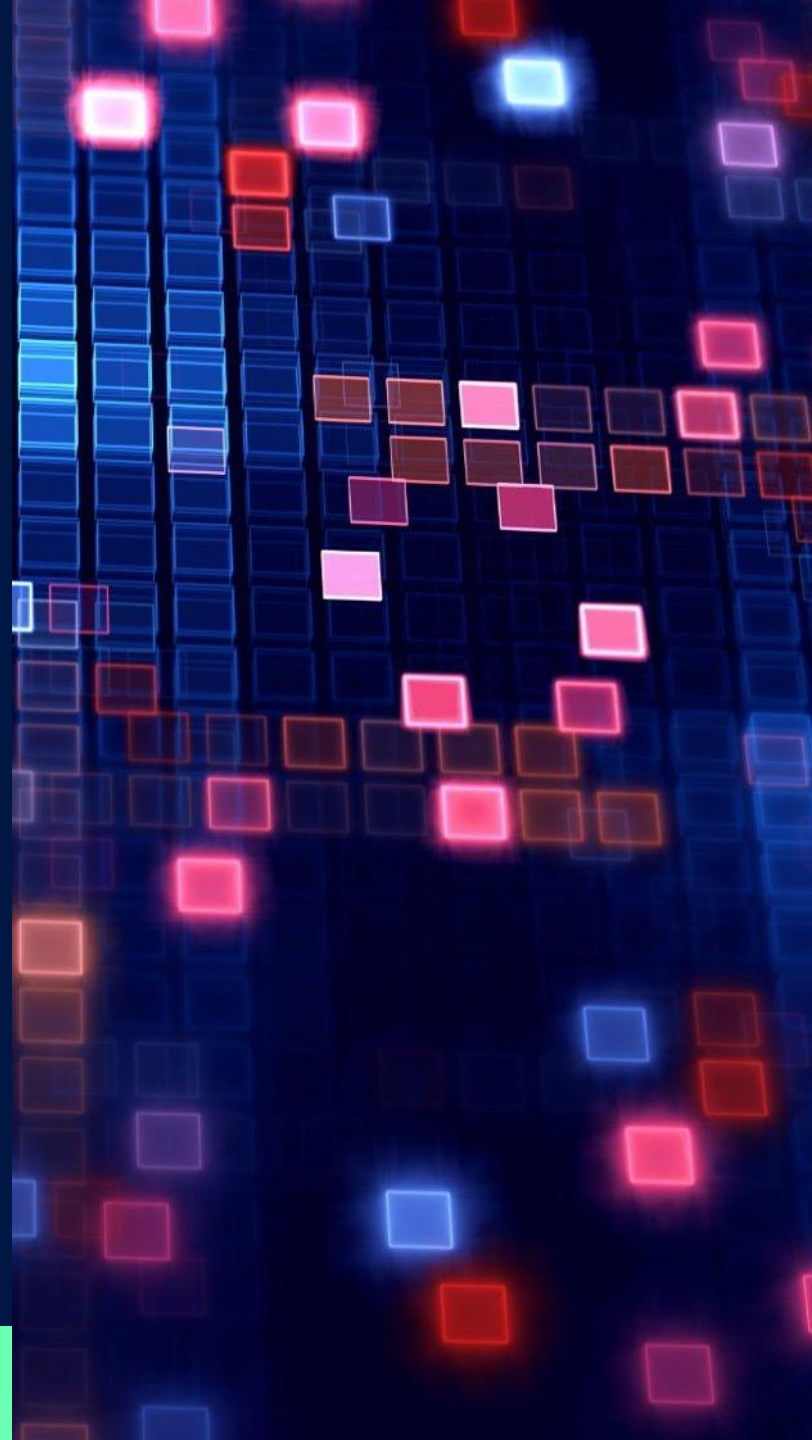
zscaler™

Securing the Long Haul

Challenges with Long-Lived TLS 1.3 and QUIC Sessions

October 10, 2025



Speaker

Yaroslav Rosomakho

Chief Scientist

- Leading strategic research at Zscaler
- Previously held leadership positions in Netskope, Forcepoint and Abor Networks
- Active contributor to TLS, QUIC, HTTP and many other IETF working groups
- Chairing SEAL and HPKE working groups
- Hands-on experience with large scale enterprise networks, service providers and cloud services

What is the “long haul”?

Traditional SSL/TLS connections are short-lived

- Historically TLS was about “wrapping” TCP socket
- Most use TLS for relatively short web transactions
- But communications is not only about web browsing. Some examples of long-lasting TLS connections include
  - VPNs and ZTNAs
  - Carrier signalling
  - Industrial IoT communication
- Reconnection can be disruptive

VPNs and ZTNAs

- Originally VPNs used IPSec exclusively
- Over the years TLS started to gain traction…
- Nowadays TLS and DTLS are often used not only for remote access but also for site-to-site VPNs
- MASQUE (aka encapsulate all things into HTTP-over-TLS or HTTP-over-QUIC) is growing
- Most ZTNAs use TLS

Connections can last for days/weeks and transfer large volume of data (Terabytes)

Carrier signalling

- Traditionally carrier signalling relied on SS7, designed decades ago with little focus on security
- Transition to IP-based signalling (SIP, Diameter, GTP, SCTP) introduced new attack surfaces
- Increasing adoption of TLS and DTLS for signalling traffic between carriers and core network elements
- 5G/6G Core architecture drives even heavier reliance on TLS/DTLS/QUIC
- Interconnect signalling can persist across long-lived channels, spanning hours or days

Connections can last for weeks/months and carry very sensitive information

IoT communication

- Industrial IoT is mission-critical
- Telemetry streaming, control channel, firmware updates and maintenance
- Could be constrained:
  - Slow CPUs (think Javacard…)
  - Battery operated
  - High latency environment (Satellites)
- TLS/DTLS/QUIC is becoming common in IoT space

Connections can last for weeks/months and re-connects could be very disruptive

Challenge #1: AEAD confidentiality and anti-forgery limits

AEAD limits

- AEAD schemes provide confidentiality and integrity – but only up to defined limits
- Every encryption scheme has a maximum number of records/messages that can be safely protected under a single key
- Exceeding these limits increases the probability of ciphertext collisions, tag forgeries, and loss of confidentiality.
- High-volume or long-lived connections are especially at risk of hitting these limits.

Recommended reading:

draft-irtf-cfrg-aead-limits

Usage Limits on AEAD Algorithms

AEAD limits per key (assuming ~1500 bytes payload)

Limit	draft-irtf-cfrg-aead-limits	RFC8446 / RFC9147	RFC9001	NIST
AES-GCM confidentiality	$2^{32.5}$ messages 8.3 TiB	$2^{24.5}$ full size records 362 GiB	**2^{23} packets 12 GiB**	2^{32} blocks 64 GiB (SP 800-38D)
AES-GCM forgery attempts*	2^{64}	**2^{36}**	2^{36}	-
Chacha20Poly1305 forgery attempts*	2^{46}	**2^{36}**	2^{36}	-
AES-128-CCM confidentiality	2^{30} messages 1.5 TiB	2^{23} packets 11.7 GiB	**$2^{21.5}$ packets 4.1 GiB**	2^{61} blocks (SP 800-38C)
AES-128-CCM forgery attempts*	2^{30} 2^{13} for CCM_8	$2^{23.5}$	**$2^{21.5}$ packets**	-

*Forgery attempts do not apply to TCP/TLS as it drops the connection after the first failure to decrypt

Challenge #2: post-compromise security

Static Key Exfiltration risk

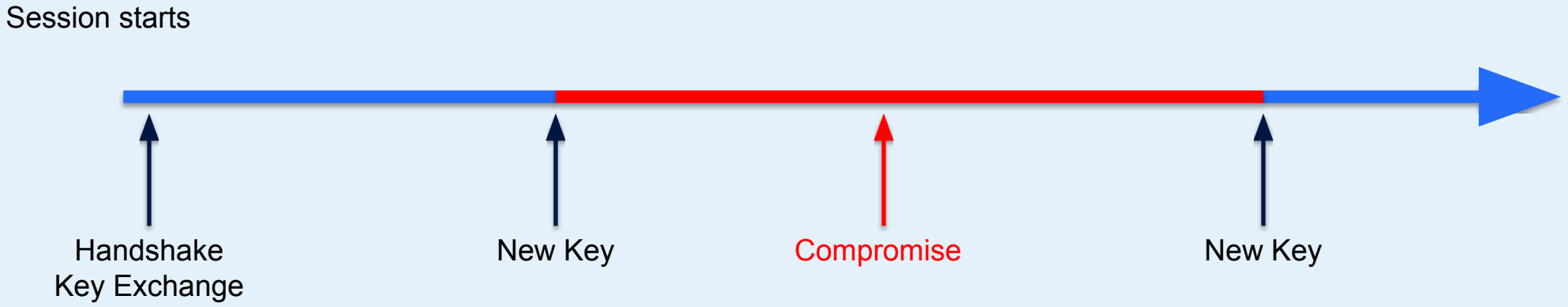
- It's enough for attacker to take a single memory snapshot to compromise session keys
- The attack is completely passive and is not detectable by the peers
- Even perfect network, application and OS level telemetry cannot guarantee protection
- Speculative execution CPU vulnerabilities can be used

Recommended reading:

RFC7624, Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement

Static and Dynamic Key Exfiltration

Static key exfiltration attack blast radius



Challenge #3: re-authentication

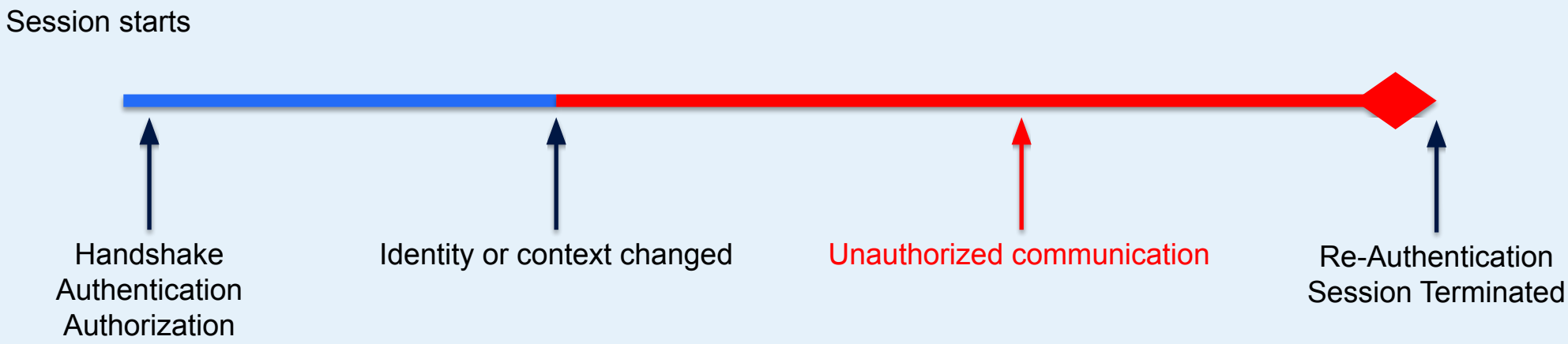
Can peer identity change?

- Long-lived session assume identity and security attributes remain valid
- But identity or context may change
  - Decommissioned virtual machine remains running
  - Employee is terminated but VPN session is still active
  - Access rights change due to role, location or policy update
  - Devices are reassigned, cloned or re-purposed

Recommended reading:

NIST SP 800-63
Digital Identity Guidelines

Stale long-lived session can be a problem



Solutions outside of TLS 1.3

(D)TLS 1.2 renegotiation

Pros:

- Allows peers to establish fresh keying material (solves AEAD limits)
- Re-establishes secrets (limits static key exfiltration attack)
- Allows peers to request updated certificates (re-authentication)

Cons:

- CVE-2009-3555 (man-in-the-middle splice attack)
- RFC 5746 **MUST** be implemented (mandates cryptographic binding between handshakes)
- Complexity to protocol state, error-prone mechanism
- Downgrade risks

Recommendation: use with caution, test thoroughly.
Consider periodic re-connect if acceptable for the application…

SSHv2 approach

Periodic Re-keying:

- Built into SSH Transport Layer Protocol spec (RFC4253)
- Either side may initiate re-keying by sending **`KEXINIT`** at any time
- Specification recommends re-keying after 1 GiB of data or after 1 hour of connection time, whichever comes first
- Implementations generally follow the recommendation

No re-authentication:

- Authentication and authorization is performed only once
- It's down to implementations to disconnect clients that no longer authorized

IKEv2 approach

Limited lifetimes for Security Associations:

- Each SA normally has explicit lifetime for traffic volume and time
- Unlike IKEv1 each endpoint is free to enforce its own SA lifetimes
- **`CREATE_CHILD_SA`** or creation new **`IKE_SA`** are acceptable mechanisms
- Creation of new SA requires a fresh key exchange

Re-authentication

- Build new SA from scratch including IKE_SA_INIT and IKE_AUTH
- Re-authentication cannot be done without re-keying
- Responder initiated re-authentication is an optional extension (RFC 4478)

Wireguard approach

Periodic Re-keying:

- Static 2-minute timer for initiator-driven rekeying
- Rekeying also performed every 2^{60} messages, but 2 minutes lapse way earlier…
- Wireguard uses ChaCha20Poly1305 that does not have a practical confidentiality limit

No re-authentication:

- Authentication and authorization is performed only once
- It's down to implementations to disconnect clients that no longer authorized

TLS 1.3 and the long-haul

TLS 1.3 KeyUpdate

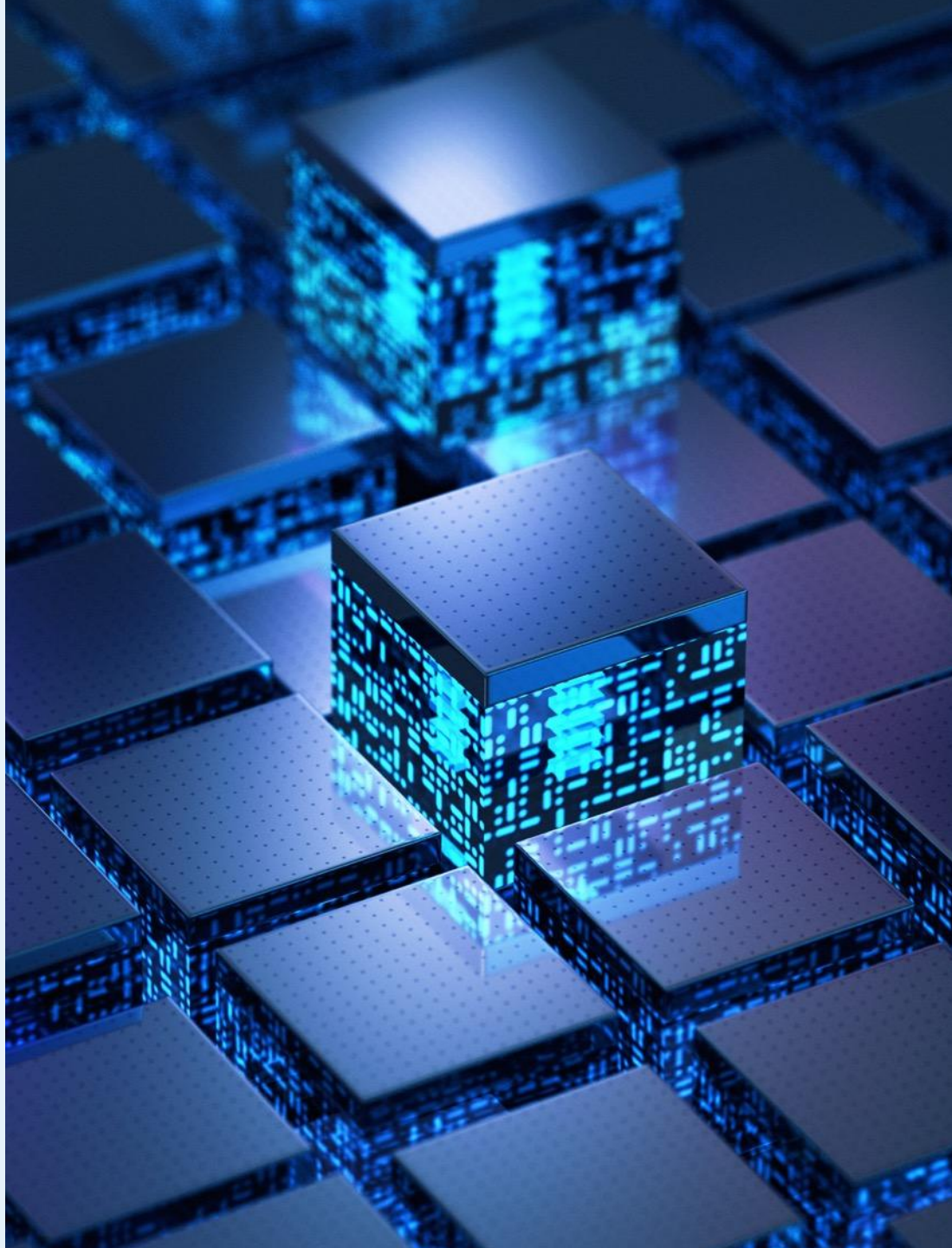
KeyUpdate instead of legacy renegotiation

- Triggered by KeyUpdate message in (D)TLS and Key Phase bit flip in QUIC
- Derives fresh set of AEAD keys
- Solves AEAD limits
- Does not solve static key exfiltration
- Does not provide re-authentication

Manual trigger

- Most TLS and QUIC libraries rely on applications to initiate key update
- Some do not provide API to trigger KeyUpdate…

1. Make sure to trigger KeyUpdate
2. Test that KeyUpdate actually works!



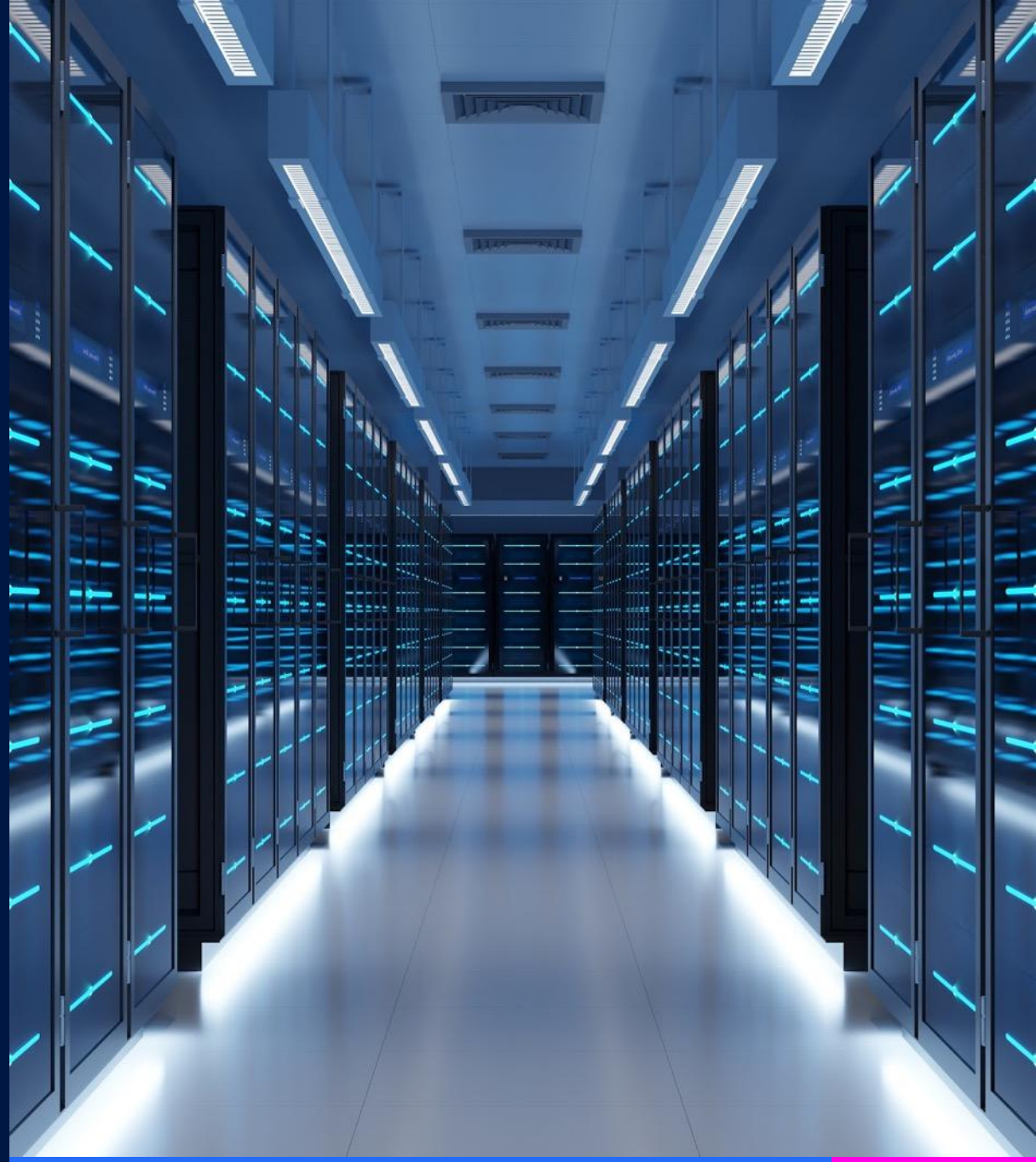
TLS 1.3 extended key update

draft-ietf-tls-extended-key-update

Replacement of TLS 1.3 KeyUpdate with full re-keying

- Developed by the IETF TLS working group
- Capability is negotiated at the start of the session
- Fresh Key Schedule is derived from new ephemeral key exchange
- Uses the same TLS group as the original handshake
- draft-ietf-quic-extended-key-update implements this capability in QUIC

Until this capability is available re-connection is required to mitigate static key exfiltration risk



Re-authentication not available in TLS 1.3

TLS working group so far declined proposals to add re-authentication capabilities in TLS 1.3

Re-authenticate on application level!

- Perform re-authentication **correctly** using exported authenticators (RFC9261)
- In HTTP/2 and HTTP/3 Fresh Server Certificates can be signaled using Secondary Certificate Authentication draft-ietf-httpbis-secondary-server-certs

Summary

Long-lived session specific solutions

Challenge	TLS 1.2	SSHv2	IKEv2	Wireguard	TLS 1.3
AEAD limits	Re-negotiate	Periodic re-key	Fresh SA	Not applicable	KeyUpdate
Static Key Exfiltration	Re-negotiate	Periodic re-key	Fresh SA	Periodic re-key	Extended Key Update (WIP)
Re-authentication	Re-negotiate	N/A	Fresh SA with re-authentication	N/A	N/A but can be done securely on application level

Recommendations for TLS 1.3 / QUIC

- Until Extended Key Update is available, perform periodic re-connect
- Design applications to handle re-connect seamlessly
- Trigger KeyUpdate periodically
- Test that your TLS and QUIC stacks support KeyUpdate
- If your threat model require re-authentication implement it on application layer. Use TLS exported authenticators for session binding.

zscaler™

THANK YOU