Security of Symmetric Ratchets and Key Chains
Implications for Protocols like TLS 1.3, Signal, and PQ3

John Preuß Mattsson

Ericsson Research, Stockholm, Sweden
john.mattsson@ericsson.com

Abstract. Symmetric ratchets and one-way key chains play a vital role in numerous important security protocols such as TLS 1.3, DTLS 1.3, QUIC, Signal, MLS, EDHOC, OSCORE, and Apple PQ3. Despite the crucial role they play, very little is known about their security properties. This paper categorizes and examines different ratchet constructions, offering a comprehensive overview of their security. Our analysis reveals notable distinctions between different types of one-way key chains. Notably, the type of ratchet used by TLS 1.3, Signal, and PQ3 exhibit a significant number of weak keys, an unexpectedly high rate of key collisions surpassing birthday attack expectations, and a predictable shrinking key space susceptible to novel Time-Memory Trade-Off (TMTO) attacks with complexity $\approx N^{1/4}$. Consequently, the security level provided by e.g., TLS 1.3 is significantly lower than anticipated. To address these concerns, we analyze the aforementioned protocols and provide numerous concrete recommendations for enhancing their security, as well as guidance for future security protocol design.

Keywords: TLS 1.3 · QUIC · DTLS 1.3 · Signal · MLS · EDHOC · OSCORE · Secret-key Cryptography · Key Derivation · Hash Functions · Ratchet · Hash Chain · KDF Chain · Key Chain · One-way Permutation · Weak Keys · Collisions · Key Space · Stream Cipher · TMTO

1 Introduction

Transport Layer Security (TLS) is widely regarded as the most important security protocol in the information and communications technology industry. Its latest version, TLS 1.3 [50], has been widely adopted by web services and various industries. Furthermore, several other crucial protocols, including QUIC [26, 61], EAP-TLS 1.3 [49], DTLS 1.3 [51], DTLS-SRTP [37], and DTLS/SCTP [63, 64] rely on the TLS 1.3 handshake. Recognizing its significance, the US National Institute of Standards and Technology (NIST) mandates support for TLS 1.3 since January 1, 2024 [38]. TLS-based protocols are increasingly employed to secure long-lived interfaces in critical infrastructure, such as 5G networks [1].

The Signal protocol [56] enjoys widespread popularity for end-to-end encryption of voice calls and instant messaging. Apart from the Signal messaging service itself, the Signal protocol is employed by WhatsApp, Meta Messenger, and Google Messages. The Signal messaging service is approved for use by the U.S. Senate and is recommended for the staff at the European Commission, further solidifying its reputation as a trusted and reliable secure communication solution.

TLS 1.3 and the Signal protocol both leverage symmetric key ratchets to facilitate efficient rekeying and forward secrecy without Diffie-Hellman. A Key Derivation Function (KDF) is used to regularly update the current key k_i through the operation $k_{i+1} = \mathrm{KDF}(k_i, \dots)$, establishing a sequence of keys $k_0, k_1, k_2, \dots$ referred to as a key chain. Inspired by Signal and TLS 1.3, several newer security protocols, such as MLS [5], EDHOC [52], OSCORE [25], and Apple PQ3 [3] incorporate symmetric key ratchets. While DTLS 1.3, QUIC, and PQ3 employ the same type of chain as TLS 1.3 and Signal, MLS utilizes a counter in its chain, OSCORE uses a randomized chain, and EDHOC uses an application-provided context, allowing the application to affect the type of chain. It is known [31] that the type of hash chains utilized in TLS 1.3 and Signal, in some aspects, behaves like a set of uniformly sampled keys. However, it was only recently recognized [47] that ratchets with too small states give rise to notable security issues. The impact of different chain constructions on the security properties of the chain and the security protocols employing them remains unknown.

Through our comprehensive analysis, we categorize and assess various key chain constructions, conducting an in-depth exploration of their security properties. This examination reveals notable distinctions in the number of collisions occurring within chains, between chains, and between a chain and a random set. Our findings unveil critical aspects concerning the type of key chain employed for example in TLS 1.3, Signal, and PQ3. We identify a substantial presence of weak keys, an unexpected number of key collisions surpassing birthday attack expectations, and that the predictable shrinking key space can be exploited in novel Time-Memory Trade-Off (TMTO) attacks with complexity $T = M \approx N^{1/4}, P \approx N^{1/2}$ and $T = M = P \approx N^{1/3}$, where N is the state space. The type of chain used in MLS has better properties for short chains; however, it provides zero bits of security against precomputation attacks if the chain length is unbounded. The chain in MLS is bounded. The new TMTO attacks improve previous work by Babbage-Golić [4,21], Biryukov-Shamir [9], and Hong-Kim [24]. Our findings show that the security issues with the types of key chains employed in TLS 1.3, Signal, PQ3, MLS, and EDHOC are considerably worse than suggested by [47]. We recommend that the type of key chain used in TLS 1.3, Signal, and PQ3 is phased out in existing protocols and not used in future protocols.

As a result, (D)TLS 1.3, QUIC, Signal, PQ3, MLS, EDHOC, and OSCORE provide a significantly lower level of security than anticipated based on their AEAD key sizes. When used with key update, the popular TLS 1.3 cipher suite CHACHA20_POLY1305_SHA256 has a high fraction of weak keys, a key space significantly smaller than the expected security level, and is vulnerable to a trivial attack that finds a (key, nonce) collision with time complexity $m/2^{256}$ where m is number of key updates. If the number of key updates is unbounded, the security level against TMTO attacks is one fourth of the state size implying that Signal, PQ3, and most TLS 1.3 cipher suites only provide ≈ 64 bits of security. Based on the aforementioned findings, we have developed several concrete recommendations for Signal, PQ3, MLS, EDHOC, OSCORE, and protocols based on TLS 1.3. These recommendations aim to address the identified vulnerabilities and enhance the security of these and future security protocols. In addition to large states, it is essential to impose strict limits on the maximum chain length.

Symmetric ratchets have important use cases for messaging applications, group communication, and constrained environments. The use in general non-constrained two-party protocols is questionable and goes against established best practices for long-lived connections [2,35,55]. Symmetrical ratchets offer very limited protection against key leakage and exfiltration. We stress that one-way key chains and ratchets should not be seen as general replacements for periodic rekeying with ephemeral key exchange in long-lived connections.

This work was inspired by and builds upon [47] which explored various aspects of the key update mechanisms in TLS 1.3 and Signal, demonstrating their modeling as non-additive synchronous stream ciphers. The introduction and preliminaries sections significantly borrow, with permission, from [47]. In Section Sect. 3, we present a comprehensive analysis of the security properties of one-way key chains and discuss their implications for TLS 1.3 and Signal. Our results can be summarized as follows:

- In Sect. 3.1, we introduce a systematic categorization of key chains and propose a novel type of key chain based on one-way permutations.
- Sect. 3.2 demonstrates that both TLS 1.3 and the Signal protocol suffer from a significant number of weak keys.
- Sect. 3.3 reveals that the number of key collisions in TLS 1.3 and Signal is considerably higher than expected from the birthday attack, and that collisions compromise replay protection.
- In Sect. 3.4 and Sect. 3.5 we analyze how fast key space sizes for various types of key chains shrink, presenting new iterative and explicit formulas.
- Sect. 3.6 describes new single- and multi-connection TMTO attacks for stream ciphers with predictable shrinking states, akin to the ratchets in TLS 1.3, Signal, and MLS. We derive security requirements for all types of chains, relating chain state size n, security level λ, and maximum chain length ℓ.
- In Sect. 3.7, we compare the security properties of different types of chains.
- In Sects. 4 and 5, we conduct a more in-depth analysis of TLS 1.3, DTLS 1.3, QUIC, Signal, PQ3, MLS, EDHOC, and OSCORE, presenting a comprehensive set of concrete recommendations for each protocol.
- Appendices A and B analyze collisions between multiple key chains and propose a construction to expand the state space of ρ-, ξ-, and ω-chains.

2 Preliminaries

2.1 Signal Protocol and the Symmetric-Key Ratchet

The following is a summary of Section 2.1 in [47]. For more detailed information, refer to [8,14,47] or the Signal technical specification [56]. The Signal protocol utilizes a chain of 256-bit keys $k_0, k_1, k_2, \ldots$ derived from the initial chain key k_0 to secure all future messages transmitted in a single direction until the Diffie-Hellman ratchet is used again. The decision on when to use the Diffie-Hellman ratchet to derive a new initial chain key k_0 is implementation-specific. Each message i is encrypted using a 256-bit message key K_i, which is used only once. Before transmitting each message, both the chain key and the message key undergo updates using the symmetric-key ratchet, as outlined in [56]. The message key K_i and the next chain key k_{i+1} are computed as

$$\begin{aligned} K_i &= \mathrm{KDF}(k_i, \mathit{label}_1, \texttt{""}, n_s) \ , \\ k_{i+1} &= \mathrm{KDF}(k_i, \mathit{label}_2, \texttt{""}, n) \ , \end{aligned} \tag{1}$$

where "" is an empty context. The only state retained between iterations is the chain key. While the Signal specification does not enforce a specific KDF or labels, it recommends HMAC-SHA2 with $\mathit{label}_1 = 0\text{x}01$ and $\mathit{label}_2 = 0\text{x}02$. Regardless of the chosen algorithms, both the size of the chain keys n and the size of the message keys n_s are always 256 bits. The Signal Protocol does not mandate any specific Authenticated Encryption with Associated Data (AEAD) algorithm but recommends AES-256-CBC with HMAC-SHA2. It suggests deriving a 32-byte encryption key, a 32-byte authentication key, and a 16-byte IV from the message key. Signal does not explicitly state the intended security level λ, but the length of the encryption key can typically be seen as the intended security level. Therefore,

$$n = n_s = \lambda = 256 \ . \tag{2}$$

2.2 TLS 1.3 and the Key Update Mechanism

The following is a summary of Section 2.2 in [47]. For more detailed information, refer to [47] or the TLS 1.3 specification [50]. In TLS 1.3, a chain of n-bit keys $k_0, k_1, k_2, \ldots$ is derived from the initial traffic secret k_0. This key chain is used by the record protocol to protect all messages sent in one direction over the connection. The size of the traffic secrets is the output size n of the hash function in the selected cipher suite. Table 1 provides a list of the initial cipher suites.

After the handshake is complete, it is possible to update the traffic secret using a key update mechanism. The next traffic secret k_{i+1} is computed using a KDF based on HKDF-Expand [33] as

$$k_{i+1} = \mathrm{KDF}(k_i, \text{"traffic upd"}, \texttt{""}, n) \ . \tag{3}$$

The only state retained between iterations is the traffic secret. From the traffic secret k_i, the AEAD key K_i and the initialization vector IV_i are derived as

$$\begin{aligned} K_i &= \mathrm{KDF}(k_i, \text{"key"}, \texttt{""}, n_s) \ , \\ IV_i &= \mathrm{KDF}(k_i, \text{"iv"}, \texttt{""}, n_{iv}) \ . \end{aligned} \tag{4}$$

For each record, the AEAD nonce is calculated by XORing IV_i with the record sequence number. The size of the key K_i depends on the AEAD key length n_s specified in the chosen cipher suite, which may not be equal to n as in the Signal Protocol. The nonce size n_{iv} is fixed at 96 bits for all the cipher suites listed in Table 1. The sequence number is initially set to zero, incremented for each message, and reset to zero every time the key update mechanism is used.

Table 1. The five initial cipher suites in TLS 1.3 [50]. The importance of the difference and the ratio between the key update state size n and the security level λ is explained in Sect. 3. n_{iv} is the size of the IV.

Cipher suite	n	$\lambda = n_s$	n_{iv}	$n - \lambda$	n/λ
AES_128_GCM_SHA256	256	128	96	128	2.0
AES_256_GCM_SHA384	384	256	96	128	1.5
CHACHA20_POLY1305_SHA256	256	256	96	0	1.0
AES_128_CCM_SHA256	256	128	96	128	2.0
AES_128_CCM_8_SHA256	256	128	96	128	2.0

In TLS 1.3, a single AEAD key K_i is typically used to protect many record protocol messages. The key update mechanism is recommended to be employed before reaching a specific limit defined by the cipher suite (e.g., every $2^{24.5}$ records for AES-GCM). Therefore, in connections with a large amount of data transfer, frequent key updates are expected. TLS 1.3 does not impose any restrictions on the number of key updates that can occur. DTLS 1.3 [51] and QUIC [26, 61], which are based on TLS 1.3, also use the TLS 1.3 handshake and cipher suites. While HTTP/2 [60] uses TLS, HTTP/3 [10] uses QUIC. QUIC does not impose any limitations on the number of key updates. However, DTLS 1.3 restricts the number of key updates to 2^{48}. TLS 1.3, DTLS 1.3, and QUIC do not explicitly state the intended security level λ, but the length n_s of the AEAD key can typically be seen as the intended security level.

2.3 Definition of Weak Keys

A good definition of weak keys is provided in [65], and the following is a summary of its main points. Weak keys are those that induce undesirable behavior in a cryptographic function. Typically, weak keys constitute a minuscule fraction of the entire key space, meaning that if keys are randomly generated, the likelihood of encountering a weak key and compromising security is very low. Nevertheless, it is considered highly desirable for a cryptographic function to possess no weak keys. In a cryptographic function with no weak keys, all keys within its key space are equally strong, resulting in a homogenous key space. The pursuit of homogeneity is a fundamental objective in cryptographic design. Conversely, a significant presence of weak keys is considered a severe flaw in any design.

3 Our Results

3.1 Classification of Key Chains

Different types of key chains $k_0, k_1, \ldots, k_{m-1}$ have not been systematically categorized and analyzed previously. In this work, we define and analyze four types of key chains, which we name ρ-, ξ-, ω-, and π-chains. These chains are derived from one-way random mappings and permutations. In the literature, chains like this are known by many names such as hash chains, rainbow chains, key chains, KDF chains, one-way chains, ratchets, and Markov chains with uniform transition probabilities. $N = 2^n$ is the state space of the chain.

ρ-chain. A chain where the same random mapping $H()$ is used in all iterations

$$k_{i+1} = H(k_i) \ . \tag{5}$$

This is a deterministic chain, commonly referred to as a hash chain. This type of chain is used in TLS 1.3 [50], Signal [56], EDHOC [52], Pollard's rho algorithm [46], and Hellman's TMTO attack [23]. The name ρ comes from the fact that this type of chain eventually ends up in a repetitive cycle. Many aspects of ρ-chains are described in [20, 24, 31, 39]. A ρ-chain can be implemented with a KDF

$$k_{i+1} = \mathrm{KDF}(k_i, \mathit{label}, \mathit{context}, n) \ , \tag{6}$$

using a fixed context. Both TLS 1.3 and Signal utilize an empty context.

ξ-chain. A chain where a different random mapping $H_i()$ is used in each iteration

$$k_{i+1} = H_i(k_i) \ . \tag{7}$$

This is a deterministic chain, commonly known as a rainbow chain. This type of chain is used in rainbow tables [44], MLS [5], EDHOC [52], and is also described in [12]. We use the name ξ as the character looks like a rainbow table with three layers. A ξ-chain can be implemented by including a counter i in the context

$$k_{i+1} = \mathrm{KDF}(k_i, \mathit{label}, i, n) \ . \tag{8}$$

ω-chain. A chain where a randomly chosen random mapping $H_{r_i}()$ is used in each iteration

$$k_{i+1} = H_{r_i}(k_i) \ . \tag{9}$$

This is a randomized chain sampled from a set of chains. It can be seen as a randomized rainbow chain, or a chain using universal hashing. This type of chain is used in EDHOC [52], OSCORE [25], and is described in [30,32,45]. As this is a randomized chain, different instantiations with the same initial key k_0 will result in different chains. We use the name ω as the key space size can be described by the omega function, see Sect. 3.5. An ω-chain can be implemented by including a new random value r_i in each context

$$k_{i+1} = \mathrm{KDF}(k_i, \mathit{label}, r_i, n) \ . \tag{10}$$

π-chain. A chain where the same one-way permutation $\pi()$ is used in all iterations.

$$k_{i+1} = \pi(k_i) \ . \tag{11}$$

This is a deterministic chain using a one-way permutation. We only consider permutations with no small cycles. Known constructions for one-way permutations involve the discrete logarithm problem in finite fields or elliptic curves. The elliptic curve construction in [29] can be used to construct permutation groups of size $2p+2$, where p is a prime. In [22] is it shown how [29] can be extended to construct a one-way permutation with a large minimum cycle length like e.g., 2^{128}. We assume that the minimum cycle length is selected to exceed the maximum chain length, ensuring the absence of collisions within the π-chains. To our knowledge the idea to use one-way permutations for key chains is novel.

Connections and Instances. Concerning the deterministic ρ-, ξ-, and π-chains, our assumption is that each protocol instance, known as a connection in TLS 1.3, uses the same context and mapping or permutation in each iteration of the key chains. This implies that the same initial k_0 always leads to the same chain. The randomized ω-chain will, with very high probability, lead to different chains even if two protocol instances happen to use the same initial k_0.

In our analysis, we will explore both single-connection attacks, which target a single key chain, and multi-connection attacks, which target multiple key chains. This approach aligns with attacks on stream cipher instances [4, 9, 21]. The multi-connection attacks compromise at least one of the connections. The single-connection attacks in this paper find, on average, the last half of the keys in the connection. Consequently, these attacks are single-key attacks on the final key in the chain. In security protocols like TLS 1.3, where each connection can involve a large number of linked keys, compromising a single key leads to the compromise of multiple keys and the single-key security of a key is the multi-key security of all the previous keys in the chain. Thus, the distinction between single-key (single-user) and multi-key (multi-user) attacks is not very meaningful.

Chain Behavior. A finite key chain $k_0, k_1, \ldots, k_{m-1}$ behaves in three different ways as shown in Fig. 1. A chain without any collision $k_i = k_j$ behaves as shown in Fig. 1a. A π-chain will never have any collisions. ρ-chains, ξ-chains, and ω-chains may have one or more collisions. A collision $k_i = k_j$ in a ρ-chain results in a repetitive cycle where $k_{i+d} = k_{j+d}$ for $d \geq 0$, as shown in Fig. 1b. ξ-chains and ω-chains will always recover from a collision as shown in Fig. 1c. ξ-chains and ω-chains do not have any repetitive cycles.

3.2 Collisions, Repetitive Cycles, and Weak Keys in ρ-chains

A ρ-chain $k_0, k_1, \ldots, k_{m-1}$ where $k_{i+1} = H(k_i)$, such as the chain of traffic secrets in TLS 1.3 or the chain keys in Signal, can only behave in two ways, either the chain has a collision $k_i = k_j$, resulting in a repetitive cycle and more collisions, or

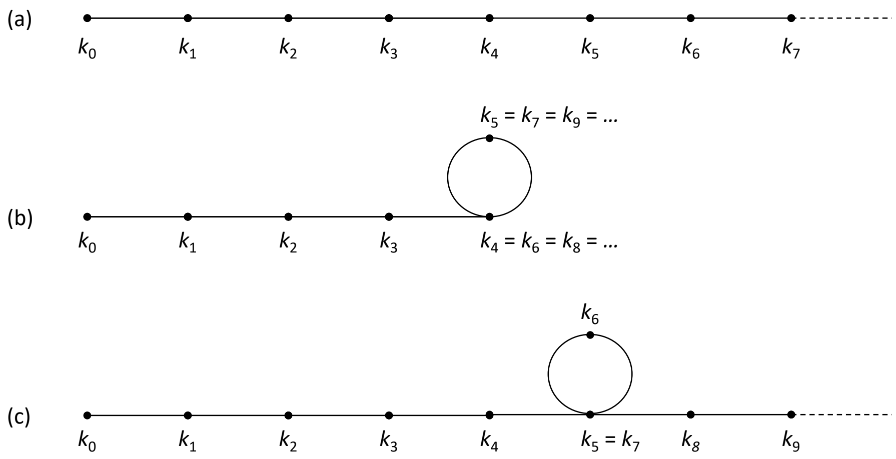

Fig. 1. Examples of the three possible behaviors of a key chain: (a) a ρ-, ξ-, ω-, or π-chain without any collision, (b) a ρ-chain from a weak key k_0 which enters into a repetitive cycle, and (c) a ξ- or ω-chain that rebounds from a collision without entering a repetitive cycle. The cycle length 2 is just an example.

the chain has no collision and no repetitive cycle. In TLS 1.3 each collision leads to many (key, nonce) collisions for the AEAD which may break confidentiality, integrity, availability, and replay protection. As a repetitive cycle is undesirable and makes the chain have a large number of collisions, an initial key k_0 that results in a repetitive cycle is to be considered weak according to the definition in Sect. 2.3. We call a key k_0 m-weak if the ρ-chain $k_0, k_1, \ldots, k_{m-1}$ has a repetitive cycle. A m-weak key is also m'-weak for all $m' > m$. ξ-, ω-, and π-chains do not have any repetitive cycles.

Number of Weak Keys. While it is infeasible to calculate the exact number of short repetitive cycles and weak keys for a specific hash function, the expected number of m-weak keys can be approximated based on the assumption that the hash function $H()$ is a random mapping of the $N = 2^n$ elements in the key space. As shown by Knuth [31], ρ-chains behave like a set of uniformly sampled keys in the sense that the collision probability is the same. The reason is that if the chain does not have a collision, we can model each successive k_i as randomly chosen from N. As the probability that the chain does not have a collision is the same as in the birthday attack, the probability that the chain has a collision is also the same, and we can use birthday attack formulas to calculate the collision probability. The probability of a collision among the first m keys in a ρ-chains is therefore

$$1 - \prod_{i=0}^{m-1} \frac{N-i}{N} \approx 1 - e^{-m^2/2N} \approx \frac{m^2}{2N} \ , \tag{12}$$

where the approximation $1 - e^{-m^2/2N}$ is valid when $m \ll N$ and the approximation $m^2/2N$ is valid when $m \ll N^{1/2}$. The expected fraction of m-weak keys in ρ-chains is therefore

$$\approx m^2/2N \ , \tag{13}$$

and the expected number of m-weak keys is $\approx m^2/2$. The expected fraction of keys k_0 leading to ξ- and ω-chains with at least one collision is also $\approx m^2/2N$. The expected number of keys k_i in a ρ-chain colliding with one or more other keys in the chain is

$$\approx m^3/2N \ . \tag{14}$$

π-chains do not have any collisions.

As stated in Sect. 2.3, it is considered highly desirable for a cryptographic function to possess no weak keys. If weak keys exist, the likelihood of encountering a weak key and compromising security should be very low. A reasonable requirement for λ-bit security is that the expected fraction Eq. (13) of weak keys is smaller than $2^{-\lambda}$, which implies that for ρ-chains

$$n \geq \lambda + 2\ell - 1 \ , \tag{15}$$

where 2^ℓ is that largest possible length of the key chain, i.e., $m \leq 2^\ell$. But note that this is not an attack complexity.

Impact on TLS 1.3 and Signal. TLS 1.3 currently defines two hash functions, SHA-256 and SHA-384 with n equal to 256 and 384, respectively. In Signal, n is always equal to 256. The expected number and fraction of m-weak keys in TLS 1.3 and Signal are presented in Table 2. If key update is used with ChaCha20 in TLS 1.3 the probability of a weak key is $m^2/2^{257}$. The other TLS 1.3 cipher suites listed in Table 1 fulfill Eq. (15) as long as $m \leq 2^{64.5}$.

Table 2. Expected number and fraction of m-weak keys when using ρ-chains in a key space of size $N = 2^n$

	$n = 256$		$n = 384$	
	Number	Fraction	Number	Fraction
2-weak	2^1	2^{-255}	2^1	2^{-383}
2^2-weak	2^3	2^{-253}	2^3	2^{-381}
2^4-weak	2^7	2^{-249}	2^7	2^{-377}
2^8-weak	2^{15}	2^{-241}	2^{15}	2^{-369}
2^{16}-weak	2^{31}	2^{-225}	2^{31}	2^{-353}
2^{32}-weak	2^{63}	2^{-193}	2^{63}	2^{-321}
2^{48}-weak	2^{95}	2^{-161}	2^{95}	2^{-289}
2^{64}-weak	2^{127}	2^{-129}	2^{127}	2^{-257}

3.3 The More Keys the Merrier Collisions in ρ-chains

Without key updates in TLS 1.3 there is a single traffic secret k_0 and therefore a single AEAD key K_0 per connection. With a single AEAD key, there is zero chance for key collision and replay inside the connection, and the best general key recovery attack is brute force. This attack has data and memory complexities $D = M = \mathcal{O}(1)$ and time complexity N.

Collisions in the Beginning of a ρ-chain. Key update was introduced in TLS 1.3 to meet confidentiality key usage limits and to achieve forward secrecy. While this is achieved, it is known that the large number of keys also decrease the security properties of the connection by introducing a risk of key collisions. As given by Eq. (12), the probability of a collision among the first $v \leq m \ll N^{1/2}$ keys $k_0, k_1, \ldots, k_{v-1}$ in a ρ-chain is

$$p_f \approx v^2/2N \ . \tag{16}$$

Assuming that $\approx \log_2 v$ bits of the first v plaintexts encrypted with a specific sequence number are known, an attacker can find a (key, nonce) collision with data and memory $D = M = v$ and expected work v using e.g., a hash table. False positives can be rejected with work v. The number of false positives can be lowered by using more bits of known plaintext. The time complexity of the attack is

$$T_f = v/p_f = 2N/v \ , \tag{17}$$

which has minimum value $2N/m$ when $v = m$. If AES-CBC is used (as is recommended in Signal) the prefixes of the plaintexts need to be identical. Partly identical plaintexts are likely to happen in practice even if the attacker cannot choose the plaintexts, see e.g., Sect. 3.4 of [47]. This attack is possible in ρ-, ξ-, and ω-chains. We will now show that a much more efficient novel collision attack is possible in ρ-chains.

Collisions in the End of a ρ-chain. As a cycle in a ρ-chain leads to collisions in the rest of the chain, the probability of a collision among the last v keys is significantly higher than $v^2/2N$. If any of the m keys are on a repetitive cycle of length less than or equal to $v - 1$, there will be a collision also in the last v keys. For $i \leq v - 1$, the probability that key k_i is the first key on a repetitive cycle of length $\leq v - 1$ is $\approx (1 - i^2/2N) \cdot i/N \approx i/N$. For $i \geq v - 1$, the probability that key k_i is the first key on a repetitive cycle of length $\leq v - 1$ is $\approx (1 - i^2/2N) \cdot (v - 1)/N \approx (v - 1)/N$. The total probability for a collision among the last v keys is

Theorem 1. *The probability of a collision among the last $v \leq m \ll N^{1/2}$ keys k_{m-v}, k_{m-v+1}, $\ldots$, k_{m-1} in a ρ-chain is*

$$p_e = \sum_{i=0}^{v-1} \frac{i}{N} + \sum_{i=v}^{m-1} \frac{v-1}{N} \approx \frac{m(v-1)}{N} \ . \tag{18}$$

The probability of collision among the last v keys is significantly higher than the probability of collision among the first v keys. Assuming that $\approx \log_2 v$ bits of the last v plaintexts encrypted with a specific sequence number are known, an attacker can find a (key, nonce) collision with data and memory $D = M = v$ and expected work v using e.g., a hash table. The time complexity of the attack is

$$T_e = \frac{v}{p_e} = \frac{v}{v-1}\frac{N}{m} \ . \tag{19}$$

The data and memory complexities can be minimized by choosing a small v without affecting the time complexity much. With a small $v \gg 1$ the time complexity in Eq. (19) is N/m, which is half of the previously best-known time complexity in Eq. (17) and the data and memory complexities are v instead of m. The security level is $\lambda = n - \ell$ and a strict requirement for λ-bit security with ρ-chains is that

$$n \geq \lambda + \ell \ , \tag{20}$$

where 2^ℓ is that largest possible length of the key chain, i.e., $m \leq 2^\ell$.

Impact on TLS 1.3 and Signal. As a collision leads to a repetitive cycle, all future keys will be collisions, and it is likely that $\approx m/2$ past keys are also collisions. If AES-GCM is used, the confidentiality and integrity of the application data protected with the colliding keys are lost. Even if the integrity is not lost in the sense that the attacker can perform impersonation or substitution forgeries, the replay protection is lost in TLS 1.3 as the attacker can replay old ciphertexts with the same sequence number. A replayed record likely breaks integrity and depending on the application broken integrity might affect availability and confidentiality. A collision in DTLS 1.3 and QUIC also very likely breaks replay protection as only the least significant 1–2 bits of the variable counting the number of key updates in used in the associated data.

Without key update, ChaCha20 in TLS 1.3 is believed to have 256-bit confidentiality even if 2^{96} encryption queries have been done [40]. If key update is used with $m \leq 2^\ell$, ChaCha20 only offer $256 - \ell$ bits of security against the above attack. The security of the other cipher suites are not affected as long as $m \leq 2^{128}$. The same attack apply to Signal where the chain key size n and message key size n_s are both always equal to 256.

3.4 Shrinking Key Space from Key Update

Let S_m be the set of m-iterate image points, i.e., the key space after the key chain has been iterated m times. The size of the key space is initially $|S_0| = N$ but shrinks with each iteration if a random mapping is used as described in [20]. An important property of ρ-chains that we will use in Sect. 3.6 is that the $m+1$-iterate image points is a subset of the m-iterate image points, i.e., $S_{m+1} \subseteq S_m$ [24]. This is not true for ξ- and ω- chains where S_{m+1} and S_m are typically almost disjoint. In π-chains $S_m = N$.

Key Space Size for a Single Chain. Flajolet and Odlyzko [20] prove that for a ρ-chain, the expected number of m-iterate image points in a random mapping of size N (i.e. the expected value of $|S_m|$) has the asymptotic form $E\big(|S_m|\big) = (1 - \tau_m)N$ where $\tau_0 = 0$ and $\tau_{i+1} = e^{\tau_i - 1}$ as $N \to \infty$. This iterative formula apply also to ξ- and ω-chains. This follows almost trivially from repeated use of the well-known fact that if s balls are thrown into t urns, then the expected number of empty urns are $e^{-s/t}$.

Theorem 2. *In ρ-, ξ-, and ω-chains the expected number of m-iterate image points has the asymptotic form*

$$E\big(|S_m|\big) = (1 - \tau_m)N \ \textit{ where } \ \tau_0 = 0 \ \textit{ and } \ \tau_{i+1} = e^{\tau_i - 1} \ \textit{ as } \ N \to \infty \ . \tag{21}$$

Flajolet and Odlyzko [20] do not describe for which values of m the iterative formula is valid for ρ-chains, but it is clearly not valid for $m \gg N^{1/2}$, as the expected number of ∞-iterate image points is $\sqrt{\pi/2} \cdot N^{1/2}$, at which point the key space only contains repetitive cycles and the random mapping works as a permutation. Our numerical simulations conducted on key spaces of size 2^{16} and 2^{24} indicates that Eq. (21) is a good approximation for $m \leq N^{1/2}$. The key spaces of ξ- and ω-chains shrinks until $S_m = 1$ (all the balls are in a single urn), which happens around $m = 2N$.

To our knowledge, no explicit formula is known for Eq. (21). The number of m-iterate image points for small values can be calculated numerically and

$$1 - \tau_m \approx 2/(m + 2) \ , \tag{22}$$

seems to be an excellent approximation. This agrees with the estimate in Section 4.3 in [24] but is a much better approximation for small m. Values of $(1 - \tau_m)$ and $\epsilon = \big|(1 - \tau_m) - 2/(m + 2)\big|$ for selected $m \leq 2^{24}$ are show in Table 3. The values were calculated with Julia v1.9 [28] using arbitrary precision floating point numbers. Based on the convincing "numerical evidence" in Table 3 we make the following conjecture.

Conjecture 1. The expected number of m-iterate image points in a ρ-, ξ-, and ω-chain using random mapping(s) of size N has the asymptotic form

$$E\big(|S_m|\big) = \frac{2}{m + 2} N \ \text{ as } \ N \to \infty \ . \tag{23}$$

When $m \gg 2$ this can be approximated as

$$2N/m \ . \tag{24}$$

Note that the shrinking key space does not increase the collision probability inside ρ-chains more than Eq. (18) as calculated in Section 3.3. The probability of a collision among the last v keys in a ρ-chain is $\approx mv/N$ and not $\approx v^2/2(2N/m) = mv^2/4N$ as one might think based on the shrinking key space. As seen in Sect. 3.5, the shrinking key space is unlikely to pose a security problem for ω-chains, given that the key space remains unpredictable for potential attackers.

Table 3. $E\big(|S_m|\big)/N = (1-\tau_m)$ and $\epsilon = \big|(1-\tau_m) - 2/(m+2)\big|$ after m iterations in ρ, ξ, and ω-chains.

m	$(1-\tau_m)$	ϵ	m	$(1-\tau_m)$	ϵ	m	$(1-\tau_m)$	ϵ
2	$2^{-1.0938}$	$2^{-4.9}$	2^9	$2^{-8.0109}$	$2^{-16.1}$	2^{17}	$2^{-16.0001}$	$2^{-31.1}$
2^2	$2^{-1.6800}$	$2^{-5.6}$	2^{10}	$2^{-9.0058}$	$2^{-17.9}$	2^{18}	$2^{-17.0000}$	$2^{-33.0}$
2^3	$2^{-2.4032}$	$2^{-6.5}$	2^{11}	$2^{-10.0031}$	$2^{-19.8}$	2^{19}	$2^{-18.0000}$	$2^{-34.9}$
2^4	$2^{-3.2306}$	$2^{-7.8}$	2^{12}	$2^{-11.0016}$	$2^{-21.6}$	2^{20}	$2^{-19.0000}$	$2^{-36.9}$
2^5	$2^{-4.1285}$	$2^{-9.2}$	2^{13}	$2^{-12.0008}$	$2^{-23.5}$	2^{21}	$2^{-20.0000}$	$2^{-38.8}$
2^6	$2^{-5.0704}$	$2^{-10.9}$	2^{14}	$2^{-13.0004}$	$2^{-25.4}$	2^{22}	$2^{-21.0000}$	$2^{-40.7}$
2^7	$2^{-6.0381}$	$2^{-12.6}$	2^{15}	$2^{-14.0002}$	$2^{-27.3}$	2^{23}	$2^{-22.0000}$	$2^{-42.6}$
2^8	$2^{-7.0204}$	$2^{-14.3}$	2^{16}	$2^{-15.0001}$	$2^{-29.2}$	2^{24}	$2^{-23.0000}$	$2^{-44.6}$

3.5 Key Space Size for the Set of All Chains

Let S^*_m be the set of m-iterate image points in the set of all possible chains, i.e., a multi-connection setting. The size of the key space is initially $|S^*_0| = N$ but shrinks with each iteration if a random mapping is used. For deterministic ρ- and ξ–chains, where the context and mapping do not depend on the connection, $S^*_m = S_m \approx 2N/m$. For π-chains, $S^*_m = S_m = N$. For the set of randomized ω-chain the expected number of m-iterate image points $E\big(|S^*_m|\big)$ decreases slowly. If the size of the universal hash function family is a, i.e., each iteration uses a random salt with $\log_2 a$ bits of entropy, the number of urns that is empty for all a hash functions is e^{-a}. If s balls are thrown into t urns, then the expected number of urns that is empty for all a hash functions is $e^{-as/t}$ and we get the following iterative formula for a ω-chain.

Theorem 3. *In ω-chains using a universal hash function family of size a, the expected number of m-iterate image points in the set of all possible chains has the asymptotic form*

$$E\big(|S^*_m|\big) = (1-\tau^*_m)N \quad \textit{where} \;\; \tau^*_0 = 0 \;\; \textit{and} \;\; \tau^*_{i+1} = e^{a(\tau^*_i - 1)} \;\; \textit{as} \;\; N \to \infty \ . \tag{25}$$

Numerical analysis of the iterative formula Eq. (25) indicates that for $a > 1$,

$$\tau^*_k \to c \;\text{ as }\; k \to \infty \ , \tag{26}$$

where $c > 0$ and that $c \to 1$ as $a \to \infty$. Solving the equation $c = e^{a(c-1)}$ gives

$$c = -W_0(-ae^{-a})/a \ , \tag{27}$$

where W_0 is the principal branch of the Lambert W-Function also called the omega function. This solution agrees very well with our numerical analysis in Table 4. For $a = 2$, $(1-\tau^*_k) \to 0.79681213\ldots$, for $a = 4$, $(1-\tau^*_k) \to 0.98017260\ldots$, and for $a = 16$, $(1-\tau^*_k) \to 0.99999989\ldots$. Even for very small hash function families the difference between c and 1 is negligible and can be ignored. Based on the convincing "numerical evidence" in Table 4 we make the following conjecture.

Table 4. $E\big(|S^*_m|\big)/N = (1 - \tau^*_m)$ and $\epsilon = \big|(1 - \tau^*_m) - 1 - W_0(-ae^{-a})/a\big|$ after m iterations in ω-chains.

$a = 2$			$a = 4$			$a = 16$		
m	$(1 - \tau^*_m)$	ϵ	m	$(1 - \tau^*_m)$	ϵ	m	$(1 - \tau^*_m)$	ϵ
2	0.82259667	$2^{-5.3}$	2	0.98029213	$2^{-13.0}$	2	0.99999989	$2^{-61.2}$
2^2	0.80092019	$2^{-7.9}$	2^2	0.98017335	$2^{-20.3}$	2^2	0.99999989	$2^{-99.4}$
2^3	0.79692341	$2^{-13.1}$	2^3	0.98017260	$2^{-35.0}$	2^3	0.99999989	$2^{-175.7}$
2^4	0.79681221	$2^{-23.5}$	2^4	0.98017260	$2^{-64.2}$	2^4	0.99999989	$2^{-328.4}$
2^5	0.79681213	$2^{-44.3}$	2^5	0.98017260	$2^{-122.7}$	2^5	0.99999989	$2^{-633.7}$
2^6	0.79681213	$2^{-85.9}$	2^6	0.98017260	$2^{-239.7}$	2^6	0.99999989	$2^{-1244.4}$

Conjecture 2. The expected number of k-iterate image points in the set of ω-chains using random mappings of size N and a hash function family of size $a \geq 2$ has the asymptotic form

$$E\big(|S^*_m|\big) = \left(1 + \frac{W_0\left(-ae^{-a}\right)}{a}\right) N \text{ as } m \to \infty \text{ and } E\big(|S^*_m|\big) = N \text{ as } a \to \infty \ .$$

Looking at the intersection of the images of all hash functions in the universal hash function family, it is easy to see that $S^*_{k+1} \subseteq S^*_k$.

Impact on TLS 1.3 and Signal. After m iterations of the key update mechanism in TLS 1.3, the predictable traffic secret key space has size $2^{n+1}/m$. After 2^{64} key updates, the key space is, for example, reduced by a factor 2^{63}. In most of the TLS 1.3 cipher suites listed in Table 1, $2^n \gg 2^{n_s}$ and $E\big(|S^*_m|\big) \geq n_s$ as long as $m \leq 2^{129}$. The exception is CHACHA20_POLY1305_SHA256 where $n = n_s = \lambda$. If key update is used, the traffic key space quickly becomes much smaller than the key space for the AEAD keys, which is a design flaw. The same result apply to Signal where $n = n_s = 256$. TLS 1.3 also derives a random IV to improve multi-key (multi-user) security [7]. A reasonable design principle is that the predictable iterated key space of the traffic secrets should be greater or equal to the AEAD key space and the IV space, i.e.,

$$E\big(|S^*_m|\big) \geq 2^{\lambda + n_{iv}} \ , \tag{28}$$

which implies that

$$n \geq \begin{cases} \lambda + n_{iv} + \ell - 1 & \text{for } \rho\text{- and } \xi\text{-chains} \ , \\ \lambda + n_{iv} & \text{for } \omega\text{- and } \pi\text{-chains} \ . \end{cases} \tag{29}$$

where 2^ℓ is that largest possible length of the key chain, i.e., $m \leq 2^\ell$. For cipher suites with $n - \lambda = 128$ and $n_{iv} = 96$ this is true as long as $m \leq 2^{33}$.

3.6 New TMTO Attacks on ρ- and ξ-chains

Preuß Mattsson [47] showed that the key update function in TLS 1.3 and the symmetric key ratchet in Signal can be modeled as non-additive synchronous stream ciphers [36], which means that the efficient Time Memory Tradeoff Attacks (TMTO) for stream ciphers such as Babbage-Golić [4,21] and Biryukov-Shamir [9] can be applied. In this section, we describe new TMTO attacks for ρ- and ξ-chains that significantly improve Babbage-Golić by making use of the shrinking state $E\big(|S^*_m|\big) \approx 2N/m$ and the fact that for ρ-chains, $S^*_{m+1} \subseteq S^*_m$.

In Babbage-Golić [4,9,21], the attacker tries to find one of the many internal states instead of the key. The attacker generates M random states k'_0, k'_1, …, k'_{M-1} from the total number of states N, calculates an output string y'_j for each state k'_j, and stores the pairs (k'_j, y'_j) ordered by y'_j. In the real-time phase the attacker collects D output strings $y_0, y_1, \ldots, y_i, \ldots, y_{D-1}$. By the birthday paradox the attacker can find a collision $y_i = y'_j$ and recover an inner state $k_i = k'_j$ in time $T = D = N/M$, memory M, and preprocessing time $P = M$, where $1 \leq T$. Example points on this tradeoff relation is $T = M = D = P = N^{1/2}$, as well as $T = D = N^{1/4}$ and $M = P = N^{3/4}$.

In a stream cipher where the states form a ρ- or ξ-chain several novel tradeoff attacks are possible. Instead of random states from the initial key space S^*_0, the attacker generates M random states from the set of m-iterate image points S^*_m. In the real-time phase the attacker collects D output strings calculated from states in S^*_m. If the states are uniformly distributed, Eq. (24) implies that the time complexity is

$$T = |S^*_m|/M = 2N/mM \ , \tag{30}$$

and not N/M as in Babbage-Golić's classical attack. In practice, the generated and collected states will follow a non-uniform distribution and the time complexity will be lower than $|S^*_m|/M$. This type of multi-connection attack on ρ-chains between two random sets sampled from S^*_m is known from [24]. We show that this attack also applies to ξ-chains, that the attack can be turned into a single-connection attack by lowering the probability of key recovery to less than 1, and that ξ-chains provide zero bits of security if the chain length is unbounded. We also describe two new attacks on ρ-chains where $S^*_{m+1} \subseteq S^*_m$, allowing us to exploit collisions between the end of a chain and a random set.

Babbage-Golić with Success Probability Less than One. An attacker can significantly lower the data and memory requirements by performing an attack where the probability of recovering a key is less than 1. The work done by the attacker is $D \leq N/M$ and the success probability is $\approx DM/N$. The time complexity is still $T = D/(DM/N) = N/M$. The complexities of the attack are time $T = N/M$, memory M, data D, and preprocessing time $P = M$, where $1 \leq D \leq T$.

Single- and Multi-connection Attacks on ξ-chains. The attacker generates M random states from S^*_m and collects D output strings calculated from states in S_m^*. The attacker can find a collision $y_i = y'_j$ and recover an inner state $k_i = k'_j$ with time complexity $T = 2N/mM$, memory M, data D, preprocessing time $P = mM$, where $1 \leq D \leq T$. For a single-connection attack, $D = 1$. The success probability is $\approx mMD/2N$. For a multi-connection attack with probability close to 1 it is required that $D = T$. Example points on this tradeoff relation are

$$T = M = \sqrt{2N/m} \text{ and } P = \sqrt{2mN} \ . \tag{31}$$

Choosing $m \approx N^{1/3}$ we get

$$T = M \approx N^{1/3} \text{ and } P \approx N^{2/3} \ , \tag{32}$$

and choosing $m \approx N$, we get

$$T = M = D \approx 1 \text{ and } P \approx N \ . \tag{33}$$

Single- and Multi-connection Attacks on ρ-chains. As $S_{m+1}^* \subseteq S_m^*$, the M random states can be the last M states from a single chain of length m. The D output strings are calculated from the last states in D chains of length m. To avoid non-negligible overlap between the D chains it is required that $m^2 D \leq N$ [23]. If $M \ll m$, the attacker can find a collision $y_i = y'_j$ and recover an inner state $k_i = k'_j$ with time complexity $T = 2N/mM$, memory M, data D, and preprocessing time $P = m$, where $1 \leq D \leq T$, $M \ll m$, and $m^2 D \leq N$. For a single-connection attack, $D = 1$. The success probability is $\approx mMD/2N$. For a multi-connection attack with probability close to 1 it is required that $D = T$. Example points on this tradeoff relation are

$$T = M = \sqrt{2N/m} \text{ and } P = m \ . \tag{34}$$

Choosing $m \approx N^{1/3}$ we get

$$T = M = P \approx N^{1/3} \ , \tag{35}$$

and choosing $m \approx N^{1/2}$ we get

$$T = M \approx N^{1/4} \text{ and } D \approx 1 \text{ and } P \approx N^{1/2} \ . \tag{36}$$

Single-connection Attack on ρ-chains with Higher Success Probability. As $S_{m+1}^* \subseteq S_m^*$, the D output strings can be calculated from the last D states from a single chain of length m, i.e., a single-connection attack. The M random states are the last states in M chains of length m. The attacker can find a collision $y_i = y'_j$ and recover an inner state $k_i = k'_j$ with time complexity $T = 2N/mM$, memory M, data D, and preprocessing time $P = mM$, where $1 \leq D \leq T$, $D \ll m$, and $m^2 M \leq N$. The success probability is $\approx mMD/2N$. For an attack with probability close to 1 it is required that $D = T$. Example points on this tradeoff relation are

$$T = M = \sqrt{2N/m} \text{ and } P = \sqrt{2mN} \ . \tag{37}$$

Choosing $m \approx N^{1/3}$, we get

$$T = M \approx N^{1/3} \text{ and } P \approx N^{2/3} \ . \tag{38}$$

Security Level and State Size. Based on Babbage-Golić [4,21] and Biryukov-Shamir [9], modern stream ciphers such as SNOW 5G [19] 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. For ρ- and ξ-chains this is not enough.

If m is unlimited, the predictable state size of ρ-chains eventually shrinks to $n/2$, the security level is $n/4$, and the state size in bits n should therefore be at least four times the security level λ. The predictable state size of ξ-chains eventually shrinks to 0 bits. To claim any security level for applications of ξ-chains, it is essential with strict limits for the maximum number of iterations $m < 2^\ell$. If m is limited to $m \leq 2^\ell \leq 2^{n/2}$, the single- and multi-connection security level of ρ- and ξ-chains are

$$\lambda \leq (n - \ell + 1)/2 \ , \tag{39}$$

and a requirement for λ-bit security is that

$$n \geq 2\lambda + \ell - 1 \ , \tag{40}$$

where 2^ℓ is the largest possible length of the key chain, i.e., $m \leq 2^\ell$. The single-connection attacks are also a single-key attack on the last key in the chain. The common practice is to disregard the preprocessing complexity when assessing the security level.

For ω- and π-chains, if m is limited to $m \leq 2^\ell \leq 2^{n/4}$, the single-connection security level against Biryukov-Shamir's attack [9] is $\lambda = 2(n - \ell)/3$ and a requirement for λ-bit security is that $n \geq 3\lambda/2 + \ell$. If $\ell \geq n/4$, the single-connection security is $n/2$. The multi-connection security is always $n/2$. The security and state size requirements as a function of state size 2^n and maximum chain length 2^ℓ for all types of chains are summarized in Tables 5 and 6. Note that the single-connection security against Hellman's TMTO attack [23,44] is $2n/3$ and that the multi-connection security against birthday attacks is $n/2$.

Table 5. Security against TMTO attacks as a function of state size 2^n and maximum chain length 2^ℓ. Constant terms have been removed. The attacks have varying preprocessing complexities. The single-connection attacks on ξ-chains and the single-connection attacks on ρ-chains where $\ell \geq n/3$ have success probabilities smaller than 1.

	Single-connection				Multi-connection		
	$\ell \leq \frac{n}{4}$	$\frac{n}{4} \leq \ell \leq \frac{n}{2}$	$\frac{n}{2} \leq \ell \leq n$	$n \leq \ell$	$\ell \leq \frac{n}{2}$	$\frac{n}{2} \leq \ell \leq n$	$n \leq \ell$
ρ	$(n-\ell)/2$	$(n-\ell)/2$	$n/4$	$n/4$	$(n-\ell)/2$	$n/4$	$n/4$
ξ	$(n-\ell)/2$	$(n-\ell)/2$	$(n-\ell)/2$	0	$(n-\ell)/2$	$(n-\ell)/2$	0
ω	$2(n-\ell)/3$	$n/2$	$n/2$	$n/2$	$n/2$	$n/2$	$n/2$
π	$2(n-\ell)/3$	$n/2$	$n/2$	$n/2$	$n/2$	$n/2$	$n/2$

Table 6. Minimum state size requirement as a function of security level λ and maximum chain length 2^{ℓ}. Constant terms have been removed.

	Single-connection				Multi-connection		
	$\ell \leq \frac{n}{4}$	$\frac{n}{4} \leq \ell \leq \frac{n}{2}$	$\frac{n}{2} \leq \ell \leq n$	$n \leq \ell$	$\ell \leq \frac{n}{2}$	$\frac{n}{2} \leq \ell \leq n$	$n \leq \ell$
ρ	$2\lambda + \ell$	$2\lambda + \ell$	4λ	4λ	$2\lambda + \ell$	4λ	4λ
ξ	$2\lambda + \ell$	$2\lambda + \ell$	$2\lambda + \ell$	∞	$2\lambda + \ell$	$2\lambda + \ell$	∞
ω	$3\lambda/2 + \ell$	2λ	2λ	2λ	2λ	2λ	2λ
π	$3\lambda/2 + \ell$	2λ	2λ	2λ	2λ	2λ	2λ

Impact on TLS 1.3 and Signal. The attacks significantly affects Signal and all of the TLS 1.3 cipher suites, which presumably were designed to provide $\min\big(n_s,\ 2(n_s + n_{iv})/3\big)$ bits of security against single-connection attacks and $(n_s + n_{iv})/2$ bits of security against multi-connection attacks. As the number of key updates is unbounded, the above attacks reveal that they only provide $n/4$ bits of security. Specifically, the AES-128 cipher suites yield ≈ 64 bits of security instead of 128 for single-connection and 112 for multi-connection, the AES-256 cipher suite provides ≈ 96 bits of security instead of 235 for single-connection and 176 for multi-connection, and the ChaCha20 cipher suite only delivers ≈ 64 bits of security instead of 235 for single-connection and 176 for multi-connection. For comparison, as of 2024, public supercomputers [62] and distributed systems [11] have capabilities of performing 2^{85} to 2^{94} operations per year. The attacks do not apply to TLS 1.3 when key updates are not used.

3.7 Comparison of Security Properties

A summary of security properties of different types of chains is presented in Table 7. For a fixed key space N, the security of ρ- and ξ-chains are inferior to ω-chains, which, in turn, are inferior to π-chains. For a fixed key space N and $m \leq N^{1/2}$, ρ-chains are inferior to ξ-chains.

ρ-chains always have a non-zero probability of repetitive cycles, irrespectively of the size of the key space. ρ-, ξ-, and ω-chains always have a non-zero probability of collisions, irrespectively of the size of the key space. Preventing short repetitive cycles has long been a fundamental objective in cryptographic design. ρ- and ξ-chains have predictable shrinking state spaces, and ρ-chains has the property that $S^*_{m+1} \subseteq S^*_m$. ρ-chains are notoriously hard to analyze and do not seem to have any significant benefits compared to ξ-chains as long as $m \leq N^{1/2}$. Several security properties of ρ-chains are still unknown, such as the collision probability between c chains of length v, and the number of m-iterate image points near $N^{1/2}$.

[1] See Appendix A. The value $4mv/N$ is an estimate based on simulations in key spaces with size 2^{16} and 2^{24}. The collision probability is at least $2mv/N$.

Table 7. Summary of expected values of several security metrics for ρ-, ξ-, ω-, and π-chains after m iterations where $v \leq m \ll N^{1/2}$.

	ρ	ξ	ω	π
Fraction of weak keys k_0 leading to repetitive cycles	$\frac{m^2}{2N}$	0	0	0
Fraction of keys k_i being part of a collision	$\frac{m^3}{4N}$	$\frac{m^2}{2N}$	$\frac{m^2}{2N}$	0
Collision probability for first v keys in one chain	$\frac{v^2}{2N}$	$\frac{v^2}{2N}$	$\frac{v^2}{2N}$	0
Collision probability for last v keys in one chain	$\frac{mv}{N}$	$\frac{v^2}{2N}$	$\frac{v^2}{2N}$	0
m-iterate key space size $\lvert S_m \rvert$ for a single chain	$\frac{2N}{m}$	$\frac{2N}{m}$	$\frac{2N}{m}$	N
m-iterate key space size $\lvert S_m^* \rvert$ for the set of all chains	$\frac{2N}{m}$	$\frac{2N}{m}$	N	N
∞-iterate key space size $\lvert S_\infty^* \rvert$ for the set of all chains	$N^{1/2}$	1	N	N
Collision probability between v random keys $\in S_m^*$ and v random keys $\in S_m^*$	$\frac{mv^2}{2N}$	$\frac{mv^2}{2N}$	$\frac{v^2}{N}$	$\frac{v^2}{N}$
Collision probability between v random keys $\in S_m^*$ and the last v keys in one chain	$\frac{mv^2}{2N}$	$\frac{v^2}{N}$	$\frac{v^2}{N}$	$\frac{v^2}{N}$
Max collision probability for $2v$ keys from two chains[1]	$\frac{4mv}{N}$	$\frac{2v^2}{N}$	$\frac{2v^2}{N}$	$\frac{v^2}{N}$
Collision probability for the last c keys k_{m-1} from c chains[1].	$\frac{mc^2}{4N}$	$\frac{mc^2}{4N}$	$\frac{c^2}{2N}$	$\frac{c^2}{2N}$

ξ-chains can easily be implemented with local counters that do not need to be transmitted. For a fixed key space N, ω-chains have better security than ξ-chains. However, ω-chains require that all parties agree on a random value to use in each iteration and cannot efficiently handle out-of-order messages, whereas ξ-chains can.

π-chains have the unique property of never having any collision. The cost of the permutation operation would be negligible in non-constrained devices using protocols where the key chain is not iterated each message. One-way permutations based on the discrete logarithm problem are however not quantum-resistant, which ρ-, ξ-, and ω-chains based on symmetrical primitives are. Rekeying with key chains based on one-way permutations also exhibits significantly inferior security properties compared to ephemeral key exchange, such as x25519 [34] and ML-KEM [42], which would have similar or lower computational cost.

While rekeying with ephemeral key exchange significantly improves security against key leakage and exfiltration, one-way key chains do not. In the worst-case scenario, assuming a rational attacker, the attacker exfiltrates the first key and rekeying with one-way key chains does not provide any benefit. One definition of perfect forward secrecy is that compromise of a single key will compromise only the data protected by the single key [13]; one-way key chains do not achieve this.

4 Analysis of Affected Security Protocols

In this section, we assess the impact of our results on various security protocols employing symmetric key ratchets. The protocols under scrutiny—TLS 1.3 [50], DTLS 1.3 [51], QUIC [26,61], Signal [56], Apple PQ3 [3], MLS [5], EDHOC [52], and OSCORE [25,53]—are widely used, recently standardized, or in the final stages of standardization. Our analysis is based on technical specifications; we have not examined implementations or deployments. For protocols that do not specify the intended security level λ for different algorithms, we assume that the intended security level corresponds to the key size n_s of the AEAD keys K_i. Unless the protocol defines limits on the length m of the key chain $k_0, k_1, \ldots, k_{m-1}$, we adopt the worst-case scenario, assuming an unlimited chain length.

4.1 TLS 1.3 Family of Protocols (TLS 1.3, DTLS 1.3, and QUIC)

The ρ-chains in the TLS 1.3 family of protocols (TLS 1.3, DTLS 1.3, and QUIC) exhibit very poor security properties, including weak keys, repetitive cycles, a high level of collisions at the end of chains, a predictable shrinking key space, and vulnerability to TMTO attacks that recover a large number of keys. In TLS 1.3 and QUIC the TMTO attack has complexity $T = M \approx N^{1/4}, P \approx N^{1/2}$ and $T = M = P \approx N^{1/3}$. Most of the TLS 1.3 cipher suites offer ≈ 64 bits of single- and multi-connection security against these pre-computation attacks. DTLS 1.3 offers $(n - 47)/2$ bits of security against the new TMTO attacks as the chain length is limited to 2^{48}. In practical applications, the security depends on the number of key updates m, but is always below the expected security level n_s when key update is used. EAP-TLS 1.3 [49], DTLS-SRTP [37], and DTLS/SCTP [63,64] are not affected as they do not use key update.

The simple attack described in Sect. 3.3 does not require any pre-computation and finds an AEAD (key, nonce) collision with complexity $T = N/m$ with negligible data and memory requirements $M = D = \mathcal{O}(1)$. The popular cipher suite CHACHA20_POLY1305_SHA256, which uses a 256-bit AEAD key, only provides $256 - \log_2 m$ bits of security against this collision attack, which compromises confidentiality, integrity, replay protection, and availability. When key update is used, CHACHA20_POLY1305_SHA256 has a traffic secret space of only $257 - \log_2 m$ bits even though $n_s + n_{iv} = 352$ bits are required to derive AEAD keys and IVs with the expected amount of entropy. The other cipher suites in Table 1 have less than the expected amount of entropy when $m > 2^{33}$.

As shown in [47], the procedures used to calculate AEAD limits specified in Appendix B of DTLS 1.3 [51] and QUIC [26], exhibit significant flaws both in theory and practical application. The results in Sect. 3, reveal additional inaccuracies, particularly when considering long chains. The advantages calculated in the procedures assume independent and uniformly distributed keys with full entropy. When key update is used, the keys are not independent, and they are not uniformly distributed if they are derived from a smaller key space. The single-key security of the last AEAD key is bounded by the multi-key security of all the traffic secrets. Another issue not mentioned in [47] is that the procedures base decisions on probabilities rather than attack complexities. This yields disparate outcomes for different algorithms unless the work done by the attacker is constant across algorithms. These findings reinforce the recommendation in [47] to deprecate the procedures.

The findings in Sects. 3.2 to 3.6 illustrate significant theoretical design flaws in TLS 1.3, DTLS 1.3, and QUIC that should be addressed in future revisions. ρ-chains are notoriously hard to analyze, and several properties remain unknown, which are not good security traits. The use of ρ-chains is more problematic in the TLS 1.3 family of protocols than in the Signal protocol, as TLS 1.3, DTLS 1.3, and QUIC can be expected to use much longer chains than the Signal protocol. The key update mechanism introduced in TLS 1.3 [50] should have been regarded as experimental new cryptography and negotiated with an optional extension.

A practical alternative in almost all use cases is to not use symmetric key update at all and instead set up new connections with ephemeral key exchange. As explained in [47], this gives much better security against key leakage/exfiltration than any key chains and aligns with zero trust principles. It also aligns with the stronger definition of perfect forward secrecy that compromise of a single key compromise only the data protected by that single key. Modern ephemeral key exchange algorithms like x25519 [34] are very fast and have small message overhead. The public keys are 32 bytes long and the cryptographic operations take 53 μs per endpoint on a single core AMD Ryzen 5 5560U [17], a mobile CPU from 2021. Ephemeral key exchange with the quantum-resistant algorithm ML-KEM [42] that NIST will standardize is even faster. For the non-standardized Kyber512 version of ML-KEM the cryptographic operations take 12 μs for the client and 8 μs for the server [16] on the same CPU. In both algorithms, key generation can be pre-computed, reducing the time required for real-time cryptographic operations to 27 μs per endpoint for x25519 and to 6 μs for the client and 8 μs for the server when ML-KEM-512 is utilized.

TLS 1.3 should clearly state the intended security levels for different cipher suites. The entropy issue with CHACHA20_POLY1305_SHA256 must be addressed irrespectively of rekeying by increasing the size of the traffic secrets and/or standardizing a new cipher suite. The replay attack described in Sect. 3.3 could have been prevented by including the epoch in the AEAD key derivation, nonce, or additional authenticated data. However, the preferred solution is to abstain from using ρ-chains altogether. The existence of weak keys and repetitive cycles can only be cured by removing the current ρ-chain and replacing it with something better.

We recommend current implementations and deployments of TLS 1.3, DTLS 1.3, and QUIC to not use key update and, instead, establish new connections using ephemeral key exchange when rekeying is needed. We recommend implementations to rekey with ephemeral key exchange every $2^{22.5}$ full-size records (2^{14} bytes) to align with current best practice and requirements recommending rekeying with ephemeral key exchange at least every hour and every 1–100 GB of data [2, 35, 55]. This approach offers significantly enhanced security against key leakage/exfiltration and the overhead of performing one ephemeral key exchange every $2^{22.5}$ messages is negligible. A mechanism for ephemeral key exchange within the connection could likely be done with smaller message sizes than resumption. We stress that the ephemeral keys should only be used once. As explained in the paper "Measuring the Security Harm of TLS Crypto Shortcuts" [57], reuse of key shares is a major practical security problem. We recommend TLS 1.3, DTLS 1.3, and QUIC to deprecate the current key update mechanism, which generates ρ-chains with insufficient state and without appropriate limitations on chain length. We also recommend all implementations and deployments to disable reuse of key shares, and for IETF to forbid such reuse.

Expanded Chain States. Increasing the chain state size can be achieved by using larger traffic secrets. Currently, the TLS 1.3 specification [50] states that the size of the traffic secrets is equal to the output size of the hash function. HKDF-Expand [33], used in TLS, can utilize up to block size bytes of entropy from the key. Changing the size of the traffic secrets to the block size would significantly increase the size of the chain space n. The output size and block size of some SHA-2 functions [41] are shown in Table 8. Modern KDF algorithms like KMAC [27] have less restrictions.

Table 8. Output size and block size of some SHA-2 hash functions.

Hash function	Output size	Block size
SHA-256	32	64
SHA-384	48	128
SHA-512	64	128

Improved Key Chains. The current key chain can be transformed into a ξ-chain by including a counter in the key derivation context, as is done in MLS [5]. DTLS 1.3 [51] already has such a counter, known as the epoch. Alternatively, the chain can be converted into an ω-chain by including a random number in the key update message and including it in the key derivation context as done in OSCORE [25]. TLS 1.3 [50] and DTLS 1.3 have dedicated key update messages while QUIC [26, 61] indicates key update with a bit in the packet header. As explained in Sect. 3.5, even a 1-bit random salt is enough to make sure that $|S_m^*| \approx N$. Introducing a π-chain built on elliptic curve cryptography would be more complex and would not be useful together with the quantum-resistant algorithms ML-KEM [42] and ML-DSA [43] that TLS 1.3 will soon introduce. We are not aware of any quantum-resistant one-way permutations.

If a symmetric key update is needed in certain use cases and TLS 1.3 decide to continue using symmetric key chains, we recommend imposing very small, strict limits ℓ on the length of ξ- or ω-chains. The state sizes n need to be increased so that the requirements outlined in Table 6 are fulfilled. Alternatively, the stated security levels can be adopted to fulfill the requirements outlined in Table 5. Note that symmetric rekeying should not be seen as a replacement for periodic rekeying with ephemeral key exchange in long-lived connections.

4.2 Signal Protocol

The Signal protocol [56] share many of the problems with the TLS family of protocols. The ρ-chains in Signal exhibit very poor security properties, including weak keys, repetitive cycles, a high level of collisions at the end of chains, a predictable shrinking key space, and vulnerability to TMTO attacks with complexity $T = M \approx N^{1/4}, P \approx N^{1/2}$ and $T = M = P \approx N^{1/3}$ that recover a large number of keys. The Signal protocol [56] offer ≈ 64 bits of single- and multi-connection security against these pre-computation attacks. The Signal protocol recommends encryption with AES-256 but only provides $256 - \log_2 m$ bits of security against the collision attack described in Sect. 3.3, which does not require any pre-computation and finds an AEAD (key, nonce) collision with complexity $T = N/m$ with negligible data and memory requirements $M = D = \mathcal{O}(1)$. The chain key space is only $257 - \log_2 m$ bits, which is smaller than the 256-bit message key and much less that the 384 bits in the AES-256 encryption key and the random IV. The findings in Sects. 3.2 to 3.6 illustrate significant theoretical design flaws in Signal that should be addressed in future revisions. ρ-chains are notoriously hard to analyze, and several properties remain unknown, which are not good security traits. The use of ρ-chains is slightly less problematic in the Signal protocol than in TLS 1.3, DTLS 1.3, and QUIC as the chains can be expected to be shorter. However, as found by [47], an attacker blocking messages in one direction can indefinitely increase the length of the chains.

Signal should clearly state the intended security levels. The entropy issue can only be addressed by increasing the size of the chain keys. The existence of weak keys and repetitive cycles can only be cured by removing the current ρ-chain and replacing it with something better. We recommend the Signal protocol to convert the key chain into an ξ-chain by including a counter in the key derivation context, as is done in MLS [5]. An ω-chain would cause problems for out-of-order messages. We recommend imposing very small, strict limits ℓ on the chain length. The chain key size n needs to be increased so that the requirements outlined in Table 6 are fulfilled. Alternatively, the stated security levels can be adopted to fulfill the requirements outlined in Table 5. To increase security against multi-key attacks we recommend Signal to increase the size of the message key so that it is at least as big as the encryption key and the IV.

4.3 Apple PQ3 Protocol for iMessage.

Background. Apple PQ3 [3,18,58] is a proprietary post-quantum cryptographic protocol for end-to-end secure messaging. Apple intends to deploy PQ3 for all iMessage conversations during 2024. Apple PQ3 is similar to Signal but has three ratchets: a symmetric ratchet, an ECDH ratchet using P-256, and a post-quantum ratchet using ML-KEM-1024 [42]. The symmetric ratchet uses 256-bit session keys k_i and HKDF-SHA384. For every message, PQ3 derives a 256-bit per-message encryption key K_i from the current session key

$$\begin{aligned} K_i &= \mathrm{KDF}(k_i, label_1, \texttt{""}, n_s) \ , \\ k_{i+1} &= \mathrm{KDF}(k_i, label_2, \texttt{""}, n) \ . \end{aligned} \tag{41}$$

The padded payload is encrypted with AES-CTR using a 256-bit encryption key and IV, both derived from the 256-bit message key. The amount of randomness in the IV has not been made public. PQ3 does not impose any limits ℓ on the chain length m and does not explicitly state the intended security level λ, but the length of the encryption key can typically be seen as the intended security level. Therefore $n = n_s = \lambda = 256$.

Analysis. The analysis of Signal applies to Apple PQ3 as well. It is commendable that PQ3 mandates message padding. Apple claims that PQ3 has the strongest security properties of any at-scale messaging protocol in the world, providing extensive defenses against even highly sophisticated quantum attacks. Our analysis shows that PQ3 provides 64-bit security against classical attackers. iMessage privacy and end-to-end encryption have been questionable in the past, with credible reports suggesting that law enforcement can request backups of a target's device, including actual messages sent and received in iMessage if they're backed up in the iCloud [59]. PQ3 does not address this issue.

Proprietary security protocols pose numerous significant problems. They carry a higher risk of both intentional and unintentional security vulnerabilities. Moreover, security vulnerabilities in proprietary security protocols are less likely to be discovered, as they hinder security researchers from examining and analyzing specifications and interactions between them. Open access to security specifications is crucial, as history has repeatedly shown that lack of analysis often leads to significant weaknesses. Proprietary and paywalled security protocols should be viewed as a cybersecurity risk. While Apple's CEO Tim Cook emphasizes that privacy is a "basic human right," we argue that open-access security specifications are also a basic human right.

4.4 The Messaging Layer Security (MLS) Protocol

Background. The Messaging Layer Security (MLS) Protocol [5] is a standardized protocol for secure group messaging developed by the Internet Engineering Task Force (IETF). MLS utilizes a ratchet tree structure [15] for the effective management of cryptographic keys within a group. It is designed to support large groups with dynamic memberships, ensuring secure communication even as members join or leave the group. MLS have been deployed at scale to protect sensitive real-time conversations in Cisco Webex and many other applications are planning to transition to MLS.

As outlined in Section 9 of [5], each sender maintains two symmetric key ratchets for every other member in the group—one for handshake messages and another for application messages. The ratchets work very similar to the ratchets in Signal with the difference that a counter is used as context. Each message i is encrypted using a ratchet key K_i and a ratchet nonce IV_i, which are used only once. Before transmitting each message, the ratchet secret, key, and nonce undergo updates using a symmetric-key ratchet, as outlined in Section 9.1 of [5]. The ratchet key K_i, the ratchet nonce IV_i, and the next ratchet secret k_{i+1} are computed using a Key Derivation Function (KDF) based on HKDF-Expand [33] as

$$\begin{aligned} K_i &= \mathrm{KDF}(k_i, \text{"key"}, i, n_s) \ , \\ IV_i &= \mathrm{KDF}(k_i, \text{"nonce"}, i, n_{iv}) \ , \\ k_{i+1} &= \mathrm{KDF}(k_i, \text{"secret"}, i, n) \ , \end{aligned} \tag{42}$$

where n is the size of the ratchet secret, n_s is the size of the ratchet key, and n_{iv} is the size of the ratchet nonce. MLS calls each iteration of the ratchet for a generation. MLS impose that $m \leq 2^{32}$ by specifying that the generation is encoded as an `uint32`. Table 9 provides a list of the KDF and AEAD algorithms in the initial MLS cipher suites. MLS specifies the intended security level λ for all cipher suites.

Table 9. The KDF and AEAD algorithms in the initial MLS cipher suites. λ is the intended security level.

Cipher suite	λ	n	n_s	n_{iv}	$n - \lambda$	n/λ
AES128GCM_SHA256	128	256	128	96	128	2.0
CHACHA20POLY1305_SHA256	128	256	256	96	128	2.0
AES256GCM_SHA512	256	512	256	96	256	2.0
CHACHA20POLY1305_SHA512	256	512	256	96	256	2.0
AES256GCM_SHA384	256	384	256	96	128	1.5

Analysis. As MLS limits the length of the chain length to $\ell = 32$, the ξ-chains in MLS only have benefits compared to ρ-chains. The ξ-chains do not exhibit any weak keys or repetitive cycles, and the single-connection TMTO attacks typically have success probability less than one, which may be less appealing to real-world attackers compared to attacks with a success probability close to one. It is commendable that MLS specifies a security level λ for every cipher suite.

However, ξ-chains have a predictable shrinking key space and only offer $(n - 31)/2$ bits of security against the new TMTO attacks described in Sect. 3.6. The attacks lower the single-connection security of AES128GCM_SHA256 and CHACHA20POLY1305_SHA256 to 112 bits and the single-connection security of AES256GCM_SHA384 to 176 bits. The new attacks do not lower the single-connection security of AES256GCM_SHA512 and CHACHA20POLY1305_-SHA512, which offer 234 bits of security against Hellman's TMTO attack [23,44].

The sizes n of the ratchet secrets should be increased so that the requirements outlined in Table 6 are fulfilled. Alternatively, the stated security levels can be adopted to fulfill the requirements outlined in Table 5. MLS uses HKDF-Expand and the size of the ratchet secrets is the values Nh from Hybrid Public Key Encryption (HPKE) [6]. Currently, all values are equal to the output sizes of the hash function, even though it's not explicitly stated as a rule. Similar to the approach described in Sect. 4.1 for TLS, MLS could expand the size of the ratchet secrets to the block size of the hash function.

4.5 Key Update for OSCORE (KUDOS)

Background. The Key Update for OSCORE (KUDOS) [25] is an extension to the Object Security for Constrained RESTful Environments (OSCORE) protocol [53]. KUDOS is currently undergoing standardization by the IETF. In OSCORE, the keying material consists of a master secret k and a master salt r. KUDOS aims to provide efficient symmetric rekeying with forward secrecy. It differs from the TLS 1.3 key update by also refreshing the master salt, and each iteration introduces fresh randomness. In KUDOS, both parties provide random numbers $N1$ and $N2$, each ranging from 8 to 128 bits in length, with the recommended size being 64 bits. The next master secret k_{i+1} and salt r_{i+1} are computed using a Key Derivation Function (KDF) based on HKDF-Expand [33] as

$$\begin{aligned} r_{i+1} &= N1 || N2 \ , \\ k_{i+1} &= \mathrm{KDF}(k_i, label1, r_{i+1}, n_k) \ , \end{aligned} \tag{43}$$

where n_k is the size of the master secret. From the master secret k_i, the AEAD key K_i and the initialization vector IV_i are derived as

$$\begin{aligned} K_i &= \mathrm{KDF}(k_i, label2, r_i, n_s) \ , \\ IV_i &= \mathrm{KDF}(k_i, label3, r_i, n_{iv}) \ . \end{aligned} \tag{44}$$

KUDOS does not enforce any limits ℓ on the chain length m and does not make any recommendations regarding key lengths, salt lengths, KDF algorithms, and AEAD algorithms beyond what is stipulated by the OSCORE specification [53]. The OSCORE specification leaves the determination of the sizes n_k of the master secret and n_r of the master salt to the application and does not explicitly state the intended security level. The mandatory-to-implement algorithm is AES-128 with $n_s = 128$ bits. In the provided examples [53], the size n_k of the master secret k is 128 bits, with or without a salt.

Analysis. The utilization of random salts r_i in KUDOS implies that the chain of master secrets forms an ω-chain, which offers relatively good security properties. However, despite the unpredictable key space in ω-chains, KUDOS remains vulnerable to TMTO attacks as discussed in [47]. A successful TMTO attack recovers half of the keys in the connection. As the master salt is included in the derivation of the AEAD key and IV, the state space in KUDOS is $n = n_k + n_r$. The size n should be configured to fulfill the requirements outlined in Table 6. Alternatively, the stated security levels can be adopted to fulfill the requirements outlined in Table 5. If the chain length remains unrestricted, the state space n should be greater or equal to twice the security level λ.

KUDOS and OSCORE give a lot of freedom and responsibility to the application but does not give any guidance. KUDOS should provide guidance on the relation between the chain state size n, security level λ, and maximum chain length ℓ. OSCORE [53] should provide guidance on the length of the master key and master salt. Even without KUDOS, the multi-key security is determined by $\min(n + n_r,\ n_s + n_{iv})/2$. Assuming that $\lambda = n_s$ and that k and r are uniformly random, OSCORE should require that $n \geq n_s$ and recommend that $n_k + n_r \geq n_s + n_{iv}$. Additionally, OSCORE should also describe the single- and multi-key security levels if these conditions are not satisfied.

While an ω-chain may offer acceptable security in the short term, as explained in Section 5.2 of [47], symmetric key exchange has significantly worse security properties than ephemeral Diffie-Hellman. We strongly recommend the KUDOS document to recommend periodic rekeying with ECDHE based on time and data using e.g., EDHOC. As explained in [47], the security advantages of employing periodic ephemeral key exchange are considerable, particularly in attack scenarios like side-channel attacks on Internet of Things (IoT) devices, where physical proximity is mandated. Current best practice and requirements for non-constrained implementations is to rekey with ephemeral key exchange at least every hour and every 1–100 GB of data [2, 35, 55]. As suggested in [47], constrained implementations should also mandate periodic rekeying with ephemeral Diffie-Hellman but could have a maximum period of 1 day, 1 week, or 1 month depending on how constrained the device and the radio is. Symmetric rekeying should not be seen as a replacement for periodic rekeying with ephemeral key exchange.

4.6 Ephemeral Diffie-Hellman Over COSE (EDHOC)

Background. Ephemeral Diffie-Hellman Over COSE (EDHOC) [52] is a Lightweight Authenticated Key Exchange (LAKE) currently undergoing standardization by the Internet Engineering Task Force (IETF). EDHOC has similar security properties as TLS 1.3 but with message sizes potentially less than 1/7 of a DTLS 1.3 handshake [48]. EDHOC has already been deployed in industry applications and many other deployments are planned. Appendix H of the EDHOC specification [52] specifies an optional key update function that can be used to update the key PRK_out k. It differs from the TLS 1.3 key update and the Signal symmetric ratchet, as it takes an application provided context as input

$$k_{i+1} = \mathrm{KDF}(k_i, \mathit{label}, \mathit{context}, n)\ . \tag{45}$$

The EDHOC specification [52] states that the context, e.g., can be a counter, a pseudorandom number, or a hash, but does not provide any recommendations.

It is mentioned that the context can be utilized to bind k_{i+1} to the event that triggered the key update. EDHOC does not impose any limits ℓ on the chain length m and does not explicitly state the intended security level λ. Table 10 provides a list of the EDHOC hash algorithm and Application AEAD algorithms in the initial cipher suites.

Table 10. The EDHOC hash algorithm and Application AEAD algorithm in the initial EDHOC cipher suites.

Cipher suite	n	$\lambda = n_s$	n_{iv}	$n - \lambda$	n/λ
AES-CCM-16-64-128, SHA-256	256	128	104	128	2.0
A128GCM, SHA-256	256	128	96	128	2.0
A256GCM, SHA-384	384	256	96	128	1.5
ChaCha20/Poly1305, SHA-256	256	256	96	0	1.0
ChaCha20/Poly1305, SHAKE256	512	256	96	256	2.0

As known deployments and protocols like Authentication and Authorization for Constrained Environments (ACE) [54] were planning to instead use KUDOS [25] together with EDHOC, the IETF LAKE working group discussed removing EDHOC key update from the specification. However, it was decided to make the mechanism optional and relocate it to an appendix instead.

Analysis. With a fixed context, the EDHOC key update produces a ρ-chain with all its associated bad properties. We strongly recommend EDHOC to mandate that the input to the context includes a nonce (number used once, e.g., a counter or a random number) to guarantee that there are no weak keys, repetitive cycles, or high level of collisions at the end of chains. Using a counter produces a ξ-chain and using a random number produces a ω-chain. EDHOC with key update is vulnerable to the TMTO attacks as discussed in Sect. 3.6. A successful TMTO attack recovers half of the keys in the connection.

It should be mentioned in the EDHOC specification that the cipher suite combining ChaCha20 and SHA-256 does not give 256-bit security when used with key update as $n = 256$. When EDHOC is used with OSCORE, the random IV does not increase protection against multi-key attacks as the 256-bit AEAD key and 96-bit IV is derived from at most 256-bits of entropy. When EDHOC is used with ρ- or ξ-chains, the entropy is only $257 - \log_2 m$ bits. The cipher suite combining ChaCha20 and SHAKE256 gives 256-bit security for ω-chains but might not be suitable for constrained devices only having support for SHA-256.

EDHOC key update gives a lot of freedom and responsibility to the application but does not give any guidance. EDHOC should clearly state the intended security levels. If EDHOC key update allows ρ- or ξ-chains, it is essential with strict limits ℓ on the chain length m. The size n of the PRK_out needs to be increased so that the requirements outlined in Table 6 are fulfilled. Alternatively, the stated security levels can be adopted to fulfill the requirements outlined in Table 5. EDHOC should provide guidance on the relation between type of context and the security properties of the resulting key chain. EDHOC should also provide guidance on the relation between the chain state size n, security level λ, and maximum chain length ℓ. Just like TLS and MLS, the size n is the output size of the hash function, and using the approach described in Sect. 4.1 for TLS, EDHOC could expand the size of the keys to the block size when hen HKDF-Expand is used and unlimited when KMAC is used.

If used for long-term connections, EDHOC should mandate periodic rekeying with ECDHE based on time and data. As explained in [47], the security advantages of employing periodic ephemeral key exchange are considerable, particularly in attack scenarios like side-channel attacks on Internet of Things (IoT) devices, where physical proximity is mandated. Current best practice and requirements for non-constrained implementations is to rekey with ephemeral key exchange at least every hour and every 1–100 GB of data [2,35,55]. As suggested in [47], constrained implementations should also mandate periodic rekeying with ECDHE but could have a maximum period of 1 day, 1 week, or 1 month depending on how constrained the device and the radio is.

5 Conclusions, Recommendations, and Future Work

The findings and attacks presented in this paper highlight significant theoretical flaws in many crucial security protocols that utilize symmetrical key ratchets, resulting in significantly lower security levels than anticipated based on the AEAD key sizes. We strongly advise against the use of ρ-chains. We recommend existing security protocols to phase out ρ-chains and replace them with better alternatives. Applications of key chains should enforce strict limits on the chain length. All stream ciphers we are aware of impose restrictions on the number of times the state update function can be iterated. One-way key chains have important use cases for messaging applications, group communication, and constrained environments. The use in general non-constrained two-party protocols is questionable and goes against established best practices for long-lived connections [2,35,55]. We stress that one-way key chains and ratchets should not be seen as general replacements for periodic rekeying with ephemeral key exchange in long-lived connections.

Based on the analysis, we recommend TLS 1.3, DTLS 1.3, and QUIC to:

- Deprecate the key update mechanism producing ρ-chains.
- Recommend ephemeral key exchange for all rekeying.
- Mandate periodic ephemeral key exchange based on time and data.
- Forbid reuse of key shares.
- Introduce a new key update mechanism producing ξ- or ω-chains.
- Increase the state size (traffic secret) of the chains.
- Introduce strict limits on the chain length.
- Clearly state intended security levels aligning with Table 5.

Based on the analysis, we recommend Signal and Apple PQ3 to:

- Deprecate the symmetric ratchet producing ρ-chains.
- Introduce a new symmetric ratchet producing ξ-chains.
- Increase the state size (chain key, session key) of the chains.
- Introduce strict limits on the chain length.
- Increase the size of the message key.
- Mandate rekeying with an asymmetric ratchet based on time and data.
- Clearly state intended security levels aligning with Table 5.

Based on the analysis we recommend MLS to:

- Increase the state size (ratchet secret) of the chains.
- Align security levels with Table 5.

Based on the analysis, we recommend OSCORE and KUDOS to:

- Introduce strict limits on the chain length.
- Provide guidance on the relation between the chain state size n, security level λ, and maximum chain length ℓ.
- Provide guidance on the relation between the length of the master key, the length of the master salt, and security levels.
- Introduce strict minimum requirements for the security level.
- Recommend frequent rekeying with ECDHE based on time and data.

Based on the analysis, we recommend EDHOC to:

- Recommend frequent rekeying with ECDHE based on time and data.
- Clearly state intended security levels.
- Provide guidance on the relation between the type of context and the security properties of the resulting key chain.
- Mandate that the key update context includes a counter or a random number.
- Provide guidance on the relation between the chain state size n, security level λ, and maximum chain length ℓ.
- Introduce strict limits on the maximum chain length.

Suggested future work:

- Evaluate the use of key chains in protocol implementations and deployments. Differences compared to specifications are often significant.
- Assess how different alternatives for rekeying (ξ-, ω-, π-chain, ephemeral key exchange inside the connection, or during resumption) impact performance.
- Investigate how often deployments perform rekeying and explore if an active attacker can influence the frequency.
- Explore methods to construct quantum-resistant one-way permutations using, for example, lattice-, code-, or isogeny-based cryptography.
- Explore the new TMTO attacks in Sect. 3.6 and propose new techniques, enhancements, and optimizations.
- Explore ρ-chains, derive formulas for the collision probability between c chains of length v and determine the number of m-iterate image points near $N^{1/2}$.

Acknowledgements

The authors would like to thank Patrik Ekdahl, Loïc Ferreira, Alexander Maximov, Erik Thormarker, Göran Selander, and Ben Smeets for their helpful comments and suggestions. The introduction and the descriptions of Signal, TLS 1.3, and Babbage-Golić draw significant inspiration from the work of [47], which has been used with proper permission and attribution.

References

1. 3GPP: Security architecture and procedures for 5g system (jan 2024), https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169
2. Agence nationale de la sécurité des systèmes d'information: Recommendations for securing networks with ipsec (August 2015), https://www.ssi.gouv.fr/uploads/2015/09/NT_IPsec_EN.pdf
3. Apple: imessage with pq3: The new state of the art in quantum-secure messaging at scale, https://security.apple.com/blog/imessage-pq3/
4. Babbage, S.: Improved "exhaustive search" attacks on stream ciphers. In: European Convention on Security and Detection, 1995. pp. 161–166 (1995). https://doi.org/10.1049/cp:19950490
5. Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., Cohn-Gordon, K.: The Messaging Layer Security (MLS) Protocol. RFC 9420 (Jul 2023). https://doi.org/10.17487/RFC9420
6. Barnes, R., Bhargavan, K., Lipp, B., Wood, C.A.: Hybrid Public Key Encryption. RFC 9180 (Feb 2022). https://doi.org/10.17487/RFC9180, https://www.rfc-editor.org/info/rfc9180
7. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 247–276. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53018-4_10
8. Bienstock, A., Fairoze, J., Garg, S., Mukherjee, P., Raghuraman, S.: A more complete analysis of the signal double ratchet algorithm. Cryptology ePrint Archive, Report 2022/355 (2022), https://eprint.iacr.org/2022/355
9. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto, T. (ed.) Advances in Cryptology – ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976, pp. 1–13. Springer, Heidelberg, Germany, Kyoto, Japan (Dec 3–7, 2000). https://doi.org/10.1007/3-540-44448-3_1
10. Bishop, M.: HTTP/3. RFC 9114 (Jun 2022). https://doi.org/10.17487/RFC9114, https://www.rfc-editor.org/info/rfc9114
11. Blockchain.com: Total hash rate (th/s), https://www.blockchain.com/explorer/charts/hash-rate
12. Bradford, P., Gavrylyako, O.: Hash chains with diminishing ranges for sensors. In: Workshops on Mobile and Wireless Networking/High Performance Scientific, Engineering Computing/Network Design and Architecture/Optical Networks Control and Management/Ad Hoc and Sensor Networks/Compil. pp. 77–83 (2004), https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.1221

13. Carrel, D., Harkins, D.: The Internet Key Exchange (IKE). RFC 2409 (Nov 1998). https://doi.org/10.17487/RFC2409, https://www.rfc-editor.org/info/rfc2409
14. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. Cryptology ePrint Archive, Report 2016/1013 (2016), https://eprint.iacr.org/2016/1013
15. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018: 25th Conference on Computer and Communications Security. pp. 1802–1819. ACM Press, Toronto, ON, Canada (Oct 15–19, 2018). https://doi.org/10.1145/3243734.3243747
16. eBACS: ECRYPT Benchmarking of Cryptographic Systems: Measurements of key-encapsulation mechanisms, https://bench.cr.yp.to/results-kem.html
17. eBACS: ECRYPT Benchmarking of Cryptographic Systems: Measurements of public-key diffie–hellman secret-sharing systems, https://bench.cr.yp.to/results-dh.html
18. David Basin, Felix Linker, R.S.: A formal analysis of the imessage pq3 messaging protocol, https://security.apple.com/assets/files/A_Formal_Analysis_of_the_iMessage_PQ3_Messaging_Protocol_Basin_et_al.pdf
19. Ekdahl, P., Johansson, T., Maximov, A., Yang, J.: SNOW-vi: an extreme performance variant of SNOW-V for lower grade CPUs. Cryptology ePrint Archive, Report 2021/236 (2021), https://eprint.iacr.org/2021/236
20. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology – EUROCRYPT'89. Lecture Notes in Computer Science, vol. 434, pp. 329–354. Springer, Heidelberg, Germany, Houthalen, Belgium (Apr 10–13, 1990). https://doi.org/10.1007/3-540-46885-4_34
21. Golic, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 239–255. Springer, Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997). https://doi.org/10.1007/3-540-69053-0_17
22. Goucher, A.P.: One-way permutations, https://cp4space.hatsya.com/2021/07/05/one-way-permutations/
23. Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory **26**(4), 401–406 (1980), https://ee.stanford.edu/~hellman/publications/36.pdf
24. Hong, J., Kim, W.H.: TMD-Tradeoff and state entropy loss considerations of streamcipher MICKEY. Cryptology ePrint Archive, Report 2005/257 (2005), https://eprint.iacr.org/2005/257
25. Höglund, R., Tiloca, M.: Key Update for OSCORE (KUDOS). Internet-Draft draft-ietf-core-oscore-key-update-06, Internet Engineering Task Force (Oct 2023), https://datatracker.ietf.org/doc/draft-ietf-core-oscore-key-update/06/, work in Progress
26. Iyengar, J., Thomson, M.: QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000 (May 2021). https://doi.org/10.17487/RFC9000
27. John Kelsey, Shu-jen Chang, R.P.: Sha-3 derived functions: cshake, kmac, tuplehash and parallelhash (dec 2016). https://doi.org/10.6028/NIST.SP.800-185
28. Julia Project: The julia programming language, https://julialang.org/
29. Kaliski Jr., B.S.: One-way permutations on elliptic curves. Journal of Cryptology **3**(3), 187–199 (Jan 1991). https://doi.org/10.1007/BF00196911
30. Kim, J.H., Montenegro, R., Peres, Y., Tetali, P.: A birthday paradox for markov chains with an optimal bound for collision in the pollard rho algorithm for discrete

logarithm. The Annals of Applied Probability **20**(2) (apr 2010). https://doi.org/10.1214/09-aap625

31. Knuth, D.E.: The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, Boston, third edn. (1997)
32. Kogan, D., Manohar, N., Boneh, D.: T/key: Second-factor authentication from secure hash chains. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 983–999. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017). https://doi.org/10.1145/3133956.3133989
33. Krawczyk, D.H., Eronen, P.: HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869 (May 2010). https://doi.org/10.17487/RFC5869
34. Langley, A., Hamburg, M., Turner, S.: Elliptic Curves for Security. RFC 7748 (Jan 2016). https://doi.org/10.17487/RFC7748, https://www.rfc-editor.org/info/rfc7748
35. Lonvick, C.M., Ylonen, T.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Jan 2006). https://doi.org/10.17487/RFC4253, https://www.rfc-editor.org/info/rfc4253
36. Mattsson, J.: Stream Cipher Design - An evaluation of the eSTREAM candidate Polar Bear. Master's thesis, Royal Institute of Technology (2006), https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.40
37. McGrew, D., Rescorla, E.: Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764 (May 2010). https://doi.org/10.17487/RFC5764
38. McKay, K., Cooper, D.: Guidelines for the selection, configuration, and use of transport layer security (tls) implementations (2019-08-29 2019). https://doi.org/10.6028/NIST.SP.800-52r2
39. Morrison, K.E.: Random maps and permutations, https://aimath.org/~morrison/Research/randommaps.pdf
40. Nir, Y., Langley, A.: ChaCha20 and Poly1305 for IETF Protocols. RFC 8439 (Jun 2018). https://doi.org/10.17487/RFC8439
41. NIST: Fips 180-4 secure hash standard (shs) (aug 2015). https://doi.org/10.6028/NIST.FIPS.180-4, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
42. NIST: Fips 203 (draft) module-lattice-based key-encapsulation mechanism standard (2023-08-24 2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf
43. NIST: Fips 204 (draft) module-lattice-based digital signature standard (2023-08-24 2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.ipd.pdf
44. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 617–630. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/978-3-540-45146-4_36
45. Perrig, A.: The biba one-time signature and broadcast authentication protocol. Proceedings of the ACM Conference on Computer and Communications Security (12 2001), https://netsec.ethz.ch/publications/papers/biba.pdf
46. Pollard, J.M.: A Monte Carlo method for factorization. BIT **15**, 331–334 (1975), https://www.cs.cmu.edu/afs/cs/academic/class/15451-f11/www/lectures/lect1122_Pollard.pdf
47. Preuß Mattsson, J.: Hidden stream ciphers and tmto attacks on tls 1.3, dtls 1.3, quic, and signal. Cryptology ePrint Archive, Paper 2023/913 (2023), https://eprint.iacr.org/2023/913

48. Preuß Mattsson, J., Palombini, F., Vučinić, M.: Comparison of CoAP Security Protocols. Internet-Draft draft-ietf-iotops-security-protocol-comparison-03, Internet Engineering Task Force (Oct 2023), https://datatracker.ietf.org/doc/draft-ietf-iotops-security-protocol-comparison/03/, work in Progress
49. Preuß Mattsson, J., Sethi, M.: EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3. RFC 9190 (Feb 2022). https://doi.org/10.17487/RFC9190
50. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446
51. Rescorla, E., Tschofenig, H., Modadugu, N.: The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. RFC 9147 (Apr 2022). https://doi.org/10.17487/RFC9147
52. Selander, G., Preuß Mattsson, J., Palombini, F.: Ephemeral Diffie-Hellman Over COSE (EDHOC). Internet-Draft draft-ietf-lake-edhoc-23, Internet Engineering Task Force (Jan 2024), https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/23/, work in Progress
53. Selander, G., Preuß Mattsson, J., Palombini, F., Seitz, L.: Object Security for Constrained RESTful Environments (OSCORE). RFC 8613 (Jul 2019). https://doi.org/10.17487/RFC8613
54. Selander, G., Preuß Mattsson, J., Tiloca, M., Höglund, R.: Ephemeral Diffie-Hellman Over COSE (EDHOC) and Object Security for Constrained Environments (OSCORE) Profile for Authentication and Authorization for Constrained Environments (ACE). Internet-Draft draft-ietf-ace-edhoc-oscore-profile-03, Internet Engineering Task Force (Oct 2023), https://datatracker.ietf.org/doc/draft-ietf-ace-edhoc-oscore-profile/03/, work in Progress
55. Siemens Industry Sector: Ruggedcom ros-f v4.2.2.f security target reference guide (August 2018), https://cache.industry.siemens.com/dl/files/226/109760226/att_993693/v1/ROS-F_v4.2.2.F_Security-Target_Reference-Guide_EN.pdf
56. Signal: Signal technical documentation, https://signal.org/docs/
57. Springall, D., Durumeric, Z., Halderman, J.A.: Measuring the security harm of tls crypto shortcuts. In: Proceedings of the 2016 Internet Measurement Conference. pp. 33–47 (2016), https://jhalderm.com/pub/papers/forward-secrecy-imc16.pdf
58. Stebila, D.: Security analysis of the imessage pq3 protocol, https://security.apple.com/assets/files/Security_analysis_of_the_iMessage_PQ3_protocol_Stebila.pdf
59. Stone, R.: Fbi document says the feds can get your whatsapp data — in real time, https://www.rollingstone.com/politics/politics-features/whatsapp-imessage-facebook-apple-fbi-privacy-1261816/
60. Thomson, M., Benfield, C.: HTTP/2. RFC 9113 (Jun 2022). https://doi.org/10.17487/RFC9113, https://www.rfc-editor.org/info/rfc9113
61. Thomson, M., Turner, S.: Using TLS to Secure QUIC. RFC 9001 (May 2021). https://doi.org/10.17487/RFC9001, https://www.rfc-editor.org/info/rfc9001
62. TOP500 project: The top500 list, https://www.top500.org/
63. Tüxen, M., Rescorla, E., Seggelmann, R.: Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP). RFC 6083 (Jan 2011). https://doi.org/10.17487/RFC6083
64. Westerlund, M., Preuß Mattsson, J., Porfiri, C.: Datagram Transport Layer Security (DTLS) over Stream Control Transmission Protocol (SCTP). Internet-Draft draft-ietf-tsvwg-dtls-over-sctp-bis-07, Internet Engineering Task Force (Oct 2023), https://datatracker.ietf.org/doc/draft-ietf-tsvwg-dtls-over-sctp-bis/07/, work in Progress
65. Wikipedia: Weak key, https://en.wikipedia.org/wiki/Weak_key

A Multi-Connection Attacks - Collisions Between Several Chains

The probabilities of a collision occurring within chains and between chains are not equal. An attacker might look at collisions between keys in several chains to be able to gather more data or, depending on the type of chain, increase the probability for a collision. A collision between two chains breaks the security of both sessions but does not indicate a cycle in any of the chains. Let p' be the probability of a collision between $2v$ keys where v keys are from one chain and the other v keys are from another chain. Let p'' be the probability of a collision between c keys k_m from c different chains.

$\boldsymbol{\pi}$-chains. While the probability of collision inside a single π-chain is zero, the probability of collisions between two π-chains is not zero. The probability of collision between two sets of v keys each from two chains using the same permutation is $\approx v^2/N$. The attacker can achieve this probability by looking for collisions between the keys $k_0, k_v, k_{2v}, \ldots, k_{(v-1)v}$ from the first chain and the keys $k_0, k_1, k_2, \ldots, k_{v-1}$ in the second chain. Note that the probability of collision between two sets of v keys each from two chains using different random permutations is also v^2/N. This probability is given by the birthday attack between two sets. The probability of a collision between c keys k_m from c different chains is $c^2/2N$. We get

$$p'_\pi \approx \frac{v^2}{N} \text{ and } p''_\pi \approx \frac{c^2}{2N} \ . \tag{46}$$

$\boldsymbol{\omega}$-chains. The probability of a collision between $2v$ keys where v keys are from one chain and the other v keys are from another chain is the same as the collision probability of $2v$ keys from a single chain. The probability is given by the birthday attack as $\approx (2v)^2/2N = 2v^2/N$. As $|S^*_m| \approx N$, the probability of a collision between c keys k_m from c different chains is $c^2/2N$. We get

$$p'_\omega \approx \frac{2v^2}{N} \text{ and } p''_\omega \approx \frac{c^2}{2N} \ . \tag{47}$$

$\boldsymbol{\xi}$-chains. The probability of a collision between $2v$ keys where v keys are from one chain and the other v keys are from another chain is the same as the collision probability of $2v$ keys from a single chain. The probability is given by the birthday attack as $\approx (2v)^2/2N = 2v^2/N$. As $|S^*_m| \approx 2N/m$, the probability of a collision between c keys k_m from c different chains is $c^2/2(2N/m) = mc^2/4N$. We get

$$p'_\xi \approx \frac{2v^2}{N} \text{ and } p''_\xi \approx \frac{mc^2}{4N} \ . \tag{48}$$

$\boldsymbol{\rho}$-chains. Based on our numerical simulations, the probability p' of a collision between $2v$ keys where v keys are the last keys from one chain and the other v keys are the last keys from another chain seems to be significantly higher than the probability $\approx 2mv/N$ of a collision between the last $2v$-keys in a single chain. Note that the collision probability is at least $2mv/N$ as the probability of a collision inside each of the chains is mv/N. Based on our numerical simulations in key spaces with size 2^{16} and 2^{24}, $p' \approx 4mv/N$ and it is beneficial for an attacker to look for collisions between two chains. Collisions between ρ-chains is shortly mentioned in [24], which write that the analysis seems to be complicated. The probability of a collision between c keys k_m from c different chains is $c^2/2(2N/m) = mc^2/4N$. We get

$$p'_\rho \approx \frac{4mv}{N} \text{ and } p''_\rho \approx \frac{mc^2}{4N} \ . \tag{49}$$

B Expanded State Space in $\boldsymbol{\rho}$-, $\boldsymbol{\xi}$-, and $\boldsymbol{\omega}$-chains.

In all existing security protocols we are aware of, the only information retained between iterations is the key k_i, resulting in the chain state size n being equal to the key size. The chain state size can be increased by increasing the key size. The state size n can also be expanded beyond the key size by incorporating a salt z with size n_z as part of the context

$$\begin{aligned} k_{i+1} &= \mathrm{KDF}(k_i, label1, [z_i, \dots], n_k) \ , \\ z_{i+1} &= \mathrm{KDF}(k_i, label2, [z_i, \dots], n_z) \ . \end{aligned} \tag{50}$$

The state size becomes $n = n_k + n_z$, where n_k is the key size. This is useful in KDFs that has a fixed key size or those that only make use of a limited amount of entropy from the key. Considering the relatively inexpensive nature of key derivation, the approach detailed in Eq. (50) seems preferable to using a fixed salt like z_0 as context, as in $k_{i+1} = \mathrm{KDF}(k_i, label, [z_0, \dots], n_k)$. The latter solution would expand the space S^*_m for all chains but not the space S_m for a single chain.