SCARF: A Low-Latency Block Cipher for Secure Cache-Randomization[1]

1. Introducere
   - SCARF = cifru cu latenta redusa pentru randomizarea cache-ului.
   - Algoritmul opereaza pe blocuri de 60 biti si foloseste o cheie de 240 biti + un tweak de 48 biti.
   - Conceput pentru a bloca atacuri de tip side-channel cum ar fi Prime+Probe (Prime: atacatorul umple o zona din cache cu propriile date. Procesele victimei ruleaza normal, posibil inlocuind datele din cache. Probe: atacatorul recitește aceleasi date si masoara timpul de access. Daca accesul e lent, inseamna ca victima a folosit acea linie de cache)

2. Modul de functionare
   - O adresa de memorie cache arata in felul urmator:

Tag: Identificatorul unic al unei zone de memorie din bloc. 48 de biti.

Este folosit pe post de **tweak**.

Index: Selecteaza linia de cache. 10 biti.

Este **tinta criptarii**.

   - Cheia secreta (240 de biti) este generata la fiecare boot folosind TRNG-uri, iar responsabilitatea revine hardware-ului pentru a o stoca in siguranta (de exemplu intr-un modul SRAM dedicat)

   - Pasul 1: Se genereaza subcheile T^i (60 biti) din tweak-ul de 48 de biti si cheia de 240 de biti $K^4 || K^3 || K^2 || K^1$ astfel:

$$
\begin{aligned}
T^1 &= \text{expansion}(T) \oplus K^1, \\
T^2 &= \Sigma(\text{SL}(T^1)) \oplus K^2, \\
T^3 &= \text{SL}(\pi(\text{SL}(T^2) \oplus K^3)), \\
T^4 &= \text{SL}(\Sigma(T^3) \oplus K^4),
\end{aligned}
$$

Functia expansion(T) ia 48 de biti si adauga un bit de 0 la fiecare 4 biti pentru a extinde tweak-ul la 60 de biti.

Functiile Sigma, SL, pi sunt descrise in continuare:

[1] https://eprint.iacr.org/2022/1228

- τ_i reprezinta shiftarea la stanga cu *i* biti. Sigma este un amestec liniar al bitilor:

$$\Sigma(x) = x \oplus \tau_6(x) \oplus \tau_{12}(x) \oplus \tau_{19}(x) \oplus \tau_{29}(x) \oplus \tau_{43}(x) \oplus \tau_{51}(x)$$

- Functia SL (Substitution Layer) aplica funcția S pe fiecare din cele 12 cuvinte de 5 biți.
- Functia S:

$$S(x) = \Big((\tau_0(x) \vee \tau_1(x)) \wedge (\overline{\tau_3(x)} \vee \overline{\tau_4(x)})\Big) \oplus \Big((\tau_0(x) \vee \tau_2(x)) \wedge (\overline{\tau_2(x)} \vee \tau_3(x))\Big)$$

- π (Permutare) redistribuie pozitiile bitilor pentru a crea difuzie, unde x_i este mapat la x_{pi}, iar *pi* este:

$$\begin{aligned} p = {} & 1,6,11,16,21,26,31,36,41,46,51,56, \\ & 2,7,12,17,22,27,32,37,42,47,52,57, \\ & 3,8,13,18,23,28,33,38,43,48,53,58, \\ & 4,9,14,19,24,29,34,39,44,49,54,59, \\ & 5,10,15,20,25,30,35,40,45,50,55,60. \end{aligned}$$

- De asemenea, fiecare runda are propria cheie *rk* (30 biti), unde rk_i este cheia pentru runda *i*:

$$rk_2 \| rk_1 = T^1, \qquad rk_4 \| rk_3 = T^2,$$
$$rk_6 \| rk_5 = T^3, \qquad rk_8 \| rk_7 = T^4.$$

- Rundele R1 si R2 – Au ca intrare *x* (10 biti) si cheia pentru runda respectiva, *k* (30 biti), care este impartita la randul ei in 6 chei k_i de cate 5 biti fiecare.
- Functia nelinara G combina valoarea *x* cu cheile $k_{1...5}$ folosind rotatii si and-uri:

$$G(x, k_1, k_2, k_3, k_4, k_5) = \left[\bigoplus_{i=0}^{4} (\tau_i(x) \wedge k_{i+1})\right] \oplus (\tau_1(x) \wedge \tau_2(x))$$

- Pasul 2: Aplicarea a 7 runde de R1 si o ultima runda de R2.
- Functiile pentru runde arata astfel:

$$y = G(x_L, k_1, k_2, k_3, k_4, k_5) \oplus x_R, \qquad x_R = G(x_L, k_1, k_2, k_3, k_4, k_5) \oplus x_R,$$
$$x_R = S(x_L \oplus k_6), \qquad x_L = S(x_L) \oplus k_6.$$
$$x_L = y,$$

- La finalul unei runde, se returneaza x_L concatenat cu x_R.

3. Implementare
    - Am implementat functiile descrise anterior, criptarea cu 7 R1 + 1 R2, dar si decriptarea (Python).
    - https://github.com/AlexLicuriceanu/criptologie/blob/main/proiect/scarf.ipynb
    - Rezultat:

```
Key: 10000111001100001011000101110000100000101110111000010111101010011000
Tweak: 00111001101100111000011001110000010111011111100
Tweakey 0: 11101110110111110000110101001101010111000000111111011001000
Tweakey 1: 11001010010111001010110000000101010111111110010100101111001 01
Tweakey 2: 00011011010101011101011001001000100100010100011111001011111 0
Tweakey 3: 01001100100111011100000010010010110000110111011111100110100 0
Round Key 0: 01010111000000111111011001000
Round Key 1: 11101110110111110000110101001 1
Round Key 2: 01010111111110010100101110010 1
Round Key 3: 11001010010111001010110000000 1
Round Key 4: 00100100010100011111001011111 0
Round Key 5: 00011011010101011101011001001 0
Round Key 6: 10110000110111011111100110100 0
Round Key 7: 01001100100111011100000010010 0
Plaintext:  1010001001
Ciphertext: 1001110100
Decrypted:  1010001001
```

4. Alte observatii
    - Performanta: ~1-2ms per bloc (interpretat, fara optimizare), timp constant pentru toate rundele. Performanta mai slaba decat cea precizata in articol deoarece ei au masurat implementarea in hardware.
    - Am implementat SBOX-ul si inversul folosind tabele de interogare, in articol se calcula rezultatul SBOX-ului la momentul criptarii. De asemenea, in articol nu se vorbeste despre procesul de decriptare, autorii considerand ca nu este atat de important, deoarece se va folosi doar cand se face write-back (insa acest lucru nu este frecvent si se poate face oricum in parallel cu alte operatii).

5. Evaluare de Securitate
    - Attacker's view: Deoarece atacatorul nu observa output-ul direct al criptarii, SCARF este analizat drept o compunere de criptari si decriptari, asta inseamna ca atacatorul trebuie sa faca de 2 ori mai multe operatii, spre deosebire de un cifru unde poate vedea output-ul.
    - Atacuri statistice – liniar, diferential, boomerang: Autorii sustin ca aceste metode nu ar functiona impotriva SCARF, fiind nevoie de 2^{40} queries (interogari

- atacatorul poate trimite cel mult 2^{40} cereri catre SCARF pentru a vedea daca doua adrese acceseaza același set in cache) si 2^{80} computations (operatii – criptari/decriptari). In articolul acesta[2], autorii prezinta un key-recovery attack care foloseste 2^{39} queries si 2^{79} time pentru "full-round SCARF" (8 runde).

- Related-Tweak attack – Tweakey schedule-ul este neliniar, astfel ca subcheile rezultate au distanța Hamming mare si nu permit construirea de related-tweaks utile. Probabilitatea atacului este estimata sub 2^{-48}, considerata sigura.
- Impossible Differential attack – SCARF are difuzie completa in 3 runde, cel mult 6 runde pot fi vulnerabile teoretic.
- Higher-order differential / Integral attack – Nu s-au gasit distingători pe 4 sau mai multe runde.
- Side-channel attacks – Nu este vulnerabil la atacuri side-channel clasice: Nu se observă direct output-ul cifrului (doar conflicte de cache).
- De asemenea, un atacator nu poate vedea cheia, tag-ul sau criptarea, ci doar daca exista coliziune intre doua adrese, ceea ce este o informatie slaba pentru un atac.

[2] https://eprint.iacr.org/2025/315