Differential Fault Analysis of Trivium

Michal Hojsík [1, 3] and Bohuslav Rudolf [2, 3]

[1] The Selmer Center, University of Bergen, Norway

[2] National Security Authority, Czech Republic

[3] Department of Algebra, Charles University in Prague, Czech Republic

Fast Software Encryption 2008
February 10-13, Lausanne

Talk outline

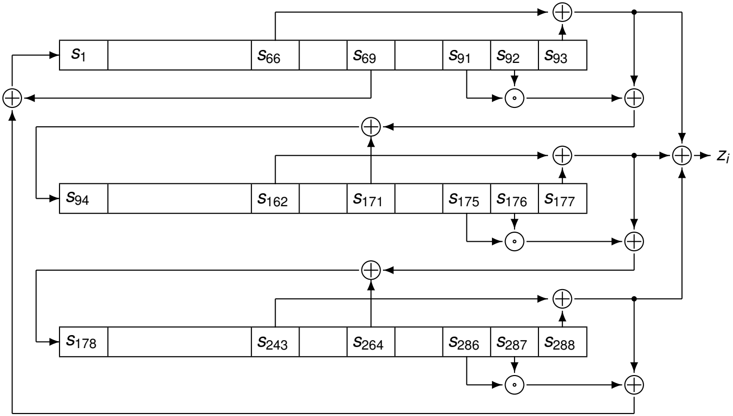
- Trivium description
- Differential fault analysis
- Differential fault analysis of Trivium
- Experimental results

Trivium

- Hardware oriented additive synchronous stream cipher
- Designed by de Cannière and Preneel in 2005 for eSTREAM Project
- Very fast in hardware and software
- 80-bit secret key and 80-bit initialisation vector
- Consists of 3 non-linear shift registers
- 288 bit inner state

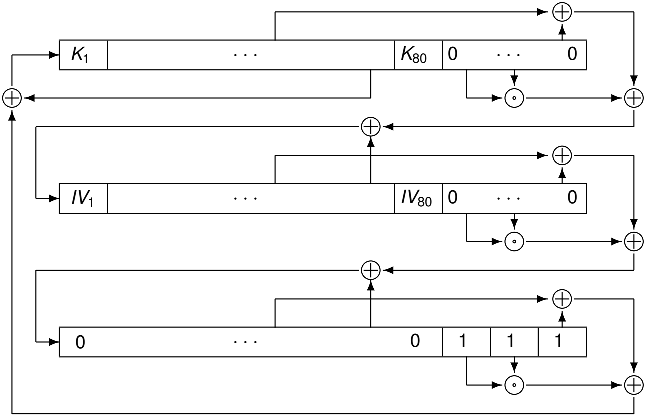
Trivium Description

- Inner state $IS = (s_1, \ldots, s_{288})$
- Keystream generation algorithm:



Trivium Description

- Secret key $K = (K_1, \ldots, K_{80})$, initialisation vector $IV = (IV_1, \ldots, IV_{80})$
- Initialisation algorithm = 1152 loops of the keystream gen. alg. without output



Differential Fault Analysis - DFA

- Type of active side-channel attack - adversary actively interferes with a cryptosystem
- First used in 1996 by Boneh et al. for RSA and by Biham and Shamir for DES
- Results on stream ciphers, e.g.
    - Hoch, Shamir 2004 – Fault Analysis of LFSR based ciphers, Lili128, Sober-t32
    - Biham, Grandboulan 2005 – Impossible Fault Analysis of RC4

DFA Attack Model

General DFA attack model:

- Attacker is able to inject a fault into a cipher inner state or intermediate result
- Attacker has only partial control over their number, location, timing ...
- Attacker can reset the device to its original state and repeat fault injection

Our assumptions:

Attacker is able to:

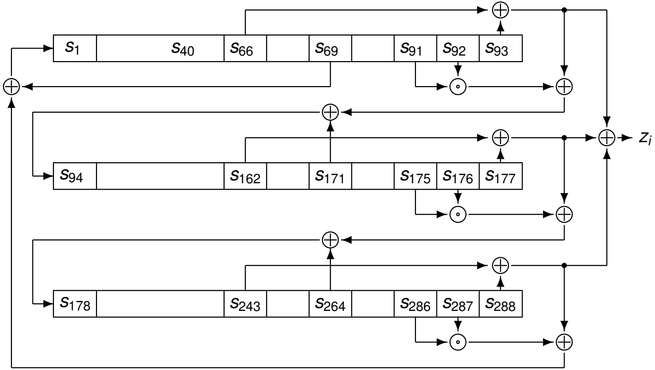
- obtain first n consecutive bits of (proper) keystream $\{z_i\}$ produced out of a state IS_t
- inject exactly one fault (bit flip) into IS_t at random position $\rightarrow$ faulty inner state IS'_t
- obtain first n consecutive bits of faulty keystream $\{z'_i\}$ produced out of IS'_t
- repeat the fault injection into the same inner state IS_t m times

Can be achieved in the Chosen ciphertext attack scenario

Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

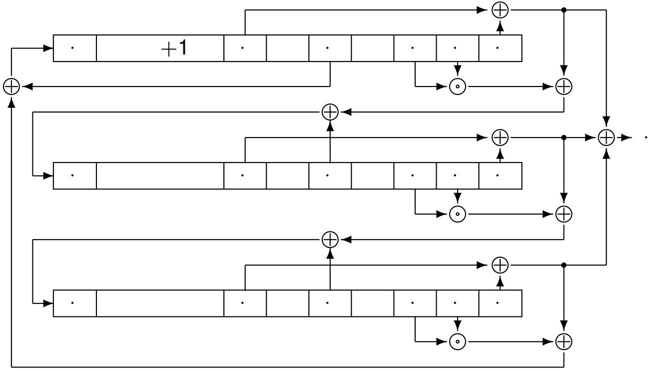
$$(x+1) \cdot y + x \cdot y = y$$



Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

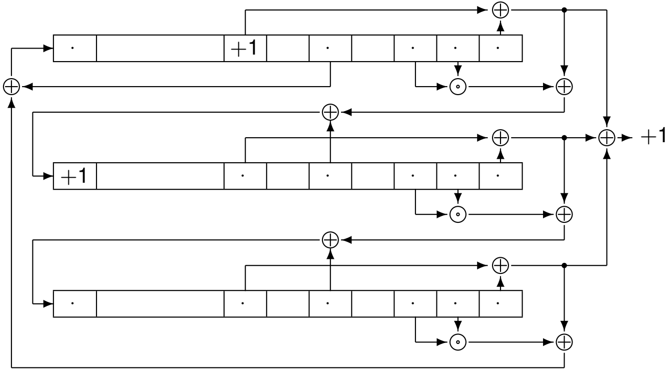
$$(x + 1) \cdot y + x \cdot y = y$$



Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

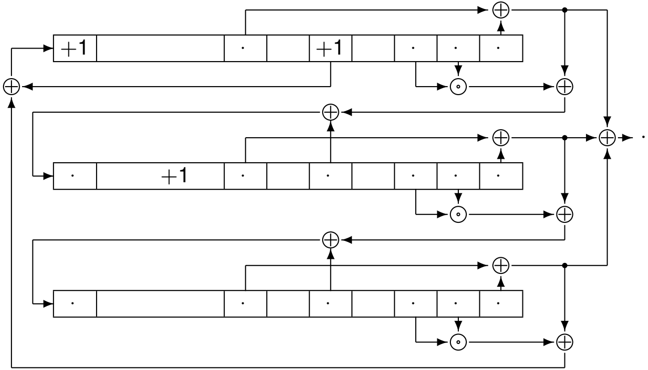
$$(x+1) \cdot y + x \cdot y = y$$



Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

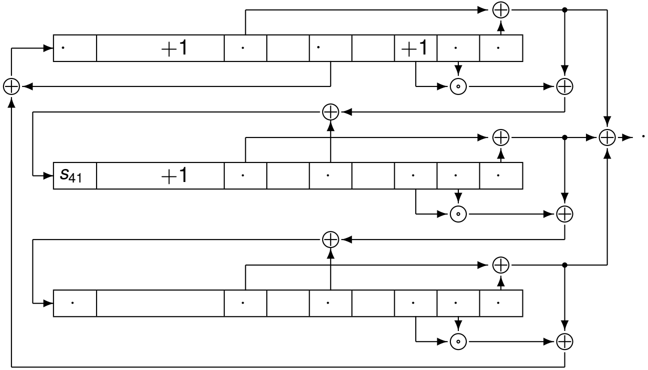
$$(x+1)\cdot y + x\cdot y = y$$



Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

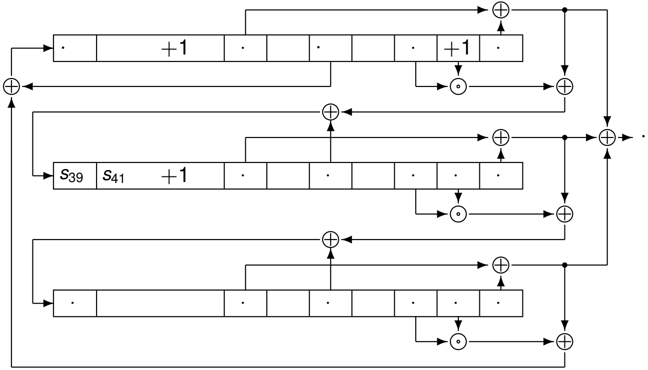
$$(s_{40} + 1) \cdot s_{41} + s_{40} \cdot s_{41} = s_{41}$$



Fault Injection - Trivium

- Attack is based on the simplicity of the Trivium feedback functions
- Attack uses simple equation

$$s_{39} \cdot (s_{40} + 1) + s_{39} \cdot s_{40} = s_{39}$$



Attack Description I

- Core of the attack - solve a system of equations in the inner state bits $IS_t = (s_1, \dots, s_{288})$
- Use equations given by the (proper) keystream $\{z_i\}$
- Use differential fault analysis to obtain more equations
- **Precomputation:** for each fault position e, $1 \leq e \leq 288$
  - express first n delta-keystream bits as expression is $(s_1, \dots, s_{288})$
  - store the equations in a table
- **Fault position determination:**
  - distance between the output bits differs for each register
  - compute the distances between nonzero bits of a keystream difference
  - determine the fault position - table lookup

Attack Description III

- Attack algorithm:

```
- obtain the proper keystream generated from IS_t
- insert the keystream equations into the system
while solution not found
    - reset the cipher to the state IS_t
    - insert a fault into IS_t at random position
    - obtain the faulty keystream
    - determine the fault position
    - insert delta keystream equations into the system
    - try to solve the system
end while
- clock Trivium backwards until initial state reached
- read the secret key and IV
```

Experimental Results

Attack:

Number of fault injections needed, m, to obtain T inner state bits (avg. over 1000 exp.)

T	60	80	100	120	140	160	180	200	220	240	260	280	288
m	28	35	39	41	42	42	42	42	42	43	43	43	43

Number of obtained equations:

The average number (among all fault positions) of equations obtained from a random fault:

number of steps	The average number of equations of degree d obtained from one fault.						
	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$	$d = 7$
200	1.99	2.52	0.89	0	0	0	0
220	1.99	4.14	1.53	0	0	0	0
240	1.99	5.99	2.82	0.03	0	0	0
260	1.99	7.76	4.15	1.13	0.45	0.37	0.28
280	1.99	9.22	5.22	3.42	1.47	1.23	0.96
300	1.99	9.77	5.86	7.10	3.55	2.66	2.09

New Results (January 2008)

- New DFA attack on Trivium
- Same assumptions as in the described attack
- Attack uses another cipher representation
- Attacker needs approx. 12 fault injections to obtain the secret key and IV

Conclusion

- Differential fault analysis of Trivium described
- The first time DFA applied to non-linear feedback shift register stream cipher
- Attacker can obtain the secret key after approx. 43 (12) fault injections
- Attack works in chosen ciphertext attack scenario
- Described attacks have low complexity and are easy to implement

Thank you for your attention!