512:

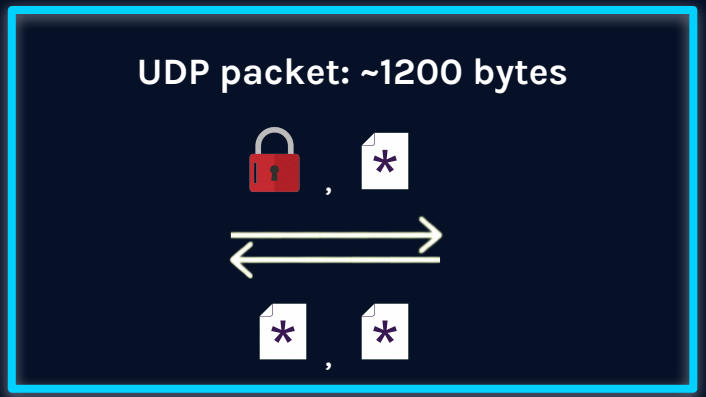
= 800 bytes

= 768 bytes

Post-Quantum WireGuard: The Load of PQ Primitives

To offer its nice properties (simplicity, efficiency, DoS protection), WireGuard builds on UDP.

Ensures no fragmentation. **The standardised PQ KEM does not fit!**



Hülsing et al. suggests to use McEliece (alternative PQ KEM) for long-term KEM

= 96 bytes.

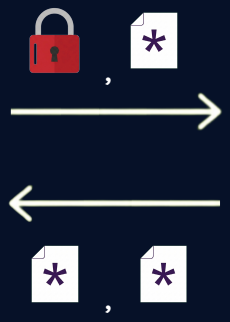
\+ **Dagger (a Saber variant) for ephemeral KEM (~900 bytes keys and ciphertexts)**

… but the load is paid somewhere else

- McEliece has huge 520 kB public keys  . → With many clients, Server's memory would quickly be exhausted.

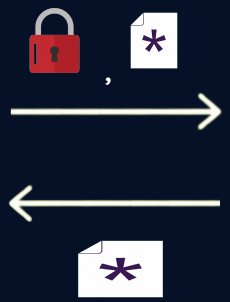
  Exacerbated by implementing WireGuard in the Linux kernel (constrained memory allocation).

Our Proposal: Reinforced KEM



- Use ML-KEM style KEM.
- Encapsulate to both client's static and ephemeral keys at the same time to reduce the size of the second message.
- Need both keys to recover the shared secret.

Our Proposal: Reinforced KEM



- Use ML-KEM style KEM.
- Encapsulate to both client's static and ephemeral keys at the same time to reduce the size of the second message.
- Need both keys to recover the shared secret.

Our Proposal: Reinforced KEM

(simplified)

The Standard ML-KEM:

KEM.Enc(): 

KEM.Enc(): 

Reinforced KEM (ours):

RKEM.Enc( , ): 

Reinforced KEM: Sizes and Cycles

	ML-KEM (x2)	Reinforced KEM	Delta
Long-term key	800B	1376B	+ 72 %
Ephemeral key	800B	864B	+ 8 %
Ciphertext for both keys	1536B	1088B	- 29 %

Now combined ciphertext fits nicely within one UDP packet.

Results

	Classic WireGuard	PQWG (Hülsing et al.)	**PQWG (ours)**
Client KEM	X25519	McEliece 460896	Rebar
Server KEM	X25519	McEliece 460896	McEliece 460896
Ephemeral KEM	X25519	Dagger	Rebar
Per-peer storage at server	**32B**	**524 160B**	**1376B**
msg1	64B	1084B	1052B
msg2	64B	1148B	1088B
Latency	**33.6ms**	**36.3ms**	**36.8ms**

Moving to Rebar saves 381x per peer storage on the server. Makes the difference for a kernel deployment.

Are we done?

Formal analysis of PQ WireGuard?

	Key indistinguishability	Entity authentication	Identity hiding	DoS mitigation
Hülsing et al. [SP:HNSWZ21]	Computational/ Symbolic	Symbolic	Symbolic	Symbolic
Ours	**Computational**	**Computational**	**Computational**	**Computational**

Hülsing et al. formalized the protocol properties in an idealized symbolic model (Tamarin). And only key indistinguishability in a computational model.

→ Computer verified but can miss some attack vectors due to the concrete KEM instantiation.

Are we done?

Formal analysis of PQ WireGuard?

Entity authentication was not fully covered by prior computational models.

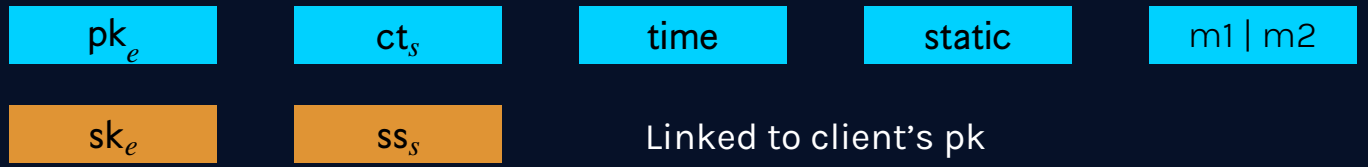
→ The ad hoc key chain of Hülsing et al. not including the client's static key and the ciphertext it receives opens the path to UKS (Unknown Key Share) attacks: it may be possible to convince a server that they established a session with a different client than the actual one.

[SP:HNSWZ21] requires additional (non-standard) KEM binding properties for this to hold.

An unknown key share attack on PQ WireGuard

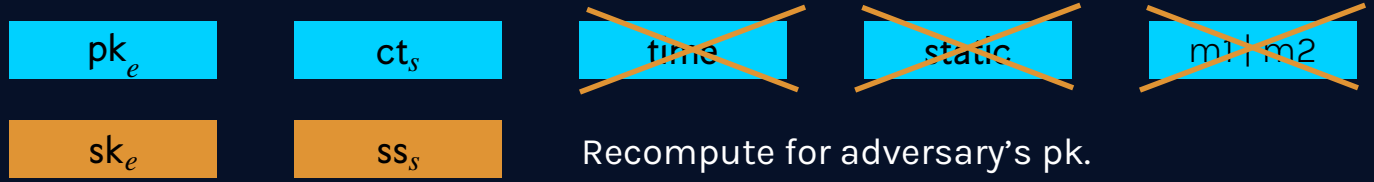
1

Attacker registers his own keys on the server, intercepts the client's msg1, and the PSK + client's state leak (notably the secret it sent to the server).



An unknown key share attack on PQ WireGuard

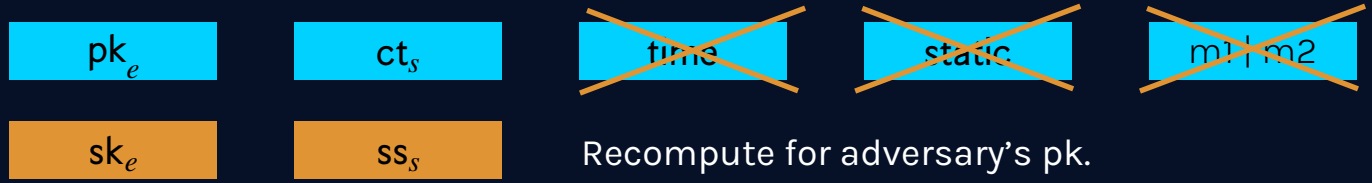
1. Attacker registers his own keys on the server, intercepts the client's msg1, and the PSK + client's state leak (notably the secret it sent to the server).



2. The adversary replaces static and time to communicate its own pk. Possible thanks to knowledge of sk_A and ss_s. Crucially, the server will extract the same shared secret ss_s.

An unknown key share attack on PQ WireGuard

1. Attacker registers his own keys on the server, intercepts the client's msg1, and the PSK + client's state leak (notably the secret it sent to the server).

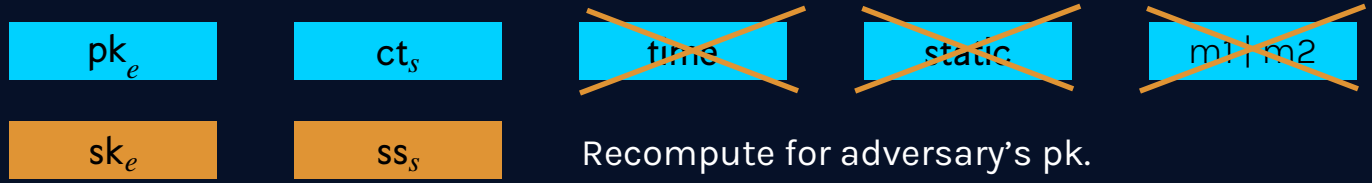


2. The adversary replaces **static** and **time** to communicate its own pk. Possible thanks to knowledge of sk_A and ss_s. Crucially, the server will extract the same shared secret ss_s.

3. Intercept msg2, with ciphertext for attacker.

An unknown key share attack on PQ WireGuard

1. Attacker registers his own keys on the server, intercepts the client's msg1, and the PSK + client's state leak (notably the secret it sent to the server).



2. The adversary replaces **static** and **time** to communicate its own pk. Possible thanks to knowledge of sk_A and ss_s. Crucially, the server will extract the same shared secret ss_s.

3. Intercept msg2, with ciphertext for attacker. Re-encapsulate same secret ss_A to client.

Since ct_c is not included in the keychain, the key derived by the client and the server is the same.

But server thinks it is talking to the attacker.

The Fix: Absorb Every PK and Shared Secret

We modify PQ WireGuard's key chain to absorb all client's and server's public keys.
This prevents two sessions with a different view to derive the same final key.

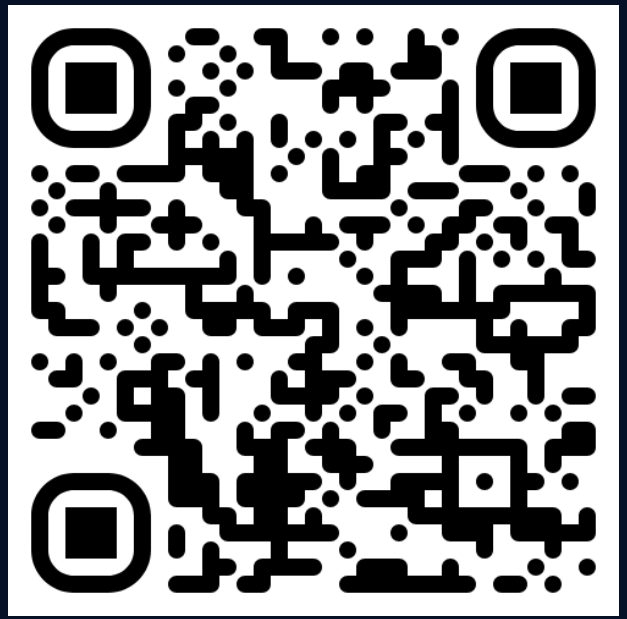
Avoids reliance on non-standard KEM binding properties.

Some Takeaways

- We do not have a direct PQ analog of Diffie Hellman (DH): Protocols need to be modified in depth, and it can be difficult to achieve the same security properties. C.f. the UKS attack discovered in Hülsing et al's ad hoc key and hash chains.

- PQ primitives are significantly bigger, going over typical network bounds. Here, we designed a specialized primitive to compress two PQ elements together and manage to stay within UDP size. Performance remains competitive although our implementation is not optimized.

Questions?



Revisiting PQ WireGuard: A Comprehensive Security Analysis With a New Design Using Reinforced KEMs

Keitaro Hashimoto[1], Shuichi Katsumata[1,2], **Guilhem Niot**[2], Thom Wiggers[2]

[1]AIST, [2]PQShield