Cryptanalysis of Polar Bear

John Mattsson

Mahdi M. Hasanzadeh
Elham Shakour
Shahram Khazaei

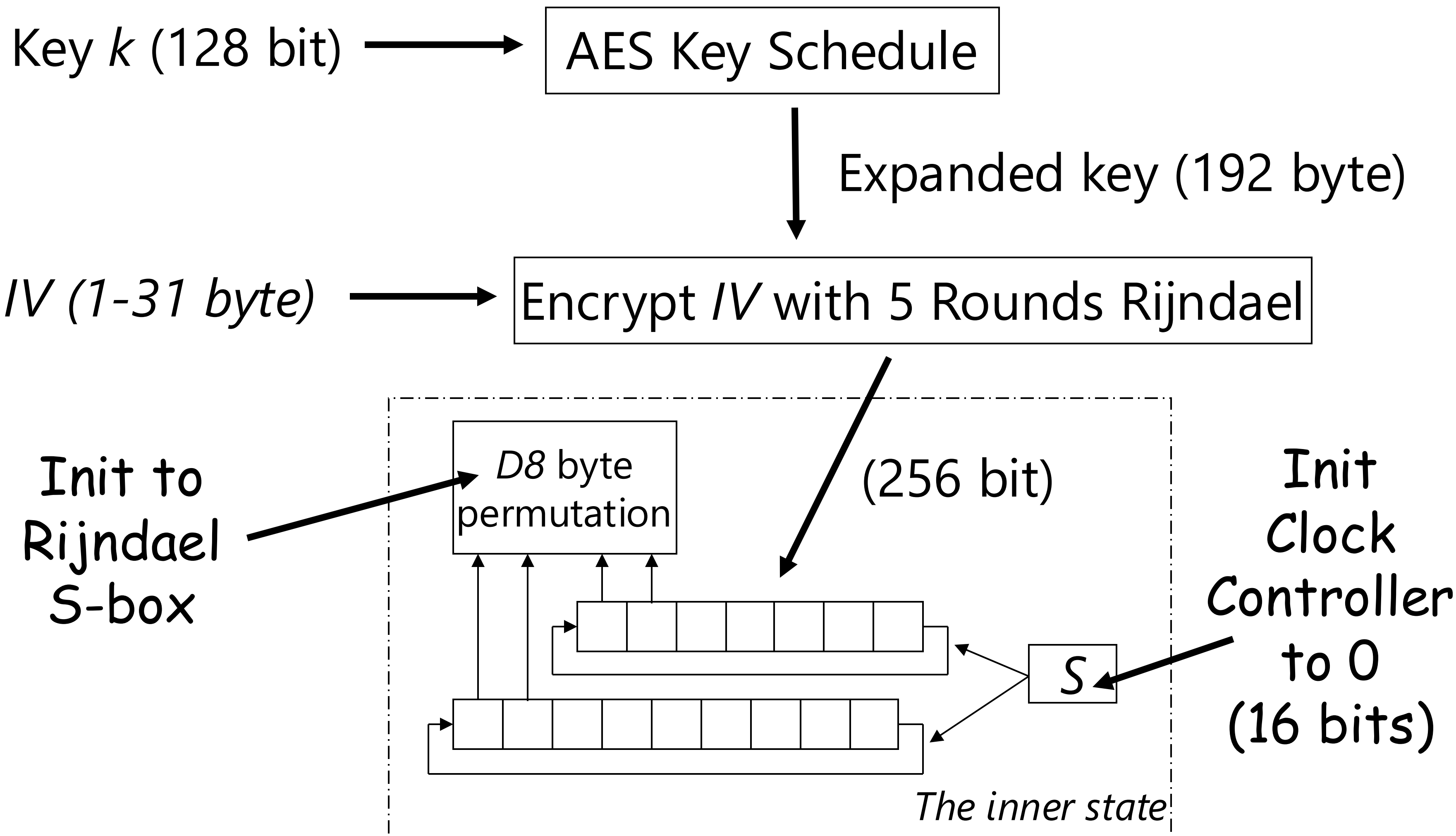
Overview

- Description of Polar Bear
- Guess-and-determine attack
- First attack (Mattsson)
- Improved attack (Hasanzadeh *et al*)
- Analysis
- Fix

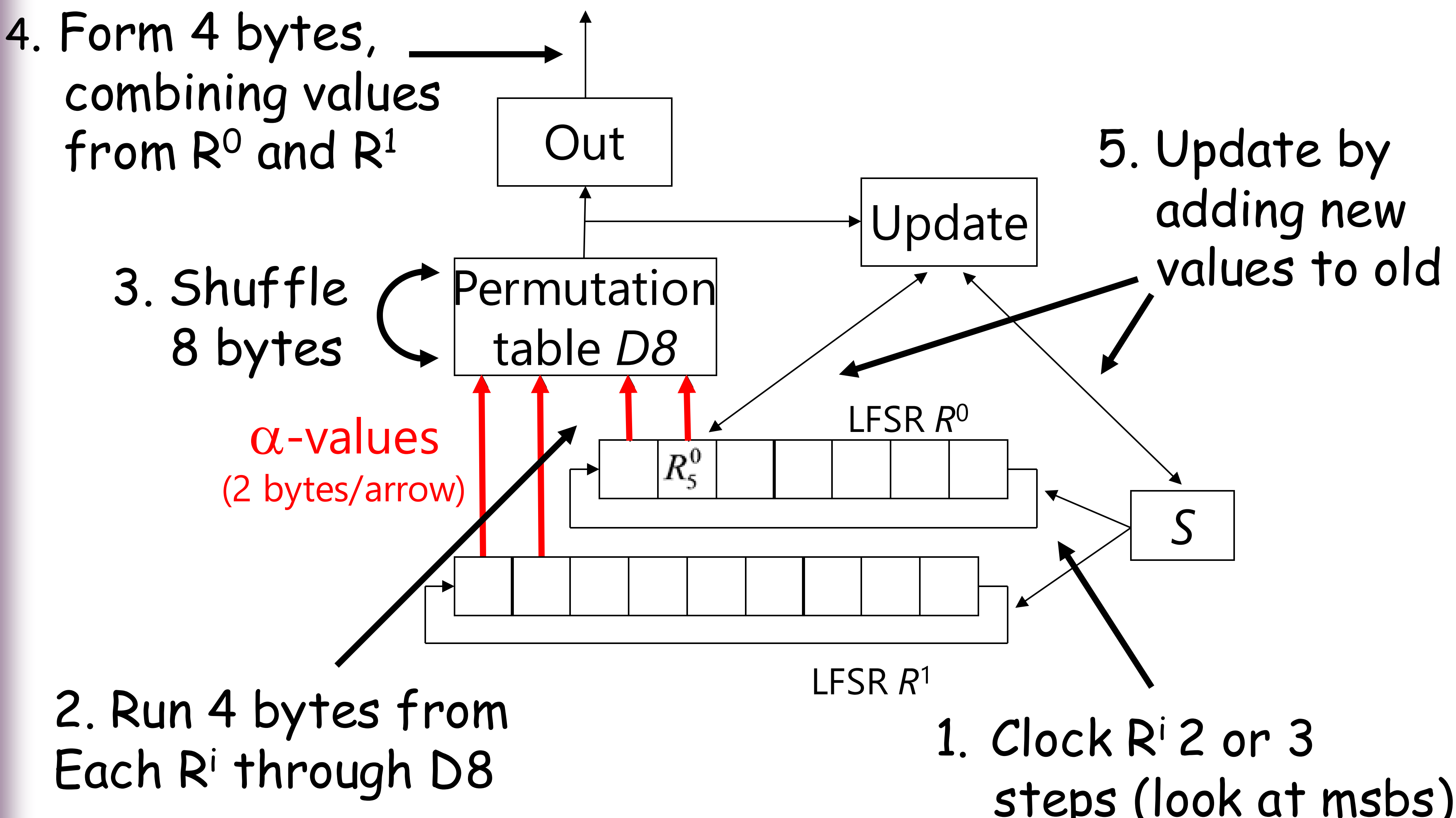
Polar Bear

- Synchronous stream cipher by Mats Näslund and Johan Håstad.
- Submitted to eStream.
- Based on RC4 table shuffling but with two irregularly clocked LFSRs for stepping.

Polar Bear Key and IV Schedule

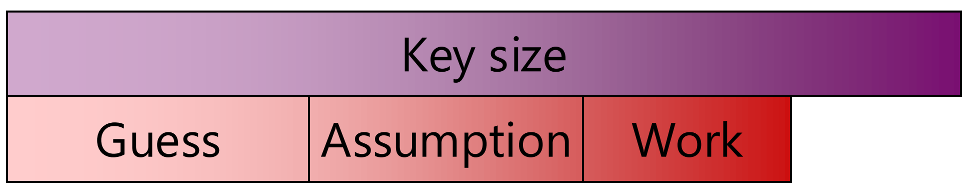


Polar Bear Output Generation



Guess-and-Determine Attack

1. Guess some parts of the internal state
2. Determine other parts of the state under some assumption.
3. Check if the guess is right and the assumption holds



- If $2^{\text{Guessed bits}} \cdot (1/\text{probability}) \cdot \text{Determine work} < 2^{\text{key size}}$

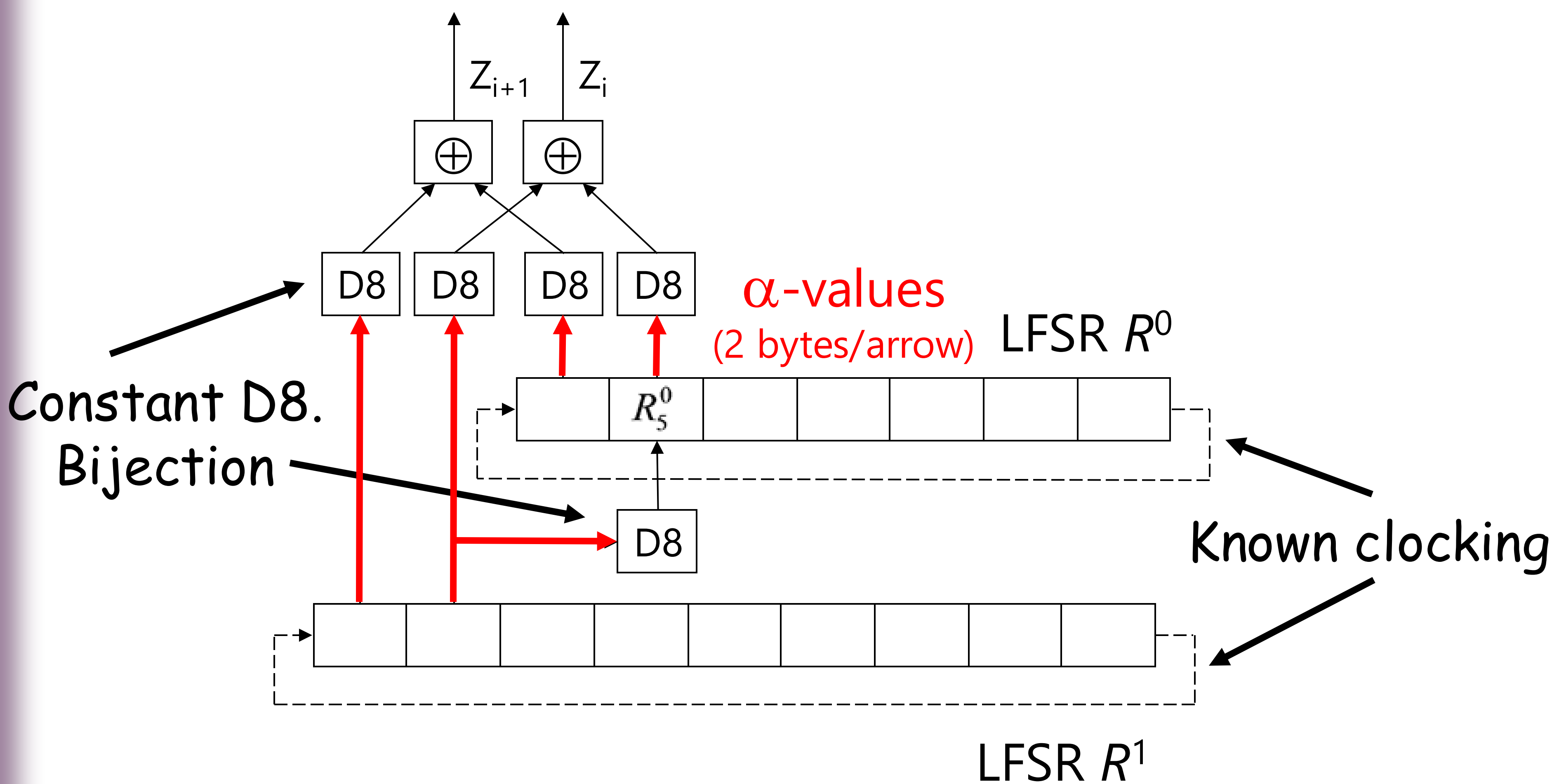
  $\Rightarrow$ Successful attack

First Attack (Mattsson)

- Assumptions:
  1. Stepping (2, 2, 2, 2, 2, 2) for R^0 and R^1
  2. First 8 α-values different
  3. Next 40 α-values different

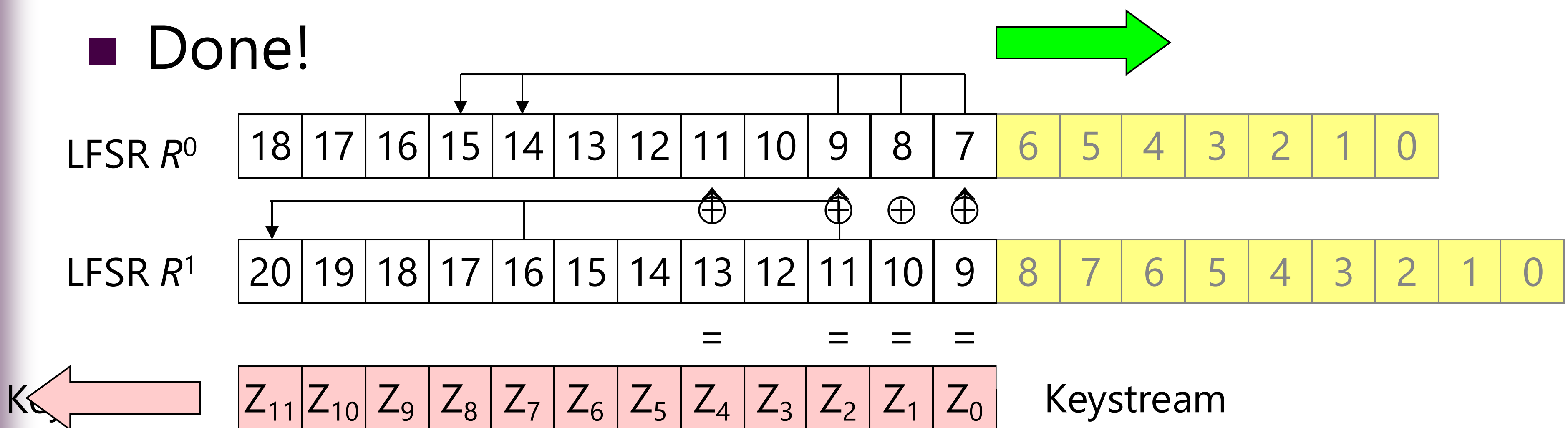
- Assumed probability = $2^{-10} \cdot 2^{-4.8} = 2^{-14.8}$

Under these assumptions Polar Bear looks like this



First Attack (Mattsson)

- Stepping (2, 2, 2, 2, 2, 2) for R^0 and R^1
- Guess 64 bits in R^1
- Done!



- Computational complexity: $O(2^{64}) \cdot O(2^{14.8}) = O(2^{78.8})$

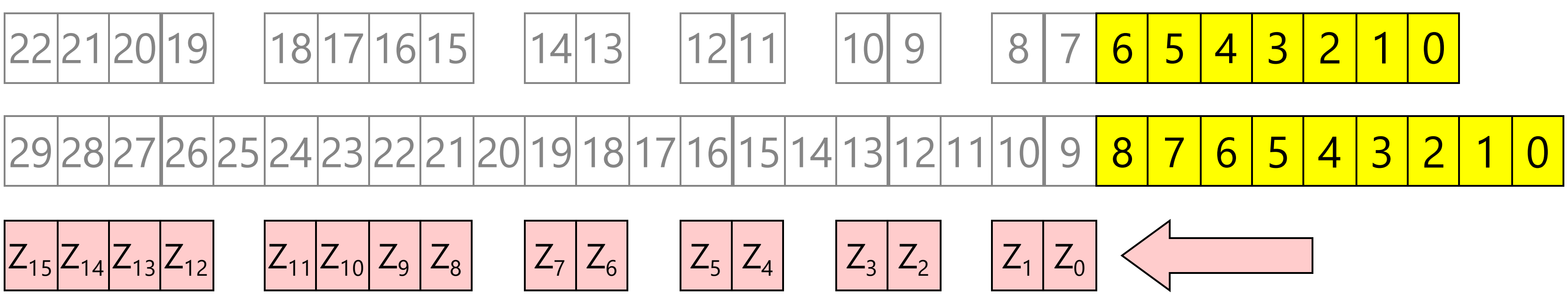
Improved Attack (Hasanzadeh *et al*)

- Assumptions:
  1. Stepping (2, 2, 2, 2, 2, 2, 2, 2) for R^0 and (2, 3, 3, 3, 3, 2, 3, 2) for R^1
  2. First 64 α-values different

- Assumed probability = $2^{-14} \cdot 2^{-12.4} = 2^{-26.4}$

Improved Attack (Hasanzadeh *et al*)

- Stepping (2, 2, 2, 2, 2, 2, 2, 2) for R^0 and (2, 3, 3, 3, 3, 2, 3, 2) for R^1
- Guess 31 bits in R^1
- Done!



- Computational complexity: $O(2^{31}) \cdot O(2^{26.4}) = O(2^{57.4})$

Why were these attacks possible?

- Table *D8* initially known, mixes slowly.
- Short LFSRs. It is even possible to guess the entire small LFSR and still be under $O(2^{128})$.
- Trinomials as feedback polynomials.
- The LFSR stages are too related by stepping, output, and update.

Fix

- We propose that the security is enhanced by adding a key-dependant premixing of D8

  1. Expand the key to 768 bytes
  2. For $i = 0$ to 767

     Swap(*D8*[*i* (mod 256)], *D8*[key[*i*]])

- We belive this is the fastest and simplest way. Only adding computational cost during the key schedule.

Summary

- Two attacks were presented. One "simple" $O(2^{78.8})$ and one more sophisticated $O(2^{57.4})$
- Analysis
- A fix was suggeste