Two General Attacks on Pomaranch-like Keystream Generators.

H. Englund, M. Hell and T. Johansson

Department of Electrical and Information Technology
Lund University, Sweden

Outline

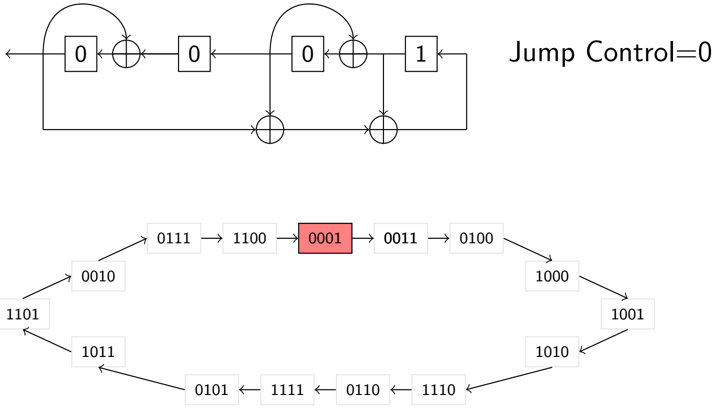
Outline

Outline

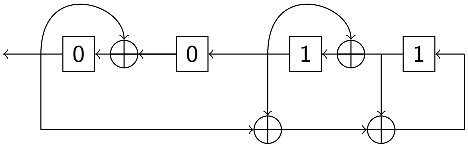
Outline

LUND UNIVERSITY

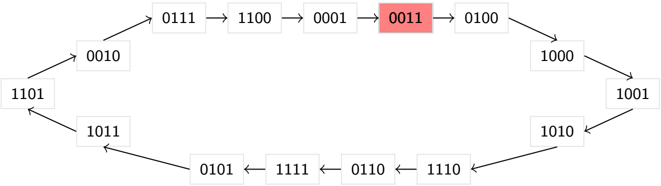
Jump Registers



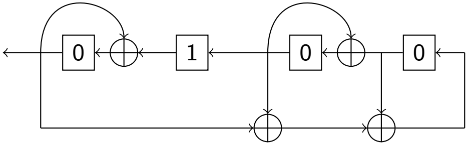
Jump Registers



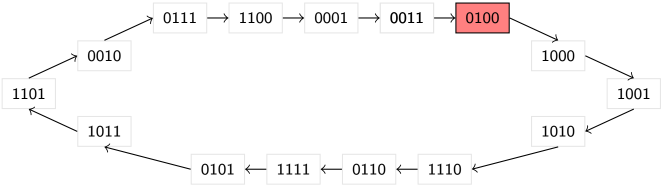
Jump Control=0



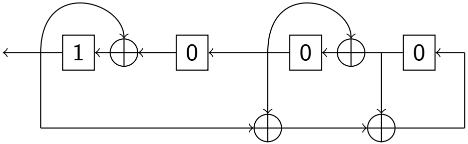
Jump Registers



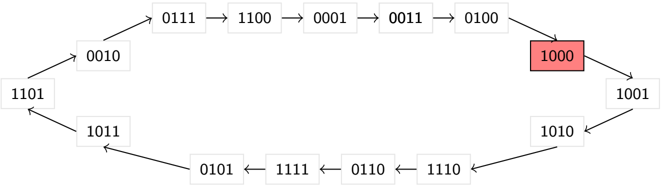
Jump Control=0



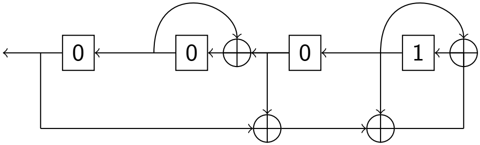
Jump Registers



Jump Control=0

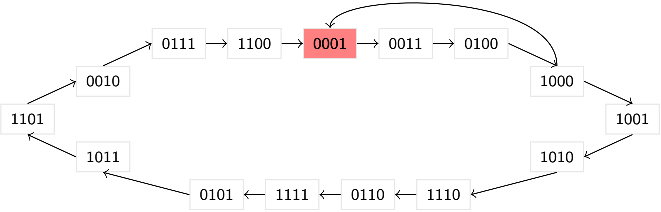


Jump Registers

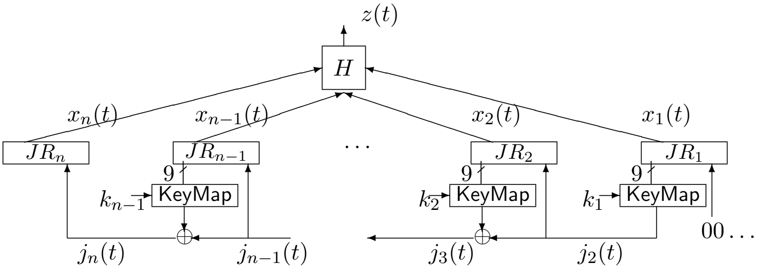


Jump Control=1

Jump Index=12



Description of Pomaranch

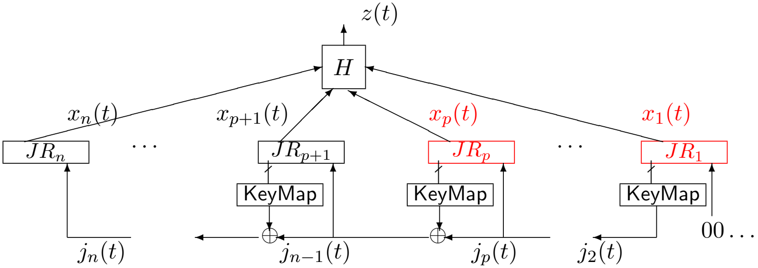


- n jump registers $JR_1, \dots, JR_n$ of length L.
- Jump sequence $j_2, \dots, j_n$ used to clock jump registers.
- KeyMap, key-dependent function, $f : \mathbb{F}_2^9 \mapsto \mathbb{F}_2$.
- Filter function H.

Different Designs

- 5 different proposals for Pomaranch ciphers.
- 3 different design ideas:
  1. Same jump register, linear filter function.
  2. Different jump registers, linear filter function.
  3. Different jump registers, non-linear filter function.

Period of Jump Registers



- Period of JR_1 is denoted T_1, $T_1 = 2^L - 1$.
- Period for register JR_p is $T_p = T_1^p \approx 2^{pL}$.

Hence

$$x_i(t) = x_i(t + T_1^i),\ 1 \leq i \leq p.$$

- Useful for p such that $T_1^p < 2^{|K|}$, $|K|$ =Key size.

Period of Jump Registers

Take samples at time t and $t + T_1^p$:

$$z(t)+z(t+T_1^p) = H\big(x_1(t),\ldots,x_n(t)\big)+H\big(x_1(t+T_1^p),\ldots,x_n(t+T_1^p)\big).$$

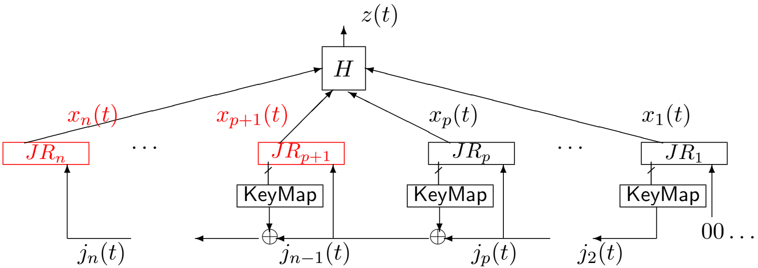
- Linear H:

$$z(t) + z(t + T_1^p) = \sum_{i=p+1}^{n} x_i(t) + x_i(t + T_1^p).$$

- Non-linear H: $H(t)$ and $H(t + T_1^p)$ have p inputs $x_1, \ldots, x_p$ in common, $0 \leq p \leq n$.

$$\Pr\left(z(t) + z(t + T_i^p) = 0\right) = \frac{1}{2}(1 - \delta),\ -1 \leq \delta \leq 1.$$

Linear Approximation of $JR_{p+1}, \ldots, JR_n$



For $JR_l,\ p+1 \le l \le n$, search for a set $\mathcal{A}$ of size w such that

$$\Pr\Big(\sum_{i\in\mathcal{A}} x_l(t+i) = 0\Big) = \frac{1}{2}(1-\varepsilon), \quad -1 \le \varepsilon \le 1,$$

Design Idea 1: Linear H, same registers

- Linear approximation of one register, (applies to all registers);
- Samples at time t and $t + T_1^p$.

$$\sum_{i \in \mathcal{A}} z(t+i) + z(t+i+T_1^p) =$$

$$= \sum_{j=p+1}^{n} \sum_{i \in \mathcal{A}} \left(x_j(t+i) + x_j(t+i+T_1^p)\right).$$

- Bias of $\sum_{i \in \mathcal{A}} x_i(t+i)$ is ε, we have $2(n-p)$ such relations

$$\varepsilon_{tot} = \varepsilon^{2(n-p)}$$

- $1/\varepsilon_{tot}^2$ samples needed to distinguish the cipher from a truly random source.

Design Idea 1: Linear H, same registers

Theorem

The computational complexity and the number N of keystream bits needed to reliably distinguish the Pomaranch family of stream ciphers using a linear filter function and n jump registers of the same type is upper bounded by

$$N \leq T_1^p + \frac{1}{\varepsilon^{4(n-p)}}, \quad p > 0,$$

where ε is the bias of the best linear approximation of the jump register.

- Pomaranch v1 (128 bit): Keystream and Complexity$=2^{71}$.
- Pomaranch v2 (80 bit): Keystream and Complexity$=2^{56}$.
  (128 bit): Keystream and Complexity$=2^{77}$.

Design Idea 2: Linear H, Different Registers

- Samples at time t and $t + T_1^p$.
- Linear approximation for all registers jointly.
- Bias for the approximation of register i is ε_i, total bias is given by

$$\varepsilon_{tot} = \prod_{i=p+1}^{n} \varepsilon_i^2.$$

Design Idea 2: Linear H, Different Registers

Theorem

Assuming there is a linear relation that is biased in all registers. The computational complexity and the number N of keystream bits needed to reliably distinguish the Pomaranch family of stream ciphers using a linear filter function and n jump registers of different types is upper bounded by

$$N \leq T_1^p + \frac{1}{\prod_{i=p+1}^{n} \varepsilon_i^4}, \quad p > 0,$$

where ε_i is the bias of jump register JR_i.

- Pomaranch v3 (128 bit): Keystream and Complexity$=2^{126}$. (Without frame length restriction)

Design Idea 3: Nonlinear H, Different Registers

- Consider the case, (can easily be extended), when the filter function H can be written on the form

$$H(x_1, \ldots, x_n) = G(x_1, \ldots, x_{n-1}) + x_n.$$

- $G(t)$ and $G(t + T_1^p)$ have p inputs $x_1, \ldots, x_p$ in common,

$$\Pr\left(z(t) + z(t + T_i^p) = 0\right) = \frac{1}{2}(1 - \delta),\ -1 \leq \delta \leq 1.$$

- Find linear approximation for JR_n.

Design Idea 3: Nonlinear H, Different Registers

$$H(x_1, \ldots, x_n) = G(x_1, \ldots, x_{n-1}) + x_n. \tag{1}$$

Theorem

The computational complexity and the number N of keystream bits needed to reliably distinguish the Pomaranch family of stream ciphers using a filter function of the form (1) is upper bounded by

$$N \leq T_1^p + \frac{1}{(\varepsilon^2 \delta^w)^2}.$$

where ε is the a biased approximation of weight w of register JR_n and δ is the bias of
$G\big(x_1(t), \ldots, x_{n-1}(t)\big) + G\big(x_1(t + T_1^p), \ldots, x_{n-1}(t + T_i^p)\big)$.

- Pomaranch v3 (80 bit): Keystream and Complexity$=2^{71}$.

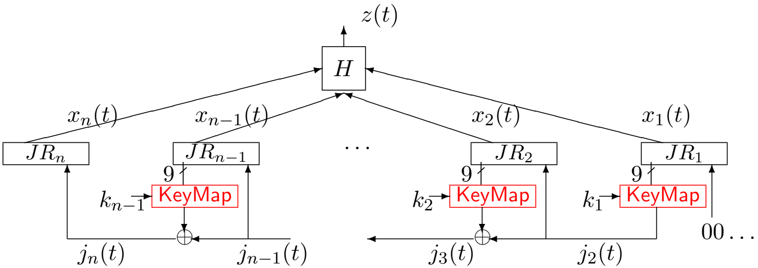
Attack II: Square Root Resynchronization Attack

- Divide the internal state into two parts,

$$State = (State_K,\ State_{K+IV}).$$

  $State_K$ only holds the key.
  $State_{K+IV}$ depends on both key and IV.
- If key size $|K| > |State_{K+IV}|/2$, attack succeeds with complexity below exhaustive key search.
- One attack scenario:
  - Fixed key.
  - One long keystream sequence from one IV.
  - Intercept ciphertexts from many IVs, knowing l plaintext bits of every ciphertext.
  - Goal is to recover more of the plaintext for one message.

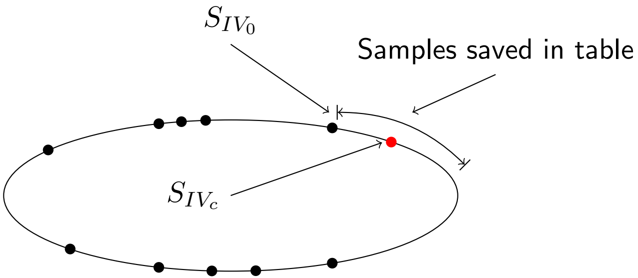
Square Root Resynchronization Attack on Pomaranch



- KeyMap, is key dependent but independent of IV.

Square Root Resynchronization Attack on Pomaranch

- Samples taken as, $S(t) = \big(z(t), z(t+1), \ldots, z(t+nL-1)\big)$.
- Fixed key defines state graph of size $(2^L - 1)^n \approx 2^{nL}$.
- Store large amount of samples from IV_0 in table.
- Find IV_c, such that $S_{IV_c}(t_c) = S_{IV_0}(t_0)$.



Attack Complexities

- Assume we have $2^{\beta nL}$ samples from IV_0.
- We need samples from $2^{(1-\beta)nL}$ different IVs to find collision.
- Time:
  - Sort table $\beta nL 2^{\beta nL}$.
  - Search in table $\beta nL 2^{(1-\beta)nL}$
- Memory: $nL2^{\beta nL}$

Results

		Attack I Keystream/Compl.	Attack II Memory/IVs/Compl.
Pomaranch v1	128 bit	2^{71} / 2^{71}	2^{67} / 2^{63} / 2^{63}
Pomaranch v2	80 bit	2^{56} / 2^{56}	2^{45} / 2^{42} / 2^{42}
	128 bit	2^{77} / 2^{77}	2^{67} / 2^{63} / 2^{63}
Pomaranch v3	80 bit	2^{71} / 2^{71}	2^{58} / 2^{54} / 2^{54}
	128 bit	2^{126} / $2^{126}*$	2^{71} / 2^{98} / 2^{104}

* Without frame length restriction.

Conclusions

- Presented the best distinguisher so far on all version and variants of Pomaranch in terms of computational complexity.
- Presented a general resynchronization attack that works for all ciphers where $|K| > |State_{K+IV}|/2$.
- First attack presented on Pomaranch Version 3.