Hidden Stream Ciphers and TMTO Attacks on TLS 1.3, DTLS 1.3, QUIC, and Signal

John Preuß Mattsson
Expert Cryptographic Algorithms and Security Protocols
Ericsson Research

Symmetric key ratchets

- Symmetric key ratchets provide efficient "forward secure" symmetric rekeying without Diffie-Hellman.
- Made popular by the Signal protocol. TLS 1.3 key update was inspired by the symmetric ratchet in Signal. DTLS 1.3 and QUIC use the TLS 1.3 handshake.
- A Key Derivation Function (KDF) $\mathrm{H}()$ is frequently used to update and replace the current key $k_{i+1} = \mathrm{H}(k_i)$. The old key k_i is destroyed. This creates a chain of keys $k_0, k_1, k_2 \ \ldots$
- With "forward secrecy", Signal and TLS 1.3 mean that compromise of key k_{i+1} does not lead to compromise of $k_0, k_1, \ldots, k_i$

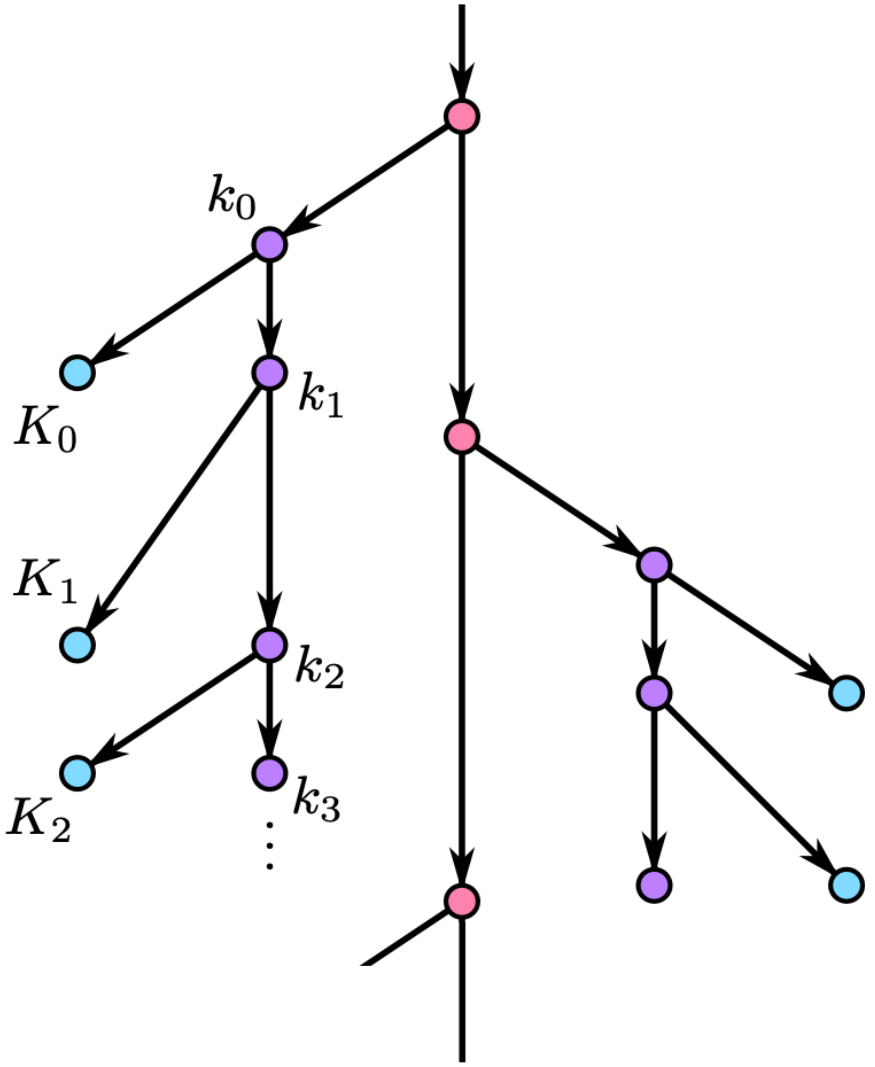
Model from Polhem's mechanical alphabet (CC BY 4.0 Deed)

Signal protocol and the double ratchet

- After the X3DH handshake is finished and at least one step of the Diffie-Hellman Ratchet has been performed, a 256-bit initial chain key k_0 is derived (to simplify things we only discuss one of the directions).
- A chain of keys k_0, k_1, k_2 ... derived from the initial chain key k_0 are used to protect all future messages sent in one direction until the Diffie-Hellman ratchet is used again.
- Message i is encrypted using a 256-bit message key K_i.
- Before each message is sent, the message key and the next chain key are computed using the symmetric-key ratchet

$$K_i = H'(k_i) = \mathrm{KDF}(k_i, label_1, n_2) \ ,$$
$$k_{i+1} = H(k_i) = \mathrm{KDF}(k_i, label_2, n) \ .$$

- The size n of the chain keys and the size n_2 of the message keys are always 256 bits, i.e., $n = n_2 = 256$.
- The Signal technical specification does not give recommendations on how often to use the Diffie-Hellman ratchet or limits on chain lengths.



TLS 1.3 and the key update mechanism

- After the TLS 1.3 handshake is finished an initial traffic secret k_0 is derived. A chain of keys k_0, k_1, k_2 ... derived from k_0 are used to protect all future messages sent in one direction of the connection.
- The size of the traffic secrets n and the size of the AEAD keys n_2 depends on the selected cipher suite.
- Once the handshake is complete, it is possible to update the traffic secret using the key update mechanism.

$$k_{i+1} = H(k_i) = \mathrm{KDF}(k_i, \text{"traffic upd"}, n)$$

- The AEAD key K_i and initialization vector IV_i are derived from k_i as

$$K_i = \mathrm{KDF}(k_i, \text{"key"}, n_2) \ ,$$
$$IV_i = \mathrm{KDF}(k_i, \text{"iv"}, 96) \ .$$

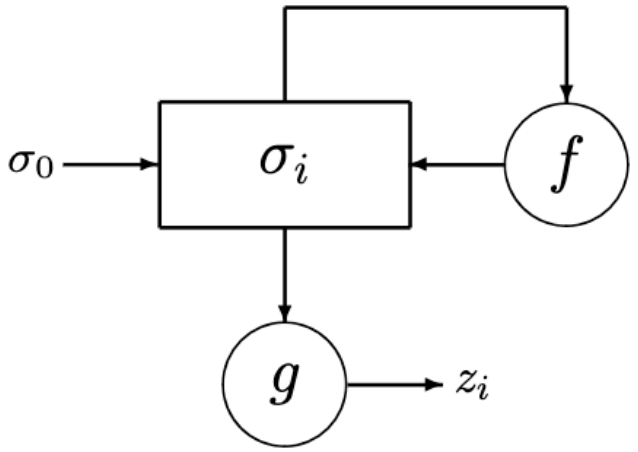
Cipher suite	n	n_2
TLS_AES_128_GCM_SHA256	256	128
TLS_AES_256_GCM_SHA384	384	256
TLS_CHACHA20_POLY1305_SHA256	256	256
TLS_AES_128_CCM_SHA256	256	128
TLS_AES_128_CCM_8_SHA256	256	128

- The AEAD nonce for each record is calculated as $IV_i \oplus S$ where S is the record sequence number. The (secret) random IV is intended to increase security against multi-key attacks.
- For each cipher suite, TLS 1.3 has a limit for the number of encryption queries q. Key update is recommended before the limit is reached (every $2^{24.5}$ records for AES-GCM). Frequent use of the key update mechanism is therefore expected in connections where a large amount of data is transferred.
- DTLS 1.3 restricts the number of key updates to 2^{48}. TLS 1.3 and QUIC do not restrict the number of key updates.
- **Our result:** The random nonce in ChaCha20 does not increase multi-key security as $n = n_2 = 256$

Hidden stream ciphers and TMTO attacks

- A synchronous stream cipher is described by the equations $\sigma_{i+1} = f(\sigma_i), z_i = g(\sigma_i), c_i = h(z_i, p_i)$, where σ is the state, f is the next-state function, g is the output function, and h is used to combine keystream and plaintext.
- It turns out that the ratchets in Signal and TLS 1.3 can be modelled as (non-additive) synchronous stream ciphers. The initial internal state is k_0, the next-state function $k_{i+1} = \mathrm{H}(k_i)$ modifies the inner state, the output function $z_i = (K_i, IV_i) = g(k_i)$ uses the inner state to produce "keystream" z_0, z_1, z_2 ..., and the ciphertexts are a function $c_i = h(z_i, p_i)$ of "keystream" and plaintext, where p_i is all the application data encrypted with the key K_i.
- Stream ciphers are vulnerable to Time Memory Trade-Off (TMTO) attacks such as Babbage-Golić and Biryukov-Shamir. These attacks take advantage of the internal state and apply to the Signal symmetric-key ratchet and the TLS 1.3 key update.

$$\begin{aligned} \sigma_i &\quad \text{state} = k_i \\ z_i &\quad \text{keystream} = (K_i, IV_i) \end{aligned}$$



Hidden stream ciphers and TMTO attacks

- In Babbage-Golić, the attacker tries to find one of the many internal states instead of the stream cipher key. The attacker generates M random states $k_0, k_1, \ldots k_i, \ldots, k_{M-1}$ from the total number of states $N = 2^n$, calculates an output string y_i for each state k_i, and stores the pairs (y_i, k_i) ordered by y_i. In the real-time phase the attacker collects D output strings $y_0, y_1, \ldots, y_j, \ldots, y_{D-1}$. Requirements on the output strings depend on the function $h()$ used to combine keystream and plaintext. By the birthday paradox the attacker can find a collision $y_i = y_j$ and recover an inner state k_i in time $T = N/M$, memory M, data D, and preprocessing time $P = M$, where $1 \leq T \leq D$.
- An example point on this trade-off is $P = T = M = D = N^{1/2}$. A difference compared to a normal birthday attack is that the attacker, on average, recovers the last $D/2$ states $k_i, \ldots k_{D-2}, k_{D-1}$ as well as any future states.
- If D is limited, a reasonable assessment (given that the attacker recovers $\approx D$ states) is that the security is reduced by $\min(d, n/2)$, where $d = \log_2 D$. If D is unlimited the security is reduced by $n/2$.
- Biryukov-Shamir's attack combines the Hellman TMTO attack and the Babbage-Golić attacks.
- The effective stream cipher specific time memory trade-offs (TMTO) will be possible if the state size is less than twice the security level. Based on these attacks, modern stream ciphers follow the design principle that the security level is at most $n/2$ and that the state size in bits n should therefore be at least twice the security level.
- TLS 1.3 and Signal do not explicitly state the intended security level, but the key length n_2 of the AEAD keys K_i can typically be seen as the intended security level and we see that TLS 1.3 and Signal do not follow design principles for stream ciphers. The state size in Signal is always equal to the security level and the state size in TLS 1.3 is in some cases equal or 1.5 times the security level. **As a result, TLS 1.3 and Signal offer far less than the expected security against these types of TMTO attacks.**

Signal protocol - analysis and recommendations

- A significant problem with the X3DH protocol is that it does not mandate ephemeral Diffie-Hellman. As stated in the specification, the server might be all out of one-time Diffie-Hellman keys. Messages sent before receiving an ephemeral public key from the responding party provide neither forward secrecy with respect to long-term keys nor replay protection
- Old ephemeral Diffie-Hellman keys are problematic as they are to be considered long term-keys and therefore cannot be used to provide forward secrecy with respect to long-term keys, which often is a desired property.

- **We recommend the Signal Protocol to:**
  - Introduce a low limit on how many times the the symmetric-key ratchet can be used.
    - With a low limit, the Signal protocol would provide a theoretical security level close to 256 bits
  - Mandate ephemeral Diffie-Hellman with fresh keys before sending messages.
    - Could also restrict the type of messages that can be sent similar to TLS 1.3 0-RTT data.
  - Mandate frequent use of the Diffie-Hellman ratchet based on time and data.
  - Allow and recommend use of 512-bit chain keys.
  - Clearly state the intended security level.

Key exfiltration attacks and frequent ECDHE in TLS 1.3

- The key update mechanism gives significantly worse protection against static key exfiltration attacks than frequent ECDHE.
- In attack scenarios like side-channel attacks on Internet of Things (IoT) devices mandating physical proximity, the distinction between static and dynamic key exfiltration is substantial.
- Two essential zero trust principles are to assume that breach is inevitable or has likely already occurred, and to minimize impact when breach occur. One type of breach is key compromise or key exfiltration.
- For IPsec, ANSSI recommends enforcing periodic rekeying with ephemeral Diffie-Hellman every hour and every 100 GB of data.

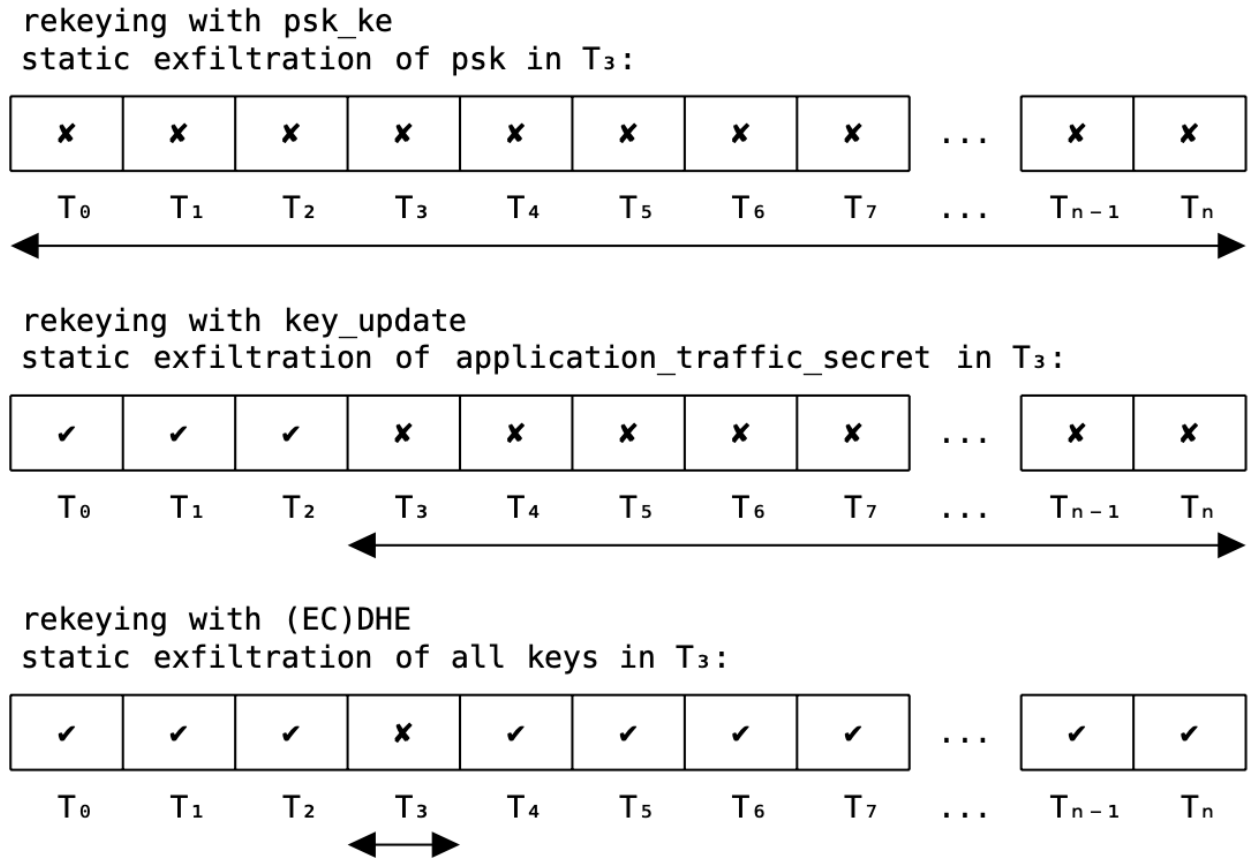


Fig. 3. TLS 1.3 - Impact of static key exfiltration in time period T_3 when psk_ke, key_update, and (EC)DHE are used.

DTLS 1.3/QUIC procedure to calculate AEAD limits

- In TLS 1.3, DTLS 1.3, and QUIC implementations should do a key update before reaching defined limits.
- TLS 1.3 just states the limits. DTLS 1.3 and QUIC define a procedure to calculate the rekeying limits.
  - The procedure suggests rekeying when the single-key confidentiality advantage (IND-CPA) is greater than 2^{-60} or when the single-key integrity advantage (IND-CTXT) is greater than 2^{-57}.
- Our analysis is that these procedures are flawed both theoretical and in practice.
  - The procedures uses single-key advantages to suggest rekeying which transform the problem to a multi-key problem and invalidates the single-key calculation used to suggest the rekeying.
  - The confidentiality rekeying limits for AES-GCM and AES-CCM and the integrity rekeying limit for AES-CCM coincides pretty well with when the confidentiality and integrity advantages starts to grow significantly faster than linear. **These rekeying limits do significantly improve security.**
  - The integrity limits for AES-GCM and ChaCha20-Poly1305 do not improve security as the single-key integrity advantages are bounded by a function linear in v, the number of forgery attempts. **The forgery probability is therefore independent of the rekeying.**
  - For CCM_8 (or an ideal 64-bit tag) the procedure gives **illogical results unsuitable for practical use**. Looking at the bound for the CCM_8 integrity advantage it is easy to see that CCM_8 performs very close to an ideal MAC for $v \lesssim 2^{36}$. Rekeying after 128 forgery attempts **can create denial-of-service problems**.

TLS 1.3 family - analysis and recommendations

We recommend TLS 1.3, DTLS 1.3, and QUIC to:

- Mandate frequent rekeying with EC(DHE) based on time and data. Every hour and every 100 GB of data for non-constrained implementations.
- Deprecate the procedure used for DTLS 1.3 and QUIC to calculate key limits.
- Mandate traffic secrets twice the AEAD key size for new cipher suites.
- Introduce strict limits on the use of the key update mechanism. DTLS 1.3 already restricts the number of key updates to 2^{48}.
- Standardize TLS_CHACHA20_POLY1305_SHA512.
- Clearly state the intended security levels.

Summary

— While we do not believe that the weaknesses pose a practical attack vector today, the attacks point to design flaws in the key update mechanisms in TLS 1.3 and Signal, alternatively a lack of clearly stated security levels.

— We find the design of the Signal protocol with a symmetric-key ratchet combined with a Diffie-Hellman ratchet very appealing as the protocol seems designed for frequent use of ephemeral Diffie-Hellman.

— We find several of the design choices in the TLS 1.3 handshake non-optimal resulting in that TLS 1.3 is problematic to use as a drop-in replacement of TLS 1.2. The problem can be overcome by setting up new connections.

Summary by Nick Youngson CC BY-SA 3.0 Alpha Stock Images

Future work

- Update TLS 1.3 and Signal specification based on recommendations. Some work already done.
- Evaluate the impact of this work on new protocols using symmetric ratchets such as MLS (RFC 9420), EDHOC (draft-ietf-lake-edhoc), and OSCORE (draft-ietf-core-oscore-key-update, RFC 8613).
- Evaluate implementations and deployments of the protocols. How often do actual deployments perform symmetric key update and ephemeral Diffie-Hellman? Can an active attacker influence the frequency?
- Improved process for how to calculate key limits.

Dark world of the future by Karhaym (CC BY-NC-ND 3.0 Deed)

https://www.ericsson.com/en/security