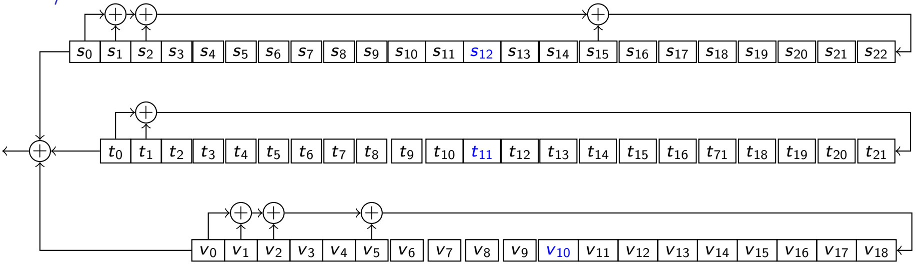
Practical use of LFSRs

Tanja Lange

Eindhoven University of Technology

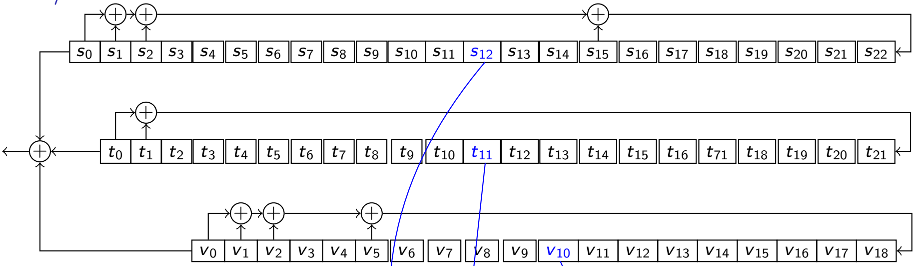
2WF80: Introduction to Cryptology

A5/1



- A5/1 was standardized for GSM, still used in 2G.
- 3 LFSRs with primitive characteristic polynomials: $x^{23} + x^{15} + x^2 + x + 1$, $x^{22} + x + 1$, and $x^{19} + x^5 + x^2 + x + 1$.

A5/1



- A5/1 was standardized for GSM, still used in 2G.
- 3 LFSRs with primitive characteristic polynomials: $x^{23}+x^{15}+x^2+x+1$, $x^{22}+x+1$, and $x^{19}+x^5+x^2+x+1$.
- Achieves some nonlinearity by
  - checking the values of s_{12}, t_{11}, and v_{10},
  - advancing only the LFSRs for which these check bits agree with the majority of the check bits.
- This means that at least 2 LFSRs advance per step.
- 64 key bits, but 10 set to 0.

A5/1 – details

- GSM uses 22-bit frame numbers ($\approx$ IV).
- The LFSRs are fixed, so where do the key bits go?

A5/1 – details

- GSM uses 22-bit frame numbers ($\approx$ IV).
- The LFSRs are fixed, so where do the key bits go?
- Run key setup with key k and frame number f.
  1. Initialize all registers to 0: $R_1 = R_2 = R_3 = 0$.
  2. for i = 0 to 63:
     clock all three registers (this advances all of them)
     $R_1[22] = R_1[22] + k[i]$; $R_2[21] = R_2[21] + k[i]$; $R_3[18] = R_3[18] + k[i]$.
  3. for i = 0 to 21
     clock all three registers (this advances all 3)
     $R_1[22] = R_1[22] + f[i]$; $R_2[21] = R_2[21] + f[i]$; $R_3[18] = R_3[18] + f[i]$.
- Run A5/1 for 100 cycles and discard the output.
  This uses clocking by s_{12}, t_{11}, and v_{10},
- Run A5/1 for 228 cycles and use the output as keystream.
  This uses clocking by s_{12}, t_{11}, and v_{10},

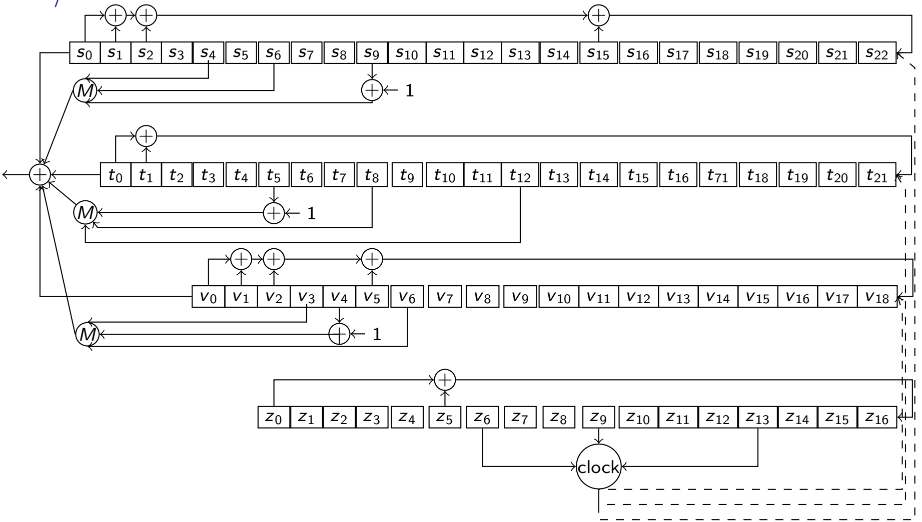
A5/1 – details

- GSM uses 22-bit frame numbers ($\approx$ IV).
- The LFSRs are fixed, so where do the key bits go?
- Run key setup with key k and frame number f.
  1. Initialize all registers to 0: $R_1 = R_2 = R_3 = 0$.
  2. for i = 0 to 63:
     clock all three registers (this advances all of them)
     $R_1[22] = R_1[22] + k[i]$; $R_2[21] = R_2[21] + k[i]$; $R_3[18] = R_3[18] + k[i]$.
  3. for i = 0 to 21
     clock all three registers (this advances all 3)
     $R_1[22] = R_1[22] + f[i]$; $R_2[21] = R_2[21] + f[i]$; $R_3[18] = R_3[18] + f[i]$.
- Run A5/1 for 100 cycles and discard the output.
  This uses clocking by s_{12}, t_{11}, and v_{10},
- Run A5/1 for 228 cycles and use the output as keystream.
  This uses clocking by s_{12}, t_{11}, and v_{10},
- The design was kept secret, though partially revealed already in 1994 by Ross Anderson; fully reverse engineered by Marc Briceno, Ian Goldberg, and David Wagner, who cryptanalyzed it and posted a readable implementation.

A5/1 – details

- GSM uses 22-bit frame numbers ($\approx$ IV).
- The LFSRs are fixed, so where do the key bits go?
- Run key setup with key k and frame number f.
  1. Initialize all registers to 0: $R_1 = R_2 = R_3 = 0$.
  2. for i = 0 to 63:
     clock all three registers (this advances all of them)
     $R_1[22] = R_1[22] + k[i]$; $R_2[21] = R_2[21] + k[i]$; $R_3[18] = R_3[18] + k[i]$.
  3. for i = 0 to 21
     clock all three registers (this advances all 3)
     $R_1[22] = R_1[22] + f[i]$; $R_2[21] = R_2[21] + f[i]$; $R_3[18] = R_3[18] + f[i]$.
- Run A5/1 for 100 cycles and discard the output.
  This uses clocking by s_{12}, t_{11}, and v_{10},
- Run A5/1 for 228 cycles and use the output as keystream.
  This uses clocking by s_{12}, t_{11}, and v_{10},
- The design was kept secret, though partially revealed already in 1994 by Ross Anderson; fully reverse engineered by Marc Briceno, Ian Goldberg, and David Wagner, who cryptanalyzed it and posted a readable implementation.
- Latest attack cost: 2^{24}; given 3 – 4 min of ciphertext or even less ciphertext, more computer power

A5/2



- A5/2 used for export control, weakened version of A5/1.
- 4th LFSRs is used to clock the other three.
  Extra inputs into output sum use majority function of bits.

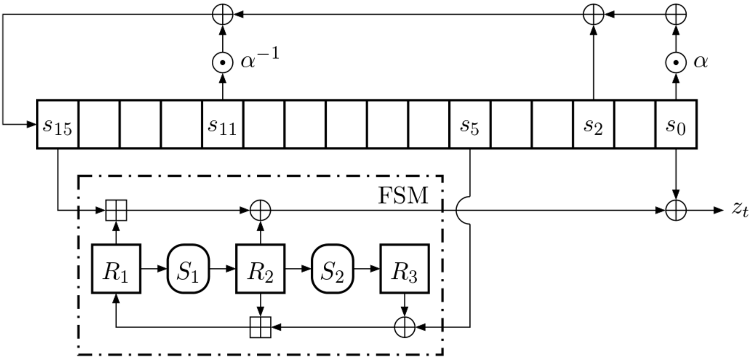
A5/2 – details

- k and f used in manner similar to A5/1 (also filling in R_4).
- Clock is controlled by 3 bits of R_4;
  R_i is advanced if i-th control bit matches majority.
- Design looks a lot more convoluted than A5/1, yet the cipher is weaker.
- Reversed and broken by Briceno, Goldberg, Wagner in 1999.

A5/2 – details

- k and f used in manner similar to A5/1 (also filling in R_4).
- Clock is controlled by 3 bits of R_4;
  R_i is advanced if i-th control bit matches majority.
- Design looks a lot more convoluted than A5/1, yet the cipher is weaker.
- Reversed and broken by Briceno, Goldberg, Wagner in 1999.
- Now broken instantly (in 2^{16} steps) by Barkan, Biham, and Keller.
- Downgrade from A5/1 was possible.
- Publicly available tables of precomputation exist.

SNOW-3G



- SNOW-3G is used in 3G communication.
- Upper part is LFSR with elements of $\mathbb{F}_{2^{32}}$; i.e., $\alpha \in \mathbb{F}_{2^{32}}$ is fixed.
- The bottom part forgets about the field structure:
  $\boxplus$ is integer addition modulo 2^{32},
  $\oplus$ is bitwise addition (matching addition in $\mathbb{F}_{2^{32}}$).
- R_1, R_2, R_3 are registers, S_1, S_2 are 32-bit to 32-bit substitution boxes.

Picture from https://www.cryptolux.org/index.php/File:SNOW-3G.png.

Wrapping up

- LFSRs are typical ingredients of hardware ciphers.
- LFSRs require some non-linear component to be secure.
  The typical attack models assume some access to keystream; ciphertext-only attacks have direct practical relevance.
- Many old designs had some "security by obscurity" and crumbled once description was known.
- See State of the Art in Lightweight Symmetric Cryptography by Alex Biryukov and Léo Perrin for a good overview.
  It mostly covers modern, not broken designs.
  Table 3 shows how much security has degraded for legacy designs:

Name	Intended platform	Key	IS	IV	Att. time	Reference
A5/1	Cell phones	64	64	22	2^{24}	[And94]
A5/2		64	81	22	2^{16}	[BBK08]
CMEA †		64	16–48	–	2^{32}	[WSK97]
ORYX		96	96	–	2^{16}	[WSD+99]
A5-GMR-1	Satellite phones	64	82	19	$2^{38.1}$	[DHW+12]
A5-GMR-2		64	68	22	2^{28}	[DHW+12]
DSC	Cordless phones	64	80	35	2^{34}	[LST+09]
SecureMem.	Atmel chips	64	109	128	$2^{29.8}$	[GvRVWS10]
[illegible]		64	[illegible]	128	2^{50}	