

# draft-ietf-tls-extended-key-update-05

Hannes  
Tschofenig

Michael  
Tüxen

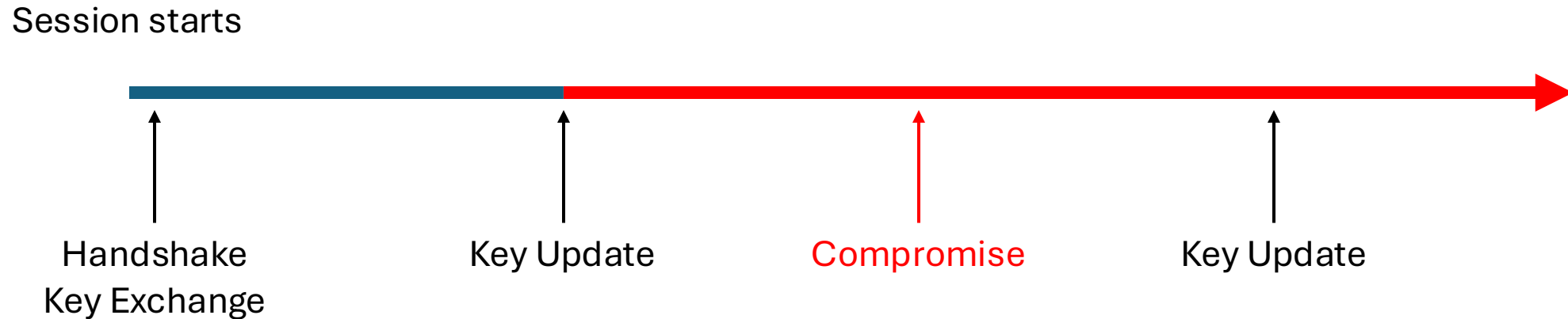
Tirumaleswar  
Reddy

Steffen  
Fries

Yaroslav  
Rosomakho

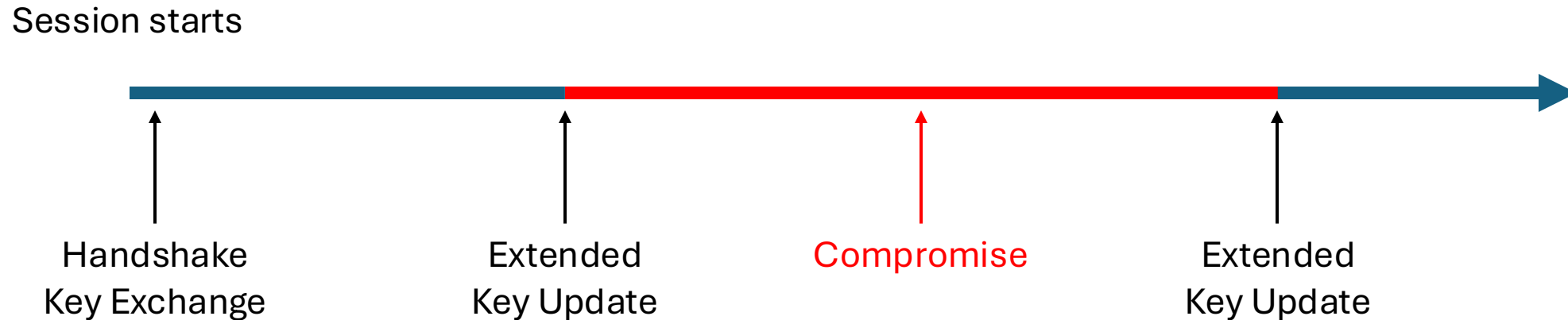
TLS  
IETF123, July 2025, Madrid

# Recap: Standard Key Update Security



Attacker can decrypt all packets since the previous Key Update until the end of the session

# Recap: **Extended** Key Update Security



Attacker can decrypt only packets between Extended Key Updates

# Ekr's review 1/4: Terminology

*I'm not sure what it means to "reestablish forward secrecy". The TLS 1.3 KeyUpdate mechanism provides forward secrecy, in that a key compromise at epoch  $N+1$  does not threaten data transmitted under key  $N$ . What this mechanism provides is post-compromise security, which is to say protection for data protected under key  $N+1$  even if key  $N$  is compromised.*

- **What does the room think?**

# Ekr's review 2/4: Conflict Resolution and Rejection

*My initial instinct is we can get rid of "retry" entirely. I see two main options.*

*Option 1: Require immediate response*

*The existing algorithms are relatively cheap and so it's not clear to me that there is a lot of value in the peer in declining to update...*

DHE operations per second, single core Xeon Gold 6442Y

	<b>X25519</b>	<b>P-256</b>	<b>P-384</b>	<b>P-521</b>
OpenSSL 3.5.1	11636.9	9364.0	9364.0	<b>260.0</b>
BoringSSL 2025-07-11 Commit cb12d9e	18542.2	12637.1	1108.8	<b>458.3</b>
AWS-LC 1.55	25471.2	12554.0	1613.0	<b>1741.4</b>

# Ekr's review 2/4 continued

*Option 2: Allow the Responder to unilaterally delay*

*Rather than having the Responder respond with "retry", the responder can just unilaterally wait  $T$  seconds (where  $T$  is whatever it would have put in the retry response). This simplifies the state machine on the Initiator and only carries slightly less information, because the Initiator has no promise that the retry would have worked anyway.*

- **What does the room think?**

# Ekr's review 2/4 continued

*## Clashed*

*I see what "clashed" is trying to do, but I think there's another opportunity to simplify: because you are determining which side wins based on information known both sides, they can each independently determine which EKURequest will be accepted and which one will be rejected [0]. So instead of having the winner send a EKUResponse rejecting the loser, I suggest it just wait for the EKUResponse from the loser.*

- **What does the room think?**

# Ekr's review 3/4: Handshake Messages

*You're registering 3 new handshake messages out of a space that has maximum size 256 but is actually quite a bit smaller, as documents in RFC 7983 (the range that does not require coordination is only 25-63). Is there a reason not to just have a single ExtendedKeyUpdate mechanism with a type field?*

- **What does the room think?**

# New in -05: complete key derivation

- Replaced single HDKFD Expand
- Derive new exporter and resumption secrets in addition to application traffic secrets

```
Master Secret N
|
v
Derive-Secret(., "key derived", "")
|
v
(EC)DHE -> HKDF-Extract = Master Secret N+1
|
+-----> Derive-Secret(., "c ap traffic2",
|           ExtendedKeyUpdateRequest ||
|           ExtendedKeyUpdateResponse)
|           = client_application_traffic_secret_N+1
|
+-----> Derive-Secret(., "s ap traffic2",
|           ExtendedKeyUpdateRequest ||
|           ExtendedKeyUpdateResponse)
|           = server_application_traffic_secret_N+1
|
+-----> Derive-Secret(., "exp master2",
|           ExtendedKeyUpdateRequest ||
|           ExtendedKeyUpdateResponse)
|           = exporter_master_secret_N+1
|
+-----> Derive-Secret(., "res master2",
|           ExtendedKeyUpdateRequest ||
|           ExtendedKeyUpdateResponse)
|           = resumption_master_secret_N+1
```

# Ekv's review 4/4: Label Strings

*Do we really need new label strings for the key derivations in S 5? I haven't done a thorough analysis, but given that you forbid the use of KeyUpdate in combination with this spec it's not clear to me how you would get a conflict in terms of the inputs.*

- **What does the room think?**

# New in -05: standard key update replacement

- If TLS peers agreed on Extended Key Update they **MUST NOT** use Standard Key Update
  - That is, KeyUpdate message is *unexpected\_message*

# New dependents on this work

## QUIC “sister” draft

- draft-ietf-quir-extended-key-update is adopted by QUIC WG
- Same mechanism, but uses key phase bit in QUIC header to indicate completion of extended key update

## Using DTLS for Key Management for SCTP DTLS Chunk

- draft-tuexen-tsvwg-dtls-chunk-key-management-00
- Proposed DTLS Chunk key update mechanism relies on this work

# Next Steps

- More DTLS considerations (PRs on GitHub)
- More implementations and interoperability testing
- More reviews and further refine wording

Thank you!