Vulnerability of Certain Stream Ciphers Based on k-Normal Boolean Functions

Miodrag Mihaljevic
RCIS-AIST, Tokyo

A Seminar Lecture at CCRG

School of Physics and Mathematical Sciences

Nanyang Technological University (NTU)

Singapore, October 06, 2010 (15:30-16:30)

Roadmap

- Introduction and Motivation for the Work
- A Class of Stream Ciphers Based on the k-Normal Boolean functions
- LILI-128 Keystream Generator
- Underlying Ideas for a Novel **Cryptanalysis Employing a Weakness of k-Normal Boolean Functions**
- Pre-Processing
- Secret Key Recovery
- Performance and Comparison
- Concluding Remarks

I. Introduction

k-Normal Boolean Functions
and
motivation for the work

k-normal Boolean functions

Definition. Let $k \leq n$. A Boolean function f on $\mathcal{F}_2^n$ is called k-normal if there exists a k-dimensional flat on which f is constant.

A Toy Example.

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_4 \oplus x_2 x_5 \oplus x_3 x_6$$

$f(x_1 = 0, x_2 = 0, x_3 = 0, x_4, x_5, x_6) = 0$
independetly of x_4, x_5, x_6.

Illustrative References on k-Normal Boolean Functions

- C. Carlet, “The complexity of Boolean functions from cryptographic point of view”, in Complexity of Boolean Functions, *Dagestuhl Seminar Proceedings 06111*, 2006.
- C. Carlet, “On the degree, nonlinearity, algebraic thickness and nonnormality of Boolean functions, with developments on symmetric functions”, *IEEE Transactions on Information Theory*, vol. 50, pp. 2178-2185, 2004.
- C. Carlet, H. Dobbertin and G. Leander, “Normal Extensions of Bent Functions”, *IEEE Trans. on Information Theory*, vol. 50} no. 11, pp. 2880 – 2885, 2004.
- P. Charpin, “Normal Boolean functions”, *Journal of Complexity*, vol. 20, pp. 245 – 265, 2004.

Illustrations of Constructions which End-up with k-Normal Boolean Functions

Maiorana-McFarland Constructions. Choice of large r is necessary to increase the non-linearity and resiliency order of a Maiorana-McFarland type Boolean function. However this increases the normality order of the function.

Partial-Spreads Constructions. In order to construct functions with high order resiliency we are required to find ϕ such that for all z in F_{2^r}, $\phi^*(z) \oplus v$ has weight greater than m, the order of resiliency, which in turns mean that we must choose high values of s since $s > m$. Therefore the resulting function becomes $(s-1)$-normal.

Statements of Claude Carlet regarding k-normal Boolean Functions

- “The complexity criterion we are interested in is non-k-normality with small k (smaller is k, harder is the criterion).”
- “**This complexity criterion is not yet related to explicit attacks on ciphers**.”
- “The situation of the degree and of the nonlinearity, when they were first considered, was similar.”
- “For instance, the linear attack has been discovered by Matsui sixteen years after Rothaus introduced the idea.”

Motivation and Goals

- **Consideration of vulnerabilities** of cryptographic primitives which employ k-normal Boolean Functions.

- **Cryptanalysis of particular stream ciphers** which employ k-normal Boolean Functions**.**
- Developing of **dedicated algebraic** which employ a weakness of k-normal Boolean Functions**.**

II. Certain Keystream Generators and k-Normal Boolean Functions

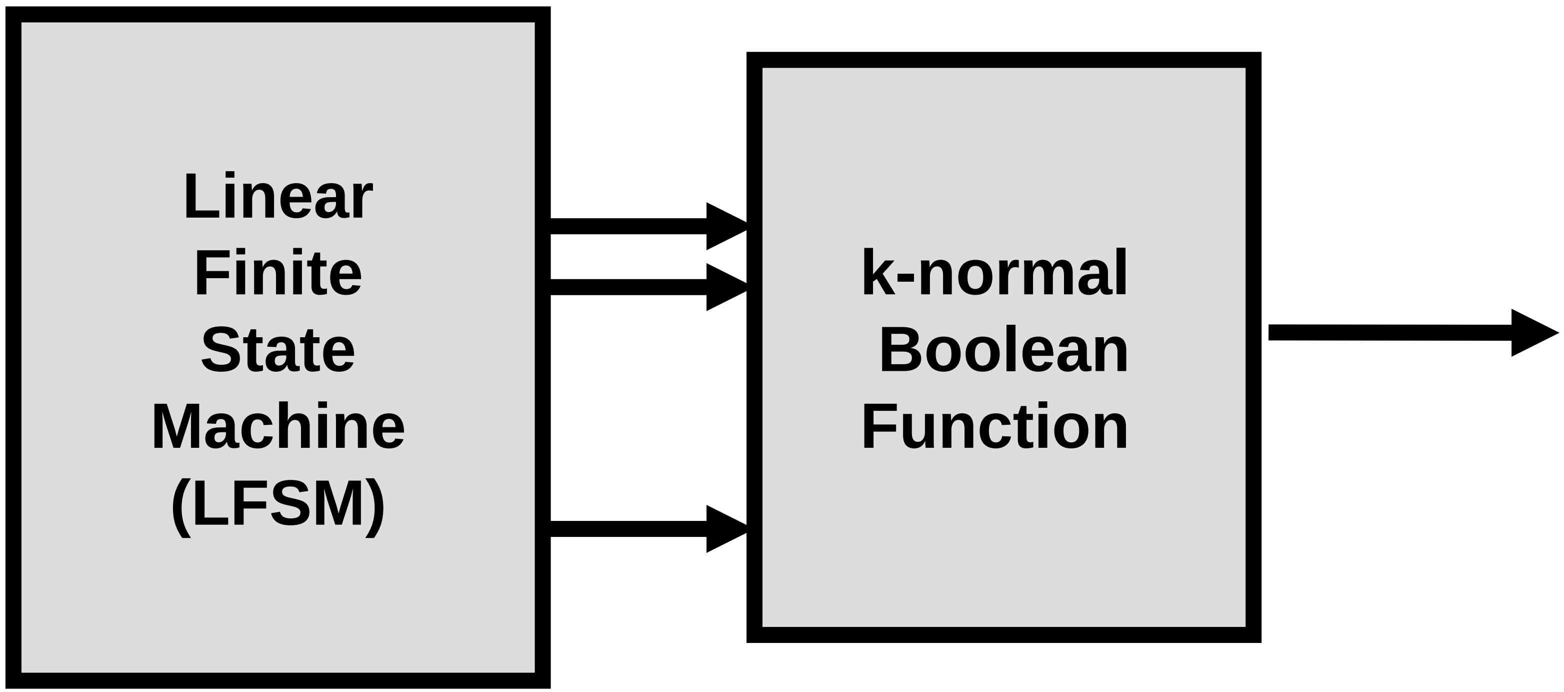
Nonlinear Filter Generator and Combination Generator with

k-Normal Boolean Functions

Boolean Functions and NF

- **Nonlinear Filter (NF)** is a textbook keystream generator but also can be considered as approximations of certain more complex generators.
- Design criteria and cryptographic complexity consideration of Boolean functions is usually related to their employment in NF.

Nonlinear Filter (NF)



Illustrative References

- M. Fossorier, M.J. Mihaljevic and H. Imai, “Modeling Block Encoding Approaches for Fast Correlation Attack”, IEEE Transactions on Information Theory, vol. 53, no. 12, pp. 4728-4737, Dec. 2007.

- E. Pasalic, "On Guess and Determine Cryptanalysis of LFSR-Based Stream Ciphers", IEEE Trans. Inform. Theory, vol. 55, pp. 3398-3406, July 2009.

A Generic Framework for Cryptanalysis

mounting an attack for internal state
or secret key recovery

Two Phases Framework for Cryptanalysis

Phase I:

- Pre-Processing: **Independent of any Secret Key or Sample**
- **Should be done only once.**
- A Preparation for the secret key recovery

Phase II:

- **Generator Internal state and Secret Key Recovery** for a given sample.

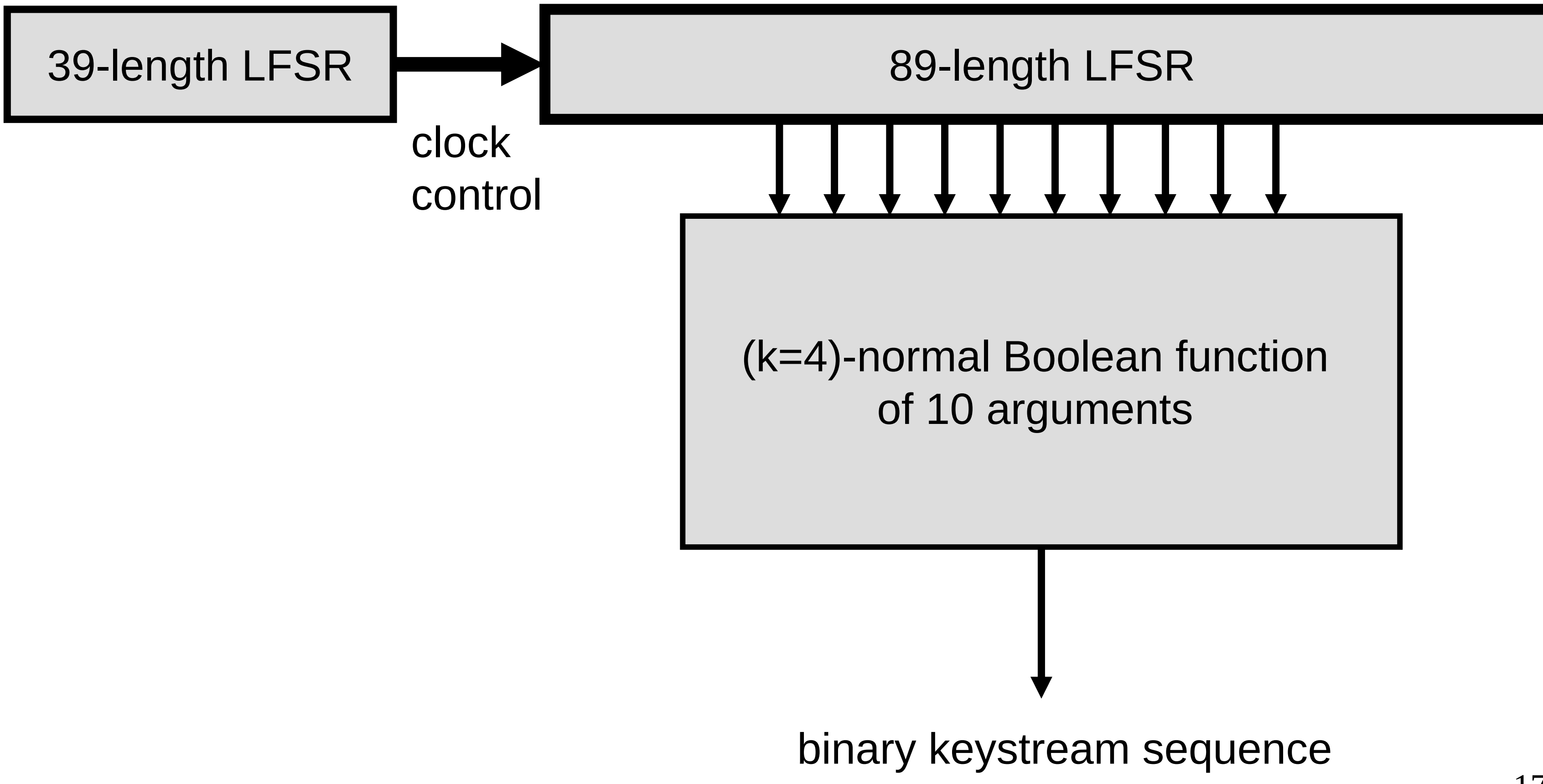
III. LILI-128 Keystream Generator

An Illustration of Stream Cipher
Vulnerable Employing a Weakness of
k-Normal Boolean Functions

A Note on LILI-128

- LILI-128 was submitted to NESSIE crypto-project and reported in SAC 2000 Proceedings (LNCS)
- Although broken via a number of attacks it still **serves as test-bad for illustration of power of novel techniques for cryptanalysis** and their comparison with the previously reported ones.

A Simplified Scheme of LILI-128 Keystream Generator



Algebraic Normal Form (ANF) of Boolean Function Employed in LILI-128

$$
\begin{aligned}
& f_d(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = \\
& x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8 \oplus x_1x_9 \oplus \\
& x_3x_9 \oplus x_4x_{10} \oplus x_6x_{10} \oplus x_3x_7x_9 \oplus x_4x_7x_9 \oplus x_6x_7x_9 \oplus \\
& x_3x_8x_9 \oplus x_6x_8x_9 \oplus x_4x_7x_{10} \oplus x_5x_7x_{10} \oplus x_6x_7x_{10} \oplus \\
& x_3x_8x_{10} \oplus x_4x_8x_{10} \oplus x_2x_9x_{10} \oplus x_3x_9x_{10} \oplus x_4x_9x_{10} \oplus \\
& x_5x_9x_{10} \oplus x_3x_7x_8x_{10} \oplus x_5x_7x_8x_{10} \oplus x_2x_7x_9x_{10} \oplus \\
& x_4x_7x_9x_{10} \oplus x_6x_7x_9x_{10} \oplus x_1x_8x_9x_{10} \oplus x_3x_8x_9x_{10} \oplus \\
& x_4x_8x_9x_{10} \oplus x_6x_8x_9x_{10} \oplus x_4x_6x_7x_9 \oplus x_5x_6x_7x_9 \oplus \\
& x_2x_7x_8x_9 \oplus x_4x_7x_8x_9 \oplus x_4x_6x_7x_9x_{10} \oplus x_5x_6x_7x_9x_{10} \oplus \\
& x_3x_7x_8x_9x_{10} \oplus x_4x_7x_8x_9x_{10} \oplus x_4x_6x_7x_8x_9 \oplus x_5x_6x_7x_8x_9 \oplus \\
& x_4x_6x_7x_8x_9x_{10} \oplus x_5x_6x_7x_8x_9x_{10}
\end{aligned}
$$

IV. Underlying Ideas and Theoretical Framework for the Cryptanalysis

for mounting an attack for internal state recovery

Main Underlying Idea

Our attack on LILI-128 is based on the observation that the function

f_d is zero if $x_1 = x_2 = x_3 = x_4 = x_5 = x_6 = 0$,

that is,

$f_d(0, 0, 0, 0, 0, 0, x_7, x_8, x_9, x_{10}) = 0$,

for all $x_7, x_8, x_9, x_{10} \in \mathcal{F}_2$

(also implying that f_d is a $k = 4$-normal Boolean function).

Notes

Note that the above is a particular example of the possibility that a **Boolean function can be substantially modified (degraded) when a subset of its arguments take certain values**.

In the considered case, *when certain variables are set to zero, the function is stuck to zero independently of all other variables*.

Theoretical Framework (1)

Let S be the transition matrix of $LFSR_d$. A sequence $\{c(t)\}_{t=0}^{m-1}$ of outputs of $LFSR_c$ is referred to as a *clocking sequence* of length m. Suppose that $\mathbf{X}_t = (X_0(t), \ldots, X_{88}(t))$ is the state of $LFSR_d$ at time t. Suppose $\mathbf{X}_0$ is the state of $LFSR_d$ after it is clocked according to the output $c(0)$. The subsequent states of $LFSR_d$ and the clocking sequence satisfy the following equations

$$\mathbf{X}_t = \mathbf{X}_{t-1} S^{c(t)}, \text{ for } t = 1, \ldots, m-1.$$

Theoretical Framework (2)

Let $S_j^{(\tau)}$ be the j-th column of the matrix S^τ, where τ is any integer. Accordingly,
$\mathbf{X}_t = \mathbf{X}_0 S^{\beta_t} = \mathbf{X}_0(S_0^{(\beta_t)}, \ldots, S_j^{(\beta_t)}, \ldots, S_{88}^{(\beta_t)})$
$= (\mathbf{X}_0 S_0^{(\beta_t)}, \ldots, \mathbf{X}_0 S_j^{(\beta_t)}, \ldots, \mathbf{X}_0 S_{88}^{(\beta_t)})$,
where $\beta_t = \sum_{i=1}^{t} c(i)$.

At any time t, the inputs $(x_1, \ldots, x_{10})$ to the filter function f_d are as follows:
$x_1 = X_0(t)$, $x_2 = X_1(t)$, $x_3 = X_3(t)$, $x_4 = X_7(t)$, $x_5 = X_{12}(t)$, $x_6 = X_{20}(t)$, $x_7 = X_{30}(t)$, $x_8 = X_{44}(t)$, $x_9 = X_{65}(t)$, $x_{10} = X_{80}(t)$.

If $X_0(t) = X_1(t) = X_3(t) = X_7(t) = X_{12}(t) = X_{20}(t) = 0$ then the output of the function f_d is 0 irrespective of the values of $X_{30}(t)$, $X_{44}(t)$, $X_{65}(t)$ and $X_{80}(t)$.

Theoretical Framework (3)

Let $\mathcal{I}_0$ be the set of all states of $LFSR_d$ at certain time instance such that:

$$X_0(t) = X_1(t) = X_3(t) = X_7(t) = X_{12}(t) = X_{20}(t) = 0 \ ,$$

$$t = i(2^{39} - 1) \ , \ \ i = 0, \ldots, m - 1 \ ,$$

and let a state belonging to $\mathcal{I}_0$ be considered as a realization of a vector random variable $\mathbf{x}$.

Theoretical Framework (4)

The importance of the set $\mathcal{I}_0$ lies in the fact that if $\mathbf{x} \in \mathcal{I}_0$ is a state of $LFSR_d$ then the inputs $x_1, \ldots, x_6$ of the function f_d are 0 at times $t = i(2^{39} - 1)$, $i = 0, \cdots, m-1$, and they specify a system of $6m$ linear equations where unknowns are bits of the considered state of $LFSR_d$. Let the rank of this system of equations is equal to $89 - \ell$, $\ell = \ell(m)$, and $\ell < 89$.

Theoretical Framework (5)

Let $\mathbf{y}$ be a random variable taking values from the set $\{0,1\}^m$ and let the keystream bits at the time instances $t = \Delta + i(2^{39} - 1)$, $i = 0, 1, ..., m-1$, for some $\Delta \in \{0, 1, ...\}$, are considered as the realizations of $\mathbf{y}$. Suppose we observe m zeros in the keystream at the positions $t = \Delta + i(2^{39} - 1)$, $i = 0, 1, ..., m - 1$: We denote this event by $\mathbf{y} = 0$. Since the keystream is pseudorandom, $Pr(\mathbf{y} = 0) = 2^{-m}$.

Theoretical Framework (6)

Theorem 1. Assuming the above notation, we have the following: $Pr(\mathbf{x} \in \mathcal{I}_0) = 2^{-(89-\ell)}$ and $Pr(\mathbf{x} \in \mathcal{I}_0 | \mathbf{y} = 0) = 2^{-(89-\ell-m)}$.

Sketch of the Proof. The underlying assumptions directly imply the following: $Pr(\mathbf{x} \in \mathcal{I}_0) = \frac{2^\ell}{2^{89}}$, $Pr(\mathbf{y} = 0) = 2^{-m}$ and $Pr(\mathbf{y} = 0 | \mathbf{x} \in \mathcal{I}_0) = 1$. On the other hand $Pr(\mathbf{x} \in \mathcal{I}_0 | \mathbf{y} = 0) = \frac{Pr(\mathbf{y}=0|\mathbf{x}\in\mathcal{I}_0)Pr(\mathbf{x}\in\mathcal{I}_0)}{Pr(\mathbf{y}=0)} = 2^{-(89-\ell-m)}$.

Theoretical Framework (7)

Finally note the following: we consider a system of $m(n-k)$ equations where n and k correspond to a k-normal Boolean function of n variables where $k = 4$ and $n = 10$. Accordingly, Theorem 1 directly implies the following corollary.

Corollary 1. When $m \in \{1, 2, ..., 14\}$ and accordingly all the equations are independent, the probability $Pr(\mathbf{x} \in \mathcal{I}_0 | \mathbf{y} = 0) = 2^{-m(n-k-1)}$ implying that this probability is an increasing function of the parameter k.

Origin for Cryptanalysis

After observing $\mathbf{y} = 0$ (i.e. a block of m-zeros in the keystream sequence decimated with the period $2^{39} - 1$), we can assume that it has been generated by the state $\mathbf{x} \in \mathcal{I}_0$, and the probability that this assumption is correct is given by the above Theorem 1. Under the considered assumption the related system of equations specifies $89 - \ell$ bits of the corresponding state of $LFSR_d$ where $89 - \ell$ is the rank of the system of $6m$ equations.

Two Phases Framework for Cryptanalysis

Phase I:

- Pre-Processing: **Independent of any Secret Key or Sample**
- **Should be done only once.**
- A Preparation for the internal state recovery.

Phase II:

- **Internal State Recovery** for a given sample.

IV. Pre-Processing

Preparation Phase:
Should be Performed Only Once

Pre-Processing Step I

- System of Equations

  1. For given $m < 15$, establish the following system of $m(n-k) = 6m$ independent equations :

  $$X_j(0) = 0 \ , \ \ \mathbf{X}_0 S_j^{(t)} = 0 \ ,$$

  $$j = 0, 1, 3, 7, 12, 20, \ \ t = i{\cdot}5(2^{38}{-}1) \ , \ \ i = 1, \ldots, m{-}1 \ .$$

  2. Specify the solutions of the system where there are $\ell = 89 - m(n-k) = 89 - 6m$ free variables (recall that the state has 89 bits and that the available system of equations has $6m$ independent equations).

Pre-Processing Step II

- Table

  For each of 2^ℓ possible patterns of $\ell = 89 - m(n-k) = 89 - 6m$ free variables, do the following:

  1. Determine a candidate $LFSR_d$ state $\hat{\mathbf{X}}_0$ as the particular solution under assumed ℓ-bits;

  2. Generate the subsequence $\{\hat{y}_{(m+i)\cdot(2^{39}-1)}\}_{i=1}^{89}$) employing $\hat{\mathbf{X}}_0\mathbf{S}^{(m+i)\cdot 5(2^{38}-1)}$, $i = 1, 2, ..., 89$, and the function $f_d(\cdot)$;

  3. Memorize in the table the pair $(\hat{\mathbf{X}}_0, \{\hat{y}_{(m+i)\cdot(2^{39}-1)}\}_{i=1}^{89})$.

Algorithm of Pre-Processing: Output

- The output of pre-computation is a table with 2^{ℓ} rows and 2 columns.

- Each row contains a pair: (Candidate $LFSR_d$ State, Corresponding 89-bit Decimated Keystream).

V. Algorithm for Internal State Recovery

for a Given Sample Recovers the Internal State

Structure of the
Algorithm for the Internal State Recovery

- *Inputs*: The sample, keystream sequence $\{y_t\}_{t=1}^{N}$, and the table constructed in the pre-processing step for given parameter m

- *Processing Steps*: Autonomous recovering of $LFSR_d$ (Phaee I) and $LFSR_c$ (Phase II) internal states.

- *Output*: The recovered internal state or the flag that the algorithm has failed.

Processing Steps (1)

For $\Delta = 0, 1, ..., \Delta_{max} = N-(m+89)\cdot(2^{39}-1)$, do the following:

- Inspect the given sample at the decimated positions $y_{\Delta+i(2^{39}-1)}$, $i = 0, 1, ..., m-1$:
  - If all the inspected positions are equal to zero (a block of m zeros is detected), select the following subsequence: $y_{\Delta+(m+i)(2^{39}-1)}$, $i = 1, 2, ..., 89$, and go to the step 1 (b);
  - otherwise increase $\Delta \rightarrow \Delta + 1 \leq \Delta_{max}$ and perform new inspection.

Processing Steps (2)

- Search the second column of the table for a possible match of the string in any of the rows and the selected subsequence $\{y_{\Delta+(m+i)(2^{39}-1)}\}_{i=1}^{89}$:
  - If the match is detected read $\hat{\mathbf{X}}_0$ from the same row and accept it as the state of $LFSR_d$;
  - If the match is not found in the table, continue the search with $\Delta \rightarrow \Delta + 1 \leq \Delta_{max}$.

- Based on the recovered $LFSR_d$ state and the sequence it generates and the given keystream sample, recover the state of $LFSR_c$ employing a suitable procedure which minimizes the overall complexity.

VI. Complexities of the Attack and Numerical Illustrations

Complexity of Pre-Processing

Required Sample

Complexity of Processing

Complexity of Pre-Processing

Theorem 2. The time complexity of pre-processing is $O(2^{89-m(n-k)})$ and the pre-processing output requires a memory of $2^{89-m(n-k)}$ 89-bit words, assuming $m < 15$.

Sketch of the Proof. The time complexity of the step I is determined by complexity of the Gaussian elimination, i.e. it is approximately $89^3 = 2^{3\log_2 89}$, The complexity of the step II is $O(2^\ell)$ and accordingly it is the dominated one. Dimension of the required memory is a direct implication of the output requiremens. Finally we take into account that $\ell = 89 - m(n-k)$.

Required Sample

Theorem 3. The data complexity of the attack is $\sim 2^{\max\{46, m(n-k)\}}$, assuming $m < 15$.

Sketch of the Proof. The probability that $\mathbf{x} \in \mathcal{I}_o$ when a block of m 0's is observed is given by and accordingly we need to check $2^{(89-\ell-m)}$ blocks of zeros of length m to get on an average one case such that $x \in \mathcal{I}_0$. The probability that a block of m zeros have appeared is equal to 2^{-m}. So in order to obtain on an average $2^{(89-l-m)}$ blocks of zeros of length m we need to inspect $2^m 2^{(89-\ell-m)} = 2^{(89-\ell)}$ candidates, $\ell = 89 - m(n-k)$ and $m < 15$. Each candidate should be checked via consideration of additional (next) 89 bits of the decimated sequence. Therefore the required keystream sample length is $\approx (89+m)2^{39} + 2^{(89-\ell)}$. If $m < 38$ then $(89+m)2^{39} \approx (2^7)(2^{39}) = 2^{46}$ and so, the data complexity can be estimated as $\approx 2^{\max\{46, 89-\ell\}}$.

Complexity of Processing

Theorem 4. The computational complexity of the online keystream processing phase of the attack is $\sim 2^{m(n-k-1)}$, assuming $m < 15$.

Theorem 5. The space complexity of the online keystream processing phase of the attack is $\sim (2^{89-m(n-k)} + 2^{max\{46,m(n-k)\}})$, assuming $m < 15$.

m	pre-processing time complexity	pre-processing space complexity	required sample for processing	processing time complexity	processing space complexity
5	2^{59}	2^{59}	2^{46}	2^{25}	2^{59}
6	2^{53}	2^{53}	2^{46}	2^{30}	2^{53}
7	2^{47}	2^{47}	2^{46}	2^{35}	2^{47}
8	2^{41}	2^{41}	2^{48}	2^{40}	2^{48}
9	2^{35}	2^{35}	2^{54}	2^{45}	2^{54}
10	2^{29}	2^{29}	2^{60}	2^{50}	2^{60}

VII. Comparison with Previously Reported Attacks

attack	pre-processing time complexity	required sample	processing time complexity	processing space complexity
correlation CRYPTO 2004	$\sim 2^{62}$ (table lookups)	$\sim 2^{29}$	$\sim 2^{62}$ (vector substitut. and mod 2 add.)	$\sim 2^{30}$
time-memory trade-off, SAC2001	$\sim 2^{48}$ (DES operations)	$\sim 2^{46}$	$\sim 2^{48}$ (DES operations)	$\sim 2^{45}$ 89-bit words
algebraic CRYPTO2004, ACISP2007	$\sim 2^{35}$ (symbolic lin. combining)	$\sim 2^{60}$	$\sim 2^{40}$ (bits substitut. and mod 2 add.)	$\sim 2^{44}$
novel $m = 7$	$\sim 2^{47}$ (vector substitut. and mod 2 add.)	$\sim 2^{46}$	$\sim 2^{35}$ (table lookups)	$\sim 2^{47}$

VIII. Concluding Notes

Summary of the Talk
and Some Open Problems

Main Messages of This Talk

- This talk points out some **possible vulnerabilities of cryptographic primitives which employ k-normal Boolean functions**.
- Particularly, this talk confirms that the **Non-Normality is an important design criteria for Boolean functions**
- A novel algorithm for **cryptanalysis** of stream cipher LILI-128 **more powerful than previously reported ones** has been proposed and discussed.
- The results on cryptanalysis of LILI-128 are a background towards **future activities** on a framework for using weaknesses of k-normal Boolean functions based on **dedicated algebraic and correlation attacking approaches**.

Some **Open Problems**

CRYPTANALYSIS

- General issues of vulnerability of nonlinear filters based on k-normal Boolean functions
- Dedicated cryptanalysis of stream ciphers which employ k-normal Boolean functions: Grain (for example)

DESIGN

- Techniques for design of Boolean functions which minimizes k-normality

Thank You Very Much for the Attention,

and

QUESTIONS Please!