riscure

Flip a bit, grab a key: Symbolic execution edition

Jasper van Woudenberg
@jzvw

CTO Riscure North America

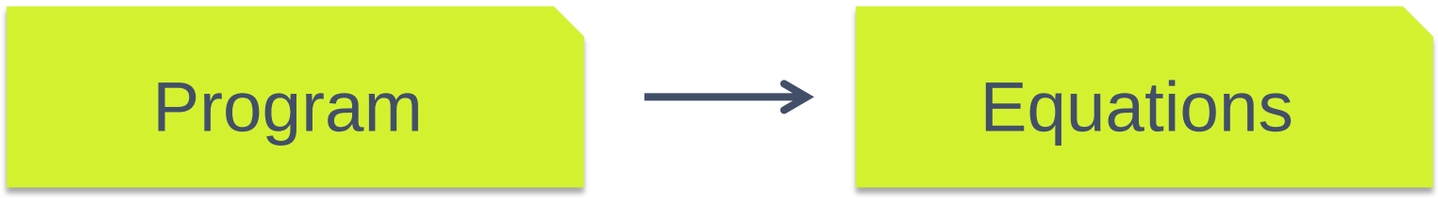
public

Concrete execution

```
r1 = 0xBE; r2 = 0x08

mov   r1,r2

add   r1,0x10

r1 = 0x18; r2 = 0x08
```

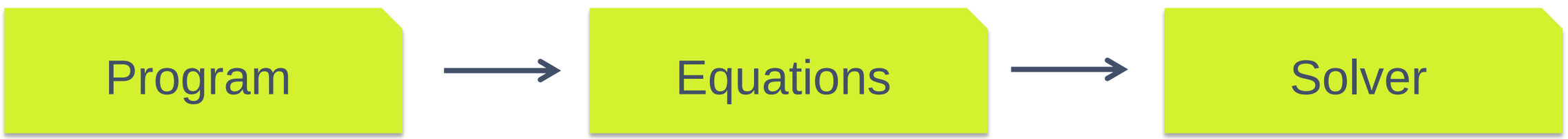
Symbolic execution

```
r1 = 0xBE; r2 = 0x??

mov   r1,r2

add   r1,0x10

r1 = r2 + 0x10; r2 = 0x??
```



Symbolic execution

```
r1 = 0xBE; r2 = 0x??
mov   r1,r2
add   r1,0x10
beq   r1,0x20,A:
mov   r3,0x00
b     B:
A:
mov   r3,0x01
B:

r2==0x10: r1 = r2 + 0x10; r2 = 0x??; r3=0x00
r2!=0x10: r1 = r2 + 0x10; r2 = 0x??; r3=0x01
```



Fault injection

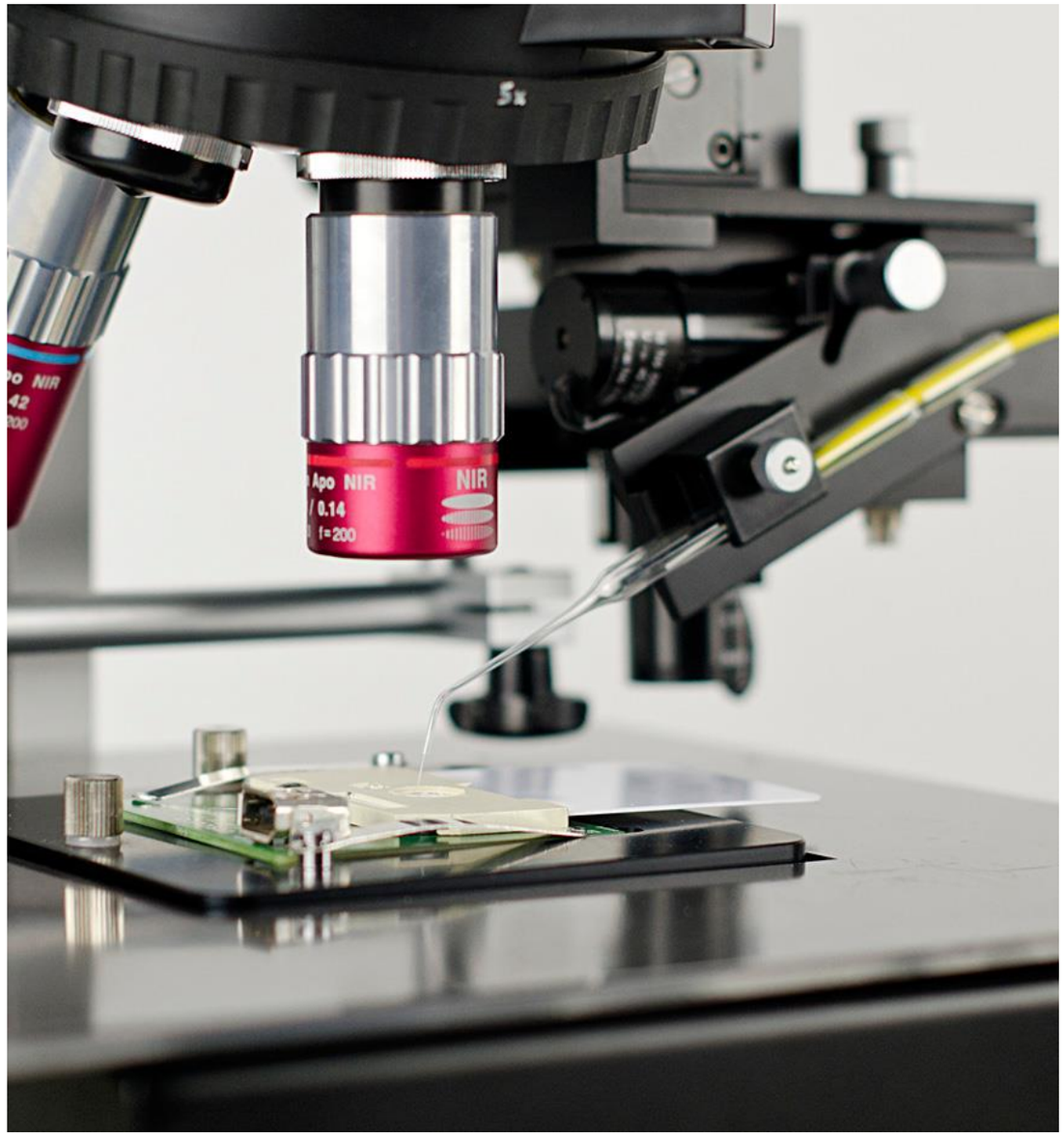
```
r1 = 0xBE; r2 = 0x08

mov   r1,r2

add   r1,0x10

r1 = 0xCE; r2 = 0x08
```

Cause a(n exploitable) corruption on a device

Fault injection (hardwear)

riscure

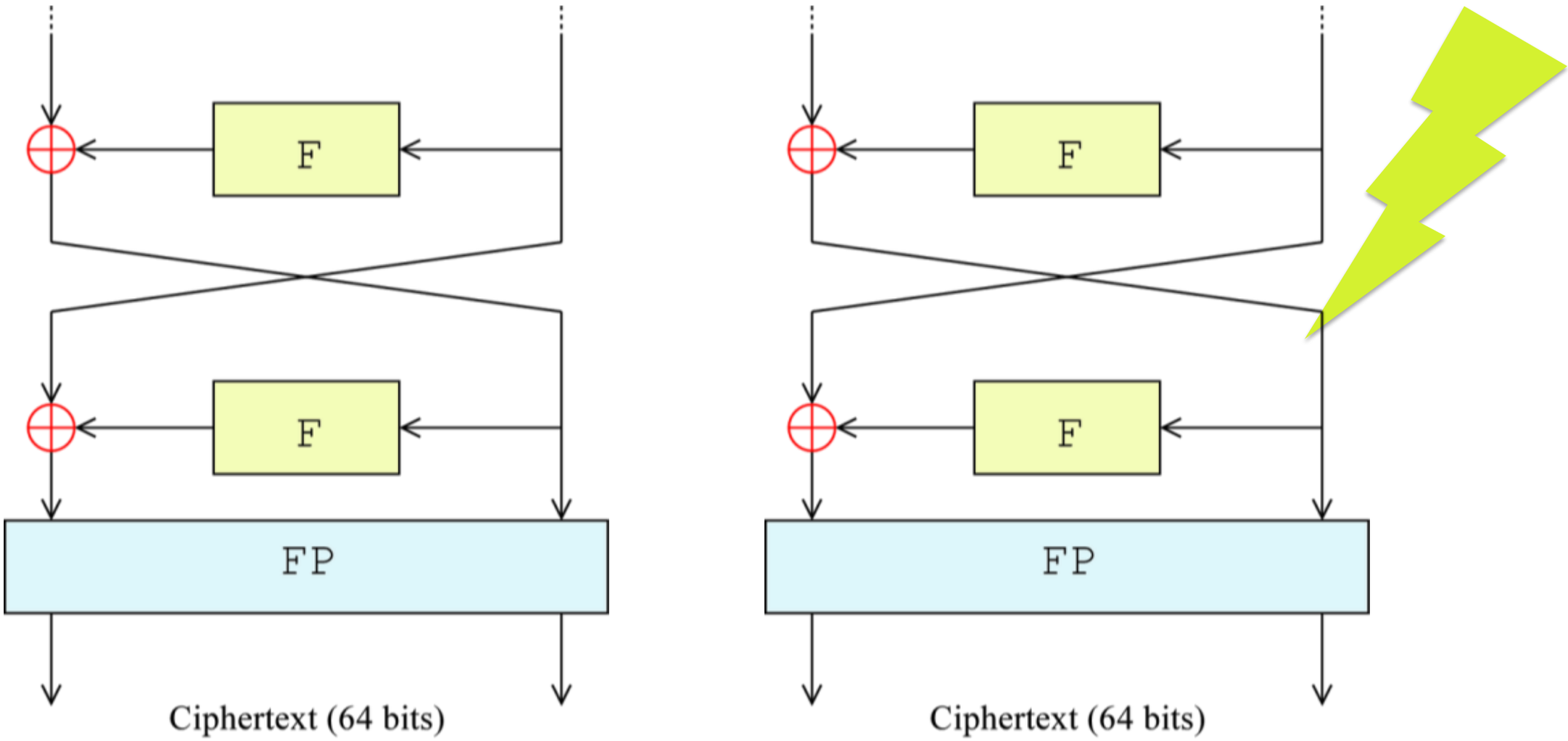
Cause a(n exploitable) corruption on a device



Fault injection (softwear)

riscure

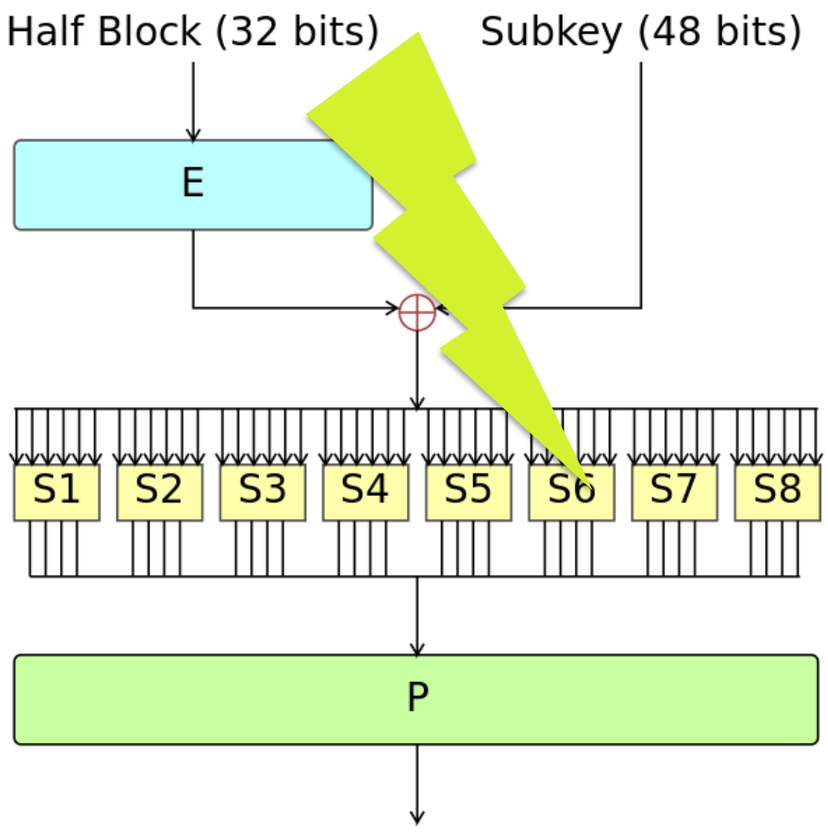
Differential Fault Analysis (DES)



When R15 faulted, only a few K16 will match both outputs

Inside F function…

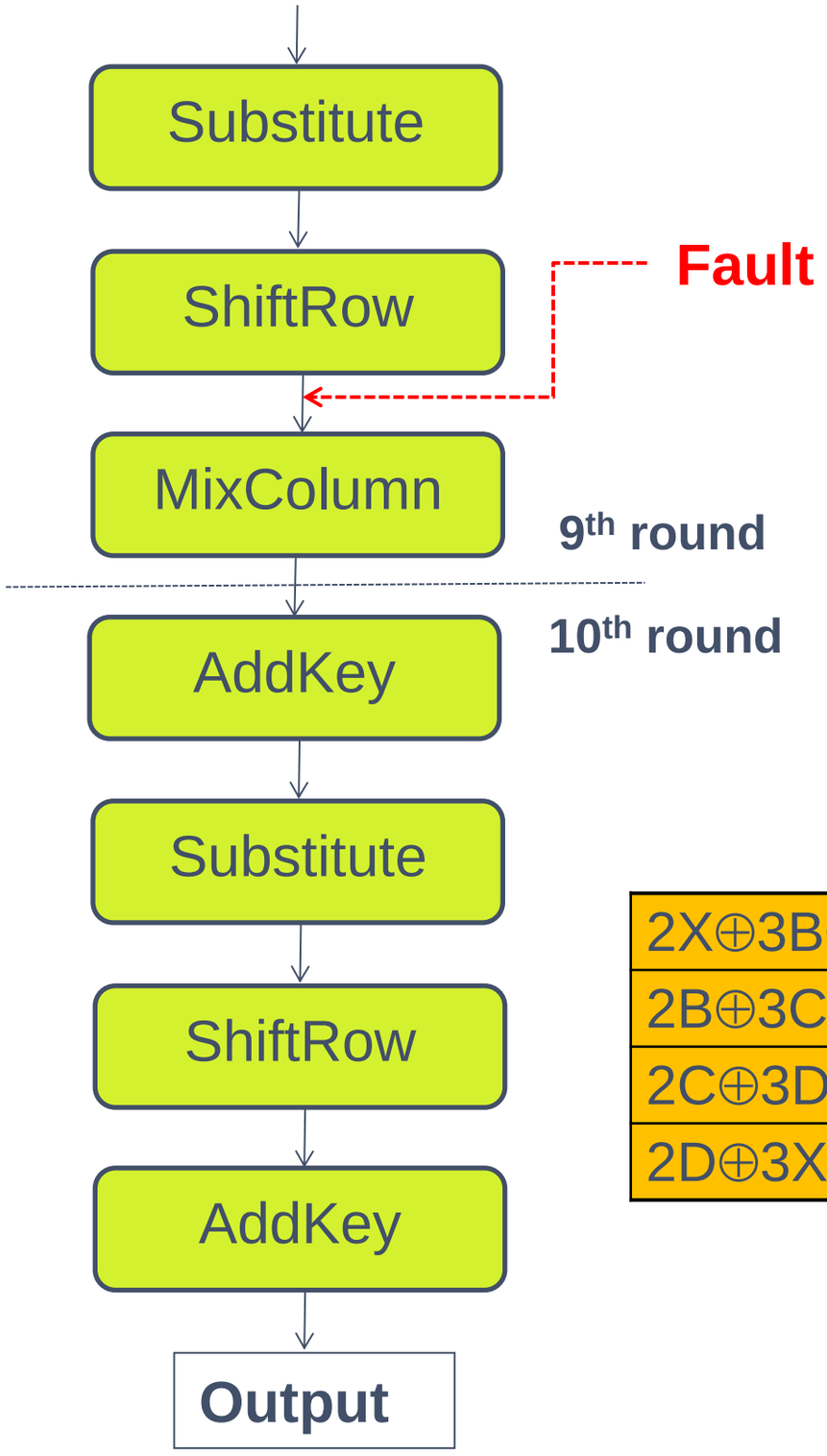
Track fault to sbox; calculate faulted sbox ⊕ normal sbox



Fault match

K	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S	B	7	E	0	1	4	7	A	6	1	0	D	8	B	D	6
S’	6	1	0	D	8	B	D	6	B	7	E	0	1	4	7	A
⊕	D	6	E	D	9	F	A	C	D	6	E	D	9	F	A	C
K	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
S	4	9	3	E	2	F	C	5	9	2	5	8	F	C	A	3
S’	9	2	5	8	F	C	A	3	4	9	3	E	2	F	C	5
⊕	D	B	6	6	D	3	6	6	D	B	6	6	D	3	6	6
K	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
S	6	0	1	D	D	3	E	4	0	E	B	7	3	5	8	B
S’	0	E	B	7	3	5	8	B	6	0	1	D	D	3	E	4
⊕	6	E	A	A	E	6	6	F	6	E	A	A	E	6	6	F
K	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
S	A	C	F	1	4	A	2	F	7	9	C	2	9	6	5	8
S’	7	9	C	2	9	6	5	8	A	C	F	1	4	A	2	F
⊕	D	5	3	3	D	C	7	7	D	5	3	3	D	C	7	7

DFA on AES math (1)



- Fault is injected in penultimate round
- State before is:

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

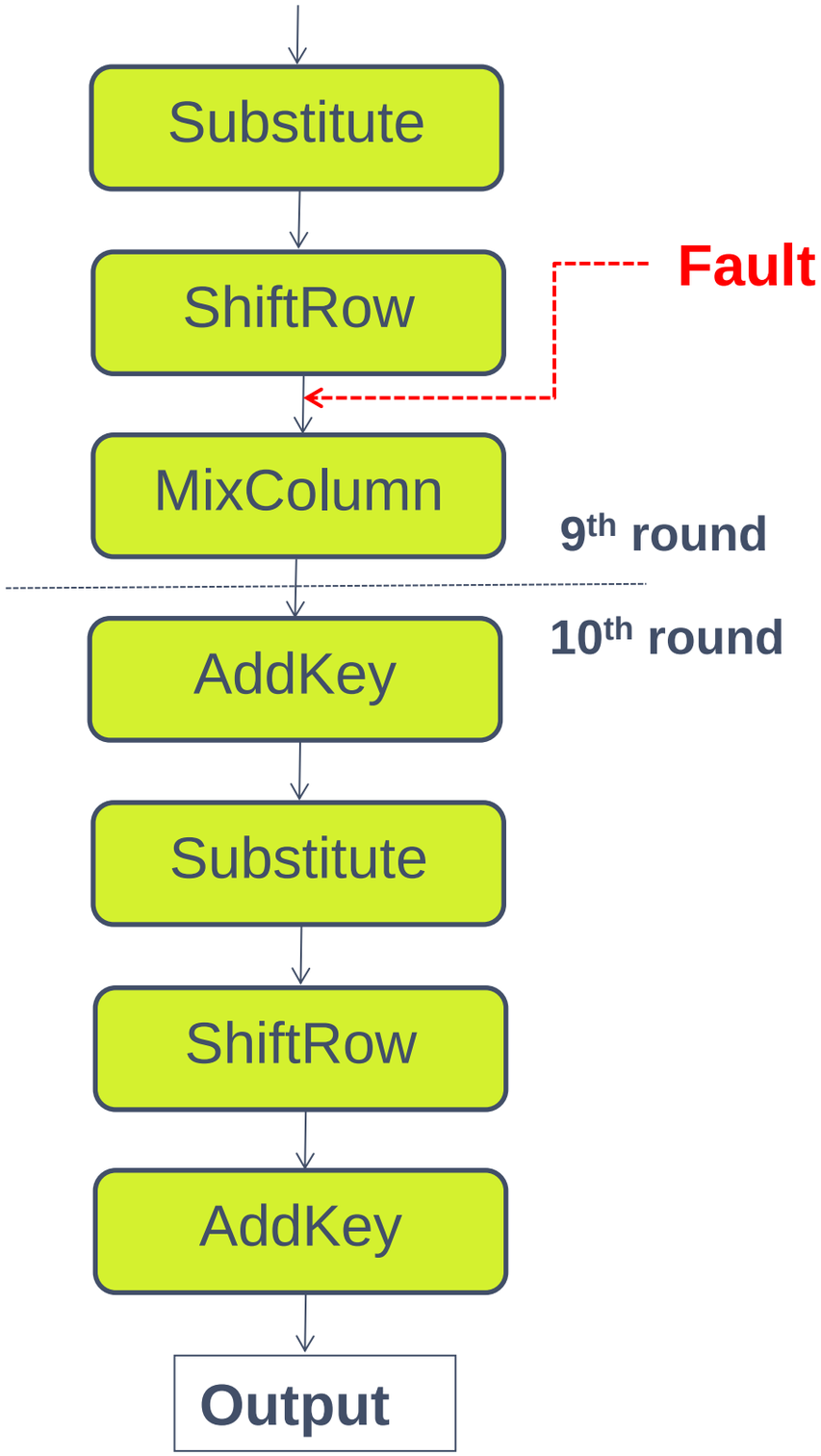
- Hit byte ‘A’
- State becomes:

X	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

- Apply MixColumn, and get

2X⊕3B⊕C⊕D	2E⊕3F⊕G⊕H	2I⊕3J⊕K⊕L	2M⊕3N⊕O⊕P
2B⊕3C⊕D⊕X	2F⊕3G⊕H⊕E	2J⊕3K⊕L⊕I	2N⊕3O⊕P⊕M
2C⊕3D⊕X⊕B	2G⊕3H⊕E⊕F	2K⊕3L⊕I⊕J	2O⊕3P⊕M⊕N
2D⊕3X⊕B⊕C	2H⊕3E⊕F⊕G	2L⊕3I⊕J⊕K	2P⊕3M⊕N⊕O

DFA on AES math (2)



- Apply AddKey, get for 1st column

$2X \oplus 3B \oplus C \oplus D \oplus K_{10,0}$
$2B \oplus 3C \oplus D \oplus X \oplus K_{10,1}$
$2C \oplus 3D \oplus X \oplus B \oplus K_{10,2}$
$2D \oplus 3X \oplus B \oplus C \oplus K_{10,3}$

- Apply Substitute and get

$S(2X \oplus 3B \oplus C \oplus D \oplus K_{10,0})$
$S(2B \oplus 3C \oplus D \oplus X \oplus K_{10,1})$
$S(2C \oplus 3D \oplus X \oplus B \oplus K_{10,2})$
$S(2D \oplus 3X \oplus B \oplus C \oplus K_{10,3})$

- ShiftRow only moves cell position
- Apply final AddKey, and get

$S(2X \oplus 3B \oplus C \oplus D \oplus K_{10,0}) \oplus K_{11,0}$
$S(2B \oplus 3C \oplus D \oplus X \oplus K_{10,1}) \oplus K_{11,13}$
$S(2C \oplus 3D \oplus X \oplus B \oplus K_{10,2}) \oplus K_{11,10}$
$S(2D \oplus 3X \oplus B \oplus C \oplus K_{10,3}) \oplus K_{11,7}$

DFA on AES math (3)

$S(2A \oplus 3B \oplus C \oplus D \oplus K_{10,0}) \oplus K_{11,0} = O_0$ Normal AES

$S(2X \oplus 3B \oplus C \oplus D \oplus K_{10,0}) \oplus K_{11,0} = O'_0$ Faulted AES

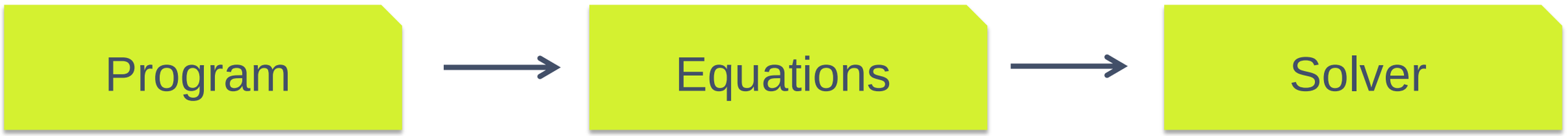
$S(Y_0) \oplus S(2Z \oplus Y_0) = O_0 \oplus O'_0$

Solve for Z, K

The insight

riscure

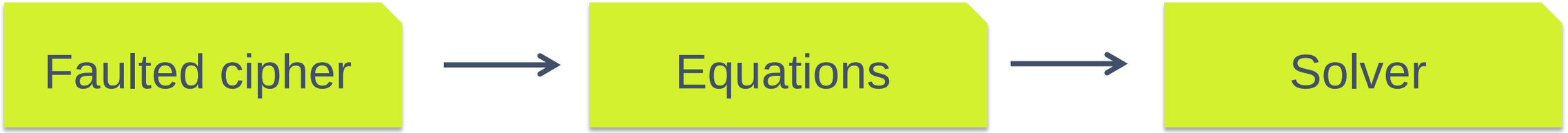
Symbolic execution



Differential fault analysis

The insight

Use SE for DFA



First experiments

Symbolic state, fault, key

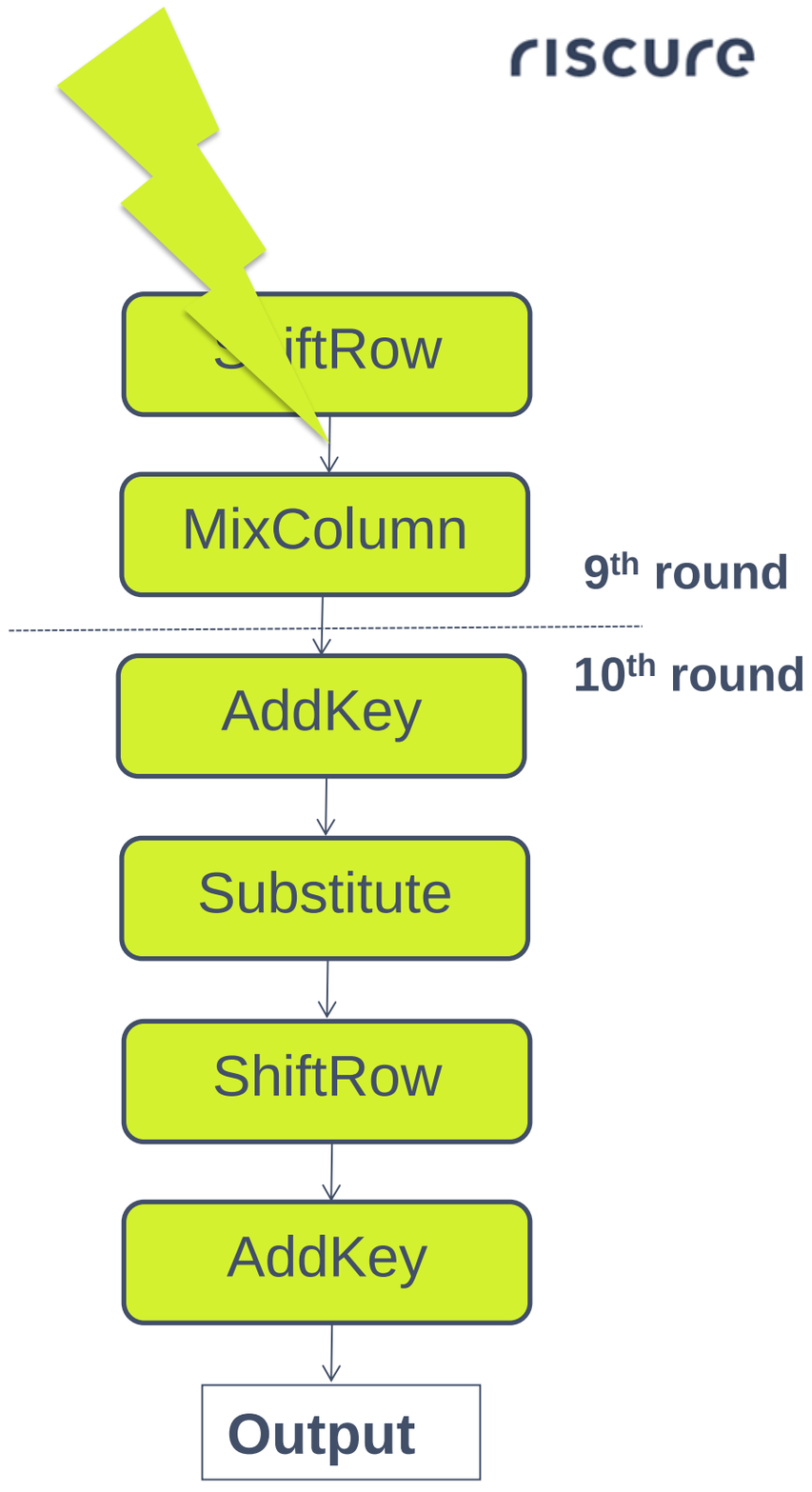
Equations:

- pAES(state,key)=output_0

…

- pAES(state⊕fault_n,key)=output_n

Solve for key!

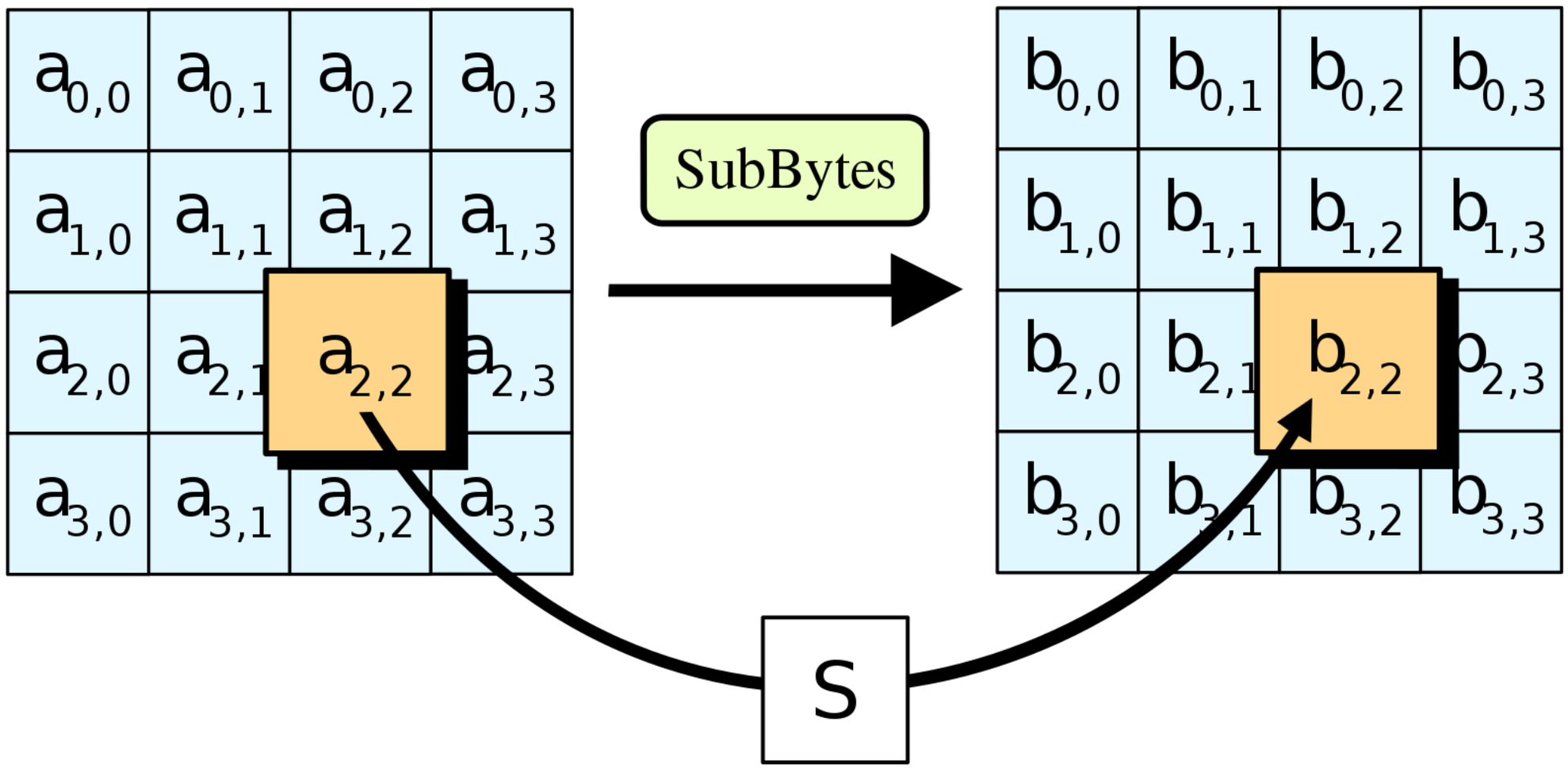


FAIL…

riscure

```
out=Sbox[in]
```

If in==0 then 0x63, else if … else if in==0xff then 0x16



Non-bitsliced crypto

$$1\,0\,1\,0 \oplus 0\,1\,0\,0 = X$$

Bitsliced crypto (slow)

1 ⊕ **0** = X0

0 ⊕ **1** = X1

1 ⊕ **0** = X2

0 ⊕ **0** = X3

Bitsliced crypto (parallel)

riscure

1 0 1 0 0 1 0 1 ⊕ **0** 1 1 0 1 0 0 1 = X0

0 1 0 0 1 1 0 0 ⊕ **1** 1 0 0 1 1 0 1 = X1

1 1 1 1 0 1 1 0 ⊕ **0** 1 1 0 0 1 1 0 = X2

0 1 1 0 1 0 1 0 ⊕ **0** 1 0 0 0 1 1 1 = X3

LUT based AES Sbox


```
out=Sbox[in]
```

Bitsliced AES Sbox

```
T1 = U[7] ^ U[4]; T2 = U[7] ^ U[2]; T3 = U[7] ^ U[1]; T4 = U[4] ^ U[2]; T5 = U[3] ^
U[1]; T6 = T1 ^ T5; T7 = U[6] ^ U[5]; T8 = U[0] ^ T6; T9 = U[0] ^ T7; T10 = T6 ^
T7; T11 = U[6] ^ U[2]; T12 = U[5] ^ U[2]; T13 = T3 ^ T4; T14 = T6 ^ T11; T15 = T5 ^
T11; T16 = T5 ^ T12; T17 = T9 ^ T16; T18 = U[4] ^ U[0]; T19 = T7 ^ T18; T20 = T1 ^
T19; T21 = U[1] ^ U[0]; T22 = T7 ^ T21; T23 = T2 ^ T22; T24 = T2 ^ T10; T25 = T20 ^
T17; T26 = T3 ^ T16; T27 = T1 ^ T12; M1 = T13 & T6; M2 = T23 & T8; M3 = T14 ^ M1;
M4 = T19 & U[0]; M5 = M4 ^ M1; M6 = T3 & T16; M7 = T22 & T9; M8 = T26 ^ M6; M9 =
T20 & T17; M10 = M9 ^ M6; M11 = T1 & T15; M12 = T4 & T27; M13 = M12 ^ M11; M14 = T2
& T10; M15 = M14 ^ M11; M16 = M3 ^ M2; M17 = M5 ^ T24; M18 = M8 ^ M7; M19 = M10 ^
M15; M20 = M16 ^ M13; M21 = M17 ^ M15; M22 = M18 ^ M13; M23 = M19 ^ T25; M24 = M22
^ M23; M25 = M22 & M20; M26 = M21 ^ M25; M27 = M20 ^ M21; M28 = M23 ^ M25; M29 =
M28 & M27; M30 = M26 & M24; M31 = M20 & M23; M32 = M27 & M31; M33 = M27 ^ M25; M34
= M21 & M22; M35 = M24 & M34; M36 = M24 ^ M25; M37 = M21 ^ M29; M38 = M32 ^ M33;
M39 = M23 ^ M30; M40 = M35 ^ M36; M41 = M38 ^ M40; M42 = M37 ^ M39; M43 = M37 ^
M38; M44 = M39 ^ M40; M45 = M42 ^ M41; M46 = M44 & T6; M47 = M40 & T8; M48 = M39 &
U[0]; M49 = M43 & T16; M50 = M38 & T9; M51 = M37 & T17; M52 = M42 & T15; M53 = M45
& T27; M54 = M41 & T10; M55 = M44 & T13; M56 = M40 & T23; M57 = M39 & T19; M58 =
M43 & T3; M59 = M38 & T22; M60 = M37 & T20; M61 = M42 & T1; M62 = M45 & T4; M63 =
M41 & T2; L0 = M61 ^ M62; L1 = M50 ^ M56; L2 = M46 ^ M48; L3 = M47 ^ M55; L4 = M54
^ M58; L5 = M49 ^ M61; L6 = M62 ^ L5; L7 = M46 ^ L3; L8 = M51 ^ M59; L9 = M52 ^
M53; L10 = M53 ^ L4; L11 = M60 ^ L2; L12 = M48 ^ M51; L13 = M50 ^ L0; L14 = M52 ^
M61; L15 = M55 ^ L1; L16 = M56 ^ L0; L17 = M57 ^ L1; L18 = M58 ^ L8; L19 = M63 ^
L4; L20 = L0 ^ L1; L21 = L1 ^ L7; L22 = L3 ^ L12; L23 = L18 ^ L2; L24 = L15 ^ L9;
L25 = L6 ^ L10; L26 = L7 ^ L9; L27 = L8 ^ L10; L28 = L11 ^ L14; L29 = L11 ^ L17;
S[7] = L6 ^ L24; S[6] = ~(L16 ^ L26); S[5] = ~(L19 ^ L28); S[4] = L6 ^ L21; S[3] =
L20 ^ L22; S[2] = L25 ^ L29; S[1] = ~(L13 ^ L27); S[0] = ~(L6 ^ L23);
```

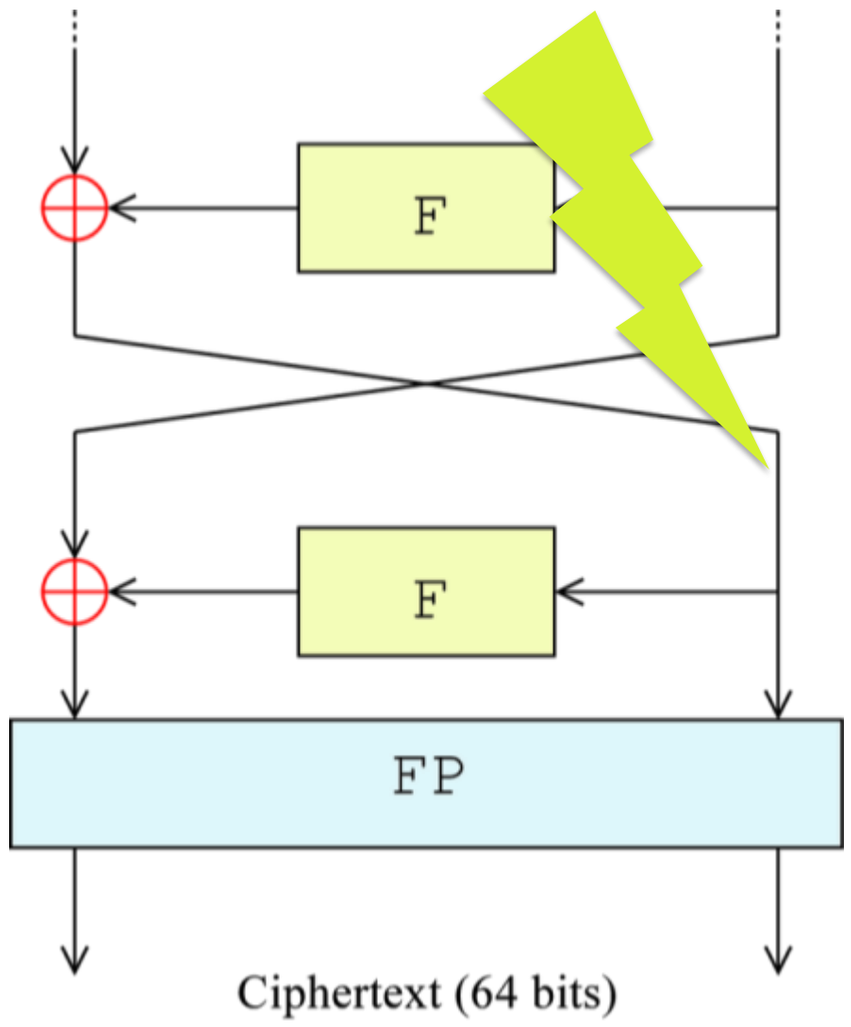
Bitsliced 8 DES sboxes


```
s1 () { x1 = ~a4; x2 = ~a1; x3 = a4 ^ a3; x4 = x3 ^ x2; x5 = a3 | x2; x6 = x5 & x1; x7 = a6 | x6; x8 = x4 ^ x7; x9 = x1 | x2; x10 = a6 & x9; x11 = x7
^ x10; x12 = a2 | x11; x13 = x8 ^ x12; x14 = x9 ^ x13; x15 = a6 | x14; x16 = x1 ^ x15; x17 = ~x14; x18 = x17 & x3; x19 = a2 | x18; x20 = x16 ^ x19;
x21 = a5 | x20; x22 = x13 ^ x21; *out4 ^= x22; x23 = a3 | x4; x24 = ~x23; x25 = a6 | x24; x26 = x6 ^ x25; x27 = x1 & x8; x28 = a2 | x27; x29 = x26 ^
x28; x30 = x1 | x8; x31 = x30 ^ x6; x32 = x5 & x14; x33 = x32 ^ x8; x34 = a2 & x33; x35 = x31 ^ x34; x36 = a5 | x35; x37 = x29 ^ x36; *out1 ^= x37;
x38 = a3 & x10; x39 = x38 | x4; x40 = a3 & x33; x41 = x40 ^ x25; x42 = a2 | x41; x43 = x39 ^ x42; x44 = a3 | x26; x45 = x44 ^ x14; x46 = a1 | x8; x47
= x46 ^ x20; x48 = a2 | x47; x49 = x45 ^ x48; x50 = a5 & x49; x51 = x43 ^ x50; *out2 ^= x51; x52 = x8 ^ x40; x53 = a3 ^ x11; x54 = x53 & x5; x55 = a2
| x54; x56 = x52 ^ x55; x57 = a6 | x4; x58 = x57 ^ x38; x59 = x13 & x56; x60 = a2 & x59; x61 = x58 ^ x60; x62 = a5 & x61; x63 = x56 ^ x62; *out3 ^=
x63;}s2 () { x1 = ~a5; x2 = ~a1; x3 = a5 ^ a6; x4 = x3 ^ x2; x5 = x4 ^ a2; x6 = a6 | x1; x7 = x6 | x2; x8 = a2 & x7; x9 = a6 ^ x8; x10 = a3 & x9; x11
= x5 ^ x10; x12 = a2 & x9; x13 = a5 ^ x6; x14 = a3 | x13; x15 = x12 ^ x14; x16 = a4 & x15; x17 = x11 ^ x16; *out2 ^= x17; x18 = a5 | a1; x19 = a6 |
x18; x20 = x13 ^ x19; x21 = x20 ^ a2; x22 = a6 | x4; x23 = x22 & x17; x24 = a3 | x23; x25 = x21 ^ x24; x26 = a6 | x2; x27 = a5 & x2; x28 = a2 | x27;
x29 = x26 ^ x28; x30 = x3 ^ x27; x31 = x2 ^ x19; x32 = a2 & x31; x33 = x30 ^ x32; x34 = a3 & x33; x35 = x29 ^ x34; x36 = a4 | x35; x37 = x25 ^ x36;
*out3 ^= x37; x38 = x21 & x32; x39 = x38 ^ x5; x40 = a1 | x15; x41 = x40 ^ x13; x42 = a3 | x41; x43 = x39 ^ x42; x44 = x28 | x41; x45 = a4 & x44; x46
= x43 ^ x45; *out1 ^= x46; x47 = x19 & x21; x48 = x47 ^ x26; x49 = a2 & x33; x50 = x49 ^ x21; x51 = a3 & x50; x52 = x48 ^ x51; x53 = x18 & x28; x54 =
x53 & x50; x55 = a4 | x54; x56 = x52 ^ x55; *out4 ^= x56;}s3 () { x1 = ~a5; x2 = ~a6; x3 = a5 & a3; x4 = x3 ^ a6; x5 = a4 & x1; x6 = x4 ^ x5; x7 = x6
^ a2; x8 = a3 & x1; x9 = a5 ^ x2; x10 = a4 | x9; x11 = x8 ^ x10; x12 = x7 & x11; x13 = a5 ^ x11; x14 = x13 | x7; x15 = a4 & x14; x16 = x12 ^ x15; x17
= a2 & x16; x18 = x11 ^ x17; x19 = a1 & x18; x20 = x7 ^ x19; *out4 ^= x20; x21 = a3 ^ a4; x22 = x21 ^ x9; x23 = x2 | x4; x24 = x23 ^ x8; x25 = a2 |
x24; x26 = x22 ^ x25; x27 = a6 ^ x23; x28 = x27 | a4; x29 = a3 ^ x15; x30 = x29 | x5; x31 = a2 | x30; x32 = x28 ^ x31; x33 = a1 | x32; x34 = x26 ^
x33; *out1 ^= x34; x35 = a3 ^ x9; x36 = x35 | x5; x37 = x4 | x29; x38 = x37 ^ a4; x39 = a2 | x38; x40 = x36 ^ x39; x41 = a6 & x11; x42 = x41 | x6;
x43 = x34 ^ x38; x44 = x43 ^ x41; x45 = a2 & x44; x46 = x42 ^ x45; x47 = a1 | x46; x48 = x40 ^ x47; *out3 ^= x48; x49 = x2 | x38; x50 = x49 ^ x13;
x51 = x27 ^ x28; x52 = a2 | x51; x53 = x50 ^ x52; x54 = x12 & x23; x55 = x54 & x52; x56 = a1 | x55; x57 = x53 ^ x56; *out2 ^= x57;}s4 () { x1 = ~a1;
x2 = ~a3; x3 = a1 | a3; x4 = a5 & x3; x5 = x1 ^ x4; x6 = a2 | a3; x7 = x5 ^ x6; x8 = a1 & a5; x9 = x8 ^ x3; x10 = a2 & x9; x11 = a5 ^ x10; x12 = a4 &
x11; x13 = x7 ^ x12; x14 = x2 ^ x4; x15 = a2 & x14; x16 = x9 ^ x15; x17 = x5 & x14; x18 = a5 ^ x2; x19 = a2 | x18; x20 = x17 ^ x19; x21 = a4 | x20;
x22 = x16 ^ x21; x23 = a6 & x22; x24 = x13 ^ x23; *out2 ^= x24; x25 = ~x13; x26 = a6 | x22; x27 = x25 ^ x26; *out1 ^= x27; x28 = a2 & x11; x29 = x28
^ x17; x30 = a3 ^ x10; x31 = x30 ^ x19; x32 = a4 & x31; x33 = x29 ^ x32; x34 = x25 ^ x33; x35 = a2 & x34; x36 = x24 ^ x35; x37 = a4 | x34; x38 = x36
^ x37; x39 = a6 & x38; x40 = x33 ^ x39; *out4 ^= x40; x41 = x26 ^ x38; x42 = x41 ^ x40; *out3 ^= x42;}s5 () { x1 = ~a6; x2 = ~a3; x3 = x1 | x2; x4 =
x3 ^ a4; x5 = a1 & x3; x6 = x4 ^ x5; x7 = a6 | a4; x8 = x7 ^ a3; x9 = a3 | x7; x10 = a1 | x9; x11 = x8 ^ x10; x12 = a5 & x11; x13 = x6 ^ x12; x14 =
~x4; x15 = x14 & a6; x16 = a1 | x15; x17 = x8 ^ x16; x18 = a5 | x17; x19 = x10 ^ x18; x20 = a2 | x19; x21 = x13 ^ x20; *out3 ^= x21; x22 = x2 | x15;
x23 = x22 ^ a6; x24 = a4 ^ x22; x25 = a1 & x24; x26 = x23 ^ x25; x27 = a1 ^ x11; x28 = x27 & x22; x29 = a5 | x28; x30 = x26 ^ x29; x31 = a4 | x27;
x32 = ~x31; x33 = a2 | x32; x34 = x30 ^ x33; *out2 ^= x34; x35 = x2 ^ x15; x36 = a1 & x35; x37 = x14 ^ x36; x38 = x5 ^ x7; x39 = x38 & x34; x40 = a5
| x39; x41 = x37 ^ x40; x42 = x2 ^ x5; x43 = x42 & x16; x44 = x4 & x27; x45 = a5 & x44; x46 = x43 ^ x45; x47 = a2 | x46; x48 = x41 ^ x47; *out1 ^=
x48; x49 = x24 & x48; x50 = x49 ^ x5; x51 = x11 ^ x30; x52 = x51 | x50; x53 = a5 & x52; x54 = x50 ^ x53; x55 = x14 ^ x19; x56 = x55 ^ x34; x57 = x4 ^
x16; x58 = x57 & x30; x59 = a5 & x58; x60 = x56 ^ x59; x61 = a2 | x60; x62 = x54 ^ x61; *out4 ^= x62;}s6 () { x1 = ~a2; x2 = ~a5; x3 = a2 ^ a6; x4 =
x3 ^ x2; x5 = x4 ^ a1; x6 = a5 & a6; x7 = x6 | x1; x8 = a5 & x5; x9 = a1 & x8; x10 = x7 ^ x9; x11 = a4 & x10; x12 = x5 ^ x11; x13 = a6 ^ x10; x14 =
x13 & a1; x15 = a2 & a6; x16 = x15 ^ a5; x17 = a1 & x16; x18 = x2 ^ x17; x19 = a4 | x18; x20 = x14 ^ x19; x21 = a3 & x20; x22 = x12 ^ x21; *out2 ^=
x22; x23 = a6 ^ x18; x24 = a1 & x23; x25 = a5 ^ x24; x26 = a2 ^ x17; x27 = x26 | x6; x28 = a4 & x27; x29 = x25 ^ x28; x30 = ~x26; x31 = a6 | x29; x32
= ~x31; x33 = a4 & x32; x34 = x30 ^ x33; x35 = a3 & x34; x36 = x29 ^ x35; *out4 ^= x36; x37 = x6 ^ x34; x38 = a5 & x23; x39 = x38 ^ x5; x40 = a4 |
x39; x41 = x37 ^ x40; x42 = x16 | x24; x43 = x42 ^ x1; x44 = x15 ^ x24; x45 = x44 ^ x31; x46 = a4 | x45; x47 = x43 ^ x46; x48 = a3 | x47; x49 = x41 ^
x48; *out1 ^= x49; x50 = x5 | x38; x51 = x50 ^ x6; x52 = x8 & x31; x53 = a4 | x52; x54 = x51 ^ x53; x55 = x30 & x43; x56 = a3 | x55; x57 = x54 ^ x56;
*out3 ^= x57;}s7 () { x1 = ~a2; x2 = ~a5; x3 = a2 & a4; x4 = x3 ^ a5; x5 = x4 ^ a3; x6 = a4 & x4; x7 = x6 ^ a2; x8 = a3 & x7; x9 = a1 ^ x8; x10 = a6
| x9; x11 = x5 ^ x10; x12 = a4 & x2; x13 = x12 | a2; x14 = a2 | x2; x15 = a3 & x14; x16 = x13 ^ x15; x17 = x6 ^ x11; x18 = a6 | x17; x19 = x16 ^ x18;
x20 = a1 & x19; x21 = x11 ^ x20; *out1 ^= x21; x22 = a2 | x21; x23 = x22 ^ x6; x24 = x23 ^ x15; x25 = x5 ^ x6; x26 = x25 | x12; x27 = a6 | x26; x28 =
x24 ^ x27; x29 = x1 & x19; x30 = x23 & x26; x31 = a6 & x30; x32 = x29 ^ x31; x33 = a1 | x32; x34 = x28 ^ x33; *out4 ^= x34; x35 = a4 & x16; x36 = x35
| x1; x37 = a6 & x36; x38 = x11 ^ x37; x39 = a4 & x13; x40 = a3 | x7; x41 = x39 ^ x40; x42 = x1 | x24; x43 = a6 | x42; x44 = x41 ^ x43; x45 = a1 |
x44; x46 = x38 ^ x45; *out2 ^= x46; x47 = x8 ^ x44; x48 = x6 ^ x15; x49 = a6 | x48; x50 = x47 ^ x49; x51 = x19 ^ x44; x52 = a4 ^ x25; x53 = x52 &
x46; x54 = a6 & x53; x55 = x51 ^ x54; x56 = a1 | x55; x57 = x50 ^ x56; *out3 ^= x57;}s8 () { x1 = ~a1; x2 = ~a4; x3 = a3 ^ x1; x4 = a3 | x1; x5 = x4
^ x2; x6 = a5 | x5; x7 = x3 ^ x6; x8 = x1 | x5; x9 = x2 ^ x8; x10 = a5 & x9; x11 = x8 ^ x10; x12 = a2 & x11; x13 = x7 ^ x12; x14 = x6 ^ x9; x15 = x3
& x9; x16 = a5 & x8; x17 = x15 ^ x16; x18 = a2 | x17; x19 = x14 ^ x18; x20 = a6 | x19; x21 = x13 ^ x20; *out1 ^= x21; x22 = a5 | x3; x23 = x22 & x2;
x24 = ~a3; x25 = x24 & x8; x26 = a5 & x4; x27 = x25 ^ x26; x28 = a2 | x27; x29 = x23 ^ x28; x30 = a6 & x29; x31 = x13 ^ x30; *out4 ^= x31; x32 = x5 ^
x6; x33 = x32 ^ x22; x34 = a4 | x13; x35 = a2 & x34; x36 = x33 ^ x35; x37 = a1 & x33; x38 = x37 ^ x8; x39 = a1 ^ x23; x40 = x39 & x7; x41 = a2 & x40;
x42 = x38 ^ x41; x43 = a6 | x42; x44 = x36 ^ x43; *out3 ^= x44; x45 = a1 ^ x10; x46 = x45 ^ x22; x47 = ~x7; x48 = x47 & x8; x49 = a2 | x48; x50 = x46
^ x49; x51 = x19 ^ x29; x52 = x51 | x38; x53 = a6 & x52; x54 = x50 ^ x53; *out2 ^= x54;}
```

Success (demo)!

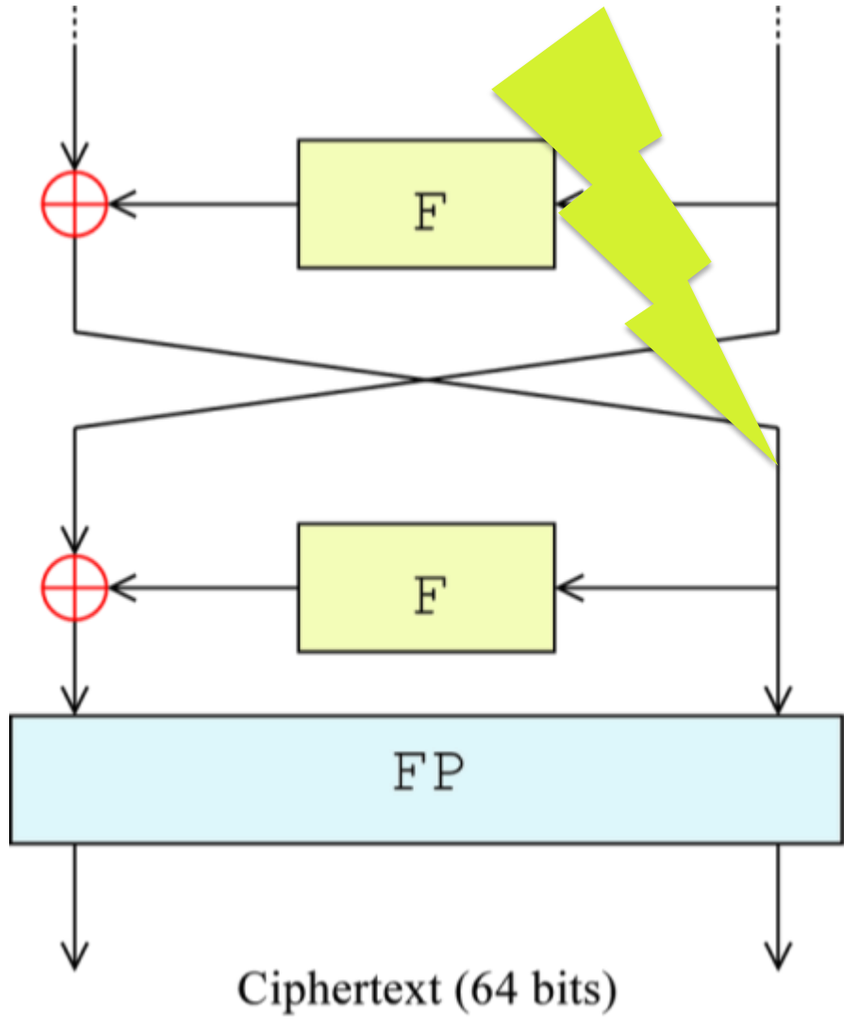
riscure

```
$ python angr_dfa
```

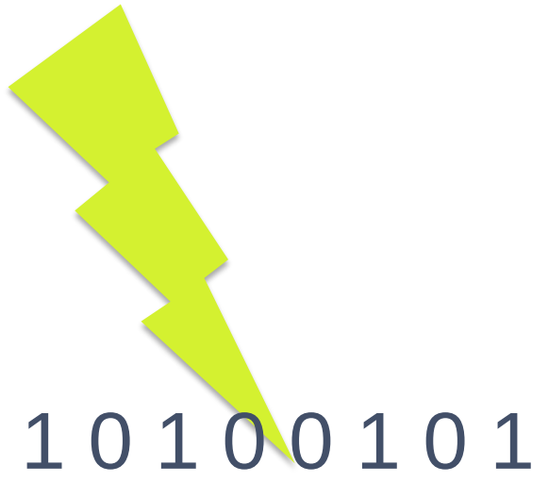


Success (but demo fail)

```
Faulty right:  0xac4b664fL
Faulty left:  0xf5d03e00L
Faulty out:  0x817e8c10a6ce6e62L
Creating state
Finding key
Try 1, key 0x3a22176eb7200L
Faulty right:  0x3f4891f9L
Faulty left:  0xf5d03e00L
Faulty out:  0xe62d6dea3d4d0dfL
Finding key
Try 2, key 0x18c21668752054L
Faulty right:  0x6537492L
Faulty left:  0xf5d03e00L
Faulty out:  0x1769582a418de220L
Finding key
Try 3, key 0x4e313f97dd0c6L
Correct!
BitflipsL: 0 BitflipsR: 16 Faults: 3 Duration: 11.043386s
```



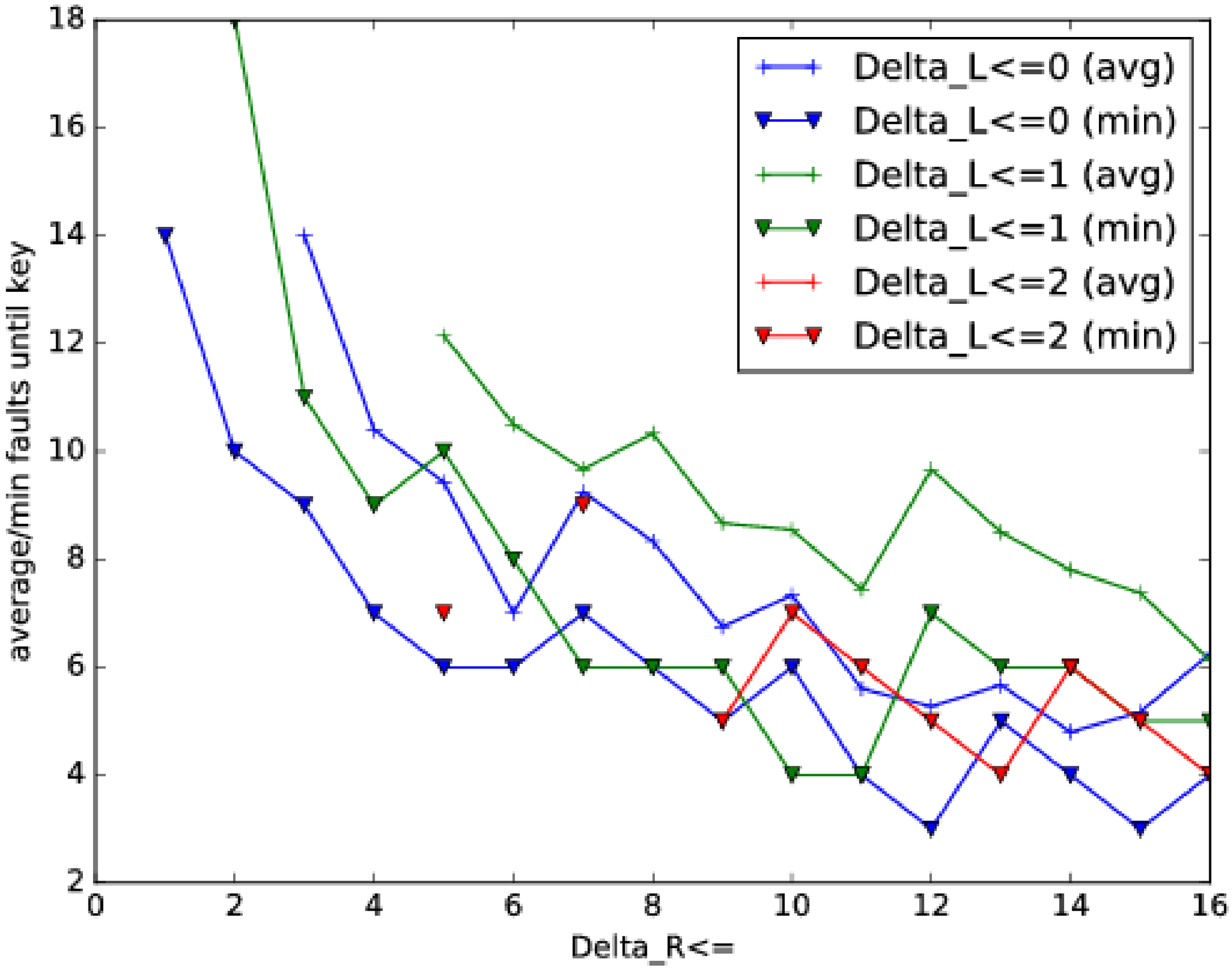
Playing with fault models



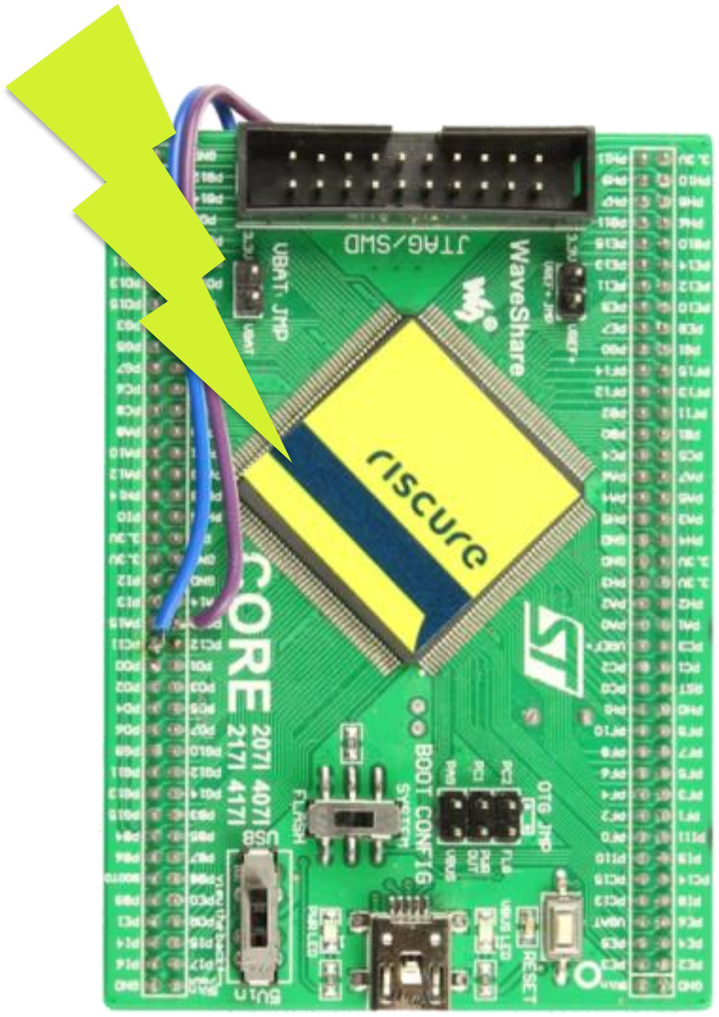
1 0 1 0 0 1 0 1

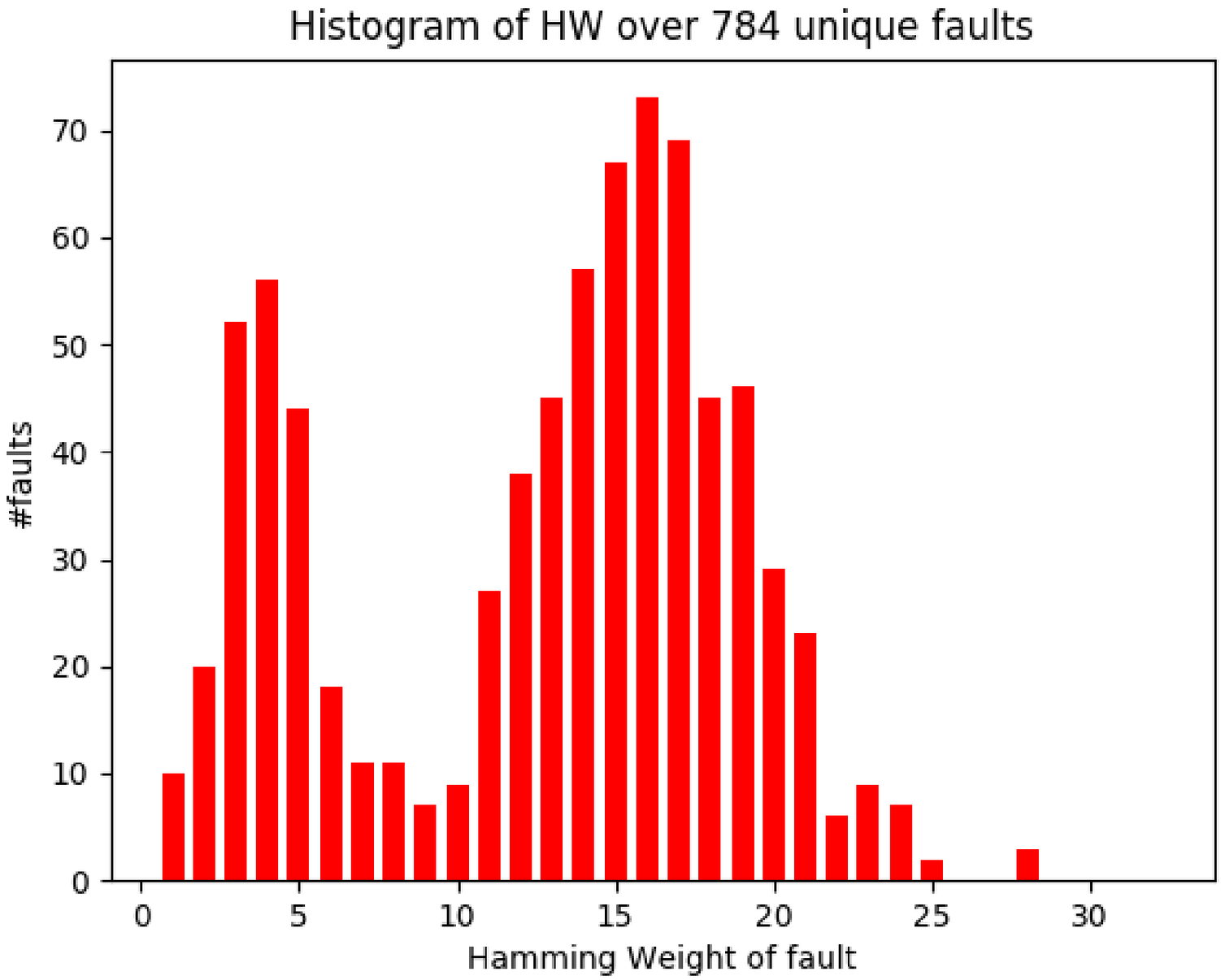
0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 1 0 0 1

DES fault models



Fault model on our test target





AES results

riscure

Δ	variant/fault	#runs	avg #secs per run (std dev)	avg #faults per run (std dev)
1	1/2	10	64 (23.2)	2.4 (0.7)
	2/2	20	9 (4.2)	2.1 (0.4)
	1/1	10	57 (19.0)	2.2 (0.4)
	2/1	20	7 (4.2)	2.1 (0.5)
2	1/2	10	4285 (1272.8)	2.7 (0.7)
	2/2	20	54 (29.6)	2.2 (0.5)
	1/1	10	3320 (567.3)	2.0 (0.0)
	2/1	10	43 (14.0)	2.0 (0.0)
3	1/2	DNF	DNF	DNF
	2/2	10	2585 (909.9)	2.3 (0.5)
	1/1	DNF	DNF	DNF
	2/1	10	2929 (648.6)	2 (0.0)
4	1/2	DNF	DNF	DNF
	2/2	10	21432 (6512.9)	2.4 (0.5)
	1/1	DNF	DNF	DNF
	2/1	10	23332 (5028.0)	2 (0.3)

Conclusions

- **On-par** or **fewer** faults than in literature
- Arbitrary fault models / ciphers
- Caveats:
  - Need LUT-free ciphers
  - Faults must fit fault model, or get UNSAT
  - Performance decreases significantly with # equations
- Future:
  - Extend to unknown ciphers / WBC
    - if we can (automatically?) convert tables into logic
  - Use bucketing and statistics to avoid UNSAT and slowness

Thanks to Cees-Bart Breunesse, Rajesh Velegalati, Sergio Gonzalez, Panasayya Yalla, and Angr people!

riscure

Challenge your security

Contact: Jasper van Woudenberg @jzvw
vanwoudenberg@na.riscure.com

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90

www.riscure.com

Riscure North America
550 Kearny St.
Suite 330
San Francisco, CA 94108
+1 (650) 646 9979

inforequest@riscure.com