SIFA on Nonce-based Authenticated Encryption: When does it fail? Application to Ascon

Viet-Sang Nguyen

joint work with Vincent Grosso and Pierre-Louis Cayrel

FDTC
Kuala Lumpur, 14 September 2025

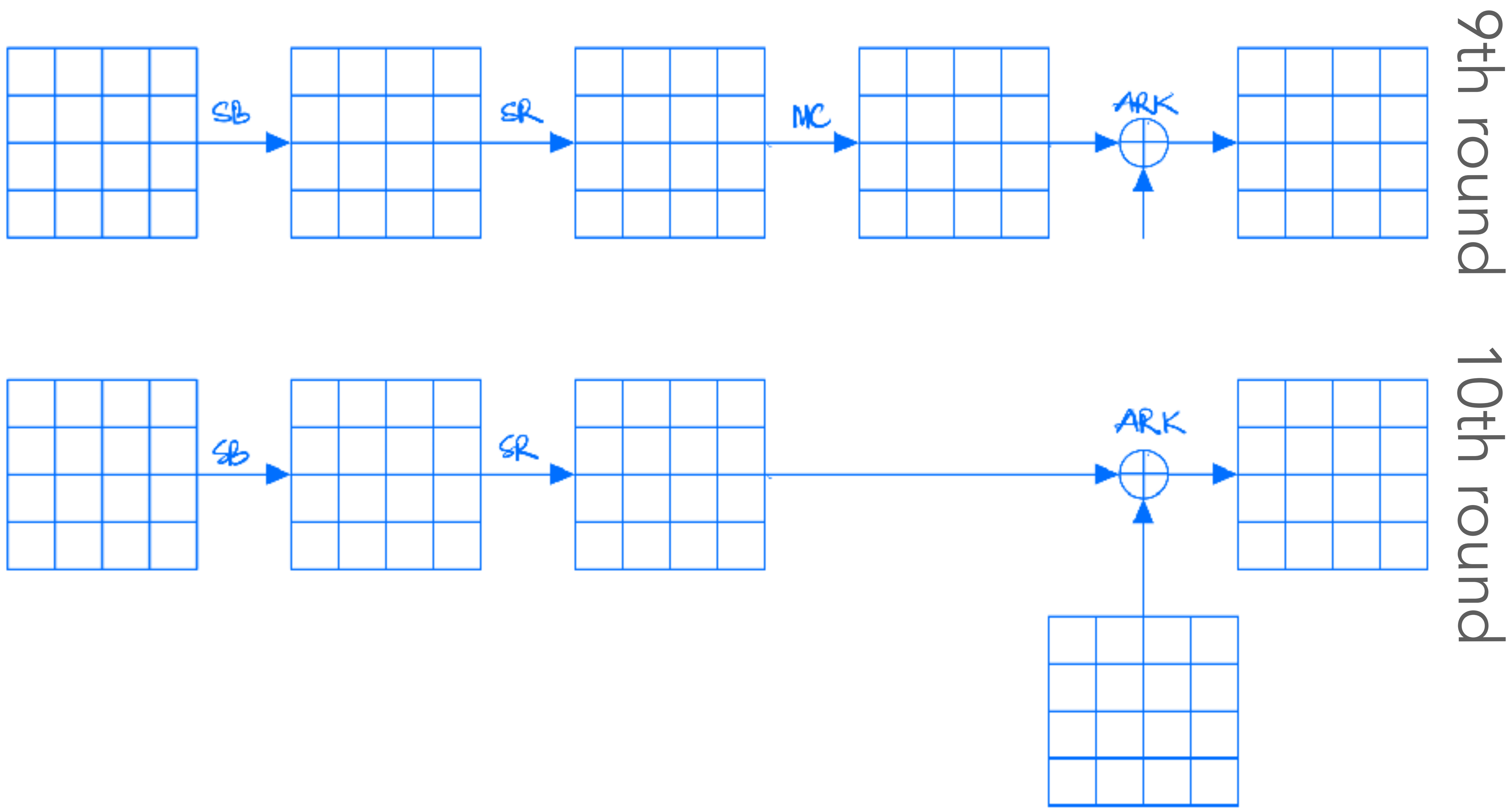
UNIVERSITÉ DE LYON
UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE
Laboratoire Hubert Curien UMR • CNRS • 5516 • Saint-Étienne

anr
PROPHY ANR-22-CE39-0008-01

SIFA

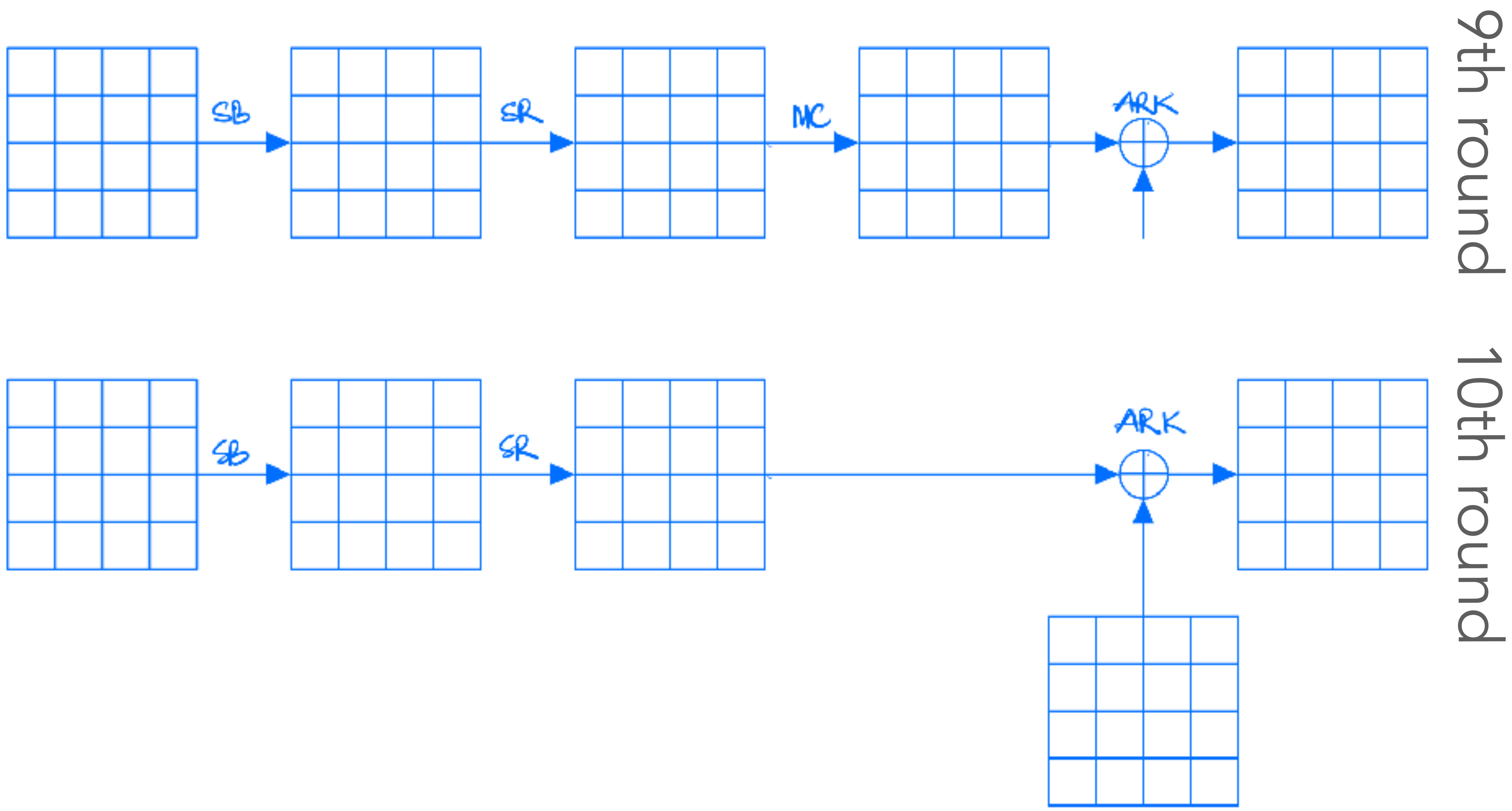
Statistical Ineffective Fault Attack

Example on AES



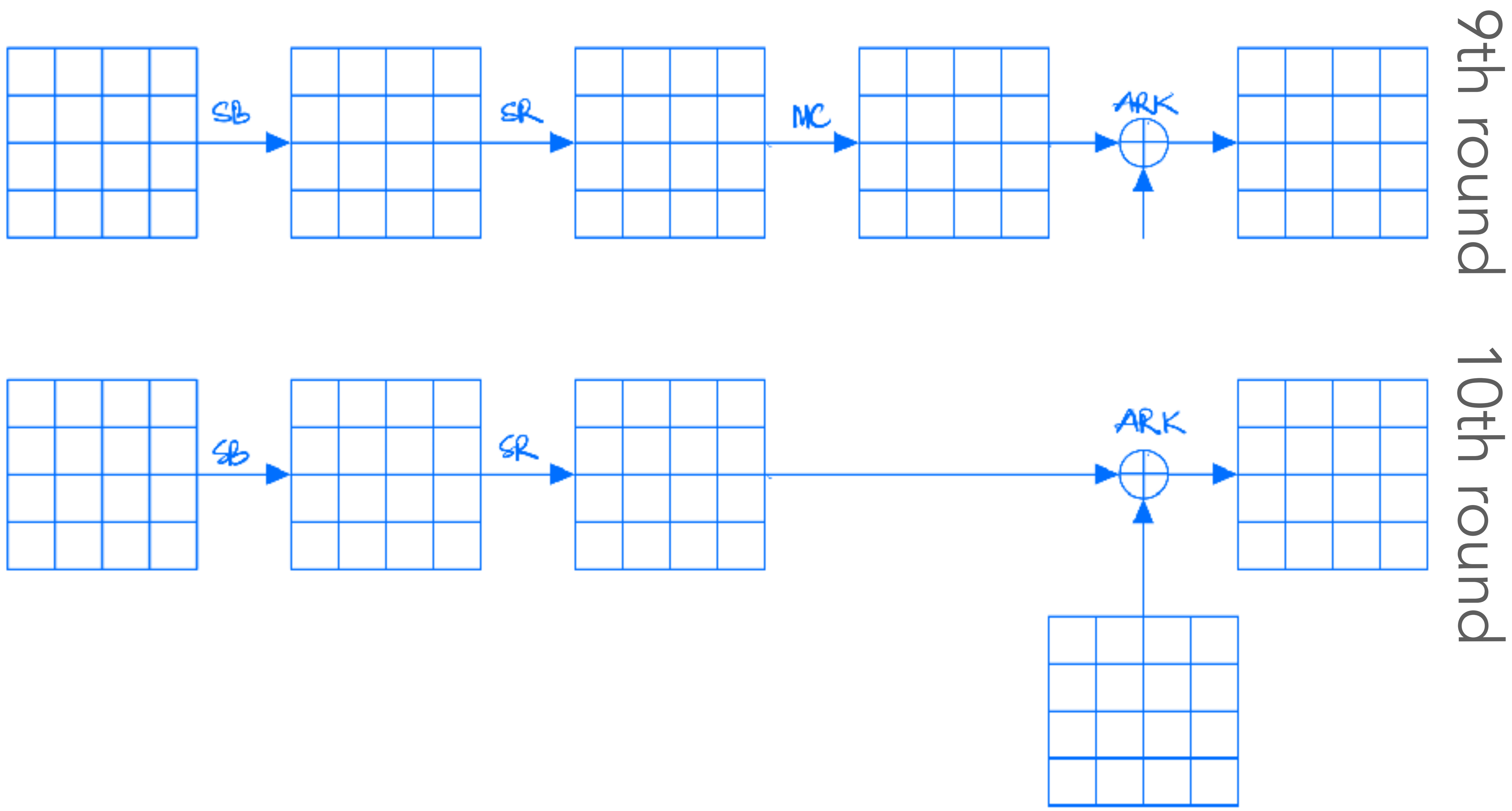
Example on AES

- Choose an intermediate value v



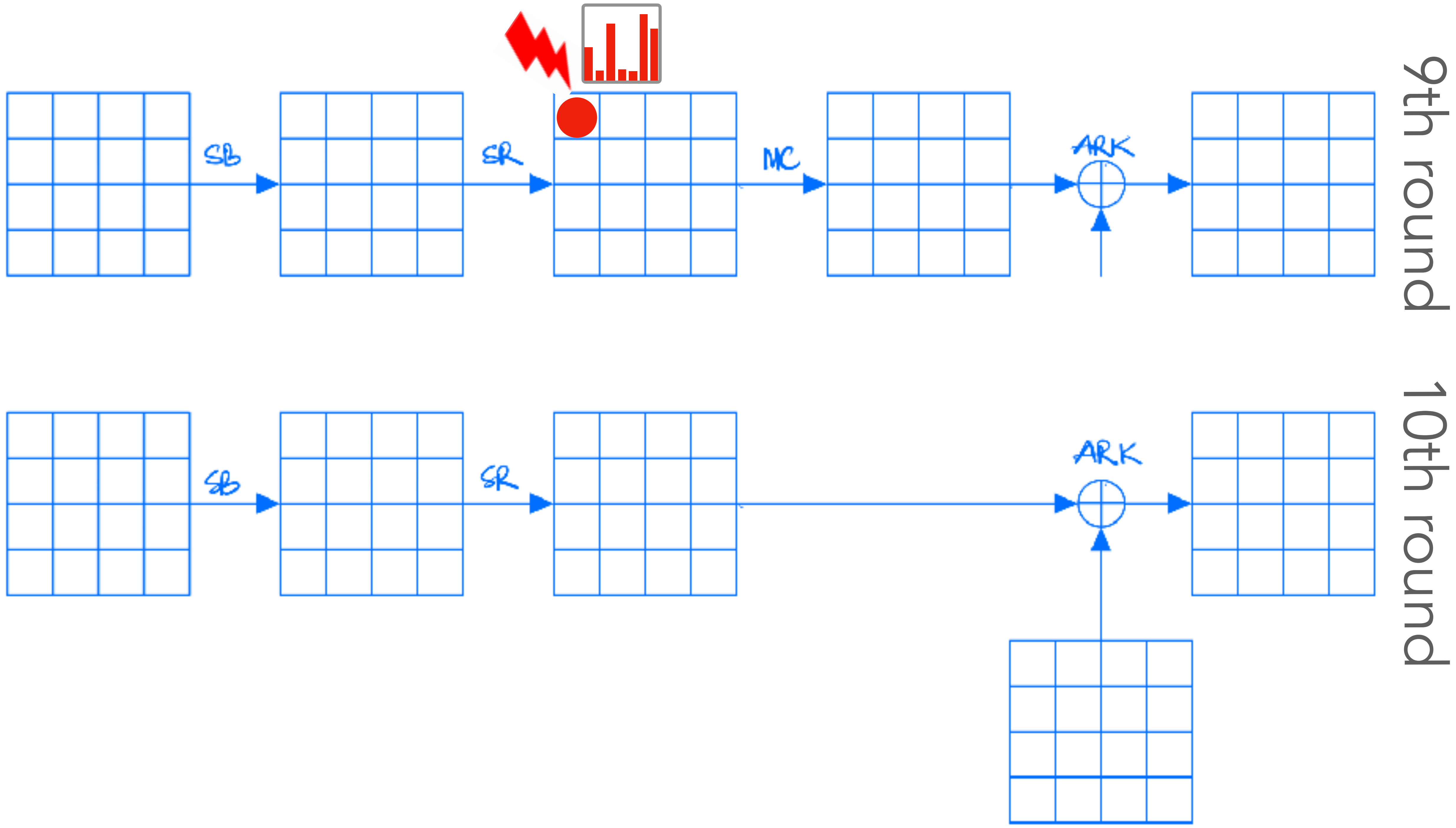
Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)



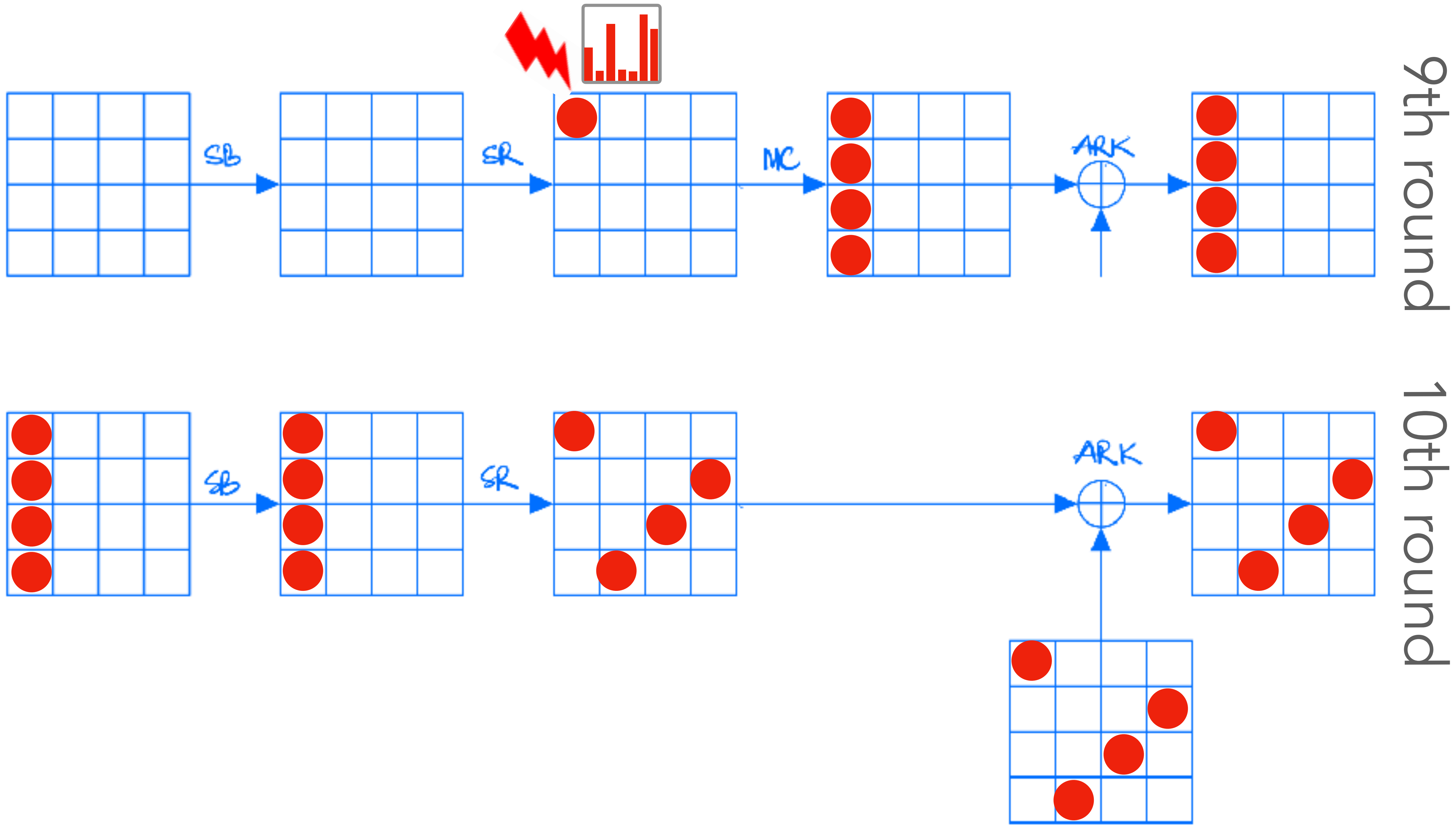
Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)



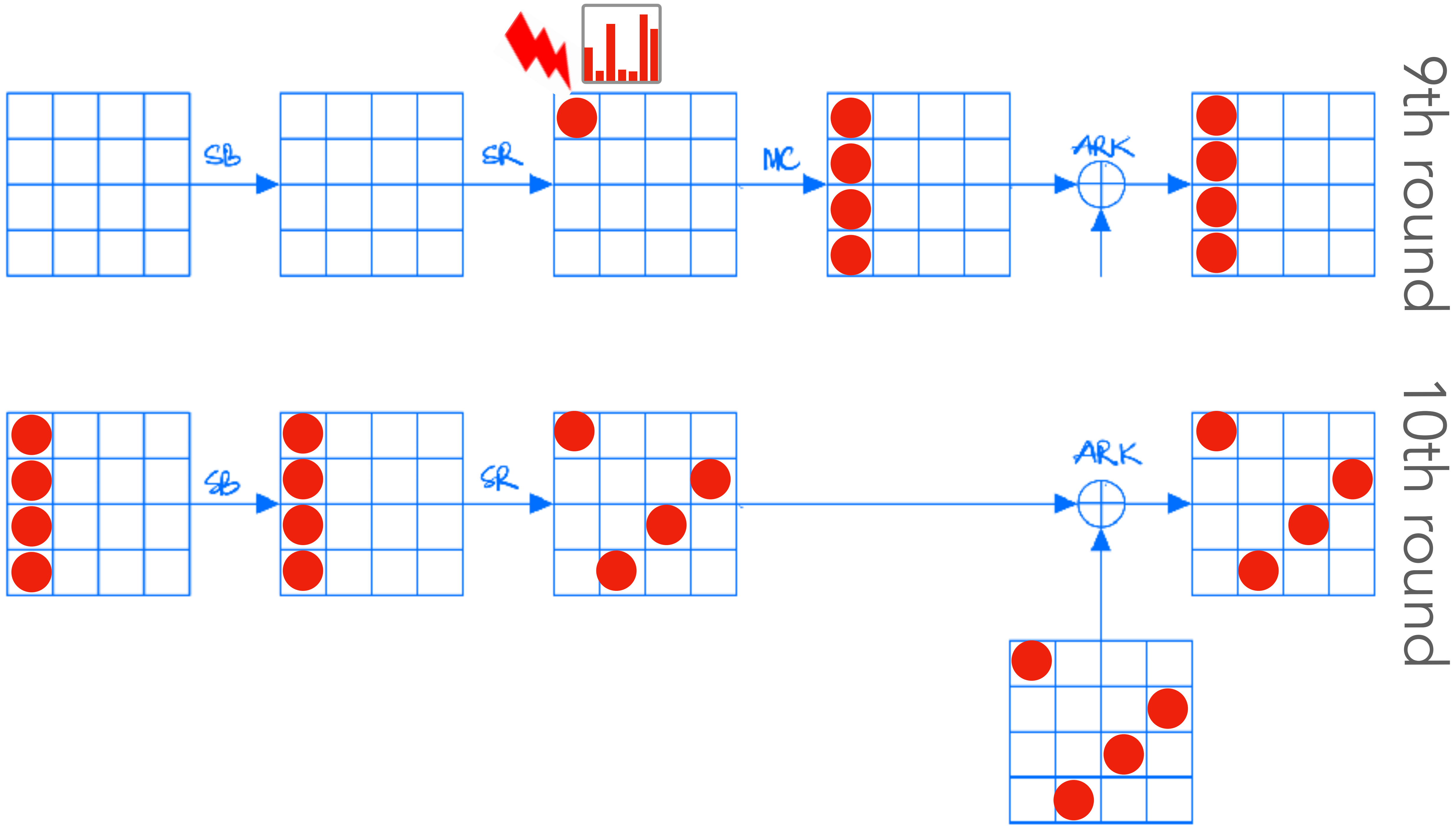
Example on AES

- ✦ Choose an intermediate value v
- ✦ Force v biased by faults (e.g., stuck-at-0)



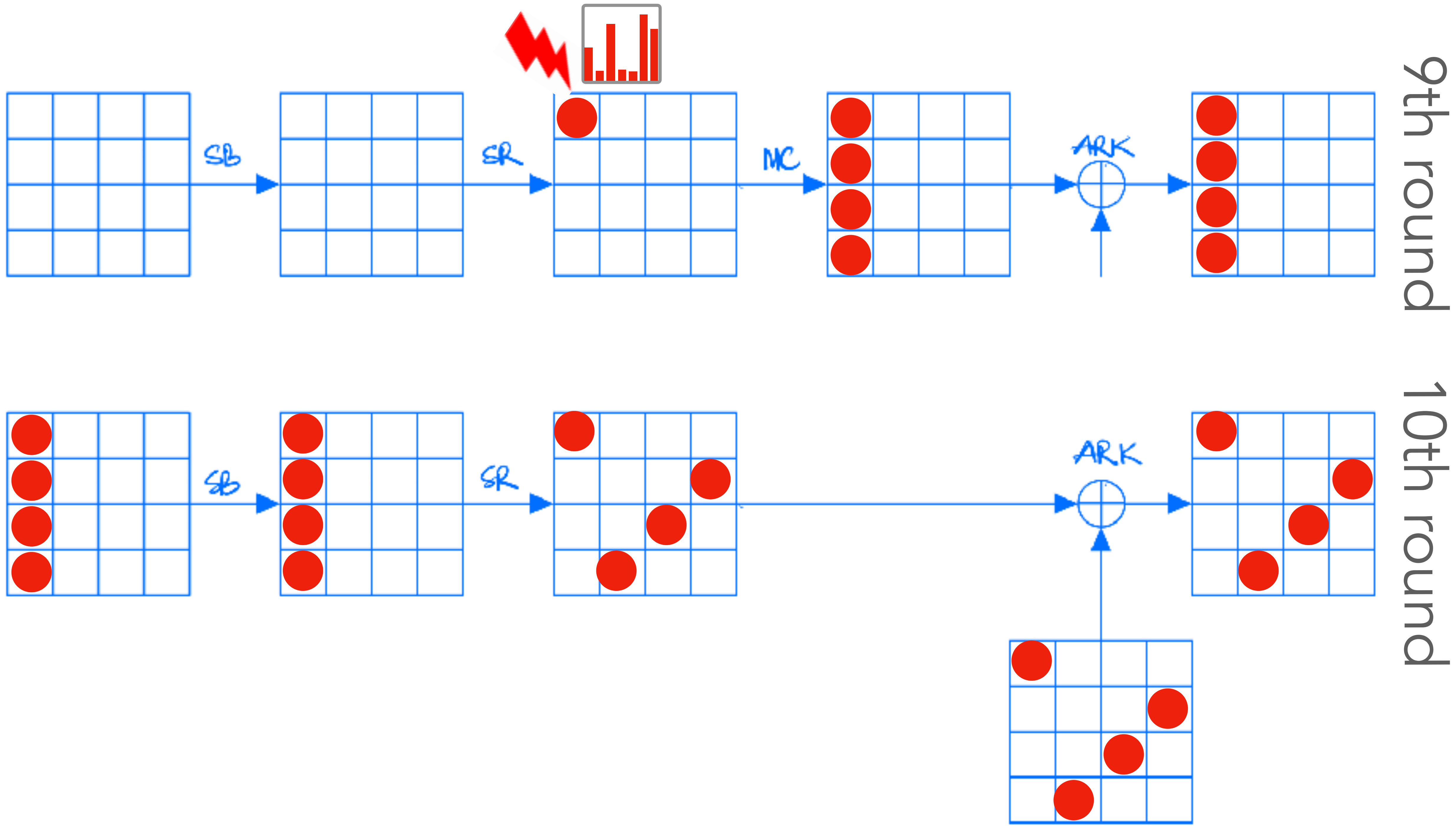
Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts



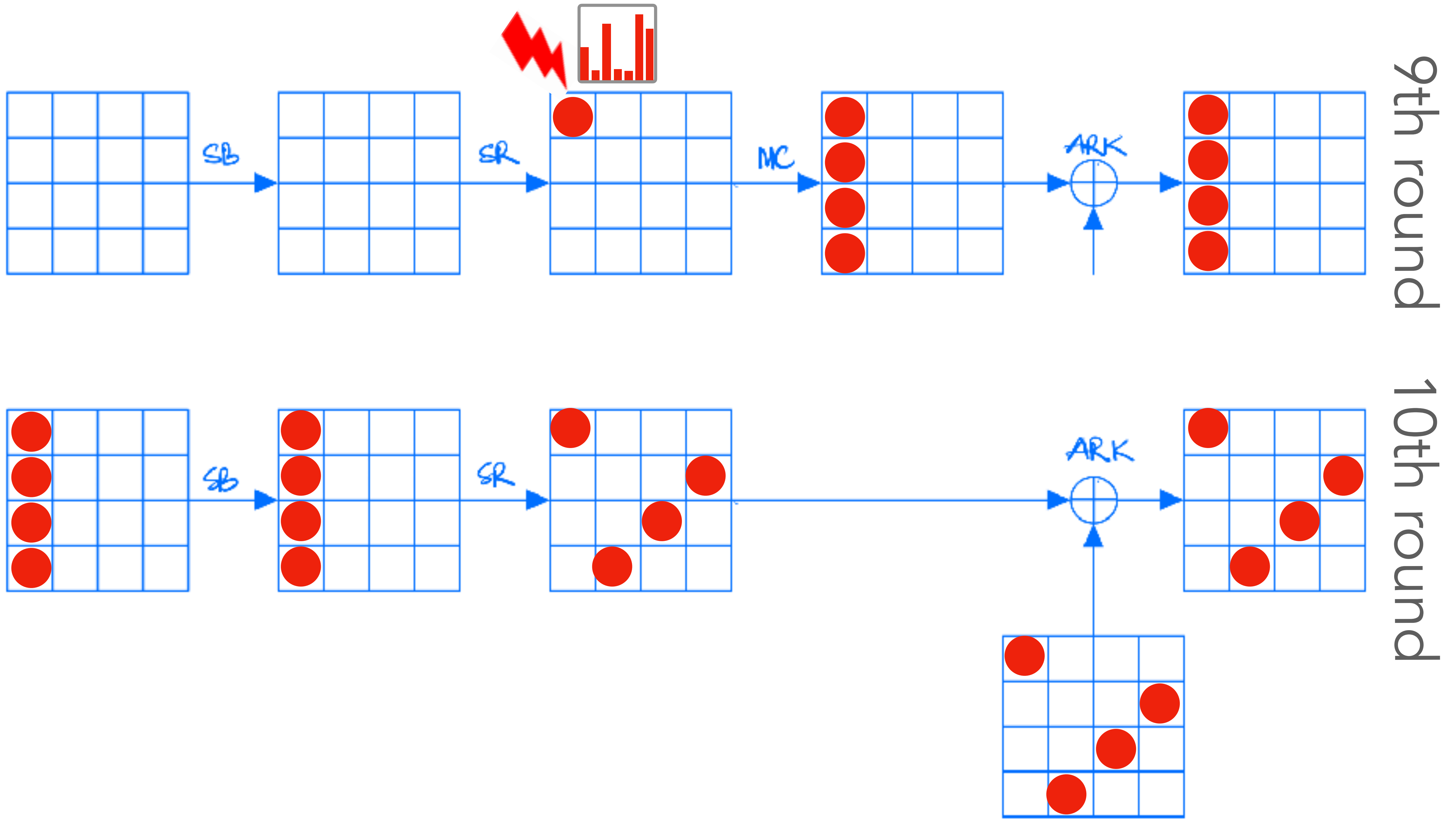
Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery: 😈
  - Guess 4 key bytes, compute v

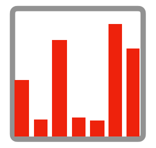
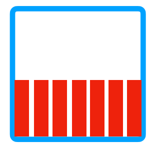


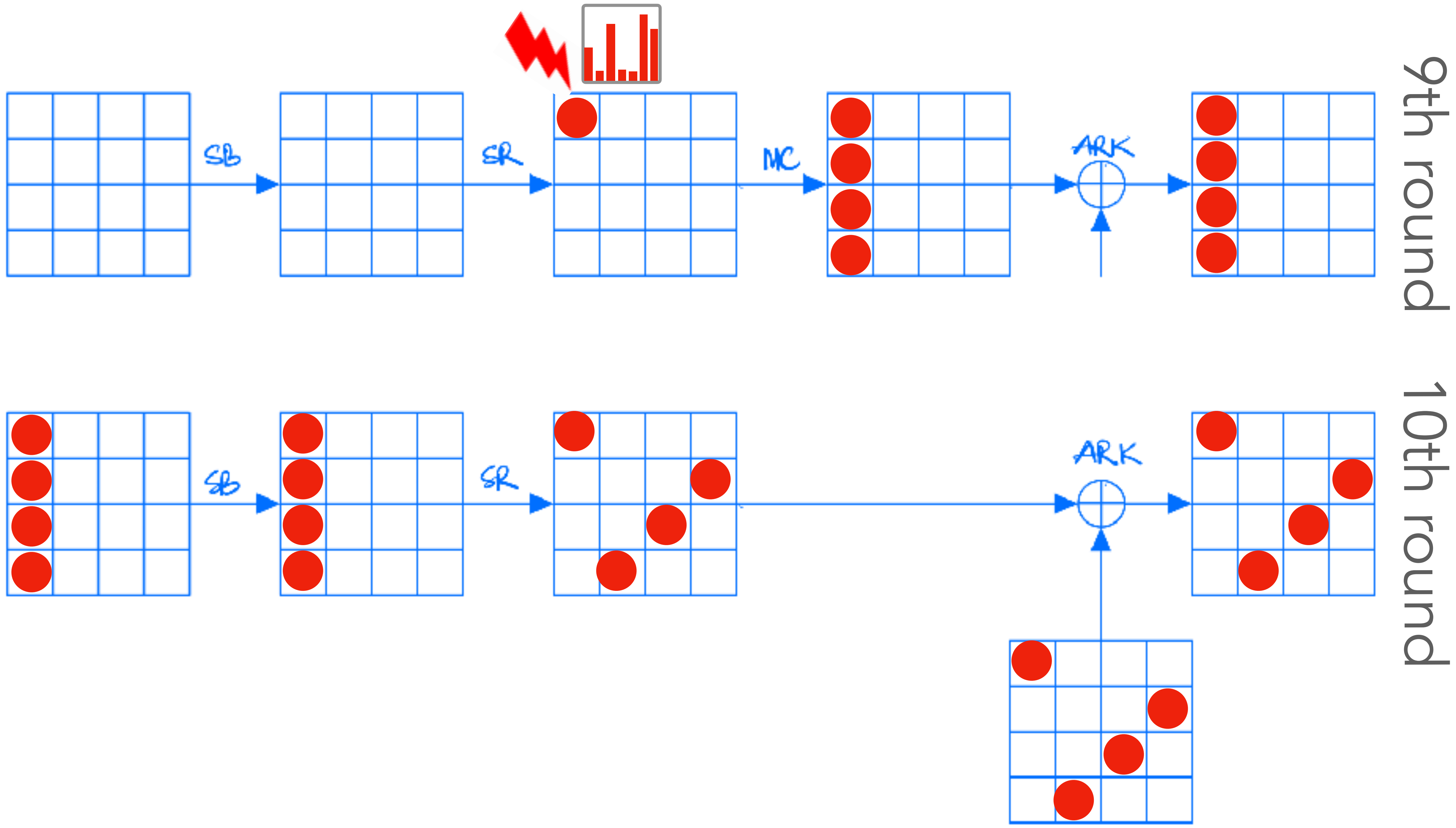
Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery: 😈
  - Guess 4 key bytes, compute v
  - If correct key guess, v is biased
  - If wrong key guess, v is uniform



Example on AES

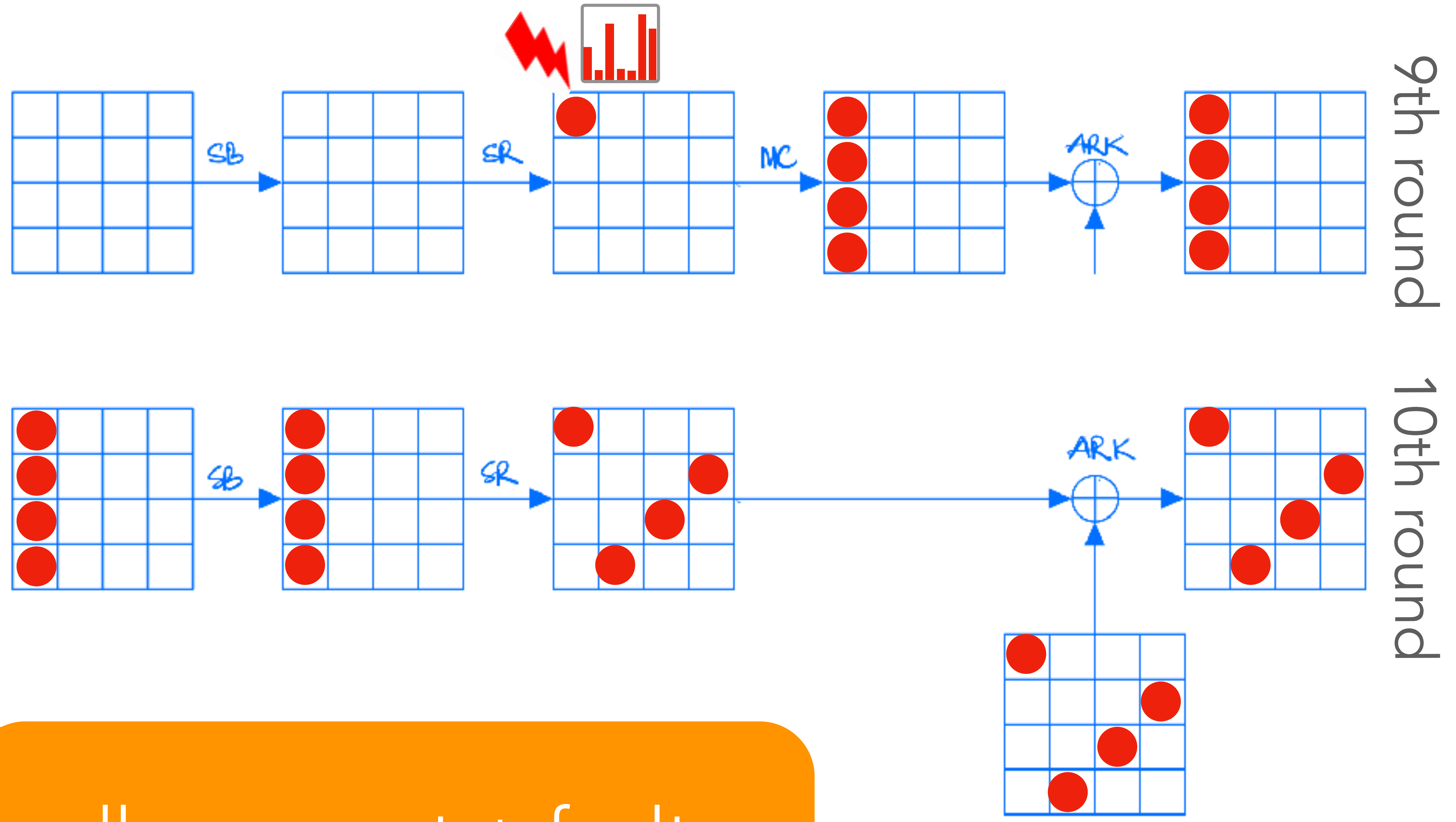
- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery: 😈
  - Guess 4 key bytes, compute v
  - If correct key guess, v is biased 
  - If wrong key guess, v is uniform 



Statistical Fault Attack (SFA) by Fuhr et al., 2013

Example on AES

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery: 😈
  - Guess 4 key bytes, compute v
  - If correct key guess, v is biased
  - If wrong key guess, v is uniform

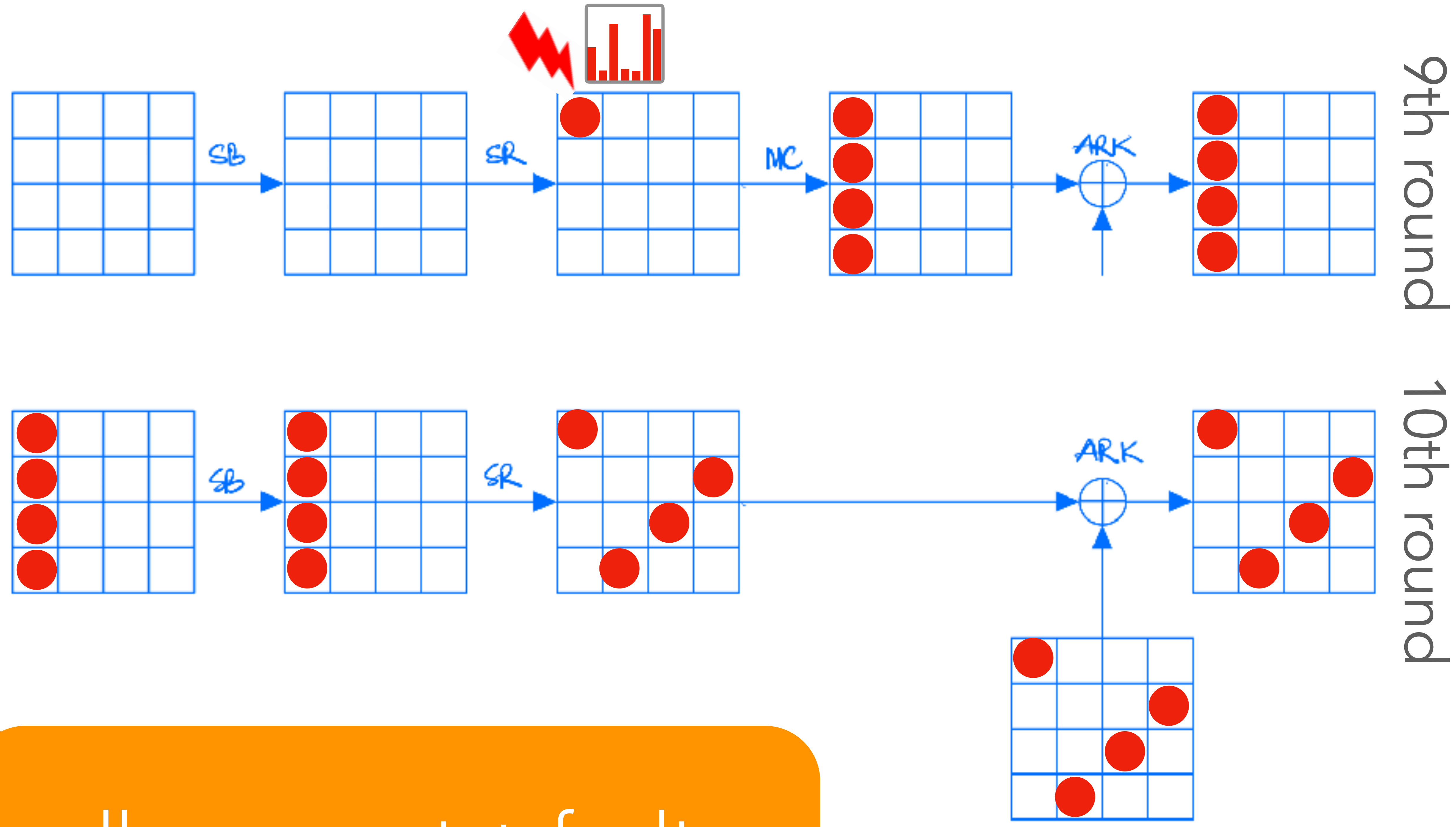


all = correct + faulty

Statistical Fault Attack (SFA) by Fuhr et al., 2013

- ✦ Choose an intermediate value v
- ✦ Force v biased by faults (e.g., stuck-at-0)
- ✦ Collect a number of ciphertexts

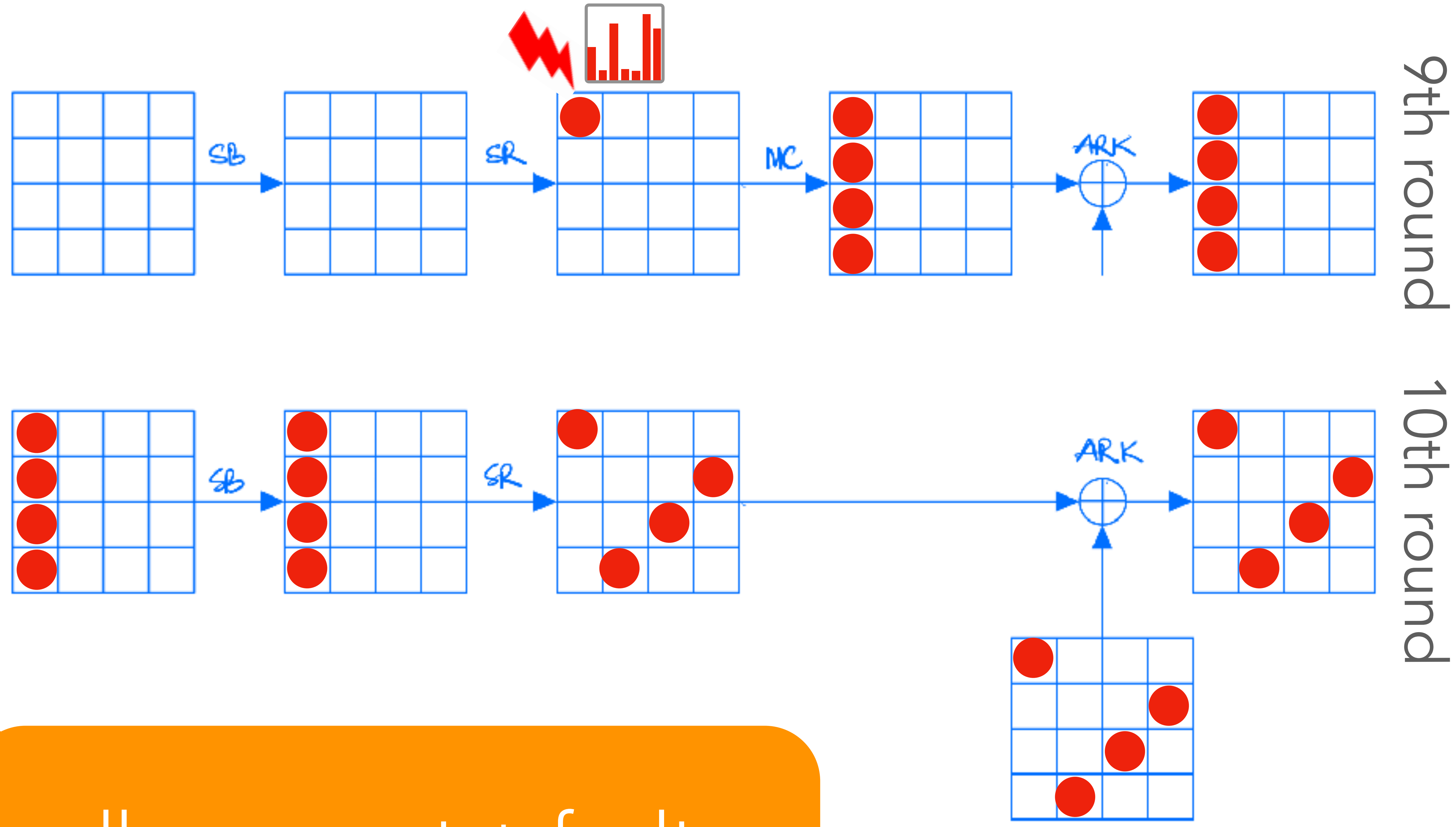
all = correct + faulty



When a countermeasure is used...

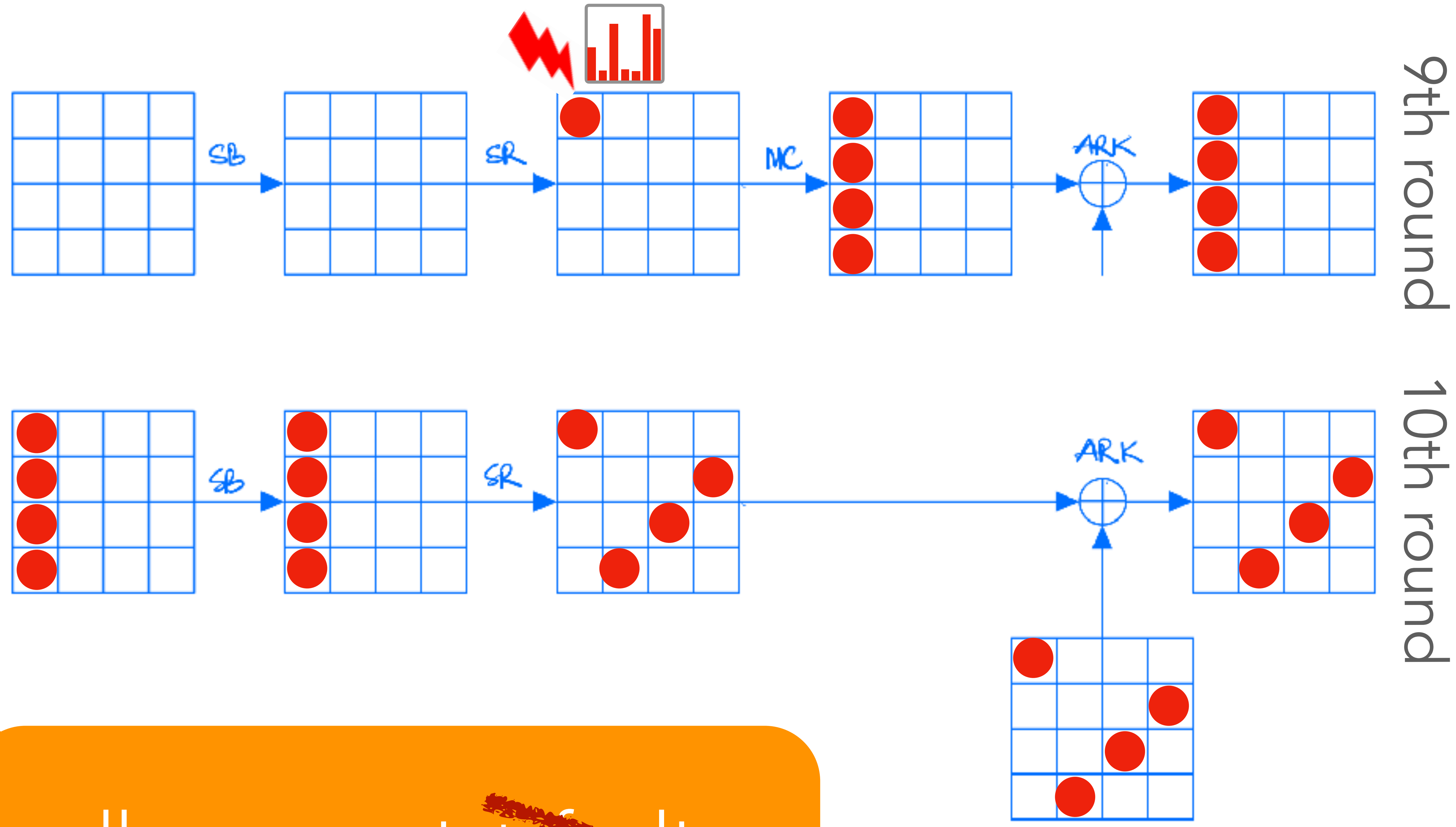
- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts

all = correct + faulty



When a countermeasure is used...

- Choose an intermediate value v
- Force v **biased** by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts

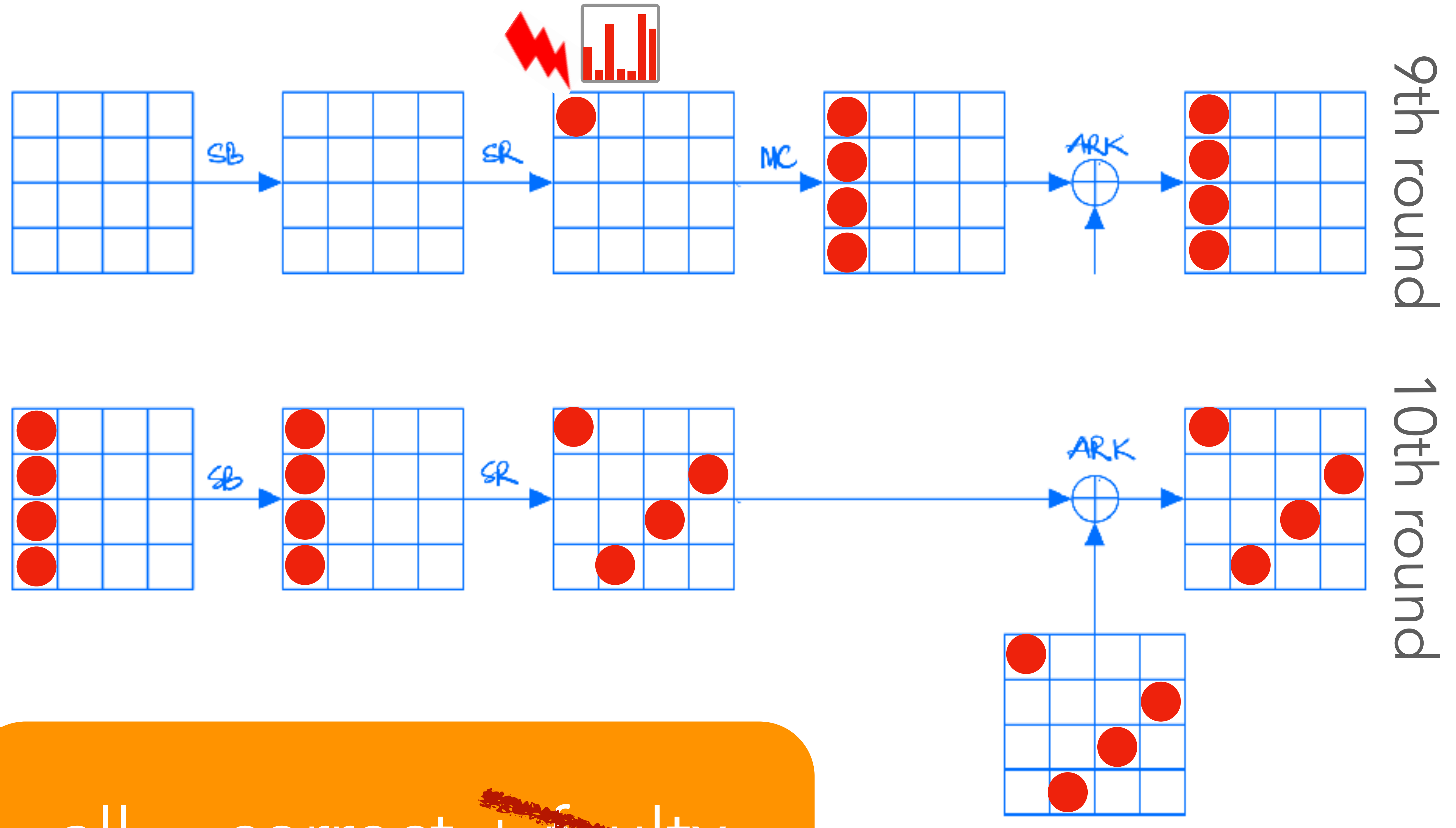


all = correct + ~~faulty~~

When a countermeasure is used...

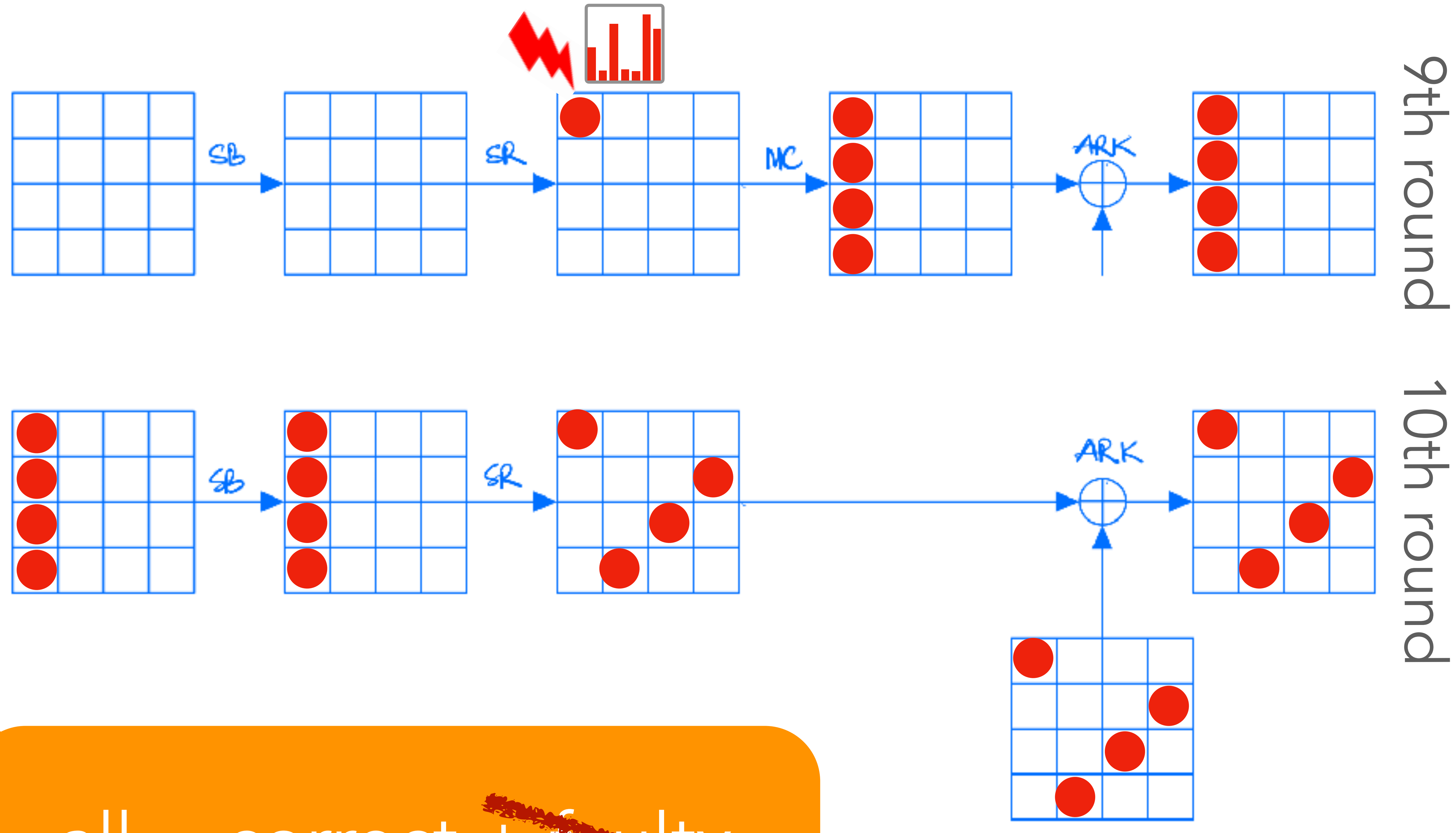
- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery (still works): 😈
  - Guess 4 key bytes, compute v
  - If correct key guess, v is biased
  - If wrong key guess, v is uniform

all = correct + ~~faulty~~



When a countermeasure is used...

- Choose an intermediate value v
- Force v biased by faults (e.g., stuck-at-0)
- Collect a number of ciphertexts
- Key recovery (still works): 😈
  - Guess 4 key bytes, compute v
  - If correct key guess, v is biased
  - If wrong key guess, v is uniform



all = correct + ~~faulty~~

Statistical Ineffective Fault Attack (SIFA)
by Dobraunig et al., 2018

SIFA on Nonce-based Authenticated Encryption

Dobraunig et al., SAC 2018

Fault Attacks on Nonce-based Authenticated Encryption: Application to Keyak and Ketje

Christoph Dobraunig[1], Stefan Mangard[1], Florian Mendel[2], and Robert Primas[1]

[1] Graz University of Technology, Austria
first.last@iaik.tugraz.at
[2] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Dobraunig et al., SAC 2018

- Proposed generic strategy to apply SIFA on Nonce-based AE

Fault Attacks on Nonce-based Authenticated Encryption: Application to Keyak and Ketje

Christoph Dobraunig[1], Stefan Mangard[1], Florian Mendel[2], and Robert Primas[1]

[1] Graz University of Technology, Austria
first.last@iaik.tugraz.at
[2] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Dobraunig et al., SAC 2018

- Proposed generic strategy to apply SIFA on Nonce-based AE
- Demonstrated on 2 ciphers: Kayak and Ketje

Fault Attacks on Nonce-based Authenticated Encryption: Application to Keyak and Ketje

Christoph Dobraunig[1], Stefan Mangard[1], Florian Mendel[2], and Robert Primas[1]

[1] Graz University of Technology, Austria
first.last@iaik.tugraz.at
[2] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Dobraunig et al., SAC 2018

- Proposed generic strategy to apply SIFA on Nonce-based AE

- Demonstrated on 2 ciphers: Kayak and Ketje

- Conjectured that this attack strategy is applicable to Ascon (new standard)

Fault Attacks on Nonce-based Authenticated Encryption: Application to Keyak and Ketje

Christoph Dobraunig[1], Stefan Mangard[1], Florian Mendel[2], and Robert Primas[1]

[1] Graz University of Technology, Austria
first.last@iaik.tugraz.at
[2] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Dobraunig et al., SAC 2018

- Proposed generic strategy to apply SIFA on Nonce-based AE
- Demonstrated on 2 ciphers: Kayak and Ketje
- Conjectured that this attack strategy is applicable to Ascon (new standard)

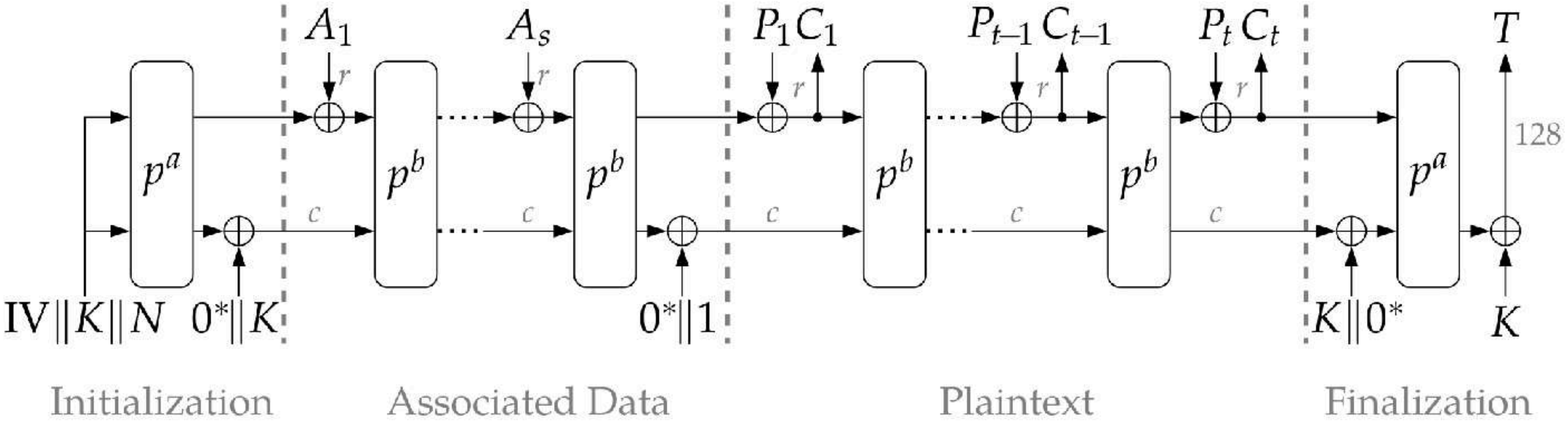
Fault Attacks on Nonce-based Authenticated Encryption: Application to Keyak and Ketje

Christoph Dobraunig[1], Stefan Mangard[1], Florian Mendel[2], and Robert Primas[1]

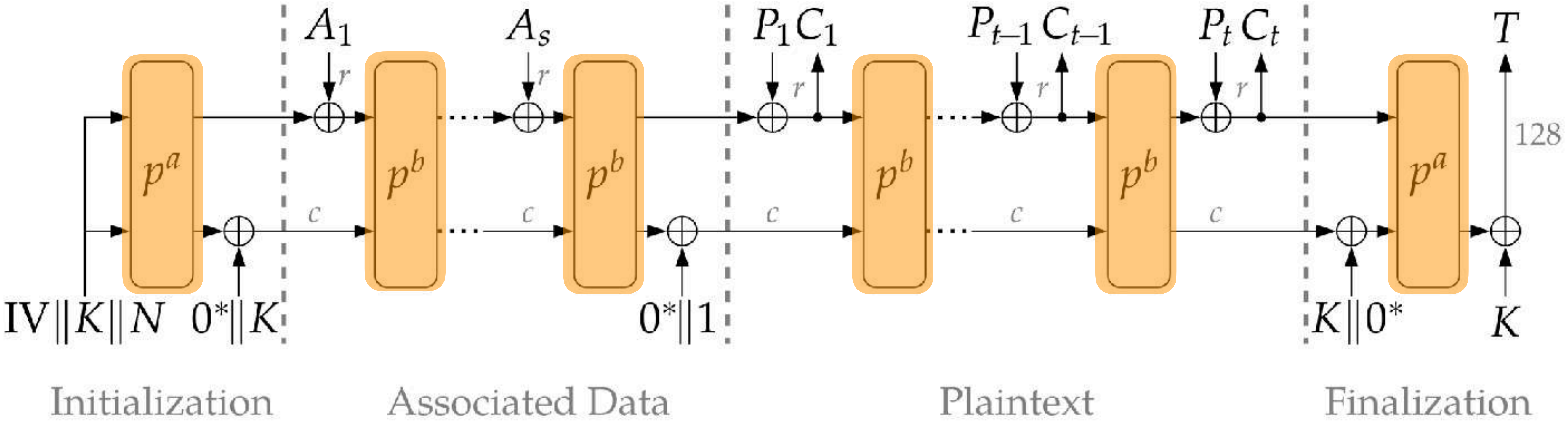
[1] Graz University of Technology, Austria
first.last@iaik.tugraz.at
[2] Infineon Technologies AG, Germany
florian.mendel@infineon.com

Hum, let's try ! 😈

Ascon

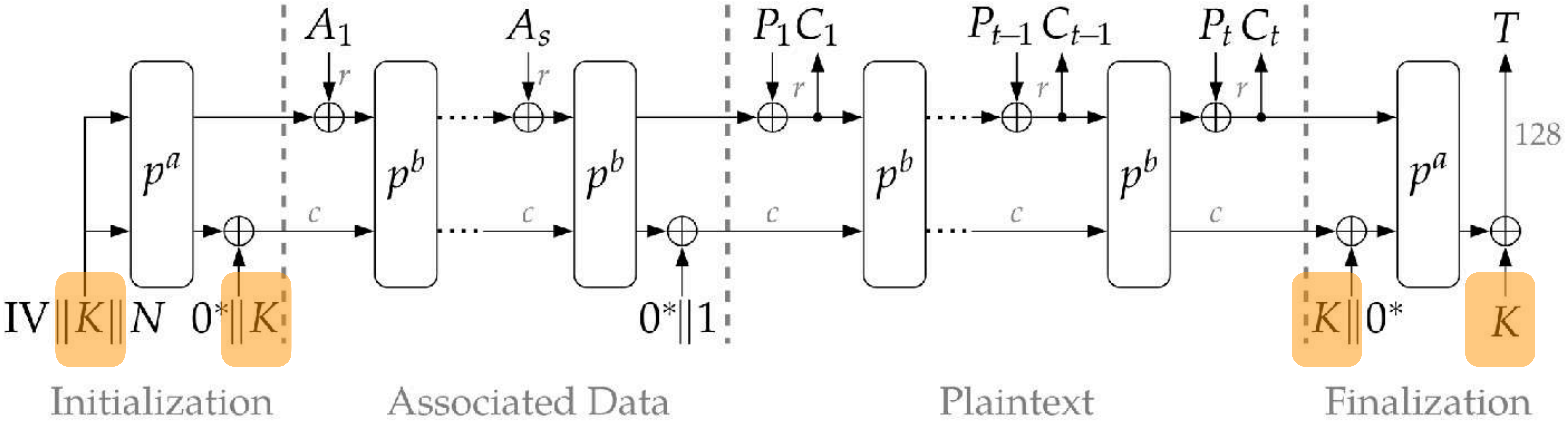

A_1
A_s
P_1 C_1
P_{t-1} C_{t-1}
P_t C_t
T
r
c
128
p^a
p^b
$IV \| K \| N$
$0^* \| K$
$0^* \| 1$
$K \| 0^*$
K
Initialization
Associated Data
Plaintext
Finalization

Permutation blocks



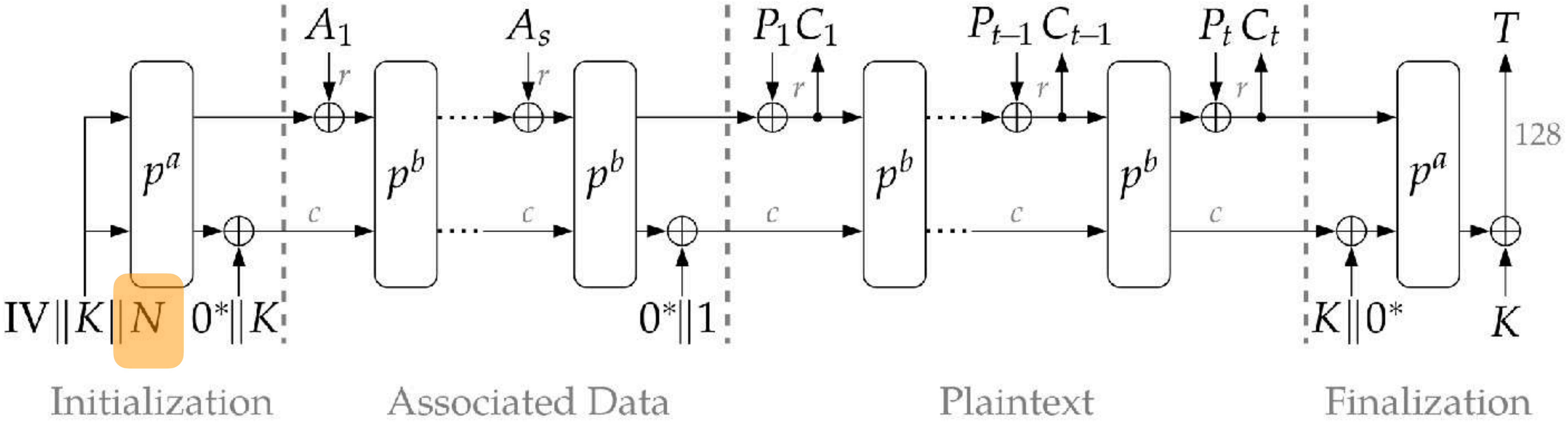
- p^a: 12 permutation rounds ($a = 12$)
- p^b: 8 permutation rounds ($b = 8$)

Key (128 bits)

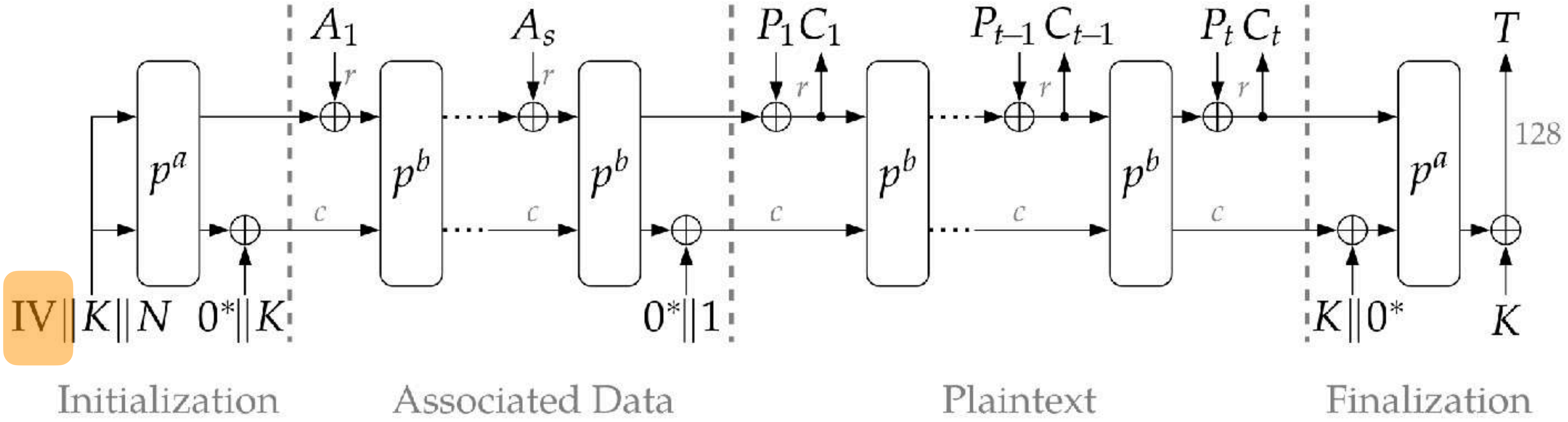


A_1
A_s
$P_1\ C_1$
$P_{t-1}\ C_{t-1}$
$P_t\ C_t$
T
r
c
p^a
p^b
128
$IV \| K \| N$
$0^* \| K$
$0^* \| 1$
$K \| 0^*$
K
Initialization
Associated Data
Plaintext
Finalization

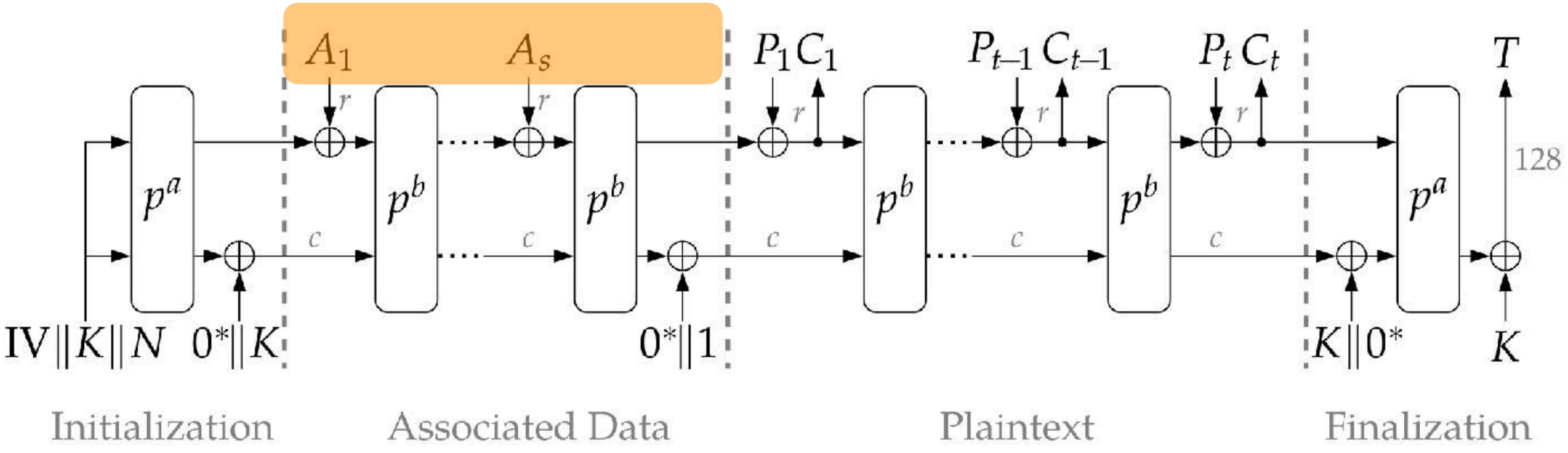
Nonce (128 bits)



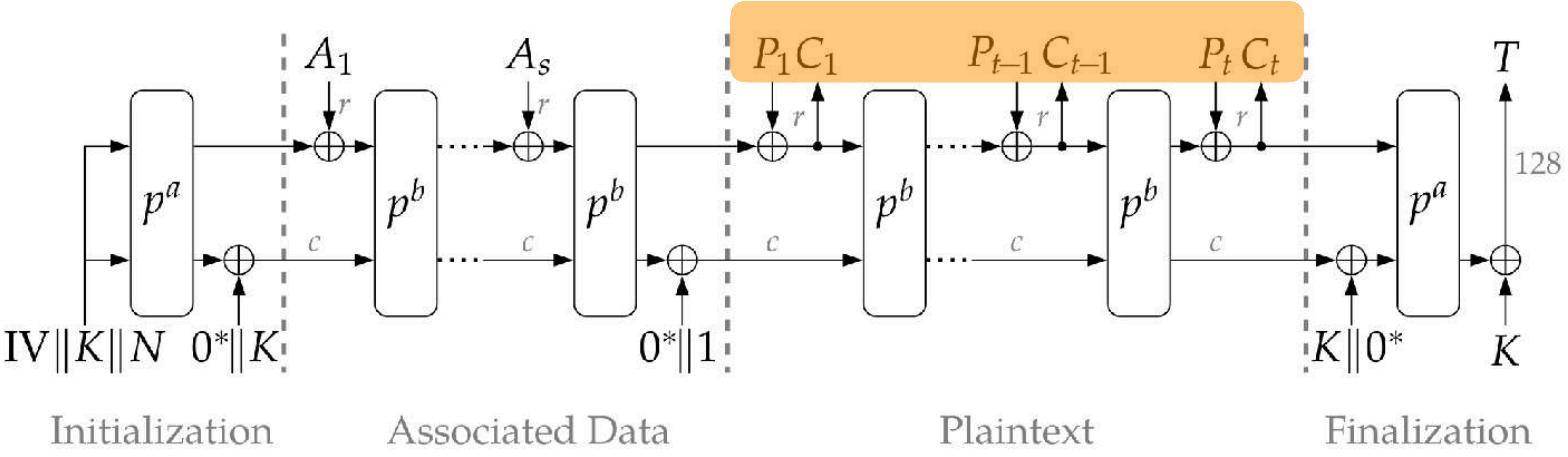
Initialization vector



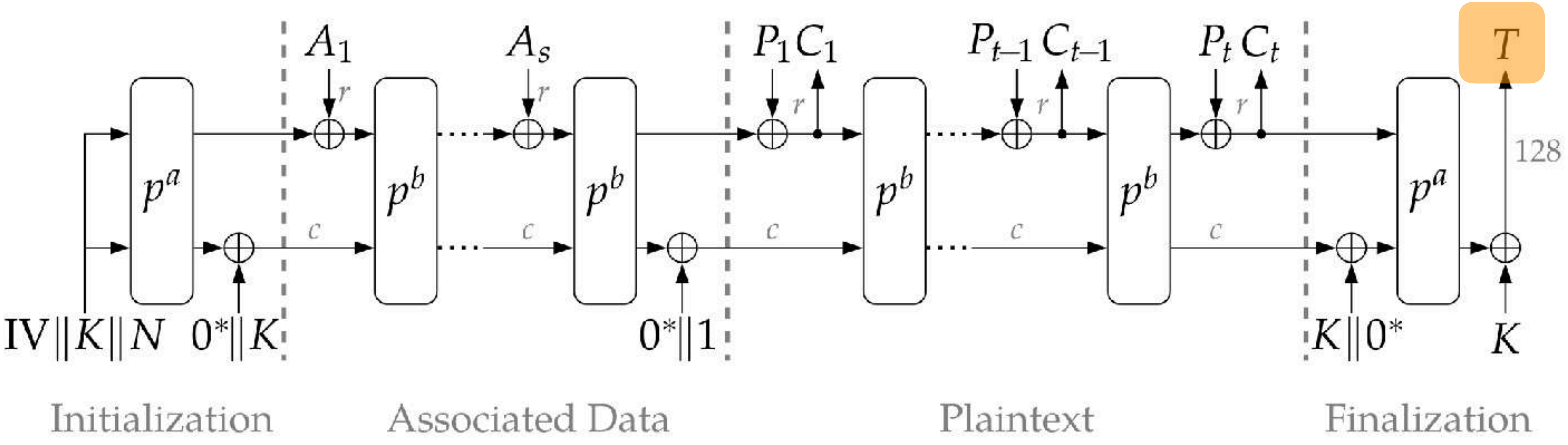
Associated data

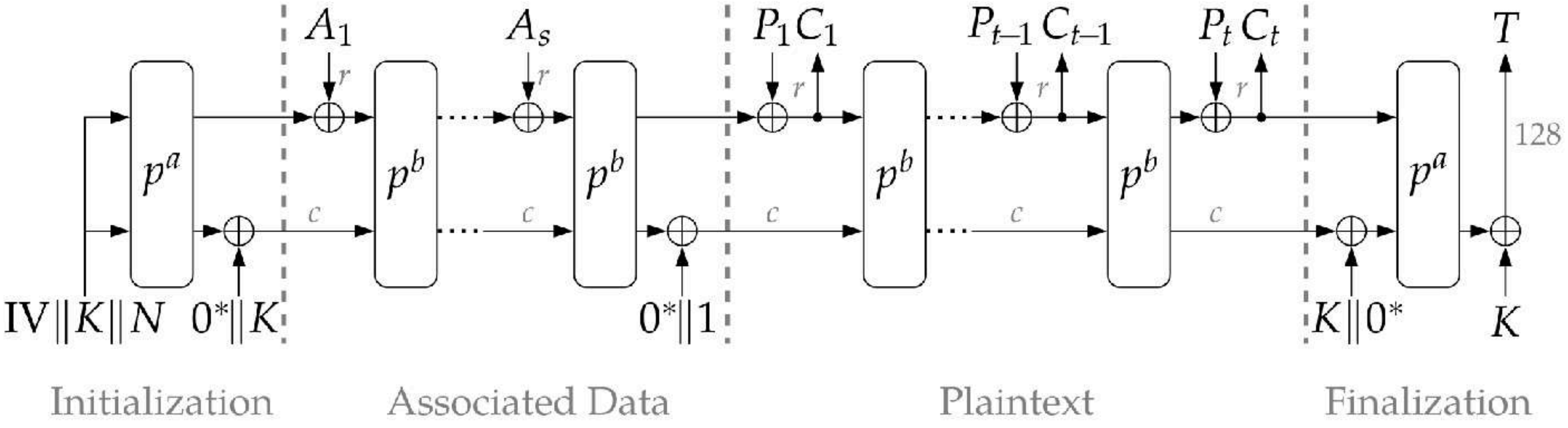


Plaintext / Ciphertext

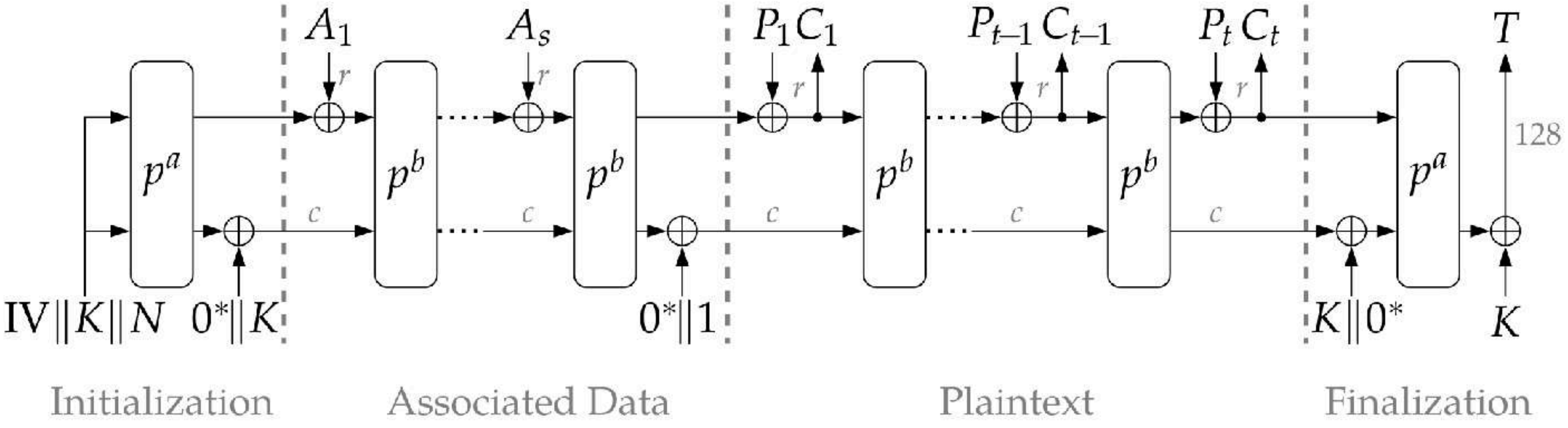


Verification tag

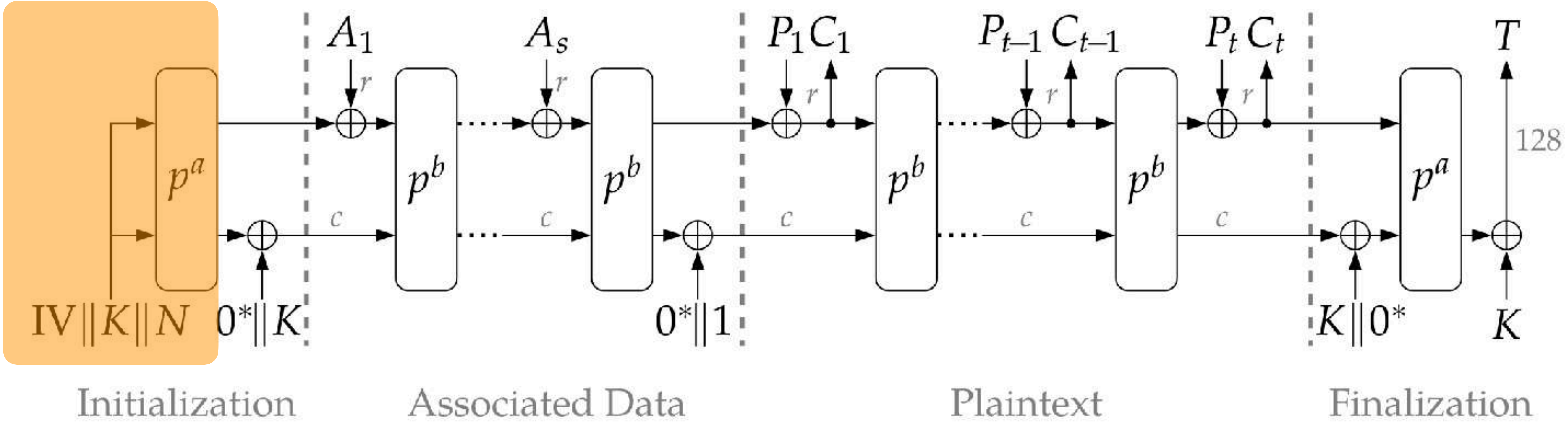



A_1
A_s
P_1 C_1
P_{t-1} C_{t-1}
P_t C_t
T
r
c
128
p^a
p^b
$IV\|K\|N$
$0^*\|K$
$0^*\|1$
$K\|0^*$
K
Initialization
Associated Data
Plaintext
Finalization

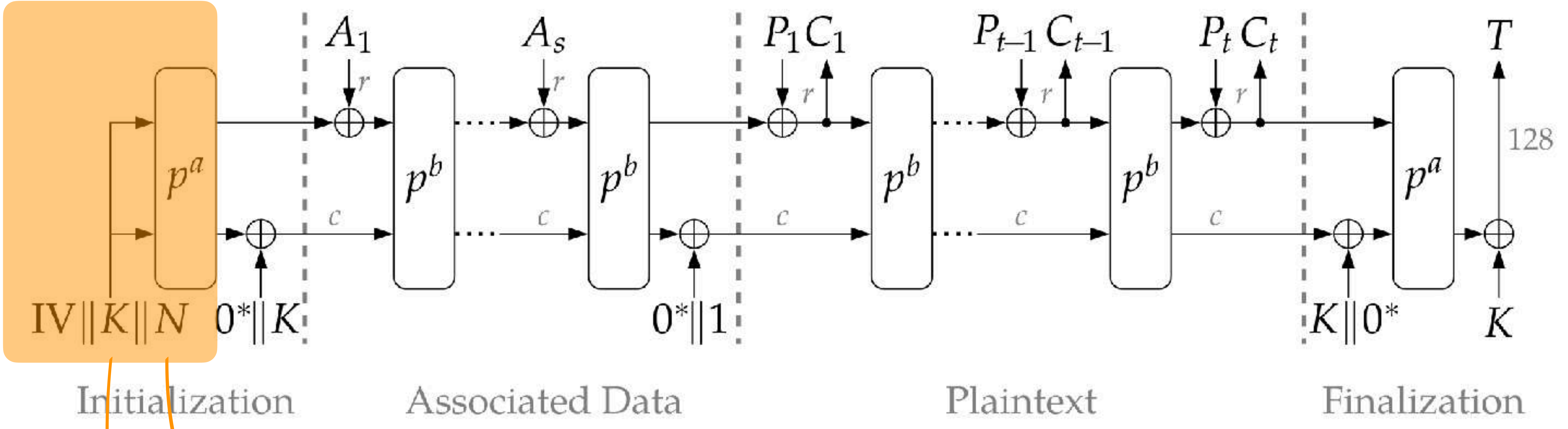
Focus on 1st round of initialization



Focus on 1st round of initialization



Focus on 1st round of initialization



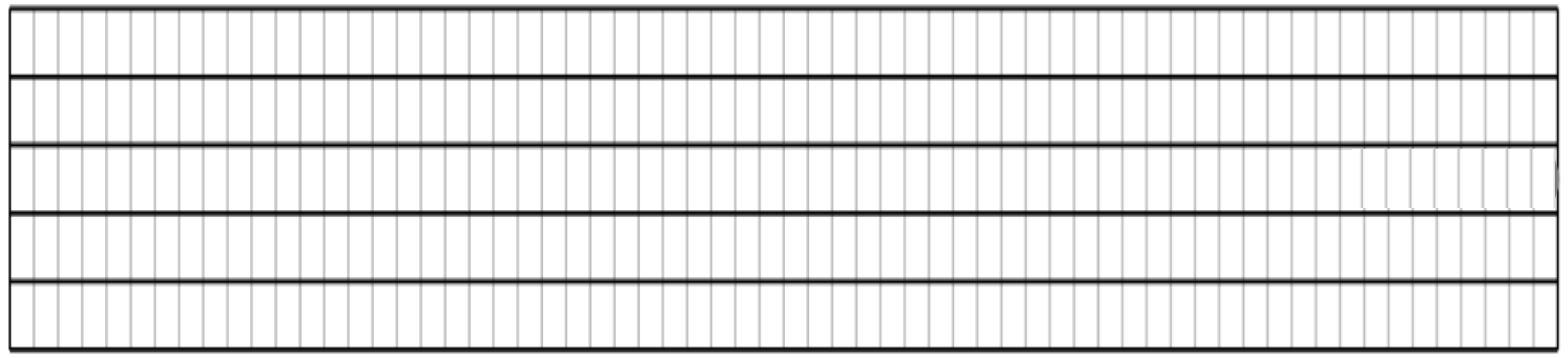
Control the nonce N

Guess the key K

1st round computation

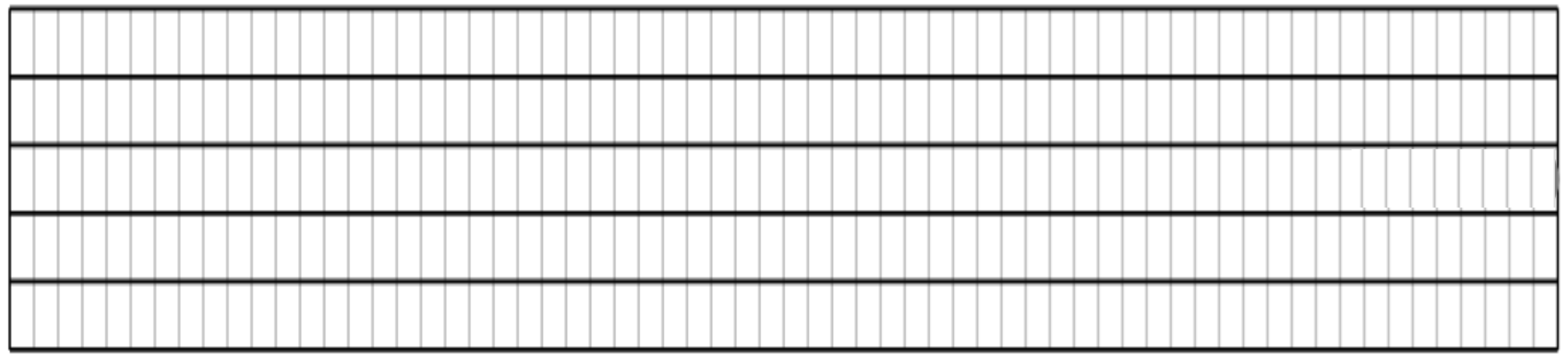
1st round computation

On 320-bit state = 5 x 64-bit words



1st round computation

On 320-bit state = 5 x 64-bit words

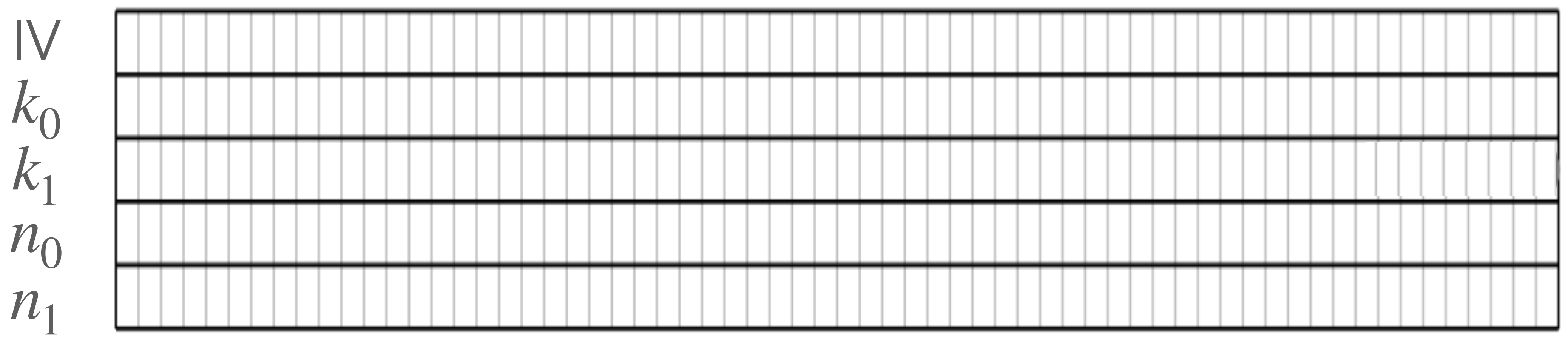


Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

1st round computation

On 320-bit state = 5 x 64-bit words

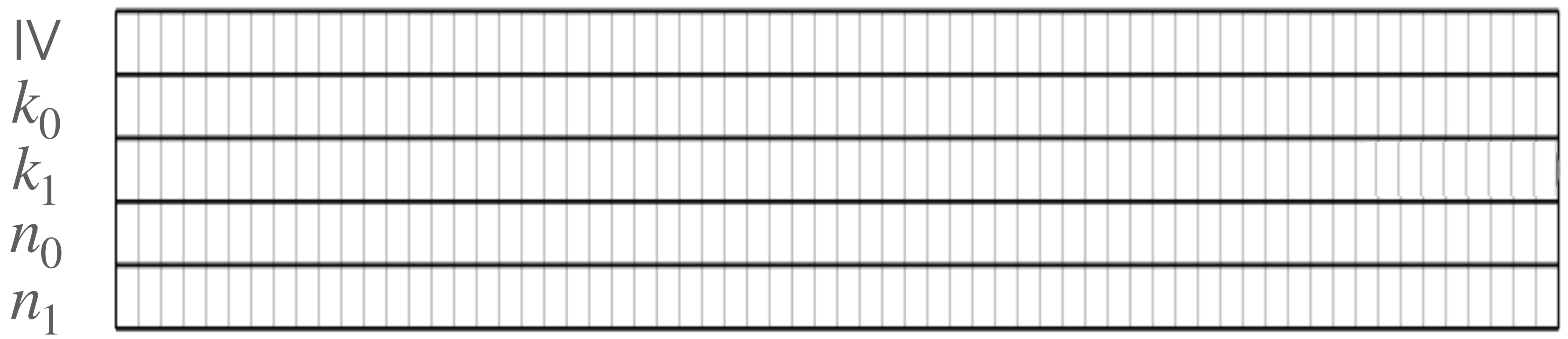


Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

1st round computation

On 320-bit state = 5 x 64-bit words



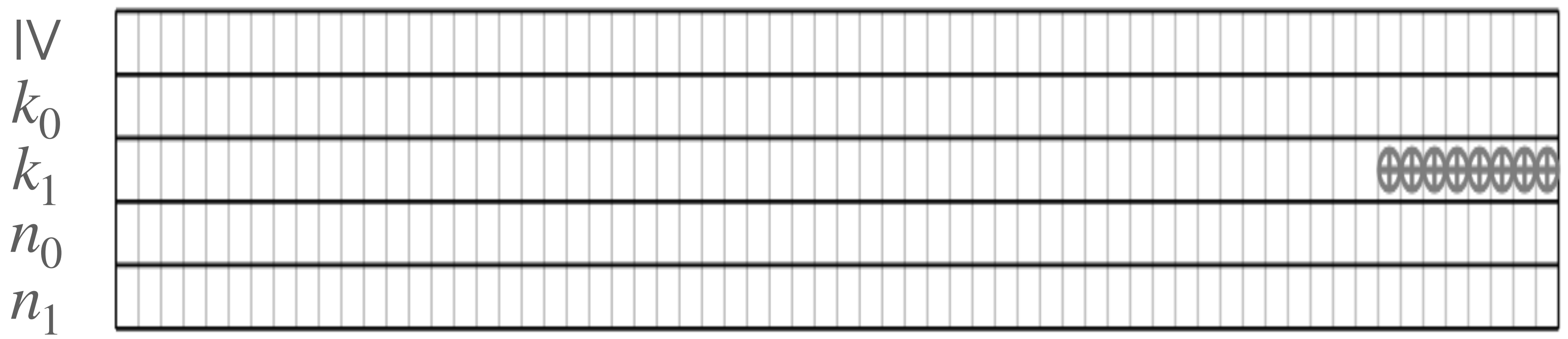
Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

3 operations in a round

1st round computation

On 320-bit state = 5 x 64-bit words



(1) Constant addition

Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

3 operations in a round

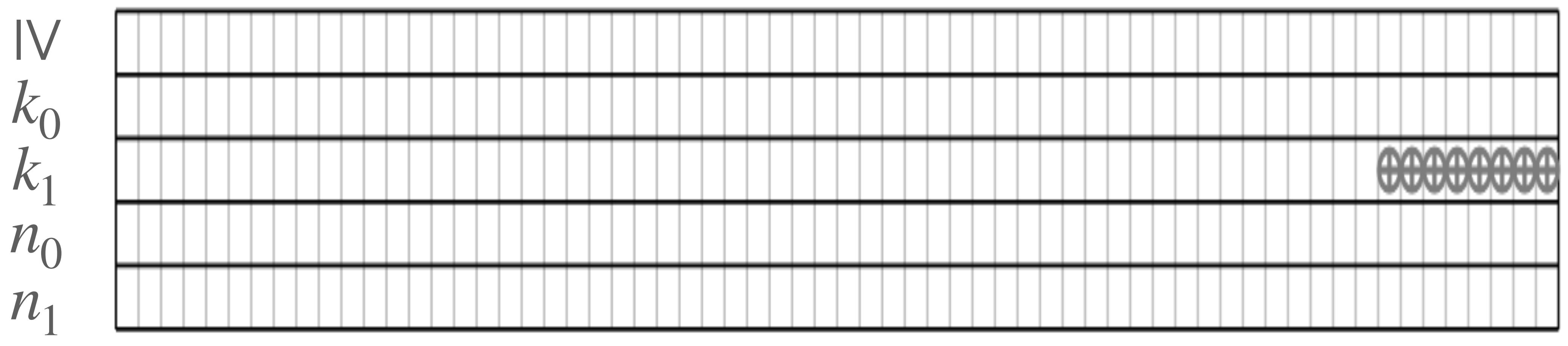
1st round computation

On 320-bit state = 5 x 64-bit words

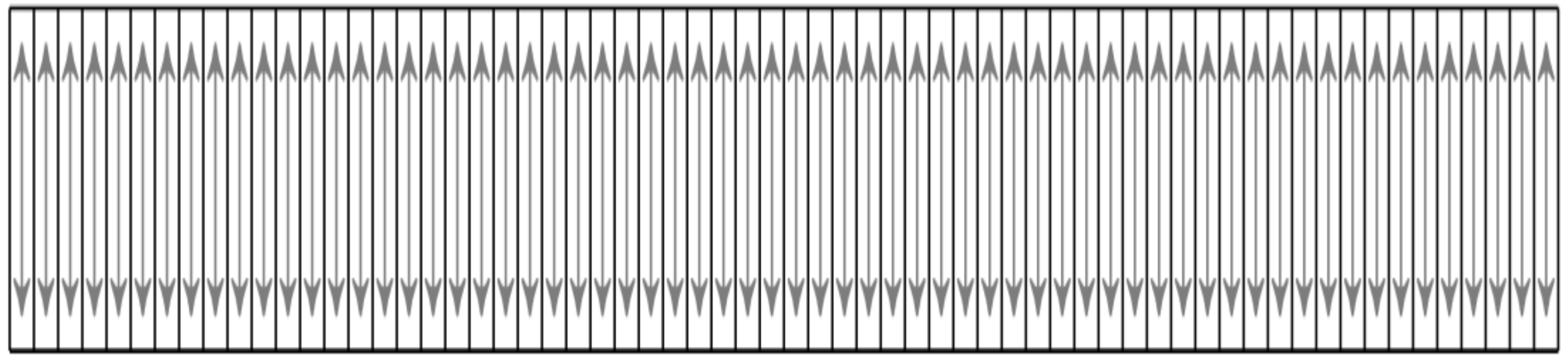
Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

3 operations in a round

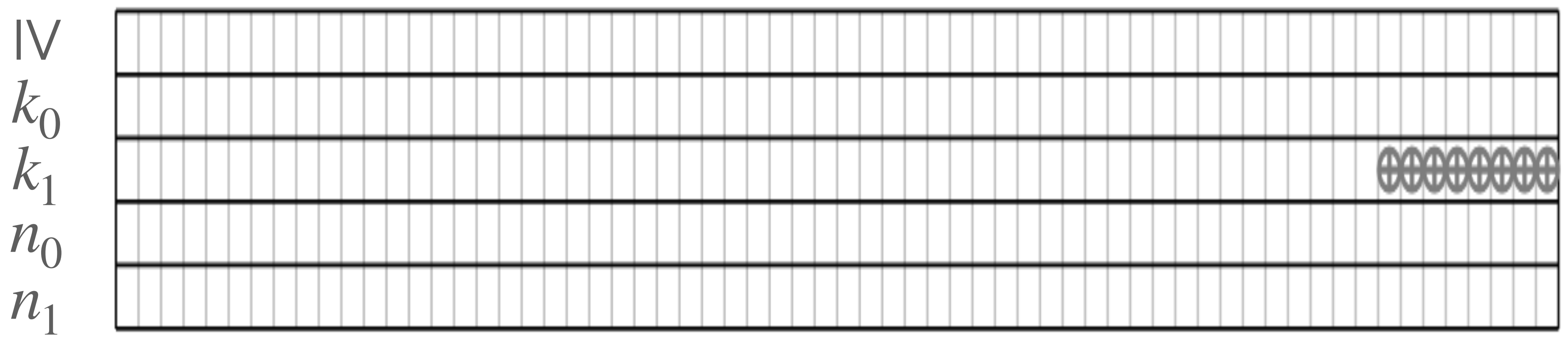


(1) Constant addition


(2) Substitution *(vertical)*

1st round computation

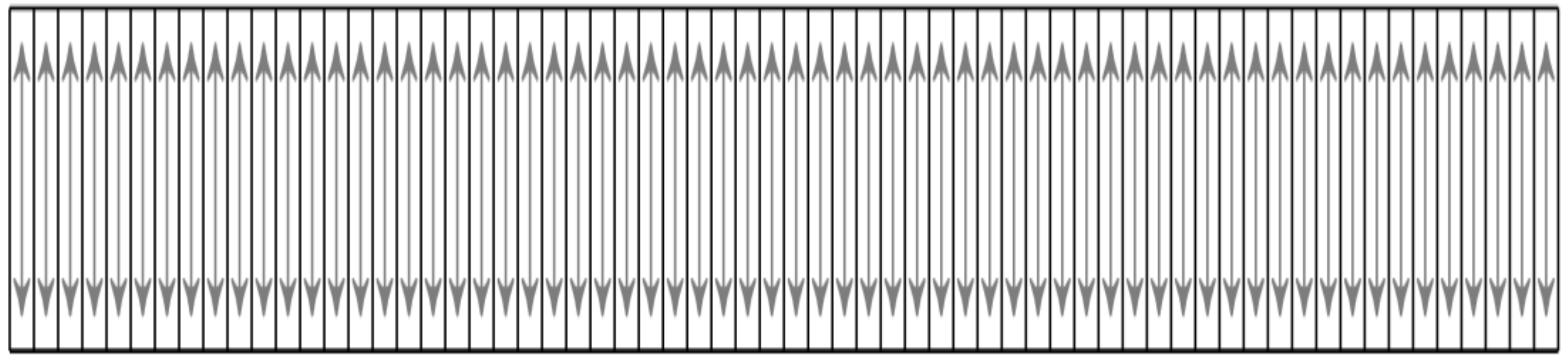
On 320-bit state = 5 x 64-bit words



(1) Constant addition

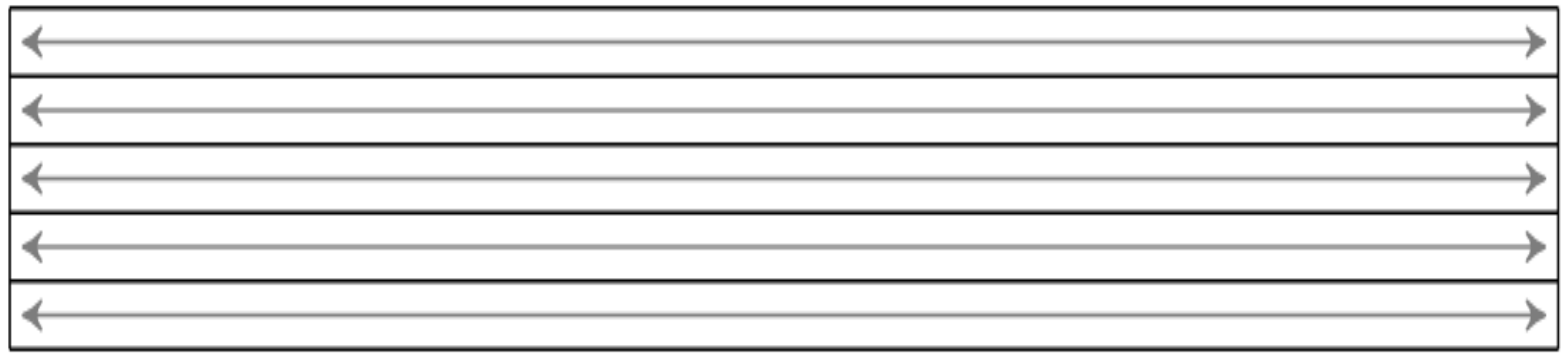
Input of the first round:

- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV



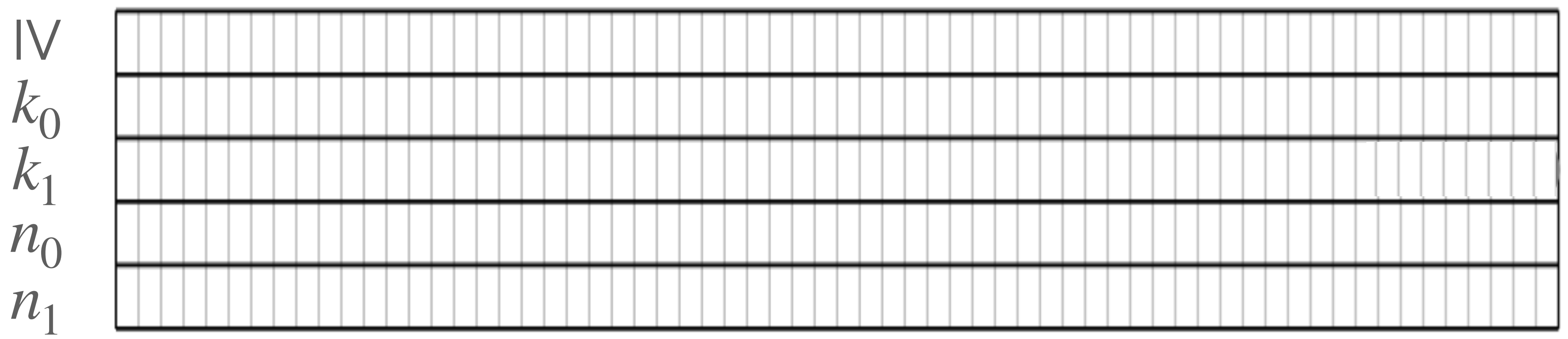
(2) Substitution *(vertical)*

3 operations in a round

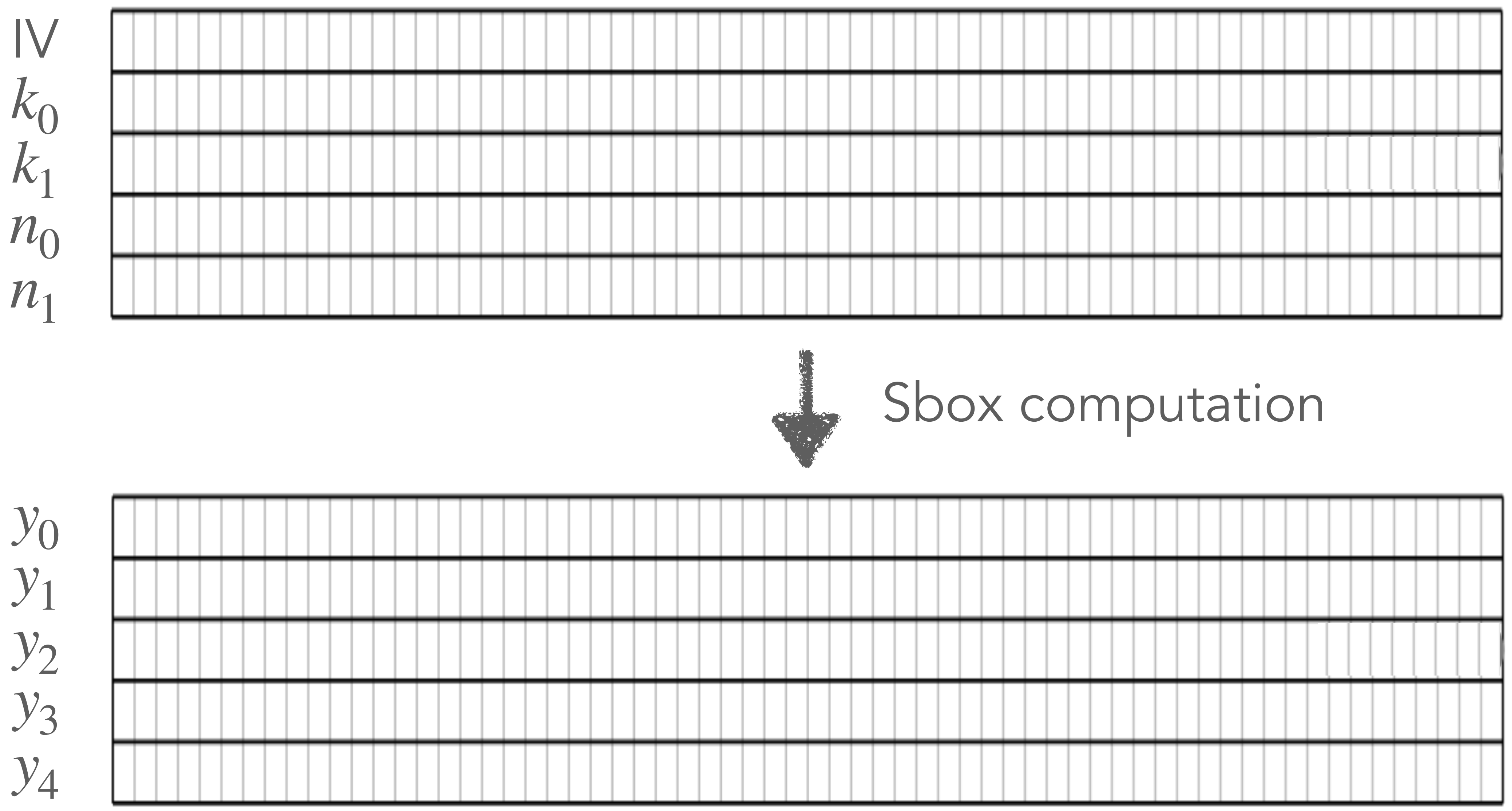


(3) Linear diffusion *(horizontal)*

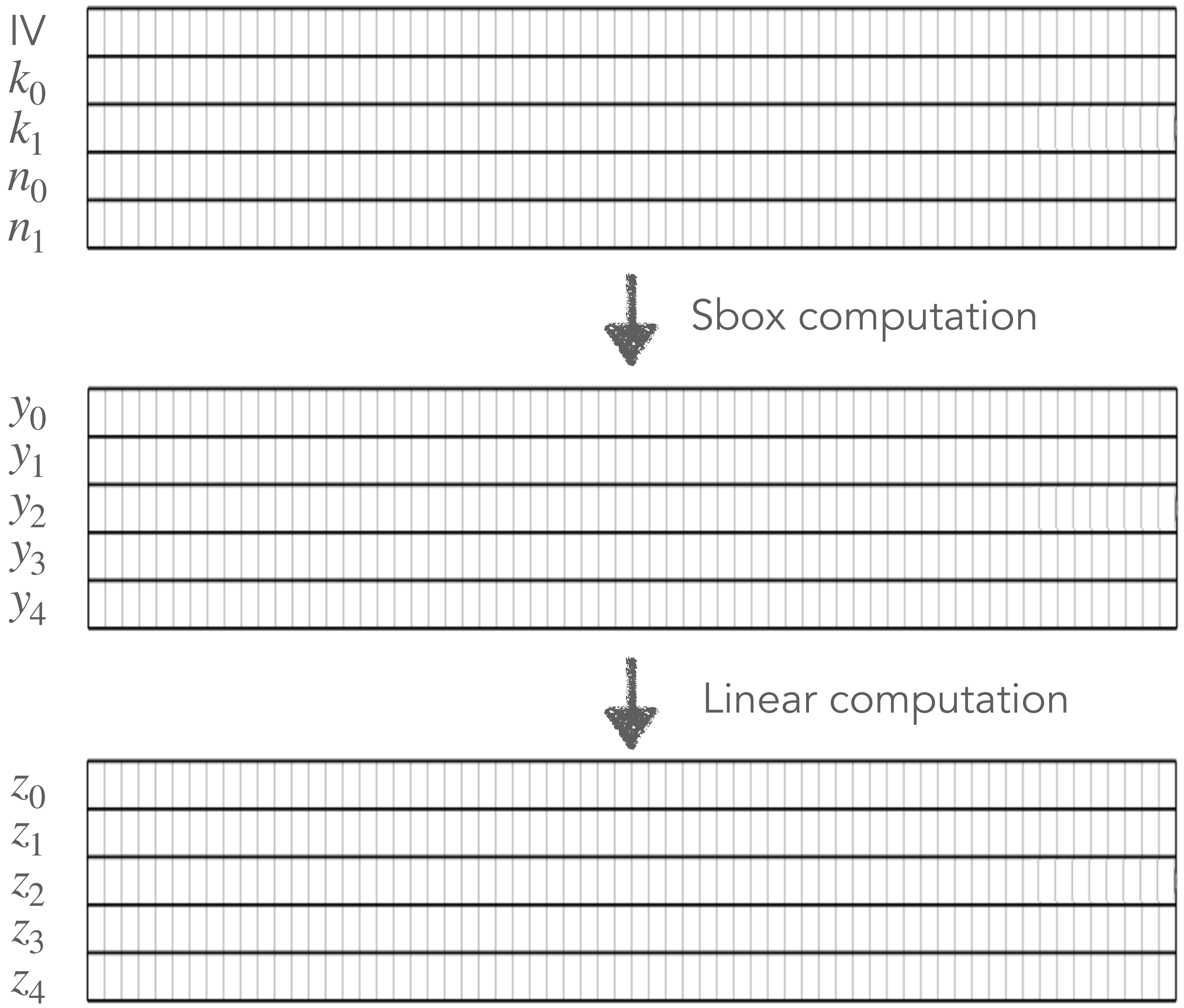
State changes in 1st round



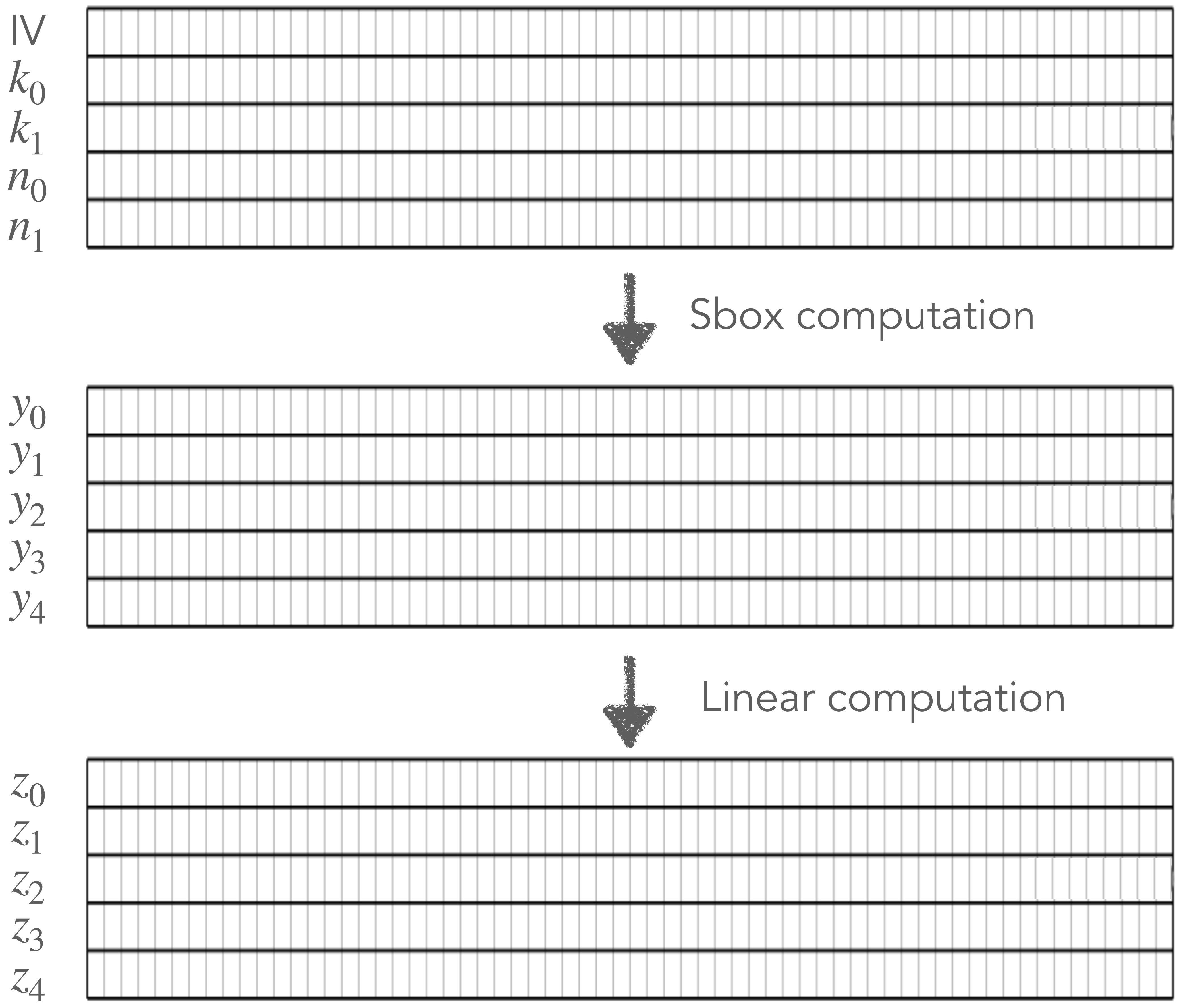
State changes in 1st round



State changes in 1st round

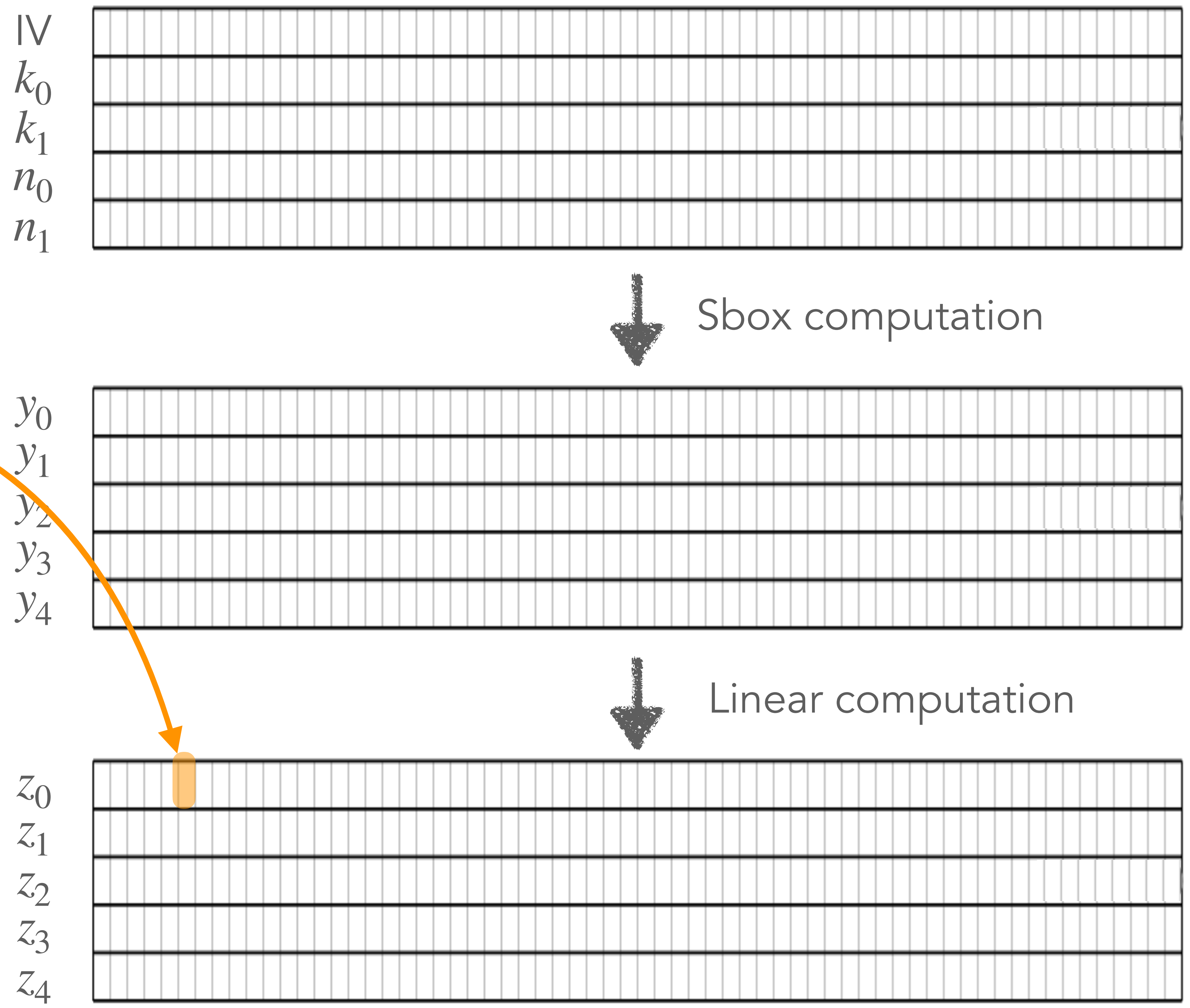


Apply Dobraunig et al.'s attack strategy



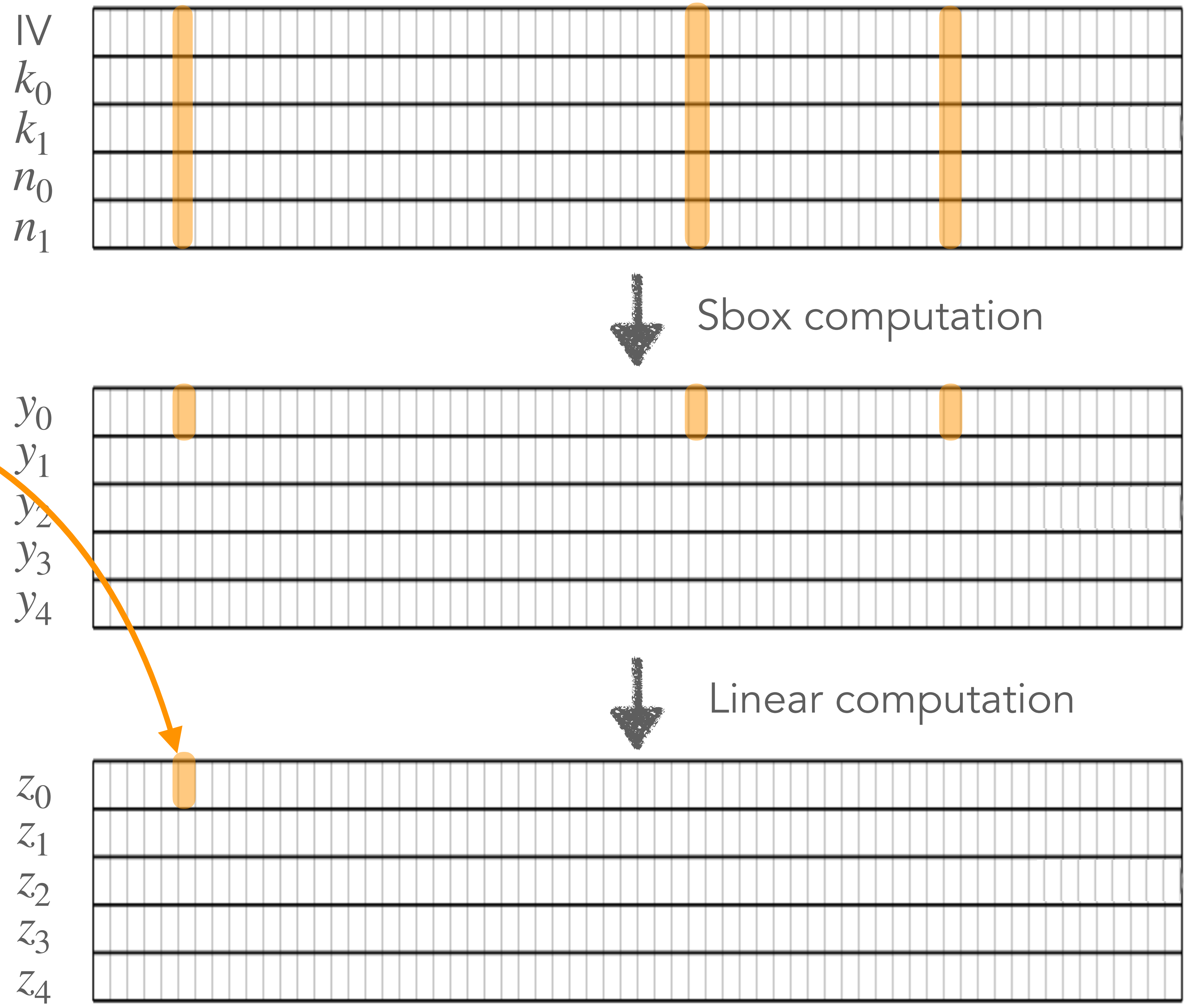
Apply Dobraunig et al.'s attack strategy

intermediate value z_0^j



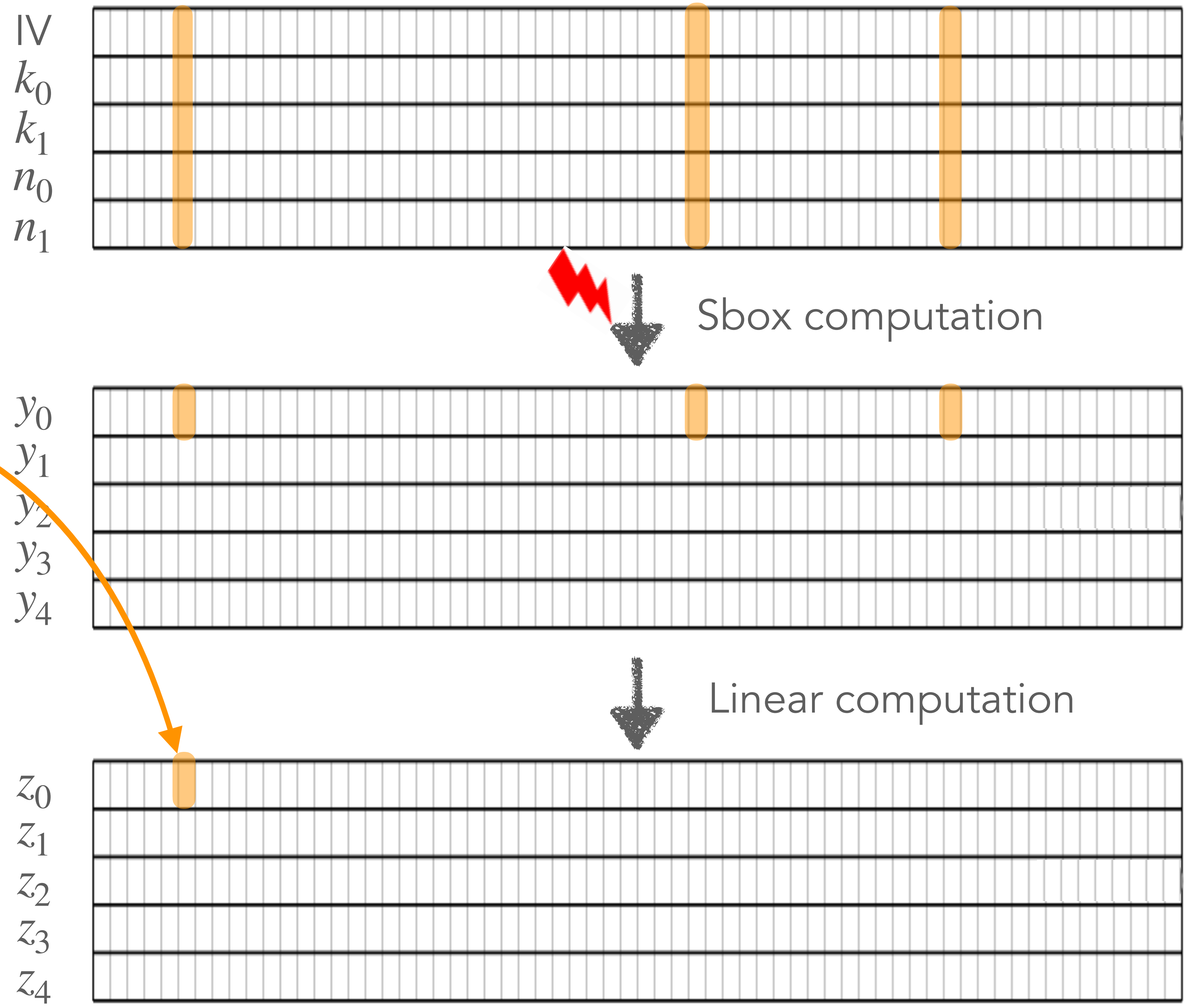
Apply Dobraunig et al.'s attack strategy

intermediate value z_0^j



Apply Dobraunig et al.'s attack strategy

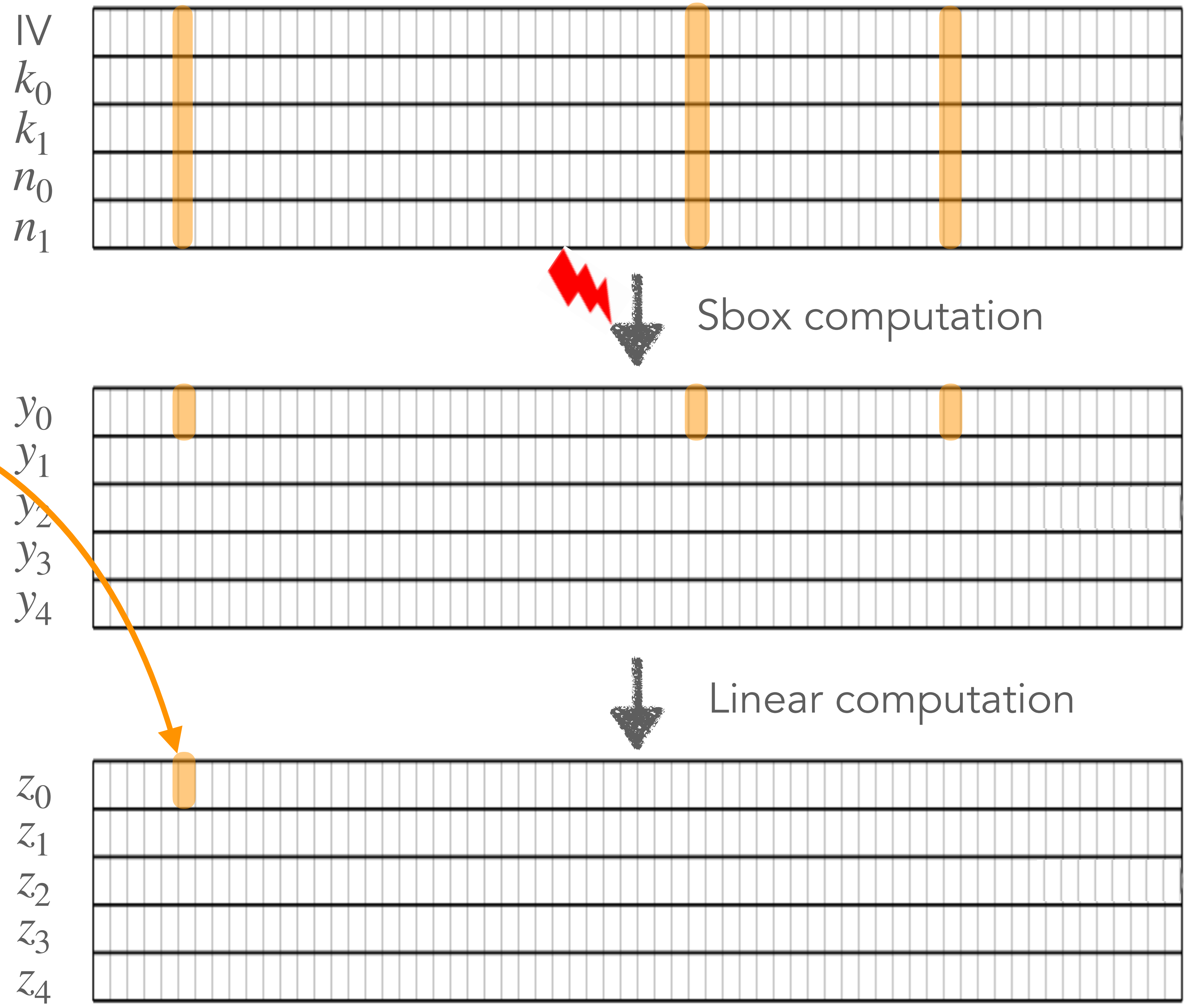
intermediate value z_0^j



Apply Dobraunig et al.'s attack strategy

biased distribution
(under ineffective faults)

intermediate value z_0^j

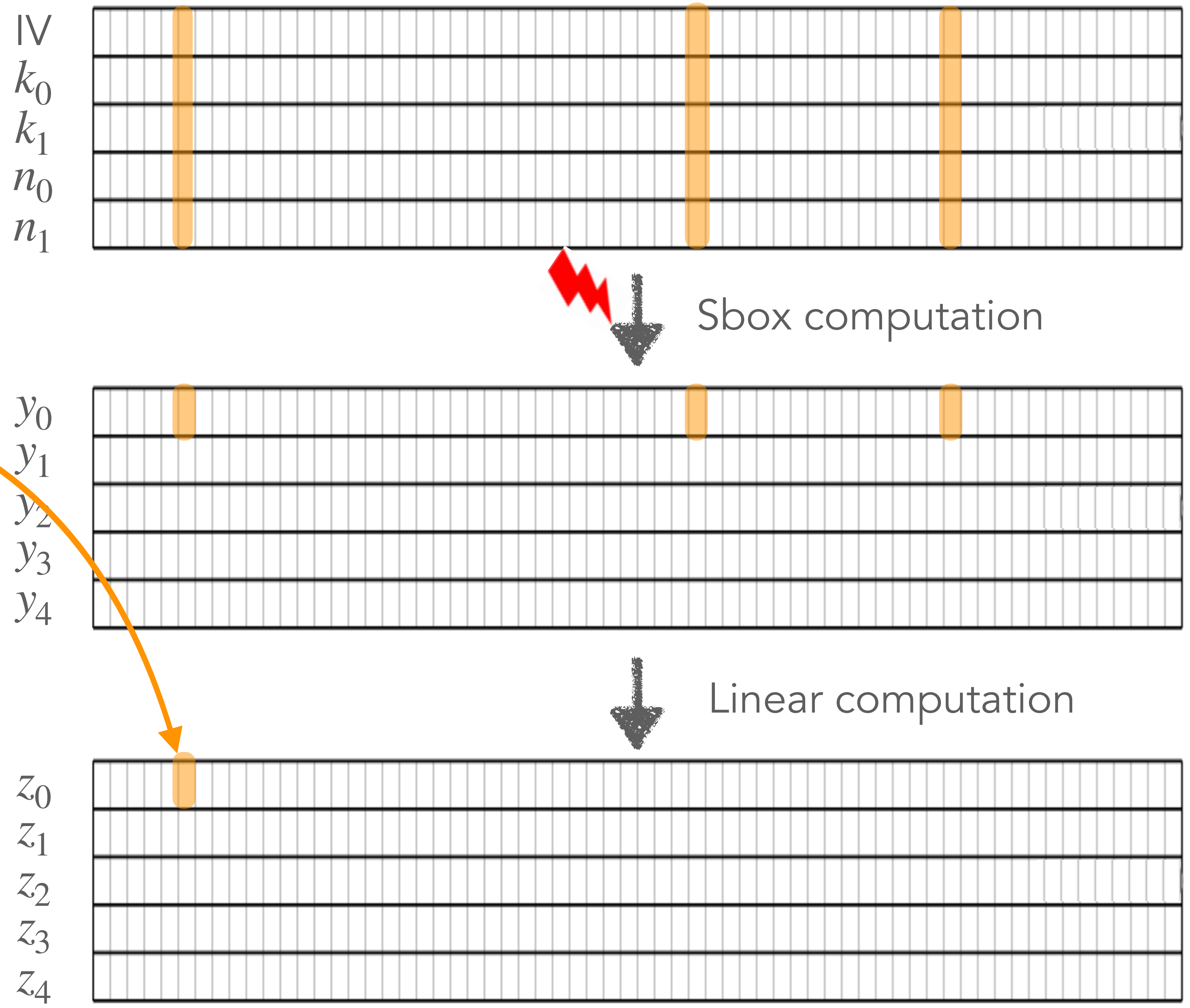


Apply Dobraunig et al.'s attack strategy

biased distribution
(under ineffective faults)

😈 key recovery

intermediate value z_0^j

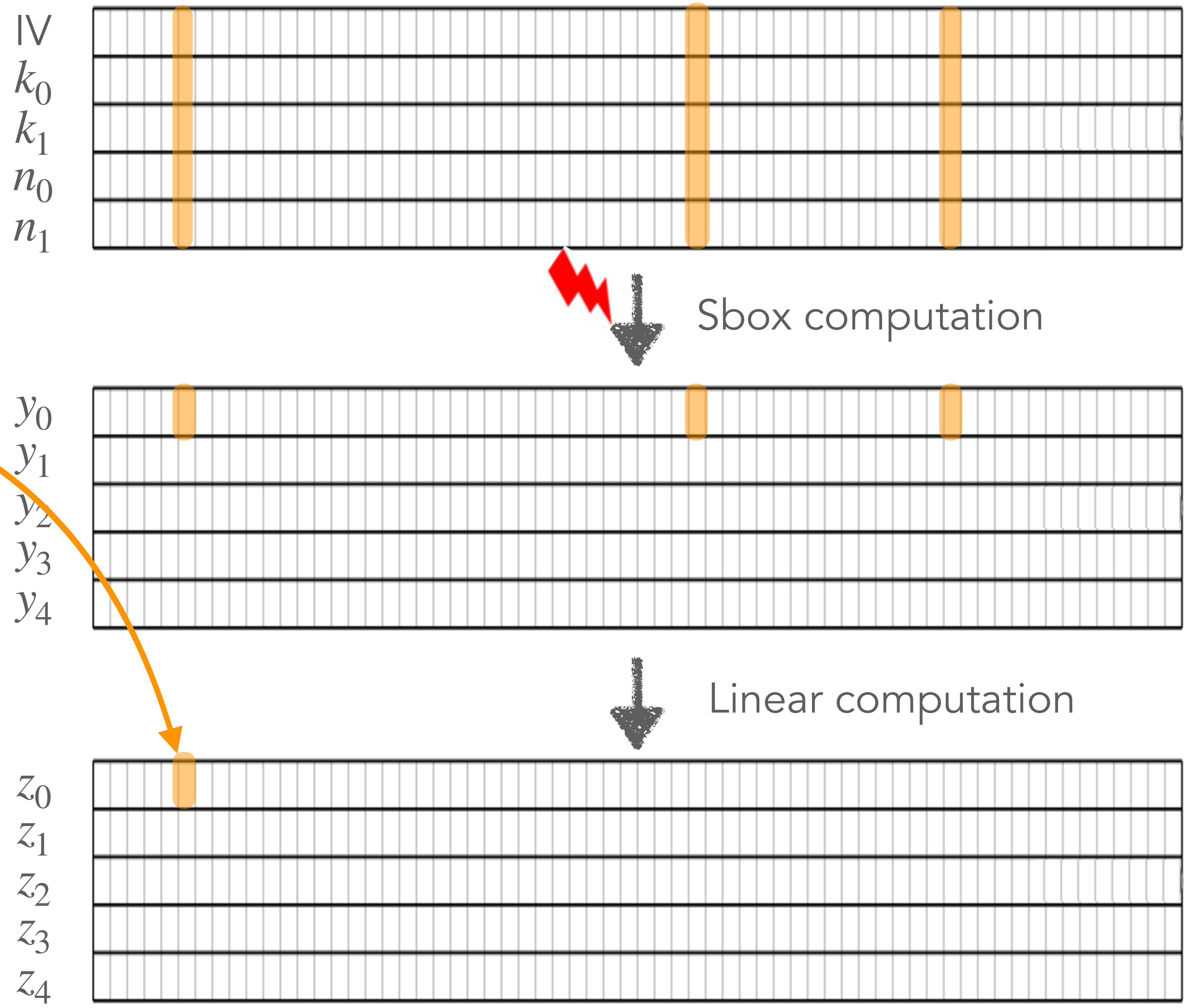


Apply Dobraunig et al.'s attack strategy

biased distribution
(under ineffective faults)

😈 key recovery

intermediate value z_0^j



But we found 2 problems ! 👿

Problem 1:

Possibly impractical to have enough ineffective faults

Problem 1:
Possibly impractical to have enough ineffective faults

(with instruction-skip faults)

Instruction skip: XOR

Instruction skip: XOR

$\mathtt{OP}_0 \quad \mathrm{R}_2 \quad - \quad -$

$\ldots$

$\mathtt{XOR} \quad \mathrm{R}_2 \quad \mathrm{R}_1 \quad \mathrm{R}_0$

$\ldots$

$\mathtt{OP}_2 \quad - \quad \mathrm{R}_2 \quad -$

Scenario 1 : $\mathrm{R}_2 = \mathrm{R}_1 \oplus \mathrm{R}_0$

Instruction skip: XOR

```
OP0   R2   -    -
      ...
XOR   R2   R1   R0
      ...
OP2   -    R2   -
```

Scenario 1 : $R_2 = R_1 \oplus R_0$

No fault	Fault occurs
$R_2 = v_2$	

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0}\quad \mathtt{R_2}\quad -\quad -$	$R_2 = v_2$	
...		
$\mathtt{XOR}\quad \mathtt{R_2}\quad \mathtt{R_1}\quad \mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	
...		
$\mathtt{OP_2}\quad -\quad \mathtt{R_2}\quad -$		

Scenario 1 : $R_2 = R_1 \oplus R_0$

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0} \quad \mathtt{R_2} \quad - \quad -$	$R_2 = v_2$	
$\dots$		
$\mathtt{XOR} \quad \mathtt{R_2} \quad \mathtt{R_1} \quad \mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	
$\dots$		
$\mathtt{OP_2} \quad - \quad \mathtt{R_2} \quad -$	v_2' is used	

Scenario 1 : $\mathrm{R_2} = \mathrm{R_1} \oplus \mathrm{R_0}$

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP}_0 \quad \mathtt{R}_2 \quad - \quad -$	$R_2 = v_2$	$R_2 = v_2$
...		
$\mathtt{XOR} \quad \mathtt{R}_2 \quad \mathtt{R}_1 \quad \mathtt{R}_0$	$R_2 = v_2' = v_1 \oplus v_0$	
...		
$\mathtt{OP}_2 \quad - \quad \mathtt{R}_2 \quad -$	v_2' is used	

Scenario $1 : \mathrm{R}_2 = \mathrm{R}_1 \oplus \mathrm{R}_0$

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0}$ $\mathtt{R_2}$ — —	$R_2 = v_2$	$R_2 = v_2$
…		
$\mathtt{XOR}$ $\mathtt{R_2}$ $\mathtt{R_1}$ $\mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
…		
$\mathtt{OP_2}$ — $\mathtt{R_2}$ —	v_2' is used	

Scenario 1 : $R_2 = R_1 \oplus R_0$

Instruction skip: XOR

Code	No fault	Fault occurs
$\mathtt{OP_0}\quad \mathtt{R_2}\quad -\quad -$	$R_2 = v_2$	$R_2 = v_2$
...		
$\mathtt{XOR}\quad \mathtt{R_2}\quad \mathtt{R_1}\quad \mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
...		
$\mathtt{OP_2}\quad -\quad \mathtt{R_2}\quad -$	v_2' is used	v_2 is used

Scenario 1 : $R_2 = R_1 \oplus R_0$

Instruction skip: XOR

	No fault	Fault occurs
OP$_0$ R$_2$ — —	$R_2 = v_2$	$R_2 = v_2$
...		
XOR R$_2$ R$_1$ R$_0$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
...		
OP$_2$ — R$_2$ —	v_2' is used	v_2 is used

Scenario 1 : $R_2 = R_1 \oplus R_0$

Ineffective fault if $v_2 = v_2'$

Instruction skip: XOR

	No fault	Fault occurs
OP$_0$ R$_2$ – –	$R_2 = v_2$	$R_2 = v_2$
…		
XOR R$_2$ R$_1$ R$_0$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
…		
OP$_2$ – R$_2$ –	v_2' is used	v_2 is used

Scenario 1 : $R_2 = R_1 \oplus R_0$

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform,

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0}$ $\mathtt{R_2}$ — —	$R_2 = v_2$	$R_2 = v_2$
...		
$\mathtt{XOR}$ $\mathtt{R_2}$ $\mathtt{R_1}$ $\mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
...		
$\mathtt{OP_2}$ — $\mathtt{R_2}$ —	v_2' is used	v_2 is used

Scenario 1 : $\mathtt{R_2} = \mathtt{R_1} \oplus \mathtt{R_0}$

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform, then $\Pr[v_2 = v_2'] = 0.5^w$, w is register bit-width

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0}\ \ \mathtt{R_2}\ \ -\ \ -$	$R_2 = v_2$	$R_2 = v_2$
...		
$\mathtt{XOR}\ \ \mathtt{R_2}\ \ \mathtt{R_1}\ \ \mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
...		
$\mathtt{OP_2}\ \ -\ \ \mathtt{R_2}\ \ -$	v_2' is used	v_2 is used

Scenario 1 : $R_2 = R_1 \oplus R_0$

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform, then $\Pr[v_2 = v_2'] = 0.5^w$, w is register bit-width

Architecture	8-bit	32-bit	64-bit
$\Pr[v_2 = v_2']$			

Instruction skip: XOR

```
OP0   R2   —    —
      ...
XOR   R2   R1   R0
      ...
OP2   —    R2   —
```

Scenario 1 : $R_2 = R_1 \oplus R_0$

	No fault	Fault occurs
$\mathrm{OP}_0\ R_2\ -\ -$	$R_2 = v_2$	$R_2 = v_2$
$\mathrm{XOR}\ R_2\ R_1\ R_0$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
$\mathrm{OP}_2\ -\ R_2\ -$	v_2' is used	v_2 is used

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform, then $\Pr[v_2 = v_2'] = 0.5^w$, w is register bit-width

Architecture	8-bit	32-bit	64-bit
$\Pr[v_2 = v_2']$	≈ 0.0039 😈		

Instruction skip: XOR

	No fault	Fault occurs
$\mathtt{OP_0}\ \ \mathtt{R_2}\ \ -\ \ -$	$R_2 = v_2$	$R_2 = v_2$
...		
$\mathtt{XOR}\ \ \mathtt{R_2}\ \ \mathtt{R_1}\ \ \mathtt{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
...		
$\mathtt{OP_2}\ \ -\ \ \mathtt{R_2}\ \ -$	v_2' is used	v_2 is used

$\mathrm{Scenario\ 1} : \mathrm{R_2} = \mathrm{R_1} \oplus \mathrm{R_0}$

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform, then $\Pr[v_2 = v_2'] = 0.5^w$, w is register bit-width

Architecture	8-bit	32-bit	64-bit
$\Pr[v_2 = v_2']$	≈ 0.0039	≈ 0	≈ 0
	😈	👿	👿

Instruction skip: XOR

	No fault	Fault occurs
$\mathrm{OP}_0 \quad \mathrm{R}_2 \quad - \quad -$	$R_2 = v_2$	$R_2 = v_2$
$\dots$		
$\mathrm{XOR} \quad \mathrm{R}_2 \quad \mathrm{R}_1 \quad \mathrm{R}_0$	$R_2 = v_2' = v_1 \oplus v_0$	(skipped)
$\dots$		
$\mathrm{OP}_2 \quad - \quad \mathrm{R}_2 \quad -$	v_2' is used	v_2 is used

Scenario $1 : \mathrm{R}_2 = \mathrm{R}_1 \oplus \mathrm{R}_0$

Ineffective fault if $v_2 = v_2'$

Suppose v_0, v_1, v_2 are uniform, then $\Pr[v_2 = v_2'] = 0.5^w$, w is register bit-width

Architecture	8-bit	32-bit	64-bit
$\Pr[v_2 = v_2']$	≈ 0.0039 😈	≈ 0 👿	≈ 0 👿

Impractical to obtain ineffective fault ! 👿

Instruction skip: XOR

OP$_0$ R$_2$ — —	$R_2 = v_2$
...	
XOR R$_2$ R$_1$ R$_0$	$R_2 = v_2' = v_1 \oplus v_0$
...	
OP$_2$ — R$_2$ —	v_2' is used
Scenario 1 : $R_2 = R_1 \oplus R_0$	

Instruction skip: XOR

$$\begin{array}{llll} \mathtt{OP}_0 & \mathtt{R}_2 & - & - \\ & \dots & & \\ \mathtt{XOR} & \mathtt{R}_2 & \mathtt{R}_1 & \mathtt{R}_0 \\ & \dots & & \\ \mathtt{OP}_2 & - & \mathtt{R}_2 & - \end{array}$$

Scenario 1 : $\mathtt{R}_2 = \mathtt{R}_1 \oplus \mathtt{R}_0$

$R_2 = v_2$

$R_2 = v_2' = v_1 \oplus v_0$

v_2' is used

$$\begin{array}{llll} \mathtt{OP}_0 & \mathtt{R}_2 & - & - \\ & \dots & & \\ \mathtt{XOR} & \mathtt{R}_2 & \mathtt{R}_2 & \mathtt{R}_0 \\ & \dots & & \\ \mathtt{OP}_2 & - & \mathtt{R}_2 & - \end{array}$$

Scenario 2 : $\mathtt{R}_2 = \mathtt{R}_2 \oplus \mathtt{R}_0$

$R_2 = v_2$

Instruction skip: XOR

$\mathtt{OP_0}$ $\mathrm{R_2}$ — —	$R_2 = v_2$
...	
$\mathtt{XOR}$ $\mathrm{R_2}$ $\mathrm{R_1}$ $\mathrm{R_0}$	$R_2 = v_2' = v_1 \oplus v_0$
...	
$\mathtt{OP_2}$ — $\mathrm{R_2}$ —	v_2' is used
Scenario 1 : $\mathrm{R_2} = \mathrm{R_1} \oplus \mathrm{R_0}$	

$\mathtt{OP_0}$ $\mathrm{R_2}$ — —	$R_2 = v_2$	
...		
$\mathtt{XOR}$ $\mathrm{R_2}$ $\boxed{\mathrm{R_2}}$ $\mathrm{R_0}$	$R_2 = v_2' = v_2 \oplus v_0$	(R_2 is reused as source register)
...		
$\mathtt{OP_2}$ — $\mathrm{R_2}$ —	v_2' is used	
Scenario 2 : $\mathrm{R_2} = \mathrm{R_2} \oplus \mathrm{R_0}$		

Instruction skip: XOR, AND, NOT

	Probability of an ineffective fault Pr$[v_2 = v_2']$
XOR (scenario 1)	0.5^w
XOR (scenario 2)	0.5^w

Instruction skip: XOR, AND, NOT

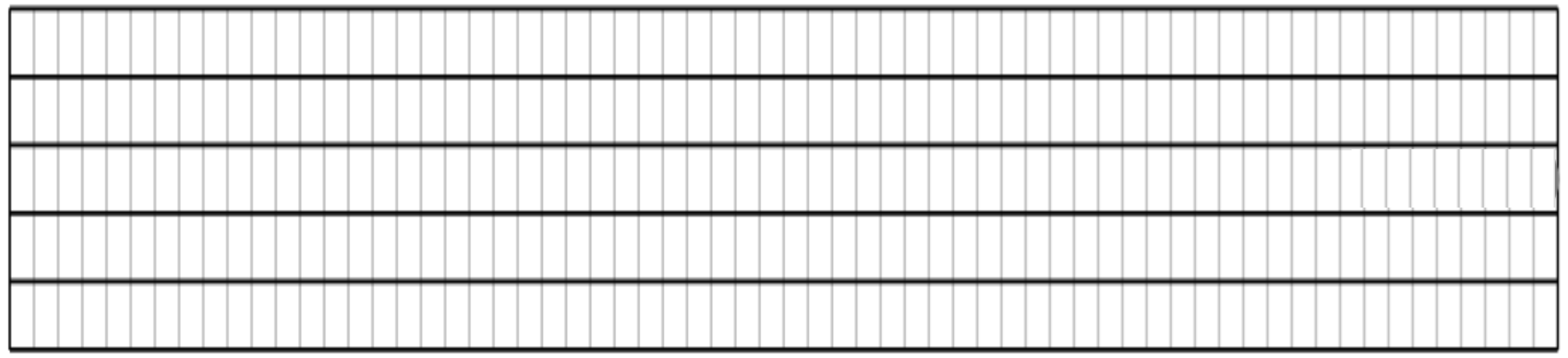
	Probability of an ineffective fault $\Pr[v_2 = v_2']$
XOR (scenario 1)	0.5^w
XOR (scenario 2)	0.5^w
AND (scenario 1)	0.5^w
AND (scenario 2)	0.75^w

Instruction skip: XOR, AND, NOT

	Probability of an ineffective fault Pr$[v_2 = v_2']$
XOR (scenario 1)	0.5^w
XOR (scenario 2)	0.5^w
AND (scenario 1)	0.5^w
AND (scenario 2)	0.75^w
NOT (scenario 1)	0.5^w
NOT (scenario 2)	0

Implementations of Ascon

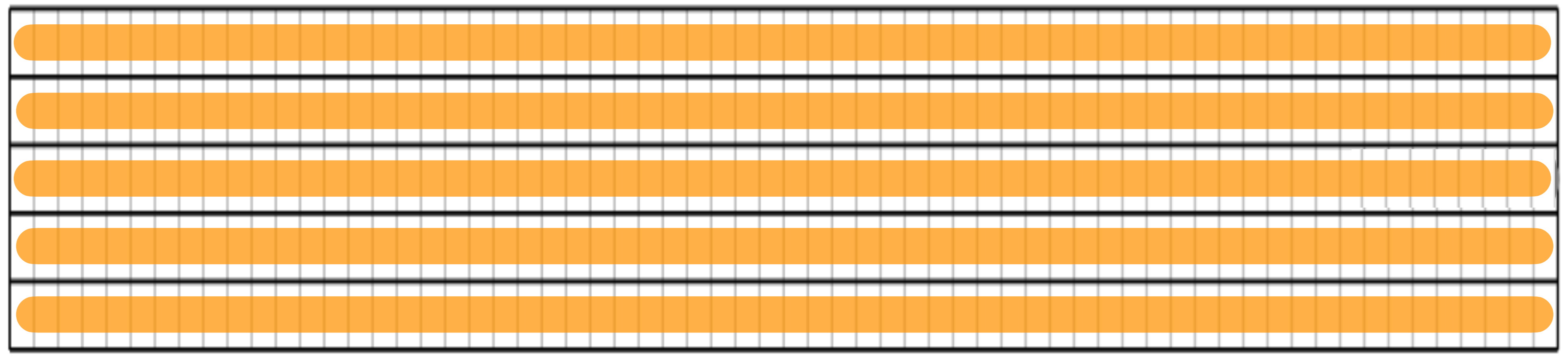
On 320-bit state = 5 x 64-bit words



Implementations of Ascon

On 320-bit state = 5 x 64-bit words

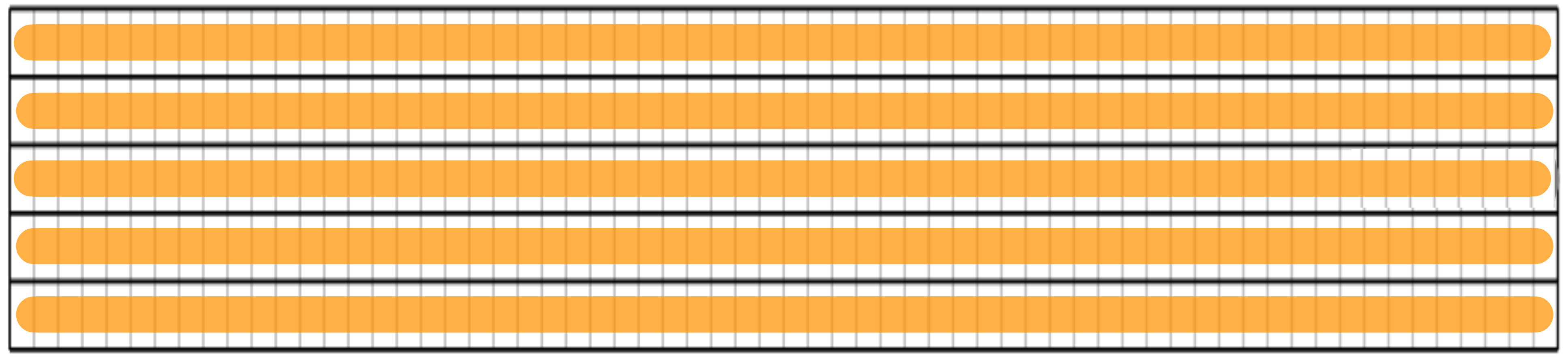
64-bit implementation →



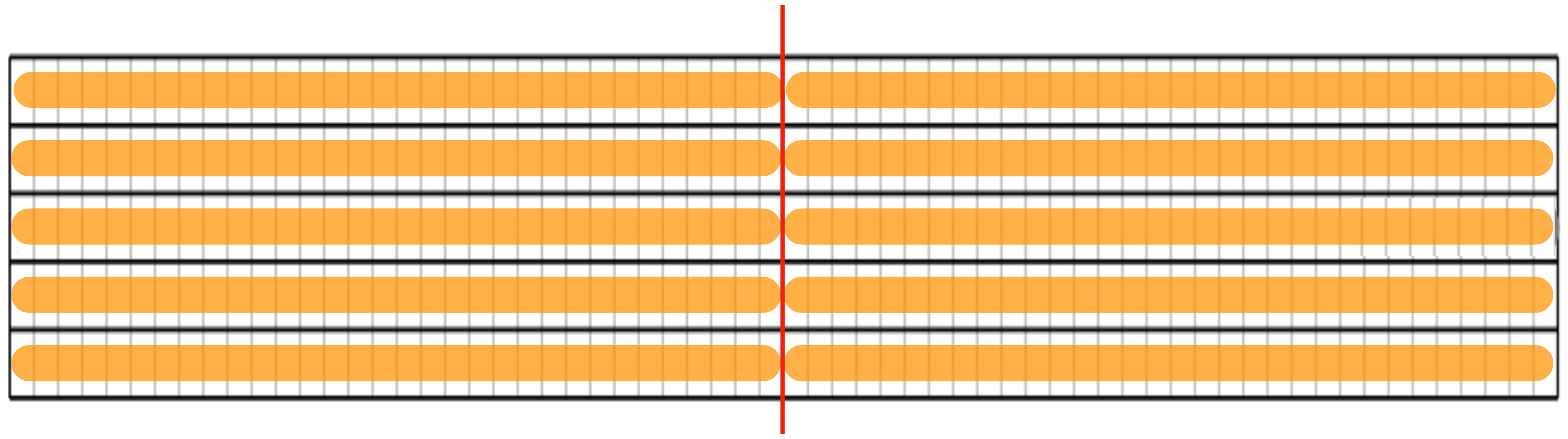
Implementations of Ascon

On 320-bit state = 5 x 64-bit words

64-bit implementation →



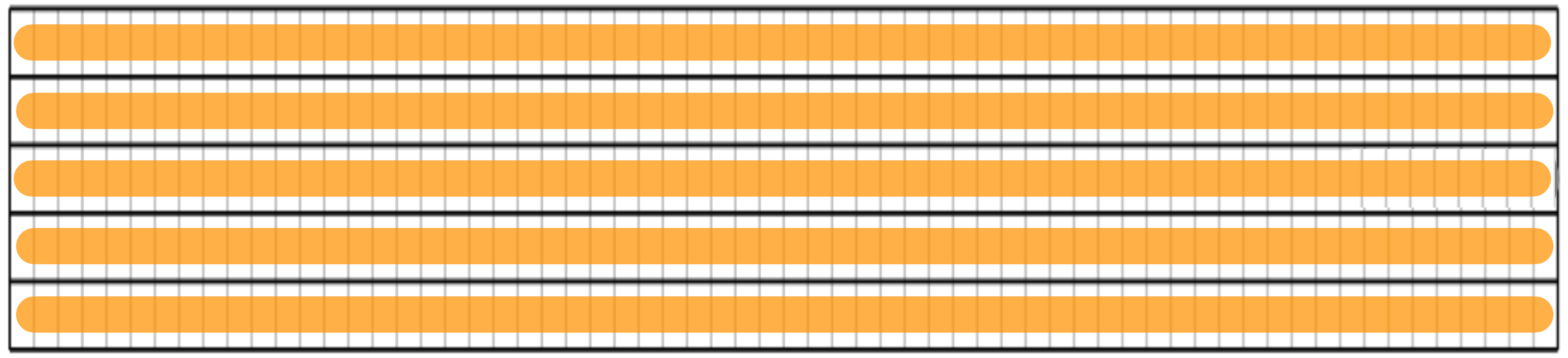
32-bit implementation →



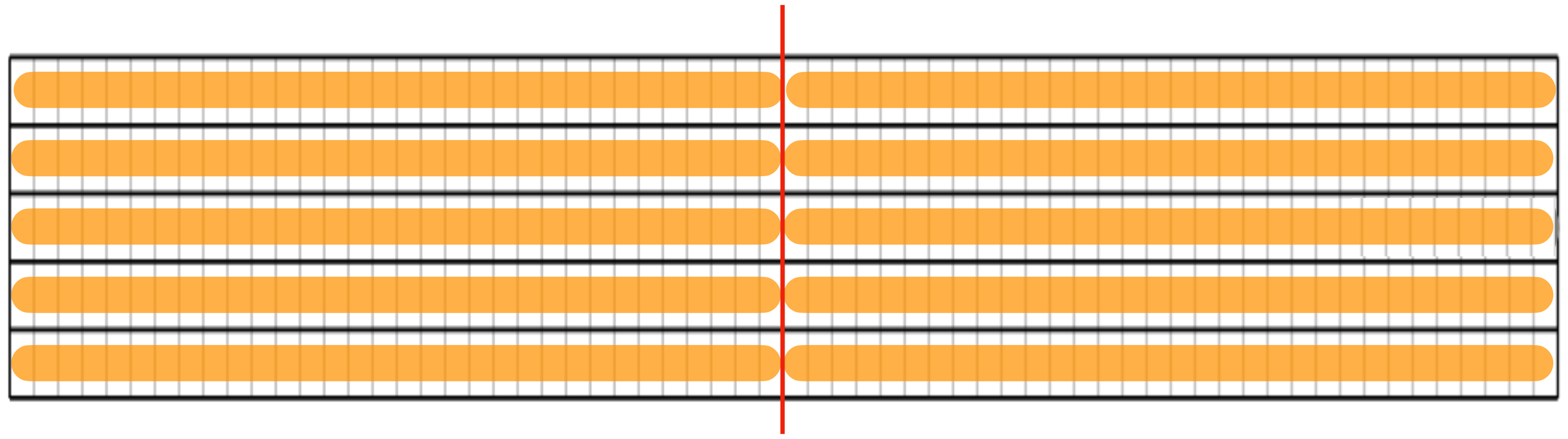
Implementations of Ascon

On 320-bit state = 5 x 64-bit words

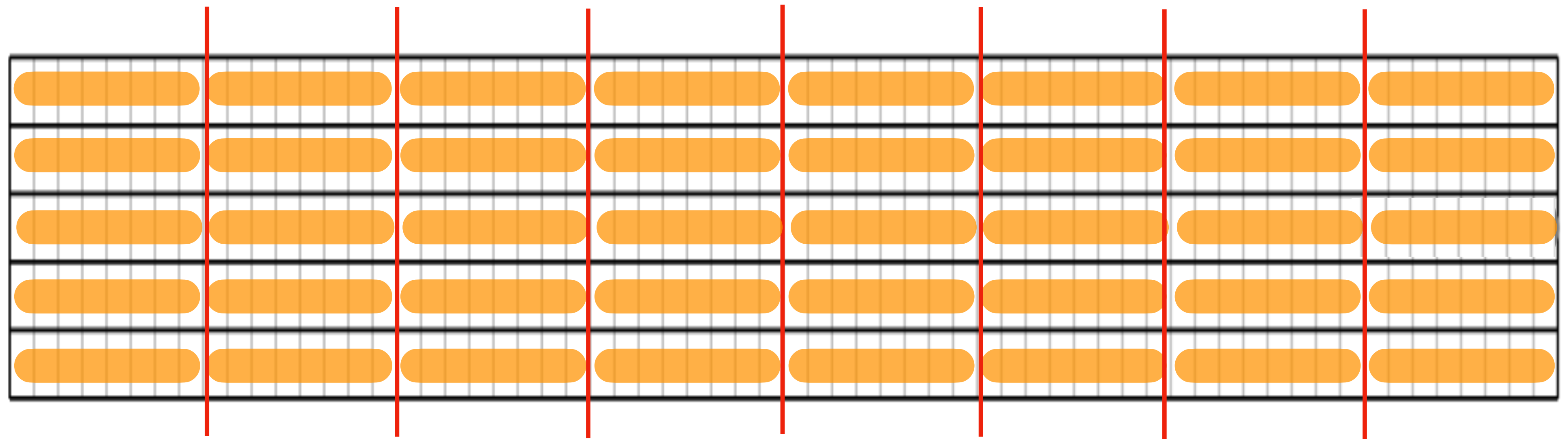
64-bit implementation →



32-bit implementation →



8-bit implementation →



Apply to 8-bit implementation of Ascon

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
`XOR S1`			
`XOR S2`			
`AND S1`			
`AND S2`			
`NOT S1`			
`NOT S2`			

Apply to 8-bit implementation of Ascon

- $w = 8$

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
XOR S1			
XOR S2			
AND S1			
AND S2			
NOT S1			
NOT S2			

Apply to 8-bit implementation of Ascon

- $w = 8$

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
XOR S1			0.0039
XOR S2			0.0039
AND S1			0.0039
AND S2			0.1001
NOT S1			0.0039
NOT S2			0

Apply to 8-bit implementation of Ascon

- $w = 8$
- 20,000 executions with random inputs

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
XOR S1			0.0039
XOR S2			0.0039
AND S1			0.0039
AND S2			0.1001
NOT S1			0.0039
NOT S2			0

Apply to 8-bit implementation of Ascon

- $w = 8$
- 20,000 executions with random inputs

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
`XOR S1`	81		0.0039
`XOR S2`	79		0.0039
`AND S1`	80		0.0039
`AND S2`	2011		0.1001
`NOT S1`	74		0.0039
`NOT S2`	0		0

Apply to 8-bit implementation of Ascon

- $w = 8$
- 20,000 executions with random inputs

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
XOR S1	81	0.0040	0.0039
XOR S2	79	0.0040	0.0039
AND S1	80	0.0040	0.0039
AND S2	2011	0.1006	0.1001
NOT S1	74	0.0037	0.0039
NOT S2	0	0	0

Apply to 8-bit implementation of Ascon

- $w = 8$
- 20,000 executions with random inputs

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
`XOR S1`	81	0.0040	0.0039
`XOR S2`	79	0.0040	0.0039
`AND S1`	80	0.0040	0.0039
`AND S2`	2011	0.1006	0.1001
`NOT S1`	74	0.0037	0.0039
`NOT S2`	0	0	0

This confirms our analysis ✅

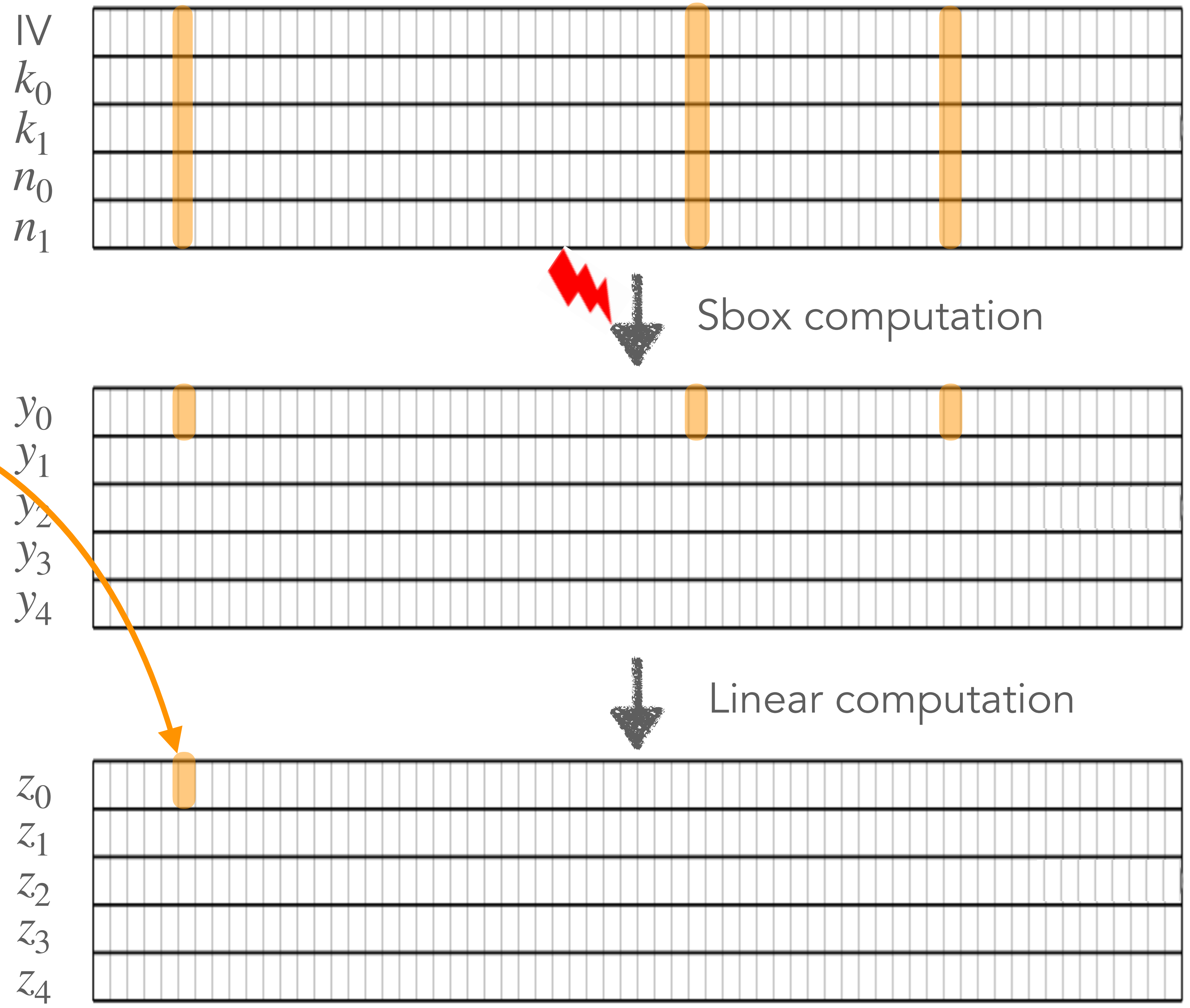
Problem 2:
Intermediate value remains uniform

Problem 2:
Intermediate value remains uniform

(with instruction-skip on 8-bit implementation)

Apply Dobraunig et al.'s attack strategy

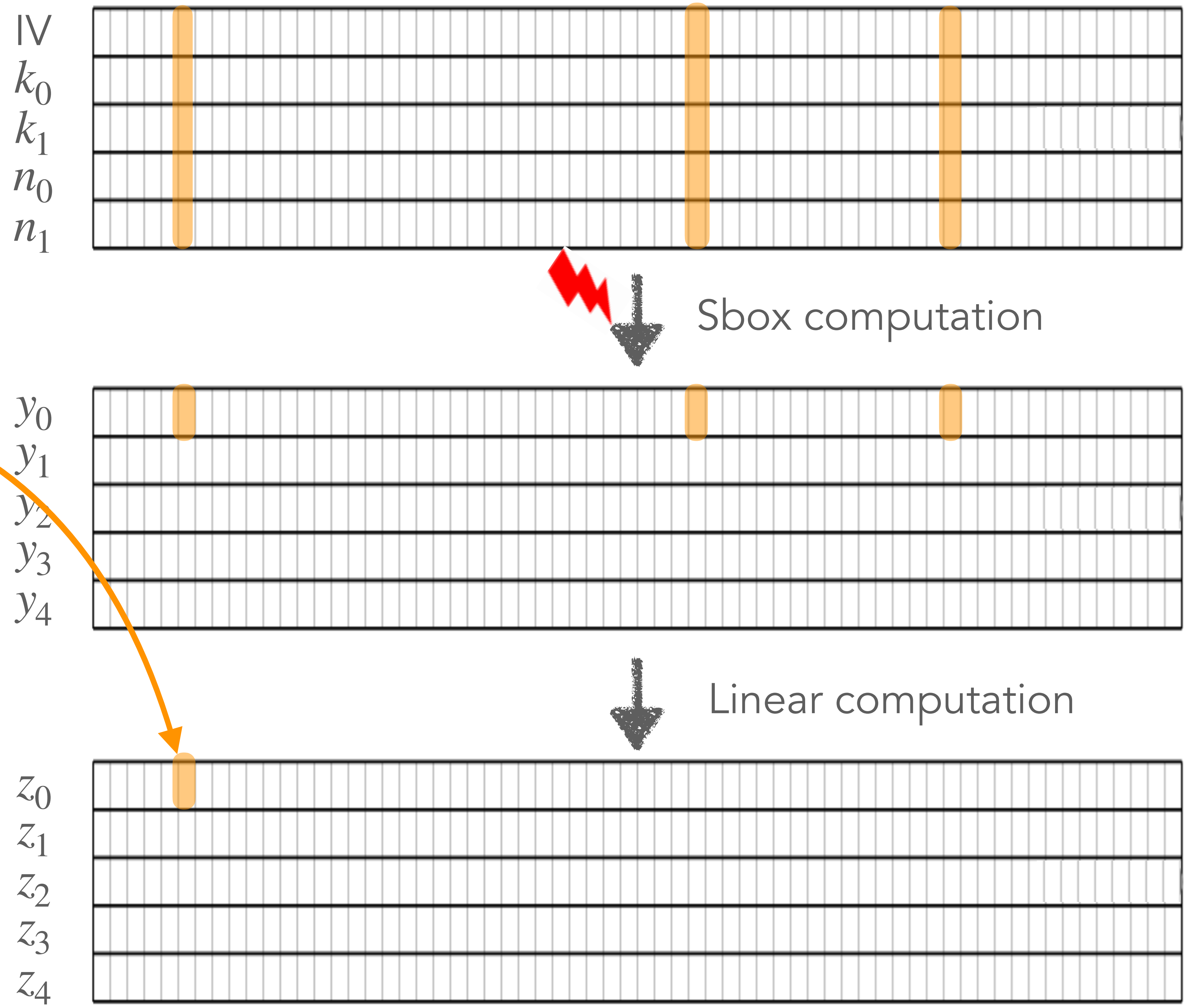
intermediate value z_0^j



Apply Dobraunig et al.'s attack strategy

intermediate value z_0^j

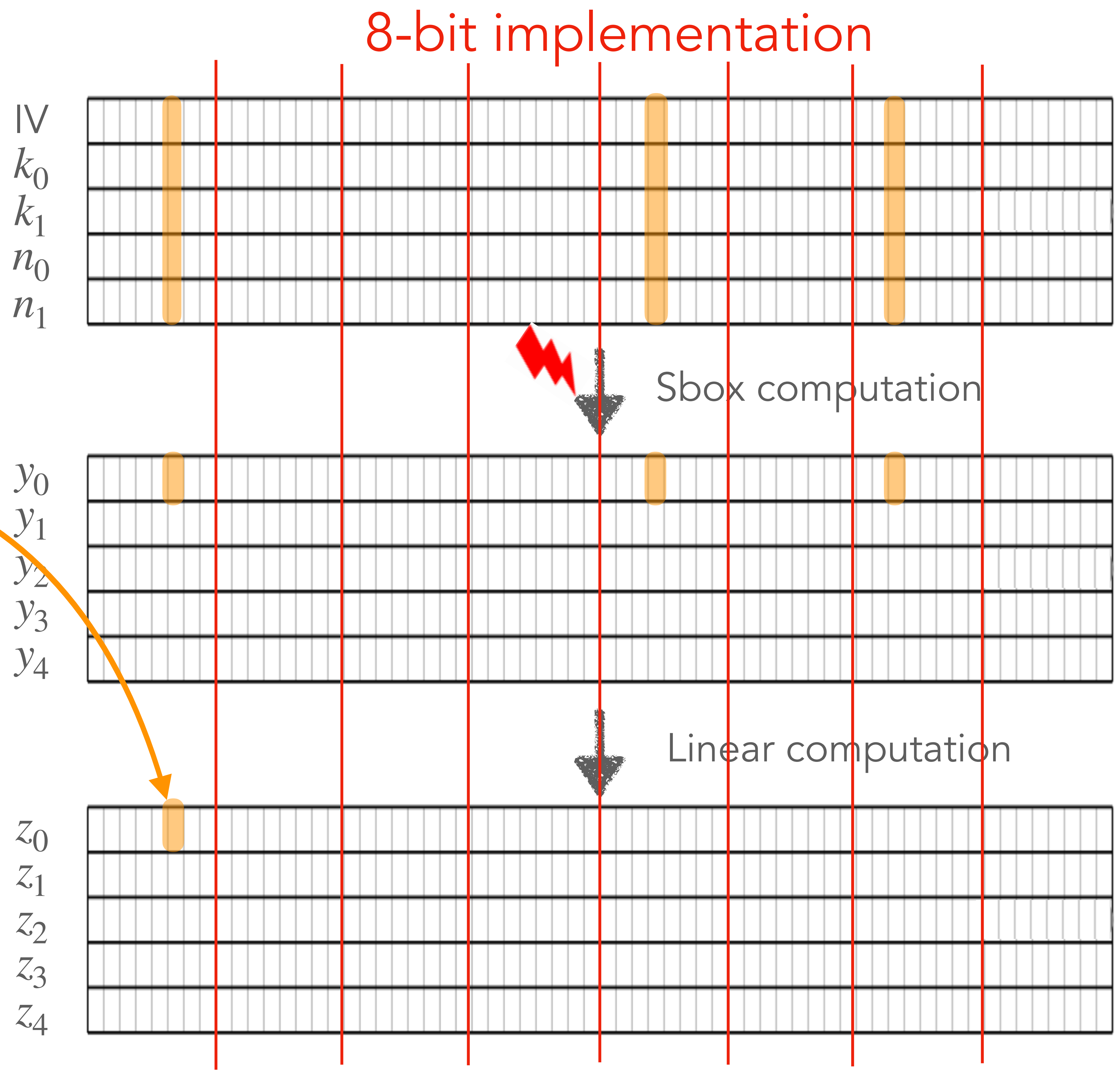
$$z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}$$



Apply Dobraunig et al.'s attack strategy

intermediate value z_0^j

$$z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}$$

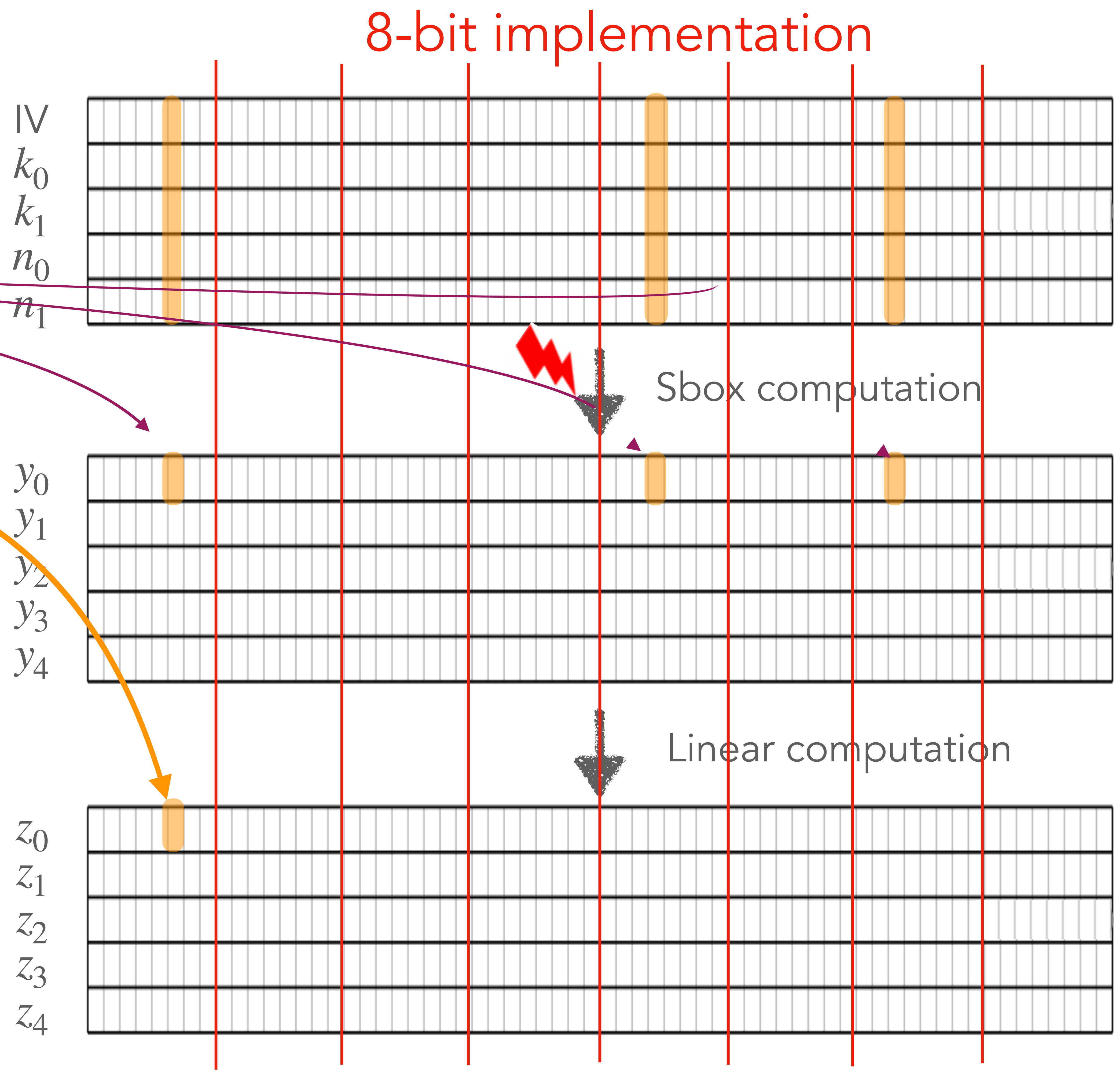


Apply Dobraunig et al.'s attack strategy

An instruction skip
does not affect all 3 bits

intermediate value z_0^j

$$z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}$$



Apply Dobraunig et al.'s attack strategy

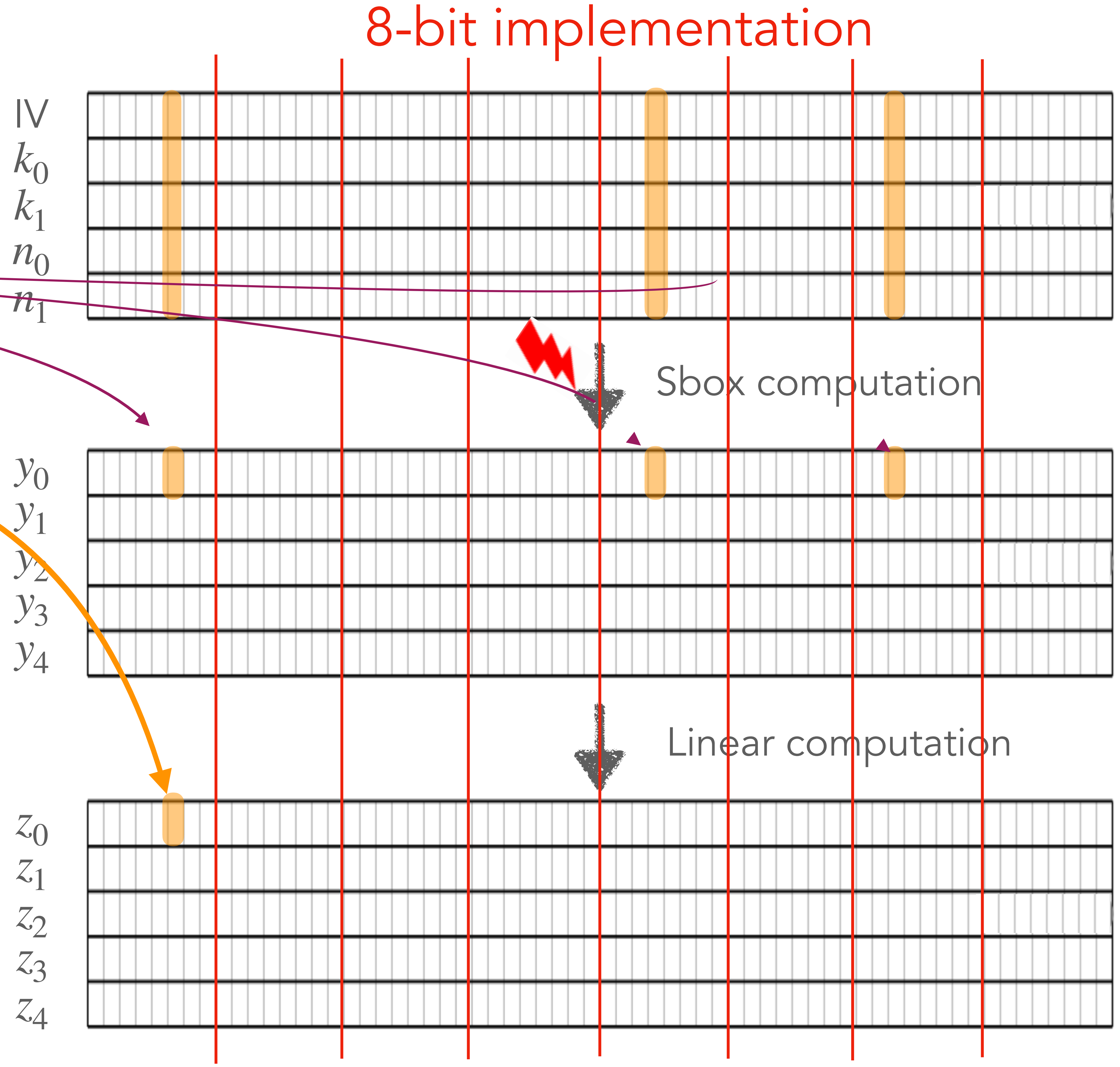
An instruction skip
does not affect all 3 bits

intermediate value z_0^j

$$z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}$$

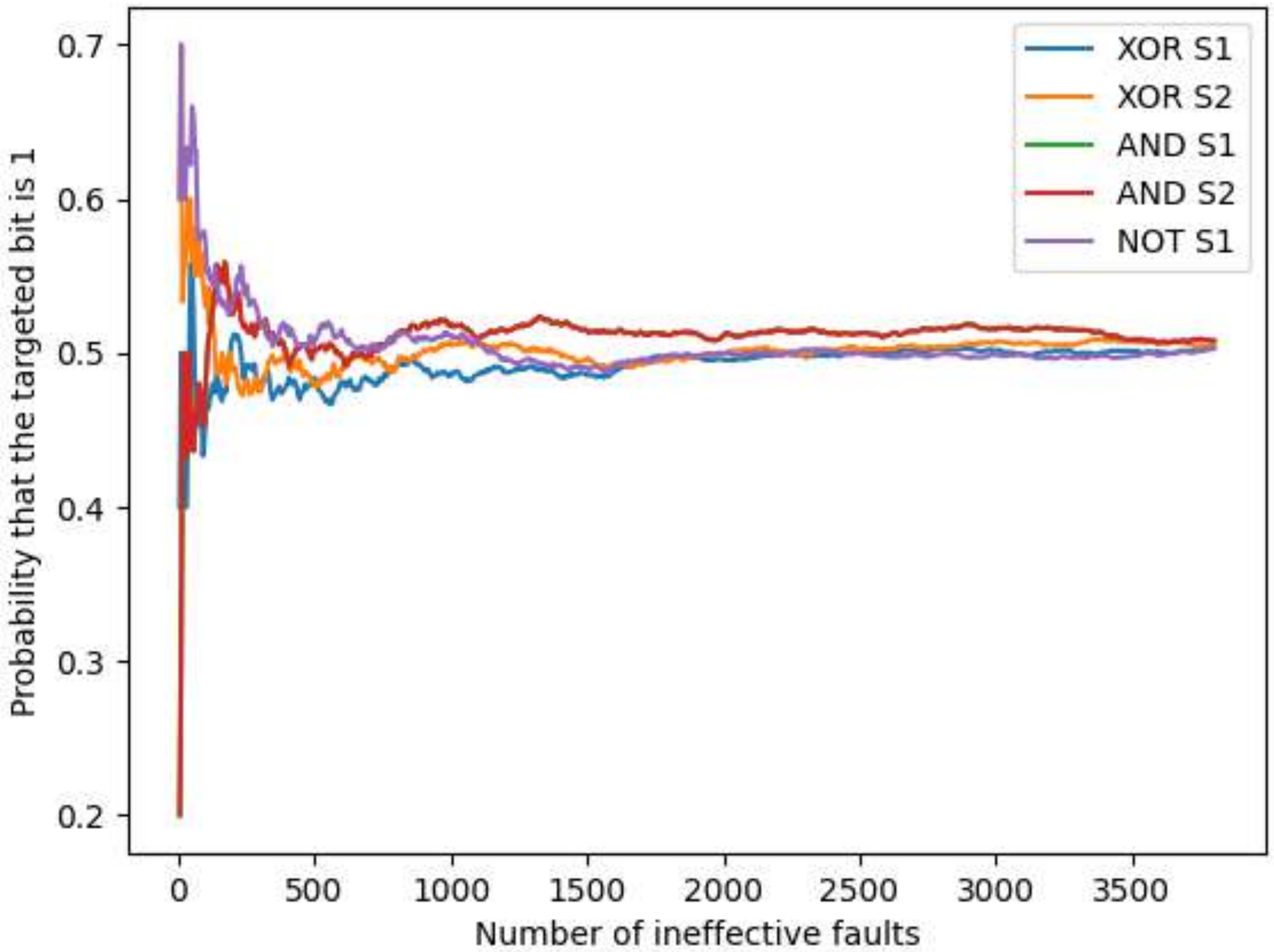
→ z_0^j remains uniform

→ SIFA is not applicable 👿



Apply to 8-bit implementation of Ascon

Apply to 8-bit implementation of Ascon



Summary

Main points

Main points

- Probability of an ineffective fault depends on
  - Instruction type
  - Architecture

Main points

- Probability of an ineffective fault depends on
  - Instruction type
  - Architecture

Other instructions (OR, ROR,...)? 🤔

Main points

- Probability of an ineffective fault depends on
  - Instruction type
  - Architecture

Other instructions (OR, ROR,...)? 🤔

Input data is not uniform? 🤔

Main points

- Probability of an ineffective fault depends on
  - Instruction type
  - Architecture

Other instructions (OR, ROR,…)? 🤔

Input data is not uniform? 🤔

- Intermediate value may not be biased
  - demonstrated on 8-bit implementation of Ascon
  - failed to apply SIFA

Main points

- Probability of an ineffective fault depends on
  - Instruction type
  - Architecture

Other instructions (OR, ROR,…)? 🤔

Input data is not uniform? 🤔

- Intermediate value may not be biased
  - demonstrated on 8-bit implementation of Ascon
  - failed to apply SIFA

Choose another intermediate value? 🤔

SIFA on Nonce-based Authenticated Encryption: When does it fail? Application to Ascon

Viet-Sang Nguyen

joint work with Vincent Grosso and Pierre-Louis Cayrel

FDTC

Kuala Lumpur, 14 September 2025

UNIVERSITÉ DE LYON | UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Laboratoire Hubert Curien UMR • CNRS • 5516 • Saint-Étienne

PROPHY ANR-22-CE39-0008-01