Adatfolyam rejtjelezők

PPKE, ITK
Csapodi Márton

Szimmetrikus kulcsú rejtjelezés

Általában a rejtjelező kulcs és a dekódoló kulcs megegyezik, vagy egyikből a másik meghatározása "könnyű".

PÉLDA:

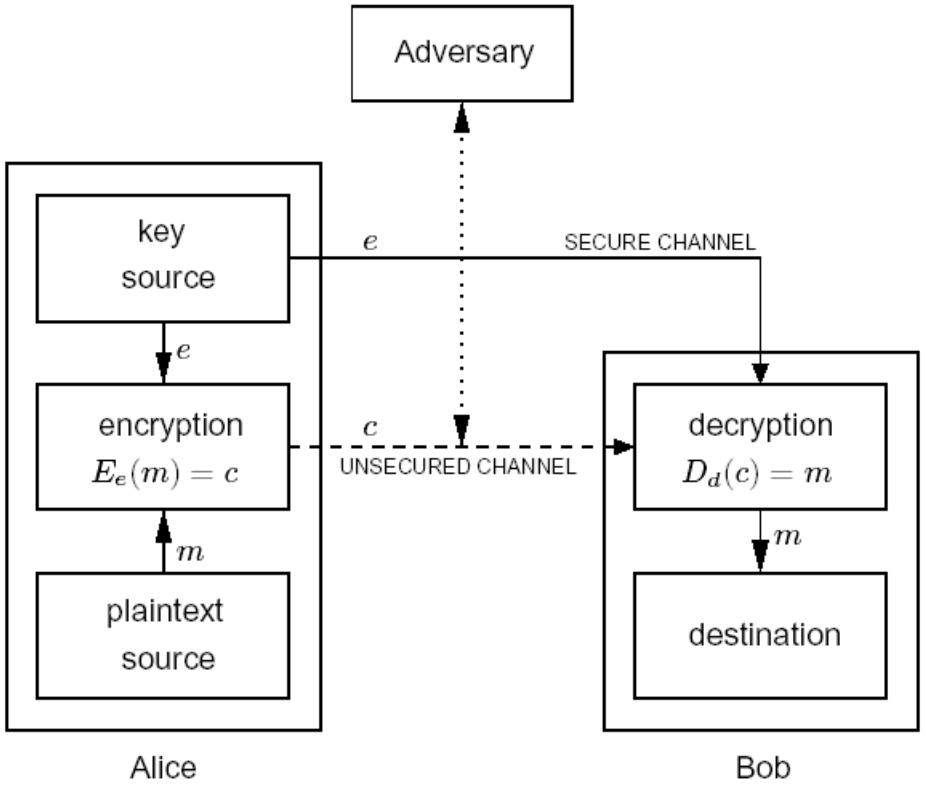
A: {A, Á, B, C,..., Z}

M, C: 5 hosszúságú karakterlánc

e: egy permutációja A-nak

A kulcsot el kell juttatni a feleknek és titokban kell tartani.

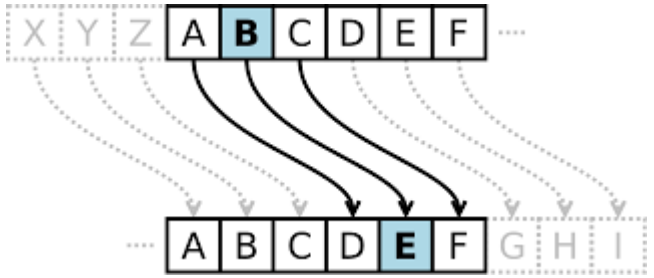
A kiosztás a legnagyobb probléma.



Blokk rejtjelezők

Definíció:

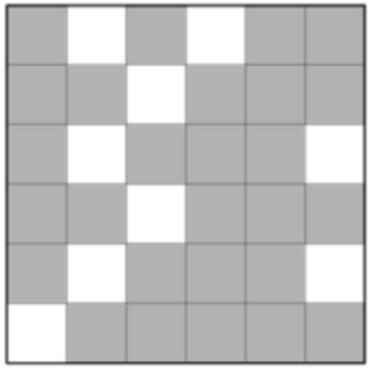
A nyílt szöveget t hosszúságú, A-beli elemekből álló láncokra bontja, és egyszerre egy blokkot rejtjelez.



Sokféle algoritmus van és sokféle felhasználási mód.

Egyszerű helyettesítéses (substitution) és felcseréléses (transposition) rejtjelezők és ezek kombinációi. Még ma is ezek a leghatékonyabb rejtjelezők (csak korszerű módon).

E	J	E	U	S	T
Y	G	L	É	Á	N
Á	E	Y	S	É	S
C	N	V	D	Í	B
O	E	R	M	M	R
N	E	Ű	N	R	Á



Adatfolyam rejtjelezők

Definíció:

Szimbólumonként (bit vagy karakter) rejtjelez. Szimbólumról-szimbólumra változó kulccsal.

kulcsfolyam: $e_1\, e_2\, e_3 \ldots e_n$

nyílt szöveg: $m_1\, m_2\, m_3 \ldots m_n$

rejtjelezett szöveg: $c_1\, c_2\, c_3 \ldots c_n$

rejtjelezés: $c_i = E_{ei}(m_i)$

dekódolás: $m_i = D_{di}(c_i)$

A transzformáció egyszerű, a kulcsfolyam generálása nehéz: teljesen véletlen, vagy kiinduló kulcsból generált folyam

Vernam rejtjelező:

A={0,1}

c=m XOR e

e=d

One-time-pad: Vernam rejtjelező teljesen véletlen kulcsfolyammal. Ez bizonyíthatóan nem feltörhető

Blokk vagy adatfolyam rejtjelező?

Minden blokk rejtjelezőből készíthető adatfolyam rejtjelező. (Blokk rejtjelezőknél tanuljuk.)

Adatfolyam rejtjelező alkalmazandó, ha:

- kis méretű vagy gyors hw megoldás kell (blokk rejtjelező alapúra nem áll fönn)
- nem lehet pufferelni, bitenként (karakterenként) kell dekódolni, várakozás nélkül
- zajos környezetben, mert adatfolyam rejtjelezésnél nincs hibaterjedés (vagy korlátozott)

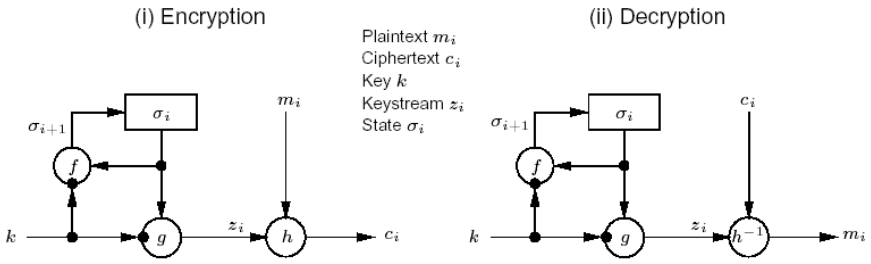
Távközlés (főleg a vezeték nélküli) általában adatfolyam rejtjelezést használ.

Kevés átfogó, széles körben alkalmazott, nyílt szabvány van adatfolyam rejtjelezőre.

Osztályozás

szinkron

álvéletlen bitsorozat a nyílt (rejtjelezett) szövegtől független

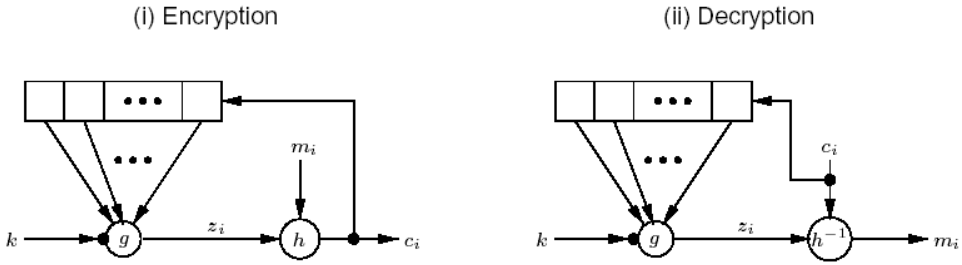


leggyakrabban: h = XOR

- szinkronizálni kell (reinicializáció, markerek, offszetek)
- bit módosulás esetén nincs hibaterjedés
- aktív támadás: bit módosítás a rejtjelezett üzenetben a nyílt üzenet bitet átbillenti

önszinkronizáló

álvéletlen bitsorozat a rejtjelezett üzenet bitjeiből



- szinkronizálja magát
- korlátozott hibaterjedés (törlés, beszúrás, módosulás esetén)

Szinkron adatfolyam rejtjelező = álvéletlen generátor?

Minden jó álvéletlen generátor jó adatfolyam kódoló is elvben.

Vagy mégse:

- gyorsaság az adatfolyam rejtjelező esetében fontosabb
- véletlenszerűség az álvéletlen generátor esetében fontosabb
- újra inicializáció az adatfolyam rejtjelező esetében fontosabb
- backtracking resistance az álvéletlen generátor esetében fontosabb

További osztályozás

LFSR alapú

HW megvalósíthatóság
nagy periódus
jó statisztikai tulajdonságok
algebrai eszközök

blokk rejtjelező alapú

minden blokk rejtjelezőhöz bizonyos működési mód esetén:

- CTR (Counter)
- OFB (Output Feedback)
- CFB (Cipher Feedback)

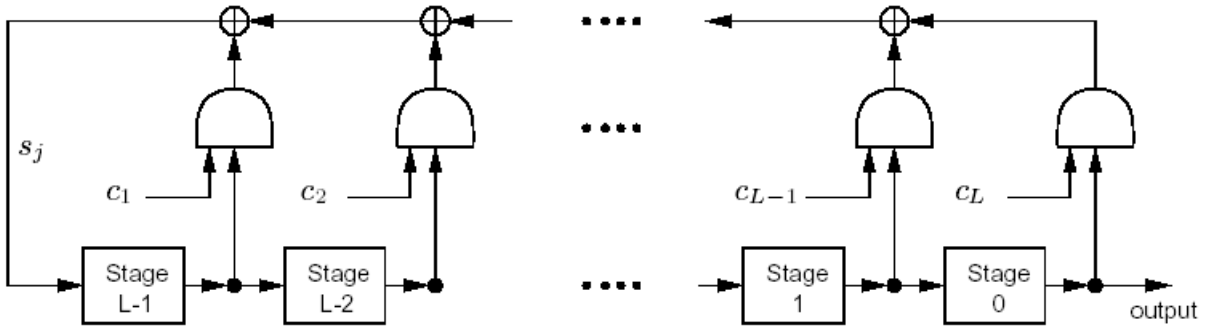
számelméleti probléma alapú

Blum-Blum-Shub (lassú)

dedikált

RC4
SEAL
SNOW
MUGI
PANAMA
TRIVIUM
stb.

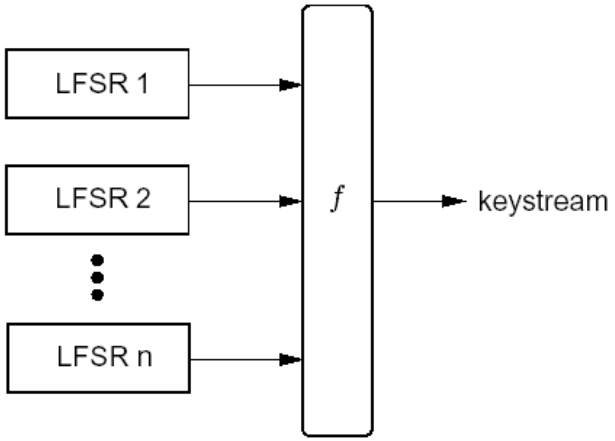
Linear Feedback Shift Registers

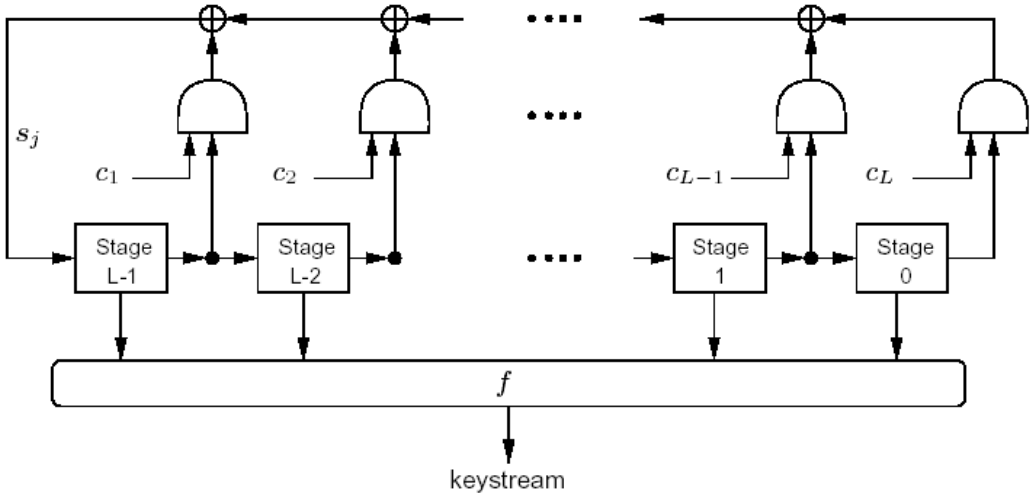


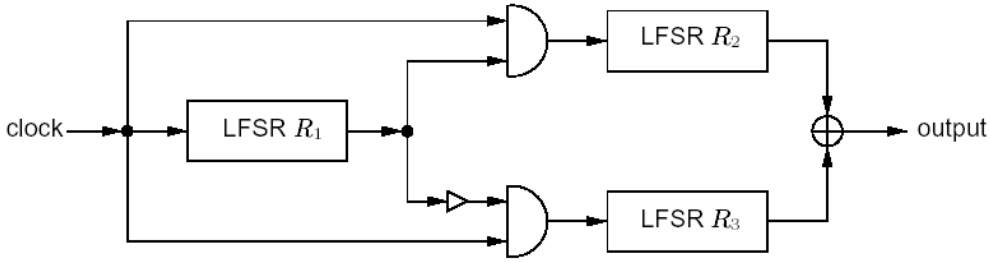
visszacsatolási (karakterisztikus) polinom:

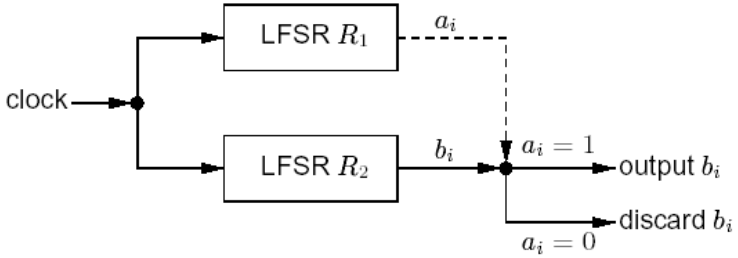
$$C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_L D^L \in \mathbb{Z}_2[D]$$

LFSR adatfolyam rejtjelezők









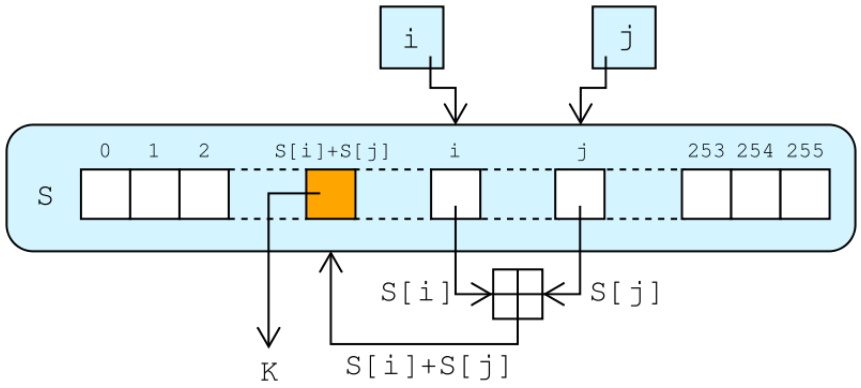
RC4

Rivest (1987), titkos algoritmus (RC2, RC4, RC5, RC6)

'94-ben nyilvánosságra került

Legszélesebb körben használt adatfolyam rejtjelező volt sokáig: WiFi, SSL/TLS, Secure Shell, Remote Desktop, PPP (point-to-point protocol)

Ma már nem biztonságos



inicializálás:

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap(&S[i],&S[j])
endfor
```

bit generálás:

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(&S[i],&S[j])
    output S[(S[i] + S[j]) mod 256]
endwhile
```

RC4 biztonság

WEP (WiFi) problémák oka nem az RC4 volt. WPA TKIP továbbra is RC4-et használ.

2011: BEAST attack on TLS 1.0. Az SSL/TLS protokollok szimmetrikus kulcsú titkosítását érinti, ha az blokk kódoló CBC módban, vagyis gyakorlatilag a nem RC4 alapú SSL/TLS verziókat mind. Emiatt még inkább előtérbe került az RC4 használata.

2013: Nagyon nagyszámú TLS csomag lehallgatása esetére van törés. Ez a gyakorlatban nem megvalósítható, de a Snowden ügy és 2015-ben megjelent praktikusabb törések miatt egyre több a figyelmeztető jel.

2015 óta tilos használni a TLS protokollban (RFC 7465)

Szabványok

Általános célú adatfolyam rejtjelező szabvány 2001-ig nem volt.

Egyes alkalmazási területek kapcsán:

GSM: A5/1, A5/2, A5/3

GPRS: GEA3

UMTS/3GPP: f8

Bluetooth: E0

IEEE 802.11 (WiFi): RC4

GSM

A5/1: erős

A5/2: gyenge

'80-as évek közepe, titkos algoritmus

'90-es években publikussá vált

ma már mindkettő gyenge

A5/3: ETSI szabvány

a kompatibilitás veszélyezteti a biztonságot

GPRS

GEA3: A5/3-hoz hasonló

UMTS/3GPP

f8: A5/3-hoz hasonló

A5/1

clock-controlled generator, 3 LFSR

$$x^{19} \oplus x^5 \oplus x^2 \oplus x \oplus 1$$

$$x^{22} \oplus x \oplus 1$$

$$x^{23} \oplus x^{15} \oplus x^2 \oplus x \oplus 1$$

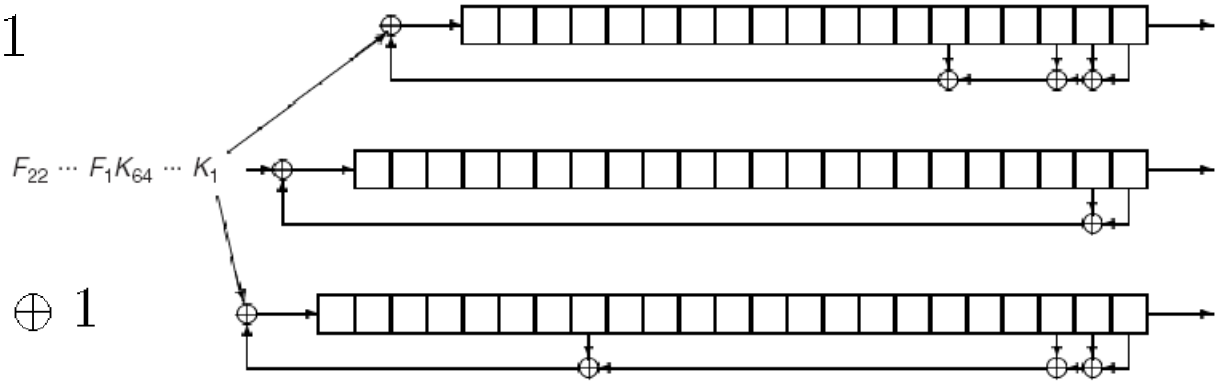


Fig. 1. Initialization of the A5/1 running-key generator

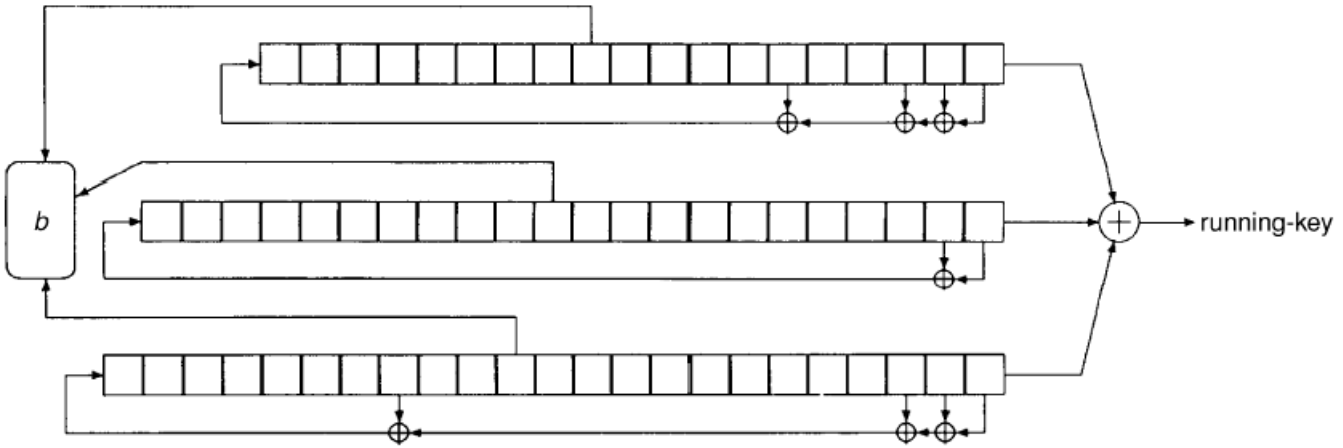


Fig. 2. A5/1 running-key generator

A5/2

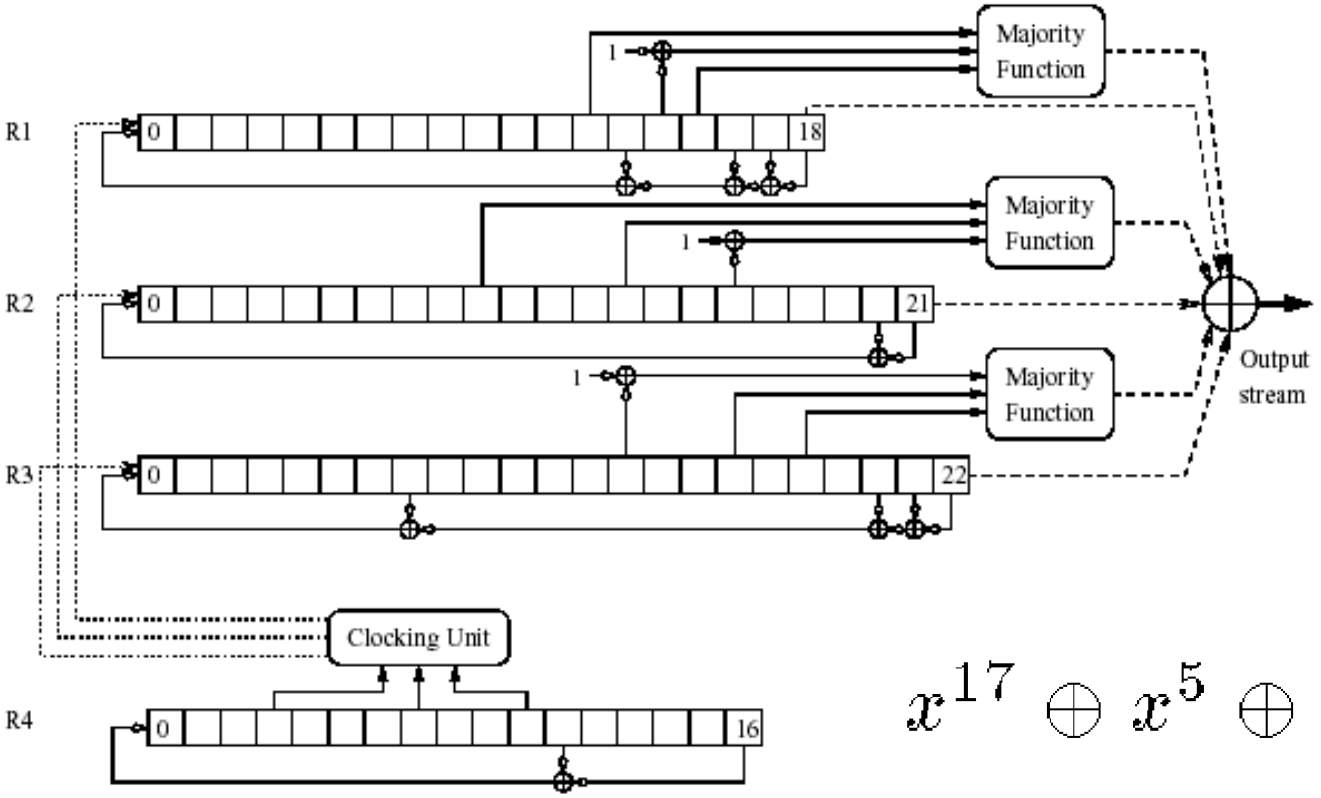


Fig. 1. The A5/2 internal structure

f8 és A5/3

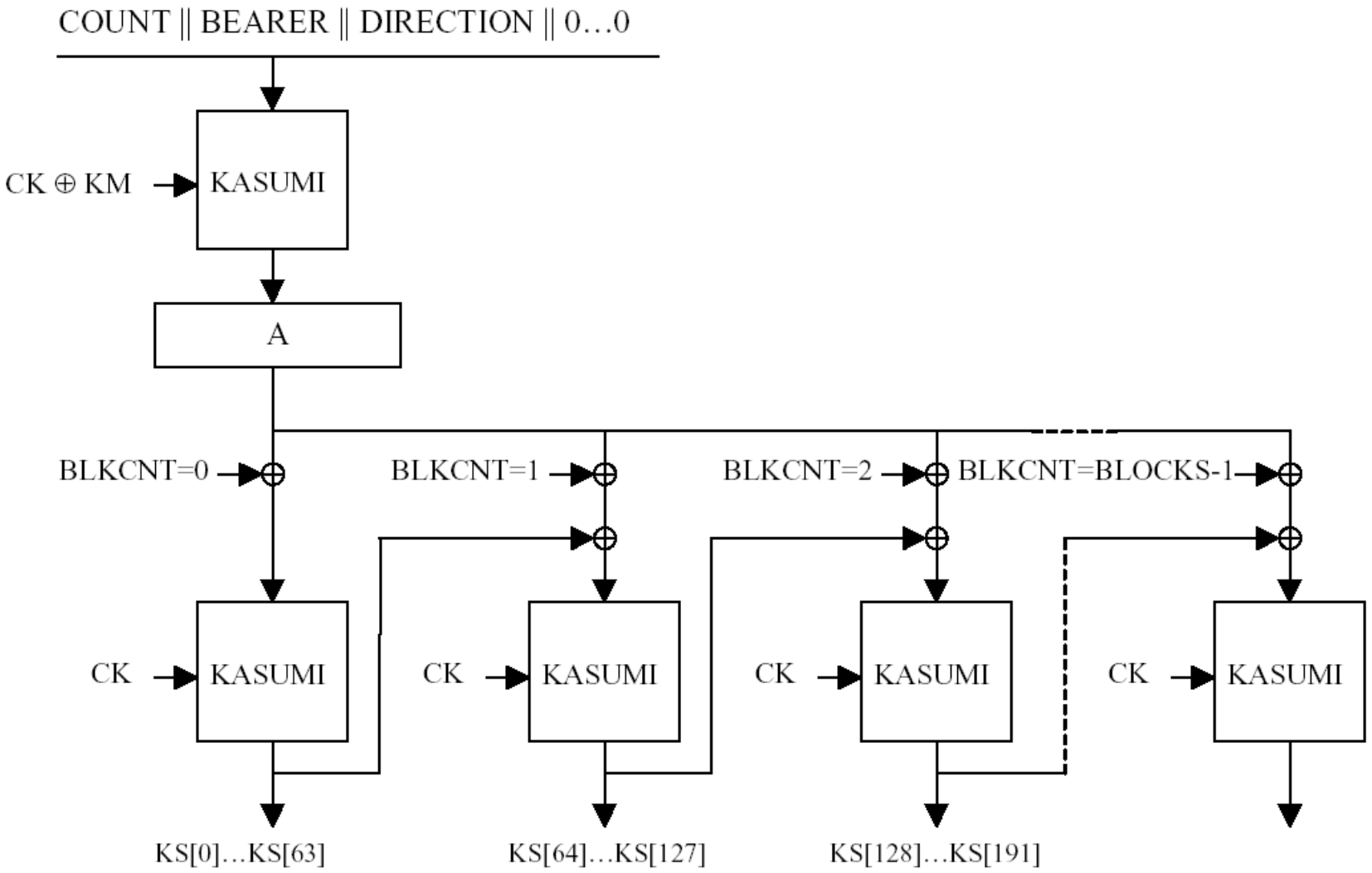
Blokk rejtjelező (KASUMI)
Output Feedback (OFB) mode-ban

KASUMI:
64 bit input
128 bit kulcs
64 bit output

f8:
input: 128 bit seed
<64 bit üzenetazonosító
output: 1-5114 álvéletlen bit

A5/3:
input: 64..128 bites kulcs kiegészítve
<64 bit üzenetazonosító
output: 2 x 114 álvéletlen bit

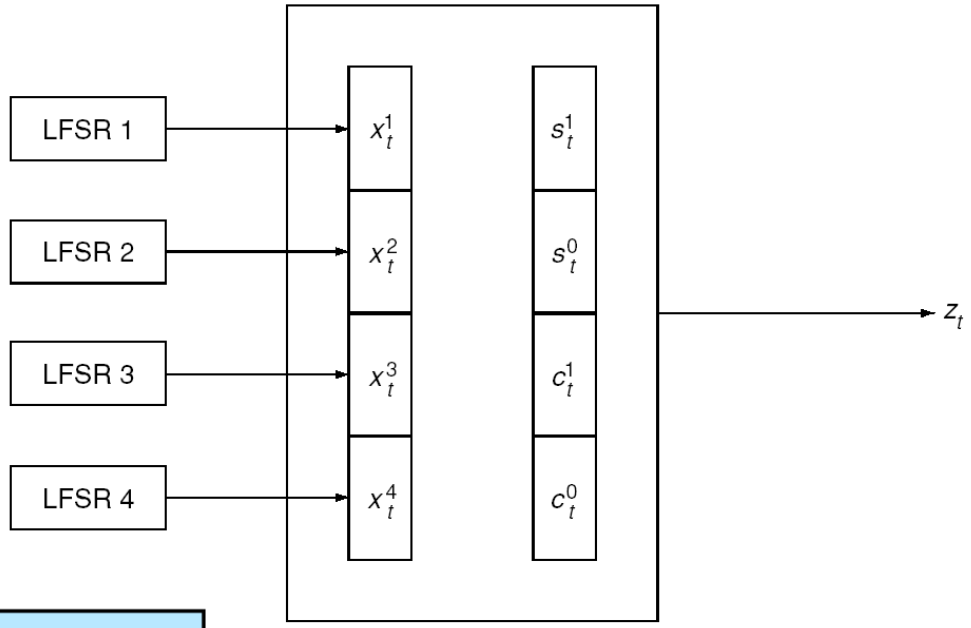
3GPP rejtjelező (f8)



Bluetooth rejtjelező (E0)

Summation generator

4 LFSR



i	L_i	feedback $f_i(t)$	weight
1	25	$t^{25} + t^{20} + t^{12} + t^8 + 1$	5
2	31	$t^{31} + t^{24} + t^{16} + t^{12} + 1$	5
3	33	$t^{33} + t^{28} + t^{24} + t^4 + 1$	5
4	39	$t^{39} + t^{36} + t^{28} + t^4 + 1$	5

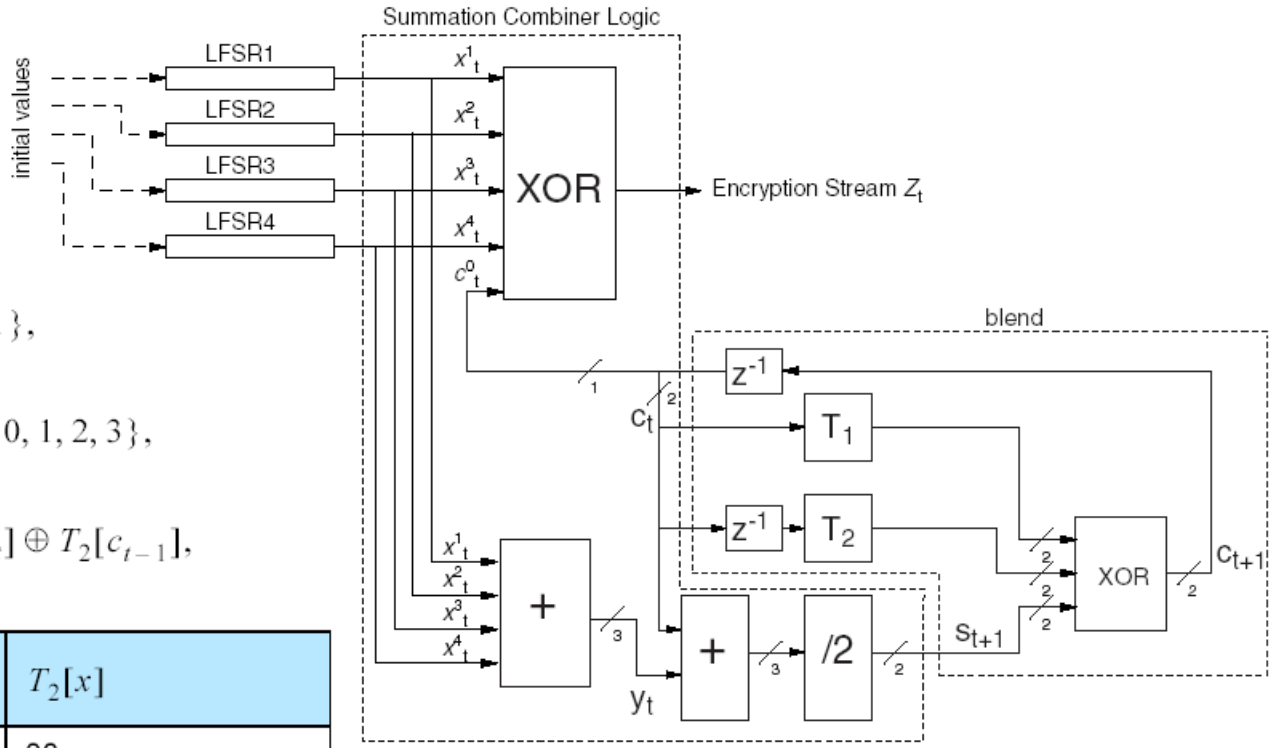
Bluetooth rejtjelező (E0)

$$y_t = \sum_{i=1}^{4} x_t^i,$$

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\},$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\},$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}],$$



x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

IEEE 802.11 (WiFi)

WEP (wired equivalent privacy) problémák

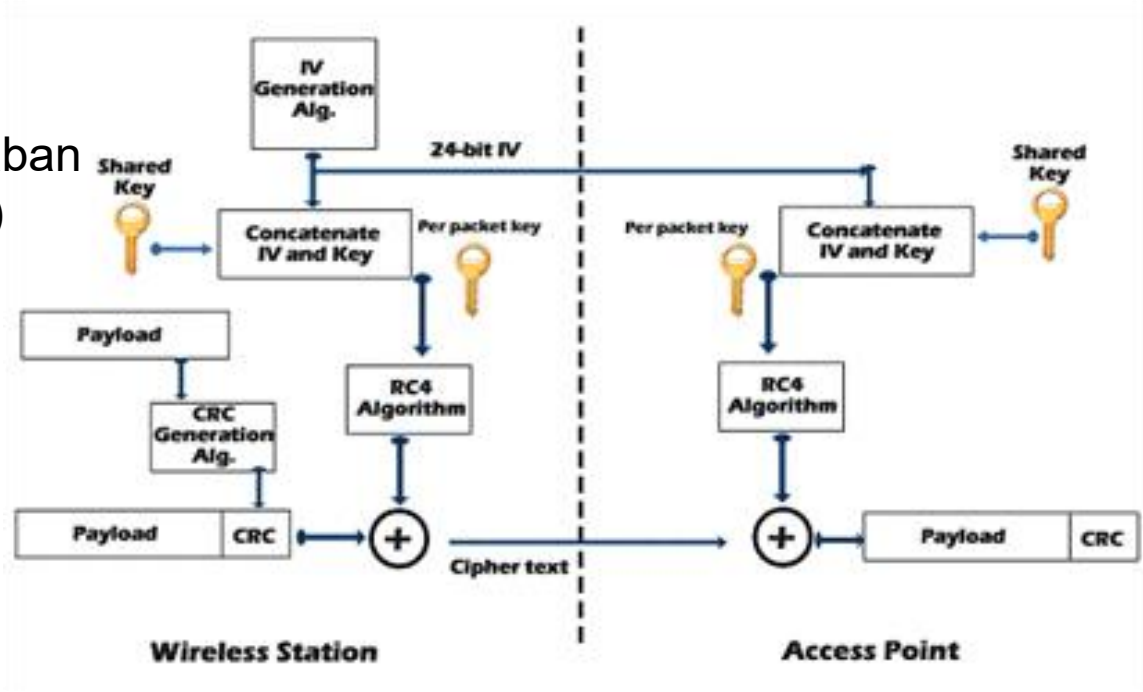
RC4 szinkron -> csomagonként kell szinkronizálni (új kulccsal inicializálni, mert egy kulcsot tilos kétszer használni - WEP amúgy megengedné)

nincs kulcsegyeztetés -> előre megosztott kulcs van (általában közös az egész hálózatban)

ezt egészíti ki egy 24 bites IV egyszerűen hozzáfűzve

WPA (WiFi protected access): RC4, de IV hozzáfűzés helyett keverve

WPA2: RC4 helyett AES



ISO/IEC 18033

2001-ben indult

ISO/IEC 18033: IT security techniques - Encryption algorithms

Part 1: General

Part 2: Asymmetric ciphers

Part 3: Block ciphers

Part 4: Stream ciphers

Part 4 osztályozás

keystream generator

- synchronous
  - blokk kódoló alapján (2)
  - dedikált (2)
- self-synchronizing
  - blokk kódoló alapján (1)

output function

- XOR
- MULTI-S01 (integritást is védi)

ISO/IEC 18033 KG

blokk kódoló alapján

Ismert technikák:

- CTR (Counter)
- OFB (Output Feedback)
- CFB (Cipher Feedback)

modes of operation

A szabvány mindegyiket megengedi

dedikált

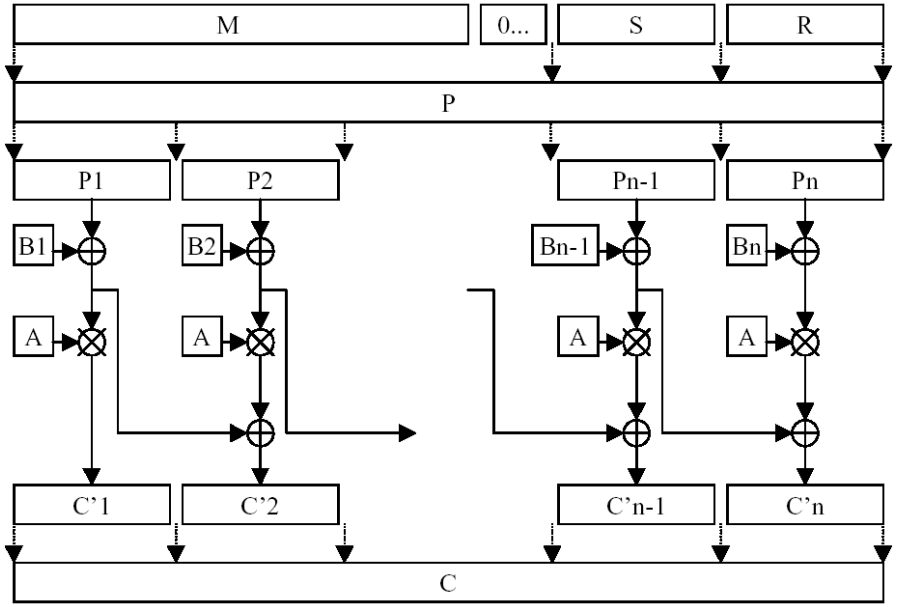
SNOW 2.0

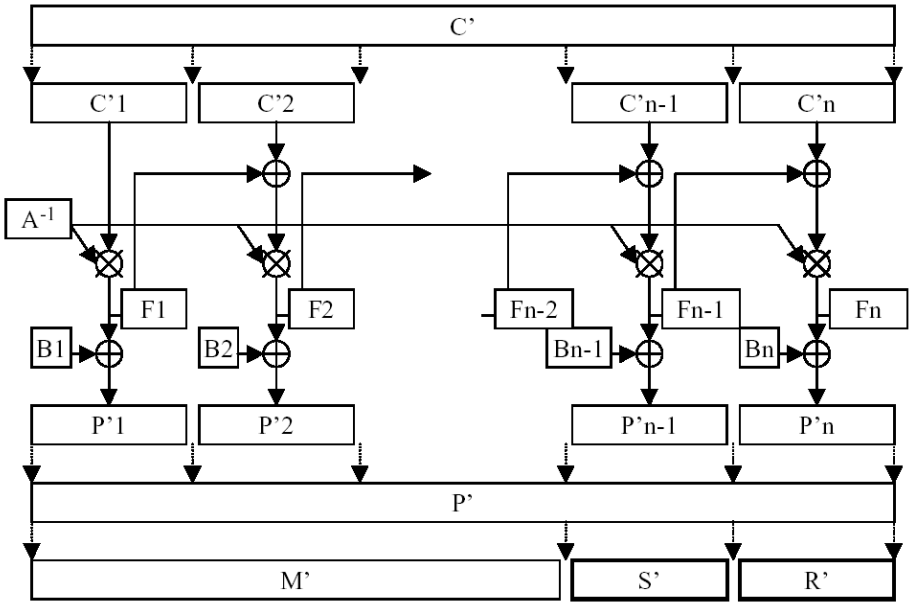
MUGI (elődje: PANAMA)

De:

- mindkettő 2002-es
- nincs self-synchronizing

ISO/IEC 18033 MULTI-S01





Encryption

Input:	Message	M ($64 \times n$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n+2$, each 64 bits)
		S (64 bits)
Output:	Ciphertext	C (($n+2$) $\times$ 64 bits)

$$F_i = P_i \oplus B_i,$$
$$C_i = A \otimes F_i \oplus F_{i-1}.$$

Decryption

Input:	Ciphertext	C ($n' \times 64$ bits)
	Redundant data	R (64 bits)
	Secret key	A ($\neq 0$, 64 bits)
		B_i ($1 \leq i \leq n'$, each 64 bits)
		S (64 bits)
Output:	Message	Reception error or message M ($64 \times (n-2)$ bits)

$$F_i = A^{-1} \otimes (C_i \oplus F_{i-1}),$$
$$P_i = F_i \oplus B_i.$$

$F_0 = 0$, and symbols $\oplus$ and $\otimes$ represent addition and multiplication in finite field $\mathbf{F}2^{64}$

eSTREAM projekt

4 éves (2004-2008) EU kutatás:

"to identify new stream ciphers that might become suitable for widespread adoption"

Előzménye: Megelőző NESSIE projekt (2000-2003) nem eredményezett új adatfolyam rejtjelezőt

két kategória:

- gyors sw implementáció
- korlátozott képességű hw-re

Eredmény:

sw:

- HC-128
- Rabbit
- Salsa20/12
- SOSEMANUK

hw:

- Grain
- MICKEY
- Trivium

TRIVIUM

for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

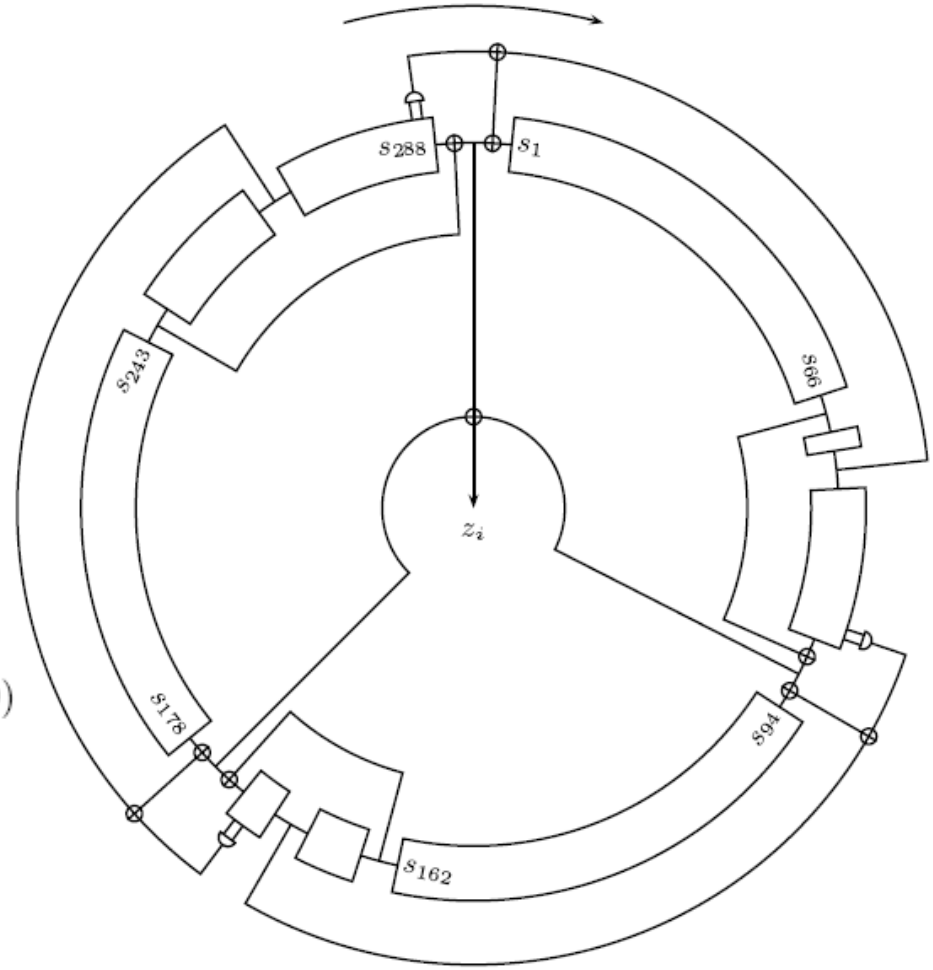
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for



$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Parameters	
Key size:	80 bit
IV size:	80 bit
Internal state:	288 bit

Salsa20

eSTREAM-nek benyújtotta Daniel J. Bernstein 2005-ben

állapotvektor: 16 db 32 bites word

4 (párhuzamosítható) körben, egyszerre 4 wordre:

⊕: bitwise exclusive OR

⊞: 32-bit addition mod 2^{32}

<<<: bit rotáció

Egy változatát (ChaCha20) a Google használja: Android, Chrome