Analysis of Message Injection in Stream Cipher-based Hash Functions

Yuto Nakano[1], Carlos Cid[2],
Kazuhide Fukushima[1], and Shinsaku Kiyomoto[1]

[1] KDDI R&D Laboratories Inc.
[2] Royal Holloway, University of London

SCH : stream cipher-based hash function

- Use stream cipher as core component
- Can be used not only as a hash function but also as a stream cipher
- Suit for resource-constrained devices
- Arbitrary length of hash value

- Message injection function is attached
- Three phases
  - Message injection
  - Blank rounds
  - Hash generation

Motivation

Not much research has been done on SCHs

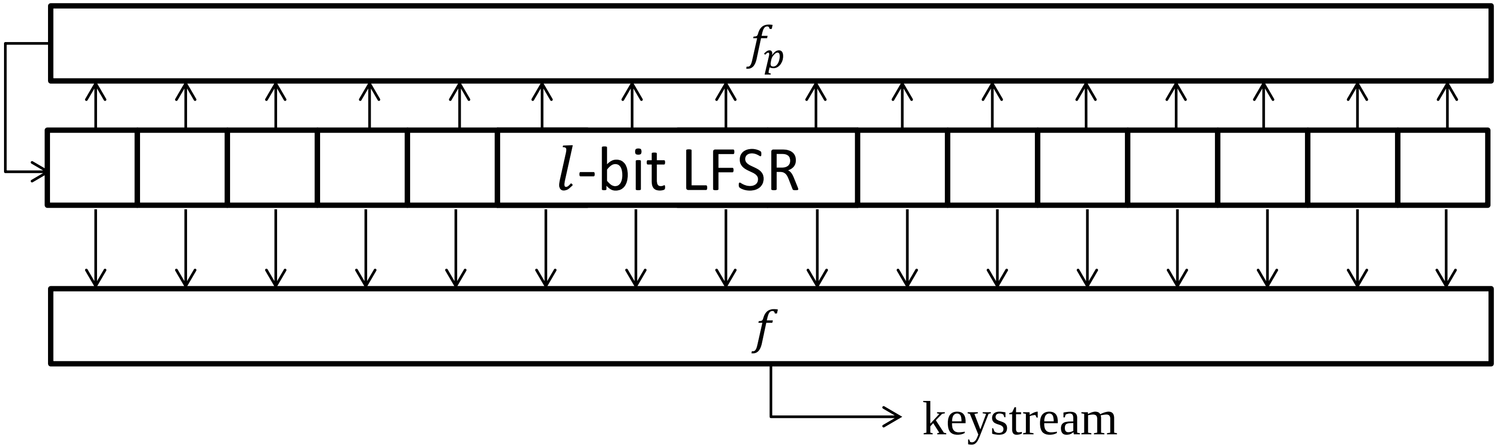
Some SHA-3 candidates are stream cipher-based, but not secure

In this talk,

- Definition of message injection functions
  - Inject into feedback
  - Inject into the internal state
- Security analysis of message injection function with
  - One LFSR and filter function
  - Two LFSRs and filter function
- Comparison to real algorithm (Abacus, Boole, MCSSHA-3)

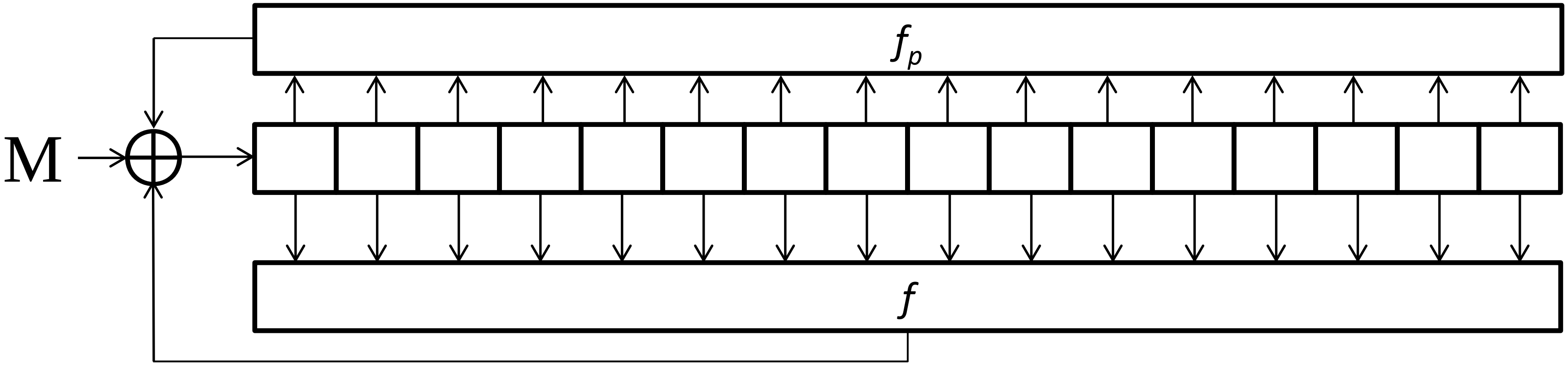
Definition of Stream cipher

- Simple stream cipher based on an l-bit LFSR and a filter function
- Feedback polynomial f_p is primitive
- Filter function takes n-bit input ($n \leq l$) and outputs 1-bit keystream



Inject into feedback

The message is XORed with keystream and feedback polynomial



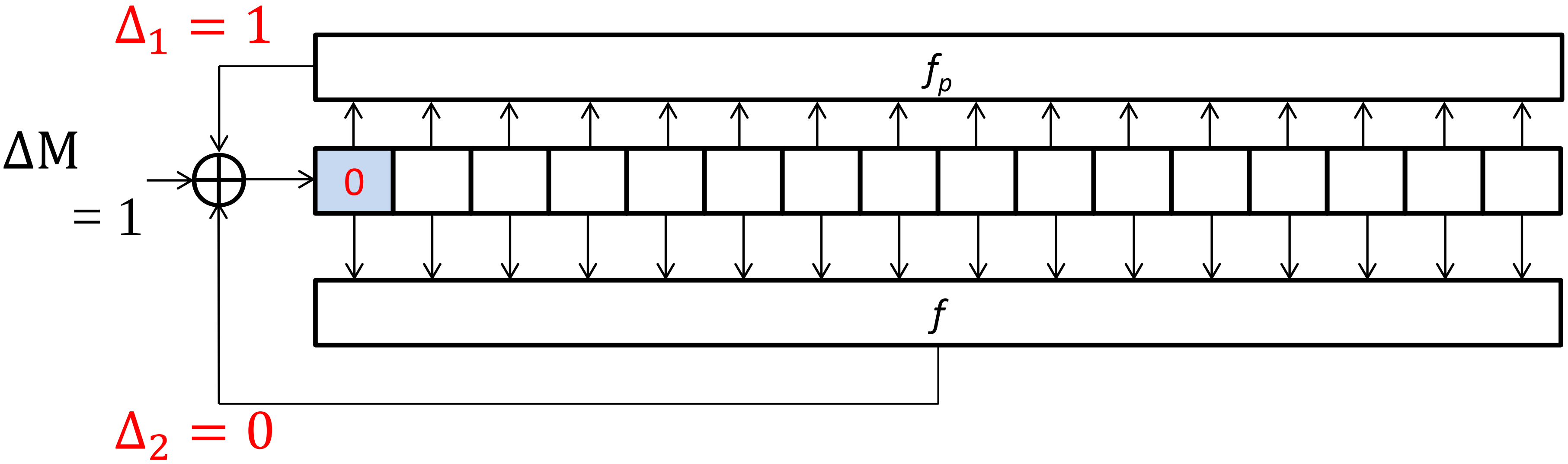
State S_t is updated into S_{t+1} as

$$s_{t+1,i} = \left[\begin{array}{l} \qquad\qquad\qquad\qquad s_{t,i+1} \\ f_p(s_{t,1}, \dots, s_{t,l}) \oplus \left(f(d_1 s_{t,1}, \dots d_l s_{t,l}) \oplus M\right) \end{array} \right.$$

The most natural way to inject message

SHA-family and MD-family apply this type

Security analysis



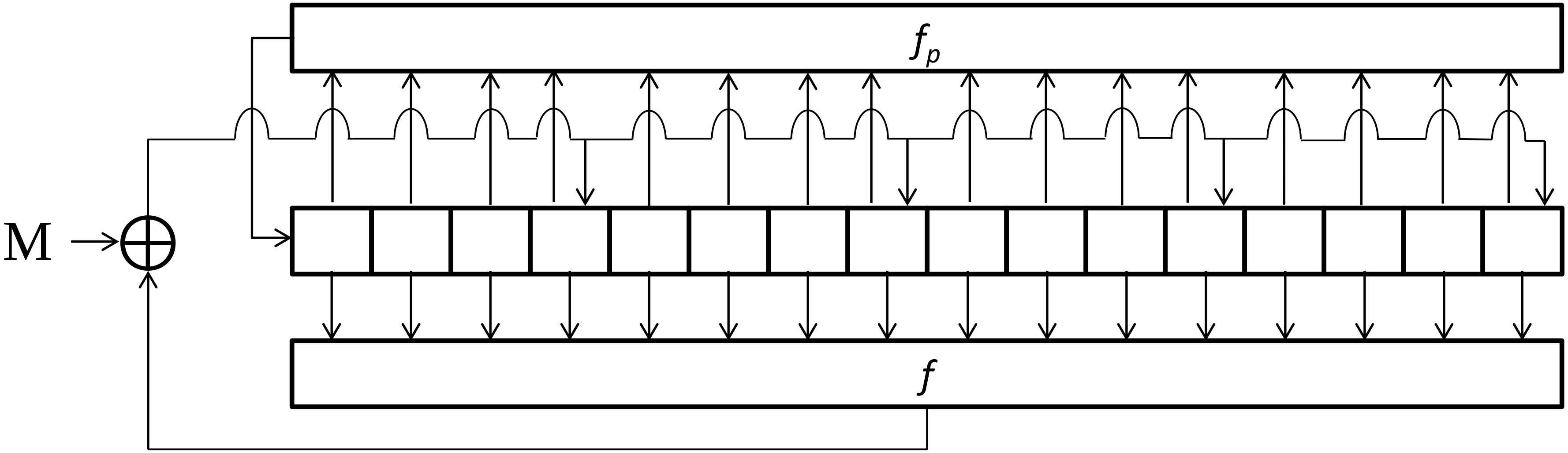
- Blue-colored register x can easily controlled by the message

$$x = \Delta_1 \oplus \Delta_2 \oplus M$$

- Difference on the LFSR is forced out and collision is easily generated
- Message expansion is required

Inject into internal state 1

- Message dependent data is XORed with r registers

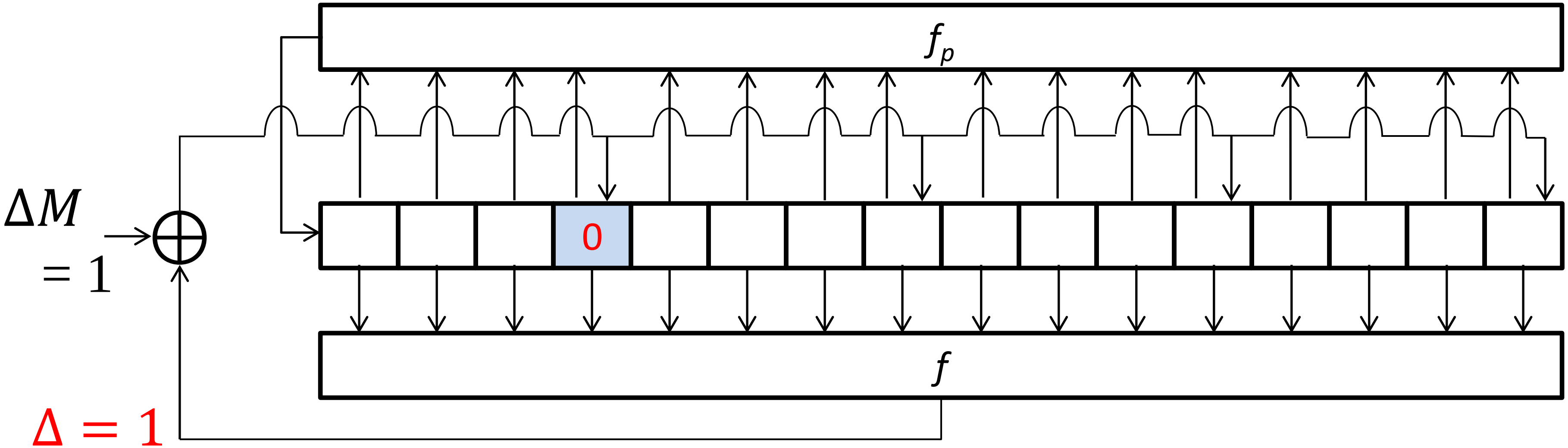


$$s_{t+1,i} = \left[\begin{array}{c} s_{t,i+1} \oplus \sigma_i\,(z_t \oplus M) \\ f_p\left(s_{t,1}, \dots, s_{t,l}\right) \end{array}\right. ,$$

where σ_i is a selector that selects which register to be updated

- Quick message diffusion over the state

Security analysis

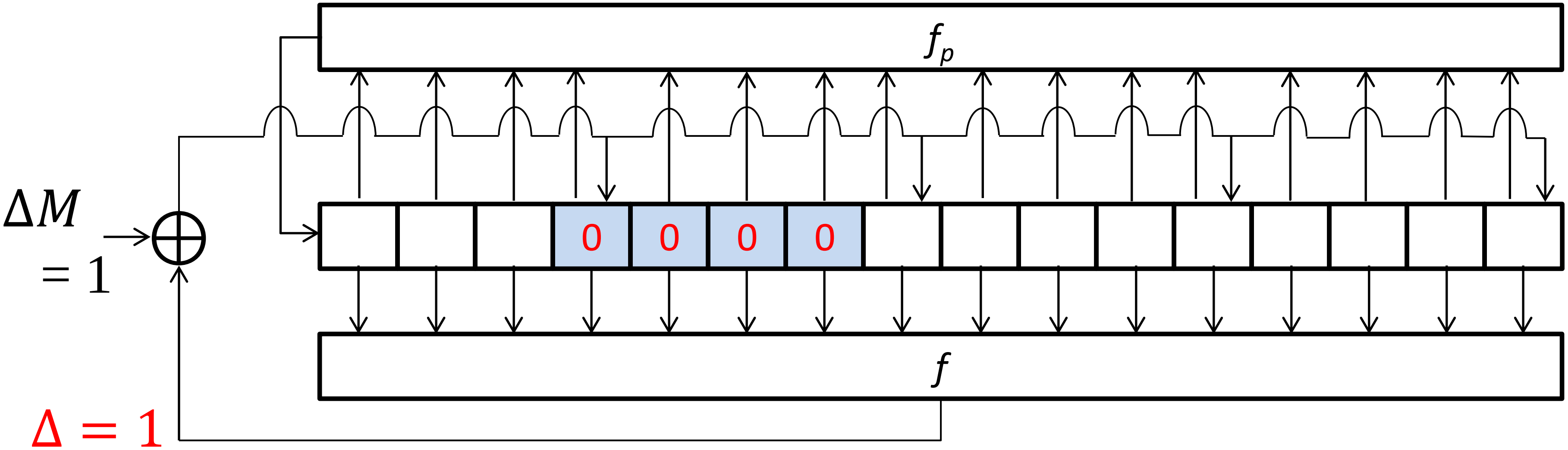


The adversary can control blue-colored l/r bits

Use the birthday attack against remaining $l(1-1/r)$ bits, the probability is given by

$$\Pr[\text{coll}] = 2^{-\frac{l(1-1/r)}{2}}$$

Security analysis



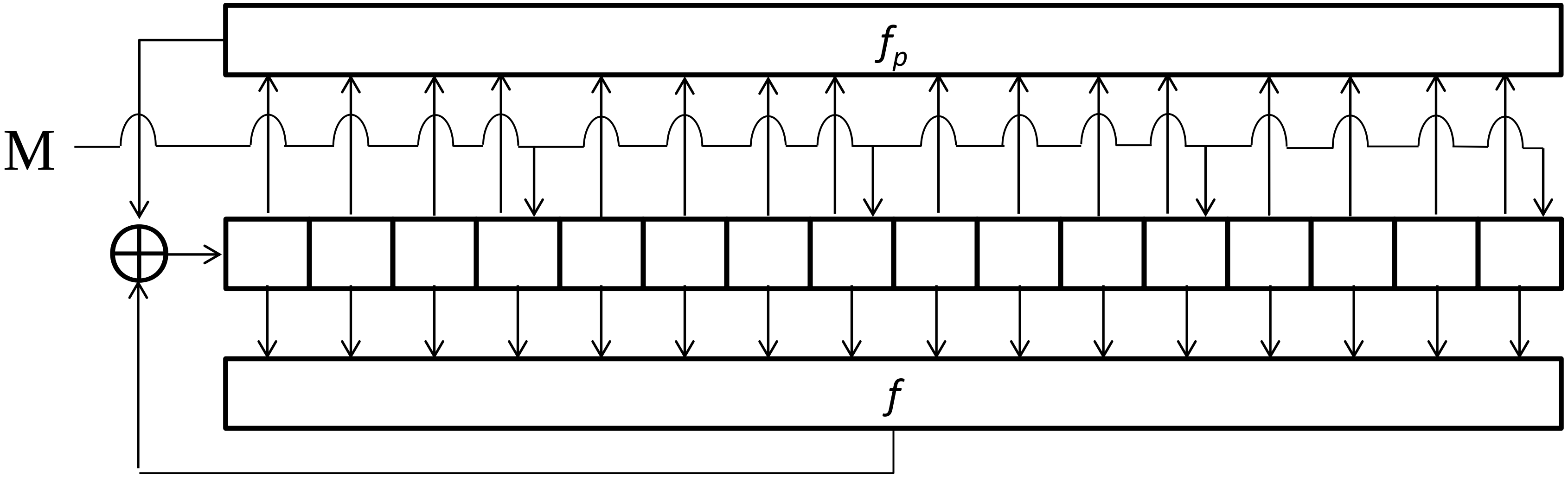
The adversary can control blue-colored l/r bits

Use the birthday attack against remaining $l(1-1/r)$ bits, the probability is given by

$$\Pr[\text{coll}] = 2^{-\frac{l(1-1/r)}{2}}$$

Inject into internal state 2

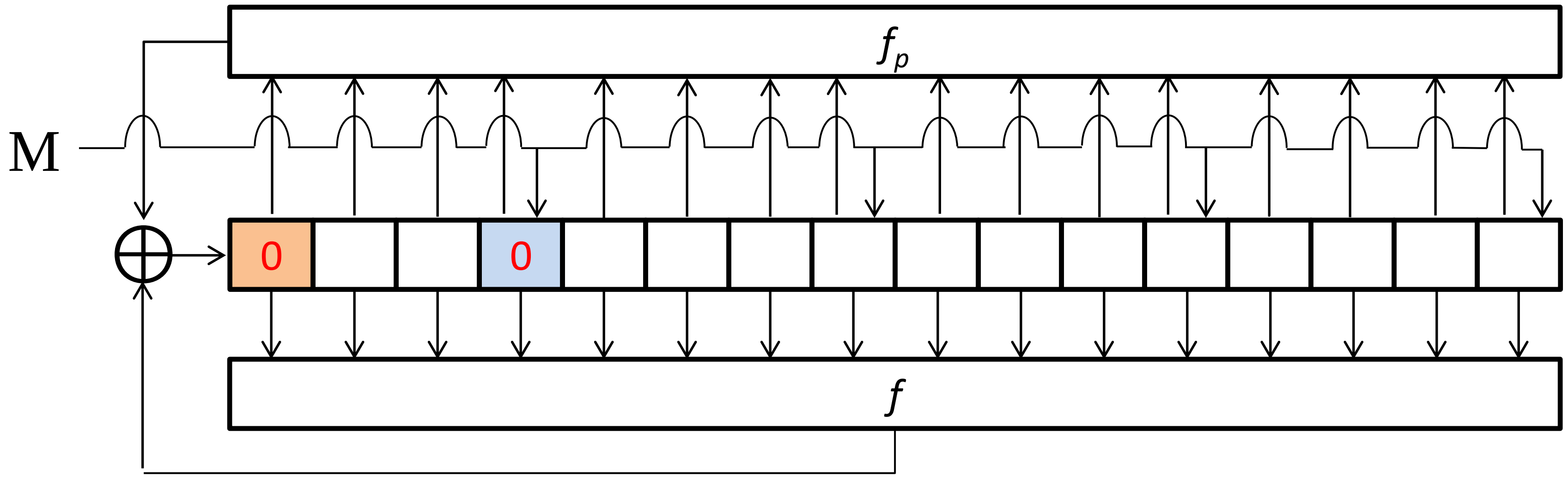
Message is XORed with r registers



$$s_{t+1,i} = \begin{cases} s_{t,i+1} \oplus \sigma_i \cdot M \\ f_p(s_{t,1}, \dots, s_{t,l}) \oplus z_t \end{cases}$$

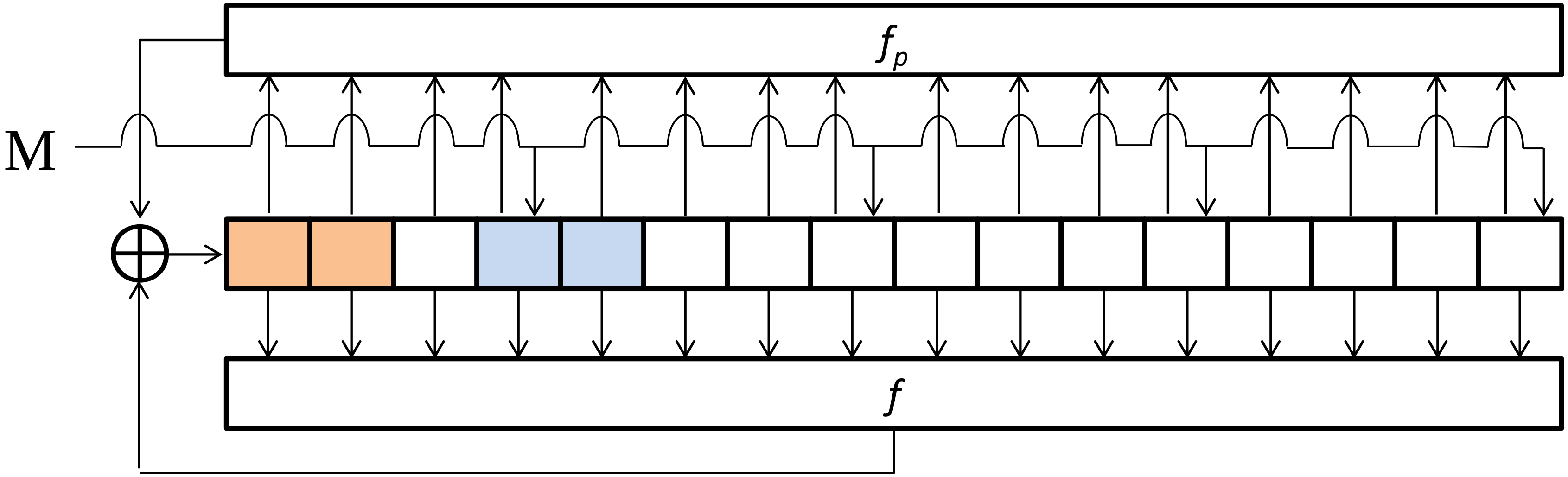
where σ_i is a selector that selects which register to be updated

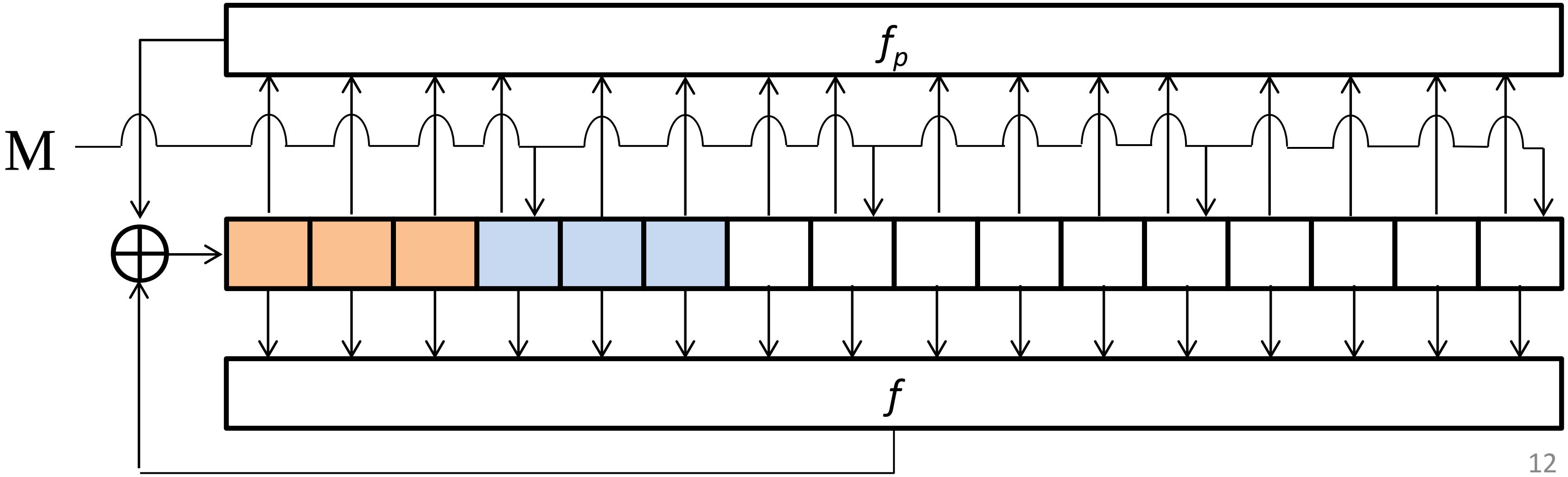
Collision attack



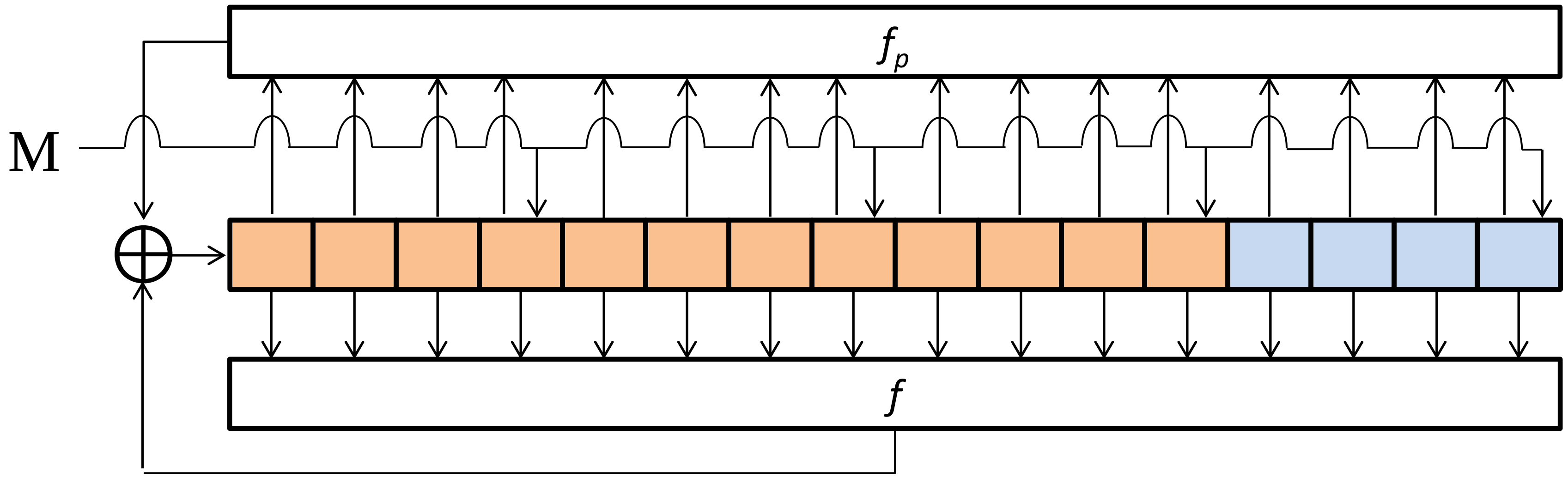
- Blue-colored registers can be controlled
- Difference on orange-colored will vanish when
  - – feedback & keystream have difference
  - – Both do not have difference

Collision attack(cont'd)





Collision attack(cont'd)



- The adversary can control r/l bits of the state
- Collision attack will be successful when difference on $l(1-1/r)$ bits vanishes

Security analysis

- Filter function outputs difference with probability p
- When the internal state has difference, feedback has also difference with $\mathbf{1/2}$

The filter function must output difference $\frac{l(1-1/r)}{2}$ times

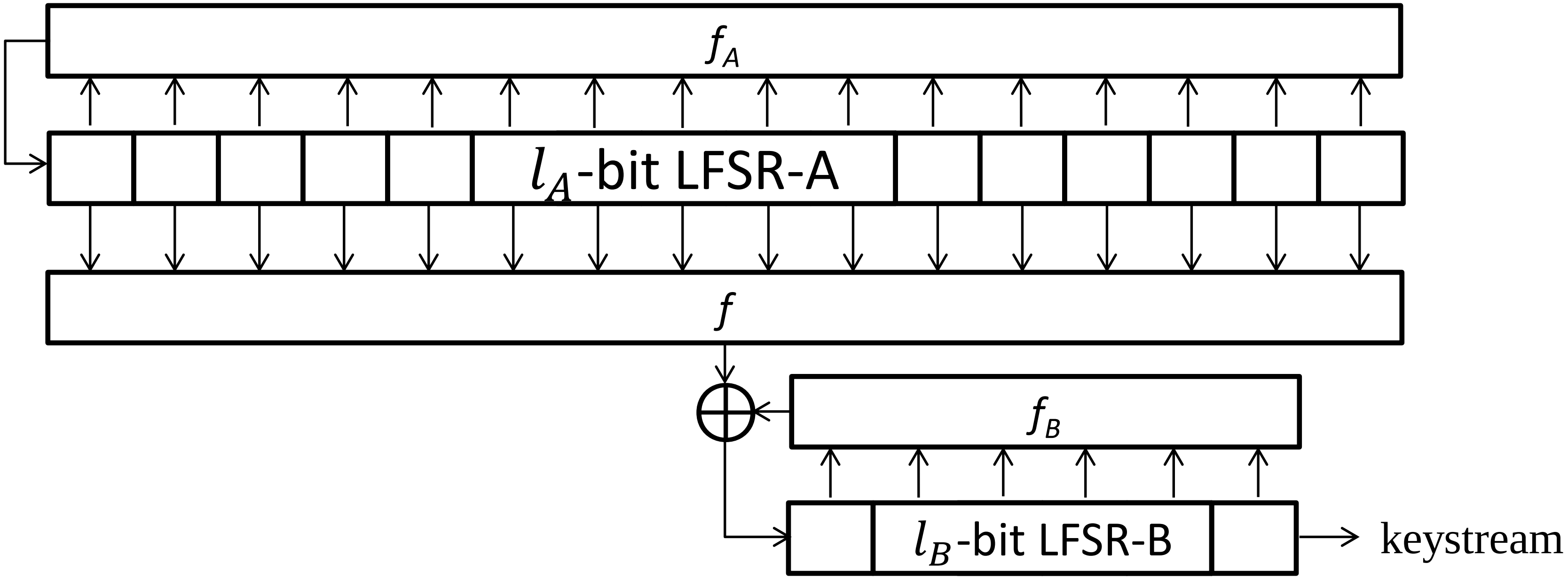
$$\Pr[\mathrm{coll}] = [p(1-p)]^{\frac{l(1-1/r)}{2}}$$

- When the filter function is balanced, then it propagates difference with $p = \mathbf{1/2}$

$$\Pr[\mathrm{coll}] = 2^{-l(1-1/r)}$$

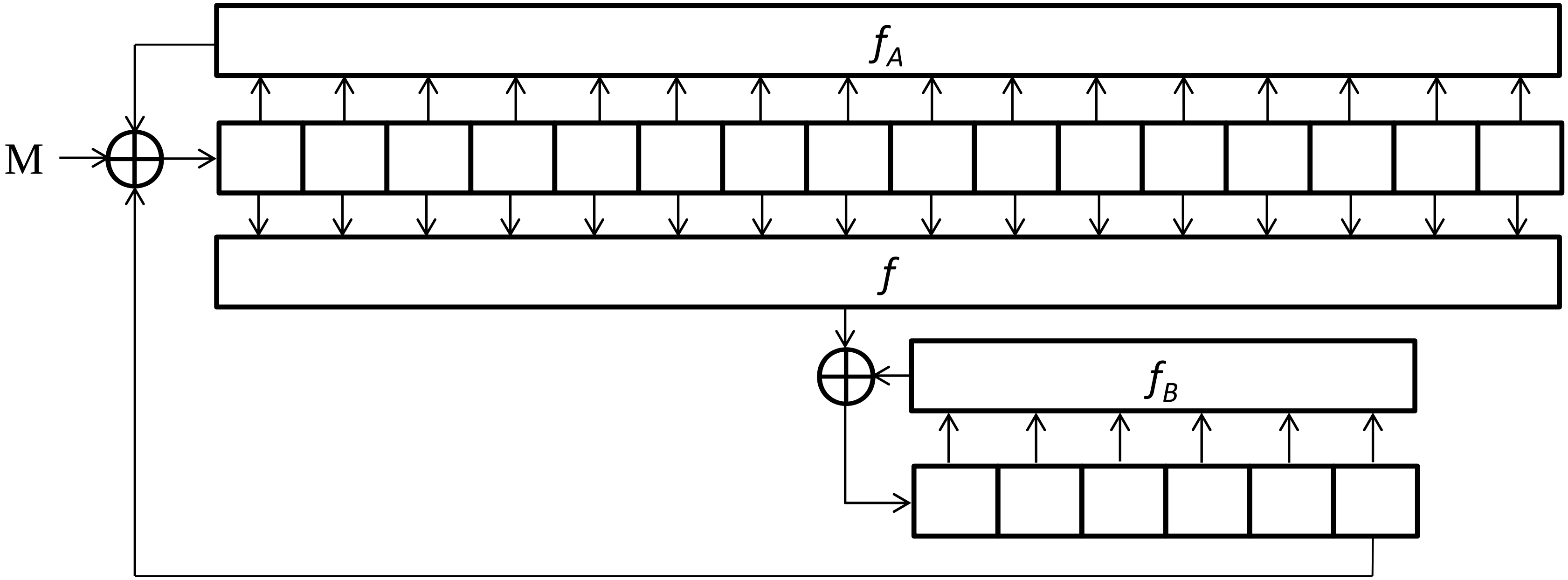
Birthday attack is more efficient: $\Pr[\mathrm{coll}] = 2^{-\frac{l(1-1/r)}{2}}$

Extension to Two LFSRs



- l_A-bit LFSR-A and l_B-bit LFSR-B ($l_A > l_B$)
- f_A and f_B are primitive
- LFSR-A is used to determine the output of filter function
- Output of filter function is XORed with feedback of LFSR-B

Inject into feedback of LFSR-A

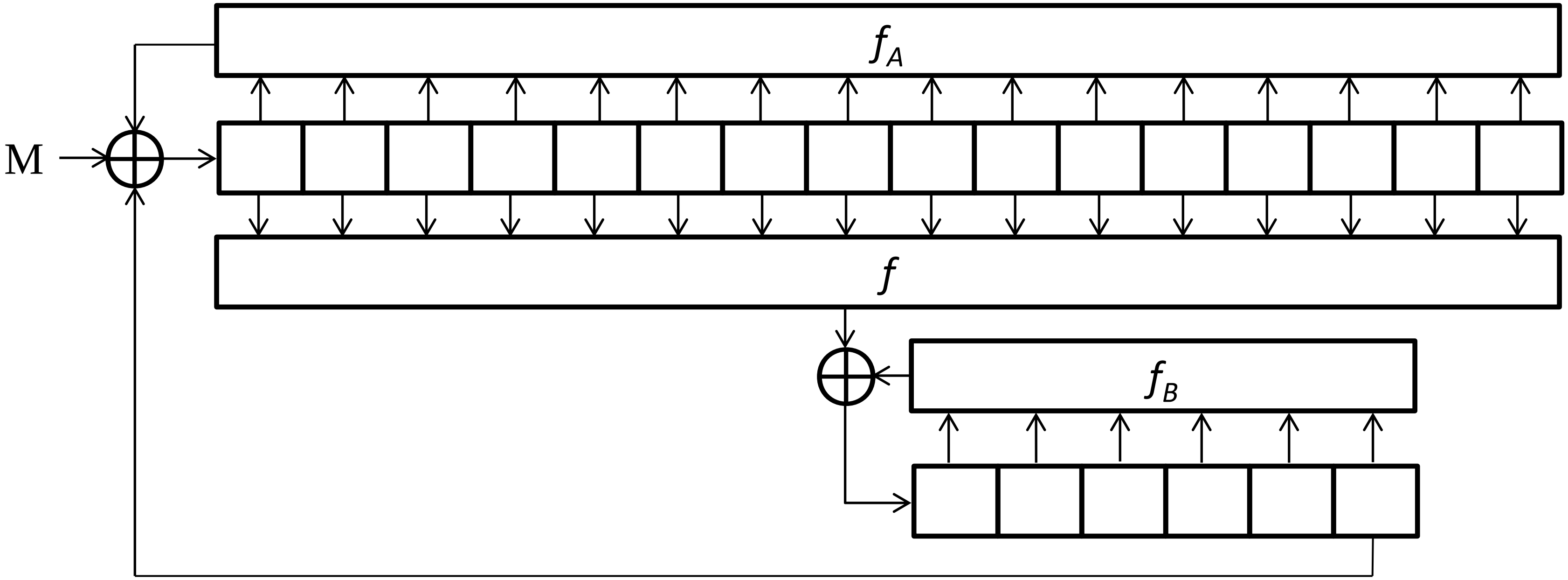


Message is XORed with feedback

$$s_{t+1,i} = \begin{cases} s_{t,i+1} \\ f_A\left(s_{t,1}, \dots, s_{t,l_A}\right) \oplus M \end{cases}$$

$$u_{t+1,i} = \begin{cases} u_{t,i+1} \\ f_B\left(u_{t,1}, \dots, u_{t,l_B}\right) \oplus f(S') \end{cases}$$

Security analysis

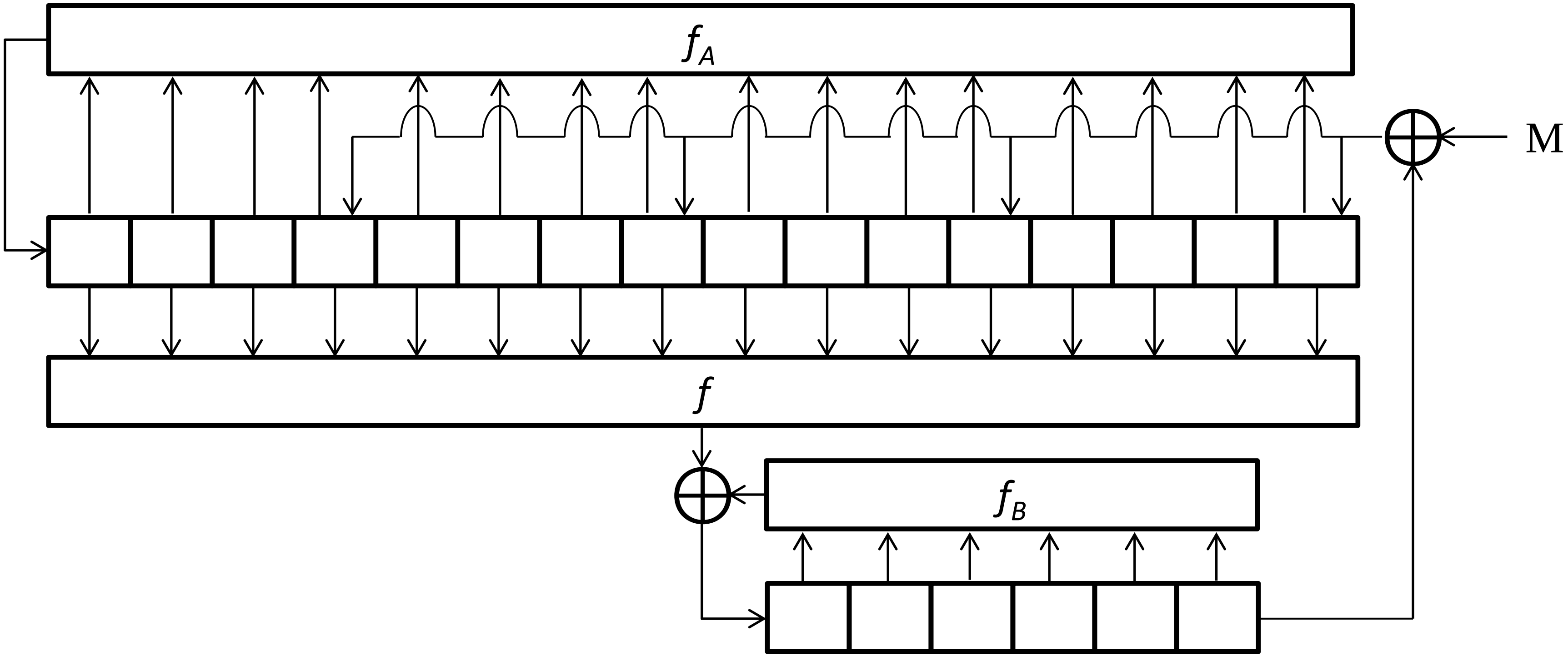


- Difference on LFSR-A can be canceled out
- Collision probability depends on that of LFSR-B

$$\Pr[\text{coll}] = \max\left(2^{-l_B/2}, \Pr[\text{diff. on B canceled}]\right)$$

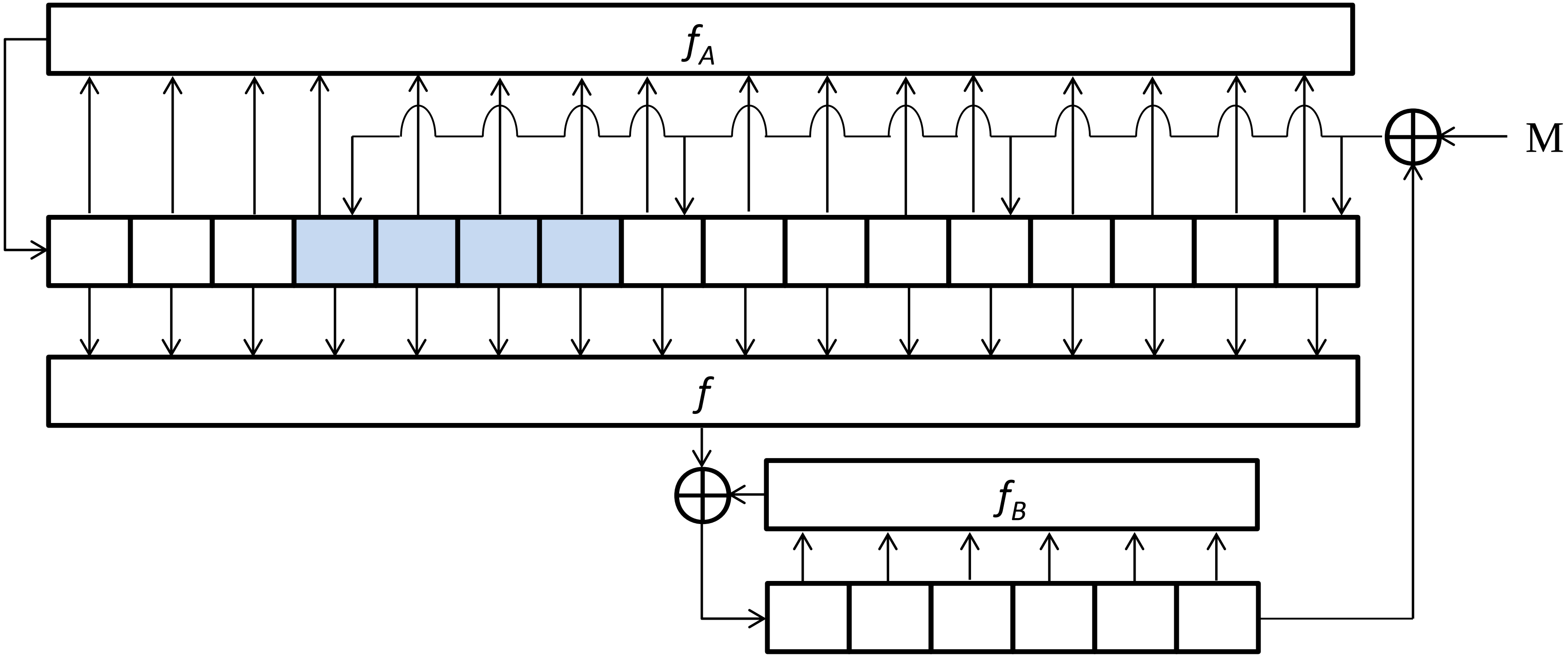
$$= 2^{-l_B/2}$$

Inject into part of the state of LFSR-A



- Message dependent data is XORed with *r* registers of LFSR-A
- Message spread over the state quickly

Security analysis



- Blue-colored l_A/r-bit registers can be controlled
- Birthday attack on $l_A(1-1/r)+l_B$ bits

$$\Pr[\mathrm{coll}] = 2^{-\frac{l_A(1-1/r)+l_B}{2}}$$

Summary

MIF	Collision probability	# of operation/cycle
Single LFSR		
Inject into feedback	1	1 XOR
Inject into the int. state	$2^{-\frac{l(1-1/r)}{2}}$	r XORs
Two LFSRs		
Inject into feedback of LFSR-A	$2^{-l_B/2}$	1 XOR
Inject into feedback of both LFSRs	$2^{-l_B/2}$	2 XORs
Inject into int. state of LFSR-A	$2^{-\frac{l_A(1-1/r)+l_B}{2}}$	r XORs
Inject into int. state of both LFSRs	$2^{-\frac{l_A(1-1/r)+l_B}{2}}$	$(r+q)$ XORs

Comparison to real algorithms

- Apply our estimation to real algorithms
  - Abacus (inject into feedback)
  - Boole (inject into the internal state)
  - MCSSHA-3 (inject into feedback)
- Assume these algorithms are bit-oriented
- Substitute register size to the estimated probability

Comparison to real algorithms

	Our estimation	**Real attack**
Abacus	2^{-172}	2^{-172}
Boole	2^{-176}	2^{-33}
MCSSHA-3	2^{-96}	2^{-96}

Our estimation can be applied to existing algorithms

Gap of Boole is due to

- Different message-dependent data is used update registers
- Boolean functions of Boole have a vulnerability

Conclusion

- Definition of message injection functions
  - Inject into feedback
  - Inject into the internal state
- Security analysis of message injection function with
  - One LFSR and filter function
  - Two LFSRs and filter function
  - Required length of LFSRs
  - Number of message-injecting registers
- Our evaluation can be applied to existing algorithm