An Improved Hardware Implementation of the Grain-128a Stream Cipher

Shohreh Sharif Mansouri and Elena Dubrova
Department of Electronic Systems
Royal Institute of Technology (KTH), Stockholm
Email:{shsm,dubrova}@kth.se

Overview

- Motivation
- Structure of Grain-128a
- 4 techniques to improve implementation
- Experimental results
- Conclusion

The Main Goal

- Improving Grain-128a in terms of **Throughput**, Area and Power

- We achieve it by modifying the architecture of Grain without changing its algorithm

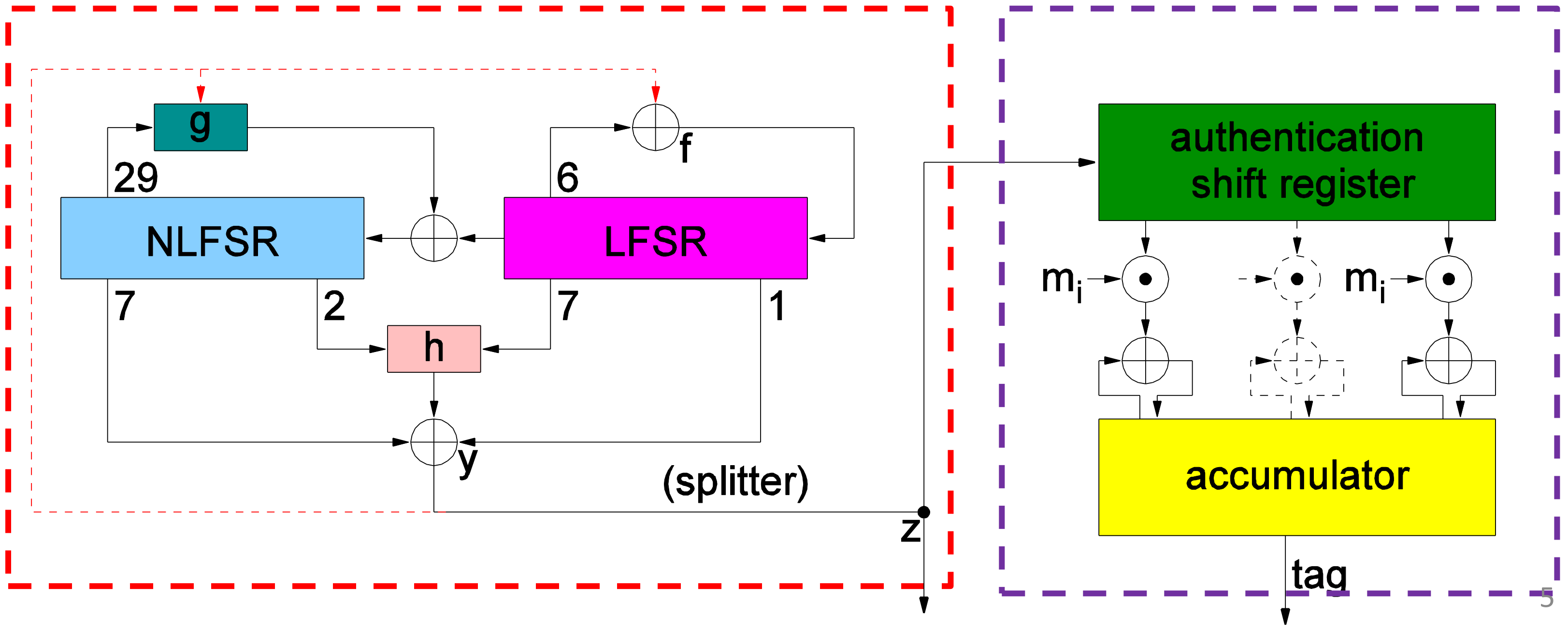
- We succeed to increase the throughput by 52% on average

Grain Family of Stream Ciphers

- Support 80-bits-key and 128-bits-key algorithms
- Support 4, 8 , 16 and 32 (for Grain-128) degrees of parallelization
- New version of Grain-128 (Grain-128a) supports authentication, with a maximal tag length of 32 bits

Grain-128a

- The cipher is divided into two parts:
  - Keystream generator section
  - Authentication section

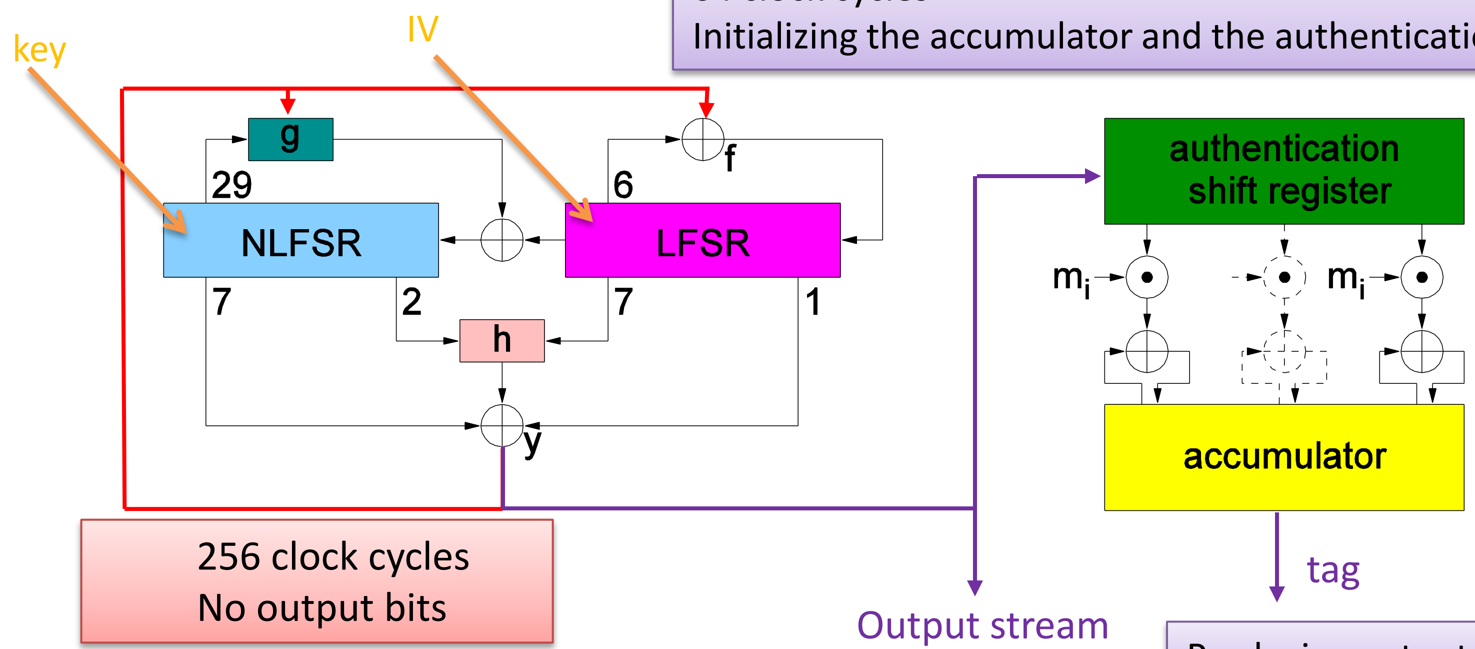


Cipher Phases

The cipher goes through the following phases:

- loading with the key and the initial value (IV)
- Keystream initialization phase
- Keystream generation phase
  - Authentication initialization phase
  - Operational phase

64 clock cycles
Initializing the accumulator and the authentication shift register

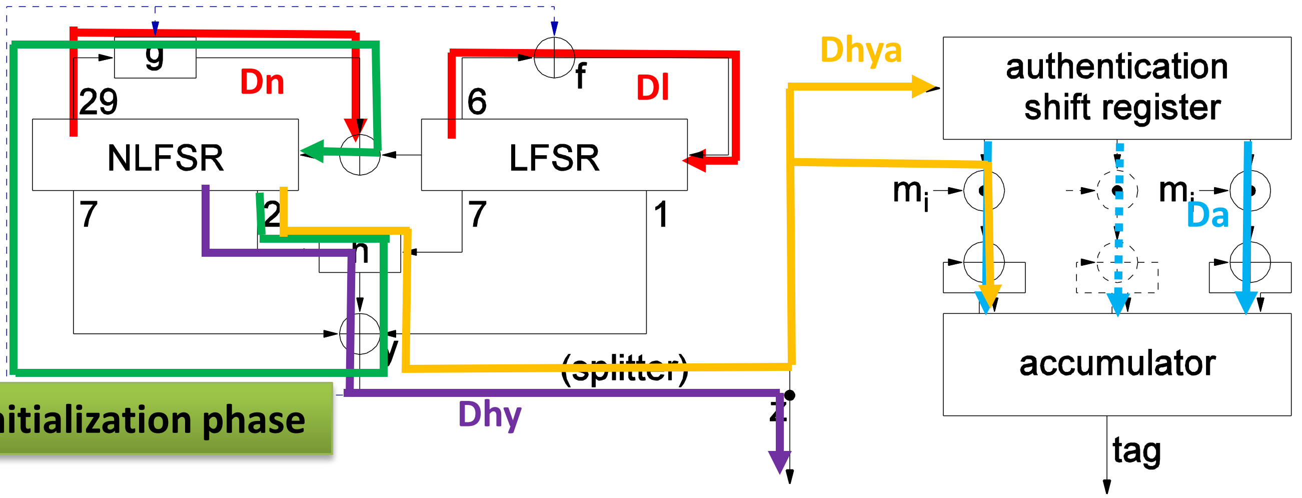


256 clock cycles
No output bits

Producing output stream and tag

How to Improve Throughput?

- Throughput is determined by the critical path, which is the longest combinational path in the system
- 5 potential candidates to critical path:
  - **Dn**: maximal delay from any NLFSR flip-flop to any other NLFSR flip-flop
  - **Dhy**: maximal delay from any NLFSR or LFSR flip-flop through the **h** and **y** functions to the output of the cipher
  - **Dhya**: maximal delay from any NLFSR or LFSR flip-flop through the **h** and **y** functions to any accumulator flip-flop
  - **Da**: maximal delay from any flip-flop in the authentication section of the cipher to any accumulator flip-flop
  - **Dhyn**: maximal delay from a flip-flop of the NLFSR or LFSR through the **h** and **y** functions to the first flip-flop of the NLFSR



Only during initialization phase

Our Approach

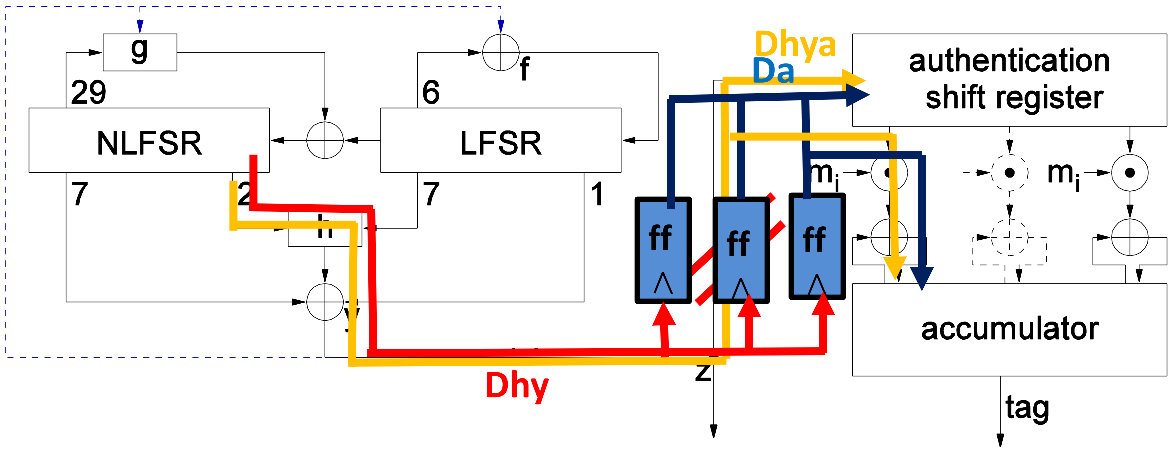
- We apply four techniques:
  - Isolation of the authentication section (improving **Dhya**)
  - Fibonacci-to-Galois transformation of shift registers (improving **Dn**)
  - Multi-frequency implementation (improving **Dhyn**)
  - Internal pipelining (improving **Dhy**)

Our Approach

- Isolation of the authentication section
- Fibonacci-to-Galois transformation of the feedback shift registers
- Multi-frequency implementation
- Internal pipelining

1. Isolation of the Auth. Section

- Problem:
  - **Dhya** increases as the degree of parallelization of Grain-128a grows
- Possible solution:
  - Isolation of the authentication section by inserting flip-flops in the authentication section on the outputs of the **h/y** function
- This solution:
  - Adds one cycle latency
  - Has no effect on security

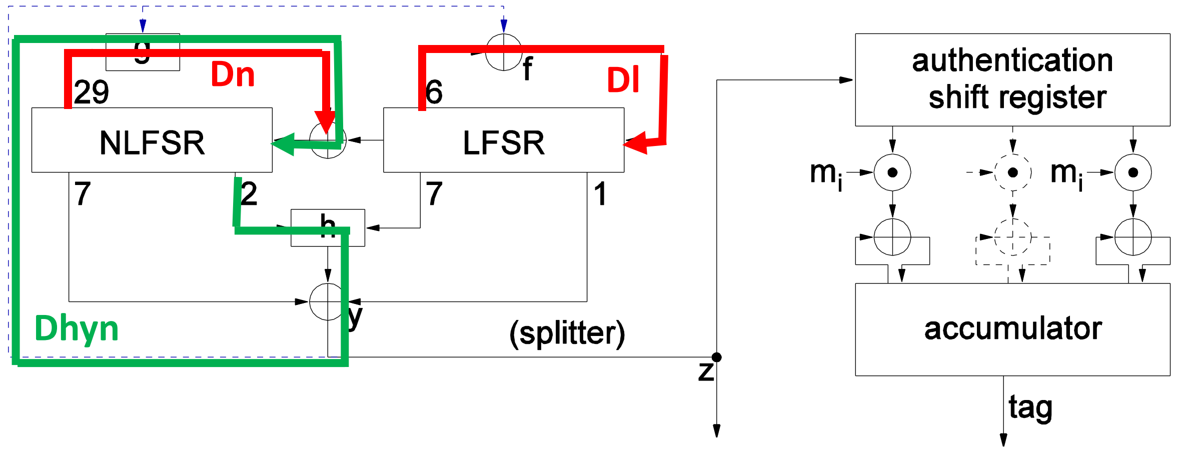


Our Approach

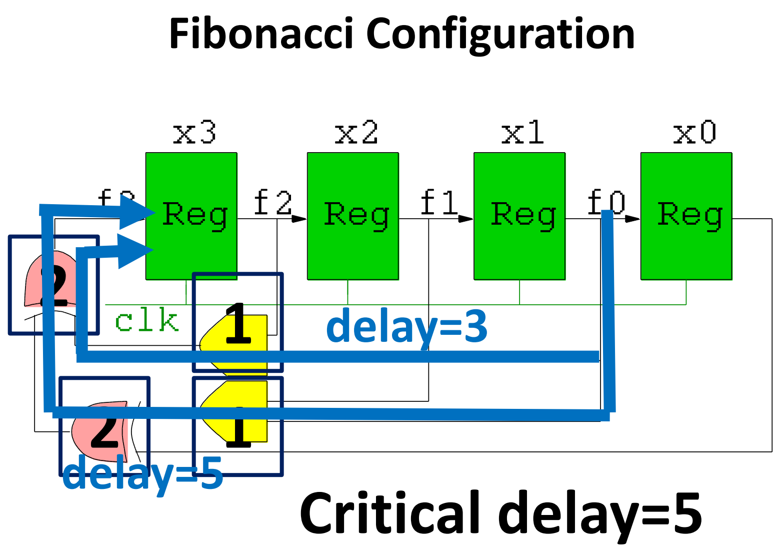
- Isolation of the authentication section
- Fibonacci-to-Galois transformation of the feedback shift registers
- Multi-frequency implementation
- Internal pipelining

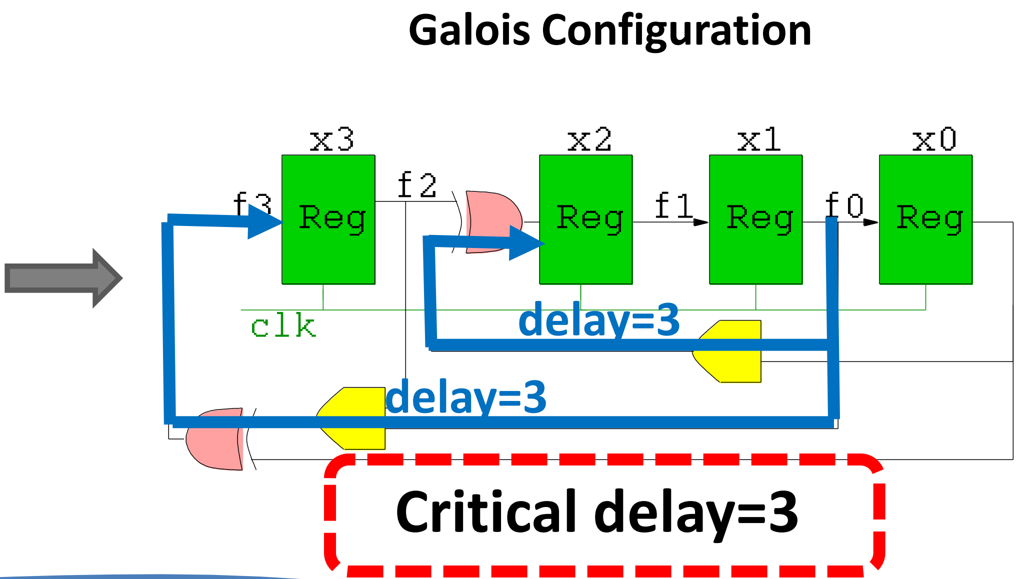
2. Fibonacci to Galois Transformation

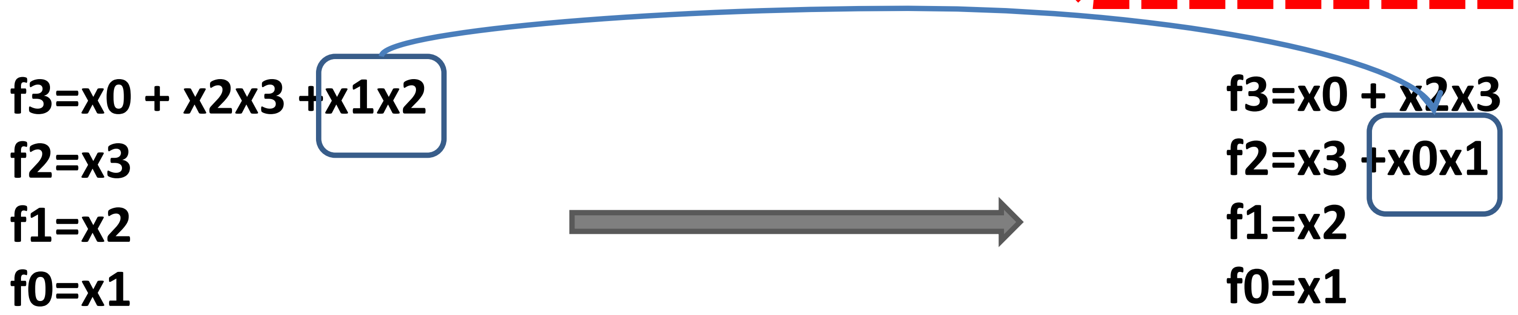
- Improves **Dhyn** and **Dn**
- Brings no area or power penalty



Fibonacci to Galois Transformation*







f3=x0 + x2x3 +x1x2
f2=x3
f1=x2
f0=x1

f3=x0 + x2x3
f2=x3 +x0x1
f1=x2
f0=x1

*A Transformation from the Fibonacci to the Galois NLFSRs", E. Dubrova,IEEE *Transactions on Information Theory*, 55:11, 2009, pp. 5263-5271

Example

The transformation from Fibonnacci to Galois is not unique

$$
\begin{aligned}
f_3 &= x_1x_2 + \boxed{x_1x_3} + x_0\\
f_2 &= x_3\\
f_1 &= x_2\\
f_0 &= x_1
\end{aligned}
\qquad
\begin{aligned}
f_3 &= \boxed{x_1x_2} + x_0\\
f_2 &= x_3 + \boxed{x_0x_2}\\
f_1 &= x_2\\
f_0 &= x_1
\end{aligned}
\qquad
\begin{aligned}
f_3 &= x_0\\
f_2 &= x_3 + \boxed{x_0x_1} + x_0x_2\\
f_1 &= x_2\\
f_0 &= x_1
\end{aligned}
$$

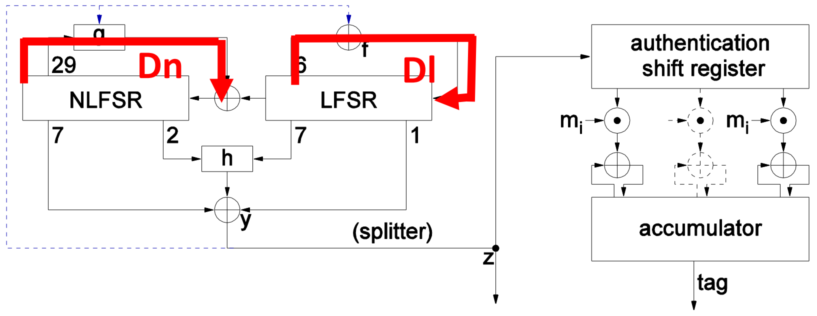
Fibonacci to Galois Transformation

- Explore the design space to find the best Galois NLFSR equivalent to a given Fibonacci NLFSR
- Optimal algorithm: synthesize every possible combination and find the best solution

  Computationally unfeasible - we need a heuristic approach*

"An Algorithm for Constructing a Fastest Galois NLFSR Generating a Given Sequence", J.-M.,Chabloz, S. Mansouri, E. Dubrova, in Sequences and Their Applications* , LNCS 6338, 2010, pp. 41-55

Improvement on Dl and Dn



- Highest improvement is achieved on **Dn** of Grain-128a x 1

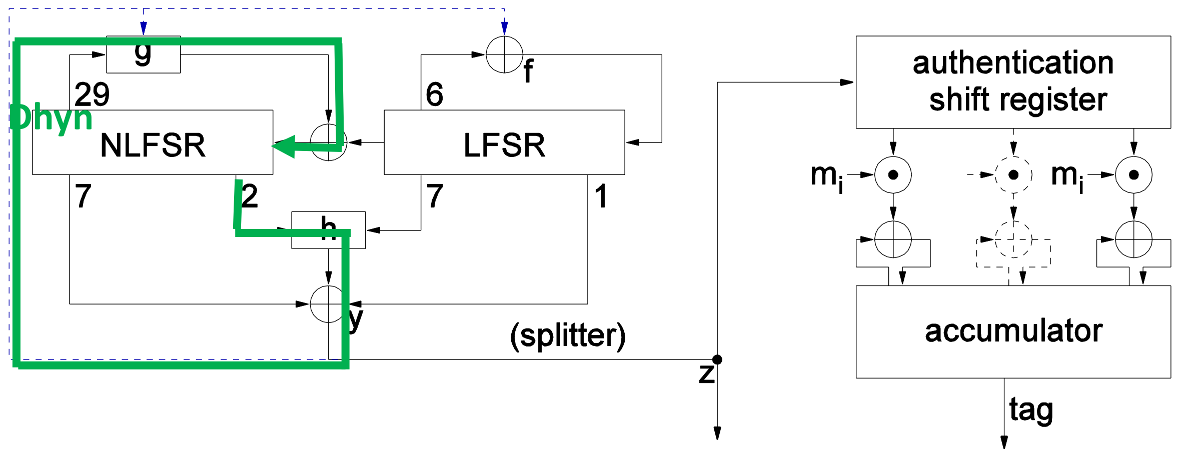
X1		X2		X4		X8		X16		X32	
LFSR	NLFSR	LFSR	NLFSR	LFSR	NLFSR	LFSR	NLFSR	LFSR	NLFSR	LFSR	NLFSR
60%	67%	53%	54%	51%	42%	26%	24%	18%	13%	0%	0%

Our Approach

- Isolation of the authentication section
- Fibonacci-to-Galois transformation of the feedback shift registers
- Multi-frequency implementation
- Internal pipelining

3. Multi-Frequency Implementation

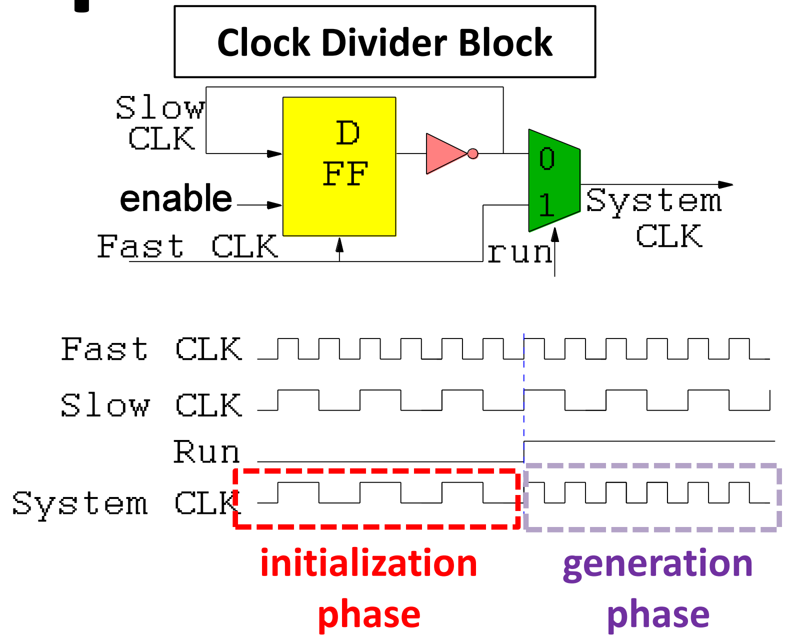
- The critical paths for all versions of grain-128a are given by **Dhyn**
- Although transforming the Grain's configuration improves the delays (**Dn** and **Dl**) up to 67 %, the clock frequency of the overall Grain cipher improves only about 10%
- **Dhyn** is active only during the keystream initialization phase
- To support efficiently both the initialization and key generation phases, we suggest a dual-frequency implementation of Grain-128a



Multi-Frequency Implementation

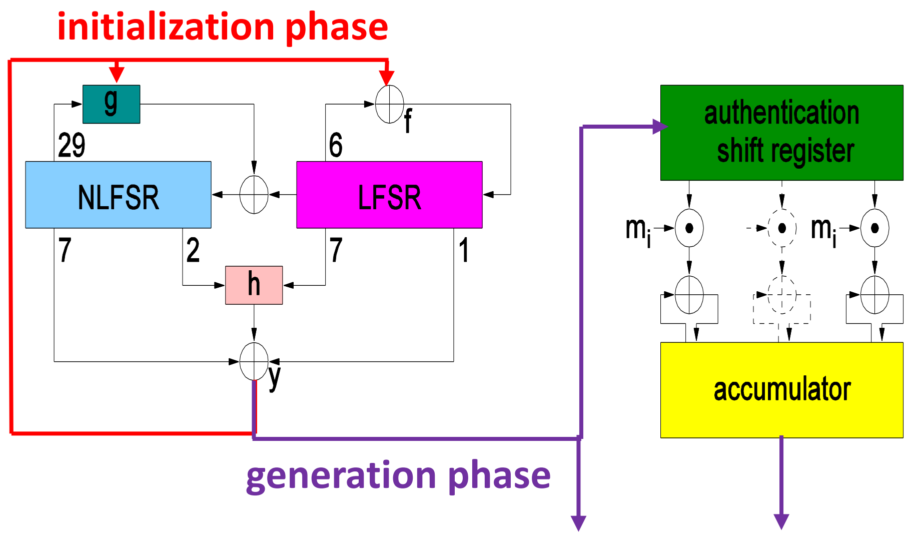
Grain128a phases:

- Keystream initialization phase (**Dhyn** path)
- Keystream generation phase (**Dn** path)



Multi-frequency based Grain128a:

- The cipher receives only one external clock signal (fast clk)
- Slow clock is made by clock divider from fast clock
- Slower clock used during the keystream initialization phase
- Faster clock used during the keystream generation phase

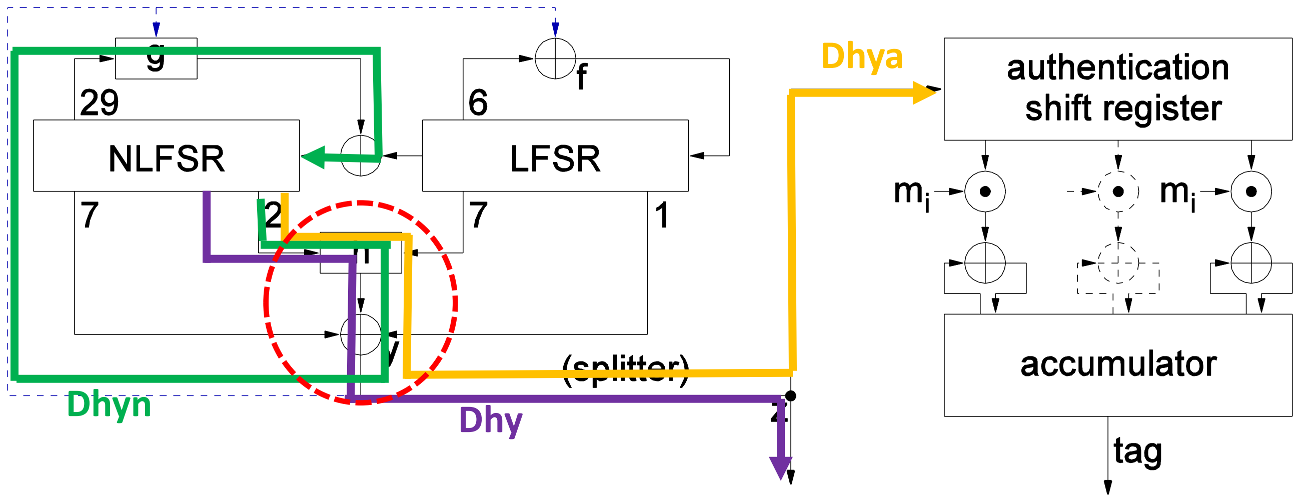


Our Approach

- Isolation of the authentication section
- Fibonacci-to-Galois transformation of the feedback shift registers
- Multi-frequency implementation
- Internal pipelining

4. Internal Pipelining

- The h/y function is pipelined during the key generation phase
- Advantage:
  - Cipher frequency is improved during key generation phase
- Disadvatage:
  - Pipeline flip-flops overhead
  - Increase the latency of a fixed number of cycles during the key generation phase



Throughput Improvement

Maximal improvement in frequency compared to the original design.

	Grain-128a X1	Grain-128a X2	Grain-128a X4	Grain-128a X8	Grain-128a X16	Grain-128a X32
Freq.	67%	80%	65%	32%	29%	33%
Area	0%	-5%	-7%	-1%	-5 %	-13%
Power	3	-7	-13	-2%	1%	4%

More information about different trade-offs can be found in the paper

Conclusion

- High throughput improvement
- Limited area/power impact
- Techniques compatible with the standard ASIC flow
- Some techniques can be applied to other ciphers

Thank You for your attention

Questions?

F2G: *http://web.it.kth.se/~dubrova/fib2gal.html*