Vectorized linear approximations for attacks on SNOW 3G

Jing Yang[1] Thomas Johansson[1] Alexander Maximov[2]

[1]Dept. of Electrical and Information Technology, Lund University
[2]Ericsson Research, Lund, Sweden

Outline

LUND UNIVERSITY

Outline

Confidentiality and Integrity Protection in Cellular Networks

- Three standardized algorithms in LTE: SNOW 3G, AES, ZUC
  - 128-bit security level

Confidentiality and Integrity Protection in Cellular Networks

- Three standardized algorithms in LTE: SNOW 3G, AES, ZUC
  - 128-bit security level
- 5G: 256-bit security algorithms

Confidentiality and Integrity Protection in Cellular Networks

- Three standardized algorithms in LTE: SNOW 3G, AES, ZUC
  - 128-bit security level
- 5G: 256-bit security algorithms
- One possible solution: reuse existing algorithms
  - Security under the 256-bit key length should be investigated

LUND UNIVERSITY

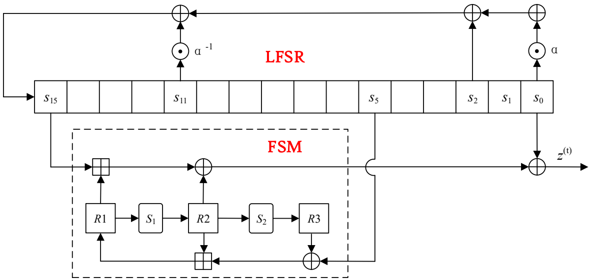
Confidentiality and Integrity Protection in Cellular Networks

- Three standardized algorithms in LTE: SNOW 3G, AES, ZUC
  - 128-bit security level
- 5G: 256-bit security algorithms
- One possible solution: reuse existing algorithms
  - Security under the 256-bit key length should be investigated
- **Contribution**: give linear cryptanalysis of SNOW 3G
  - Distinguishing attack 2^{172}
  - Correlation attack 2^{177}

Outline

LUND UNIVERSITY

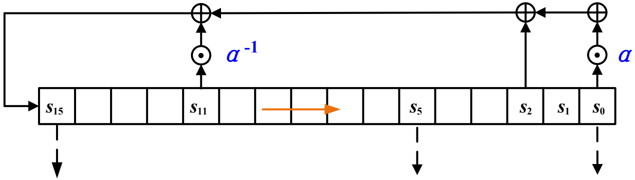
- A stream cipher with a linear part and a non-linear part



- **Linear part:** linear feedback shift register (LFSR)
- **Non-linear part**: finite state machine (FSM)

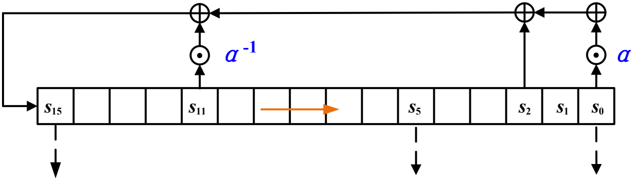
LUND UNIVERSITY

LFSR in SNOW 3G



- Defined over $GF(2^{32})$, 16 cells $\times$ 32 bits / cell $=$ 512 bits

LFSR in SNOW 3G

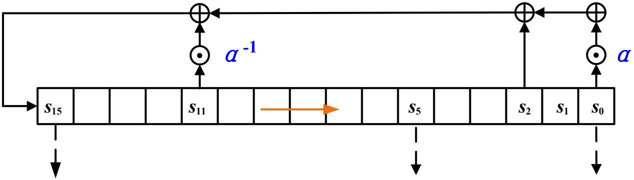


- Defined over $GF(2^{32})$, 16 cells $\times$ 32 bits $/$ cell $=$ 512 bits
- **Feedback polynomial**:

$$P(x) = \alpha x^{16} + x^{14} + \alpha^{-1} x^5 + 1 \in GF(2^{32})[x]$$

  - α is a root of a polynomial in $GF(2^8)[x]$

LFSR in SNOW 3G



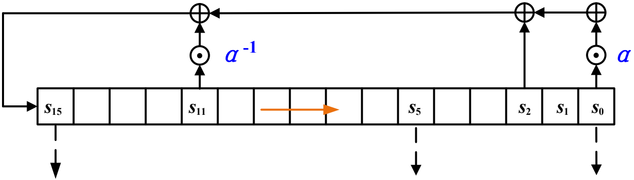
- Defined over $GF(2^{32})$, 16 cells $\times$ 32 bits / cell $=$ 512 bits
- **Feedback polynomial**:

$$P(x) = \alpha x^{16} + x^{14} + \alpha^{-1} x^5 + 1 \in GF(2^{32})[x]$$

  - α is a root of a polynomial in $GF(2^8)[x]$
- **LFSR update**:

$$s_i^{(t+1)} = s_{i+1}^{(t)} \quad (0 \le i \le 14),$$
$$s_{15}^{(t+1)} = \alpha^{-1} s_{11}^{(t)} + s_2^{(t)} + \alpha s_0^{(t)}.$$

LFSR in SNOW 3G



- Defined over $GF(2^{32})$, 16 cells $\times$ 32 bits / cell $= 512$ bits
- **Feedback polynomial**:

$$P(x) = \alpha x^{16} + x^{14} + \alpha^{-1} x^5 + 1 \in GF(2^{32})[x]$$

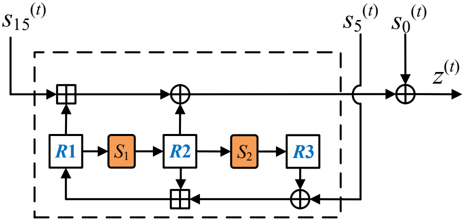
  - α is a root of a polynomial in $GF(2^8)[x]$
- **LFSR update**:

$$s_i^{(t+1)} = s_{i+1}^{(t)} \quad (0 \le i \le 14),$$
$$s_{15}^{(t+1)} = \alpha^{-1} s_{11}^{(t)} + s_2^{(t)} + \alpha s_0^{(t)}.$$

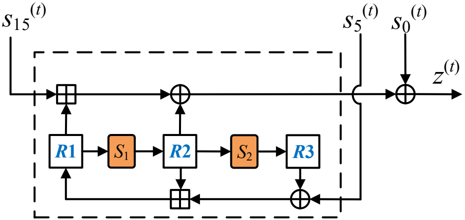
- $s_{15}^{(t)}, s_5^{(t)}, s_0^{(t)}$ used to update FSM and generate keystream

LUND UNIVERSITY

FSM in SNOW 3G

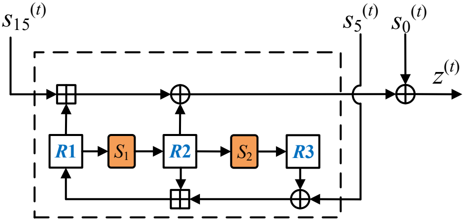


FSM in SNOW 3G



- **Keystream block**: $z^{(t)} = (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus R2^{(t)} \oplus s_0^{(t)}$

FSM in SNOW 3G

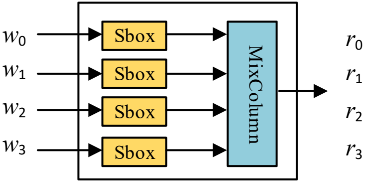


- **Keystream block**: $z^{(t)} = (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus R2^{(t)} \oplus s_0^{(t)}$
- **FSM update**:

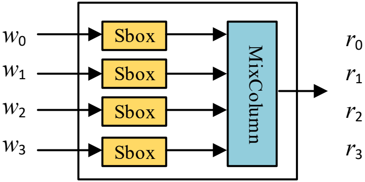
$$
\begin{aligned}
R1^{(t+1)} &= R2^{(t)} \boxplus_{32} (R3^{(t)} \oplus s_5^{(t)}) \\
R2^{(t+1)} &= S_1(R1^{(t)}) \\
R3^{(t+1)} &= S_2(R2^{(t)})
\end{aligned}
$$

- S_1, S_2 are 32-to-32 S-transforms

S-transforms in FSM



S-transforms in FSM

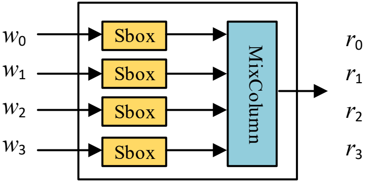


- $S_1 = L_1 \cdot S_R$, S_R is the AES S-box

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \cdot \begin{pmatrix} S_R(w_0) \\ S_R(w_1) \\ S_R(w_2) \\ S_R(w_3) \end{pmatrix}$$

LUND UNIVERSITY

S-transforms in FSM



- $S_1 = L_1 \cdot S_R$, S_R is the AES S-box

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \cdot \begin{pmatrix} S_R(w_0) \\ S_R(w_1) \\ S_R(w_2) \\ S_R(w_3) \end{pmatrix}$$

- $S_2 = L_2 \cdot S_Q$, S_Q is derived from the Dickson polynomials

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} y & y+1 & 1 & 1 \\ 1 & y & y+1 & 1 \\ 1 & 1 & y & y+1 \\ y+1 & 1 & 1 & y \end{pmatrix} \cdot \begin{pmatrix} S_Q(w_0) \\ S_Q(w_1) \\ S_Q(w_2) \\ S_Q(w_3) \end{pmatrix}$$

Outline

LUND UNIVERSITY

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:

LUND UNIVERSITY

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks

LUND UNIVERSITY

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]
  - Consider general vectorized linear approximation

LUND UNIVERSITY

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]
  - Consider general vectorized linear approximation
  - $\boldsymbol{e}$ has distribution D, the SEI (Squared Euclidean Imbalance):

$$\epsilon = |D| \cdot \sum_{e=0}^{|D|-1} \left(D(e) - \frac{1}{|D|} \right)^2$$

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]
  - Consider general vectorized linear approximation
  - $\boldsymbol{e}$ has distribution D, the SEI (Squared Euclidean Imbalance):

$$\epsilon = |D| \cdot \sum_{e=0}^{|D|-1} \left(D(e) - \frac{1}{|D|} \right)^2$$

  - Required Samples: $n = O(1/\epsilon)$ to distinguish e from random

LUND UNIVERSITY

Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]
  - Consider general vectorized linear approximation
  - $\boldsymbol{e}$ has distribution D, the SEI (Squared Euclidean Imbalance):

$$\epsilon = |D| \cdot \sum_{e=0}^{|D|-1} \left(D(e) - \frac{1}{|D|} \right)^2$$

  - Required Samples: $n = O(1/\epsilon)$ to distinguish e from random

LUND UNIVERSITY

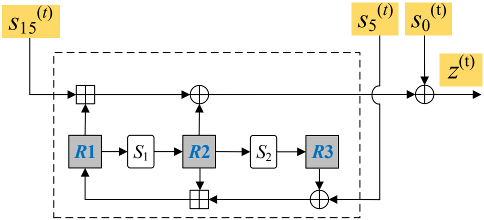
Basics for Linear Cryptanalysis of Stream Ciphers

- **Basic Idea**: approximate non-linear components as linear ones, further derive some linear relationships, involving:
  - LFSR states and keystream symbols$\Rightarrow$ Correlation attacks
  - Keystream symbols only $\Rightarrow$ Distinguishing attacks
- Linear approximation: $z = NF(s) = LF(s) + \boldsymbol{e}$ [biased noise]
  - Consider general vectorized linear approximation
  - $\boldsymbol{e}$ has distribution D, the SEI (Squared Euclidean Imbalance):

$$\epsilon = |D| \cdot \sum_{e=0}^{|D|-1} \left(D(e) - \frac{1}{|D|} \right)^2$$

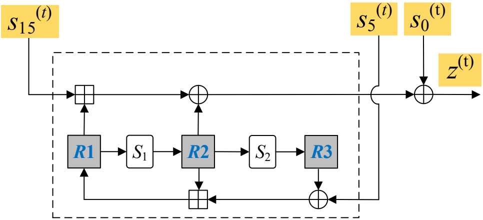
  - Required Samples: $n = O(1/\epsilon)$ to distinguish e from random
- **Key Point**: to find a good approximation with a large bias

Linear Approximation of FSM in SNOW 3G



LUND UNIVERSITY

Linear Approximation of FSM in SNOW 3G



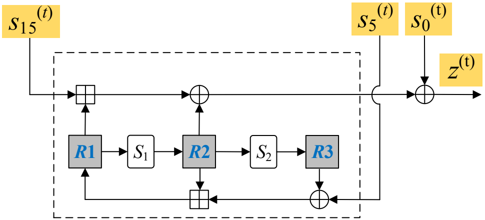
- Explore linear expression including only s_{15}, s_5, s_0, z

$$\bigoplus_{i \in I} \left(c_z^{(t+i)} z^{(t+i)} \oplus c_{15}^{(t+i)} s_{15}^{(t+i)} \oplus c_5^{(t+i)} s_5^{(t+i)} \oplus c_0^{(t+i)} s_0^{(t+i)} \right)$$

- $c_z^{(t+i)}, c_{15}^{(t+i)}, c_5^{(t+i)}, c_0^{(t+i)}$ are linear masking matrices

LUND UNIVERSITY

Linear Approximation of FSM in SNOW 3G



- Explore linear expression including only s_{15}, s_5, s_0, z

$$\bigoplus_{i \in I} \left(c_z^{(t+i)} z^{(t+i)} \oplus c_{15}^{(t+i)} s_{15}^{(t+i)} \oplus c_5^{(t+i)} s_5^{(t+i)} \oplus c_0^{(t+i)} s_0^{(t+i)} \right)$$

  - $c_z^{(t+i)}, c_{15}^{(t+i)}, c_5^{(t+i)}, c_0^{(t+i)}$ are linear masking matrices
- The SEI of it evaluates the quality of the approximation
  - Find good time set I and masking matrices

Linear Approximation of FSM

Consider 3 consecutive keystream blocks to cancel out R_1, R_2, R_3

Registers update and recursion at three time instances

$$R2^{(t+1)} = L_1 \cdot S_R(R1^{(t)}) \qquad R1^{(t-1)} = S_R^{-1} \cdot L_1^{-1}(R2^{(t)})$$
$$R3^{(t+1)} = L_2 \cdot S_Q(R2^{(t)}) \qquad R2^{(t-1)} = S_Q^{-1} \cdot L_2^{-1}(R3^{(t)})$$
$$R1^{(t+1)} = R2^{(t)} \boxplus_{32} (R3^{(t)} \oplus s_5^{(t)})$$

LUND UNIVERSITY

Linear Approximation of FSM

Consider 3 consecutive keystream blocks to cancel out R_1, R_2, R_3

Registers update and recursion at three time instances

$$R2^{(t+1)} = L_1 \cdot S_R(R1^{(t)})$$
$$R3^{(t+1)} = L_2 \cdot S_Q(R2^{(t)})$$
$$R1^{(t+1)} = R2^{(t)} \boxplus_{32} (R3^{(t)} \oplus s_5^{(t)})$$

$$R1^{(t-1)} = S_R^{-1} \cdot L_1^{-1}(R2^{(t)})$$
$$R2^{(t-1)} = S_Q^{-1} \cdot L_2^{-1}(R3^{(t)})$$

Keystream symbols at 3 consecutive time instances

$$z^{(t-1)} = (S_R^{-1} L_1^{-1}(R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus S_Q^{-1} L_2^{-1}(R3^{(t)}) \oplus s_0^{(t-1)}$$
$$z^{(t)} = (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus R2^{(t)} \oplus s_0^{(t)}$$
$$L_1^{-1} z^{(t+1)} = L_1^{-1}(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus S_R(R1^{(t)}) \oplus L_1^{-1} s_0^{(t+1)}$$

L_1^{-1} is the inverse of L_1, used as a linear masking matrix

24-bit Linear Approximation

Build 24-bit symbols: combining the first bytes

$$\underbrace{\begin{pmatrix} z^{(t-1)} \\ z^{(t)} \\ L_1^{-1} z^{(t+1)} \end{pmatrix}_{[0,0,0]}}_{\text{Sample at } (t):\ Z^{(t)}} = \underbrace{\begin{pmatrix} (S_R^{-1}(L_1^{-1} R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus s_{15}^{(t-1)} \oplus S_Q^{-1}(L_2^{-1} R3^{(t)}) \\ R2^{(t)} \\ L_1^{-1}[(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N2^{(t)}}$$

$$\oplus \underbrace{\begin{pmatrix} 0 \\ (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus s_{15}^{(t)} \\ S_R(R1^{(t)}) \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N1^{(t)}} \oplus \underbrace{\begin{pmatrix} s_0^{(t-1)} \oplus s_{15}^{(t-1)} \\ s_{15}^{(t)} \oplus s_0^{(t)} \\ L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{Contribution from LFSR } S^{(t)}}$$

LUND UNIVERSITY

24-bit Linear Approximation

Build 24-bit symbols: combining the first bytes

$$\underbrace{\begin{pmatrix} z^{(t-1)} \\ z^{(t)} \\ L_1^{-1} z^{(t+1)} \end{pmatrix}_{[0,0,0]}}_{\text{Sample at } (t):\ Z^{(t)}} = \underbrace{\begin{pmatrix} (S_R^{-1}(L_1^{-1} R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus s_{15}^{(t-1)} \oplus S_Q^{-1}(L_2^{-1} R3^{(t)}) \\ R2^{(t)} \\ L_1^{-1}[(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N2^{(t)}}$$

$$\oplus \underbrace{\begin{pmatrix} 0 \\ (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus s_{15}^{(t)} \\ S_R(R1^{(t)}) \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N1^{(t)}} \oplus \underbrace{\begin{pmatrix} s_0^{(t-1)} \oplus s_{15}^{(t-1)} \\ s_{15}^{(t)} \oplus s_0^{(t)} \\ L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{Contribution from LFSR } S^{(t)}}$$

- $Z^{(t)} = S^{(t)}$, $\quad N^{(t)} = N1^{(t)} \oplus N2^{(t)}$

LUND UNIVERSITY

24-bit Linear Approximation

Build 24-bit symbols: combining the first bytes

$$\underbrace{\begin{pmatrix} z^{(t-1)} \\ z^{(t)} \\ L_1^{-1} z^{(t+1)} \end{pmatrix}_{[0,0,0]}}_{\text{Sample at } (t)\text{: } Z^{(t)}} = \underbrace{\begin{pmatrix} (S_R^{-1}(L_1^{-1} R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus s_{15}^{(t-1)} \oplus S_Q^{-1}(L_2^{-1} R3^{(t)}) \\ R2^{(t)} \\ L_1^{-1}[(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N2^{(t)}}$$

$$\oplus \underbrace{\begin{pmatrix} 0 \\ (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus s_{15}^{(t)} \\ S_R(R1^{(t)}) \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N1^{(t)}} \oplus \underbrace{\begin{pmatrix} s_0^{(t-1)} \oplus s_{15}^{(t-1)} \\ s_{15}^{(t)} \oplus s_0^{(t)} \\ L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{Contribution from LFSR } S^{(t)}}$$

- $Z^{(t)} = S^{(t)}$, $\quad N^{(t)} = N1^{(t)} \oplus N2^{(t)}$
- $N1^{(t)}, N2^{(t)}$ independent

LUND UNIVERSITY

24-bit Linear Approximation

Build 24-bit symbols: combining the first bytes

$$\underbrace{\begin{pmatrix} z^{(t-1)} \\ z^{(t)} \\ L_1^{-1} z^{(t+1)} \end{pmatrix}_{[0,0,0]}}_{\text{Sample at } (t):\ Z^{(t)}} = \underbrace{\begin{pmatrix} (S_R^{-1}(L_1^{-1} R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus s_{15}^{(t-1)} \oplus S_Q^{-1}(L_2^{-1} R3^{(t)}) \\ R2^{(t)} \\ L_1^{-1}[(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N2^{(t)}}$$

$$\oplus \underbrace{\begin{pmatrix} 0 \\ (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus s_{15}^{(t)} \\ S_R(R1^{(t)}) \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N1^{(t)}} \oplus \underbrace{\begin{pmatrix} s_0^{(t-1)} \oplus s_{15}^{(t-1)} \\ s_{15}^{(t)} \oplus s_0^{(t)} \\ L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{Contribution from LFSR } S^{(t)}}$$

- $Z^{(t)} = S^{(t)}$, $\quad N^{(t)} = N1^{(t)} \oplus N2^{(t)}$
- $N1^{(t)}, N2^{(t)}$ independent
  - $\epsilon(N1^{(t)})$: loop over $R1^{(t)}[0], s_{15}^{(t)}[0]$ in $O(2^{16})$

24-bit Linear Approximation

Build 24-bit symbols: combining the first bytes

$$\underbrace{\begin{pmatrix} z^{(t-1)} \\ z^{(t)} \\ L_1^{-1} z^{(t+1)} \end{pmatrix}_{[0,0,0]}}_{\text{Sample at } (t)\text{: } Z^{(t)}} = \underbrace{\begin{pmatrix} (S_R^{-1}(L_1^{-1} R2^{(t)}) \boxplus s_{15}^{(t-1)}) \oplus s_{15}^{(t-1)} \oplus S_Q^{-1}(L_2^{-1} R3^{(t)}) \\ R2^{(t)} \\ L_1^{-1}[(R2^{(t)} \boxplus (R3^{(t)} \oplus s_5^{(t)}) \boxplus s_{15}^{(t+1)}) \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N2^{(t)}}$$

$$\oplus \underbrace{\begin{pmatrix} 0 \\ (R1^{(t)} \boxplus s_{15}^{(t)}) \oplus s_{15}^{(t)} \\ S_R(R1^{(t)}) \end{pmatrix}_{[0,0,0]}}_{\text{24-bit Noise } N1^{(t)}} \oplus \underbrace{\begin{pmatrix} s_0^{(t-1)} \oplus s_{15}^{(t-1)} \\ s_{15}^{(t)} \oplus s_0^{(t)} \\ L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}] \end{pmatrix}_{[0,0,0]}}_{\text{Contribution from LFSR } S^{(t)}}$$

- $Z^{(t)} = S^{(t)}$, $\quad N^{(t)} = N1^{(t)} \oplus N2^{(t)}$
- $N1^{(t)}, N2^{(t)}$ independent
  - $\epsilon(N1^{(t)})$: loop over $R1^{(t)}[0], s_{15}^{(t)}[0]$ in $O(2^{16})$
  - How about $\epsilon(N2^{(t)})$? (4 32-bit variables: $R2, R3, s_5, s_{15}$)

LUND UNIVERSITY

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them

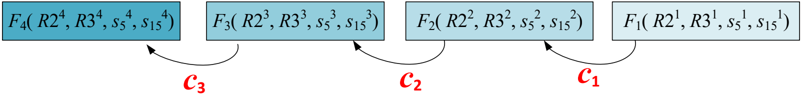
[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

LUND UNIVERSITY

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them



- Consider carries between different bytes

[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them

$$F_4(R2^4, R3^4, s_5^{\,4}, s_{15}^{\,4}) \xleftarrow{c_3} F_3(R2^3, R3^3, s_5^{\,3}, s_{15}^{\,3}) \xleftarrow{c_2} F_2(R2^2, R3^2, s_5^{\,2}, s_{15}^{\,2}) \xleftarrow{c_1} F_1(R2^1, R3^1, s_5^{\,1}, s_{15}^{\,1})$$

- Consider carries between different bytes
- FWHT can be used to speed up

[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them

$$F_4(R2^4, R3^4, s_5{}^4, s_{15}{}^4) \xleftarrow{c_3} F_3(R2^3, R3^3, s_5{}^3, s_{15}{}^3) \xleftarrow{c_2} F_2(R2^2, R3^2, s_5{}^2, s_{15}{}^2) \xleftarrow{c_1} F_1(R2^1, R3^1, s_5{}^1, s_{15}{}^1)$$

- Consider carries between different bytes
- FWHT can be used to speed up
- Complexity: $O(2^{40.53})$, bias: $\epsilon(N2) \approx 2^{-29.391880}$

[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them

$$F_4(R2^4, R3^4, s_5^4, s_{15}^4) \xleftarrow{c_3} F_3(R2^3, R3^3, s_5^3, s_{15}^3) \xleftarrow{c_2} F_2(R2^2, R3^2, s_5^2, s_{15}^2) \xleftarrow{c_1} F_1(R2^1, R3^1, s_5^1, s_{15}^1)$$

- Consider carries between different bytes
- FWHT can be used to speed up
- Complexity: $O(2^{40.53})$, bias: $\epsilon(N2) \approx 2^{-29.391880}$

- **The total bias**: $\epsilon(N) \approx 2^{-37.37}$, $\epsilon(4 \times N) \approx 2^{-162.76}$.

[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

Bias Computing of $\epsilon(N2^{(t)})$

Split variables / noise expression into smaller fields [ZXM15][1][MJ05] [2]

- Compute sub-distributions and combine them

$$F_4(R2^4, R3^4, s_5^4, s_{15}^4) \xleftarrow{c_3} F_3(R2^3, R3^3, s_5^3, s_{15}^3) \xleftarrow{c_2} F_2(R2^2, R3^2, s_5^2, s_{15}^2) \xleftarrow{c_1} F_1(R2^1, R3^1, s_5^1, s_{15}^1)$$

- Consider carries between different bytes
- FWHT can be used to speed up
- Complexity: $O(2^{40.53})$, bias: $\epsilon(N2) \approx 2^{-29.391880}$

- **The total bias**: $\epsilon(N) \approx 2^{-37.37}$, $\epsilon(4 \times N) \approx 2^{-162.76}$.

Q: Is the derived bias correct?

[1] Zhang B., et al. Fast correlation attacks over extension fields, large-unit...CRYPTO'2015.
[2] Maximov A, et al. Fast computation of large distributions and... ASIACRYPT 2005.

Experimental Verification of the Bias

- **Recall**: for a distribution P_X with bias ϵ, $O(1/\epsilon)$ samples are required to distinguish P_X from random

Experimental Verification of the Bias

- **Recall**: for a distribution P_X with bias ϵ, $O(1/\epsilon)$ samples are required to distinguish P_X from random
- **Idea**: if with $O(1/\epsilon)$ samples, we can distinguish P_X from random, the bias of P_X could not be much smaller than ϵ

Experimental Verification of the Bias

- **Recall**: for a distribution P_X with bias ϵ, $O(1/\epsilon)$ samples are required to distinguish P_X from random
- **Idea**: if with $O(1/\epsilon)$ samples, we can distinguish P_X from random, the bias of P_X could not be much smaller than ϵ
- **Tool**: hypothesis testing

$$\begin{cases} H_0 : P_X = P_N, & \text{the computed noise distribution,} \\ H_1 : P_X = P_U, & \text{the uniform distribution.} \end{cases}$$

LUND UNIVERSITY

Experimental Verification of the Bias

- **Recall**: for a distribution P_X with bias ϵ, $O(1/\epsilon)$ samples are required to distinguish P_X from random
- **Idea**: if with $O(1/\epsilon)$ samples, we can distinguish P_X from random, the bias of P_X could not be much smaller than ϵ
- **Tool**: hypothesis testing

$$\begin{cases} H_0 : P_X = P_N, & \text{the computed noise distribution,} \\ H_1 : P_X = P_U, & \text{the uniform distribution.} \end{cases}$$

- Decision rule:

$$P_X = \begin{cases} P_N, & \text{if } D(P_X||P_U) > D(P_X||P_N), \\ P_U, & \text{if } D(P_X||P_U) < D(P_X||P_N). \end{cases}$$

  - $D(x||y)$: KL divergence (or relative entropy) between x, y
  - The closer x, y is, the smaller $D(x||y)$ would be

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased

LUND UNIVERSITY

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased
- **Verify**: collect samples $Z^{(t)} \oplus S^{(t)}$, verify it follows P_N or P_U

LUND UNIVERSITY

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased
- **Verify**: collect samples $Z^{(t)} \oplus S^{(t)}$, verify it follows P_N or P_U
- run 64 SNOW 3G instances up to 2^{40} iterations, build samples

$$X^{(t)} = Z^{(t)} \oplus S^{(t)} = \begin{pmatrix} (z^{(t-1)} \oplus s_0^{(t-1)} \oplus s_{15}^{(t-1)})[0] \\ (z^{(t)} \oplus s_{15}^{(t)} \oplus s_0^{(t)})[0] \\ (L_1^{-1}[z^{(t+1)} \oplus s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}])[0] \end{pmatrix}$$

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased
- **Verify**: collect samples $Z^{(t)} \oplus S^{(t)}$, verify it follows P_N or P_U
- run 64 SNOW 3G instances up to 2^{40} iterations, build samples

$$X^{(t)} = Z^{(t)} \oplus S^{(t)} = \begin{pmatrix} (z^{(t-1)} \oplus s_0^{(t-1)} \oplus s_{15}^{(t-1)})[0] \\ (z^{(t)} \oplus s_{15}^{(t)} \oplus s_0^{(t)})[0] \\ (L_1^{-1}[z^{(t+1)} \oplus s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}])[0] \end{pmatrix}$$

- Build random sequences: lower 24 bits of keystream symbols

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased
- **Verify**: collect samples $Z^{(t)} \oplus S^{(t)}$, verify it follows P_N or P_U
- run 64 SNOW 3G instances up to 2^{40} iterations, build samples

$$X^{(t)} = Z^{(t)} \oplus S^{(t)} = \begin{pmatrix} (z^{(t-1)} \oplus s_0^{(t-1)} \oplus s_{15}^{(t-1)})[0] \\ (z^{(t)} \oplus s_{15}^{(t)} \oplus s_0^{(t)})[0] \\ (L_1^{-1}[z^{(t+1)} \oplus s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}])[0] \end{pmatrix}$$

- Build random sequences: lower 24 bits of keystream symbols
- For every sequence, check which distribution it follows

Experimental Verification

- **Recall**: $Z^{(t)} = S^{(t)} \oplus N^{(t)}$
  - $Z^{(t)} \oplus S^{(t)} = N^{(t)}$, biased
- **Verify**: collect samples $Z^{(t)} \oplus S^{(t)}$, verify it follows P_N or P_U
- run 64 SNOW 3G instances up to 2^{40} iterations, build samples

$$X^{(t)} = Z^{(t)} \oplus S^{(t)} = \begin{pmatrix} (z^{(t-1)} \oplus s_0^{(t-1)} \oplus s_{15}^{(t-1)})[0] \\ (z^{(t)} \oplus s_{15}^{(t)} \oplus s_0^{(t)})[0] \\ (L_1^{-1}[z^{(t+1)} \oplus s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}])[0] \end{pmatrix}$$

- Build random sequences: lower 24 bits of keystream symbols
- For every sequence, check which distribution it follows
- Errors:
  - TYPE I: a noise distribution is judged as random
  - TYPE II: a random distribution is judged as biased

Results of the Experimental Verification

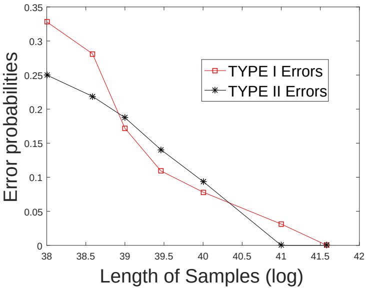


Figure. Error probabilities under different lengths of samples

LUND UNIVERSITY

Results of the Experimental Verification

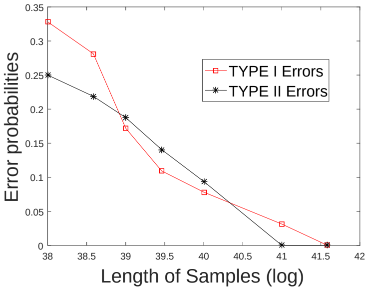


Figure. Error probabilities under different lengths of samples

- Error probabilities decrease with the increase of sample length

Results of the Experimental Verification

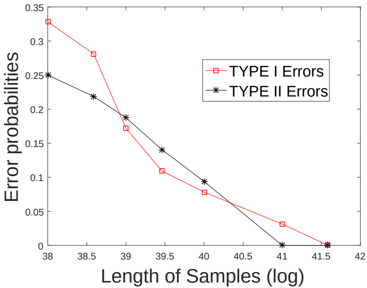


Figure. Error probabilities under different lengths of samples

- Error probabilities decrease with the increase of sample length
- Length 2^{40}: error probabilities < 0.1

  Length $2^{41.5}$: no errors (out of 21 sample sequences)

Results of the Experimental Verification

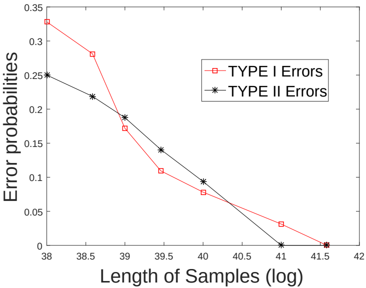


Figure. Error probabilities under different lengths of samples

- Error probabilities decrease with the increase of sample length
- Length 2^{40}: error probabilities < 0.1
  Length $2^{41.5}$: no errors (out of 21 sample sequences)
- With $(8 \sim 16) \cdot (1/\epsilon(N))$ ($\epsilon(N) \approx 2^{-37.37}$) samples, we could distinguish the sequences with large success probabilities

LUND UNIVERSITY

Results of the Experimental Verification

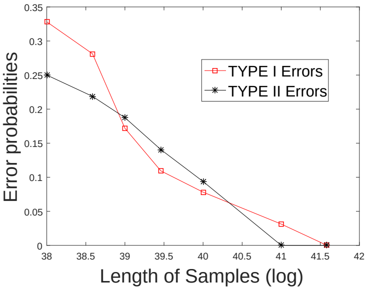


Figure. Error probabilities under different lengths of samples

- Error probabilities decrease with the increase of sample length
- Length 2^{40}: error probabilities < 0.1
  Length $2^{41.5}$: no errors (out of 21 sample sequences)
- With $(8 \sim 16) \cdot (1/\epsilon(N))$ ($\epsilon(N) \approx 2^{-37.37}$) samples, we could distinguish the sequences with large success probabilities
- The bias should be correct!

Distinguisher Constructions

- Distinguish the keystream sample sequence from random

Distinguisher Constructions

- Distinguish the keystream sample sequence from random
- Recall again: $Z^{(t)} \oplus S^{(t)} = N^{(t)}$

LUND UNIVERSITY

Distinguisher Constructions

- Distinguish the keystream sample sequence from random
- Recall again: $Z^{(t)} \oplus S^{(t)} = N^{(t)}$
  - If $S^{(t)}$ can be canceled, $Z^{(t)}$ would become biased
  - With enough samples, $Z^{(t)}$ can be distinguished from random

Distinguisher Constructions

- Distinguish the keystream sample sequence from random
- Recall again: $Z^{(t)} \oplus S^{(t)} = N^{(t)}$
  - If $S^{(t)}$ can be canceled, $Z^{(t)}$ would become biased
  - With enough samples, $Z^{(t)}$ can be distinguished from random

Q: How to cancel out $S^{(t)}$?

Distinguisher Constructions

- Distinguish the keystream sample sequence from random
- Recall again: $Z^{(t)} \oplus S^{(t)} = N^{(t)}$
  - If $S^{(t)}$ can be canceled, $Z^{(t)}$ would become biased
  - With enough samples, $Z^{(t)}$ can be distinguished from random

Q: How to cancel out $S^{(t)}$?

- **Idea**: find time set I with, usually 3,4 or 5, time instances

$$\bigoplus_{t\in I} S^{(t)} = 0, \qquad \bigoplus_{t\in I} Z^{(t)} = \bigoplus_{t\in I} N^{(t)}$$

LUND UNIVERSITY

Distinguisher Constructions

- Distinguish the keystream sample sequence from random
- Recall again: $Z^{(t)} \oplus S^{(t)} = N^{(t)}$
  - If $S^{(t)}$ can be canceled, $Z^{(t)}$ would become biased
  - With enough samples, $Z^{(t)}$ can be distinguished from random

Q: How to cancel out $S^{(t)}$?

- **Idea**: find time set I with, usually 3,4 or 5, time instances

$$\bigoplus_{t \in I} S^{(t)} = 0, \qquad \bigoplus_{t \in I} Z^{(t)} = \bigoplus_{t \in I} N^{(t)}$$

- Equivalent to finding a multiple of the generating polynomial $P(x)$ of weight 3, 4, or 5, with all coefficients being 1

Finalize the Distinguishing Attack

- Find a weight-4 multiple $K(x)$ using method from [LJ14] [3]
  - Time and storage complexities $O(2^{172})$

[3] Löndahl, C., & Johansson, T. Improved algorithms for finding low-weight polynomial multiples in $\mathbb{F}_2[x]$ and some cryptographic applications.DCC 2014.

Finalize the Distinguishing Attack

- Find a weight-4 multiple $K(x)$ using method from [LJ14] [3]
  - Time and storage complexities $O(2^{172})$
  - Suppose $K(x) = Q(x)P(x) = x^{t4} + x^{t3} + x^{t2} + x^{t1}$
  - $S^{(t1)} \oplus S^{(t2)} \oplus S^{(t3)} \oplus S^{(t4)} = 0$

[3] Löndahl, C., & Johansson, T. Improved algorithms for finding low-weight polynomial multiples in $\mathbb{F}_2[x]$ and some cryptographic applications.DCC 2014.

Finalize the Distinguishing Attack

- Find a weight-4 multiple $K(x)$ using method from [LJ14] [3]
  - Time and storage complexities $O(2^{172})$
  - Suppose $K(x) = Q(x)P(x) = x^{t4} + x^{t3} + x^{t2} + x^{t1}$
  - $S^{(t1)} \oplus S^{(t2)} \oplus S^{(t3)} \oplus S^{(t4)} = 0$
- Any time shifts t of $K(x)$, $x^t K(x)$, are still weight-4 multiples
  - $S^{(t+t1)} \oplus S^{(t+t2)} \oplus S^{(t+t3)} \oplus S^{(t+t4)} = 0$

[3] Löndahl, C., & Johansson, T. Improved algorithms for finding low-weight polynomial multiples in $\mathbb{F}_2[x]$ and some cryptographic applications.DCC 2014.

LUND UNIVERSITY

Finalize the Distinguishing Attack

- Find a weight-4 multiple $K(x)$ using method from [LJ14] [3]
  - Time and storage complexities $O(2^{172})$
  - Suppose $K(x) = Q(x)P(x) = x^{t4} + x^{t3} + x^{t2} + x^{t1}$
  - $S^{(t1)} \oplus S^{(t2)} \oplus S^{(t3)} \oplus S^{(t4)} = 0$
- Any time shifts t of $K(x)$, $x^t K(x)$, are still weight-4 multiples
  - $S^{(t+t1)} \oplus S^{(t+t2)} \oplus S^{(t+t3)} \oplus S^{(t+t4)} = 0$

New biased keystream samples, $t = 0, 1, 2...$

$$\begin{aligned} X^{(t)} &= Z^{(t+t1)} \oplus Z^{(t+t2)} \oplus Z^{(t+t3)} \oplus Z^{(t+t4)} \\ &= N^{(t+t1)} \oplus N^{(t+t2)} \oplus N^{(t+t3)} \oplus N^{(t+t4)} \end{aligned}$$

[3] Löndahl, C., & Johansson, T. Improved algorithms for finding low-weight polynomial multiples in $\mathbb{F}_2[x]$ and some cryptographic applications.DCC 2014.

Finalize the Distinguishing Attack

- Find a weight-4 multiple $K(x)$ using method from [LJ14] [3]
  - Time and storage complexities $O(2^{172})$
  - Suppose $K(x) = Q(x)P(x) = x^{t4} + x^{t3} + x^{t2} + x^{t1}$
  - $S^{(t1)} \oplus S^{(t2)} \oplus S^{(t3)} \oplus S^{(t4)} = 0$
- Any time shifts t of $K(x)$, $x^t K(x)$, are still weight-4 multiples
  - $S^{(t+t1)} \oplus S^{(t+t2)} \oplus S^{(t+t3)} \oplus S^{(t+t4)} = 0$

New biased keystream samples, $t = 0, 1, 2...$

$$\begin{aligned} X^{(t)} &= Z^{(t+t1)} \oplus Z^{(t+t2)} \oplus Z^{(t+t3)} \oplus Z^{(t+t4)} \\ &= N^{(t+t1)} \oplus N^{(t+t2)} \oplus N^{(t+t3)} \oplus N^{(t+t4)} \end{aligned}$$

- Bias: $\epsilon(X) = \epsilon(4 \times N) > 2^{-163}$ (regarded as independent)
  - Data complexity $O(2^{163})$

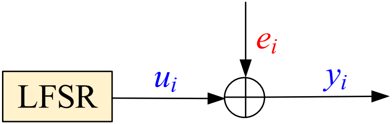
[3] Löndahl, C., & Johansson, T. Improved algorithms for finding low-weight polynomial multiples in $\mathbb{F}_2[x]$ and some cryptographic applications.DCC 2014.

Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$

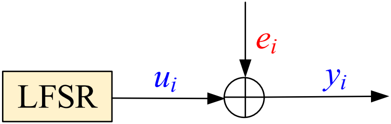
Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$



Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$

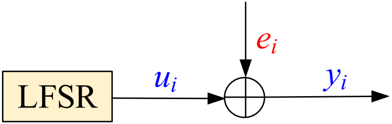


  - **Information symbol**: LFSR initial state

$$\mathbf{s} = (s_0, s_1, ..., s_{l-1})$$

Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$



  - **Information symbol**: LFSR initial state

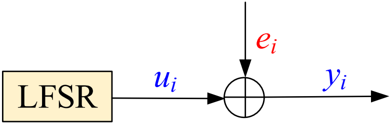
$$\mathbf{s} = (s_0, s_1, ..., s_{l-1})$$

  - **Codeword**: LFSR output

$$\mathbf{u} = (u_0, u_1, ..., u_{N-1}) = \mathbf{sG}, \mathbf{G} \in GF(2^n)^{l \times N}$$

Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$



  - **Information symbol**: LFSR initial state

$$\mathbf{s} = (s_0, s_1, ..., s_{l-1})$$

  - **Codeword**: LFSR output

$$\mathbf{u} = (u_0, u_1, ..., u_{N-1}) = \mathbf{sG}, \mathbf{G} \in GF(2^n)^{l \times N}$$

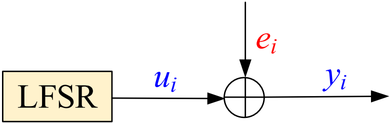
  - **Received codeword**: keystream samples

$$\mathbf{y} = (y_0, y_1, ..., y_{N-1}), \qquad y_i = u_i \oplus e_i$$

LUND UNIVERSITY

Basics of Correlation Attacks

- Modeled as decoding problems in $GF(2)$ or $GF(2^n)$



  - **Information symbol**: LFSR initial state

$$\mathbf{s} = (s_0, s_1, ..., s_{l-1})$$

  - **Codeword**: LFSR output

$$\mathbf{u} = (u_0, u_1, ..., u_{N-1}) = \mathbf{sG}, \mathbf{G} \in GF(2^n)^{l \times N}$$

  - **Received codeword**: keystream samples

$$\mathbf{y} = (y_0, y_1, ..., y_{N-1}), \qquad y_i = u_i \oplus e_i$$

- When $R = \log(2^n) \cdot l/N < C$: can be successfully decoded

Correlation Attacks on SNOW 3G

- Decoding problems are defined over $GF(2)$ or $GF(2^n)$
  - 24-bit approximation could not be directly used

LUND UNIVERSITY

Correlation Attacks on SNOW 3G

- Decoding problems are defined over $GF(2)$ or $GF(2^n)$
  - 24-bit approximation could not be directly used
- Instead, we build a new 8-bit approximation by

$$N' = \Lambda N[0] \oplus N[1] \oplus \Gamma N[2]$$

Correlation Attacks on SNOW 3G

- Decoding problems are defined over $GF(2)$ or $GF(2^n)$
  - 24-bit approximation could not be directly used
- Instead, we build a new 8-bit approximation by

$$N' = \Lambda N[0] \oplus N[1] \oplus \Gamma N[2]$$

  - Best $\Lambda = 0x18, \Gamma = 0x9c$: $\epsilon(N') = 2^{-40.97}$

Correlation Attacks on SNOW 3G

- Decoding problems are defined over $GF(2)$ or $GF(2^n)$
  - 24-bit approximation could not be directly used
- Instead, we build a new 8-bit approximation by

$$N' = \Lambda N[0] \oplus N[1] \oplus \Gamma N[2]$$

  - Best $\Lambda = 0x18, \Gamma = 0x9c$: $\epsilon(N') = 2^{-40.97}$
- The codeword and received codeword symbols:

$$u_t = (\Lambda(s_0^{(t-1)} \oplus s_{15}^{(t-1)}) \oplus s_0^{(t)} \oplus s_{15}^{(t)} \oplus \Gamma L_1^{-1}[s_0^{(t+1)} \oplus s_5^{(t)} \oplus s_{15}^{(t+1)}])[0]$$
$$y_t = \Lambda z^{(t-1)}[0] \oplus z^{(t)}[0] \oplus \Gamma(L_1^{-1} z^{(t+1)})[0]$$

- Recover $\mathbf{s}$ according to the y sequence
  - Preprocessing: generating parity checks
  - Processing: decoding

Finalize the Correlation Attack [ZXM15][4]

- Preprocessing: generating parity checks
  - Generating parity checks involving fewer LFSR states
  - Requires parity checks $O(2^{171.67})$
  - Time/space complexity $O(2^{176.56})$

[4] Zhang B., et al. Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. CRYPTO'2015.

Finalize the Correlation Attack [ZXM15][4]

- Preprocessing: generating parity checks
  - Generating parity checks involving fewer LFSR states
  - Requires parity checks $O(2^{171.67})$
  - Time/space complexity $O(2^{176.56})$
- Processing: decoding
  - Build a distinguisher: would be biased if a guess is correct
  - FWT can help to compute the bias
  - Decoding complexity $2^{174.75}$, 160 bits are recovered

[4] Zhang B., et al. Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. CRYPTO'2015.

Finalize the Correlation Attack [ZXM15][4]

- Preprocessing: generating parity checks
  - Generating parity checks involving fewer LFSR states
  - Requires parity checks $O(2^{171.67})$
  - Time/space complexity $O(2^{176.56})$
- Processing: decoding
  - Build a distinguisher: would be biased if a guess is correct
  - FWT can help to compute the bias
  - Decoding complexity $2^{174.75}$, 160 bits are recovered
- 16-bit correlation attack: same complexity, fewer parity checks

[4] Zhang B., et al. Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. CRYPTO'2015.

Outline

Conclusions

- We give linear cryptanalysis of SNOW 3G

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}
  - Verified the bias by collecting a large number of samples

LUND UNIVERSITY

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}
  - Verified the bias by collecting a large number of samples
  - Distinguishing attack with complexity 2^{172}

LUND UNIVERSITY

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}
  - Verified the bias by collecting a large number of samples
  - Distinguishing attack with complexity 2^{172}
  - Correlation attack with complexity 2^{177}

LUND UNIVERSITY

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}
  - Verified the bias by collecting a large number of samples
  - Distinguishing attack with complexity 2^{172}
  - Correlation attack with complexity 2^{177}
- If the key length in SNOW 3G would be increased to 256 bits, there are academic attacks on it

Conclusions

- We give linear cryptanalysis of SNOW 3G
  - 24-bit linear approximation of bias 2^{-37}
  - Verified the bias by collecting a large number of samples
  - Distinguishing attack with complexity 2^{172}
  - Correlation attack with complexity 2^{177}
- If the key length in SNOW 3G would be increased to 256 bits, there are academic attacks on it
- Not an immediate threat for 5G.

Thank you

Thank you for your attention!