Bahan kuliah IF4020 Kriptografi

Review Beberapa Block Cipher dan Stream Cipher
(Bagian 5: RC4 dan A5)

Oleh: Dr. Rinaldi Munir

Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
ITB

RC4

RC4

- Termasuk ke dalam *cipher* alir (*stream cipher*)
- Dibuat oleh Ron Rivest (1987) dari Laboratorium *RSA*
- *RC* adalah singkatan dari *Ron's Code*. Versi lain mengatakan *Rivest Cipher* .
- Digunakan di dalam sistem keamanan seperti:

  - protokol *SSL* (*Secure Socket Layer*).

  - WEP (Wired Equivalent Privacy)

  - WPA (Wi-fi Protect Access) untuk nirkabel

- RC4 awalnya rahasia dagang
- Pada September 1994, deskripsi RC4 dikirim secara anonim ke milis *Cypherpunks.* Lalu dikirim ke *newsgroup* `sci.crypt` dan menyebar di internet.

- Kode programnya berhasil di-r*eengineering* dan dikonfirmasi sama dengan kode program aslinya.

- Karena telah diketahui orang, RC4 bukan lagi rahasia dagang

- Status sekarang, implementasi tidak resmi adalah legal, tapi tidak boleh menggunakan nama RC4. Maka digunakan nama ARCFOUR untuk menghindari masalah *trademark*.

- *RC4* membangkitkan kunci-alir (*keystream*) dalam ukuran byte setiap kalinya, yang kemudian di-XOR-kan dengan karakter plainteks

- Jadi, *RC4* memproses data dalam ukuran *byte*, bukan dalam bit.

- Untuk membangkitkan kunci-alir, *cipher* menggunakan status internal yang terdiri dari:
  - Permutasi angka 0 sampai 255 di dalam larik S_0, S_1, …, S_{255}. Permutasi merupakan fungsi dari kunci *K* dengan panjang variabel.
  - Dua buah pencacah indeks, *i* dan *j*

- RC4 terdiri dari dua sub-proses:
  1. *Key-Scheduling Algorithm* (*KSA*)
  2. *Pseudo-random generation algorithm* (*PRGA*).

Key-Scheduling Algorithm (*KSA*)

1. Inisialisasi larik S: $S_0 = 0$, $S_1 = 1$, …, $S_{255} = 255$

```
for i ← 0 to 255 do
   S[i] ← i
end
```

0	1	2	3	4	5	6	7	…	…	…	252	253	254	255
0	1	2	3	4	5	6	7	…	…	…	252	253	254	255

2. Lakukan pengacakan (permutasi) nilai-nilai di dalam larik *S* berdasarkan kunci eksternal K:

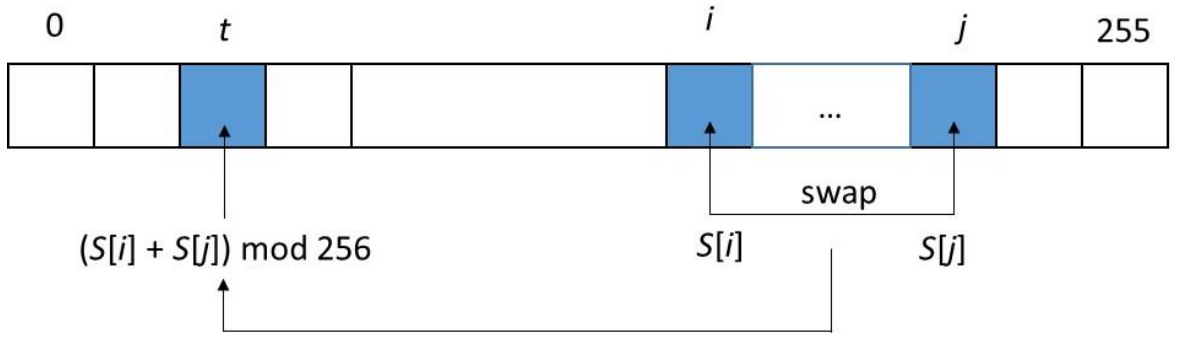
```
j ← 0
for i ← 0 to 255 do
  j ← (j + S[i] + K[i  mod Length(K)]) mod 256
  swap(S[i], S[j]) {* Pertukarkan S[i] & S[j] *}
end
```

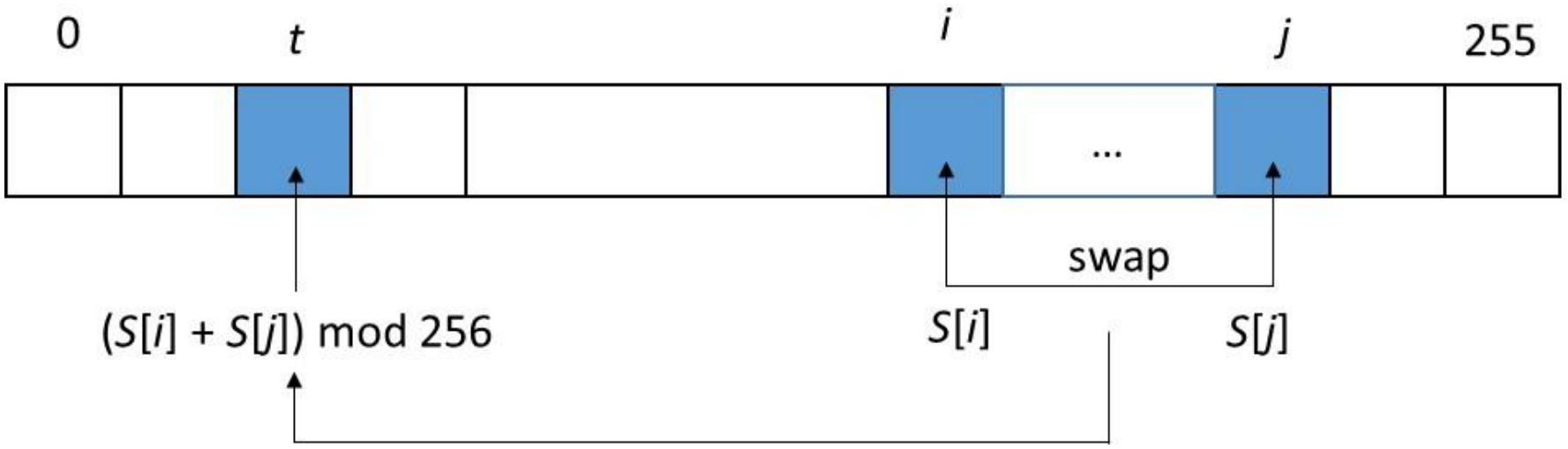
- Pemutasi ini menyebabkan elemen-elemen di dalam larik *S* teracak.

Pseudo-random generation algorithm (*PRGA*)

- *PRGA* membangkitkan kunci-alir (*keystream*) dengan cara mengambil nilai *S*[*i*] dan *S*[*j*], mempertukarkannya, lalu menjumlahkan keduanya dalam modulus 256.
- Kunci alir tersebut kemudian di-XOR-kan dengan sebuah karakter plainteks.

```
i ← 0
j ← 0
for idx ← 0 to Length(P) – 1 do
    i ← (i + 1) mod 256
    j ← (j + S[i]) mod 256
    swap(S[i], S[j])    {* Pertukarkan nilai S[i] dan S[j] *}
    t ← (S[i] + S[j]) mod 256
    u ← S[t]            (* keystream *)
    c ← u ⊕ P[idx]      { enkripsi }
end
```




0
t
i
j
255
...
swap
(S[i] + S[j]) mod 256
S[i]
S[j]

Keamanan RC4

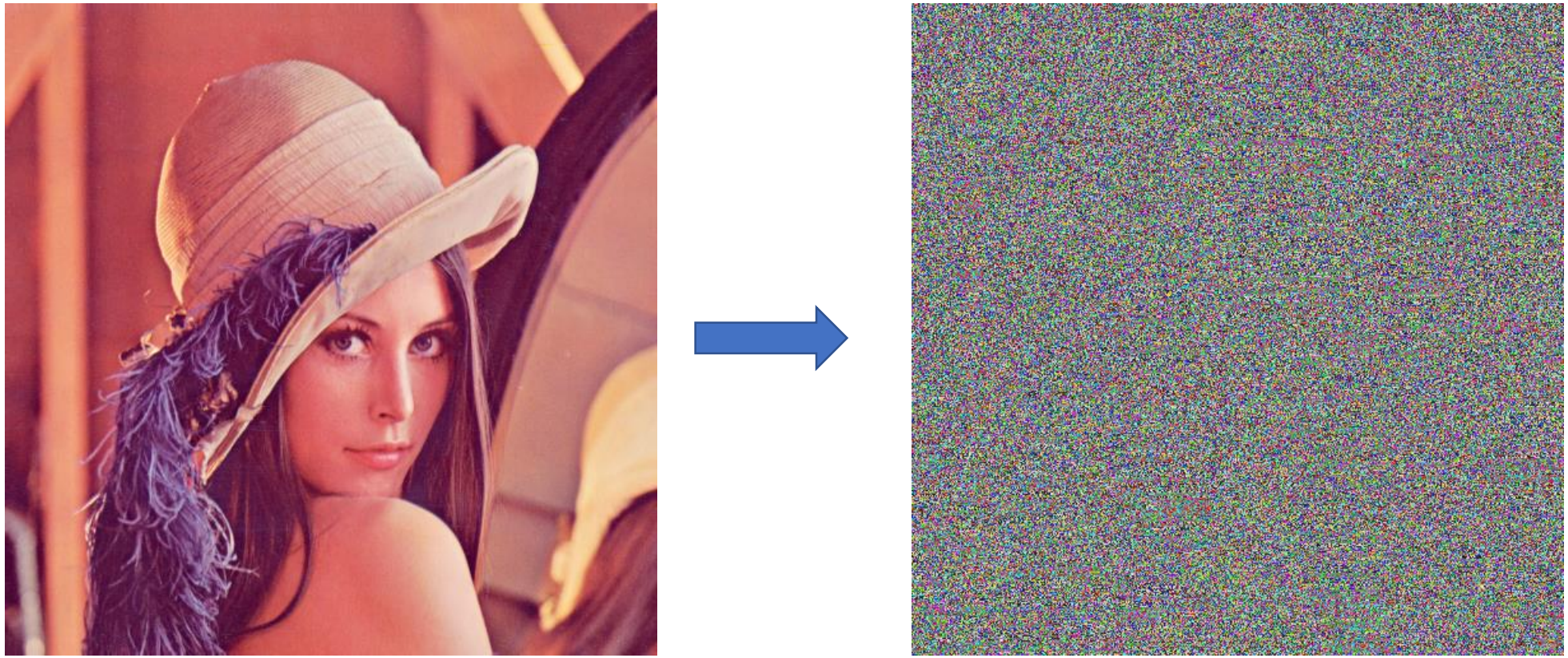
- RC4 adalah *cipher* alir, maka ia tidak kuat terhadap serangan seperti *flip-bit attack* maupun serangan-serangan *stream attack* lainnya.

- Saat ini keamanan RC4 sudah berhasil dipecahkan dalam hitungan jam atau hari.

- Pada bulan Februari 2015, RC4 dilarang penggunaannya di dalam *Transport Layer Security* (TLS) seperti disebutkan di dalam RFC 7465 (https://tools.ietf.org/html/rfc7465).

Aplikasi:

1. Cocok untuk enkripsi berkas citra (derajat keabuan 0 – 255)



2. Cocok untuk enkripsi *record* atau *field* basis data (karena ukuran cipherteks = ukuran plainteks)

siswa.dbf:

NIM	Nama	Tinggi	Berat
000001	Elin Jamilah	160	50
000002	Fariz RM	157	49
000003	Taufik Hidayat	176	65
000004	Siti Nurhaliza	172	67
000005	Oma Irama	171	60
000006	Aziz Burhan	181	54
000007	Santi Nursanti	167	59
000008	Cut Yanti	169	61
000009	Ina Sabarina	171	62

Siswa2.dbf:

NIM	Nama	Tinggi	Berat
j@wqpu	4#0(jkTBfKgv%	Tv#	*D
000002	Fariz RM	157	49
&^gRdn	jHbVtf%z!0(klTb	Nb%	Kv
000004	Siti Nurhaliza	172	67
000005	Oma Irama	171	60
000006	Aziz Burhan	181	54
000007	Santi Nursanti	167	59
Knb5$c	jBtv^0)!hJT	9b6	YT
000009	Ina Sabarina	171	62

A5

- *A5*: *cipher* alir yang digunakan untuk mengenkripsi transmisi sinyal percakapan dari standard telepon seluler *GSM* (*Group Special Mobile*).

- Sinyal *GSM* dikirim sebagai barisan *frame*. Satu *frame* panjangnya 228 bit dan dikirim setiap 4,6 milidetik.

- *A5* digunakan untuk untuk menghasilkan kunci-alir (*keystream*) 228-bit yang kemudian di-*XOR*-kan dengan *frame*. Kunci eksternal (*session key*) panjangnya 64 bit.

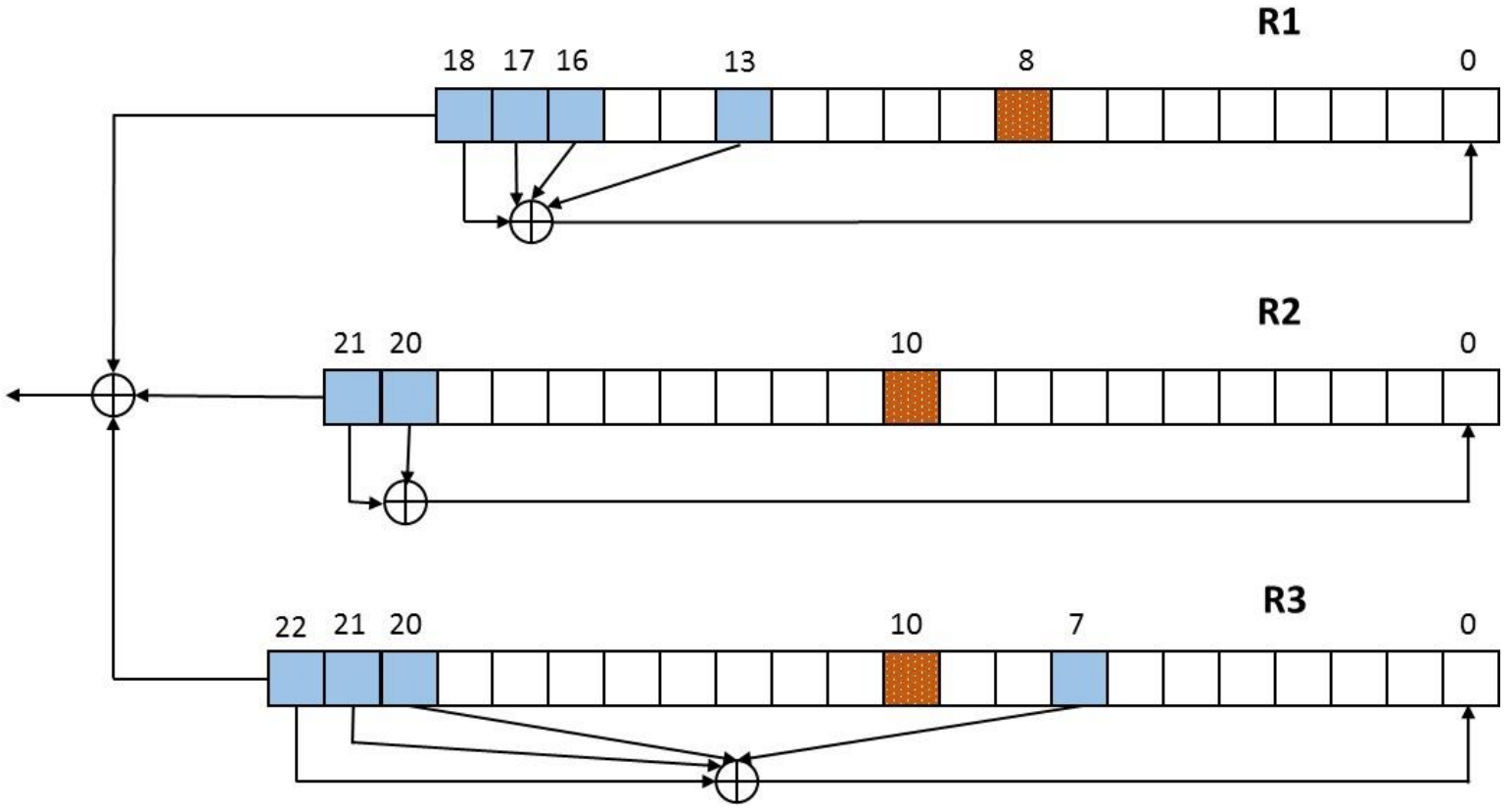
- GSM merupakan standard telepon seluler Eropa. A5 dikembangkan oleh Perancis

- Tidak semua operator GSM mengimplementasikan enkripsi, bergantung regulasi di negara tersebut (seperti di Indonesia)

- A5 ada dua versi:
    1. A5/1 : versi kuat A5, digunakan di Eropa
    2. A5/2 (Kasumi) : versi ekspor, lebih lemah

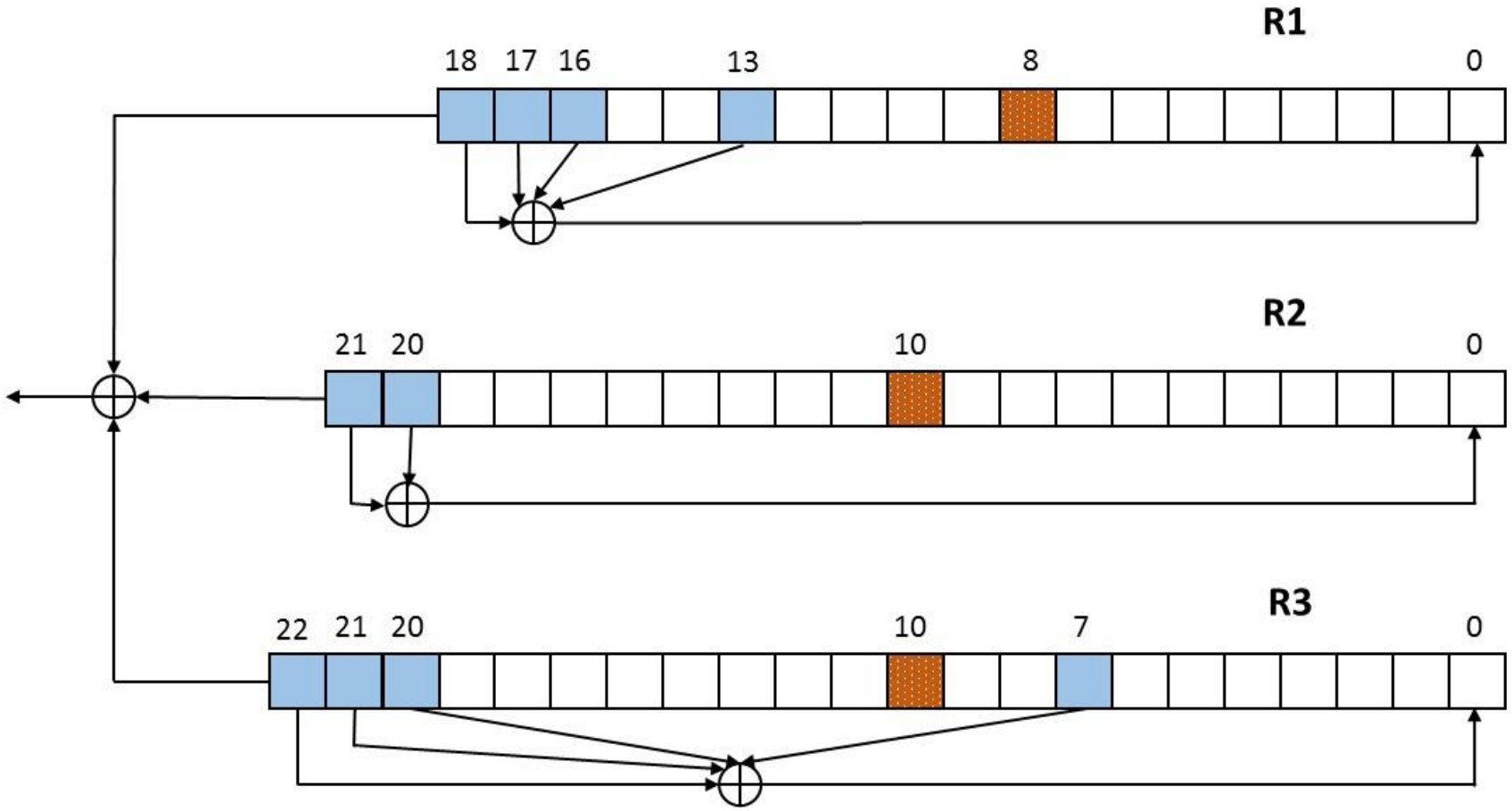
- Algoritma A5/1 pada awalnya rahasia, tetapi pada tahun 1994 melalui *reverse engineering*, algoritmanya terbongkar.

- *A5* terdiri dari 3 buah *LFSR* , masing-masing panjangnya 19, 22, dan 23 bit (total = 19 + 22 + 23 = 64). Luaran (*output*) dari *A5* adalah hasil *XOR* dari ketiga buah *LFSR* ini.



- Bit-bit di dalam register diindeks dimana bit paling tidak penting (*LSB*) diindeks dengan 0 (elemen paling kanan).

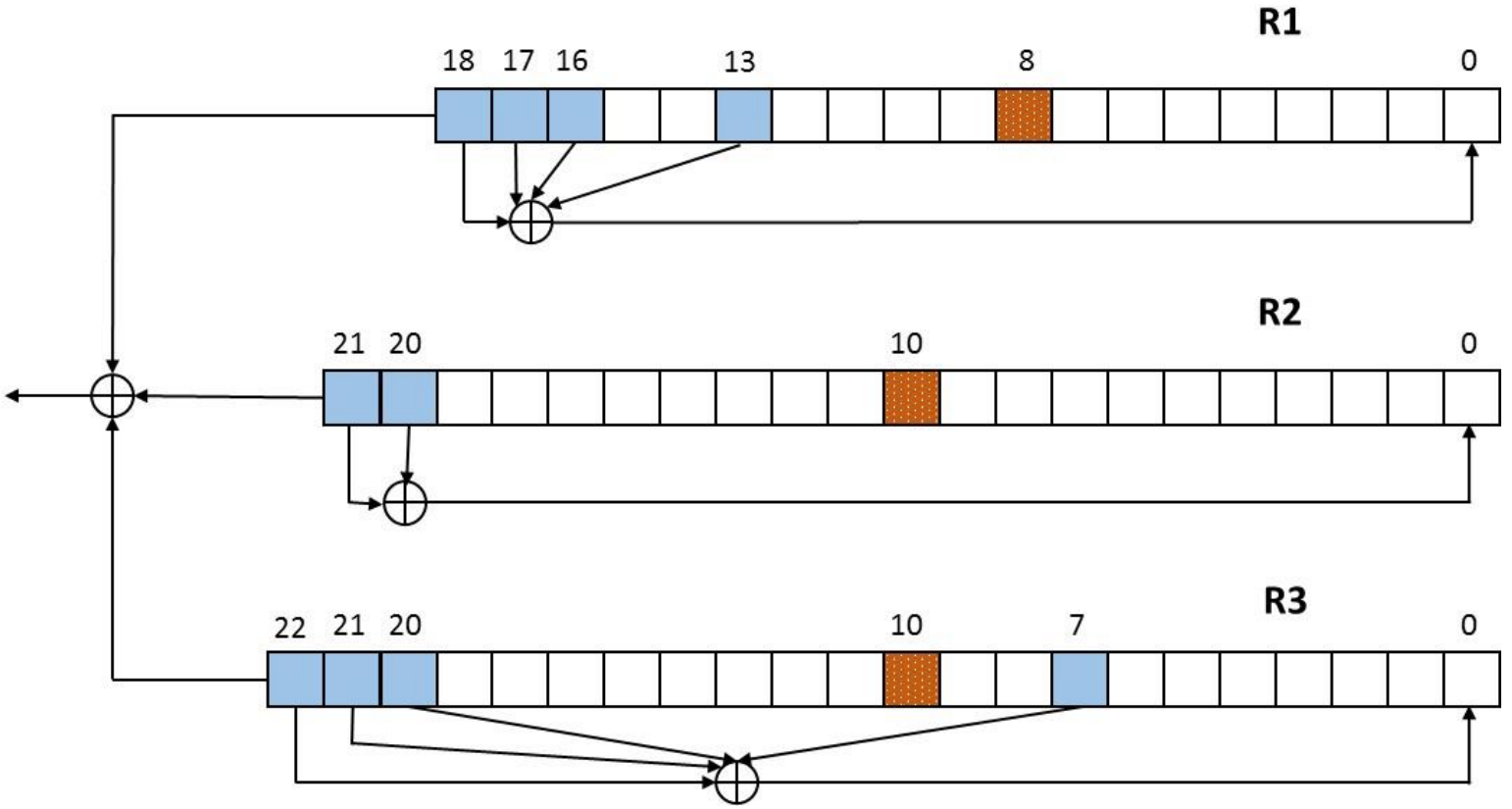
- *A5* menggunakan tiga buah kendali detak (*clock*) yang variabel:
  - Bit ke-8 pada register 1. Bit-bit detak pada bit 13, 16, 17, dan 18
  - Bit ke-10 pada register 2. Bit-bit detaknya adalah pada bit 20 dan 21
  - Bit ke-10 pada register 3. Bit-bit detaknya adalah pada bit 7, 20, 21, dan 22



- Setiap register didetak dalam fase *stop/go* dengan menggunakan kaidah mayoritas:

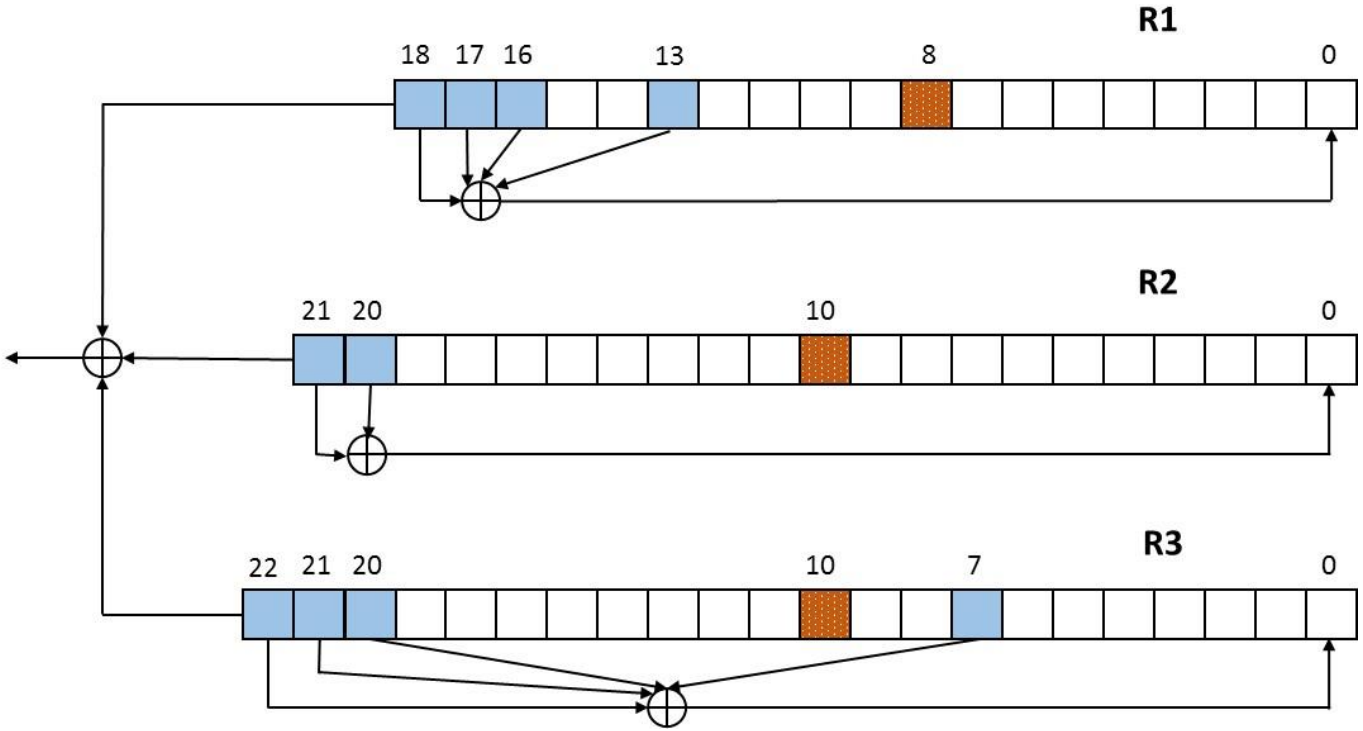
  "Pada setiap putaran (*i*=1..64), bit-bit tengah setiap register diperiksa dan bit mayoritasnya (50% + 1) ditentukan. Jika bit tengah sebuah register sama dengan bit mayoritas, maka register tersebut didetak"

.



- Ketika sebuah register didetak, semua bit detaknya di-XOR-kan dan hasilnya diletakkan pada posisi LSB (posisi ke-0) dengan mekanisme pergeseran bit-bit ke kiri. (Biasanya pada setiap putaran dua atau tiga buah register didetak. Peluang sebuah register didetak pada setiap putaran adalah ¾)

- Bit paling kiri (MSB) terlempar keluar. Bit yang terlempar dari setiap register di-XOR-kan bersama, bit inilah yang menjadi luaran dari ketiga buah register tadi.



Proses pembangkitan bit-bit acak di dalam A5/1 berdasarkan kunci sesi *K* adalah sebagai berikut:

1. Ketiga register pada awalnya diinisialisasi seluruh bitnya dengan 0. Selanjutnya dilakukan 64 putaran pertama, 64 bit kunci sesi *K* dicampur dengan bit register berdasarkan skema berikut:

   pada putaran $0 \leq i < 64$, bit $K[i]$ ditambahkan ke bit *LSB* dari setiap register *R* dengan menggunakan operasi *XOR*:

   $$R[0] = R[0] \oplus K[i]$$

2. Selanjutnya, ketiga register didetak selama 22 putaran tambahan. Selama 22 putaran tersebut, 22-bit dari nomor *frame* (F_n) dicampur dengan bit register berdasarkan skema berikut:

   pada putaran $0 \leq i < 22$, bit $F_n[i]$ ditambahkan ke bit *LSB* dari setiap register *R* dengan menggunakan operasi *XOR*:

   $$R[0] = R[0] \oplus F_n[i]$$

Isi register pada akhir putaran menyatakan kondisi awal untuk pembangkitan *frame* sepanjang 228-bit.

3. Ketiga register didetak selama 100 putaran dalam fase *stop/go* dengan menggunakan kaidah mayoritas, namun bit-bit luarannya dibuang (tidak dipakai).

4. Selanjutnya, ketiga register didetak selama 228 putaran dalam fase *stop/go* menggunakan kaidah mayoritas untuk menghasilkan bit-bit kunci-alir sepanjang 228 bit.

   Pada setiap putaran dihasilkan 1 bit yang merupakan hasil XOR bit-bit MSB yang terlempar.

   Kunci-alir 228-bit inilah yang kemudian digunakan untuk mengenkripsi *frame* pesan sepanjang 228 bit.

- Algoritma A5/1 telah digunakan untuk mengenkripsi semua percakapan dan komunikasi data melalui telepon seluler GSM di Eropa.

- Program A5/1 ditanam di dalam *chip* pada kartu *Simcard*.

- Di negara-negara di mana operator telepon seluler dilarang melakukan enkripsi percakapan, seperti di Indonesia, maka program algoritma A5 tidak diaktifkan (*disabled*), sehingga semua sinyal terkirim dalam bentuk plainteks.

Keamanan A5

- Keamanan A5/1 terletak pada pembangkitan bit-bit acak yang dihasilkan oleh tiga buah LFSR di atas.

- Tujuan kriptanalisis A5/1 adalah untuk menemukan kunci sesi yang digunakan dalam pembangkitan bit-bit acak.

- Dalam serangan tersebut diasumsikan penyerang mengetahui luaran A5/1 pada periode awal komunikasi, untuk selanjutnya menemukan kunci sesi yang digunakan untuk mendekripsi sisa komunikasi selanjutnya.

- Serangan yang dilakukan terhadap A5/1 adalah *known-plaintext attack*.

- Serangan semacam ini pertama kali dilakukan oleh Ross Anderson pada tahun 1994.

- Ross Anderson mencoba menerka 42 bit isi register R1 dan R2, dan menurunkan 23 bit R3 dari 42 bit tersebut. Pekerjaan menemukan kunci tersebut dilakukan pada komputer PC dan membutuhkan waktu komputasi lebih dari sebulan.

- Pada tahun-tahun selanjutnya, para kriptanalis berhasil menemukan kunci hanya dalam waktu beberapa menit saja.

- Secara umum, A5/1 sudah berhasil dikriptanalisis.

TAMAT