DIFFERENTIAL FAULT ANALYSIS OF SHA3-224 AND SHA3-256

Pei LUO [1], **Yunsi FEI** [1], Liwei ZHANG [2], A. Adam DING [2]

1. Electrical & Computer Engineering, Northeastern University

Northeastern University Energy-Efficient and Secure Systems Lab (http://nueess.coe.neu.edu)

2. Department of Mathematics, Northeastern University

Northeastern University

Outline

- Motivation and contribution
- Preliminary of SHA-3
- Fault propagation in SHA-3
- Fault injection attacks simulation results
- Conclusion

Motivation and Contribution

- Motivation
  - Security of SHA-3/Keccak is very important
  - Previous work [1]
    - Under single-bit fault model
    - Targets only two modes of SHA-3: SHA3-384 and SHA3-512

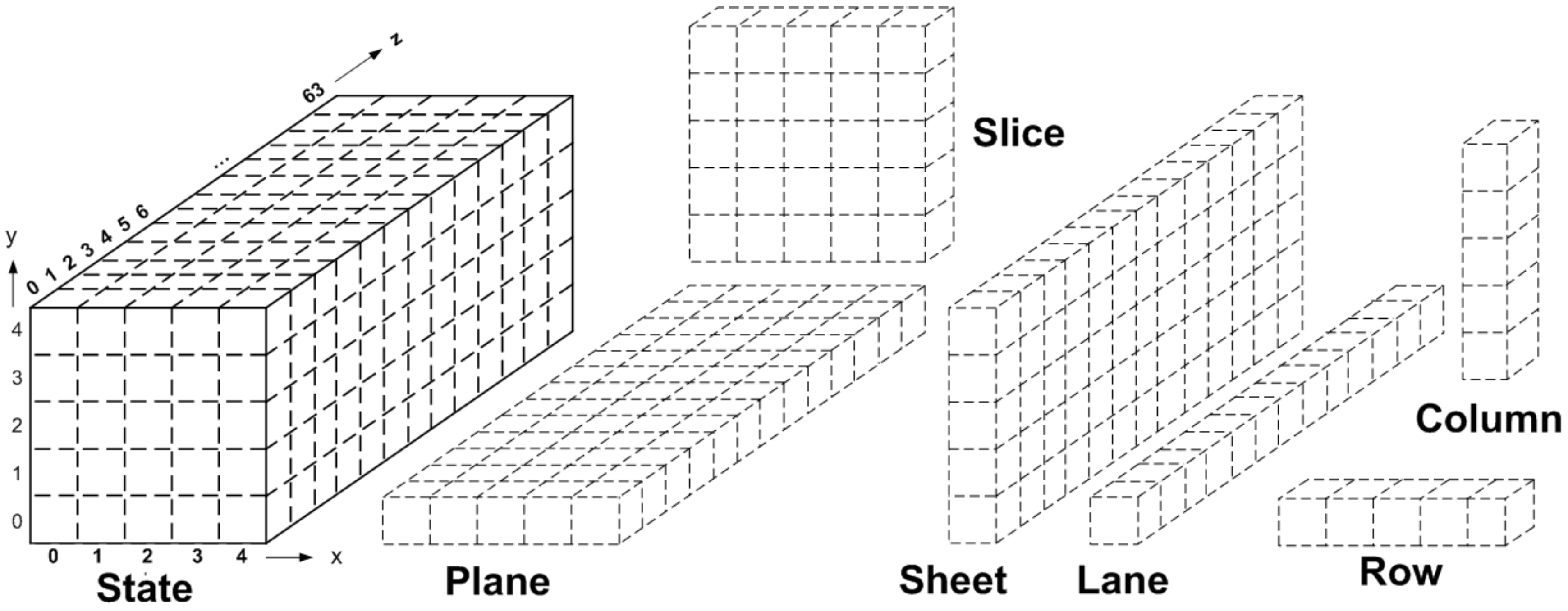
- Our Contribution
  - Extend differential fault analysis to relaxed fault models
  - Conquer other two modes of SHA-3: SHA3-224 and SHA3-256

1. Bagheri, Nasour, Navid Ghaedi, and Somitra Kumar Sanadhya. "Differential fault analysis of SHA-3." *International Conference in Cryptology in India*. Springer International Publishing, 2015.

Outline

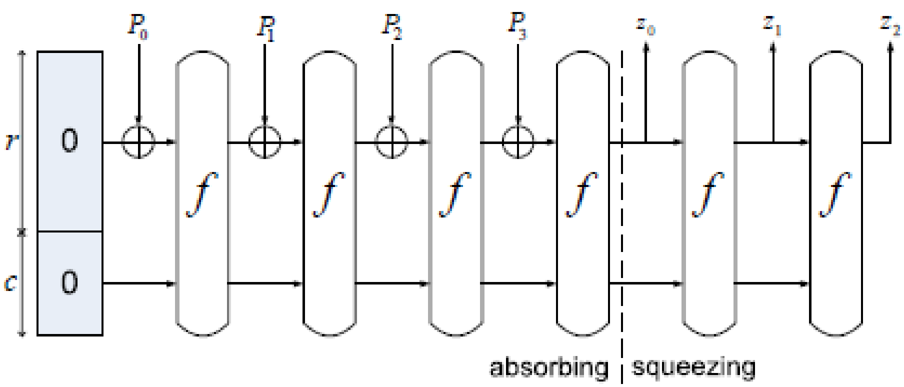
- Motivation and contribution
- Preliminary of SHA-3
- Fault propagation in SHA-3
- Fault injection attacks simulation results
- Conclusion

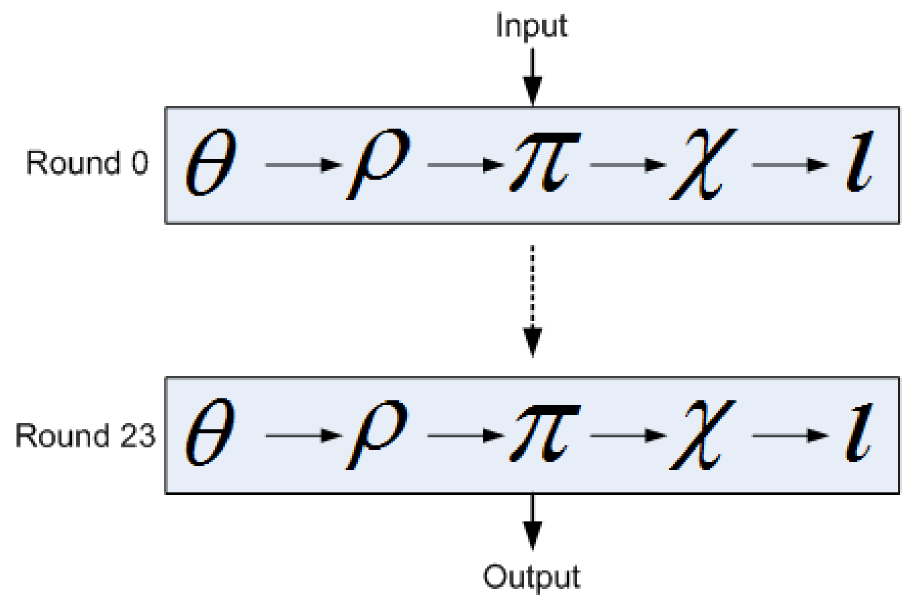
Preliminary of SHA-3



Preliminary of SHA-3

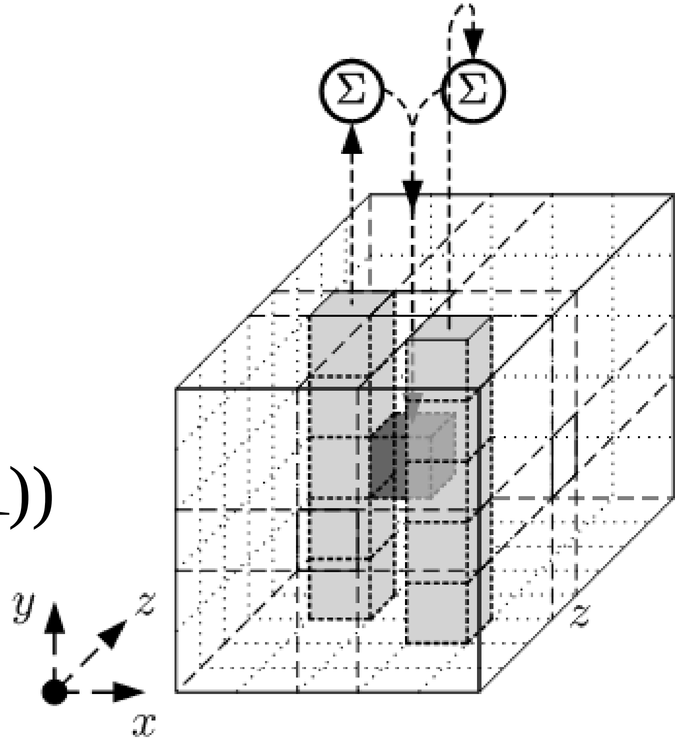
- Sponge function: repeated permutation function, *f*, for message absorbing and digest squeezing
- One *f* function for 1600 bits: 24 rounds, 5 operations in each round





Operations of SHA-3 - θ

$$\theta_{\mathrm{o}}(x,y,z)=\theta_{\mathrm{i}}(x,y,z)\oplus\Big(\bigoplus_{y=0}^{4}\theta_{\mathrm{i}}(x-1,y,z)\Big)$$
$$\oplus\Big(\bigoplus_{y=0}^{4}\theta_{\mathrm{i}}(x+1,y,z-1)\Big)$$

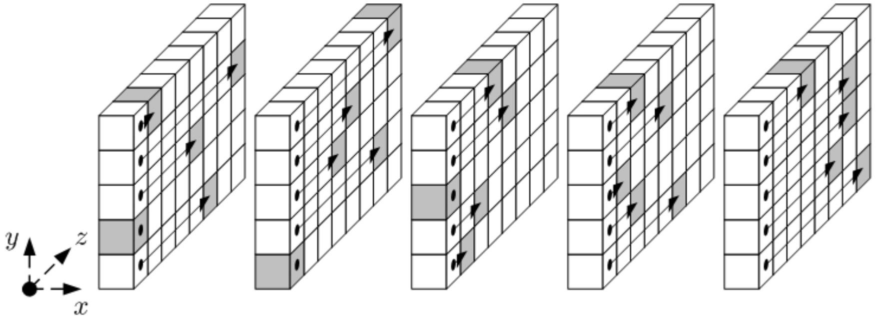


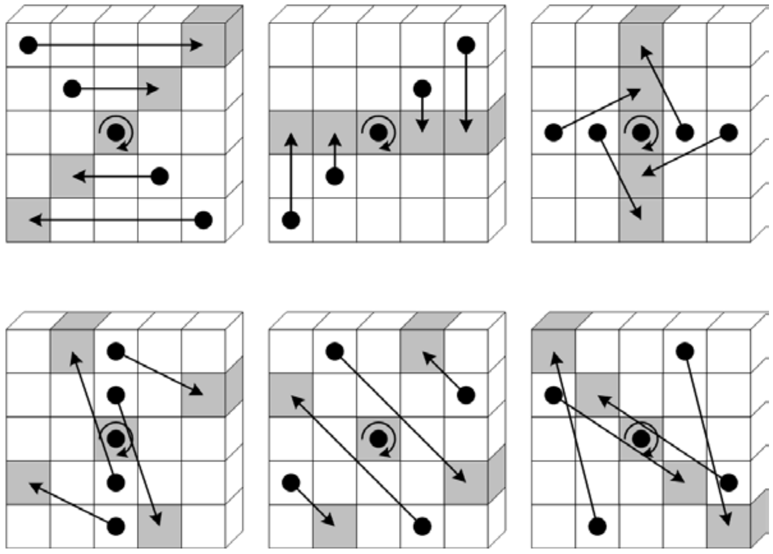
- In another view, one input θ bit will affect 11 output bits

Operations of SHA-3 - permutations

ρ changes the positions of bits along each lane

π changes the positions of bits inside each slice

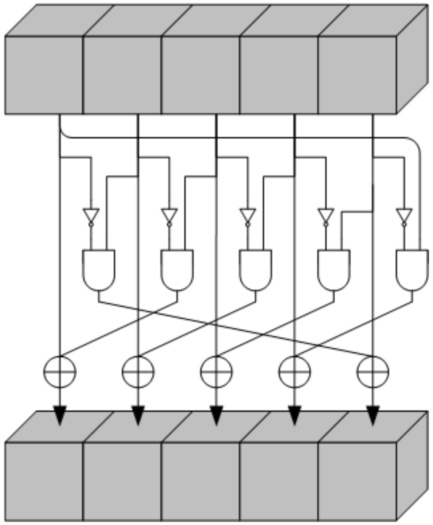




Operations of SHA-3 – non-linear χ

χ involves nonlinear operations, and it is reversible:

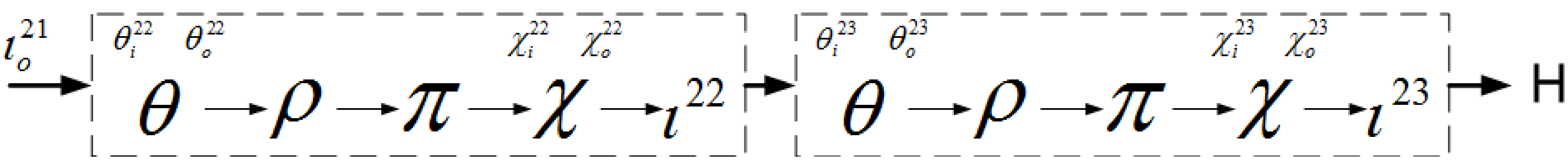
$$\chi_{\mathrm{o}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \chi_{\mathrm{i}}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \oplus (\overline{\chi_{\mathrm{i}}(\boldsymbol{x}+1,\boldsymbol{y},\boldsymbol{z})} \cdot \chi_{\mathrm{i}}(\boldsymbol{x}+2,\boldsymbol{y},\boldsymbol{z}))$$



$$\chi_{\mathrm{i}}(x,y,z) = \chi_{\mathrm{o}}(x,y,z) \oplus \overline{\chi_{\mathrm{o}}(x+1,y,z)} \cdot [\chi_{\mathrm{o}}(x-1,y,z) \oplus \chi_{\mathrm{o}}(x+2,y,z) \oplus \chi_{\mathrm{o}}(x-1,y,z) \cdot \chi_{\mathrm{o}}(x+3,y,z)]$$

Fault Model and Notations

- Attack goal: recover one internal state – χ_i^{22}
- Fault model:
  - Random single-byte faults injected at θ_i^{22}
  - Observable digest *H, d* bits for SHA3-*d* function
    - 224 bits for SHA3-224 (three and half lanes on the bottom plane)
    - 256 bits for SHA3-256 (four lanes on the the bottom plane)
  - Attacker can inject multiple faults for the same message

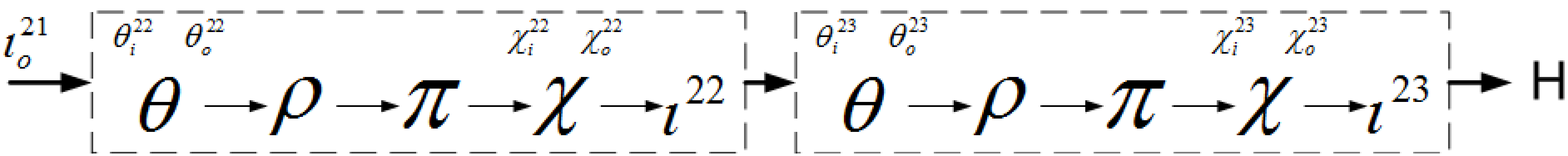


Outline

- Motivation and contribution
- Preliminary of SHA-3
- Fault propagation in SHA-3
- Fault injection attacks simulation results
- Conclusion

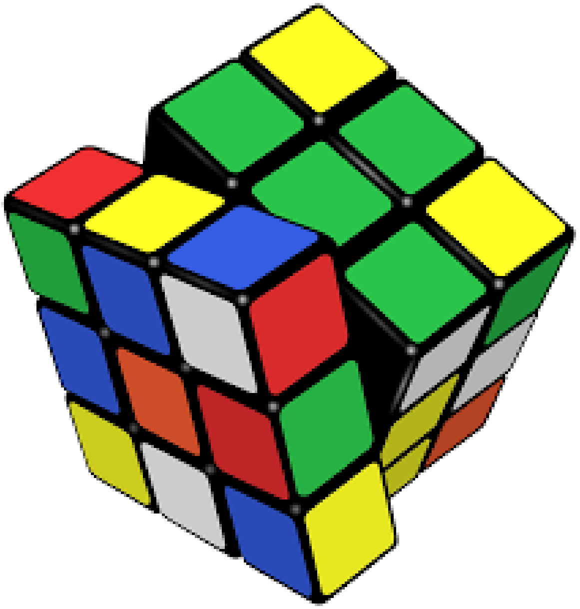
Attack Method

- Attack method
  - Inject a random fault at an internal state (θ_i^{22})
  - Observe the pair of original digest and faulty digest under this fault injection (H and H')
  - Select an internal state as the comparison point (χ_i^{23})
  - Derive the differential (fault) on the comparison state from the observed pair of digest reversely ($\Delta\chi_i^{23}$)
  - Compare this differential against the fault signatures under all possible faults (FS[P][F])
  - Identify the unique fault injected and recover some internal state bits



Fault Signature - Fault Propagation in SHA-3

- We define **fault signature (FS)** as the differential between the original state and faulty state under a specific fault injection
- Previous block ciphers like AES are operated at byte level
- SHA-3 is operated at bit level



Fault Propagation by SHA-3 Operations

- Operations that do not change the value of FS bits (ρ, π, and ı)

$$\Delta\rho_o = \rho(\Delta\rho_i) \qquad \Delta\pi_o = \pi(\Delta\pi_i) \qquad \Delta\iota_o = \Delta\iota_i$$

- Operations that change the value of FS bits $\Delta\theta_o = \theta(\Delta\theta_i)$
  - Operation θ:
  - Operation χ, denote the fault propagation function as FP_χ

$$\Delta\chi_i^{23} = \pi \circ \rho \circ \theta \circ \boldsymbol{FP}_\chi(\Delta\chi_i^{22})$$

$$\theta_i^{22}(0,0,0)$$

- Example in this talk: fault injected at

$$\Delta\theta_i^{22}(0,0,0) = 1 \text{ while other bits are } 0$$

Faults Signature at χ_i^{22}

```
x=0:
  10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  y=0
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00001000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   4
x=1:
  00000000 00000000 00000000 00000000 00000000 00001000 00000000 00000000  y=0
  00000000 00000000 00000100 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   4
x=2:
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  y=0
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 10000000 00000000 00000000   4
x=3:
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  y=0
  00000000 00000000 00000000 00000000 00000000 00000100 00000000 00000000   .
  00000000 01000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   4
x=4:
  00000000 00000001 00000000 00000000 00000000 00000000 00000000 00000000  y=0
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   .
  00100000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   4
z=0                         ... ... ... ... ... ... ... ...                 63
```

Fault Propagation of χ^{22}

Fault at χ input $\Delta\chi_i^{22}([x:x+2],y,z)$	Fault signature at output $FS_{\chi_o^{22}}(x,y,z)$
[1,0,0]	1
[0,1,0]	$\chi_i^{22}(x+2,y,z)$
[0,0,1]	$\overline{\chi_i^{22}(x+1,y,z)}$
[1,1,0]	$\overline{\chi_i^{22}(x+2,y,z)}$
[0,1,1]	$\chi_i^{22}(x+1,y,z)\oplus\chi_i^{22}(x+2,y,z)$
[1,0,1]	$\chi_i^{22}(x+1,y,z)$
[1,1,1]	$\overline{\chi_i^{22}(x+1,y,z)\oplus\chi_i^{22}(x+2,y,z)}$

Fault Signature at χ_i^{23}

xx100000 00**xx**000**1** 00000**x1**0 0000**x**000 00000000 0**x**00**x1x**0 00000000 00000000

$$E(0,0)=1\oplus\chi_i^{22}(1,0,0);\ E(0,1)=\chi_i^{22}(1,2,1);\ E(0,10)=1\oplus\chi_i^{22}(2,2,9);\ E(0,11)=\chi_i^{22}(3,3,10);\ E(0,46)=1\oplus\chi_i^{22}(2,1,45);$$
$$E(0,21)=1\oplus\chi_i^{22}(0,1,21);\ E(0,28)=\chi_i^{22}(1,3,28);\ E(0,41)=\chi_i^{22}(3,4,40);\ E(0,44)=1\oplus\chi_i^{22}(0,0,44)\oplus\chi_i^{22}(2,0,44);$$

0**x**000000 **1**0000000 0000**x1**00 **xxx**00000 00000000 0000**11**0**x** 000000**x1** 0000**x**000

$$E(1,1)=1\oplus\chi_i^{22}(2,1,21);\ E(1,20)=1\oplus\chi_i^{22}(1,4,40);\ E(1,24)=\chi_i^{22}(2,0,44);\ E(1,25)=1\oplus\chi_i^{22}(2,1,45);$$
$$E(1,26)=\chi_i^{22}(4,1,45);\ E(1,47)=1\oplus\chi_i^{22}(3,4,2);\ E(1,54)=1\oplus\chi_i^{22}(4,2,9)\oplus\chi_i^{22}(1,3,10);\ E(1,60)=1\oplus\chi_i^{22}(3,0,15);$$

10000000 **x**0000000 000**x**000**1** **x1**000000 00000000 0000**xxx**0 0000**xx**00 000**x**0000

$$E(2,8)=1\oplus\chi_i^{22}(4,3,28);\ E(2,19)=\chi_i^{22}(3,4,40);\ E(2,24)=1\oplus\chi_i^{22}(2,1,45);\ E(2,44)=1\oplus\chi_i^{22}(4,0,0);\ E(2,45)=1\oplus\chi_i^{22}(4,2,1);$$
$$E(2,46)=\chi_i^{22}(0,4,2);\ E(2,52)=1\oplus\chi_i^{22}(2,2,9)\oplus\chi_i^{22}(4,2,9);\ E(2,53)=1\oplus\chi_i^{22}(3,3,10);\ E(2,59)=\chi_i^{22}(0,0,15);$$

00**x**00000 00000000 000000**xx** **1**00000**x1** 0000**x1**00 000**x**0000 0**xx**00000 00000**1**00

$$E(3,2)=1\oplus\chi_i^{22}(0,0,44)\oplus\chi_i^{22}(4,1,45);\ E(3,22)=\chi_i^{22}(1,0,0);\ E(3,23)=1\oplus\chi_i^{22}(1,2,1)\oplus\chi_i^{22}(3,4,2);\ E(3,30)=\chi_i^{22}(4,2,9);$$
$$E(3,36)=1\oplus\chi_i^{22}(3,0,15);\ E(3,43)=1\oplus\chi_i^{22}(0,1,21);\ E(3,49)=1\oplus\chi_i^{22}(4,3,28);\ E(3,50)=\chi_i^{22}(1,3,28);$$

00000000 000000**xx** **x**000000**1** 0**x**000**x**00 0000**x**000 00**x1**0000 0000000**x** 000**x**0000

$$E(4,14)=1\oplus\chi_i^{22}(4,0,0);\ E(4,15)=\chi_i^{22}(4,2,1);\ E(4,16)=\chi_i^{22}(0,4,2);\ E(4,25)=1\oplus\chi_i^{22}(1,3,10);\ E(4,29)=\chi_i^{22}(0,0,15);$$
$$E(4,36)=\chi_i^{22}(2,1,21);\ E(4,42)=1\oplus\chi_i^{22}(4,3,28);\ E(4,55)=1\oplus\chi_i^{22}(1,4,40);\ E(4,59)=1\oplus\chi_i^{22}(2,0,44);$$

- Each FS bit can be denoted as ‘0’, ‘1’, ‘x’
  - ‘x’ means it depends on some χ_i^{22} bits

$\Delta\chi_i^{23}$ Bits Recovery from the Digests

χ is reversible:

$$\left\{\begin{aligned}
a_i &= a_o \oplus \overline{b_o} \cdot (e_o \oplus c_o \oplus e_o \cdot d_o) \\
b_i &= b_o \oplus \overline{c_o} \cdot (a_o \oplus d_o \oplus a_o \cdot e_o) \\
c_i &= c_o \oplus \overline{d_o} \cdot (b_o \oplus e_o \oplus b_o \cdot a_o) \\
d_i &= d_o \oplus \overline{e_o} \cdot (c_o \oplus a_o \oplus c_o \cdot b_o) \\
e_i &= e_o \oplus \overline{a_o} \cdot (d_o \oplus b_o \oplus d_o \cdot c_o)
\end{aligned}\right.$$

For example, for $a_i = a_o \oplus \overline{b_o} \cdot (e_o \oplus c_o \oplus e_o \cdot d_o)$:

If $d_o = 1$, $a_i = a_o \oplus \overline{b_o} \cdot c_o$;

If $b_o = 1$, $a_i = a_o$;

The probability of recovering a_i with the output row known is: $P(d_o = 1 \mid b_o = 1) = 0.75$

	a_i	b_i	c_i	d_i		e_i
				1-32	33-64	
SHA3-224	0.75	0.75	0.5	0.5	0	0
SHA3-256	0.75	0.75	0.5	0.5	0.5	0

$\Delta\chi_i^{23}$ bits recovery – a simple method

$$\begin{cases} (a_i^0, b_i^0, c_i^0, d_i^0, e_i^0) = \chi^{-1}(a_o, b_o, c_o, d_o, 0) \\ (a_i^1, b_i^1, c_i^1, d_i^1, e_i^1) = \chi^{-1}(a_o, b_o, c_o, d_o, 1) \end{cases}$$

If $a_i^0 = a_i^1$, a_i does not depend on e_o, attacker can recover a_i;

If $a_i^0 \neq a_i^1$, a_i depends on e_o, attacker cannot recover a_i.

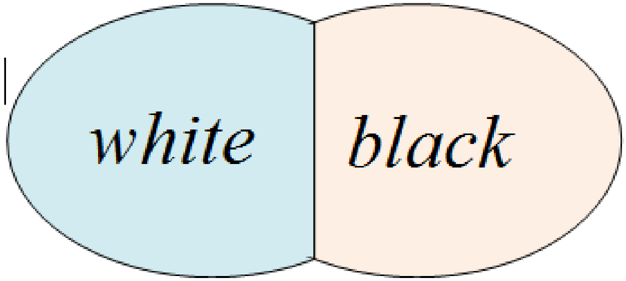
	Number of recovered bits	
	χ_i^{23}	$\Delta\chi_i^{23}$
SHA3-224	111.84	93.68
SHA3-256	160.12	136.42

Outline

- Motivation and contribution
- Preliminary of SHA-3
- Fault propagation in SHA-3
- Fault injection attacks simulation results
- Conclusion

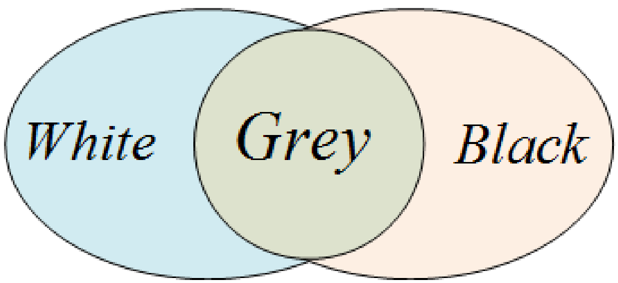
Fault Identification

- $\Delta\chi_i^{23}$ and $FS_{\chi_i^{23}}$ both have two grou
  - {black}: bits that are flipped
  - {white}: bits that are not flipped



$\Delta\chi_i^{23}$

$$\begin{cases} FS_{\chi_i^{23}}[P][F].white \subseteq \Delta\chi_i^{23}.white \\ FS_{\chi_i^{23}}[P][F].black \subseteq \Delta\chi_i^{23}.black \end{cases}$$

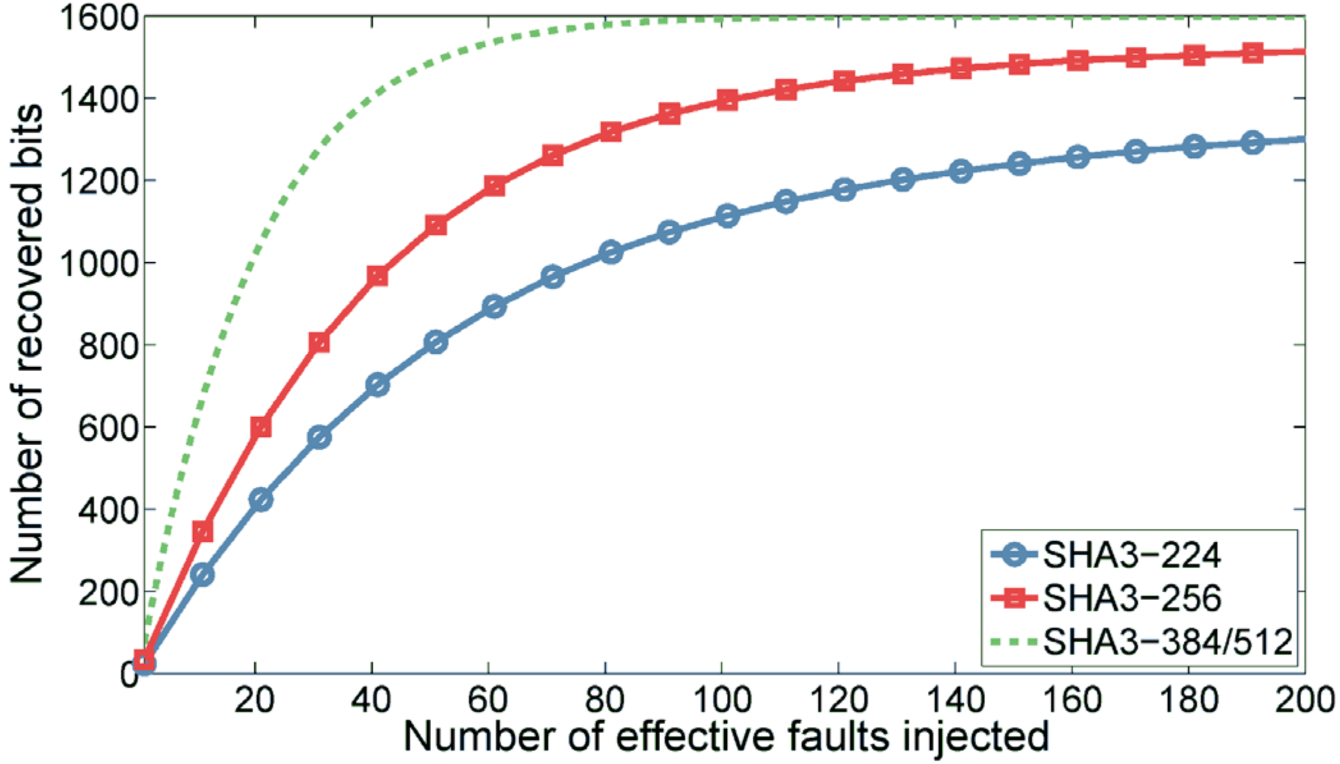


$FS_{\chi_i^{23}}$

- $FS_{\chi_i^{23}}[P][F]$ has another group $FS_{\chi_i^{23}}[P][F].grey$
  - ‘x’ bits can be either 0 or 1,

$$\begin{cases} \Delta\chi_i^{23}.white \subseteq \{FS_{\chi_i^{23}}[P][F].white \cup FS_{\chi_i^{23}}[P][F].grey\} \\ \Delta\chi_i^{23}.black \subseteq \{FS_{\chi_i^{23}}[P][F].black \cup FS_{\chi_i^{23}}[P][F].grey\} \end{cases}$$

Fault Identification and Bits Recovery



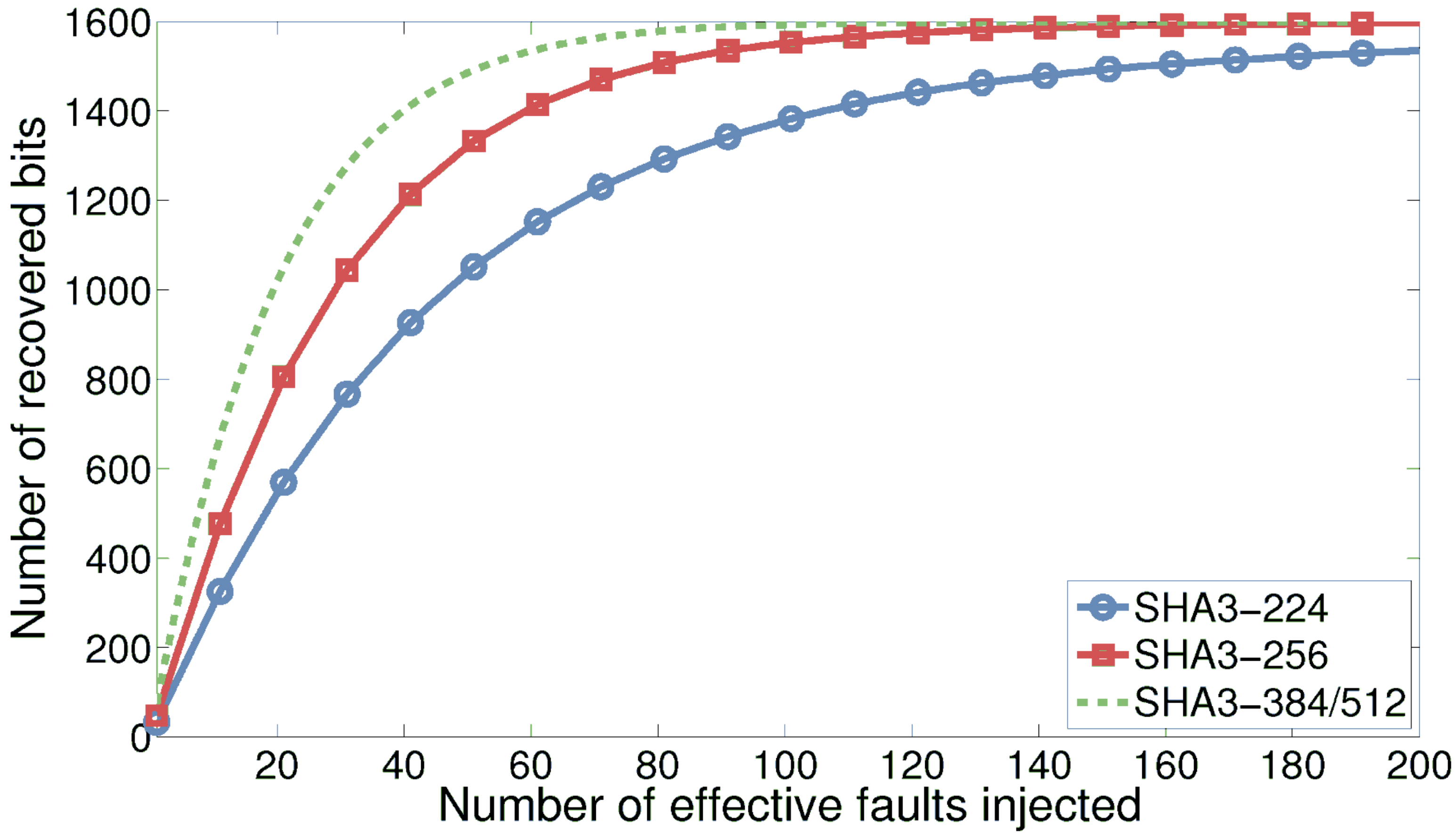
	Number of recovered bits		Probability of unique fault
	χ_i^{23}	$\Delta\chi_i^{23}$	
SHA3-224	111.84	93.68	30.67%
SHA3-256	160.12	136.42	66.61%

Improvement

- The proposed method
  - Can efficiently recover χ_i^{23} bits
  - Can identify the injected fault and then recover χ_i^{22} bits
  - Attacks on SHA3-224/256 less efficient than SHA3-384/512
    - Limited number of $\Delta\chi_i^{23}$ and $FS_{\chi_i^{23}}$ bits

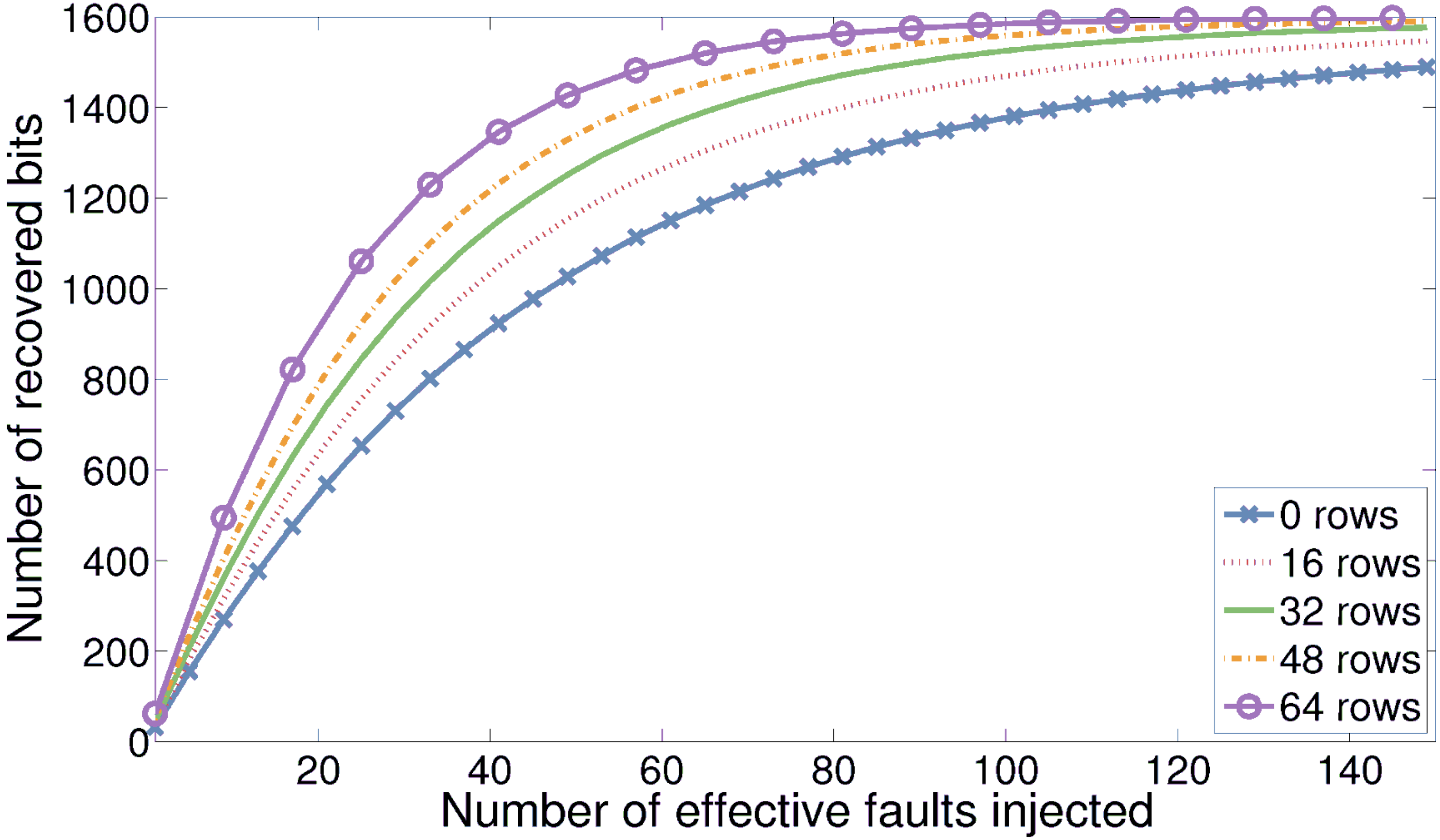
- Improvement
  - Make use of $FS_{\chi_o^{23}}$ toghther with $FS_{\chi_i^{23}}$
    - $FS_{\chi_o^{23}}$ contains extra information
  - Inject faults at θ_i^{23} to recover more bits of χ_i^{23} and $\Delta\chi_i^{23}$
    - More $\Delta\chi_i^{23}$ bits contain more information

Improvement – Involve $FS_{\chi_o^{23}}$



Improvement – Recover more χ_i^{23}

- Assume different number of χ_i^{23} rows recovered for SHA3-224



Outline

- Motivation and contribution
- Preliminary of SHA-3
- Fault propagation in SHA-3
- Fault injection attacks simulation results
- Conclusion

Conclusion and Future Work

- Conclusion
  - The proposed method can effectively conquer SHA3-224 and SHA3-256
  - The proposed improvement method can further improve the efficiency
  - SHA3-224 and SHA3-256 are more difficult to conquer than SHA3-384 and SHA3-512 under DFA
- Future work
  - More relaxed fault model
  - Different fault injection position
  - Further improve effective fault ratio

Acknowledgement

- This work was supported in part by National Science Foundation under grants SaTC-1314655 and MRI-1337854.

- Simulation code used in this paper is available at http://tescase.coe.neu.edu/

Thanks!