Improving OBDD-Attacks against Stream Ciphers

Matthias Hamann, Matthias Krause, Alexander Moch

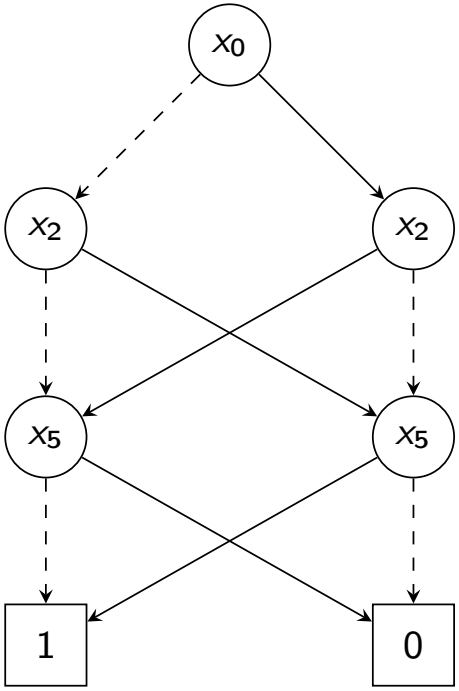
Uni Mannheim

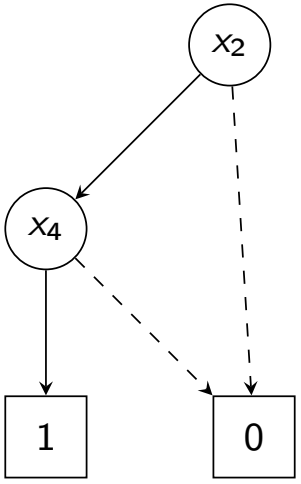
Svetlogorsk, Russia, June 5, 2019

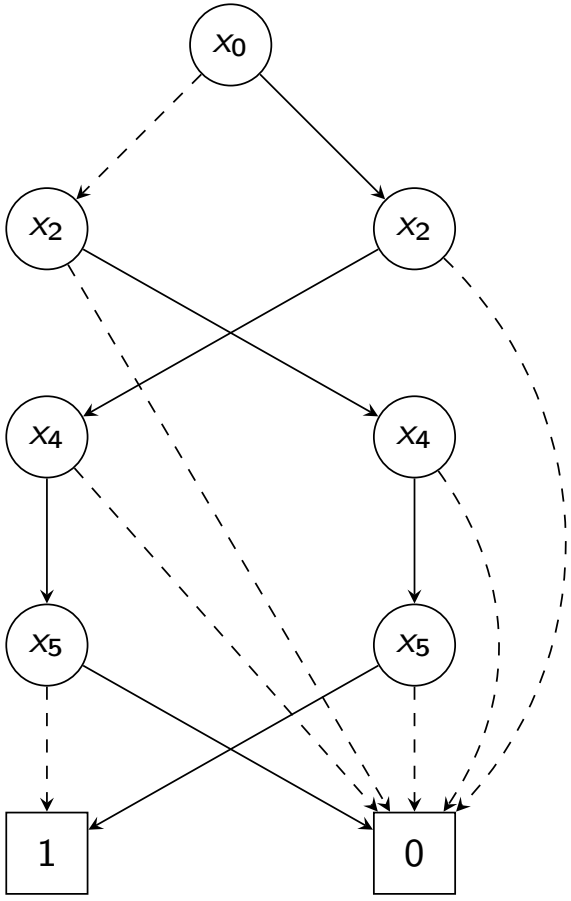
Overview

Ordered Binary Decision Diagrams (OBDDs)

. . . data structure for representing Boolean functions.







Stream Ciphers are . . .

- . . . intended to encrypt a bitstream X passing an insecure channel between Alice and Bob (mobil phones, WLAN etc.)
- . . . generate a keystream $STREAM(q(k, IV))$ depending on a secret symmetric session key k, shared by Alice and Bob, and a public initial value IV.
- **Encryption:** $Y \leftarrow X \oplus STREAM(q(k, IV))$
- **Decryption:** $X \leftarrow Y \oplus STREAM(q(k, IV))$
- **Phase 1:** Compute $q(k, IV)$, the initial inner state for the **Keystream Generator (KSG)**.
- **Phase 2:** KSG computes $STREAM(q(k, IV))$ on $q(k, IV)$.

Keystream Generators (KSG)

. . . are **stepwise working finite state machines** (usually built by LFSRs and NFSRs) defined by

- a set of inner states $\{0,1\}^n$, n inner state length
- a state update function $\pi : \{0,1\}^n \longrightarrow \{0,1\}^n$,
- an output function $outbit : \{0,1\}^n \longrightarrow \{0,1\}$.

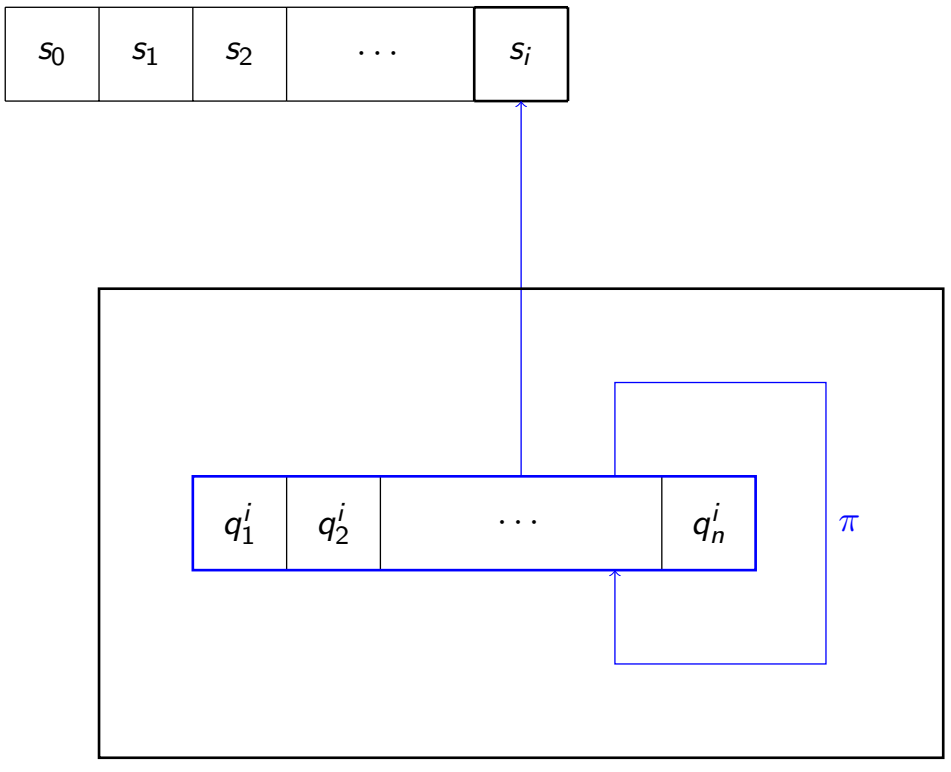
The **Keystream**

$$STREAM(q^0) = s_0 s_1 s_2 \cdots$$

on **initial state** $q^0 \in \{0,1\}^{SL}$ is defined as

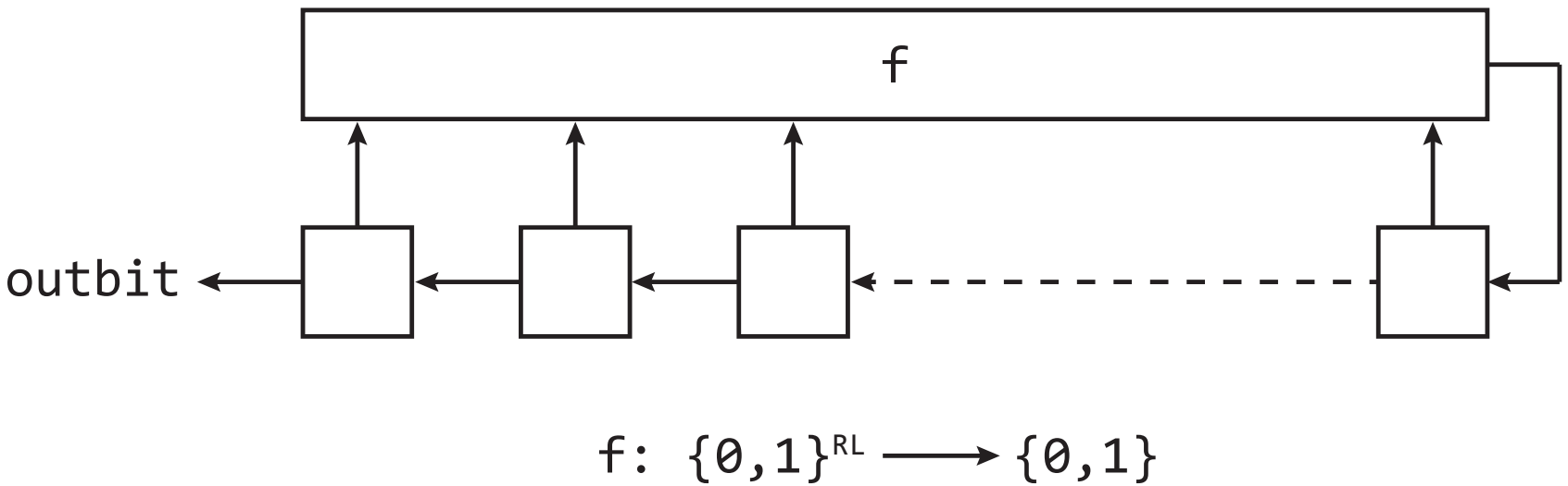
- $s_i = outbit(q^i)$ for $i = 0, 1, 2, \cdots$
- $q^{i+1} = \pi(q^i)$ for $i = 0, 1, 2, \cdots$
- **Note** $s_i = outbit(\pi^i(q^0))$.

Keystream Generators, Picture

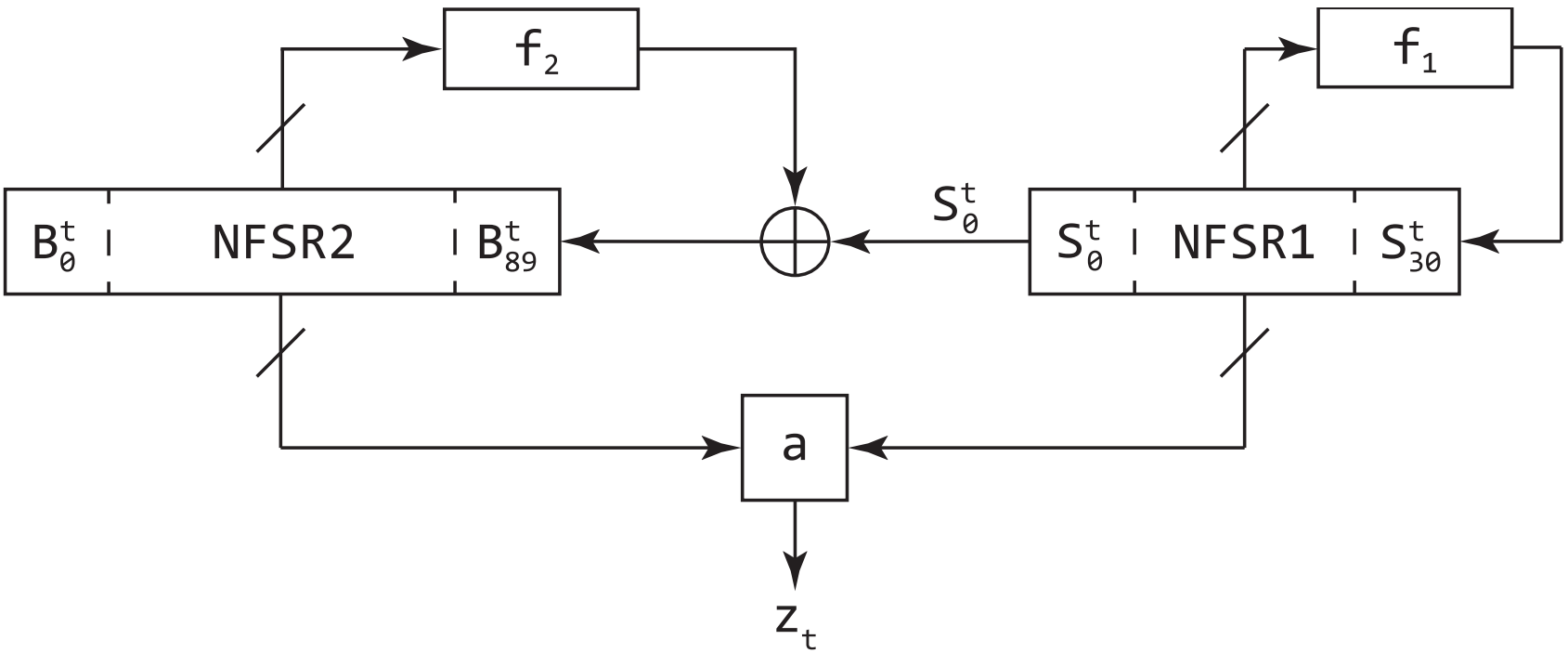


Feedback Shift Registers (FSRs)

Typical KSG building blocks are **Feedback Shift Registers (FSRs)**



The LIZARD-KSG (Hamann, Krause, Meier 2018)



Security of Stream Ciphers

Main Security Requirement

Hardness of distinguishing $STREAM(q_0)$, $q_0 \in_U \{0,1\}^n$, from a truly random bitstream.

$\Longrightarrow$ Hardness of computing $q \in \{0,1\}^n$ from given S prefix of $STREAM(q)$.

Attacks on Stream Ciphers

- **Long Keystream Attacks:** Time-Memory-Data Tradeoff Attacks ($|S| = 2^{n/2}$), (Fast) Correlation Attacks, Algebraic Attacks, ...
- **Short Keystream Attacks ($|S| = n$):** Exhaustive Search (Time $O(2^n)$), OBDD-attacks (Time and Space $O(2^{\beta \cdot n})$, $\beta < 1$).

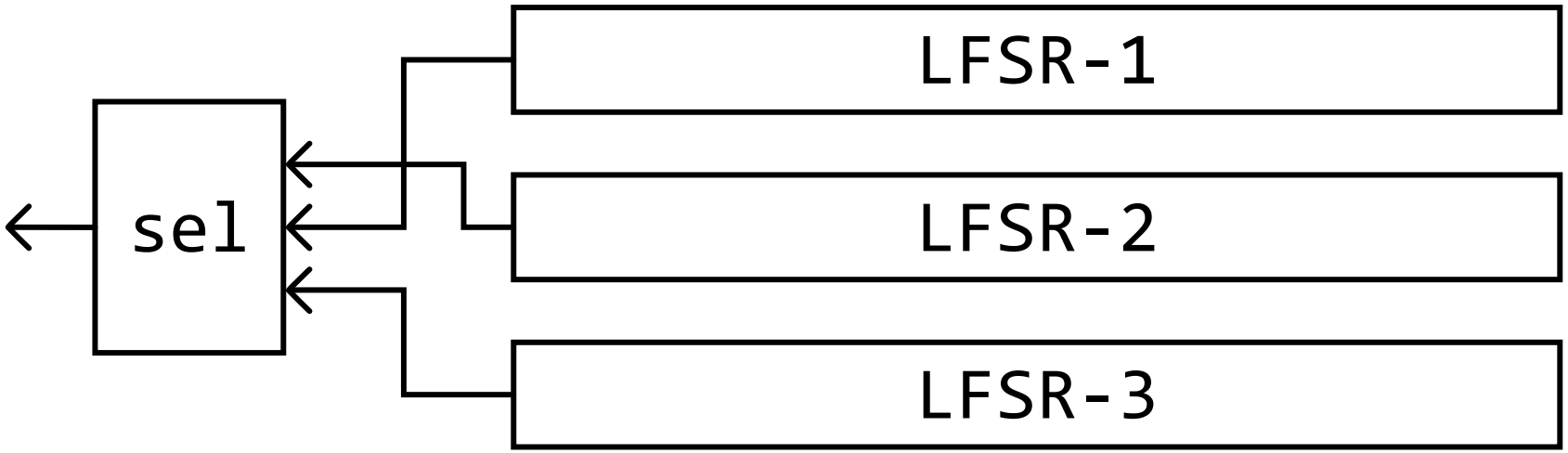
OBDD-attacks

OBDD-attacks (Krause, Eurocrypt 2002):

- ... refer to **FSR-based KSGs** (GSM standard A5/1, Bluetooth standard E_0, Shrinking generators, Grain, Trivium etc.)
- ... are based on looking at the keystream generation process as dividid into
  - the generation of a secret inner stream $y = \mathit{InnerStream}(q)$,
  - and the output bitstream $\mathit{STREAM}(q) = \mathit{Compress}(y)$
- **Toy Example:** Geffe generator (3 LFSRs)

$$z^t = \mathit{sel}(y_2^t, y_1^t, y_3^t) = (y_2^t \wedge y_1^t) \vee (\bar{y_2^t} \wedge y_3^t).$$

The Geffe Generator



- $Innerstream(q) = (y_1^0, y_2^0, y_3^0, \cdots, y_1^t, y_2^t, y_3^t, \cdots)$,
  where q is prefix of length n of $Innerstream(q)$, and

$$y_j^t = L_j^t(x_1, \cdots, x_n)$$

  for $t > n_j$.

- **Compression:** $STREAM(q) = (z^0, z^1, \cdots, z^t, \cdots)$ with

$$z^t = sel(y_2^t, y_1^t, y_3^t).$$

Preconditions for OBDD-attacks

Definition

A FSR-based KSG fulfills the Easy-Relations-Condition if the decision

$$y = \mathit{InnerStream}(q) \ \wedge \ z = \mathit{Compress}(y)$$

can be expressed as a system $\mathcal{R}$ of $O(n)$ **inner stream relations** in the q, y-variables and **output relations** in the y, z-variables, resp., which all have small OBDDs.

Example Geffe Generator

$\mathcal{R}$ contains the following OBDD-width 2 relations:

- **Inner stream relations** $y_j^t = L_j(y_j^{t-1}, \cdots, y_j^{t-n_j})$, $j = 1, 2, 3$, where L_j is the feedback relation of LFSR-j (length n_j).
- **Output relations** $z^t = (y_2^t \wedge y_1^t) \vee (\bar{y_2^t} \wedge y_3^t)$

OBDD Attack Main Theorem

Definition

The compression rate of a FSR-based KSG is defined to be $\alpha = \frac{1}{A}$, where A denotes the average number of inner stream bits needed for generating one keystream bit.

Example Geffe Generator $\alpha = \frac{1}{3}$.

Theorem (Krause 2002)

If a FSR-based KSG with compression rate α satisfies the Easy-Relations-Condition then the secret initial state can be computed in expected time and space $O(n \cdot 2^{\frac{1-\alpha}{1+\alpha}n})$ on the basis of $\approx n$ consecutive keystream bits. $\square$

Example Geffe Generator Expected time and space $O(2^{n/2})$.

Proof Idea Main Theorem

The Standard OBDD Attack:

Consider $\mathcal{R} = \{R_1, \cdots, R_s\}$ with small OBDDs $P_1, \cdots, P_s$.

(1) $P \leftarrow P_1$, $i \leftarrow 1$
(2) **Repeat** $i \leftarrow i + 1$; $P \leftarrow \min(P \wedge P_i)$
(3) **until** $|P^{-1}(1)| = 1$

OBDD-attacks rely on **three algorithmic properties:**

- Each OBDD P can be minimized in time $O(size(P))$.
- For each minOBDD P it holds $width(P) \leq |P^{-1}(1)|$.
- For each two OBDDs P, Q (defined over the same variable ordering) an OBDD $P \wedge Q$ computing $P^{-1}(1) \cap Q^{-1}(1)$ can be computed in time and space $O(size(P) \cdot size(Q))$.

Proof Idea Main Theorem

If the order $P_1, \cdots, P_s$ is choosen in a clever way then

- $|P_1^{-1}(1)| \approx 2^{(1-\alpha)n}$.
- In each iteration $width(P)$ grows by at most a factor two and $|P^{-1}(1)|$ decreases by a factor α.

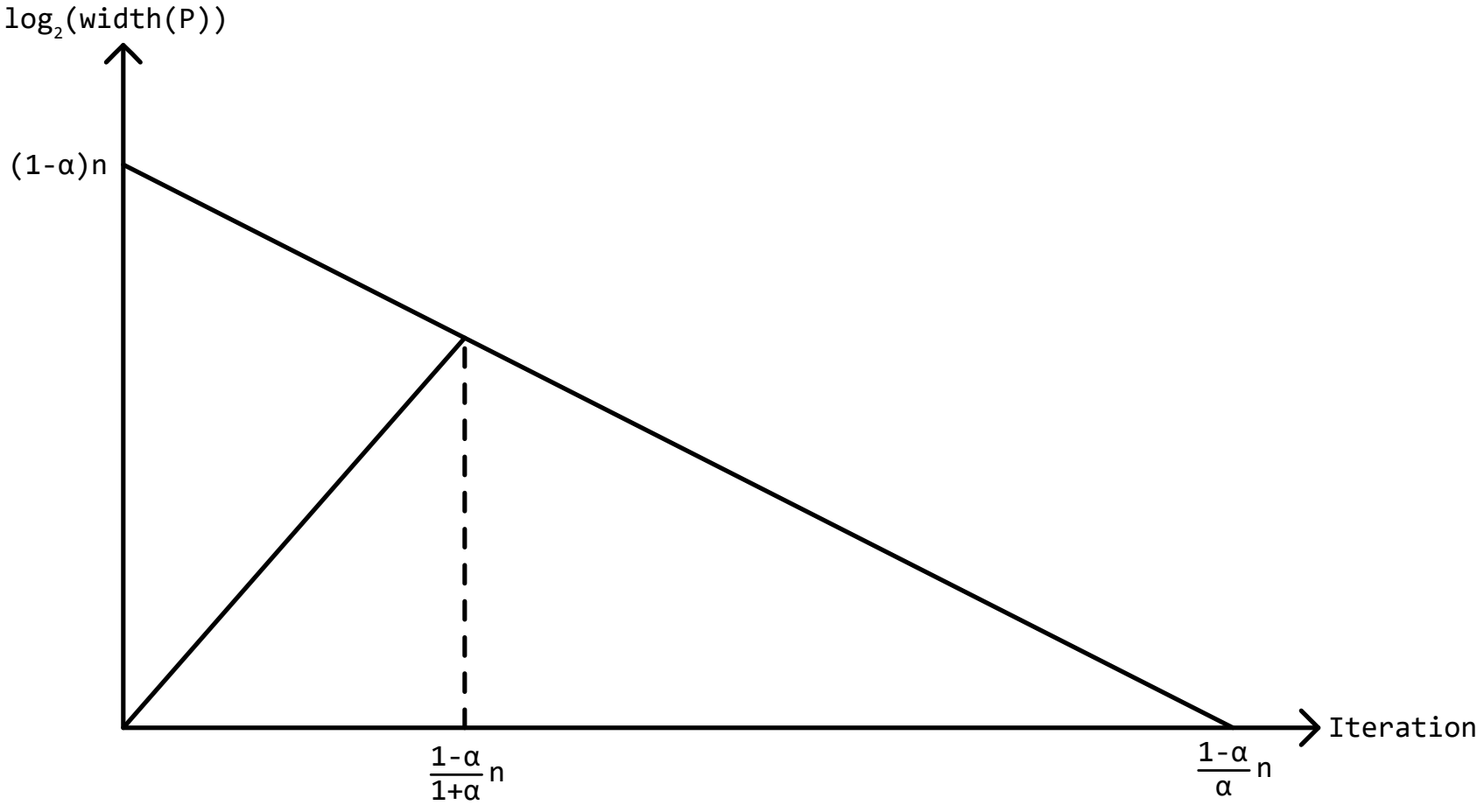
Consequence: After t iterations

- $\log_2(width(P)) \leq \min\{t, (1-\alpha)n - \alpha \cdot t\}$,
- $|P^{-1}(1)| \approx 1$ after $\frac{1-\alpha}{\alpha} n$ iterations.
- $\max \log_2(width(P)) \approx \frac{1-\alpha}{1+\alpha} n$ around the **critical iteration** $t_{max} \approx \frac{1-\alpha}{1+\alpha} n$. $\square$

Problem: Large intermediate OBDD-width $O(2^{\frac{1-\alpha}{1+\alpha} n})$.

OBDD attack behavior

Growth of OBDD width during an OBDD attack:



One promising approach to improve OBDD-attacks

Choose $\mathcal{R}_1$ and $\mathcal{R}_2$ of $\mathcal{R}$ in a clever way such that $|\mathcal{R}_1|, |\mathcal{R}_2| < t_{\max}$ and $|\mathcal{R}_1 \cap \mathcal{R}_2| > t_{\max}$ such that

- The sizes of $P(\mathcal{R}_1)$ and $P(\mathcal{R}_2)$ are moderately bounded.
- The number of satisfying assignments of $P(\mathcal{R}_1) \wedge P(\mathcal{R}_2)$ is small (e.g., one).

Experiments with NFSR-KSG ($n = 39$) with

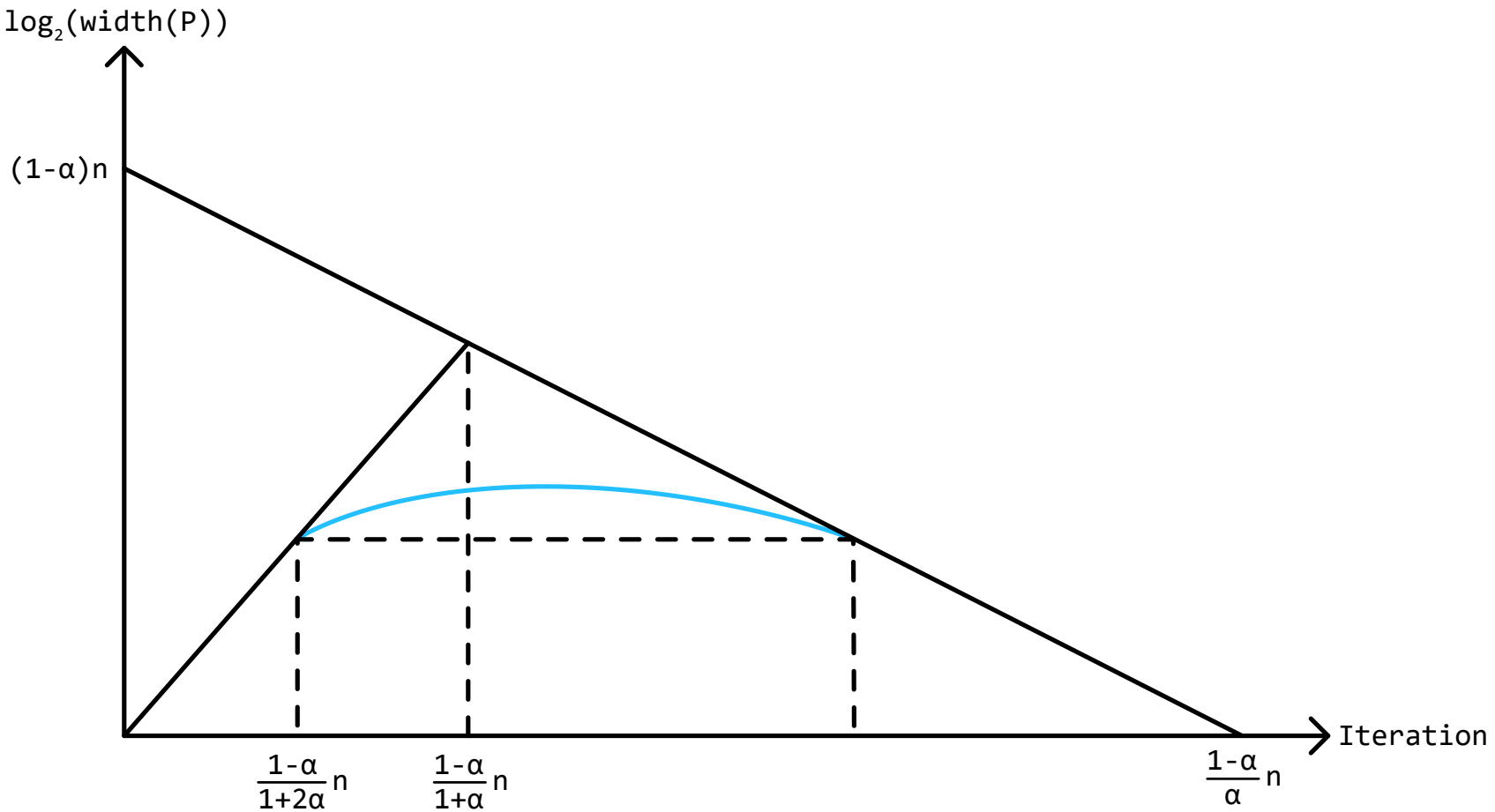
feedback $x_{t+39} := x_t \oplus x_{t+13} \oplus x_{t+5} \cdot x_{t+17} \oplus x_{t+24} \cdot x_{t+29}$

output $z_t := x_{t+9} \oplus x_{t+19} \oplus x_{t+29}$.

- **Standard Attack** (with CUDD) 900 MB, 215 seconds
- **Improved Attack** (with two OBDDs) 110 MB, 8 seconds

Improved OBDD attack behavior

Growth (theoretically) of OBDD width during an improved OBDD attack:



$\Longrightarrow$ Two algorithmic challenges

- **How to choose $\mathcal{R}_1$ and $\mathcal{R}_2$?**.
- **How to compute efficiently** $(P(\mathcal{R}_1) \wedge P(\mathcal{R}_2))^{-1}(1)$.

  (under the condition that it is know that this set is small).

$\Longrightarrow$ The Bounded $\wedge$-Synthesis Problem

Definition

- **Input:** P, Q OBDDs with $|(P \wedge Q)^{-1}(1)| = 1$.
- **Output:** $(P \wedge Q)^{-1}(1)$

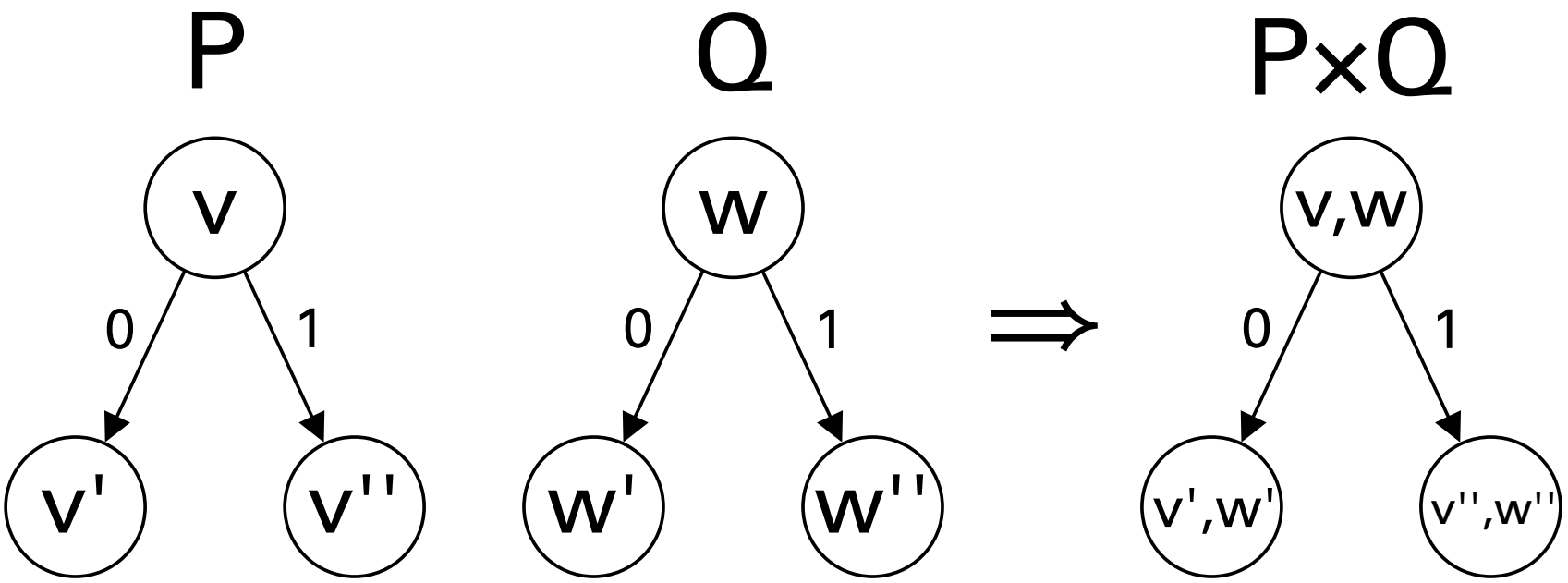
Standard Solution: Compute $\min(P \wedge Q)$ by the standard OBDD synthesis algorithm (idea see next slide), time and space $O(size(P) \cdot size(Q))$.

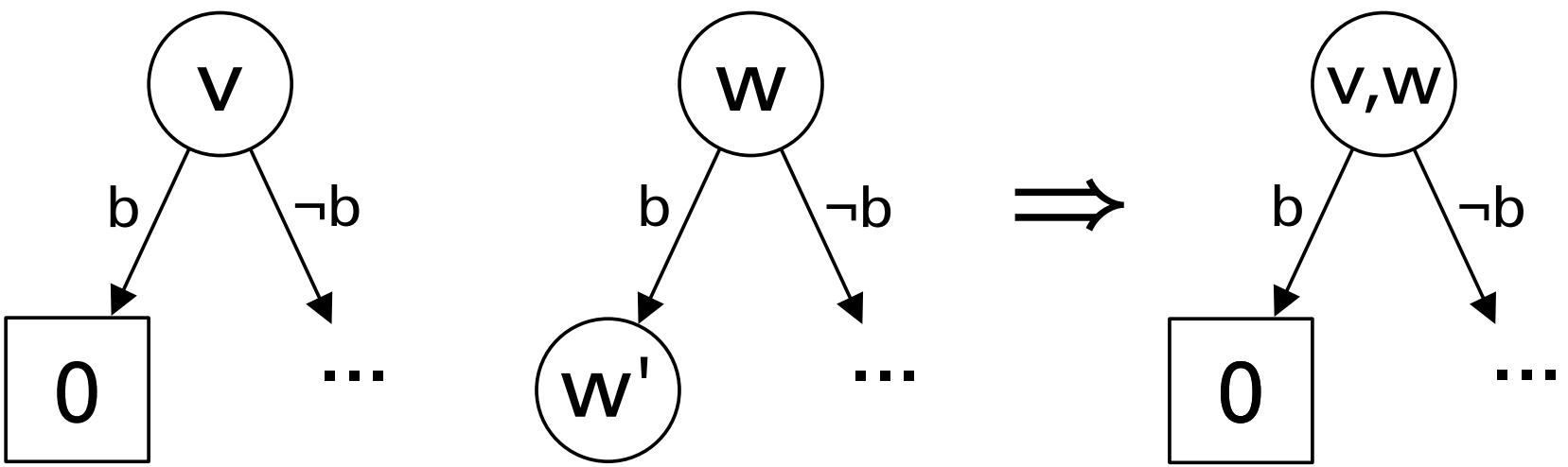
Main question: Can we do better?

Lower Bound: $\Omega(size(P) \cdot size(Q))$ by constructing P and Q with $size(min(P \wedge Q)) = \Omega(size(P) \cdot size(Q))$.

However: In our situation we know that $width(\min(P \wedge Q)) = 1$, i.e., $size(min(P \wedge Q)) \leq n$.

The OBDD $P \wedge Q$ (for levelled OBDDs)





The DFS-Approach ...

Bounded Synthesis Problem $\Longrightarrow$ Reachability Problem

- **Given** P and Q, $|(P \wedge Q)^{-1}(1)| = 1$
- **Compute** a path from the root to the 1-sink in $P \wedge Q$.

Is this possible without generating the whole unminimized OBDD $P \wedge Q$?

How we can algorithmically make use of the promise that $|(P \wedge Q)^{-1}(1)| = 1$.

The DFS-Approach ...

... uses a stack S, which is initially empty

1 $root(P \wedge Q).color \leftarrow gray$
2 **push**$(root(P \wedge Q))$
3 **while** $1sink \notin S$
4 $\quad (v, w) \leftarrow head(S)$
5 $\quad$ **if** $\exists$ uncolored successor (v', w') of (v, w)
6 $\quad\quad$ **then** $(v', w').color \leftarrow gray$, **push**$((v', w'))$
7 $\quad\quad$ **else** $(v, w).color \leftarrow black$, **pop**

Problems

- Black nodes have to be stored
- no usage of the fact that $|(P \wedge Q)^{-1}(1)| = 1$.

The BFS-approach ...

... uses a queue Q, which is initially empty

```
 1  root(P ∧ Q).color ← gray
 2  enqueue(root(P ∧ Q))
 3  while 1sink ∉ Q
 4        (v, w) ←dequeue(Q)
 5        if (v, w).color = gray
 6              then for all successors (v′, w′) of (v, w)
 7                    do if (v′, w′).color = gray
 8                          then (v′, w′).color ← black
 9                          else if (v′, w′) uncolored
10                                then (v′, w′).color ← gray
11                                     enqueue((v′, w′))
```

Advantages of BFS

- BFS saves (potentially) storage as always only nodes of depths s and $s+1$ for some $1 \leq s \leq n$ are stored.
- BFS makes use of the fact that each $P \wedge Q$-node (and all its successors), which can be reached at more than one path from the root can be cancelled.
- This is done via the tests in lines 5 and 7. If node in Q is discovered more than once, it is colored black and not considered any longer.

Experimental Results (Storage in Nodes)

Geffe-KSG ($n = 28$):

- DFS colored nodes: $4,003,455$
- BFS queue: $667,694$
- Synthesis-based: $6,530,494$

NFSR-KSG ($n = 26$):

- DFS colored nodes: $7,631,335$
- BFS queue: $557,169$
- Synthesis-based: $10,977,359$

Concluding Remarks

- Find better algorithms for the Bounded Synthesis Problem.
- Find better algorithms for computing $|P^{-1}(1) \cap Q^{-1}(1)|$ under the condition that this set contains only a few elements.
- **Observation:** A modified BFS-approach works also for this case.
- Apply improved OBDD attacks to lightweight block ciphers (Standard OBDD attacks do not work here)

Concluding Remarks: Generalized OBDD-attacks

- **Given:** Set $\mathcal{R} = \{R_1, \cdots, R_t\}$ of relations over the set of inner stream variables Y, induced by a stream cipher and a given piece b of keystream such that the secret initial state behind b is the only satisfying assignment of $\bigwedge_{i=1}^{t} R_i$.
- **Given:** An ordering τ of $\{1, \cdots, t\}$ and a variable ordering π on Y, such that all relations in $\mathcal{R}$ have small π-OBDDs $\{P_1, \cdots, P_t\}$.
- **The (τ, π)-Attack on $\mathcal{R}$:**

```
1  P ← P_τ(1)
2  For i ← 2 to t
3      do P ← P ∧ P_τ(i)
```

- **The standard attack:** (τ, π) are defined by the order of time inner stream bits and output bits are generated.

Concluding Remarks: Better Orderings?

- **Experiments** show that there are better orderings than the standard orderings.
- $\Longrightarrow$ **Optimization Problem** *OptOBDD*: Find (τ, π) such that $\mathit{MaxOBDDSize}(\mathcal{R}, \tau, \pi)$, the maximal *OBDD*-size occuring during the (τ, π)-Attack on $\mathcal{R}$, is minimal.
- **Important Observation:** There is an **efficient algorithm** computing $\mathit{MaxOBDDSize}(\mathcal{R}, \tau, \pi)$ for given $\mathcal{R}, \tau, \pi$, i.e., *OptOBDD* is *NP*-easy.
- **Future Research:** Solve *OptOBDD* for practical ciphers like the EStream finalist ciphers GRAIN-128 (inner state length: 256, keylength 128) and TRIVIUM (inner state length: 288, keylength 80)