On Re-keying Mechanisms for Extending The Lifetime of Symmetric Keys

draft-smyshlyaev-re-keying

Stanislav V. Smyshlyaev, Ph.D.
Head of Information Security Department, CryptoPro LLC

Key capacity

Key capacity — the amount of data that is (directly) encrypted with the key.

Must be limited due to:

- general combinatorial properties of cipher modes of operation (recent example: Sweet32);
- estimations of key material needed for success of various cryptanalysis methods for a used cipher (linear, algebraic, differential, logical etc.);
- bounds following from side-channel cryptanalysis methods of block ciphers.

What should be done, when it is exceeded?

Do a new key establishment

- A full new key establishment on sides' keypairs...
- ... or obtaining a new agreement key using VKO (hashing the result of DH, multiplied by random UKM — see RFC 7836) with the same key pairs and other UKM...
- ... with new IVs, etc.

Advantages

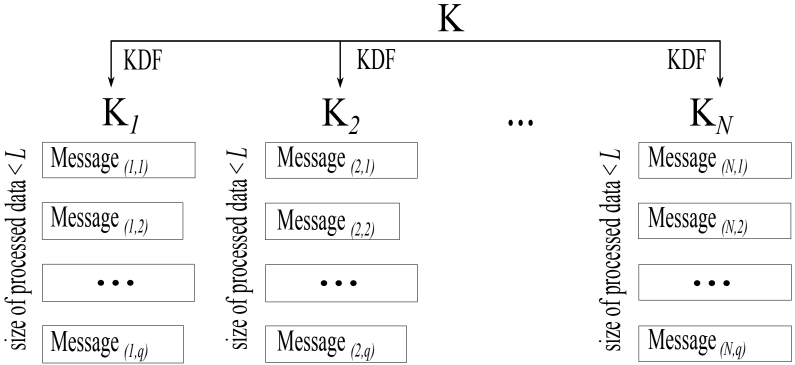
- „Starting a new life" for the session keys — no deep additional security analysis needed (except for the key establishment itself).

Disadvantages

- Inserting key agreement operations between sections would drastically reduce encryption performance.
- Multiple (and possibly related) operations with asymmetric crypto — additional analysis for key establishment is need for the particular case.

Key diversification (external re-keying)

- Uses an initial (negotiated) key as a master key, which is never used directly for the encryption but is used for session key derivation.
- A new derived session key (and IV) for each separate section.



Key diversification (external re-keying)

It is quite efficient and secure, but is appropriate only in case of two conditions satisfied:

- section size is large enough and key diversification operation in key tree is fast enough to make delays for session key generation negligible compared to total key usage time (for some mechanisms the second key and following ones the process can be made in parallel);
- the total size of encrypted data is large enough, so the starting delay (for generation of the first session key from the root (master) key) is acceptable.

NIST Special Publication 800-108

KDF in Counter Mode

$$K_i = PRF(K, [i]_2 | label | 0x00 | Context | [L]_2)$$

KDF in Feedback Mode

$$K_i = PRF(K, K_{i-1} | [i]_2 | label | 0x00 | Context | [L]_2)$$

KDF in Double-Pipeline Iteration Mode

$$A_i = PRF(K, A_{i-1});$$
$$K_i = PRF(K, A_i | [i]_2 | label | 0x00 | Context | [L]_2)$$

Re-keying in TLS 1.3 (draft-ietf-tls-tls13-18)

KeyUpdate

$$\begin{aligned}\text{traffic\_secret\_N} = \mathrm{HKDF}_{\text{Expand}}(&\text{traffic\_secret\_N} - 1,\\ &\text{“application traffic secret“},\ \text{““},\ \text{Hash.length})\end{aligned}$$

$$\text{write\_key} = \mathrm{HKDF}_{\text{Expand}}(\text{traffic\_secret\_N},\ \text{“key“},\ \text{““},\ \text{key\_length})$$

Security analysis

«Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-Keying Techniques», M. Abdalla, M. Bellare.

- Parallel variant: $\mathrm{K_i} = \mathrm{PRF(K, i)}$.
- Serial variant: $\mathrm{K_i} = \mathrm{PRF(StateKey_i, 0)}$, $\mathrm{StateKey_{i+1}} = \mathrm{PRF(StateKey_i, 1)}$, $\mathrm{StateKey_1} = \mathrm{K}$.

Complete and correct security proofs exist.

Parallel variant: the capacity of the initial key

If we have really hard restrictions on key capacity (e.g., an adversary with powerful side-channel analysis equipment), it can exceed even for the initial („master“) key.

NIST Special Publication 800-108: Key Hierarchy

			RootKey			
		KDF(	RootKey	, i&Mask1)		
	KDF(	Divers(	RootKey	, i&Mask1)	, i&Mask2)	
Key[i] = KDF(	Divers(	Divers(	RootKey	, i&Mask1)	, i&Mask2)	, i&Mask3)

- Parallel variant: $\mathrm{K_i = PRF(K, i)}$.
- Serial variant: $\mathrm{K_i = PRF(StateKey_i, 0)}$,
  $\mathrm{StateKey_{i+1} = PRF(StateKey_i, 1)}$, $\mathrm{StateKey_1 = K}$.

Advantages

- The material encrypted on each derived key $\mathrm{K_i}$ can be strictly limited by the side of the section without any limits on the lifetime of the original key K.
- The adversary cannot combine the information (input-output behaviour, side-channel information...) obtained when observing work on several derived keys.
- The leakage of one derived key $\mathrm{K_i}$ does not have any impact on other derived keys.
- The mechanism can be chosen independently of a mode of operation.

- Parallel variant: $\mathrm{K_i} = \mathrm{PRF}(\mathrm{K}, \mathrm{i})$.
- Serial variant: $\mathrm{K_i} = \mathrm{PRF}(\mathrm{StateKey_i}, 0)$, $\mathrm{StateKey_{i+1}} = \mathrm{PRF}(\mathrm{StateKey_i}, 1)$, $\mathrm{StateKey_1} = \mathrm{K}$.

Disadvantages

- In both variants $\mathrm{K_1} \neq \mathrm{K}$ — thus, we always have to make at least one PRF calculation, even for extremely short plaintexts.
- Security proofs significantly depend on keeping some state (K and $\mathrm{StateKey_i}$) unused with the cipher itself.

And what if we have chosen the mode of operation (CTR or GCM) and want to

1) keep the property of

- having the strong limits of the section that is explicitly encrypted with any symmetric key;
- impossibility of an adversary to combine the information obtained from different sections

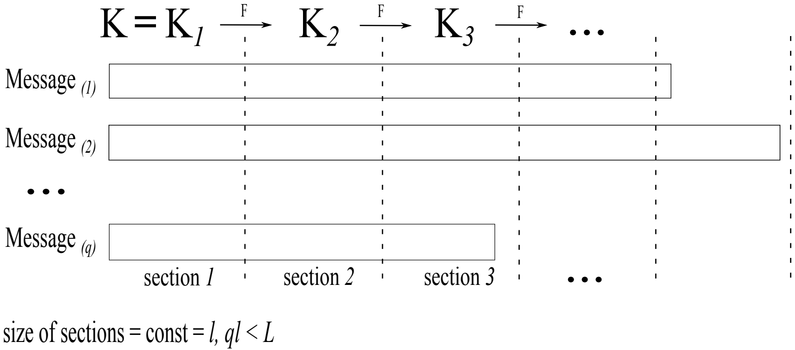
— to limit the possibility of an adversary to study anything about any encryption key by one section;

2) obtain also the properties of

- being efficient on small messages (does not need any additional operations on such);
- not restarting encryption with new IV's (and MAC calculation) frequently.

Internal re-keying („key meshing“)

- $K_1 = K$, $IV_1 = IV$;
- $(K_{i+1}, IV_{i+1}) = F(K_i, IV_i, i+1)$.

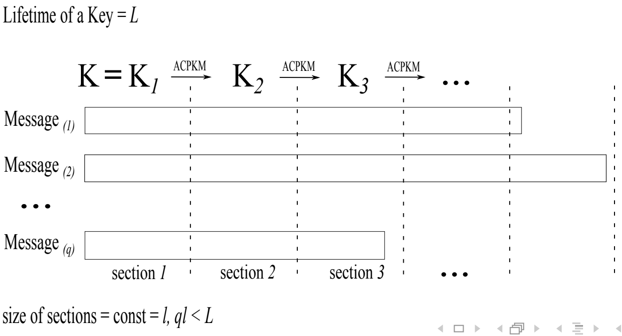


The proposed method of re-keying („key meshing“)

draft-smyshlyaev-re-keying

For CTR/GCM with n-bit block, $\mathrm{CTR_i} = (\mathrm{ICN} \mid \mathrm{counter})$, with c-bit counter value and (n − c)-bit ICN.

- $K_{i+1} = \text{ACPKM-CTR}(K_i) = \mathrm{MSB}_k(E_{K_i}(W_1)|\ldots|E_{K_i}(W_J))$,
- The section size MUST be less than $2^{c/2-1}$ blocks.



draft-smyshlyaev-re-keying

For CTR/GCM with n-bit block, $\mathrm{CTR_i} = (\mathrm{ICN} \mid \mathrm{counter})$, with c-bit counter value and $(n - c)$-bit ICN.

- $K_{i+1} = \mathrm{ACPKM\text{-}CTR}(K_i) = \mathrm{MSB}_k(E_{K_i}(W_1)|\ldots|E_{K_i}(W_J))$,
- The section size MUST be less than $2^{c/2-1}$ blocks;
- $(n - c + 1)$-th bit of each W_j is 1;
- W_j are defined for any
  - block size n of $64 \leqslant n \leqslant 512$,
  - counter size c of $32 \leqslant c \leqslant \frac{3}{4}n$,
  - key size k of $128 \leqslant k \leqslant 512$.
- W_j are pairwise different fixed constants for all allowed n, c, k.

draft-smyshlyaev-re-keying

$K_{i+1} = \mathrm{ACPKM\text{-}CTR}(K_i) = \mathrm{MSB}_k(E_{K_i}(W_1)|...|E_{K_i}(W_J))$.

Disandvantages

- The mechanism must not be chosen independently of a mode of operation.
- The leakage of one section key K_i will lead to a leakage of all following keys.

draft-smyshlyaev-re-keying

$$K_{i+1} = \mathrm{ACPKM\text{-}CTR}(K_i) = \mathrm{MSB}_k(E_{K_i}(W_1)|...|E_{K_i}(W_J))\text{.}$$

Advantages (Performance)

- There is no additional unnecessary operations for short messages — if the message length is shorter than the section size, the initial key will be used;
- The message size is not needed to be known in advance — key transformations are made when and only when needed;
- No need to restart the encryption process with new IV's (and GHASH calculation).

draft-smyshlyaev-re-keying

$K_{i+1} = \text{ACPKM-CTR}(K_i) = \text{MSB}_k(E_{K_i}(W_1)|...|E_{K_i}(W_J))$.

Advantages (Security)

- The material encrypted on each derived key K_i can be strictly limited by the side of the section without any limits on the lifetime of the original key K;
- The adversary cannot combine the information (input-output behaviour, side-channel information...) obtained when observing work on several derived keys;
- The total lifetime of an initial key drastically increases.

Really?

draft-smyshlyaev-re-keying

$K_{i+1} = \text{ACPKM-CTR}(K_i) = \text{MSB}_k(E_{K_i}(W_1)|...|E_{K_i}(W_J))$.

Advantages (Security)

- The material encrypted on each derived key K_i can be strictly limited by the side of the section without any limits on the lifetime of the original key K;
- The adversary cannot combine the information (input-output behaviour, side-channel information...) obtained when observing work on several derived keys;
- The total lifetime of an initial key drastically increases.

Really?

Security model

Cipher mode with re-keying is considered as an extension of a «base» cipher mode of operation, since it affects the process of processing of every single message.

Key meshing method must not be considered without specifying cipher mode of operation.

Security models

Security models for block ciphers

- PRF — «Pseudorandom function»;
- PRP-CPA — «Pseudorandom permutation in chosen plaintext attack»;
- PRP-CCA — «Pseudorandom permutation in chosen ciphertext attack»;

Security models for cipher modes

- LOR-CPA — «Left Or Right in Chosen Plaintext Attack» (Bellare M., Desai A., Jokipii E., Rogaway P. A Concrete Security Treatment of Symmetric Encryption, 2000);

Comparison with CTR

Base assumptions

In case of the block cipher that has no specific methods to decrease the security, the values of adversary's advantages are bounded in the following way:

$$\mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRF}}(t, q) \approx \frac{t}{2^k} + \frac{q^2}{2^n},$$
$$\mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRP\text{-}CPA}}(t, q) \approx \frac{t}{2^k},$$
$$\mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRP\text{-}CCA}}(t, q) \approx \frac{t}{2^k}.$$

Known for CTR

$$\mathrm{Adv}_{\mathrm{CTR}}^{\mathrm{LOR\text{-}CPA}}(\mathrm{t},\mathrm{q},\mathrm{m}) \leqslant 2 \cdot \mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRF}}(\mathrm{t}+\mathrm{q}+\mathrm{nqm},\mathrm{qm}).$$

Main result for ACPKM (preprint: eprint.iacr.org/2016/628)

$$\mathrm{Adv}_{\mathrm{ACPKM}_\ell}^{\mathrm{LOR\text{-}CPA}}(\mathrm{t},\mathrm{q},\mathrm{m}\ell) \leqslant 4\mathrm{m} \cdot \mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRP\text{-}CCA}}\left(\mathrm{t}+\mathrm{m}\ell\mathrm{q}+\mathrm{q}\cdot\frac{\mathrm{n}}{2},\mathrm{q}\cdot\ell,\frac{\mathrm{k}}{\mathrm{n}}\right)+$$
$$+\,2\mathrm{m}\cdot\mathrm{Adv}_{\mathrm{E}}^{\mathrm{PRF}}(\mathrm{t}+\mathrm{qml},\mathrm{ql})+2\mathrm{m}\delta,$$

where $\delta = \frac{8\mathrm{q}\ell+6}{2^{\mathrm{n}}} + \left(\frac{4\mathrm{q}\ell}{2^{\mathrm{n}}}\right)^2$, if $\mathrm{k}=4\mathrm{n}$, and $\delta = \frac{4\mathrm{q}\ell+1}{2^{\mathrm{n}}} + \left(\frac{2\mathrm{q}\ell}{2^{\mathrm{n}}}\right)^2$, if $\mathrm{k}=2\mathrm{n}$.

Comparison

$$\mathrm{Adv}^{\mathrm{LOR\text{-}CPA}}_{\mathrm{CTR}}(\mathrm{t}, \mathrm{q}, \mathrm{m}\ell) \approx \frac{2\mathrm{m}^2\mathrm{q}^2\ell^2}{2^{\mathrm{n}}} + \frac{\mathrm{t} + \mathrm{q} + \mathrm{nmq}\ell}{2^{\mathrm{k}}} \approx \mathrm{m}^2 \cdot \frac{2\mathrm{q}^2\ell^2}{2^{\mathrm{n}}},$$

$$\mathrm{Adv}^{\mathrm{LOR\text{-}CPA}}_{\mathrm{ACPKM}_\ell}(\mathrm{t}, \mathrm{q}, \mathrm{m}\ell) \approx 2\mathrm{m}\left(2 \cdot \frac{\mathrm{t} + \mathrm{qm}\ell}{2^{\mathrm{k}}} + \frac{\mathrm{q}^2\ell^2}{2^{\mathrm{n}}} + \frac{\mathrm{t} + \mathrm{qm}\ell}{2^{\mathrm{k}}} + \delta\right) \approx \approx \mathrm{m} \cdot \frac{2\mathrm{q}^2\ell^2}{2^{\mathrm{n}}}.$$

Comparison

$$\mathrm{Adv}^{\mathrm{LOR\text{-}CPA}}_{\mathrm{CTR}}(t, q, m\ell) \approx \frac{2m^2q^2\ell^2}{2^n} + \frac{t + q + nmq\ell}{2^k} \approx m^2 \cdot \frac{2q^2\ell^2}{2^n},$$

$$\mathrm{Adv}^{\mathrm{LOR\text{-}CPA}}_{\mathrm{ACPKM}_\ell}(t, q, m\ell) \approx 2m\left(2 \cdot \frac{t + qm\ell}{2^k} + \frac{q^2\ell^2}{2^n} + \frac{t + qm\ell}{2^k} + \delta\right) \approx$$

$$\approx m \cdot \frac{2q^2\ell^2}{2^n}.$$

Performance for AES

Does not it reduce speed?

Machine characteristics

Intel Core i5-6500 CPU 3.20GHz, L1 D-Cache 32 KB x 4, L1 I-Cache 32 KB x 4, L2 Cache 256 KB x 4.

Speed of the encryption (OpenSSL) process in the base CTR mode with the hardware supported AES-256 was: 3800 MB/s.

KB	64	128	256	512	1024	2048	4096
MB/s	3700.4	3722.0	3753.7	3765.3	3770.0	3786.5	3795.2
%	2.6	2.1	1.2	0.9	0.8	0.4	0.2

The CTR-ACPKM mode with the AES-256 cipher (hardware support).

Performance for AES

Does not it reduce speed?

Machine characteristics

Intel Core i5-6500 CPU 3.20GHz, L1 D-Cache 32 KB x 4, L1 I-Cache 32 KB x 4, L2 Cache 256 KB x 4.

Speed of the encryption (OpenSSL) process in the base CTR mode with the hardware supported AES-256 was: 3800 MB/s.

KB	64	128	256	512	1024	2048	4096
MB/s	3700.4	3722.0	3753.7	3765.3	3770.0	3786.5	3795.2
%	2.6	2.1	1.2	0.9	0.8	0.4	0.2

The CTR-ACPKM mode with the AES-256 cipher (hardware support).

Speed of the encryption process in the base CTR mode with the hardware supported AES-128 cipher is 5160 MB/s.

KB	64	128	256	512	1024	2048	4096
MB/s	5040.4	5061.3	5080.6	5105.0	5120.1	5139.4	5150.2
%	2.3	1.9	1.0	0.9	0.7	0.4	0.2

The CTR-ACPKM mode with the AES-128 cipher (hardware support).

Summary

- External re-keying (defined independently of a mode):
  - drastically increases the lifetime of keys (considering general bounds, classical and side-channel attacks on used block cipher);
  - does not affect short messages transformation at all;
  - provides forward and backward security of section keys;
  - requires additional operations (KDFs) even for very short messages;
  - all procedures must be handled separately — not transparent;
  - in case of restrictive section lengths the key tree should be used — it becomes ineffective, if we do not use some additional techniques.
- Key meshing approach (defined for a particular mode):
  - drastically increases the lifetime (considering general bounds, classical and side-channel attacks on used block cipher);
  - almost does not affect performance for long messages;
  - does not affect short messages transformation at all;
  - transparent (works like any encryption mode) does not require changes of IV's and restarting MACing;
  - but does not provide backward security of section keys — if needed, might be combined with ext. re-keying (for much larger sections).

Open questions

- Can we define a similar key meshing with no additional keys for CFB/CBC/OMAC/...? („This type of method should work for a large class of schemes" — Mihir Bellare.)
- Alternative approaches for internal („transparent") re-keying?
- Do we need a CFRG RFC with various re-keying techniques (existing and new ones)? („A menu of choices for developers" — Kenny Paterson.)

Thank you for your attention!

Questions?

- Materials, questions, comments:
  - svs@cryptopro.ru

A security model for the cipher mode (for encryption) — LOR-CPA

An adversary A has access to an oracle $\mathcal{O}^{\mathrm{LOR}}$. Before starting the work the oracle $\mathcal{O}^{\mathrm{LOR}}$ chooses $\mathrm{b} \in_{\mathcal{U}} \{0,1\}$. The adversary A can make requests to the oracle $\mathcal{O}^{\mathrm{LOR}}$. Each of these requests is a pair of strings $(\mathrm{M}^0, \mathrm{M}^1)$, where $|\mathrm{M}^0| = |\mathrm{M}^1|$. In response to the request $(\mathrm{M}^0, \mathrm{M}^1)$ the oracle returns a string C that is a result of the processing of the string M^{b} according to the $\mathcal{SE}$ cipher mode.