

Extended BDD-based Cryptanalysis of Keystream Generators

Dirk Stegemann

Theoretical Computer Science
University of Mannheim, Germany

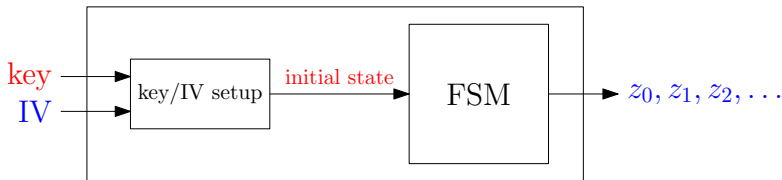
14th Workshop on Selected Areas in Cryptography
August 16-17, 2007
Ottawa, Canada

Overview

- 1 Introduction
- 2 Binary Decision Diagrams
- 3 BDD-based Attacks and Extensions

Stream Ciphers — Basic Structure

Many stream ciphers use a keystream generator



to encrypt the plaintext bits p_i via $p_i \oplus z_i = c_i$.

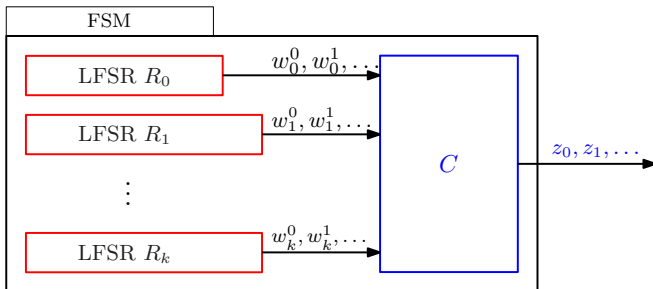
Usually, an attacker

- knows the **IV** and a prefix of the **keystream**

The attackers' goal:

- Compute the **key** or the **initial state**.

Popular hardware-oriented FSM-Design



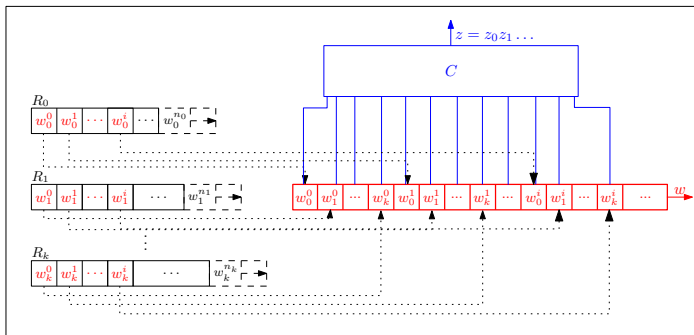
The FSM

- stores the state in LFSRs,
- produces keystream by combining the LFSRs' output bits.

Practical Examples:

- Bluetooth keystream generator E_0
- GSM keystream generator A5/1

Equivalent FSM-Representation



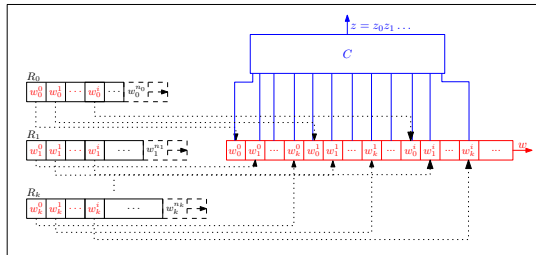
- LFSRs produce **internal bitstream w**
- **Compression function C** computes keystream $z = C(w)$

Cryptanalysis

Problem: Given a prefix of the **keystream** z , compute the **initial states** of the LFSRs (initial state recovery).

Observation

The **initial states** are contained in the first bits of the **internal bitstream** w .

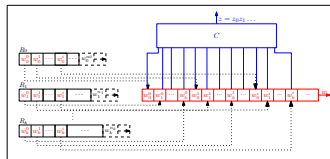


→ Reconstruct the **internal bitstream** w from z .

Reconstruction of w

Restrictions on w

- (1) LFSR-relations hold for the w_j .
- (2) $C(w)$ is a prefix of z .



Generic Algorithm

Compute for $m = 1, 2, \dots, m^*$ the candidate sets

$$\begin{aligned}
 \mathcal{P}_m &= \{w \in \{0, 1\}^m \mid w \text{ fulfills (1) and (2)}\} \\
 &= \mathcal{P}_{m-1} \times \{0, 1\} \\
 &\quad \cap \{(w_0, \dots, w_{m-1}) \in \{0, 1\}^m \mid w_{m-1} \text{ fulfills (1)}\} \\
 &\quad \cap \{w \in \{0, 1\}^m \mid w \text{ fulfills (2)}\}
 \end{aligned}$$

until only bitstreams fully consistent with z are left.

Reconstruction of w

Problem: How to maintain the $\mathcal{P}_m \subseteq \{0, 1\}^m$ efficiently?

Represent $S \subseteq \{0, 1\}^m$ by its characteristic Boolean function

$$\begin{aligned} \delta_S : \{0, 1\}^m &\rightarrow \{0, 1\} \\ z &\mapsto \begin{cases} 1 & z \in S \\ 0 & z \notin S \end{cases} \end{aligned}$$

and find an efficient data structure for δ_S .

→ *Binary Decision Diagrams (BDDs)*

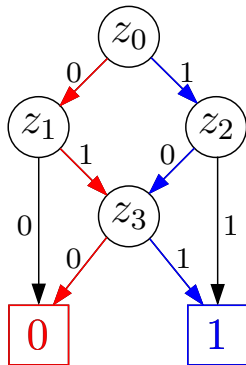
Binary Decision Diagram (BDD)

directed acyclic graph G_f

- 1 source and 2 sinks
- inner nodes have $outdeg = 2$
- node labels
 - sinks: 0 and 1
 - inner nodes: variables from $Z_n = \{z_0, \dots, z_{n-1}\}$
- edge labels from $\in \{0, 1\}$
- represents Boolean function

$$f : \{0, 1\}^m \rightarrow \{0, 1\}$$

$z \mapsto$ end of the z -path



Examples:

$z = (z_0, z_1, z_2, z_3)$	$f(z)$
$(0, 1, 1, 0)$	0
$(1, 0, 0, 1)$	1

Efficiency of BDD-operations

For Boolean functions f, g, h and corresponding BDDs G_f, G_g and G_h , we need to construct

- $G_f \wedge G_g \wedge G_h$ computing $f \wedge g \wedge h$,
- the set of satisfying inputs $G_f^{-1}(1)$ of G_f

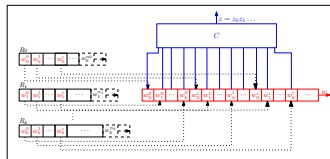
If each variable is read at most once and the reading order is the same on each path in the BDD, we have:

operation	runtime and memory
$G_f \wedge G_g \wedge G_h$	$O(G_f \cdot G_g \cdot G_h)$
$G_f^{-1}(1)$	$O(G_f + n \cdot G_f^{-1}(1))$

Reconstruction of w with BDDs

Restrictions on w

- (1) LFSR-relations hold for the w_j .
- (2) $C(w)$ is a prefix of z .



BDD-Algorithm

Compute for $m = 1, 2, \dots, m^*$ the **BDDs**

$$P_m = P_{m-1} \wedge \underbrace{\{w \in \{0, 1\}^m \mid w_{m-1} \text{ fulfills (1)}\}}_{BDD_m^{(1)}} \\ \wedge \underbrace{\{w \in \{0, 1\}^m \mid w \text{ fulfills (2)}\}}_{BDD_m^{(2)}}$$

until only bitstreams fully consistent with z are left.

Reconstruction of z with BDDs

Runtime of the algorithm depends on

- number of iterations m^* and $\max_m \{|P_m|\}$
- parameters of the keystream generator
 - information rate $\alpha \in (0, 1]$
 - (best case) compression rate $\gamma \in (0, 1]$

Theorem (Krause 2002)

For keystream generators of state size n , $|BDD_m^{(1)}| \in O(2^{m-n})$,
and $|BDD_m^{(2)}| \in m^{O(1)}$,

- $m^* = \lceil \alpha^{-1} n \rceil$
- $|P_m| \leq n^{O(1)} 2^{\frac{1-\alpha}{1+\alpha} n}$ for $m \leq m^*$

→ w can be reconstructed in time and with space $n^{O(1)} 2^{\frac{1-\alpha}{1+\alpha} n}$ from the first $\lceil \gamma \alpha^{-1} n \rceil$ consecutive bits of z .

What about more recent FSM-designs?

Several recent designs (partly) replace the LFSRs by NFSRs and use more complicated compression functions

Examples: eSTREAM Phase 3 ciphers

- TRIVIUM: 3 interconnected NFSRs
- Grain: 1 LFSR + 1 NFSR (interconnected)

What about BDD-attacks against these ciphers?

Observation: Checking the NFSR-relations more complicated,
but attack still works.

More precisely...

BDD-Attack on NFSR-based FSMs

Theorem

For NFSR-based keystream generators of state size n ,

$|\text{BDD}_m^{(1)}| \in O(2^{p \cdot (m-n)})$ for a small cipher-dependent parameter

$p \geq 1$, and $|\text{BDD}_m^{(2)}| \in m^{O(1)}$, we have

- $m^* = \lceil \alpha^{-1} n \rceil$
- $|P_m| \leq n^{O(1)} 2^{\frac{p(1-\alpha)}{p+\alpha} n}$ for $m \leq m^*$

→ w can be reconstructed in time and with space

$$n^{O(1)} 2^{\frac{p(1-\alpha)}{p+\alpha} n}$$

from the first $\lceil \gamma \alpha^{-1} n \rceil$ consecutive bits of z .

Performance on TRIVIUM and Grain

BDD-attack:

	keylength	state size	α	p	time and memory
TRIVIUM	80	288	0.25	1.75	$\approx 2^{189}$
Grain	128	256	0.2	2	$\approx 2^{187}$

Other initial state recovery attacks:

- TRIVIUM:
 - 2^{100} operations from $2^{61.5}$ keystream bits (Maximov/Biryukov)
 - 2^{135} operations from $O(1)$ keystream bits (eStream forum)
 → BDD-attack not competitive
- Grain:
 - 2^{128} operations and keystream bits (generic TMD-tradeoff)
 → BDD-attack best non-trivial short keystream attack

BDD-Attack on F-FCSR

Observation: Theorems 5,6,7 in the preproceedings-version of the paper are flawed.

→ BDD-attack is applicable to F-FCSR, but with lower efficiency.

More details to follow. . .

Conclusion

BDD attack on LFSR-based keystream generators

- needs much memory, but as little keystream as possible
- can be extended to NFSRs
- is applicable to recent designs TRIVIUM and Grain
- as a generic attack, can be outperformed by cipher-specific techniques

The End

`dstegema@th.informatik.uni-mannheim.de`

Application to E_0 and A5/1

	keylength n	α	γ	# required keystream bits
E_0	128	0.25	0.25	n
A5/1	64	0.2193	0.25	$1.14n$

Performance of the BDD-attack:

	theory		simulations
	time and space	q	time and space
E_0	$2^{76.8}$	0.85	$2^{65.28}$
A5/1	2^{41}	0.9	$2^{36.9}$

Simulations on reduced instances indicate that

- practical BDD-sizes only deviate from theory by a constant 2^q
- memory is the main bottleneck