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

1. Description of Pomaranch
2. Attack I: Linear Distinguisher
3. Attack II: Square Root IV Attack
4. Results and Conclusions
Outline

1. Description of Pomaranch
2. Attack I: Linear Distinguisher
3. Attack II: Square Root IV Attack
4. Results and Conclusions
Outline

1. Description of Pomaranch
2. Attack I: Linear Distinguisher
3. Attack II: Square Root IV Attack
4. Results and Conclusions
Outline

1. Description of Pomaranch
2. Attack I: Linear Distinguisher
3. Attack II: Square Root IV Attack
4. Results and Conclusions
Jump Registers

Jump Control = 0

H. Englund, M. Hell and T. Johansson

Two General Attacks on Pomaranch-like Keystream Generators
Jump Registers

Jump Control = 0

H. Englund, M. Hell and T. Johansson
Two General Attacks on Pomaranch-like Keystream Generators
Jump Registers

Jump Control = 0

H. Englund, M. Hell and T. Johansson
Two General Attacks on Pomaranch-like Keystream Generators
Jump Registers

H. Englund, M. Hell and T. Johansson
Two General Attacks on Pomaranch-like Keystream Generators
Jump Registers

Jump Control = 1
Jump Index = 12

H. Englund, M. Hell and T. Johansson
Two General Attacks on Pomaranch-like Keystream Generators
Description of Pomaranch

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

**H. Englund, M. Hell and T. Johansson**

Two General Attacks on Pomaranch-like Keystream Generators
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

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

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

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

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

- **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, $0 \leq p \leq n$.

  $$\Pr (z(t) + z(t + T_1^p) = 0) = \frac{1}{2} (1 - \delta), \quad -1 \leq \delta \leq 1.$$
Linear Approximation of $JR_{p+1}, \ldots, JR_n$

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

$$\Pr\left( \sum_{i \in \mathcal{A}} x_l(t + i) = 0 \right) = \frac{1}{2} (1 - \varepsilon), \quad -1 \leq \varepsilon \leq 1,$$
Design Idea 1: Linear $H$, same registers

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

\[
\sum_{i \in A} z(t + i) + z(t + i + T_1^p) = \\
= \sum_{j=p+1}^{n} \sum_{i \in A} (x_j(t + i) + x_j(t + i + T_1^p)).
\]

- Bias of $\sum_{i \in A} x_i(t + i)$ is $\varepsilon$, 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.
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 $\varepsilon$ 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 $\varepsilon_i$, 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

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

where $\varepsilon_i$ is the bias of jump register $JR_i$.

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

$$\Pr(z(t) + z(t + T_1^p) = 0) = \frac{1}{2}(1 - \delta), \quad -1 \leq \delta \leq 1.$$  

Find linear approximation for $JR_n$. 

H. Englund, M. Hell and T. Johansson  
Two General Attacks on Pomaranch-like Keystream Generators
Design Idea 3: Nonlinear $H$, Different Registers

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

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

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

H. Englund, M. Hell and T. Johansson

Two General Attacks on Pomaranch-like Keystream Generators
Attack II: Square Root Resynchronization Attack

- Divide the internal state into two parts,

\[ \text{State} = (\text{State}_K, \text{State}_{K+IV}). \]

- \( \text{State}_K \) only holds the key.
- \( \text{State}_{K+IV} \) depends on both key and IV.

- If key size \( |K| > |\text{State}_{K+IV}|/2 \), 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.
KeyMap, is key dependent but independent of IV.
Square Root Resynchronization Attack on Pomaranch

- Samples taken as, $S(t) = (z(t), z(t + 1), \ldots, z(t + nL - 1))$.
- 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 $S_{IV_c}(t_c) = S_{IV_0}(t_0)$.

H. Englund, M. Hell and T. Johansson

Two General Attacks on Pomaranch-like Keystream Generators
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 $\beta nL2^{\beta nL}$.
- Search in table $\beta nL2^{(1-\beta)nL}$

Memory: $nL2^{\beta nL}$
## Results

<table>
<thead>
<tr>
<th></th>
<th>Attack I Keystream/Compl.</th>
<th>Attack II Memory/IVs/Compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pomaranch v1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128 bit</td>
<td>$2^{71} / 2^{71}$</td>
<td>$2^{67} / 2^{63} / 2^{63}$</td>
</tr>
<tr>
<td><strong>Pomaranch v2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80 bit</td>
<td>$2^{56} / 2^{56}$</td>
<td>$2^{45} / 2^{42} / 2^{42}$</td>
</tr>
<tr>
<td>128 bit</td>
<td>$2^{77} / 2^{77}$</td>
<td>$2^{67} / 2^{63} / 2^{63}$</td>
</tr>
<tr>
<td><strong>Pomaranch v3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80 bit</td>
<td>$2^{71} / 2^{71}$</td>
<td>$2^{58} / 2^{54} / 2^{54}$</td>
</tr>
<tr>
<td>128 bit</td>
<td>$2^{126} / 2^{126*}$</td>
<td>$2^{71} / 2^{98} / 2^{104}$</td>
</tr>
</tbody>
</table>

* 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.