

# Reducing the Space Complexity of BDD-based Attacks on Keystream Generators

Matthias Krause  
**Dirk Stegemann**

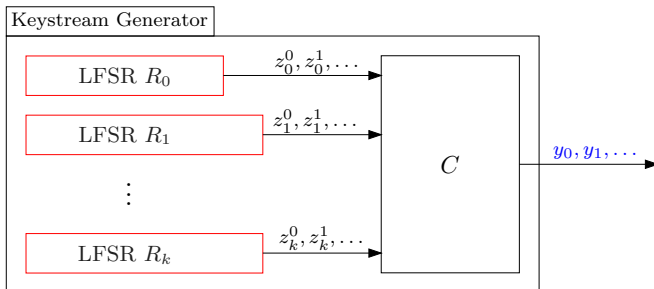
Theoretical Computer Science  
University of Mannheim, Germany

Fast Software Encryption 2006  
March 15-17, Graz, Austria

# Overview

- 1 Introduction
- 2 Binary Decision Diagrams
- 3 BDD-based Attacks and Improvements
- 4 Conclusion

# LFSR-based Keystream Generators



secret key = initial states of the LFSRs

Practical Examples:

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

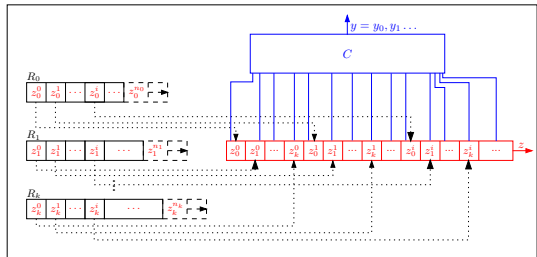


# Cryptanalysis

Problem: Given a prefix of the **keystream**  $y$ , compute the **initial state** of the LFSRs.

## Observation

The **initial state** is contained in the first bits of the **internal bitstream**  $z$ .

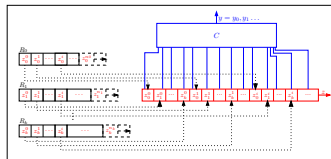


→ Reconstruct the **internal bitstream**  $z$  from  $y$ .

# Reconstruction of $z$

Restrictions on  $z$

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



## Algorithm

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

$$\begin{aligned}
 \mathcal{P}_m &= \{z \in \{0, 1\}^m \mid z \text{ fulfills (1) and (2)}\} \\
 &= \mathcal{P}_{m-1} \times \{0, 1\} \\
 &\quad \cap \{(z_0, \dots, z_{m-1}) \in \{0, 1\}^m \mid z_{m-1} \text{ fulfills (1)}\} \\
 &\quad \cap \{z \in \{0, 1\}^m \mid z \text{ fulfills (2)}\}
 \end{aligned}$$

until only bitstreams fully consistent with  $y$  are left.

# Reconstruction of $z$

**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  $\delta_S$ .

→ *Binary Decision Diagrams (BDDs)*

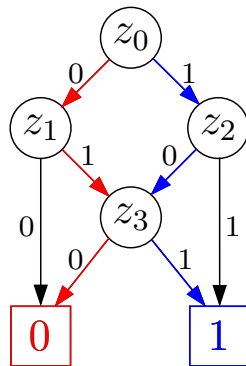
## Binary Decision Diagram (BDD)

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



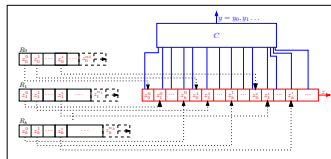
Examples:

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

# Reconstruction of $z$ with BDDs

Restrictions on  $z$

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



## BDD-Algorithm

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

$$\begin{aligned}
 P_m &= P_{m-1} \wedge \underbrace{\{z \in \{0, 1\}^m \mid z_{m-1} \text{ fulfills (1)}\}}_{BDD_m^{(1)}} \\
 &\quad \wedge \underbrace{\{z \in \{0, 1\}^m \mid z \text{ fulfills (2)}\}}_{BDD_m^{(2)}}
 \end{aligned}$$

until only bitstreams fully consistent with  $y$  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 keylength  $n$  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  $m \leq m^*$

→  $z$  and therefore  $x$  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  $y$ .

Application to  $E_0$  and A5/1

	keylength $n$	$\alpha$	$\gamma$	# 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

# Reducing the Memory Requirements

Possible approach: Divide-and-Conquer strategies (DCS)

- 1 Divide the key space into segments  $t_i \in T$
- 2 BDD-attack the segments individually

→ Attack in

- time  $\leq n^{O(1)} \cdot |T| \cdot 2^{w^*(n)}$
- space  $n^{O(1)} 2^{w^*(n)}$

for a generator-specific parameter  $w^*(n)$

In order to gain from the DCS, we need

- $|T|$  not too large
- $w^* < \frac{1-\alpha}{1+\alpha} n$

Example: Guess the shortest LFSR

# Guessing the shortest LFSR of $E_0$ and A5/1

Performance of the modified BDD-attack

	theory			simulations	
	time	space	$q$	time	space
$E_0$	$2^{76.8}$	$2^{76.8}$	0.85	$2^{65.28}$	$2^{65.28}$
$E_0^{DCS}$	$2^{76.5}$	$2^{51.5}$	0.95	$2^{72.68}$	$2^{48.93}$
A5/1	$2^{41}$	$2^{41}$	0.9	$2^{36.9}$	$2^{36.9}$
A5/1 <sup>DCS</sup>	$2^{51.86}$	$2^{32.85}$	0.77	$2^{39.93}$	$2^{25.3}$

Observations

- time requirements slightly increase
- memory requirements significantly decrease

# Conclusion

Experimental results for BDD-attacks indicate that

- practical time and memory requirements do not seem to deviate from theory by more than a constant
- memory is the main bottleneck

Divide-and-Conquer Strategies can

- lower the memory requirements
- at the expense of slightly increased runtime

→ Are there better Divide-and-Conquer Strategies than guessing the shortest LFSR?

# Thank you!

`{krause,stegemann}@th.informatik.uni-mannheim.de`