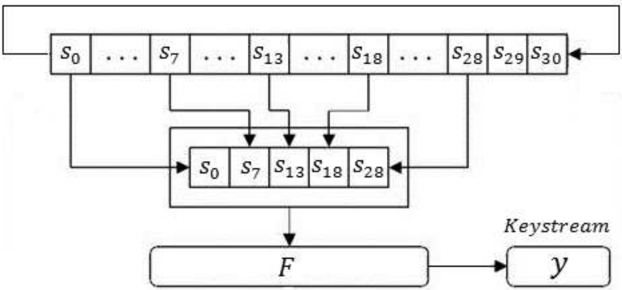
Optimizing the placement of tap positions

Samir Hodžić
joint work with
Enes Pasalic, Samed Bajrić and Yongzhuang Wei



- Linear feedback shift register (**LFSR**).
- Nonlinear filtering function $F : GF(2)^n \rightarrow GF(2)^m$, whose inputs are taken from **Tap positions** of register.

Outputs of F are keystream blocks $\mathbf{y^t} = (y_1^t, \ldots, y_m^t)$.

Attacks?

Different properties of Boolean function vs different attacks:

- Algebraic degree and resiliency vs Berlekamp–Massey synthesis algorithm and Correlation attacks.
- Algebraic immunity vs Algebraic attacks (Fast algebraic attacks, Probabilistic algebraic attacks).
- Filter state guessing attack (**FSGA**).
- and others...

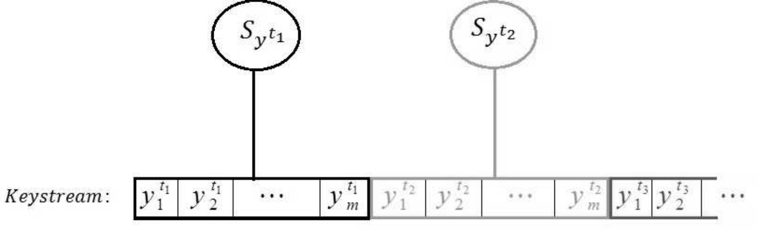
What about **tap positions**, can we use these in an attack ?

Filter state guessing attack (FSGA)

- Observe several outputs $y^{t_1}, \ldots, y^{t_c}$ so that $c \times n > L$, where L is length of LFSR.
- Look at the preimage space

$$S_y = \{x \in GF(2)^n \ : \ F(x) = y\}$$

- Given any output y^{t_u} there is 2^{n-m} possibilities for input $(x_1^{t_u}, \ldots, x_n^{t_u})$, where $x_i^{t_u} = \sum_{j=0}^{L-1} a_{i,j}^{t_u} s_j$ (**linear equation**)
- Solve linear system and check whether the solution is correct.



Regarding the preimage spaces, it may happen that

$$x_j^{t_1} \rightarrow x_k^{t_2}$$

and preimage space reduces...

Design should prevent from finding many $x_j^{t_1} \rightarrow x_k^{t_2}, \quad x_u^{t_1} \rightarrow x_v^{t_2}$

Generalized Filter state guessing attack (GFSGA)

Unlike FSGA, **GFSGA** (Y. Wei *et al.* '11) utilizes the tap positions!

- The outputs $y^{t_1}, \ldots, y^{t_c}$ may give identical equations
- **Distance** between the consecutive outputs is σ.
- If $\mathcal{I}_0 = \{i_1, i_2, \ldots, i_n\}$ is the set of tap positions, then

$$r_i = \#\mathcal{I}_i, \qquad r_i - \textit{ number of repeated bits per state},$$

$$\mathcal{I}_i = \mathcal{I}_{i-1} \cup \{\mathcal{I}_0 \cap \{i_1 + i\sigma, i_2 + i\sigma, \ldots, i_n + i\sigma\}\}.$$

Satisfying $nc - R > L$, the total number of repeated equations R:

- If $c \leq k$: $R = \sum_{i=1}^{c-1} r_i$
- If $c > k$: $R = \sum_{i=1}^{k} r_i + (c-k-1)r_k$, where $k = \lfloor \frac{i_n - i_1}{\sigma} \rfloor$.

Complexities of the attack in both cases:

$$T_{Comp.}^{c \leq k} = 2^{(n-m)} \times 2^{(n-m-r_1)} \times \ldots \times 2^{(n-m-r_{(c-1)})} \times L^3.$$
$$T_{Comp.}^{c > k} = 2^{(n-m)} \times \ldots \times 2^{(n-m-r_k)} \times 2^{(n-m-r_k)\times(c-k-1)} \times L^3.$$

Problem: How to maximize $T_{Comp.}$ for **any** σ?

Designer/attacker rationales

In the position of the **attacker**:

- Search for **optimal** σ that gives minimal $T_{Comp.}$!

Q1: What about parameters R and c in the formula

$$T_{Comp.} = 2^{(n-m)c-R} \times L^3?$$

A1: For a given set of taps $\mathcal{I}_0 = \{i_1, i_2, \ldots, i_n\}$, (not optimally taken?) the step σ which results in maximal R **does not imply** minimal complexity!

Our approach...

Can we calculate R in a different way? Can we get some new information ?

Example: Let $\mathcal{I}_0 = \{i_1, i_2, i_3, i_4, i_5\} = \{1, 4, 8, 9, 11\}$, $L = 15$ and $\sigma = 2$.

We adopt the notation:

- For easier tracking of repeated bits in LFSR states, we use the notation $s_k \rightarrow (k+1)$.
- We consider only bits on tap positions on states which differ for σ.

States	i_1	i_2	i_3	i_4	i_5
$\mathbf{s}^{t_1}$	$s_0 \rightarrow 1$	$s_3 \rightarrow 4$	$s_7 \rightarrow 8$	$s_8 \rightarrow 9$	$s_{10} \rightarrow 11$
$\mathbf{s}^{t_2}$	$s_2 \rightarrow 3$	$s_5 \rightarrow 6$	$s_9 \rightarrow 10$	$s_{10} \rightarrow 11$	$s_{12} \rightarrow 13$
$\mathbf{s}^{t_3}$	$s_4 \rightarrow 5$	$s_7 \rightarrow 8$	$s_{11} \rightarrow 12$	$s_{12} \rightarrow 13$	$s_{14} \rightarrow 15$
$\mathbf{s}^{t_4}$	$s_6 \rightarrow 7$	$s_9 \rightarrow 10$	$s_{13} \rightarrow 14$	$s_{14} \rightarrow 15$	$s_{16} \rightarrow 17$
$\mathbf{s}^{t_5}$	$s_8 \rightarrow 9$	$s_{11} \rightarrow 12$	$s_{15} \rightarrow 16$	$s_{16} \rightarrow 17$	$s_{18} \rightarrow 19$
$\mathbf{s}^{t_6}$	$s_{10} \rightarrow 11$	$s_{13} \rightarrow 14$	$s_{17} \rightarrow 18$	$s_{18} \rightarrow 19$	$s_{20} \rightarrow 21$
$\mathbf{s}^{t_7}$	$s_{12} \rightarrow 13$	$s_{15} \rightarrow 16$	$s_{19} \rightarrow 20$	$s_{20} \rightarrow 21$	$s_{22} \rightarrow 23$
$\mathbf{s}^{t_8}$	$s_{14} \rightarrow 15$	$s_{17} \rightarrow 18$	$s_{21} \rightarrow 22$	$s_{22} \rightarrow 23$	$s_{24} \rightarrow 25$
$\mathbf{s}^{t_9}$	$s_{16} \rightarrow 17$	$s_{19} \rightarrow 20$	$s_{23} \rightarrow 24$	$s_{24} \rightarrow 25$	$s_{26} \rightarrow 27$
$\mathbf{s}^{t_{10}}$	$s_{18} \rightarrow 19$	$s_{21} \rightarrow 22$	$s_{25} \rightarrow 26$	$s_{26} \rightarrow 27$	$s_{28} \rightarrow 29$

Questions: When will bit from tap position i_3 repeat on i_1? Will ever repeat? If yes, in how many states?

States	i_1	i_2	i_3	i_4	i_5
$\mathbf{s}^{t_1}$	1	4	8	9	11
$\mathbf{s}^{t_2}$	3	6	10	11	13
$\mathbf{s}^{t_3}$	5	8	12	13	15
$\mathbf{s}^{t_4}$	7	10	14	15	17
$\mathbf{s}^{t_5}$	9	12	16	17	19
$\mathbf{s}^{t_6}$	11	14	18	19	21
$\mathbf{s}^{t_7}$	13	16	20	21	23
$\mathbf{s}^{t_8}$	15	18	22	23	25
$\mathbf{s}^{t_9}$	17	20	24	25	27
$\mathbf{s}^{t_{10}}$	19	22	26	27	29

We define the set of differences (from $\mathcal{I}_0 = \{1, 4, 8, 9, 11\}$) between the consecutive tap positions as

$$D = \{ \ d_j \mid d_j = i_{j+1} - i_j, \ \ j = 1, 2, 3, 4\} = \{3, 4, 1, 2\}.$$

Regarding the non-consecutive differences, we construct the **scheme of differences**:

Row\Columns	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	d_1	d_2	d_3	d_4
Row 2	$d_1 + d_2$	$d_2 + d_3$	$d_3 + d_4$	
Row 3	$d_1 + d_2 + d_3$	$d_2 + d_3 + d_4$		
Row 4	$d_1 + d_2 + d_3 + d_4$			

In our example, the scheme of differences is given as

Row\Columns	Col. 1	Col. 2	Col. 3	Col. 4
Row 1	3	4	1	2
Row 2	7	5	3	
Row 3	8	7		
Row 4	10			

Total sum of all repeated bits on all tap positions is given as

$$R = \sum_{i=1}^{n-1} (c - \frac{1}{\sigma} \sum_{k=i}^{m} d_k),$$

where $\sigma \mid \sum_{k=i}^{m} d_k$ for some $m \in \mathbb{N}$, $i \leq m \leq n-1$.

Further analysing

From the previous formula, the complexity will increase if

1. We maximize $\sum_{k=i}^{m} d_k$, and
2. Avoid the divisibility by σ in the table of differences.

It turns out that:

- Maximizing $\sum_{k=i}^{m} d_k$ means to distribute the taps over entire LFSR.
- Regarding the divisibility, what about prime numbers?

Suboptimal algorithms

Which differences to choose:

- Prime numbers are still favourable (for many reasons).
- In many cases, we will have to choose the same differences.
- In general choose co-prime numbers. HOW ?

Permutation algorithm:

- **Input:** The set D and the numbers L, n and m.
- **Output:** The best ordering of the chosen differences, that is, an ordered set D that maximizes the complexity of the attack.

Complexity of algorithm is $O(K \cdot n!)$, where K is a constant (large)

Open problem: Find an efficient algorithm, which returns the best ordering of the set D without searching all permutations.

When $\#D$ is large, we give **a modified algorithm** - construct D by parts:

- Choose a starting set (6-7 elements) in its best ordering (use previous algorithm).
- Chose another few elements and find a permutation which fits best to the starting set - maximized complexity.
- **Measuring the quality:** Lower value of optimal σ is a greater indicator than the complexity.
- By putting the parts from right to left, continue the previous steps until you obtain the set D.

Example: Let $L = 160$, $n = 17$ and $m = 6$.

- Starting set in its best ordering $X = \{5, 13, 7, 26, 11, 17\}$
- The second set (part) is $Y_p = \{9, 1, 2, 23, 15\}$ in its best ordering which fits to the set X, i.e. we have

$$Y_pX = \{9, 1, 2, 23, 15, 5, 13, 7, 26, 11, 17\}$$

- The last part in its best ordering is $Z_p = \{5, 11, 4, 3, 7\}$ which fits to the set Y_pX. Finally we get $D = Z_pY_pX$, i.e.

$$D = \{5, 11, 4, 3, 7, 9, 1, 2, 23, 15, 5, 13, 7, 26, 11, 17\}.$$

Since $\sum d_i = 159$, we need to take the first tap to be 1, which implies the last one to be L.

The set of tap positions is given by

$$\mathcal{I}_0 = \{1, 6, 17, 21, 24, 31, 40, 41, 43, 66, 81, 86, 99, 106, 132, 143, 160\}.$$

- Optimal step of the attack is $\sigma = 1$ with complexity $T_{Comp.} \approx 2^{86.97}$.
- Exhaustive search requires 2^{80}.
- In some cases we have a space to increase the number of output bits m, and still preserve the security margins.

SOBER-t32: The tap positions are given by $\mathcal{I}_0 = \{1, 4, 11, 16, 17\}$, and we have $D = \{3, 7, 5, 1\}$.

In GFSGA article, the complexity of the attack is

$$T_D = (17 \times 32)^3 \times 2^{266}.$$

According to the rules for choosing elements and permutation algorithm, we take $D^* = \{5, 2, 7, 2\}$ and we have

$$T_{D^*} = (17 \times 32)^3 \times 2^{291}.$$

SFINX: The set of differences is given as

$$D = \{1, 5, 3, 10, 2, 23, 14, 16, 24, 7, 29, 27, 32, 34, 17, 11\}.$$

Estimated complexity is $T_{Comp.} = 2^{256}$ with $R = 200$ and $\sigma = 2$ as an optimal step of the attack.

Modified algorithm may be used to improve the existing set D.

In its best orderings, we take the following parts:

- $X = \{29, 32, 17, 34, 27, 11\}$, $Y_p = \{2, 23, 14, 16, 24, 7\}$ and $Z_p = \{1, 5, 3, 10\}$.

- Estimated complexity is $T_{Comp.} = 2^{257}$ with $R = 167$, thus only a minor improvement has been achieved.

- We get the set $D^* = Z_p Y_p X$ given as

$$D^* = \{1, 5, 3, 10, 2, 23, 14, 7, 16, 24, 29, 32, 17, 34, 27, 11\},$$

with the optimal steps $\sigma \in \{1, 2\}$ for the attack.

Thanks for your attention!