# The GLUON family: a lightweight Hash function family based on FCSRs

T. P. Berger[1], J. D'Hayer[2], K. Marquet[2], M. Minier[2], G. Thomas[1]

[1]XLIM (UMR CNRS 7252), Limoges [2], CITI INRIA, INSA-Lyon

Africacrypt 2012, Ifrane, 10-12 July 2012



This work was partially supported by the French National Agency of Research: ANR-11-INS-011.

# Lightweight cryptography

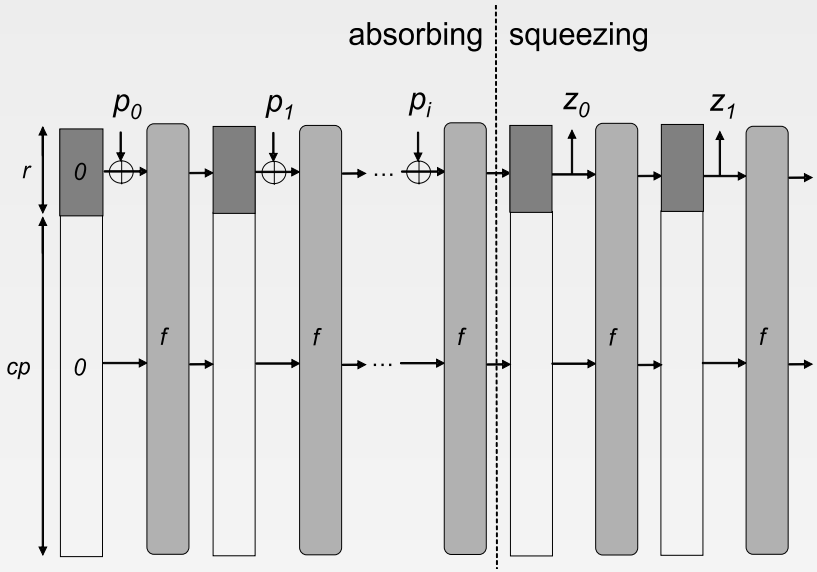
## Rules of Lightweight cryptography

- Hardware implementation (or limited software e.g. 8 bits processors)
- Limited resources. Upper Bound: 10 000 GE (gate equivalents), objective: around 3 000 GE
- Tradeoff between Security / Area / Power Consumption / Throughput
- $\rightarrow$ No use of lookup tables. No AES-like processes. Replaced by iterative computations
- $\rightarrow$ Variable security levels: 64 bits, 80 bits, 112 bits...

# The sponge model for hash functions

- Introduced in 2008 by G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche.

- Generic proved criteria for the design of hash functions under the assumption that the internal function is random.




1. **Initialization:** Message padded by appending a '1' bit and some 0.
2. **Absorbing phase:** XOR of the $r$-bit message blocks, interleaved with applications of the $f$ function.
3. **Squeezing phase:** Output of $r$ bits of the state, interleaved with applications of the $f$ function.

# Sponge security

- $n$ size of the output
- $r$ size of the input/output part of the automaton
- $c$ size of the internal part of the automaton

Under the assumptions $f$ is a random function and $2n \leq c$ the resistance of sponge construction is:

- Collision: $2^{n/2}$
- Preimage: $2^n$
- Second preimage: $2^n$

- $f$: function or permutation?
- No Matter!
- Deal: how to construct an efficient $f$ function which looks like random?
- More generally, for Lightweight cryptography,
  How to construct large Sbox what are iteratively computed?
  (more than 64 bits of input/output)

- $f$: function or permutation?
- No Matter!
- Deal: how to construct an efficient $f$ function which looks like random?
- More generally, for Lightweight cryptography,
  How to construct large Sbox what are iteratively computed?
  (more than 64 bits of input/output)

- $f$: function or permutation?
- No Matter!
- Deal: how to construct an efficient $f$ function which looks like random?
- More generally, for Lightweight cryptography,
  How to construct large Sbox what are iteratively computed?
  (more than 64 bits of input/output)

- $f$: function or permutation?
- No Matter!
- Deal: how to construct an efficient $f$ function which looks like random?
- More generally, for Lightweight cryptography,
  How to construct large Sbox what are iteratively computed?
  (more than 64 bits of input/output)

# From Stream Ciphers to functions

## A generic construction:

If one have an "ideal pseudo-random generator" with a secret key of size $k$

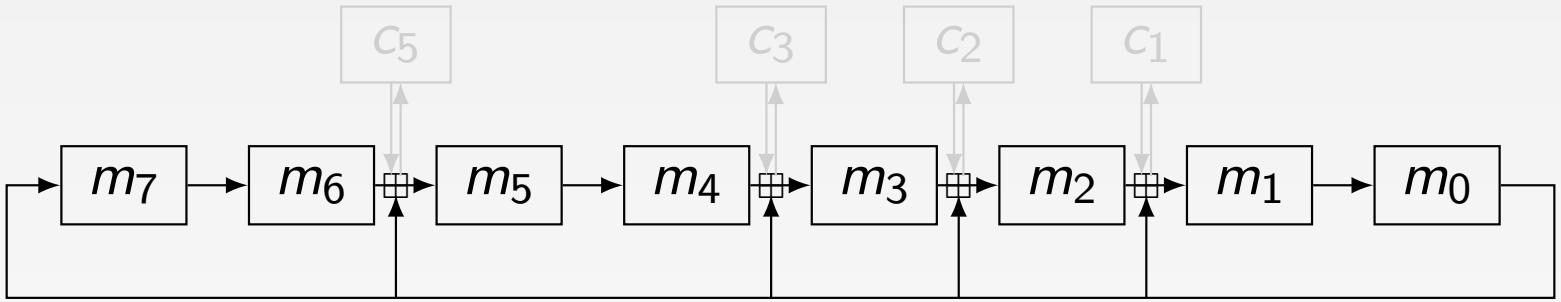
- Input: a message $m$ of $k$ bits
- Initialize the pseudo-random generator with $m$
- Output: the first $k$ bits of the pseudo-random sequence

$\Rightarrow f$ is not necessary a permutation, but a random function.

In the true life: non "ideal pseudorandom generators"
only hardware or software dedicated stream ciphers.
For example the lightweight hash function QUARK is essentially based on the design of the hardware stream cipher GRAIN.

# FCSR automata

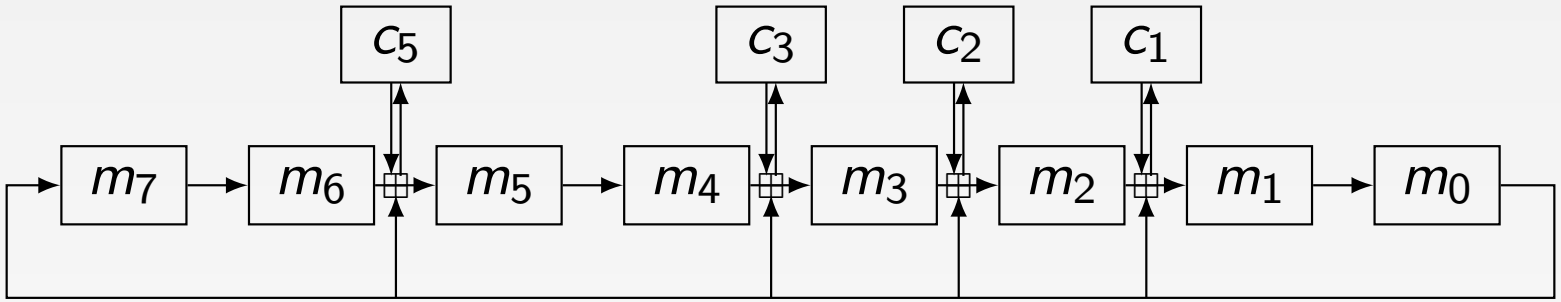
FCSR : Feedback with Carry Shift Register
close to LFSR




But with carries propagations $\Rightarrow$ 2-adic theory

# FCSR automata

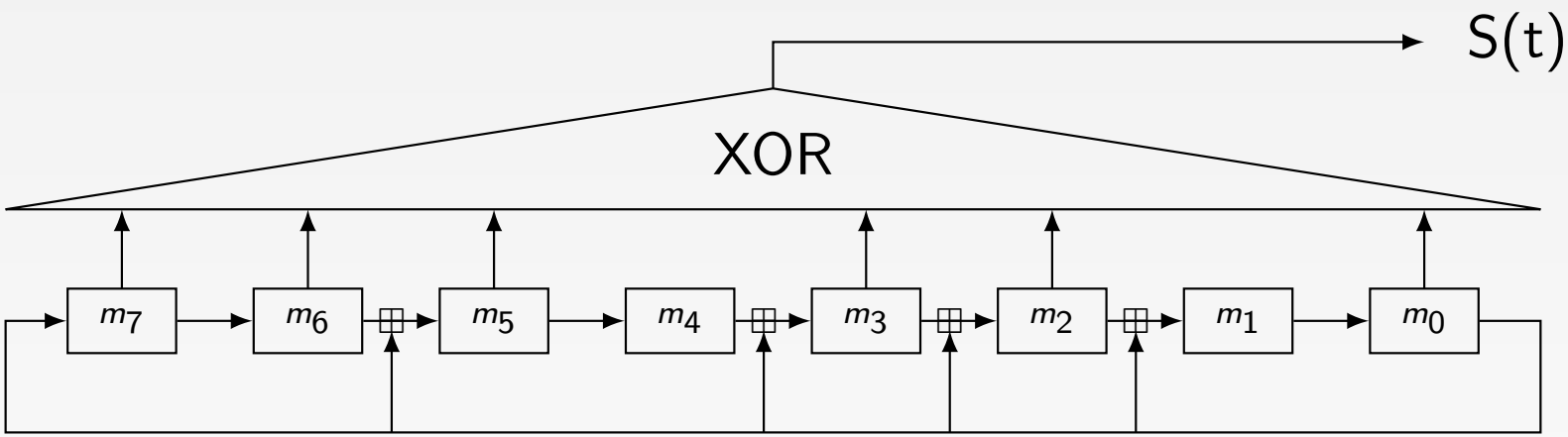
FCSR : Feedback with Carry Shift Register
close to LFSR




But with carries propagations $\Rightarrow$ 2-adic theory

# F-FCSR Stream Cipher family

F-FCSR steam ciphers are filtered FCSR automata with a linear filter.

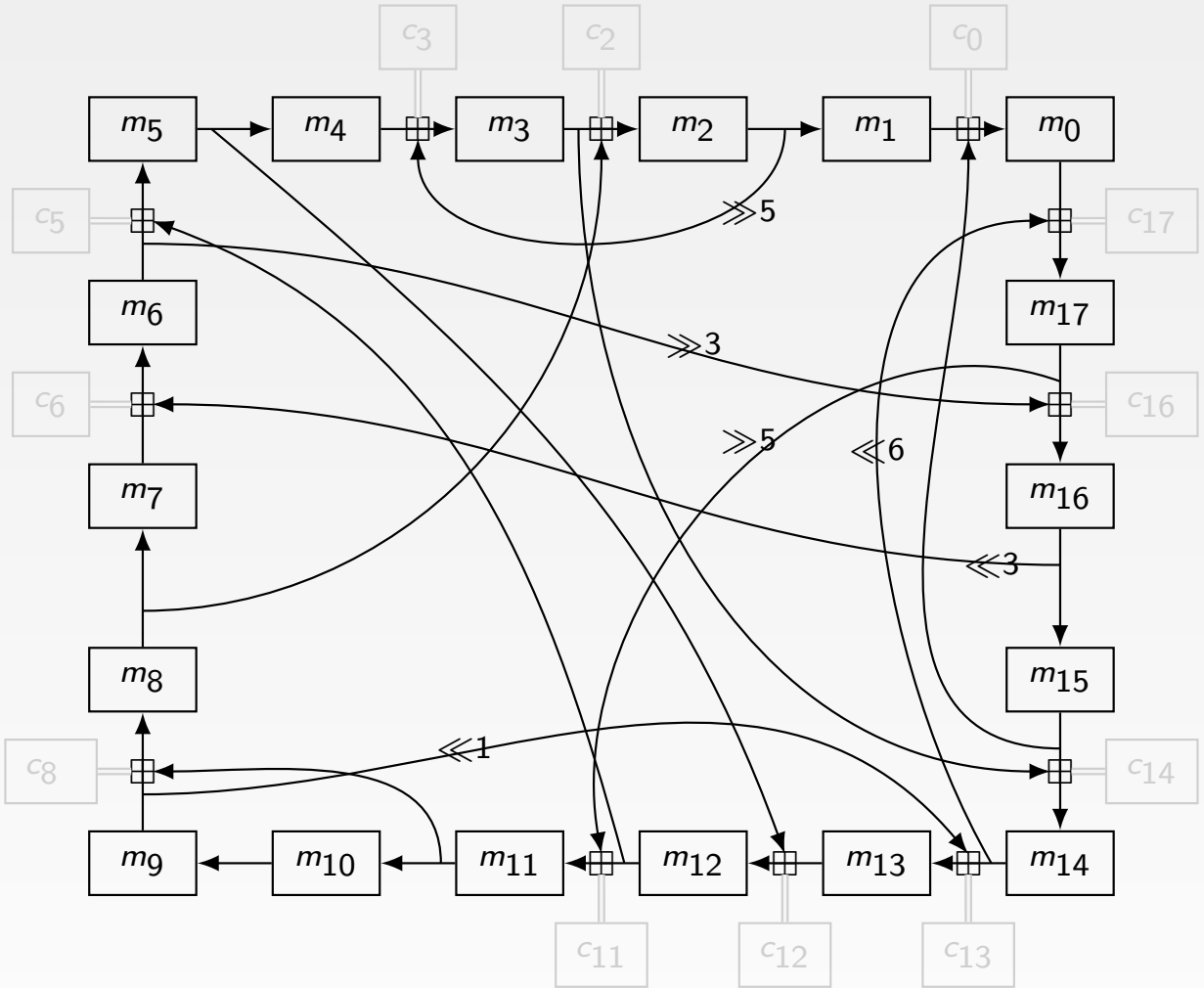


# Design of FCSR for cryptographic applications

A FCSR automaton must fulfill some requirements:

- Resistance to LFSRization (Cryptanalysis of Hell, Johannson) Avoid Galois and Fibonacci modes
- Efficient hardware implementations: ring structure with fan-out and span less or equal to 2
- Efficient software implementations on dedicated structures: use of blocks of size 8 bits (or 16 bits....)

# FCSR automaton for GLUON 64

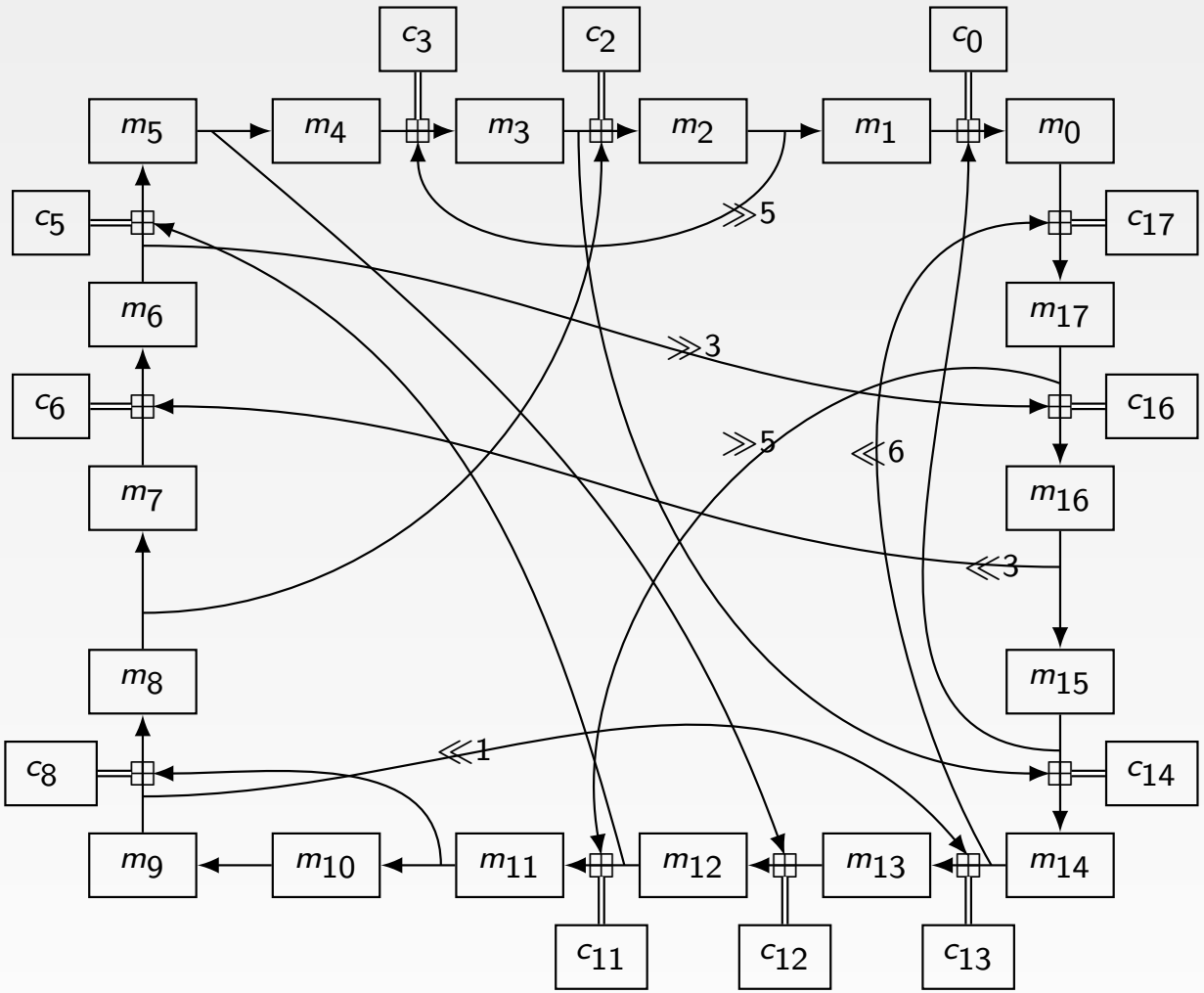



18 blocs of 8 bits

FCSR of 144 bits

+ 73 bits of carries

# FCSR automaton for GLUON 64




18 blocs of 8 bits

FCSR of 144 bits

+ 73 bits of carries

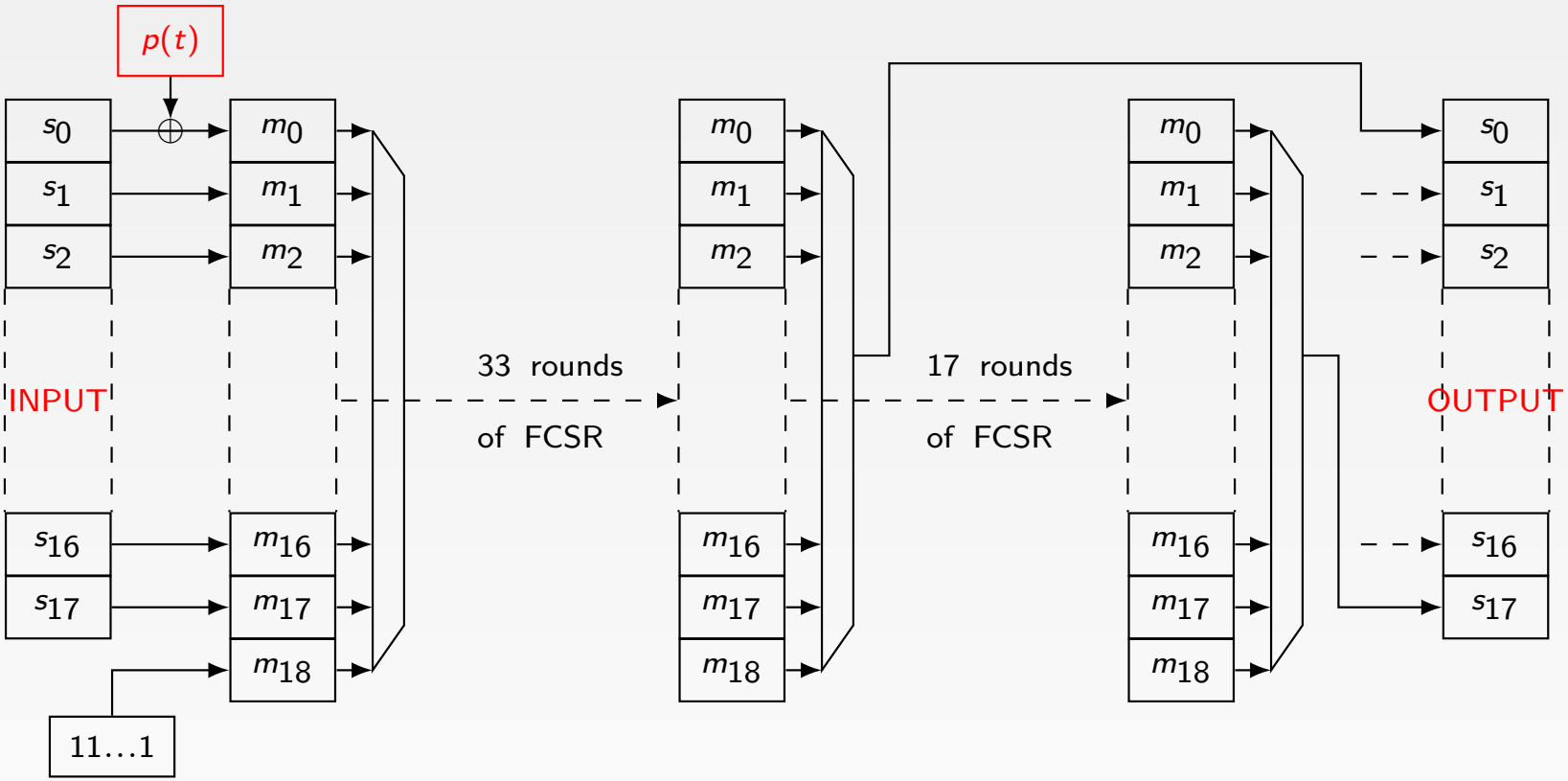
# Characteristics of the automaton

- Connection integer:
  $q = -27013336179990468777742546164977981767038829$
- $q$ is prime
- Order of 2 mod $q$: $1 - q \Rightarrow$ $m$-sequences
- Diameter (diffusion delay): 29
- Number of cells of the main register: 144
- Number of cells of the carry register: 73
- Optimized for hardware and software implementations

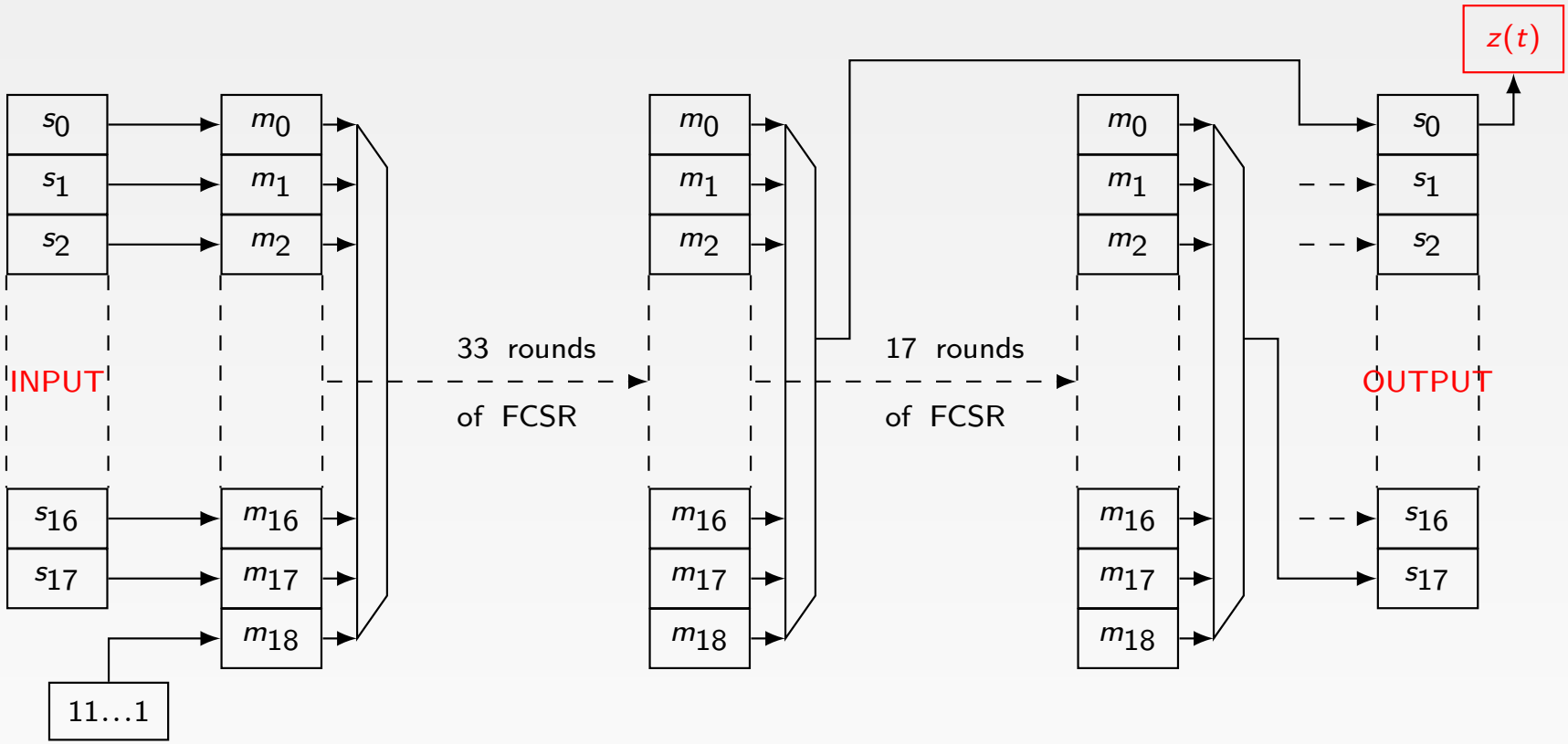
# Details of the $f$-function for GLUON 64

- Input: 17 blocs $S_0$, ... $S_{16}$ of 8 bits: 136 bits
- Initialization:
  Carry register: all-zero string
  Main register: $M_i := S_i$, for $i = 0$ to 16
  $M_{17} := (11111111)$
- State update: $d + 4 = 33$ iterations of the FCSR
- Output: 17 iterations of the FCSR, extraction of 8 bits at each iteration with a linear filter $\Rightarrow$ 17 blocs of 8 bits

# The sponge construction: one round of absorbing phase



# The sponge construction: one round of squeezing phase



# GLUON **variants**

| Version | Security | Output size | $r$ | $c$ | FCSR size |
|---|---|---|---|---|---|
| GLUON-64 | 64 | 128 | 8 | 128 | $17\times8$ |
| GLUON-80 | 80 | 160 | 16 | 160 | $12\times16$ |
| GLUON-112 | 112 | 224 | 32 | 224 | $9\times32$ |

- The underling FCSRs are chosen to be *m*-sequences and have a sufficient number of carries

# GLUON **variants**

| Version | Security | Output size | $r$ | $c$ | FCSR size |
|---|---|---|---|---|---|
| GLUON-64 | 64 | 128 | 8 | 128 | $17\times8$ |
| GLUON-80 | 80 | 160 | 16 | 160 | $12\times16$ |
| GLUON-112 | 112 | 224 | 32 | 224 | $9\times32$ |

- The underling FCSRs are chosen to be *m*-sequences and have a sufficient number of carries

# Security Level

- Collision attack and preimage attack
  Combinatorial explosion when trying to inverse the $f$ function
  $\Rightarrow$ (second) preimage: $2^{3wr/2} > 2^{3n/2}$, collision: $2^{3wr/4} > 2^{3n/2}$

- Cube attacks and Cube testers
  [BM 05]: No particular structure for the ANF of a FCSR
  Example: Galois FCSR of length 16 bits, after 7 clocks,
  nb monomials of degree $\geq 10 = 125420$

- Linear and differential attacks
  - Linear attacks discarded by design of word ring FCSRs
  - Differential properties are largely spread by the sufficient number of clocks

# Security Level

- Collision attack and preimage attack
  Combinatorial explosion when trying to inverse the $f$ function
  $\Rightarrow$ (second) preimage: $2^{3wr/2} > 2^{3n/2}$, collision:
  $2^{3wr/4} > 2^{3n/2}$

- Cube attacks and Cube testers
  [BM 05]: No particular structure for the ANF of a FCSR
  Example: Galois FCSR of length 16 bits, after 7 clocks,
  nb monomials of degree $\geq 10 = 125420$

- Linear and differential attacks
  - Linear attacks discarded by design of word ring FCSRs
  - Differential properties are largely spread by the sufficient number of clocks

# Security Level

- Collision attack and preimage attack
  Combinatorial explosion when trying to inverse the $f$ function
  $\Rightarrow$ (second) preimage: $2^{3wr/2} > 2^{3n/2}$, collision:
  $2^{3wr/4} > 2^{3n/2}$

- Cube attacks and Cube testers
  [BM 05]: No particular structure for the ANF of a FCSR
  Example: Galois FCSR of length 16 bits, after 7 clocks,
  nb monomials of degree $\geq 10 = 125420$

- Linear and differential attacks
  - Linear attacks discarded by design of word ring FCSRs
  - Differential properties are largely spread by the sufficient number of clocks

# Performances (1/2)

- **Hardware performances**

| Hash function | Security Pre. | Security Coll. | Block [bits] | Area [GE] | Lat. [cycles] | Thr. kbps |
|---|---|---|---|---|---|---|
| GLUON-64 | 128 | 64 | 8 | 2071 | 66 | 12.12 |
| GLUON-80 | 160 | 80 | 16 | 2799.3 | 50 | 32 |
| GLUON-112 | 224 | 112 | 32 | 4724 | 55 | 58.18 |
| U-QUARK×8 | 128 | 64 | 8 | 2392 | 68 | 11.76 |
| D-QUARK×8 | 160 | 80 | 16 | 2819 | 88 | 18.18 |
| S-QUARK×16 | 224 | 112 | 32 | 4640 | 64 | 50.00 |
| PHOTON-80 | 160 | 80 | 16 | 1168 | 132 | 12.15 |

# Performances (2/2)

- **Software performances** in cycles per byte

| | |
|---|---|
| GLUON-64 | 17319 |
| U-QUARK | 43373 |

| | |
|---|---|
| GLUON-80 | 8523 |
| D-QUARK | 35103 |
| PHOTON-80 | 1243 |

| | |
|---|---|
| GLUON-112 | 1951 |
| S-QUARK | 25142 |

# Conclusion

## ...New lightweight design

- Based on a well known primitive: word ring FCSR
- Well known properties
- Flexible, depending on applications (hardware, software 8 bits...)

## New lightweight design...

- Please, try to attack!

# Conclusion

## ...New lightweight design

- Based on a well known primitive: word ring FCSR
- Well known properties
- Flexible, depending on applications (hardware, software 8 bits...)

## New lightweight design...

- Please, try to attack!