Practical Second-Order CPA Attack on Ascon with Proper Selection Function

Viet-Sang Nguyen

joint work with Vincent Grosso and Pierre-Louis Cayrel

CASCADE Conference

Saint-Etienne, 2 April, 2025

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

PROPHY ANR-22-CE39-0008-01

2018 Lightweight cryptography competition

NIST

2018 Lightweight cryptography competition

2023 Selected Ascon

2018 Lightweight cryptography competition

2023 Selected Ascon

2018 Lightweight cryptography competition

2023 Selected Ascon

Possible attacks

Secure implementations

In this talk

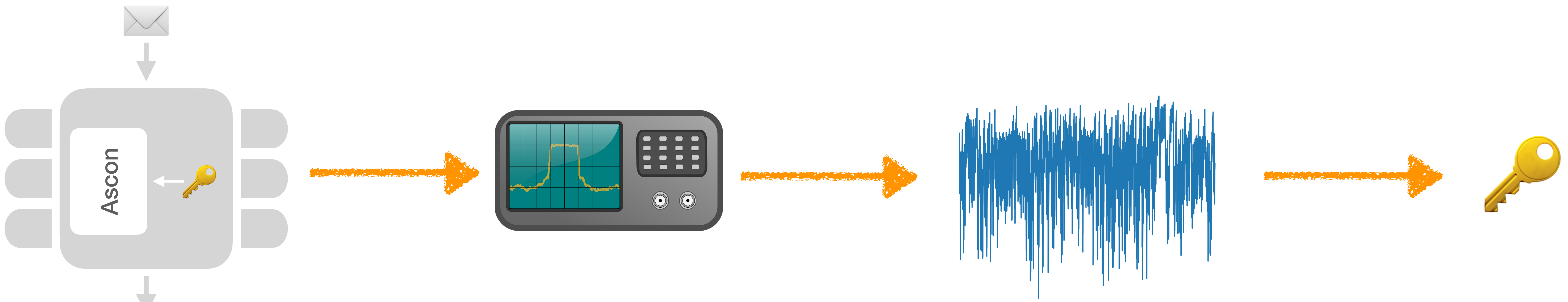
Possible attacks

In this talk

Possible attacks

Correlation Power Analysis (CPA) attack

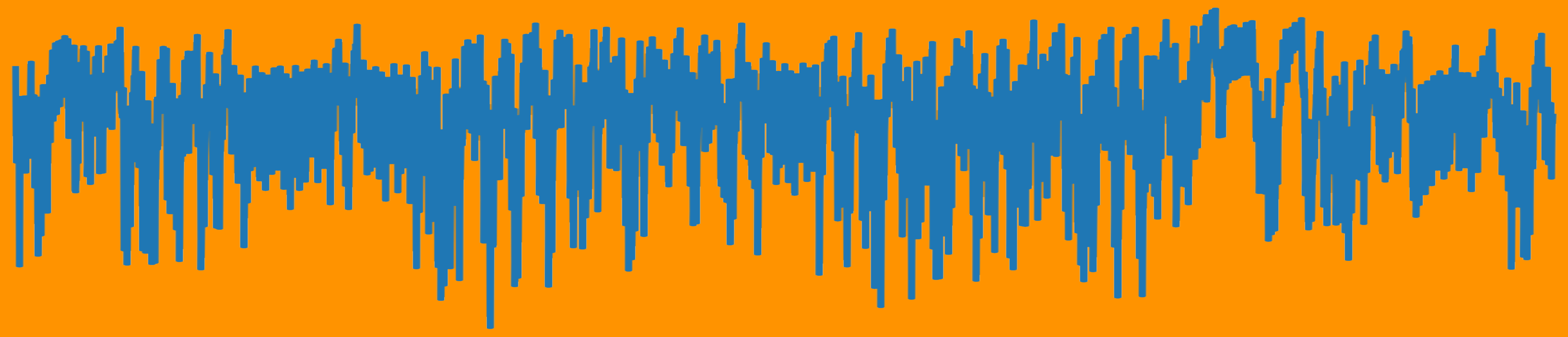
In this talk



Possible attacks

Correlation Power Analysis (CPA) attack

CPA attack



CPA attack

CPA attack

- Choose attack point: intermediate variable v

CPA attack

- Choose attack point: intermediate variable v

Selection function: $v = f(d, k)$

known non-constant data

part of the key

CPA attack

- Choose attack point: intermediate variable v

Selection function: $v = f(d, k)$

known non-constant data

part of the key

Well-known CPA on AES: $v = \text{Sbox}(\text{plaintext}, \text{key})$

CPA attack

- Choose attack point: intermediate variable v

Selection function: $v = f(d, k)$

known non-constant data part of the key

Well-known CPA on AES: $v = \text{Sbox}(\text{plaintext}, \text{key})$

- Choose leakage model for v

CPA attack

- Choose attack point: intermediate variable v

Selection function: $v = f(d, k)$

known non-constant data   part of the key

Well-known CPA on AES: $v = \mathrm{Sbox}(\mathrm{plaintext}, \mathrm{key})$

- Choose leakage model for v

This work: Hamming weight

CPA attack

- Choose attack point: intermediate variable v

Selection function: $v = f(d, k)$

known non-constant data

part of the key

Well-known CPA on AES: $v = \mathrm{Sbox}(\mathrm{plaintext}, \mathrm{key})$

- Choose leakage model for v

This work: Hamming weight

Hypothetical power consumption: $h = \mathrm{HW}(v) = \mathrm{HW}\left(f(d, k)\right)$

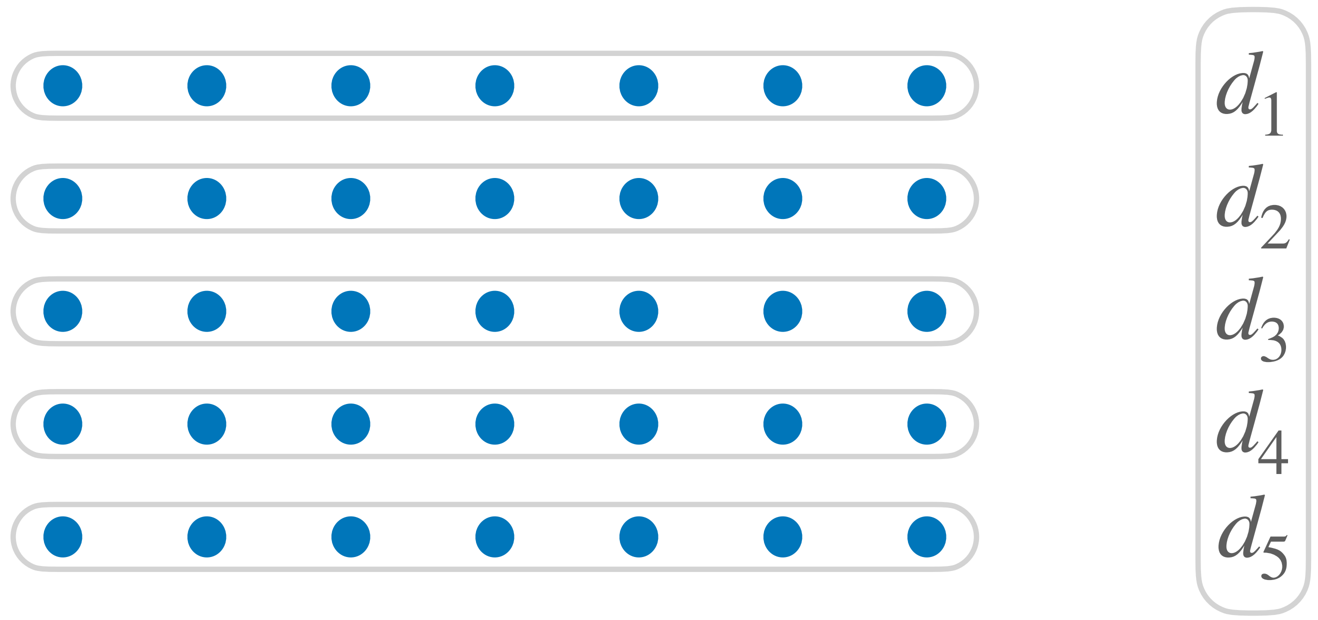
CPA attack

CPA attack

- Measure power consumption traces and record d

CPA attack

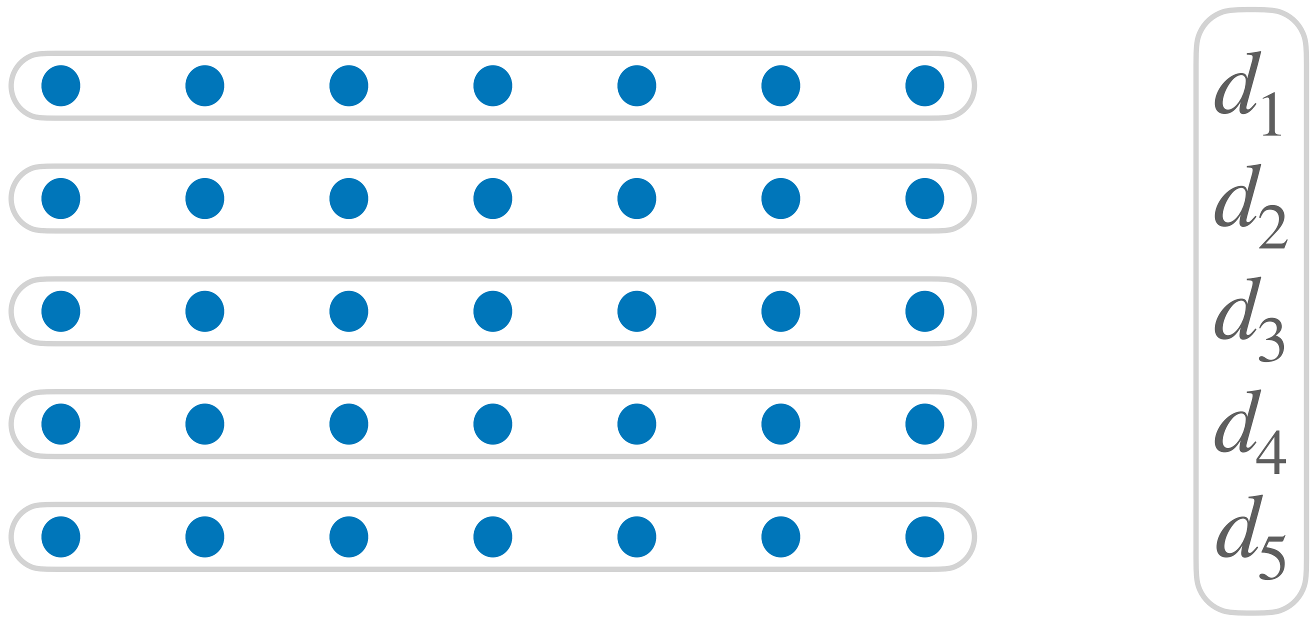
- Measure power consumption traces and record d



CPA attack

- Measure power consumption traces and record d

- Compute hypothetical power consumption

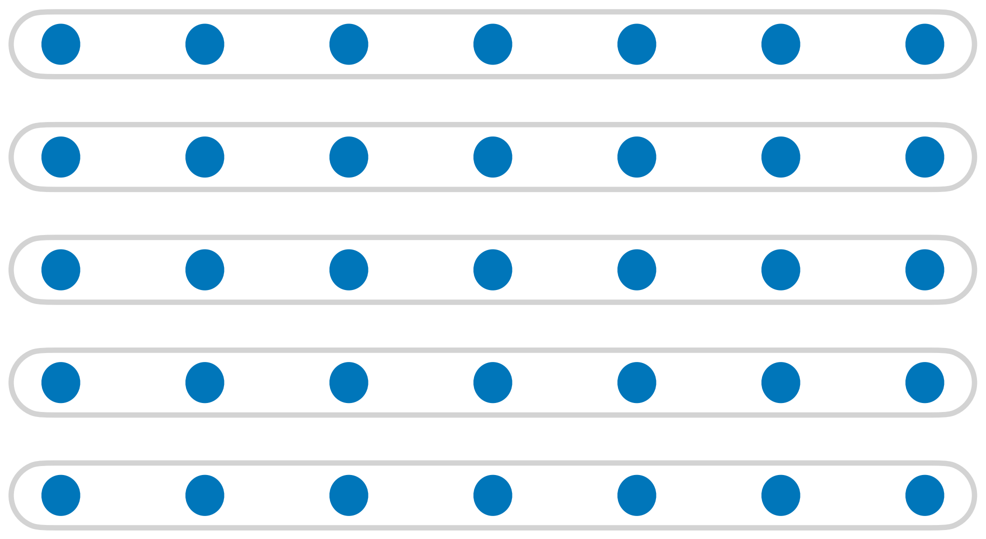


CPA attack

- Measure power consumption traces and record d

- Compute hypothetical power consumption

$$h = \mathrm{HW}\left(f(d, k)\right)$$



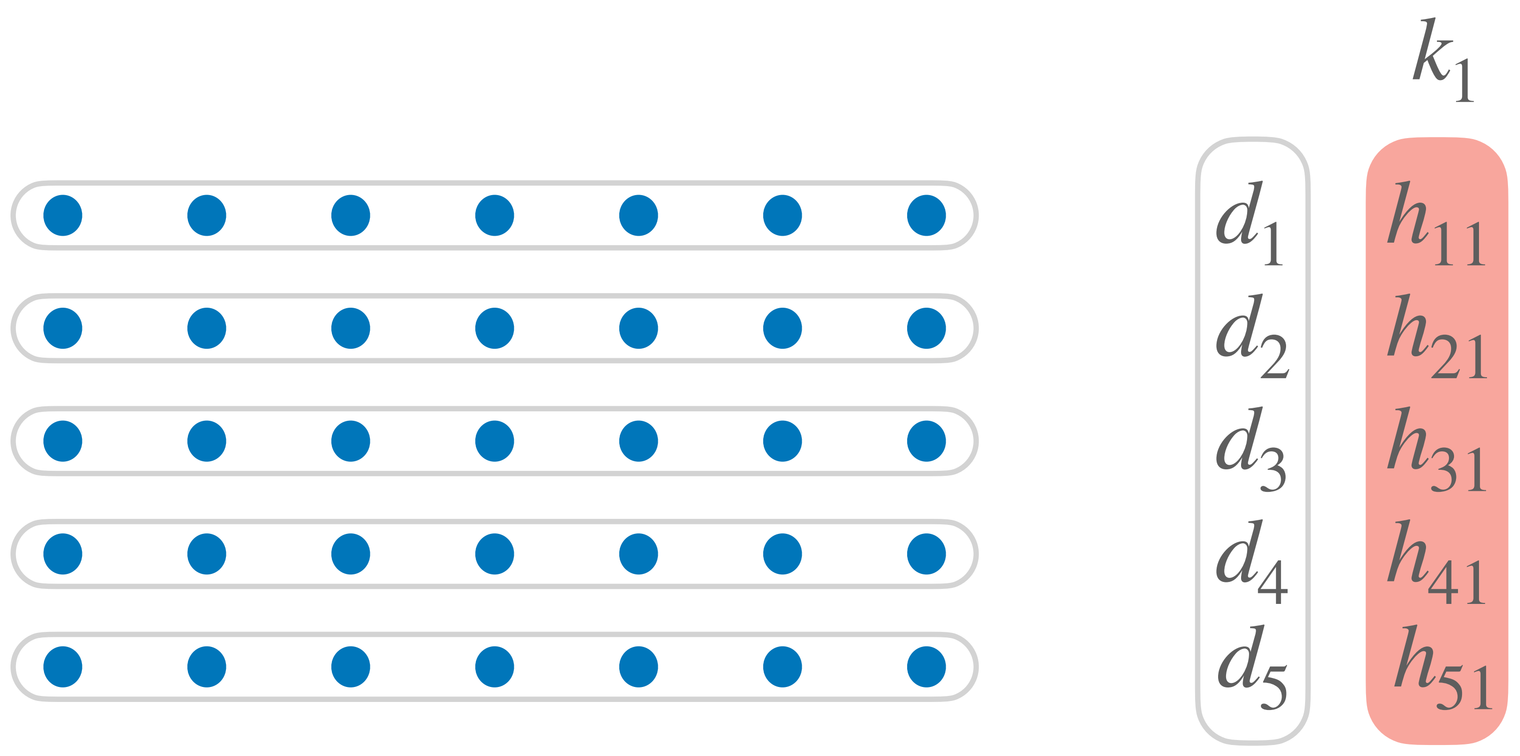
$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix}$$

CPA attack

- Measure power consumption traces and record d

- Compute hypothetical power consumption

$h = \text{HW}\left(f(d,k)\right)$

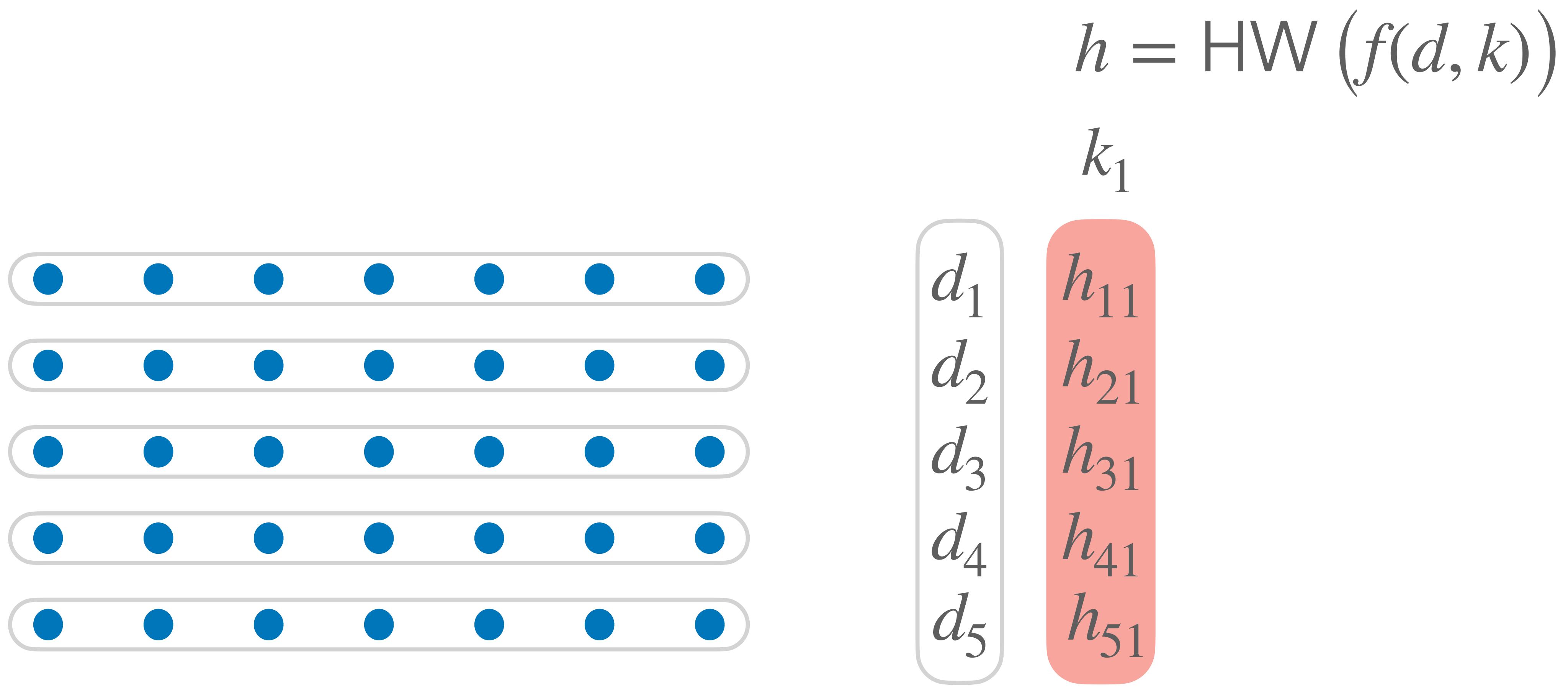


CPA attack

- Measure power consumption traces and record d

- Compute hypothetical power consumption

- Compare with measured power consumption

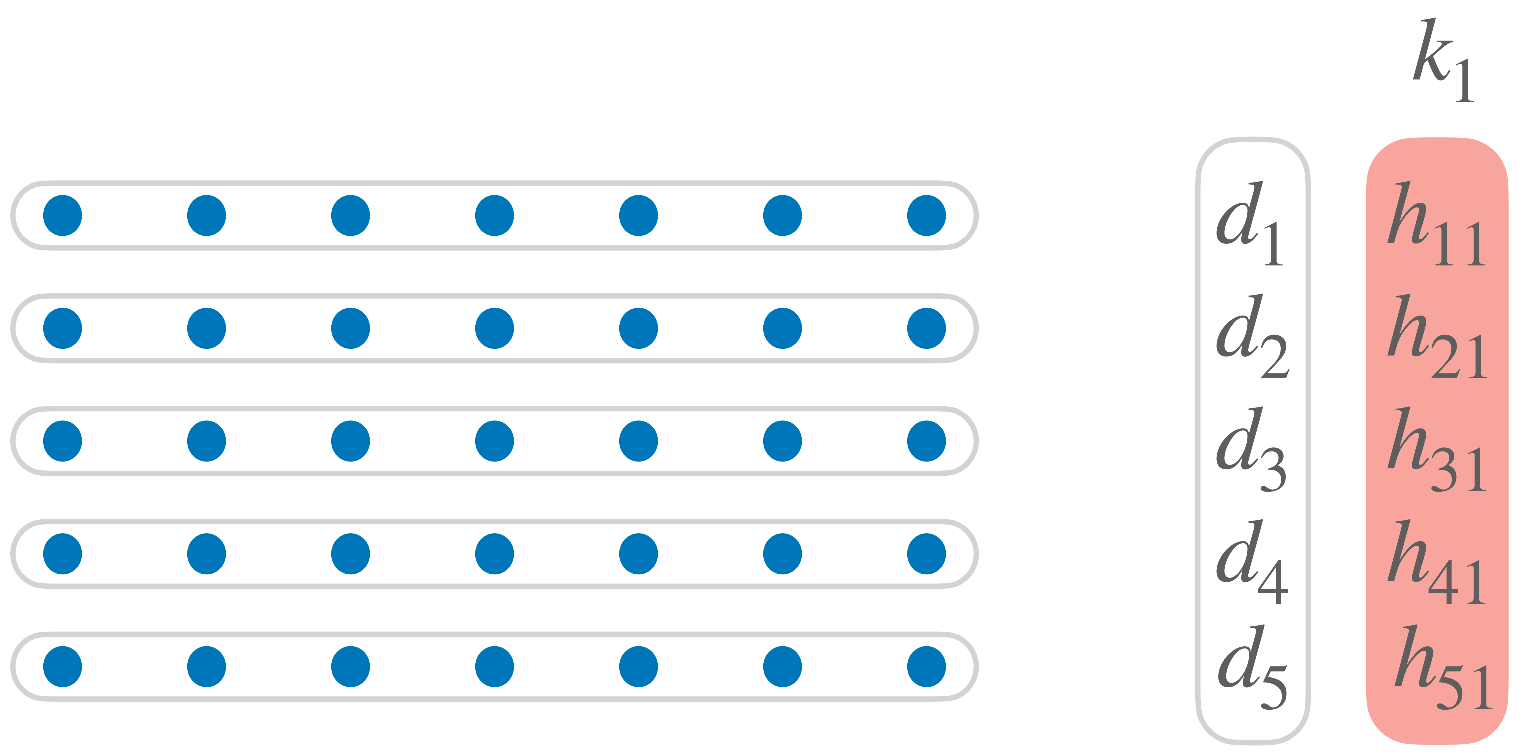


CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

  Linearity relationship
  with Pearson's correlation coefficient

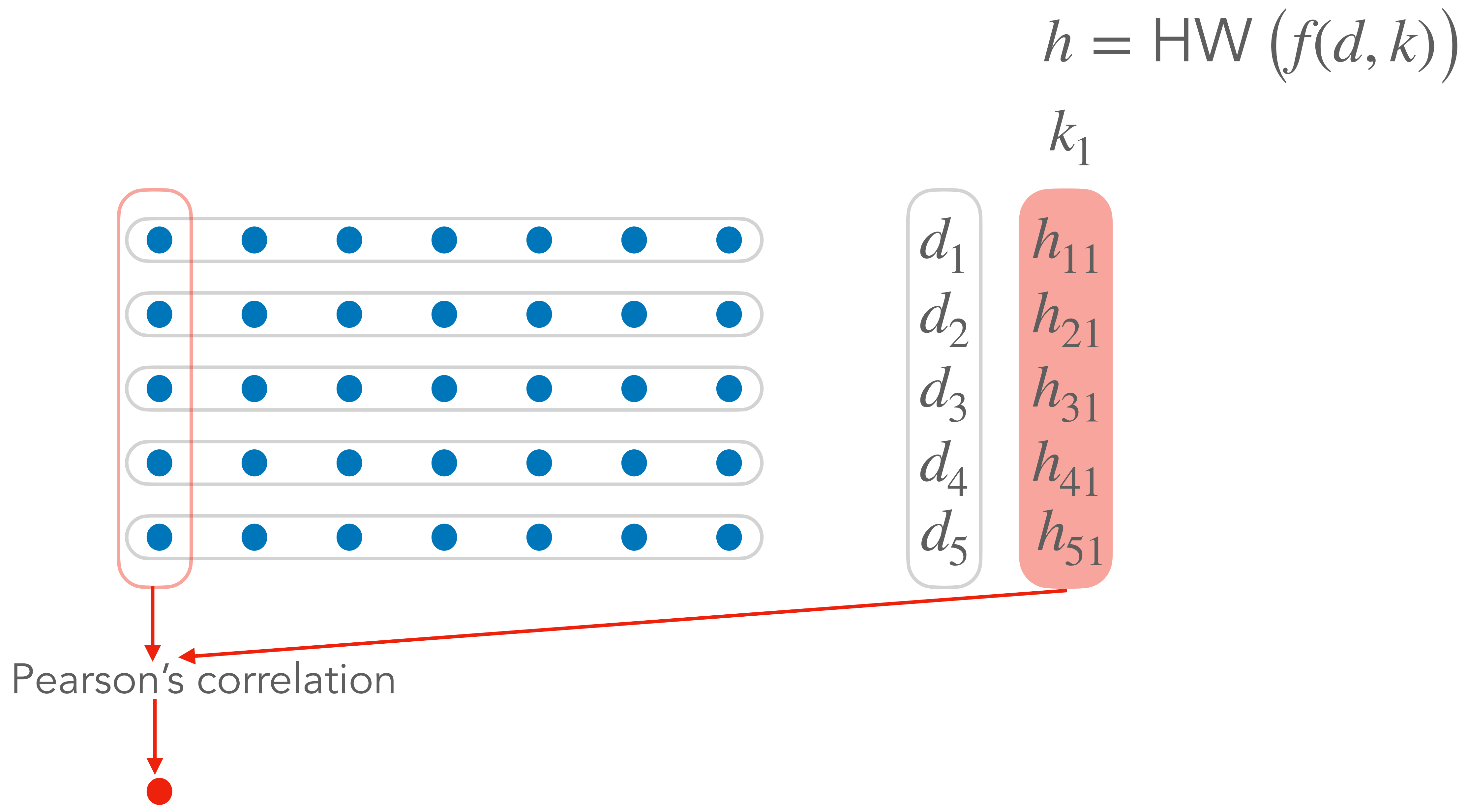
$h = \mathrm{HW}\left(f(d,k)\right)$



CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient



CPA attack

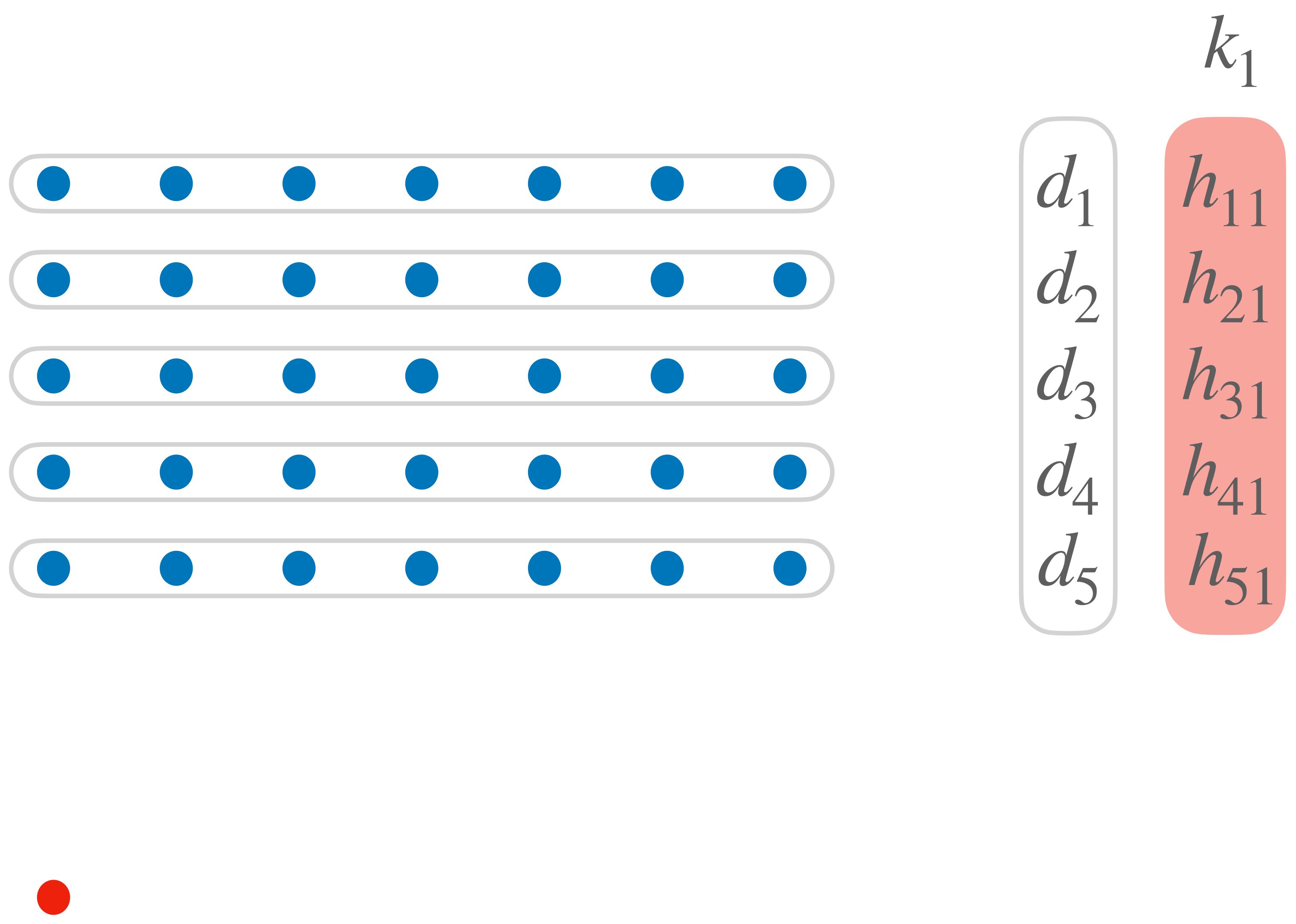
- Measure power consumption traces and record d

- Compute hypothetical power consumption

- Compare with measured power consumption

  Linearity relationship
  with Pearson's correlation coefficient

$h = \mathrm{HW}\left(f(d,k)\right)$



CPA attack

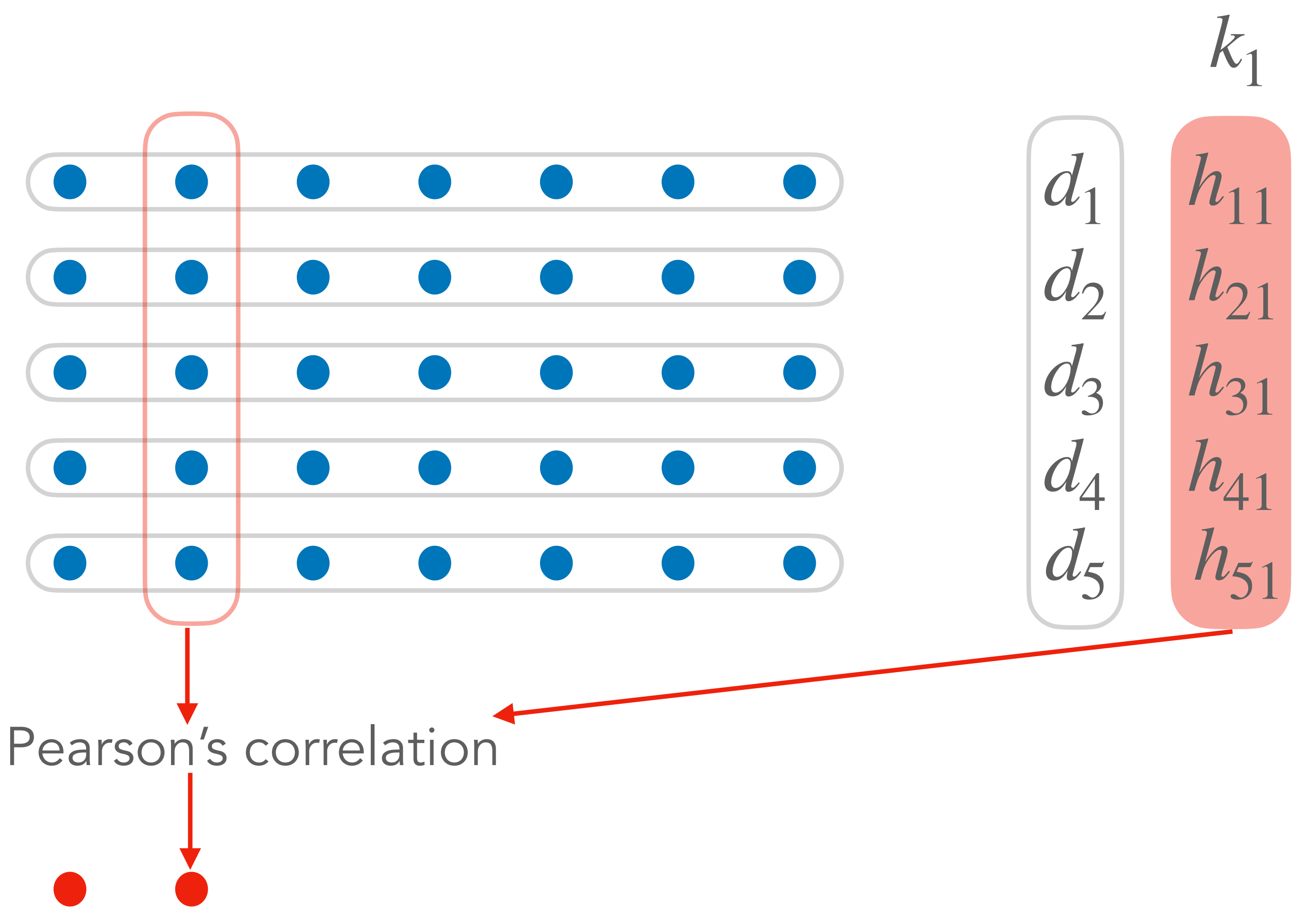
- Measure power consumption traces and record d

- Compute hypothetical power consumption

- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient

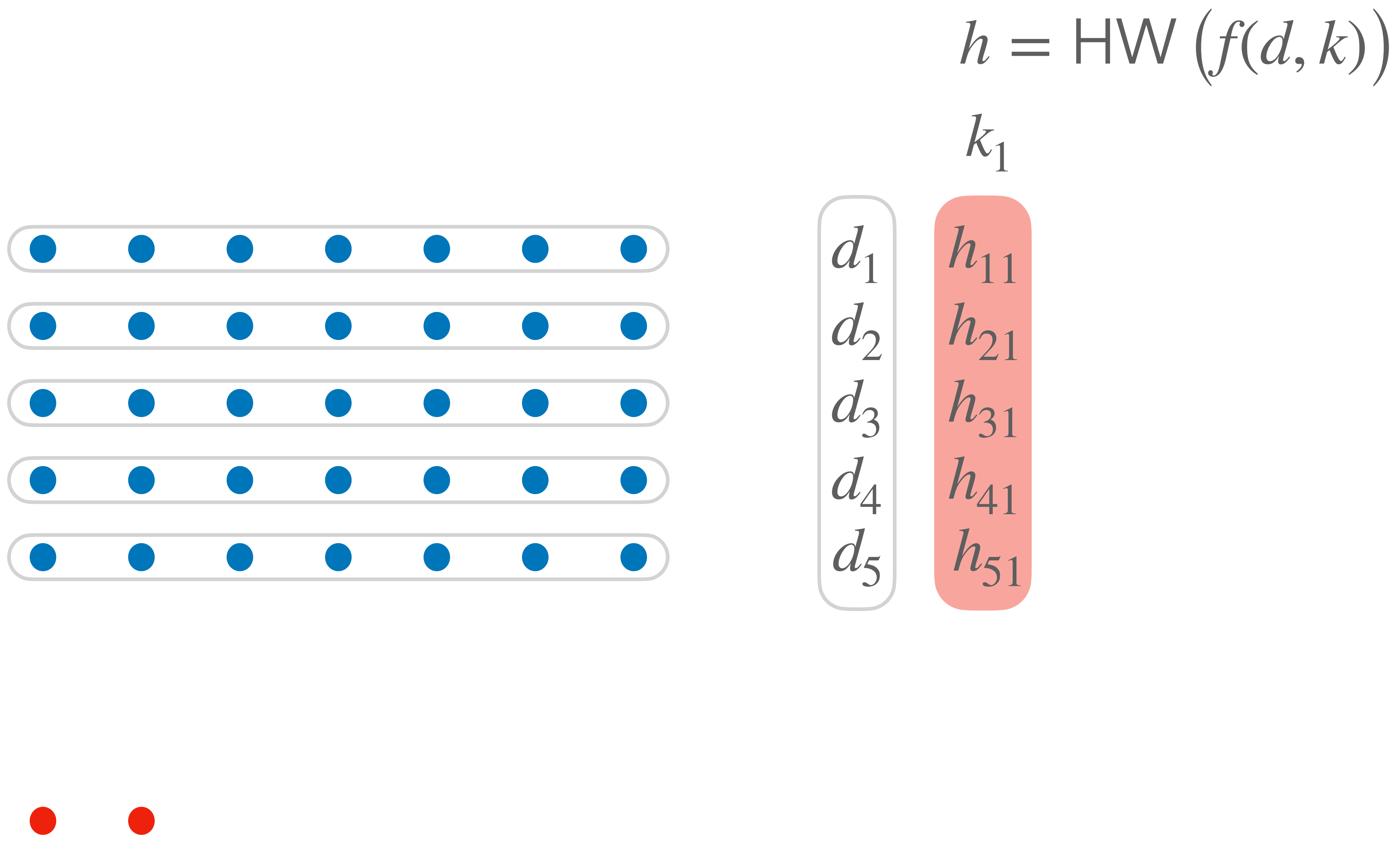
$h = \text{HW}\left(f(d,k)\right)$



CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient

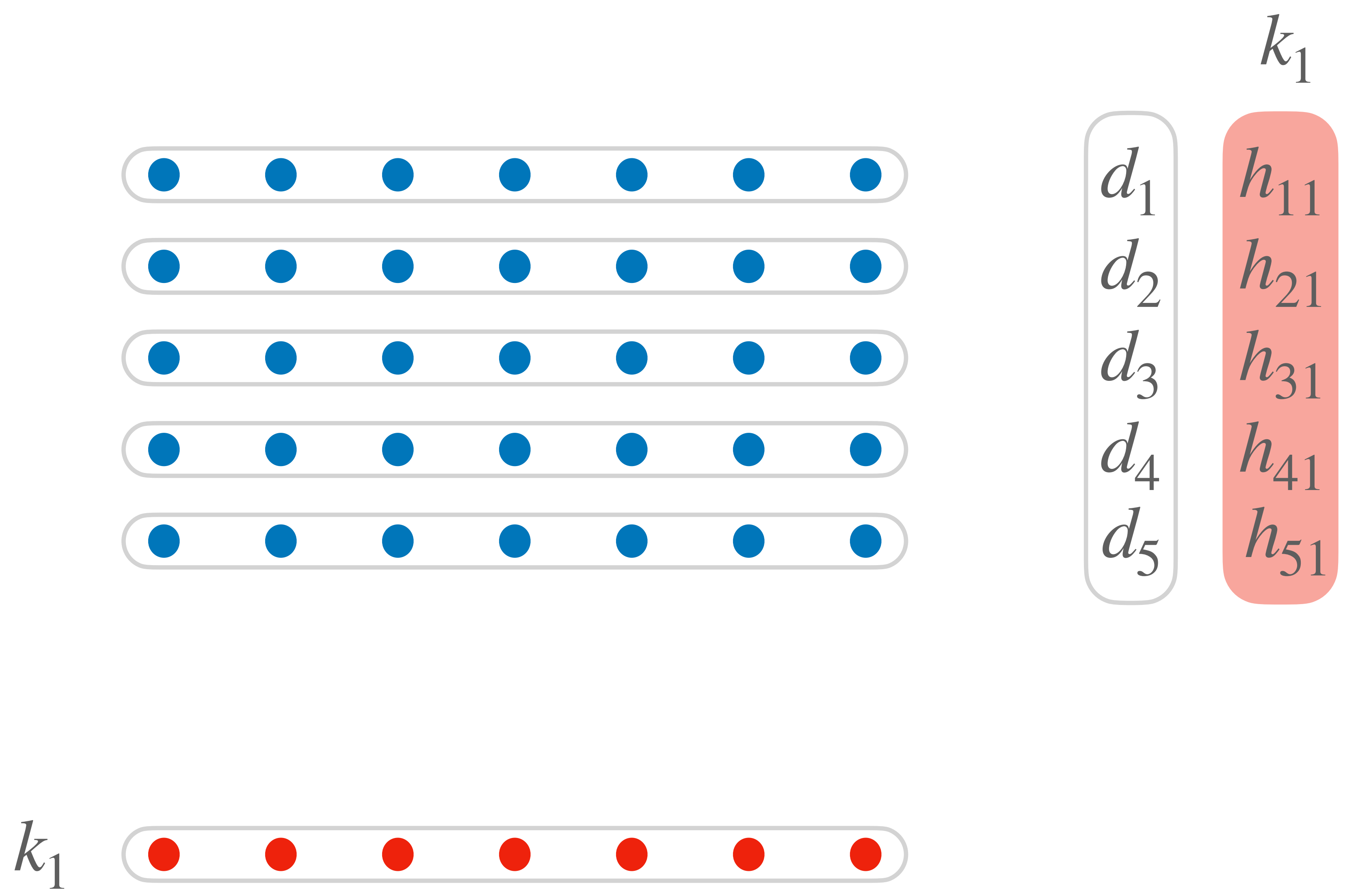


CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient

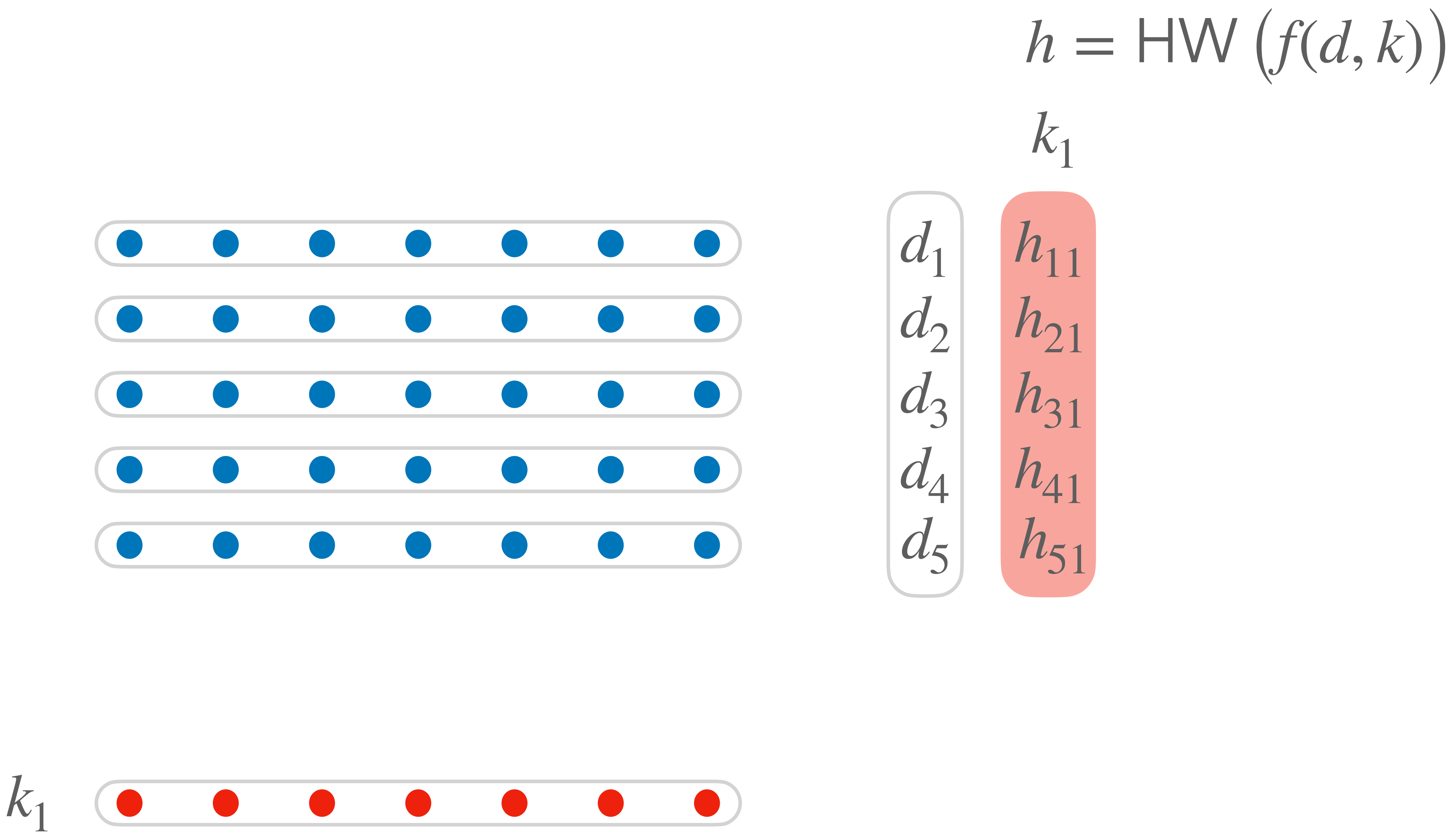
$$h = \mathrm{HW}\left(f(d,k)\right)$$



CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

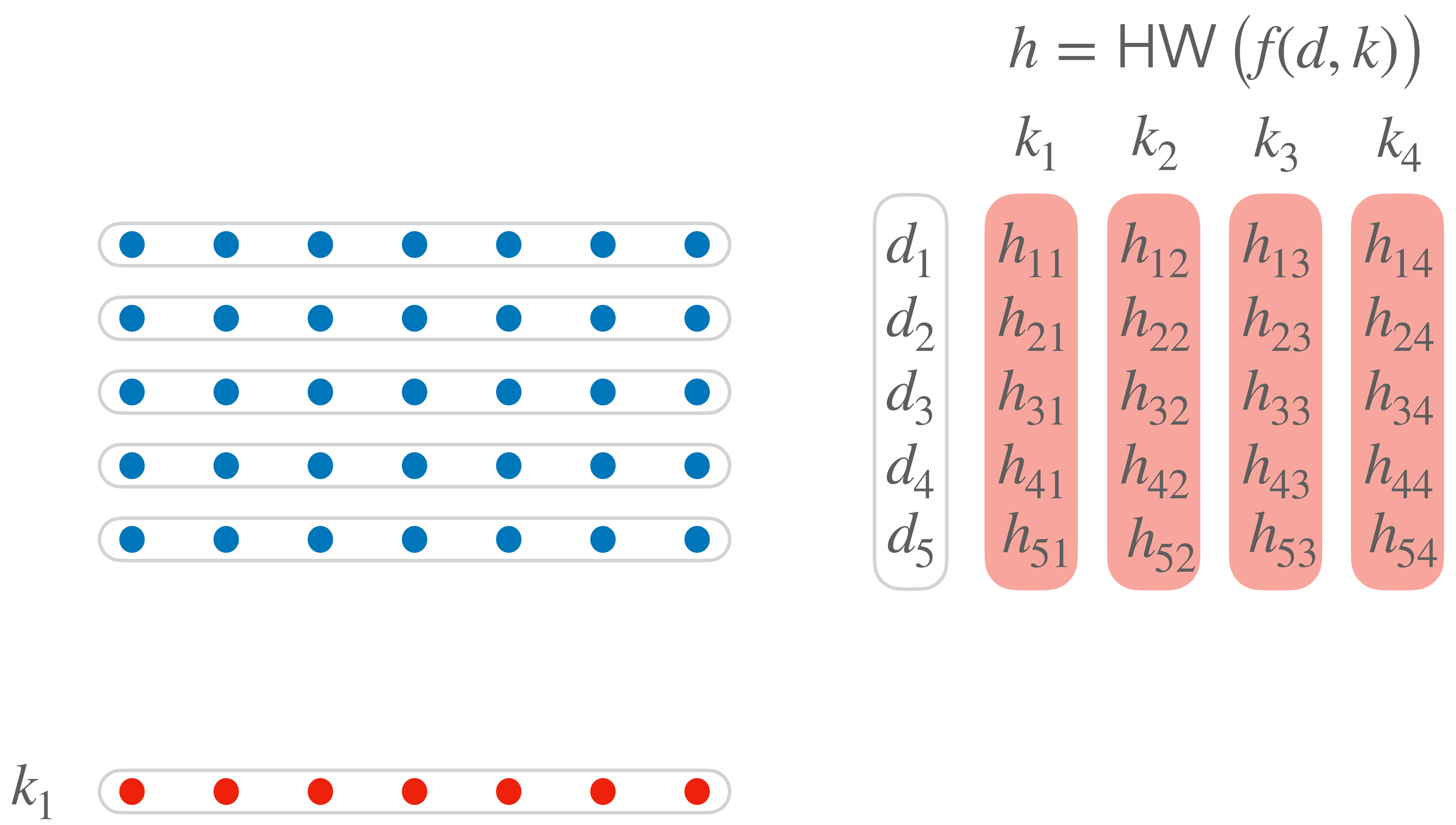
Linearity relationship
with Pearson's correlation coefficient



CPA attack

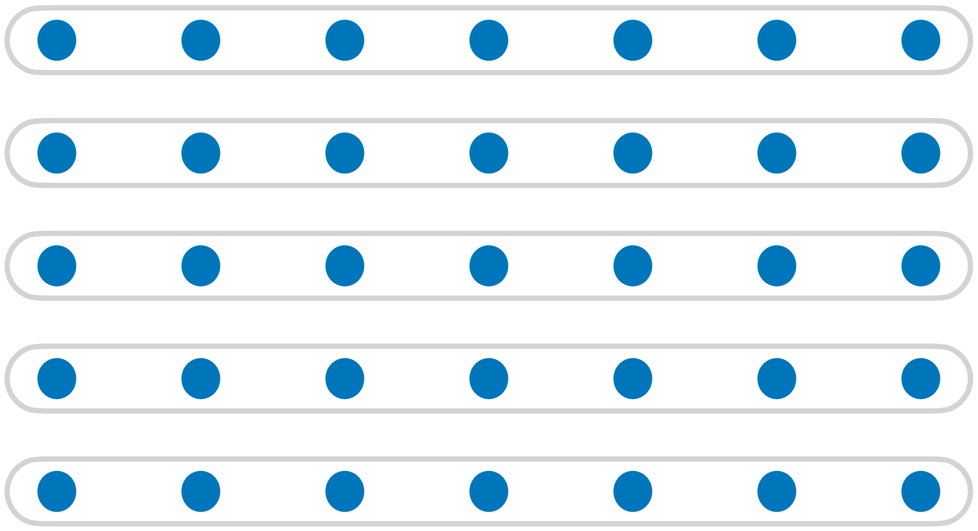
- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient



CPA attack

- Measure power consumption traces and record d



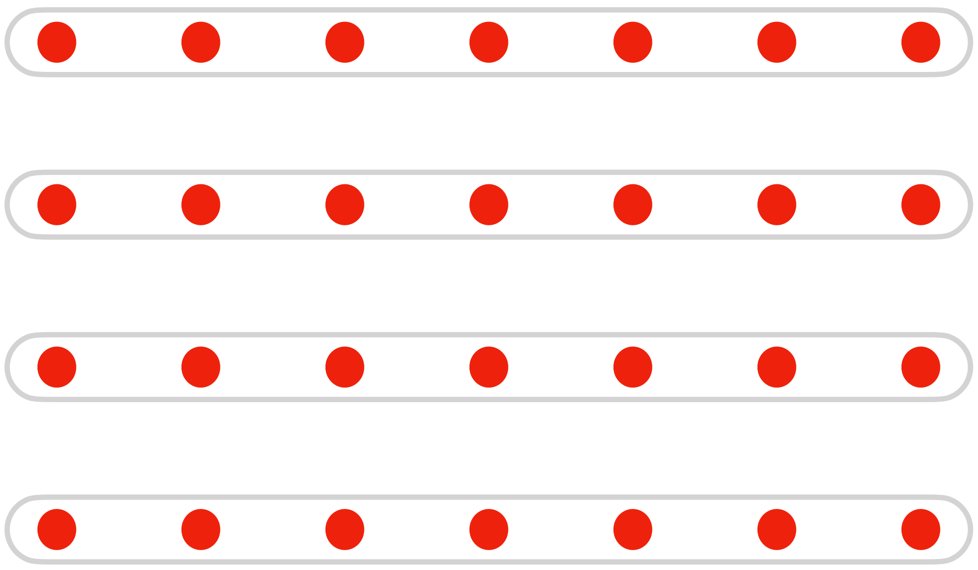
- Compute hypothetical power consumption

$$h = \mathrm{HW}\left(f(d,k)\right)$$

	k_1	k_2	k_3	k_4
d_1	h_{11}	h_{12}	h_{13}	h_{14}
d_2	h_{21}	h_{22}	h_{23}	h_{24}
d_3	h_{31}	h_{32}	h_{33}	h_{34}
d_4	h_{41}	h_{42}	h_{43}	h_{44}
d_5	h_{51}	h_{52}	h_{53}	h_{54}

- Compare with measured power consumption

k_1
k_2
k_3
k_4



Linearity relationship
with Pearson's correlation coefficient

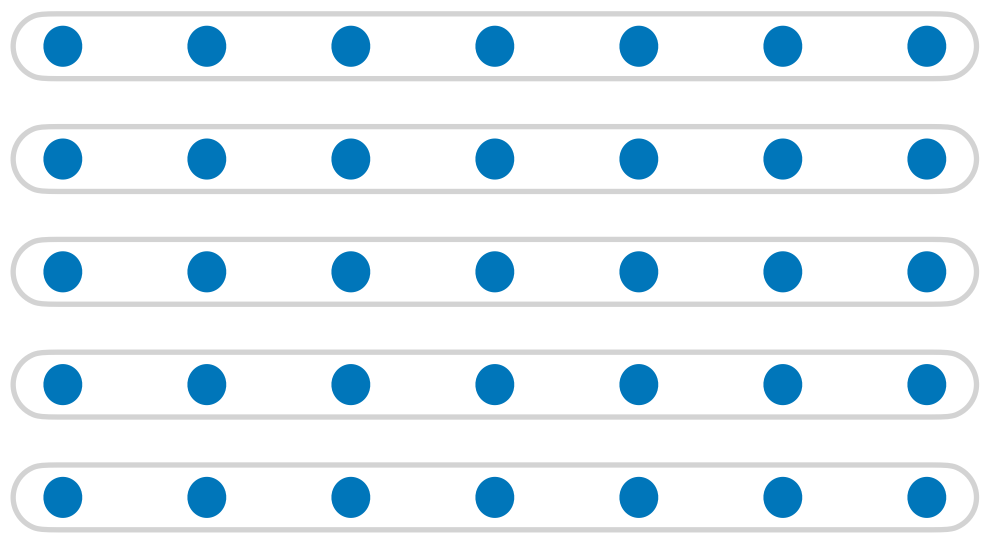
CPA attack

- Measure power consumption traces and record d

- Compute hypothetical power consumption

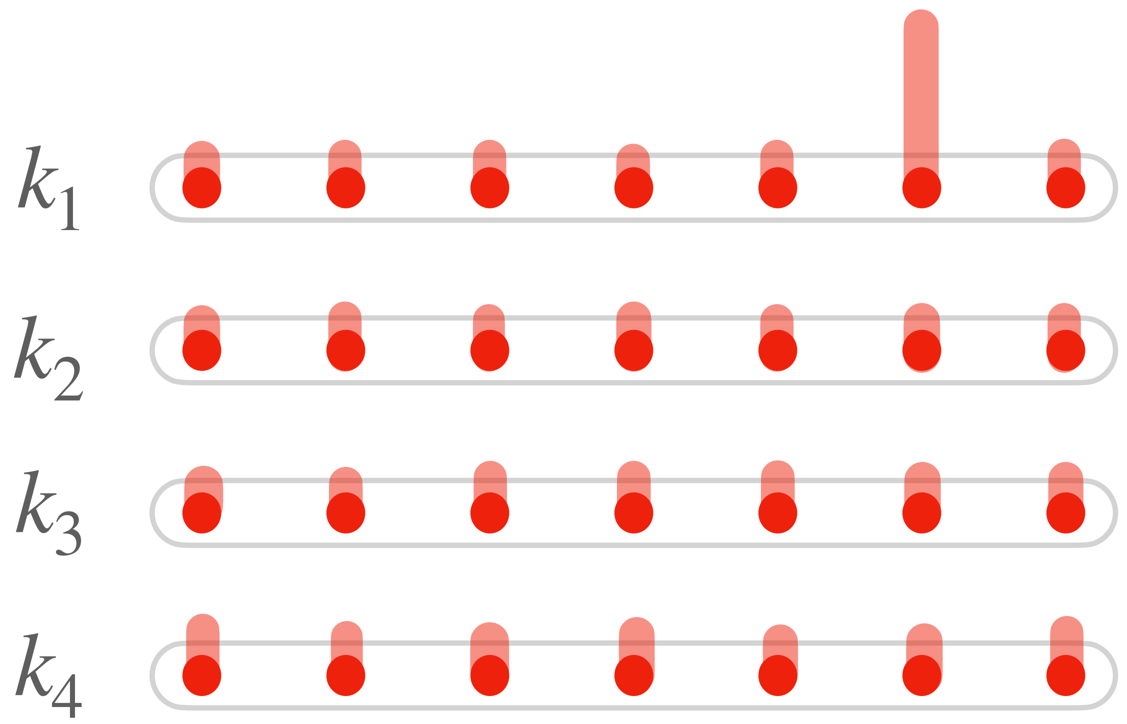
- Compare with measured power consumption

Linearity relationship
with Pearson's correlation coefficient



$h = \mathrm{HW}\left(f(d,k)\right)$

	k_1	k_2	k_3	k_4
d_1	h_{11}	h_{12}	h_{13}	h_{14}
d_2	h_{21}	h_{22}	h_{23}	h_{24}
d_3	h_{31}	h_{32}	h_{33}	h_{34}
d_4	h_{41}	h_{42}	h_{43}	h_{44}
d_5	h_{51}	h_{52}	h_{53}	h_{54}



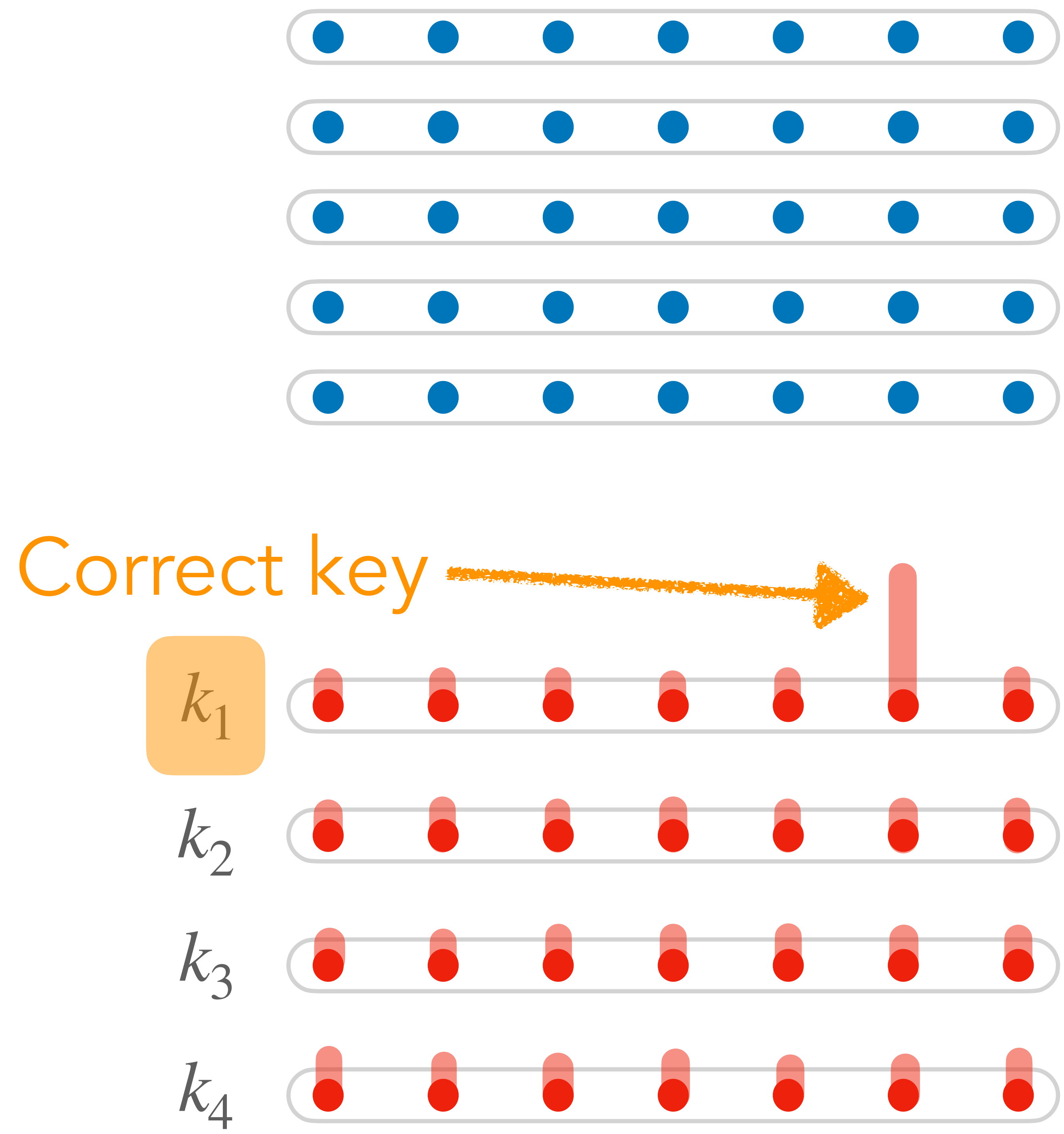
CPA attack

- Measure power consumption traces and record d
- Compute hypothetical power consumption
- Compare with measured power consumption

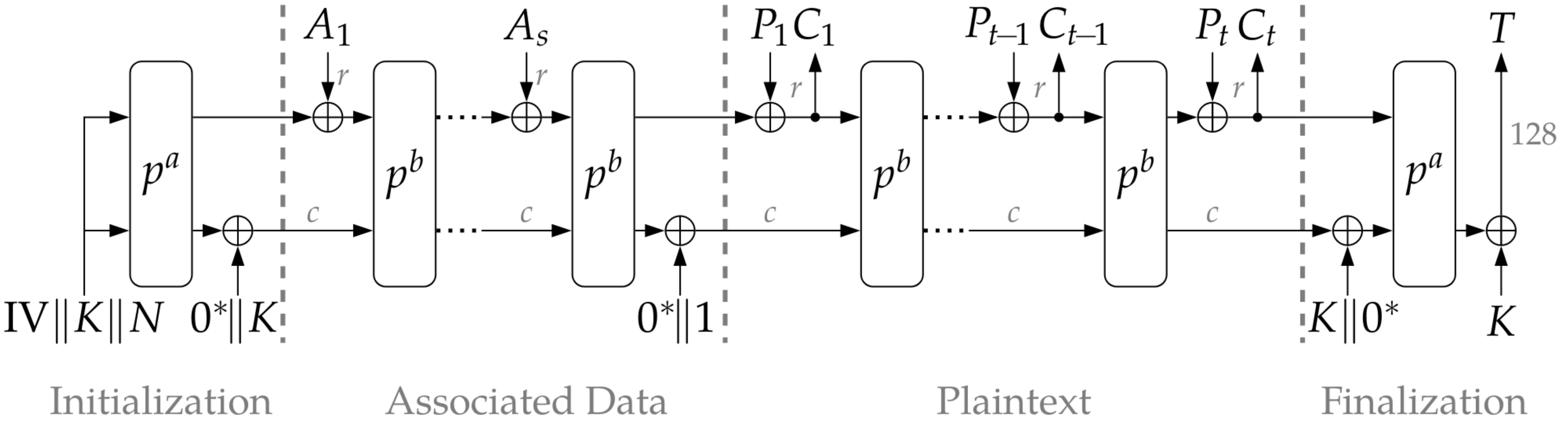
Linearity relationship
with Pearson's correlation coefficient

$$h = \mathrm{HW}\left(f(d,k)\right)$$

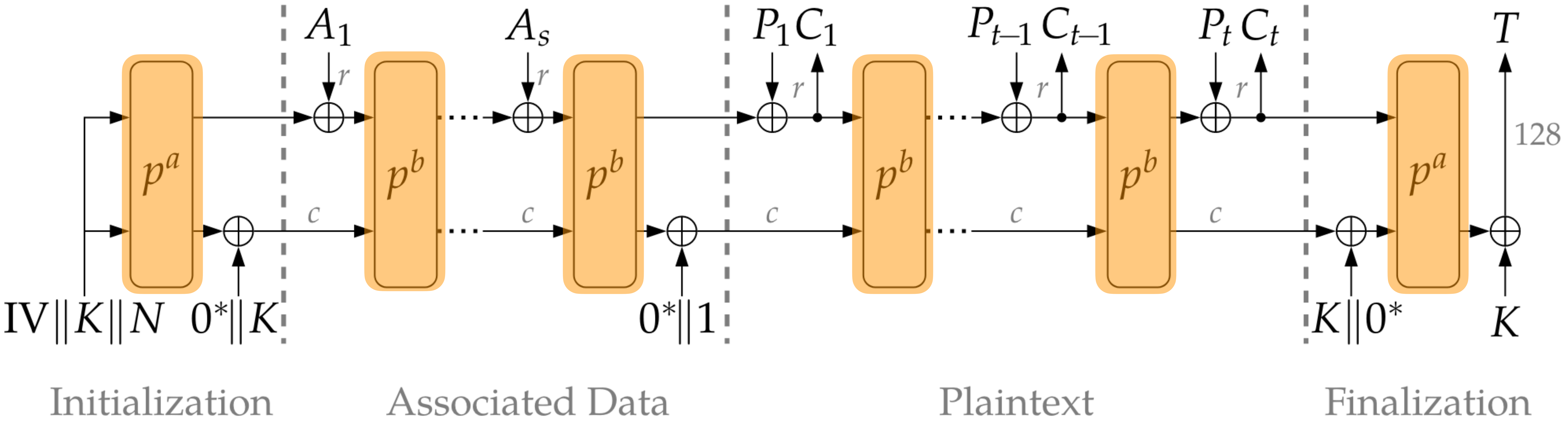
$$\begin{array}{c|cccc} & k_1 & k_2 & k_3 & k_4 \\ \hline d_1 & h_{11} & h_{12} & h_{13} & h_{14} \\ d_2 & h_{21} & h_{22} & h_{23} & h_{24} \\ d_3 & h_{31} & h_{32} & h_{33} & h_{34} \\ d_4 & h_{41} & h_{42} & h_{43} & h_{44} \\ d_5 & h_{51} & h_{52} & h_{53} & h_{54} \end{array}$$



Ascon

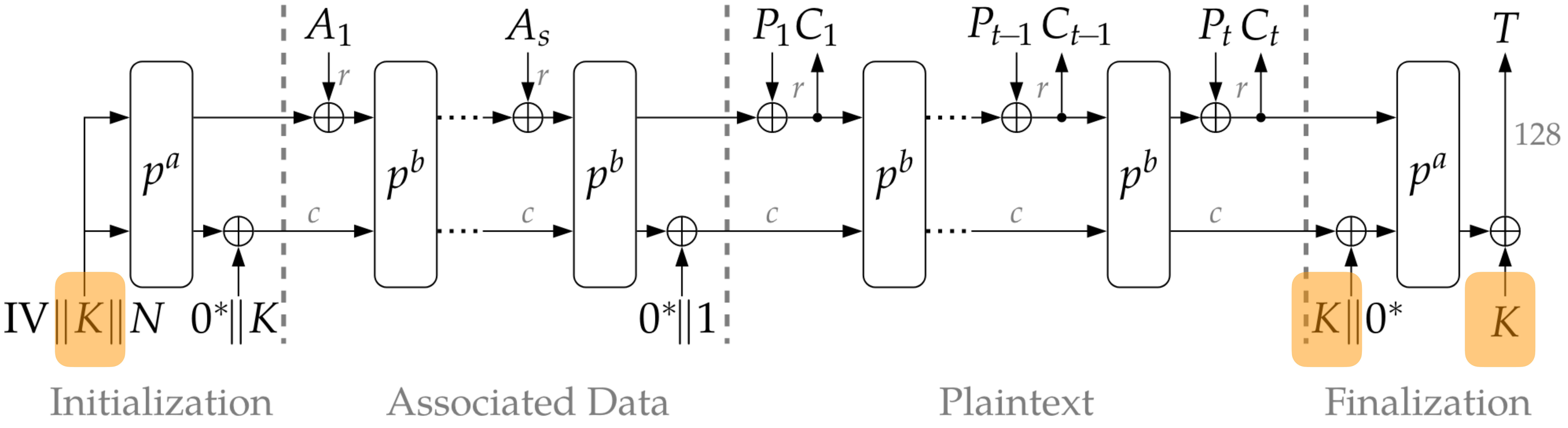

A_1
A_s
P_1 C_1
P_{t-1} C_{t-1}
P_t C_t
T
r
c
128
p^a
p^b
$\mathrm{IV}\|K\|N$
$0^*\|K$
$0^*\|1$
$K\|0^*$
K
Initialization
Associated Data
Plaintext
Finalization

Permutation blocks



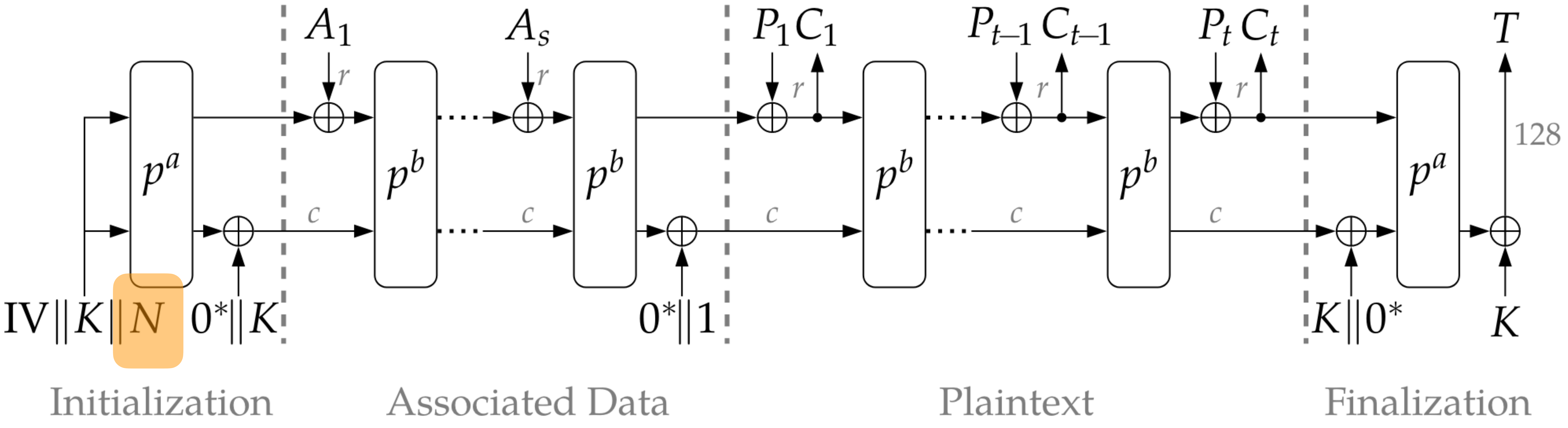
- p^a: 12 rounds
- p^b: 6 rounds

Key (128 bits)

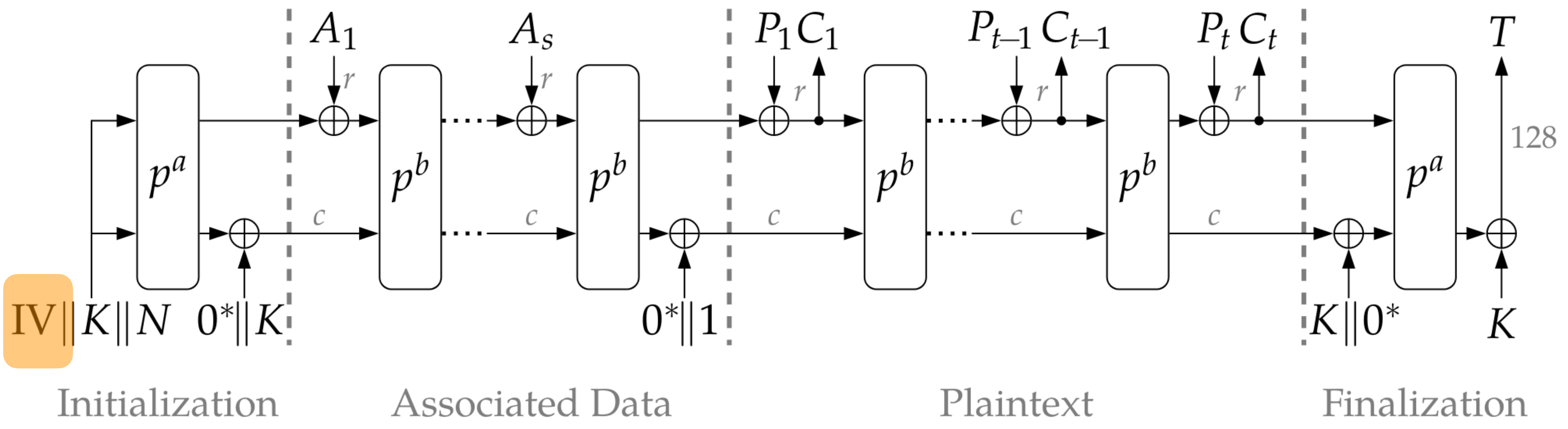


A_1
A_s
P_1 C_1
P_{t-1} C_{t-1}
P_t C_t
T
r
c
p^a
p^b
128
$IV \| K \| N$
$0^* \| K$
$0^* \| 1$
$K \| 0^*$
K
Initialization
Associated Data
Plaintext
Finalization

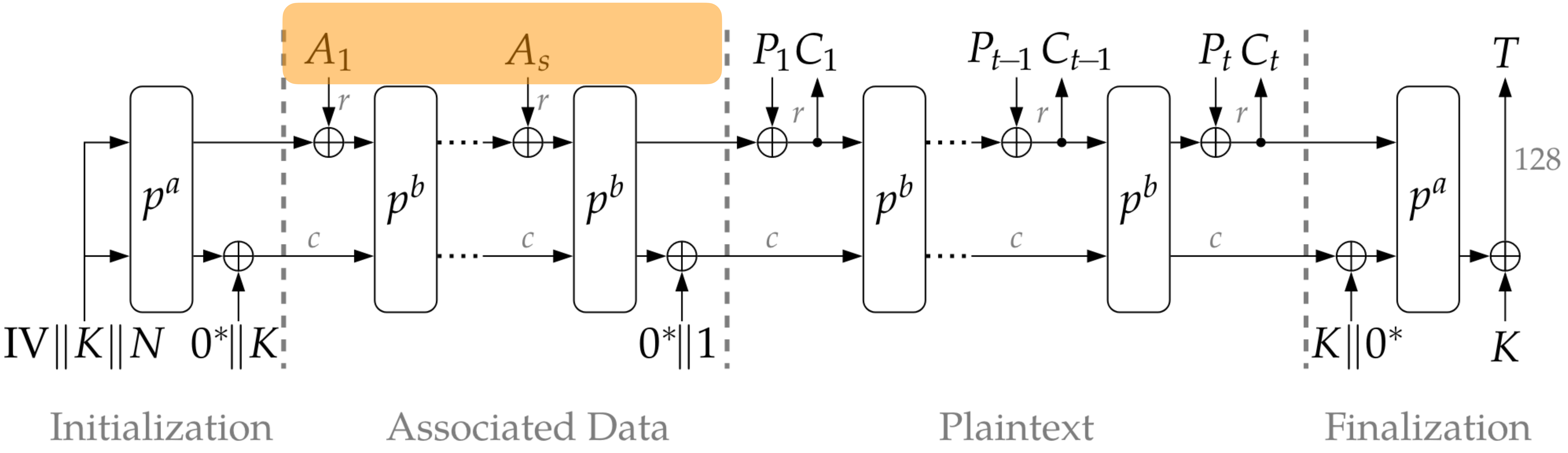
Nonce (128 bits)



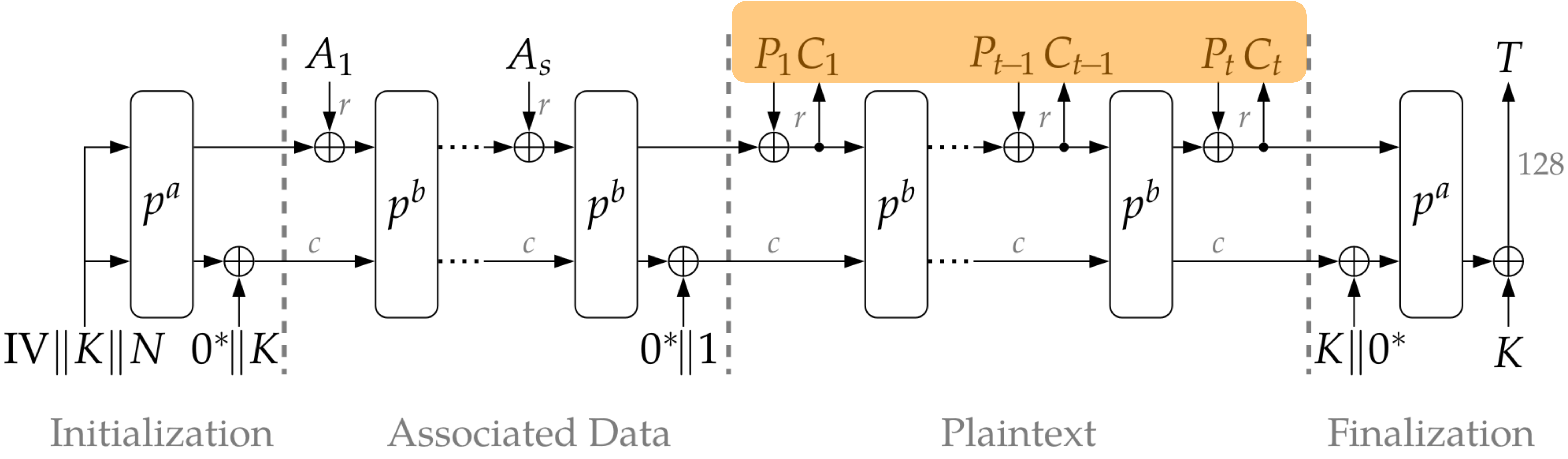
Initialization vector



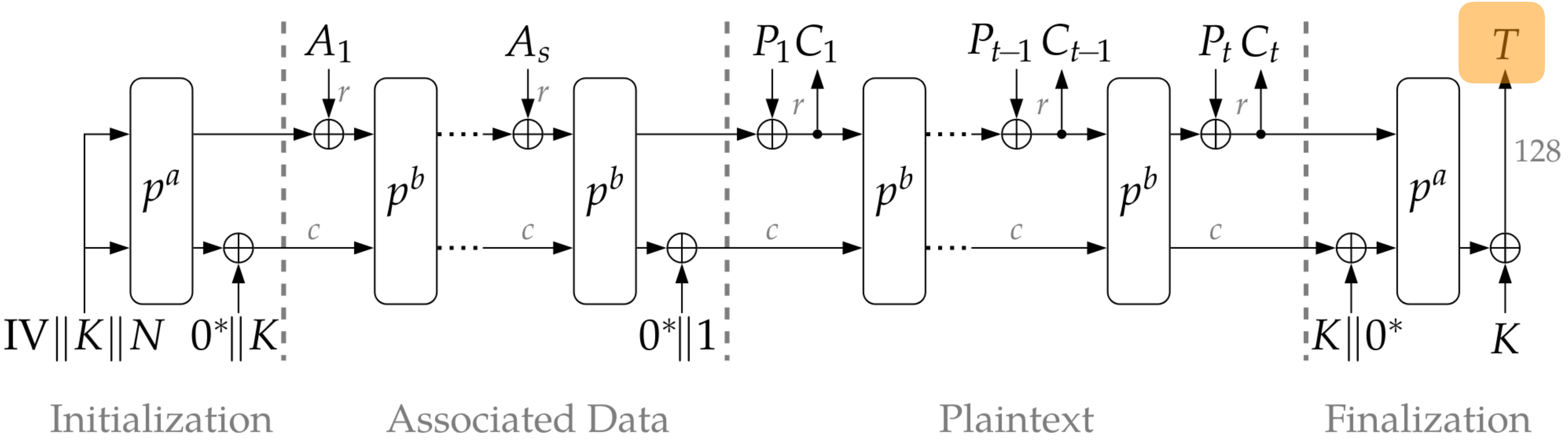
Associated data

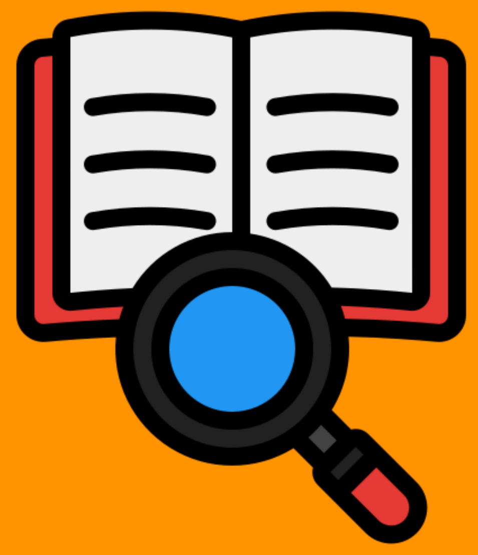


Plaintext / Ciphertext



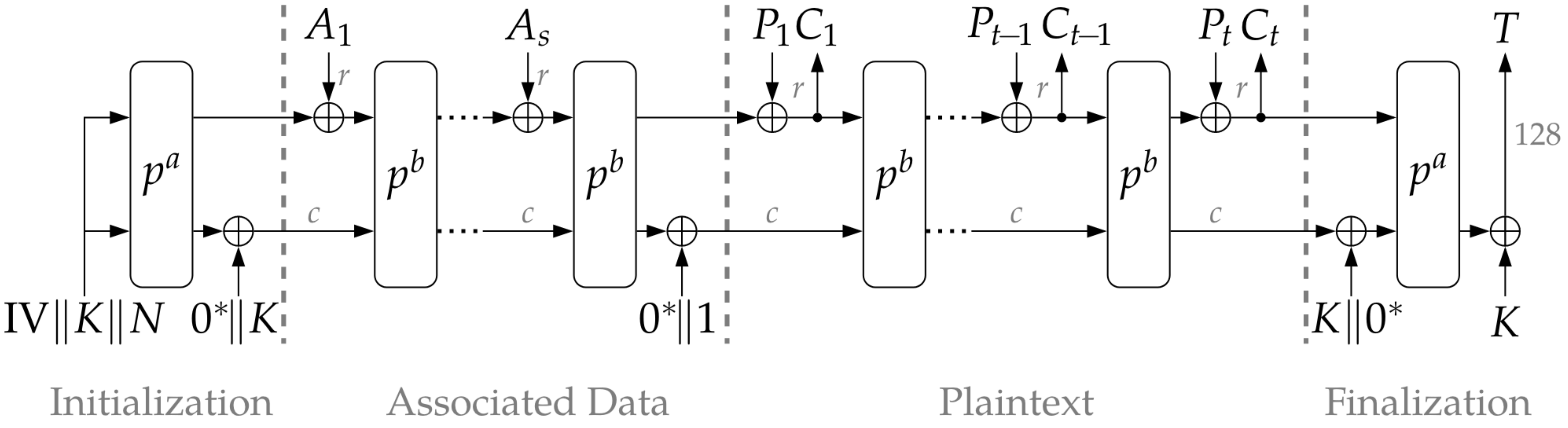
Verification Tag



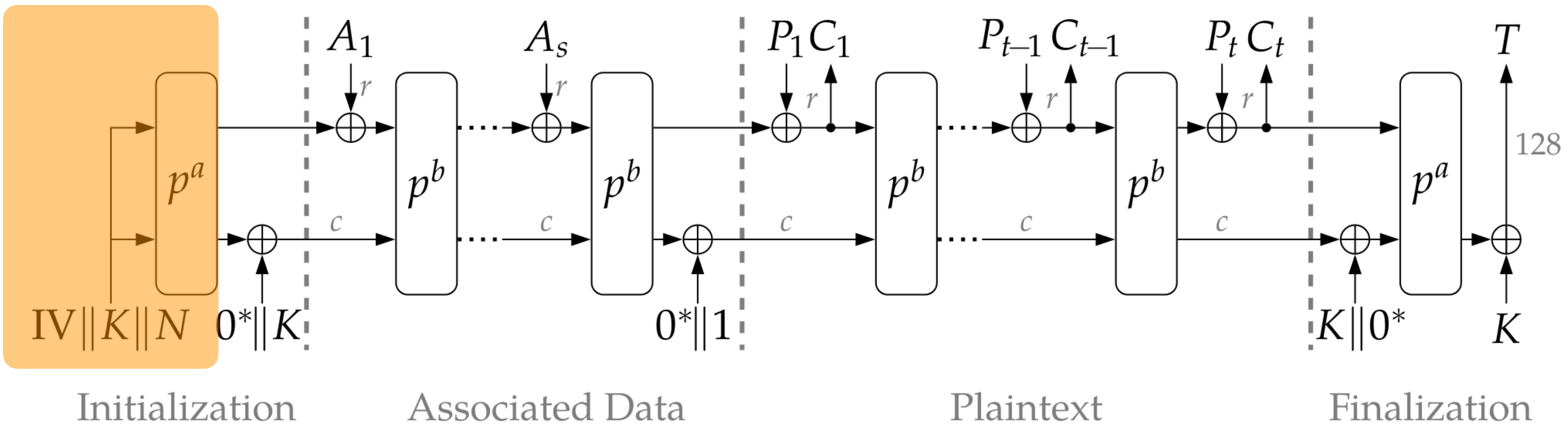


Existing CPA attacks on Ascon

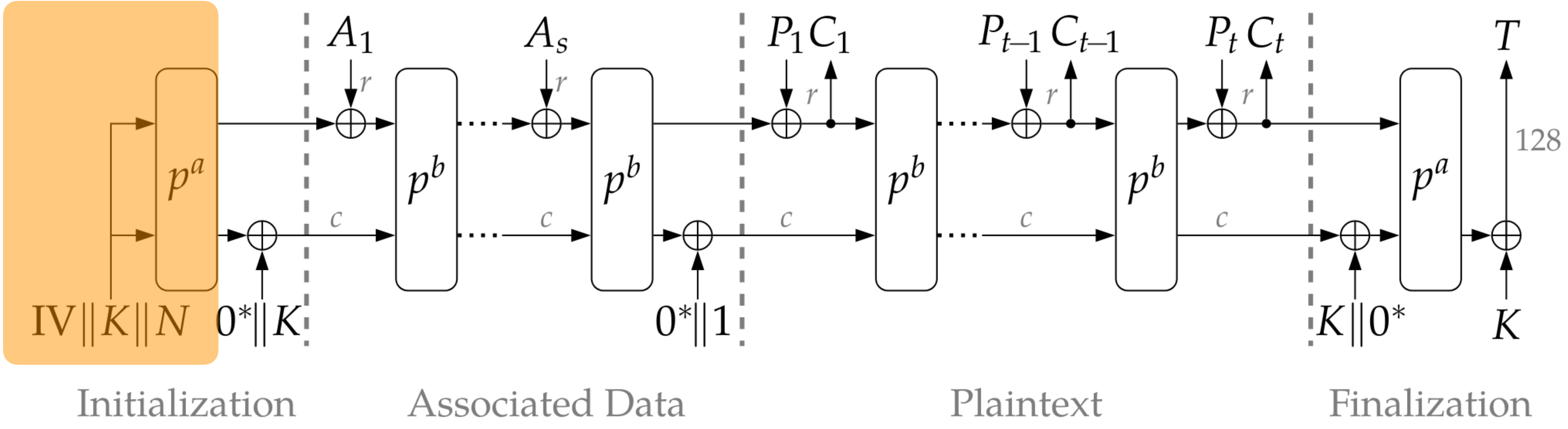
Selection function $v = f(d, k)$



Selection function $v = f(d, k)$



Selection function $v = f(d, k)$



Choose $v = f(\text{nonce}, \text{key})$

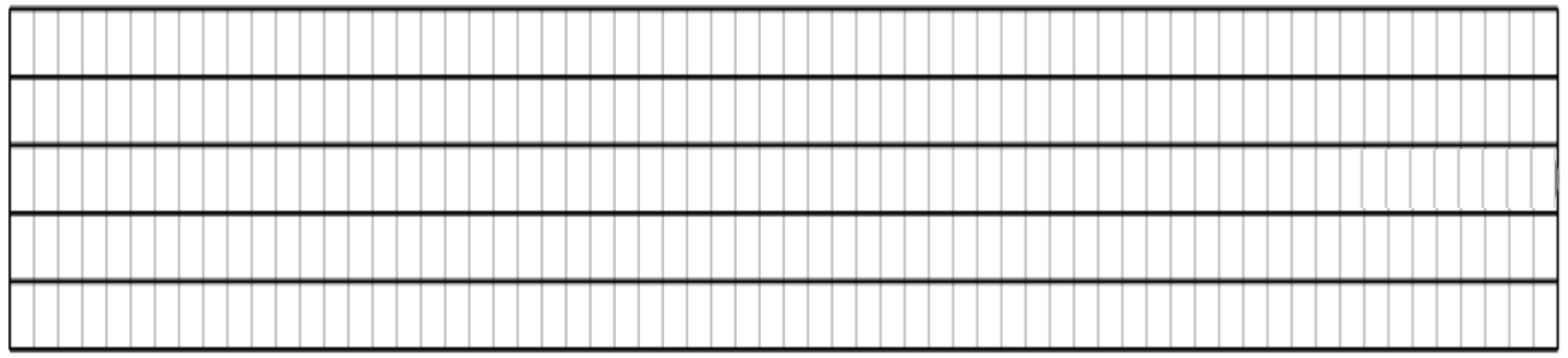
intermediate variable v in the *first round*

round computation

Round computation

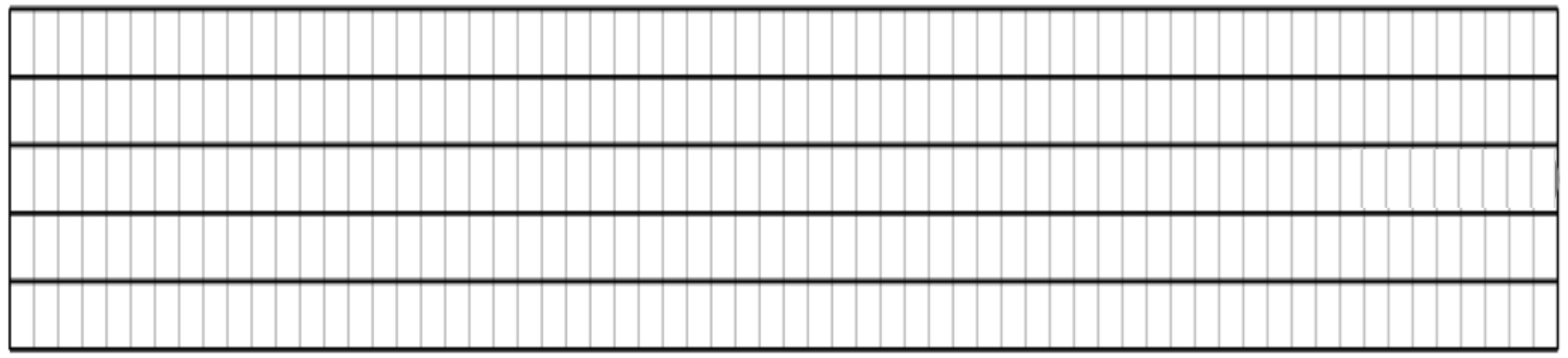
Round computation

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



Round computation

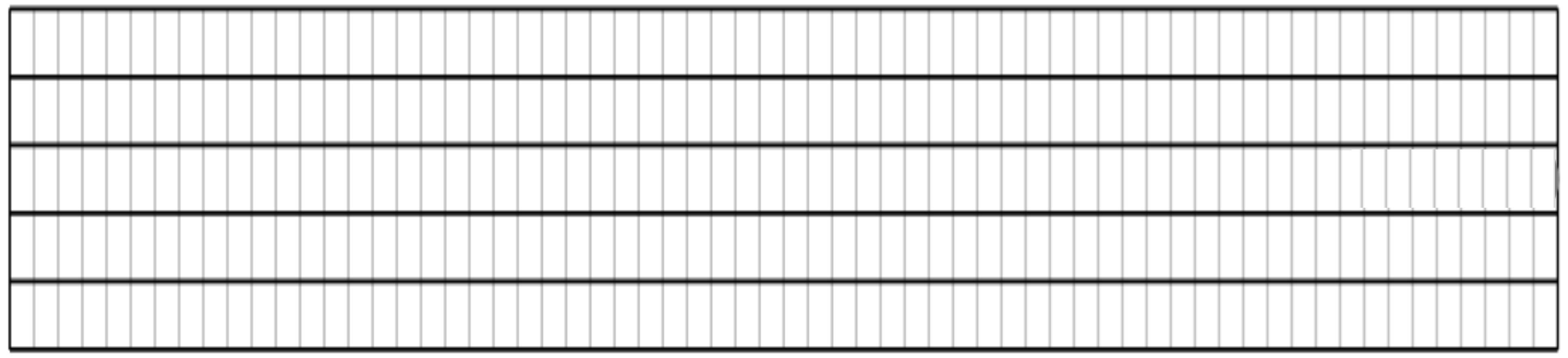
On 320-bit state = 5 x 64-bit words



Input of the first round:

Round computation

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

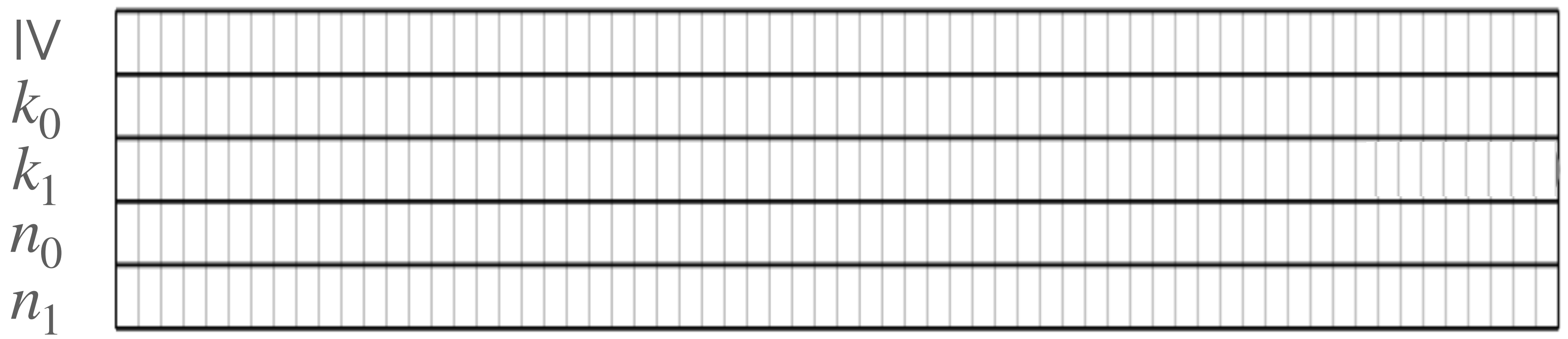


Input of the first round:

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

Round computation

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

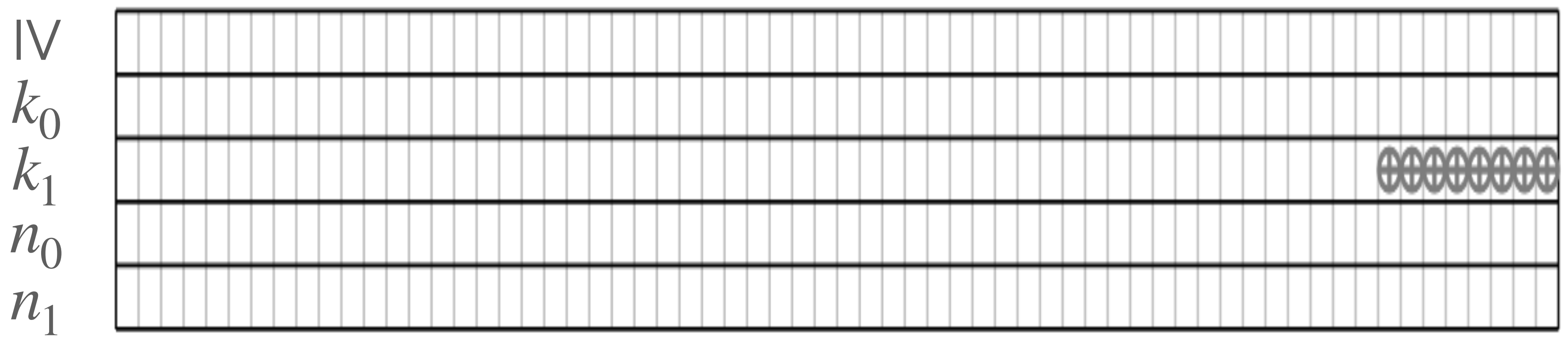


Input of the first round:

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

Round computation

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



(1) Round constant addition

Input of the first round:

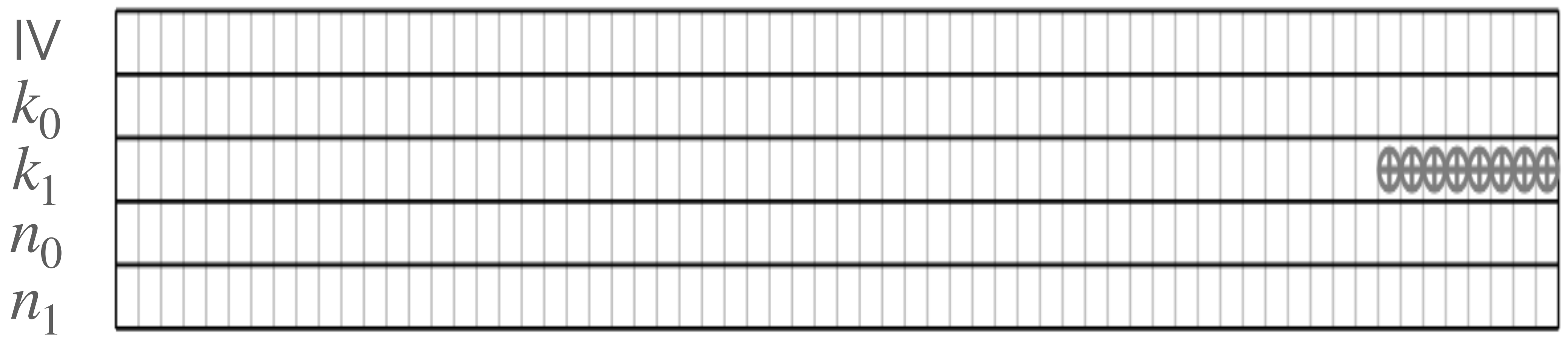
- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV

Round computation

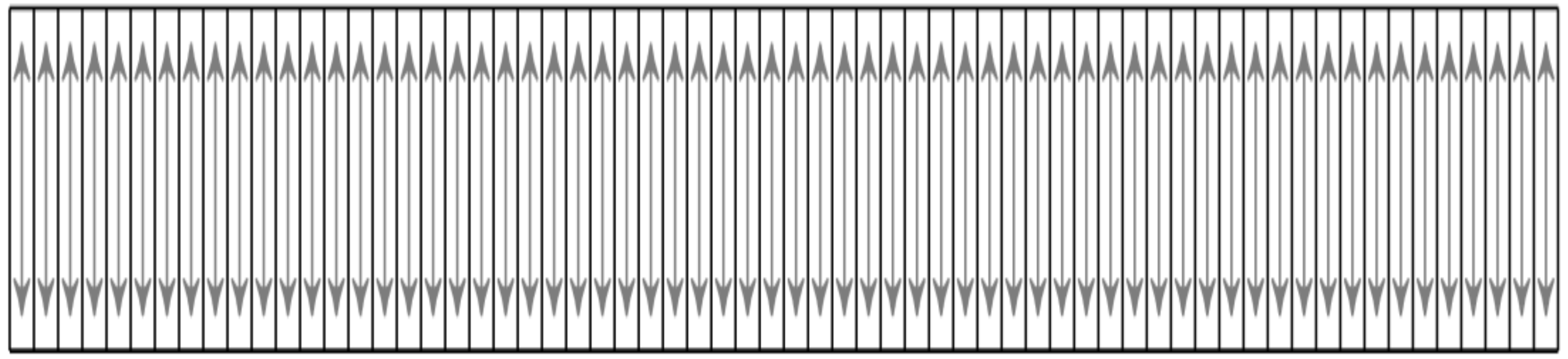
On 320-bit state = 5 x 64-bit words

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



(1) Round constant addition



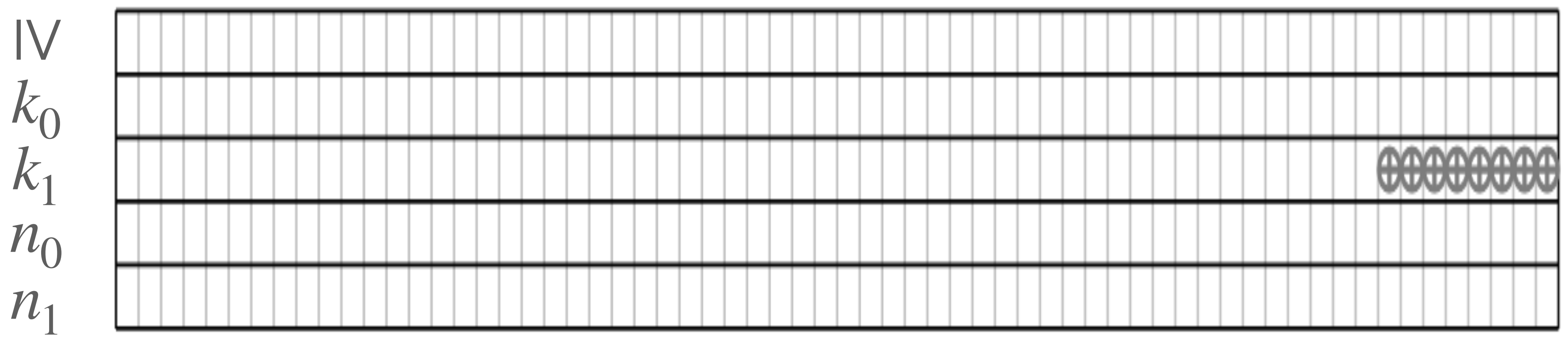
(2) Substitution *(vertical)*

Round computation

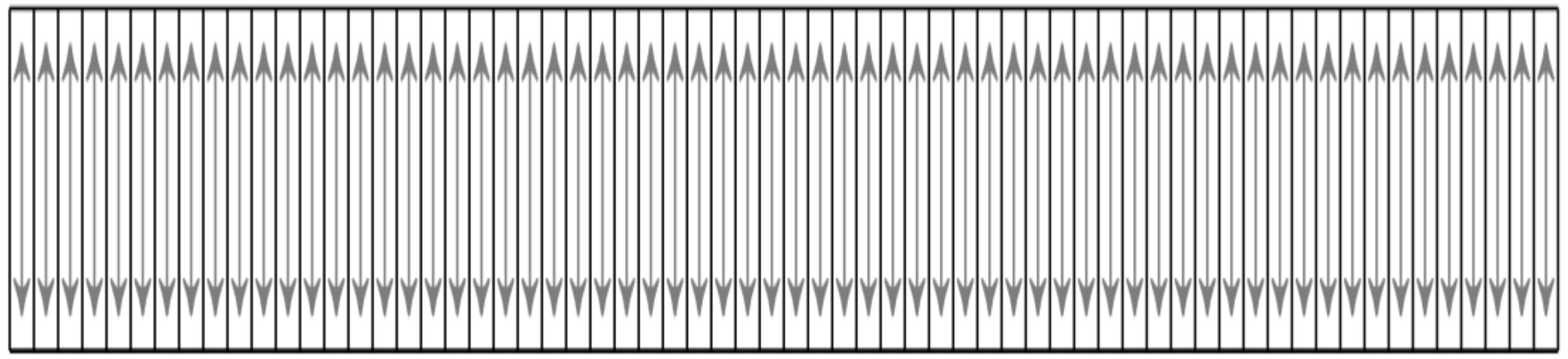
On 320-bit state = 5 x 64-bit words

Input of the first round:

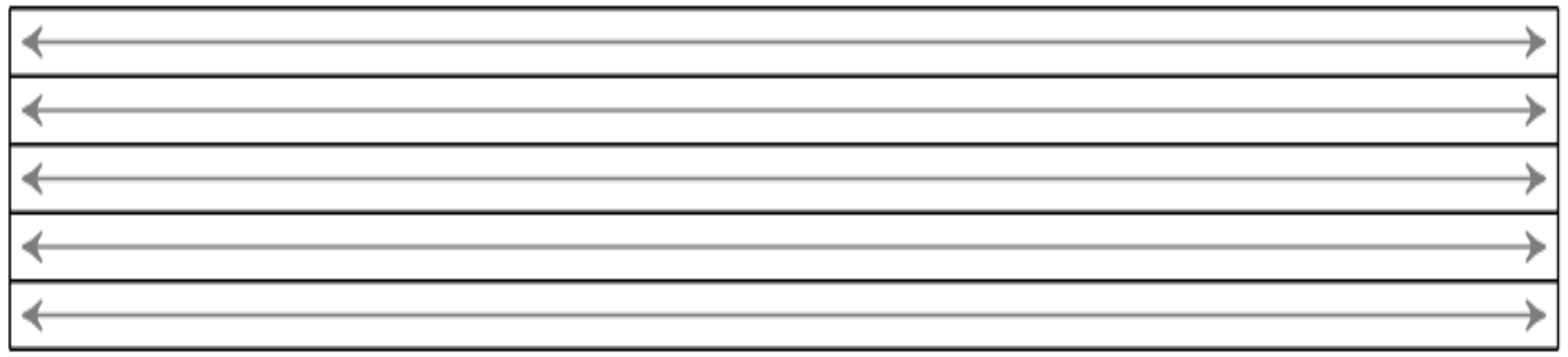
- 128-bit key : $K = (k_0, k_1)$
- 128-bit nonce : $N = (n_0, n_1)$
- 64-bit init. vector : IV



(1) Round constant addition

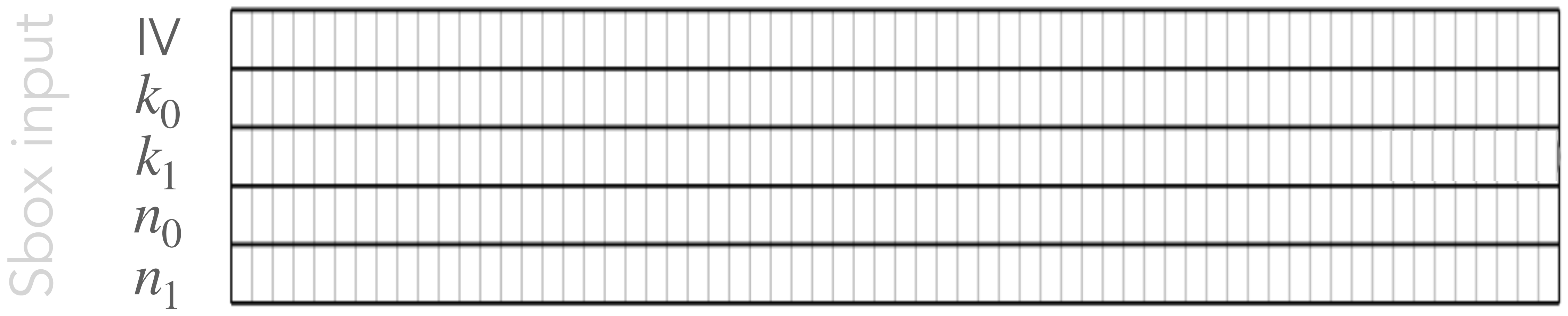


(2) Substitution *(vertical)*

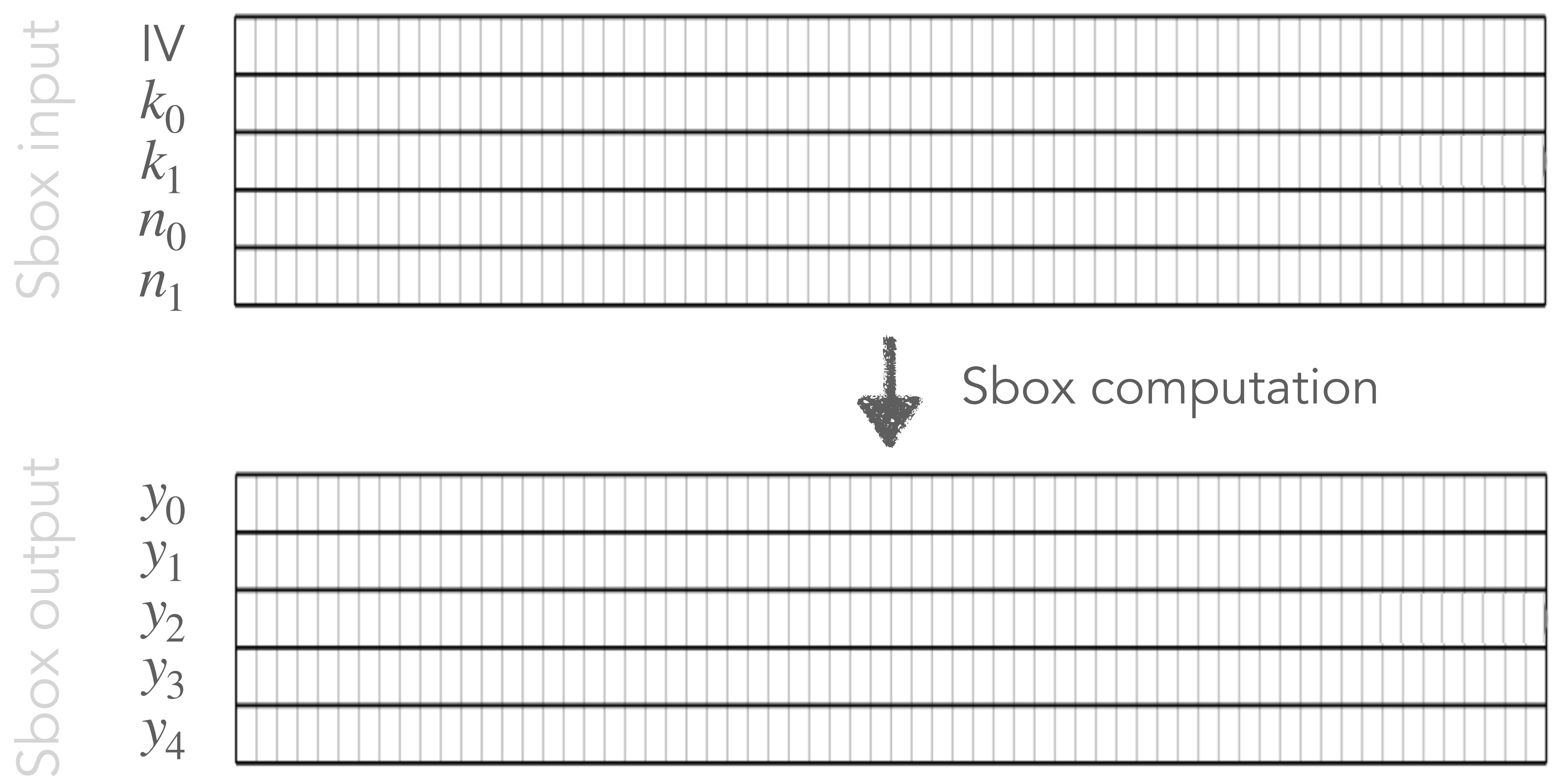


(3) Linear diffusion *(horizontal)*

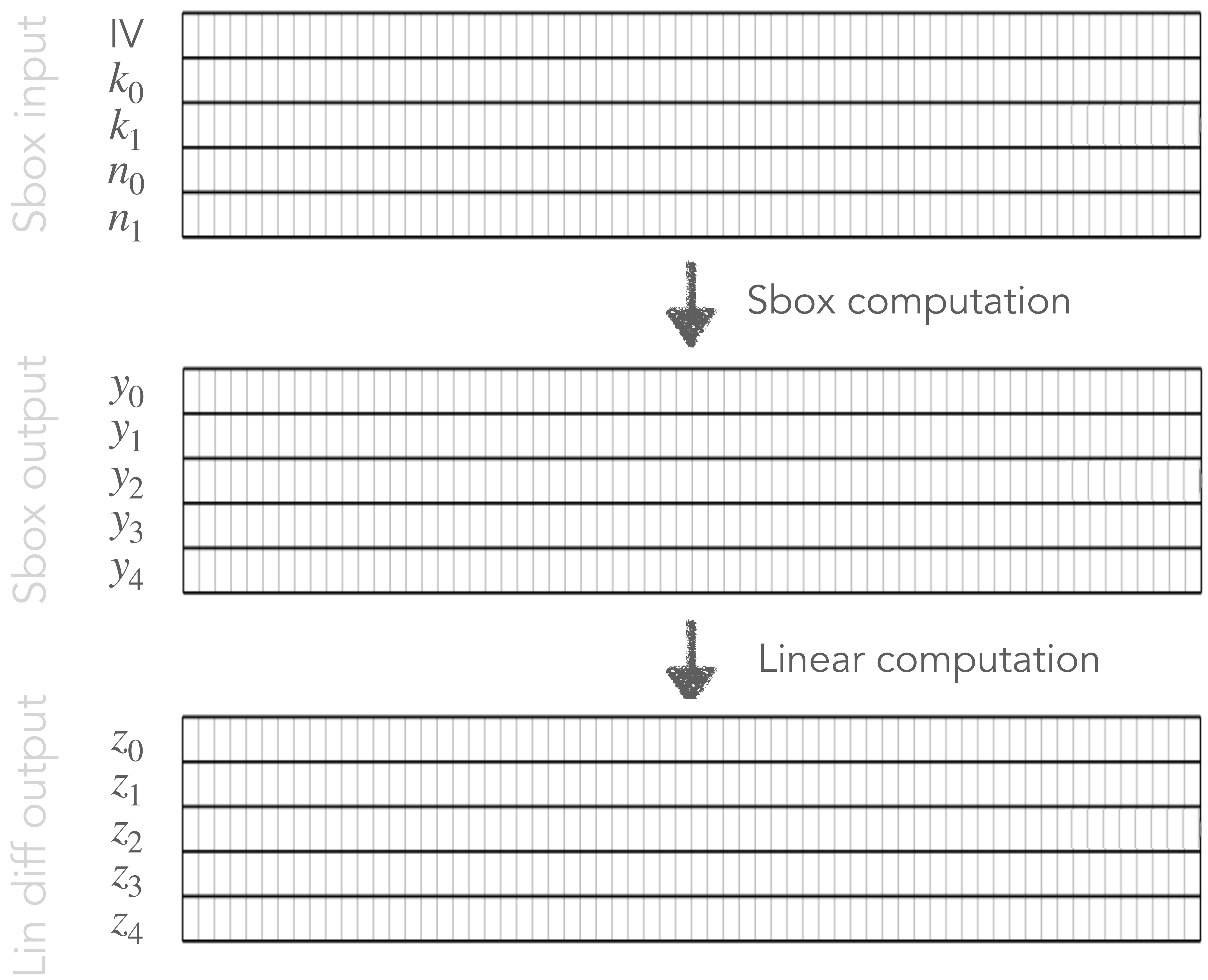
State changes



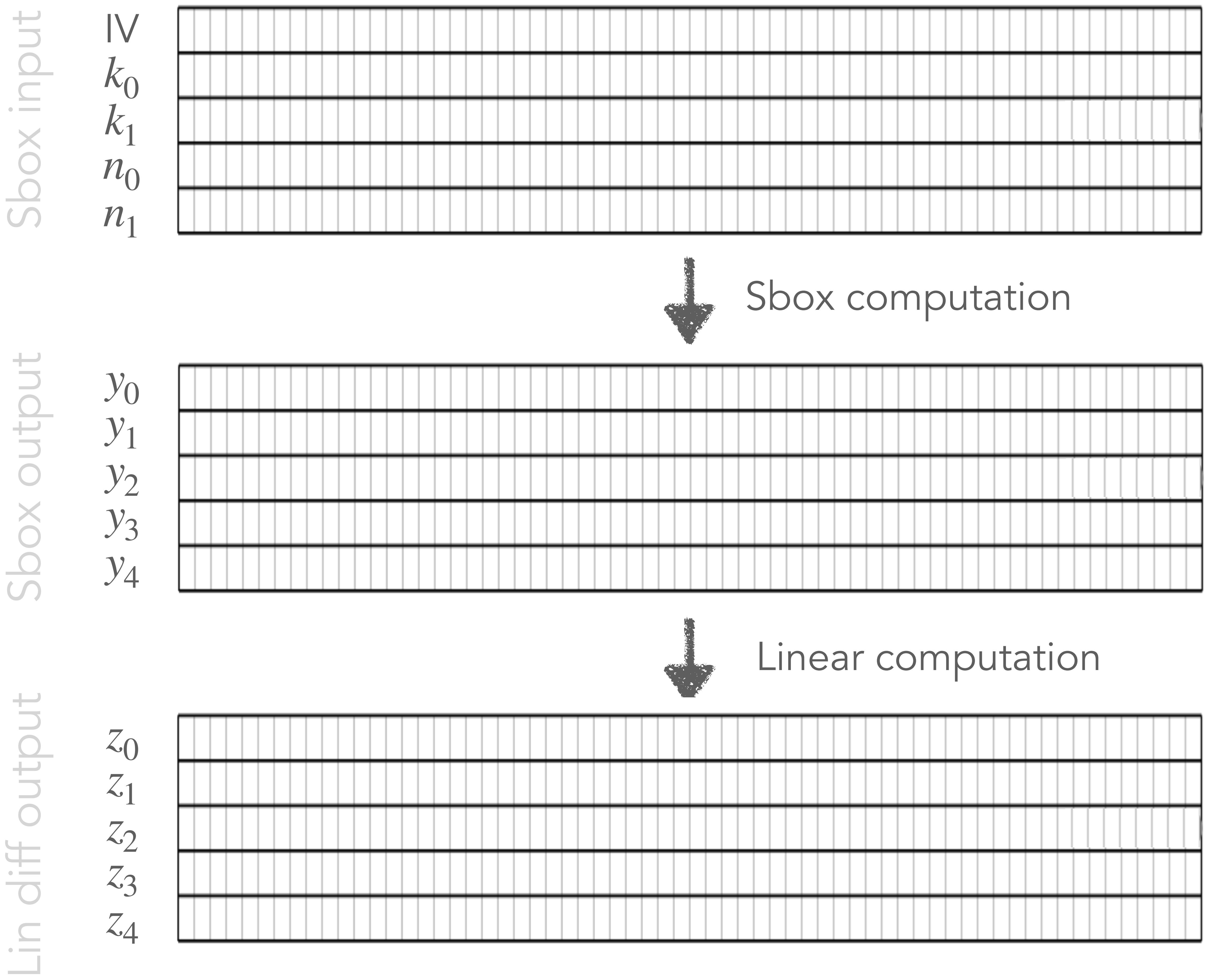
State changes



State changes



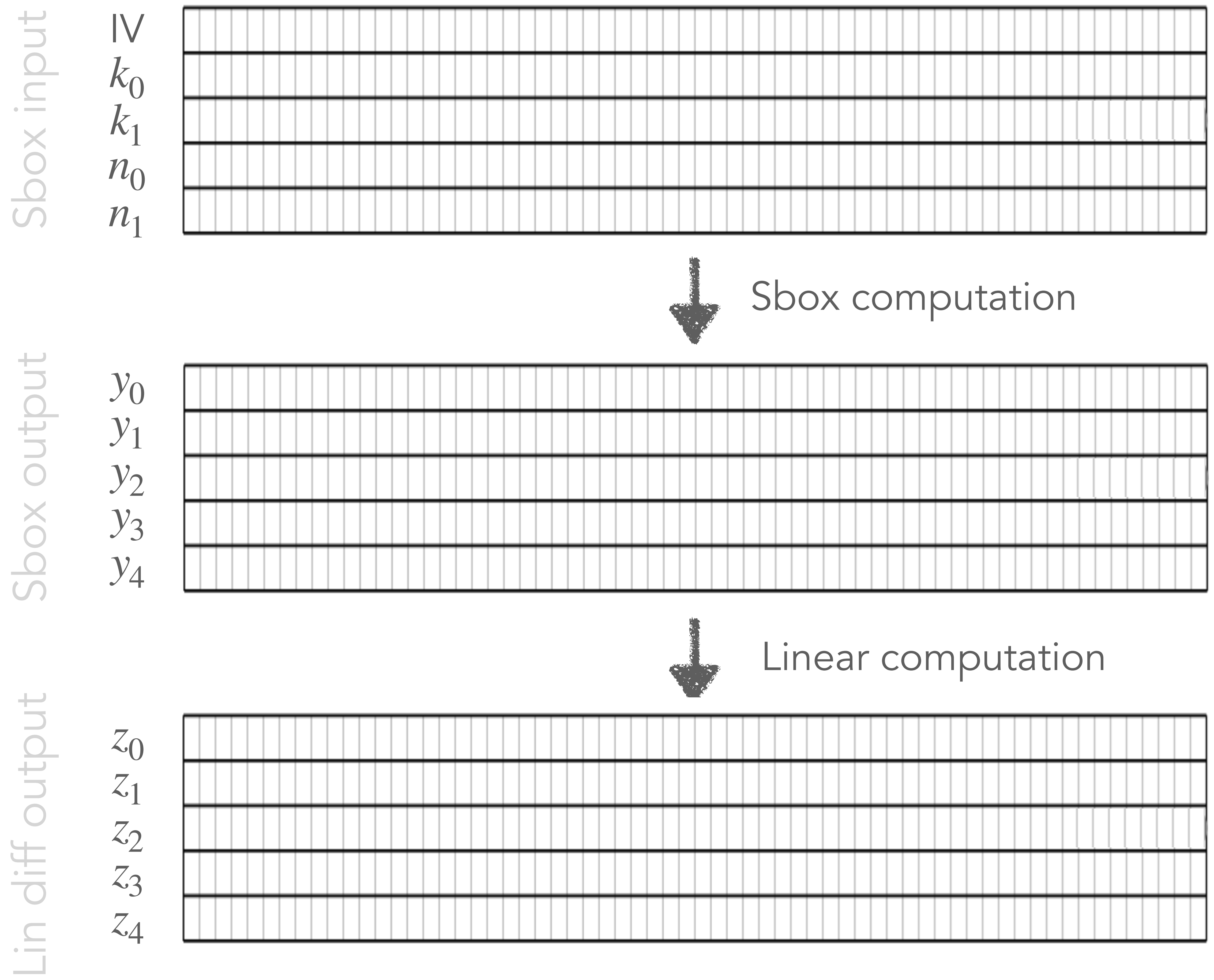
Previous CPA (1)



Previous CPA (1)

Ramezanpour et al., 2020

Choose Sbox output as attack point

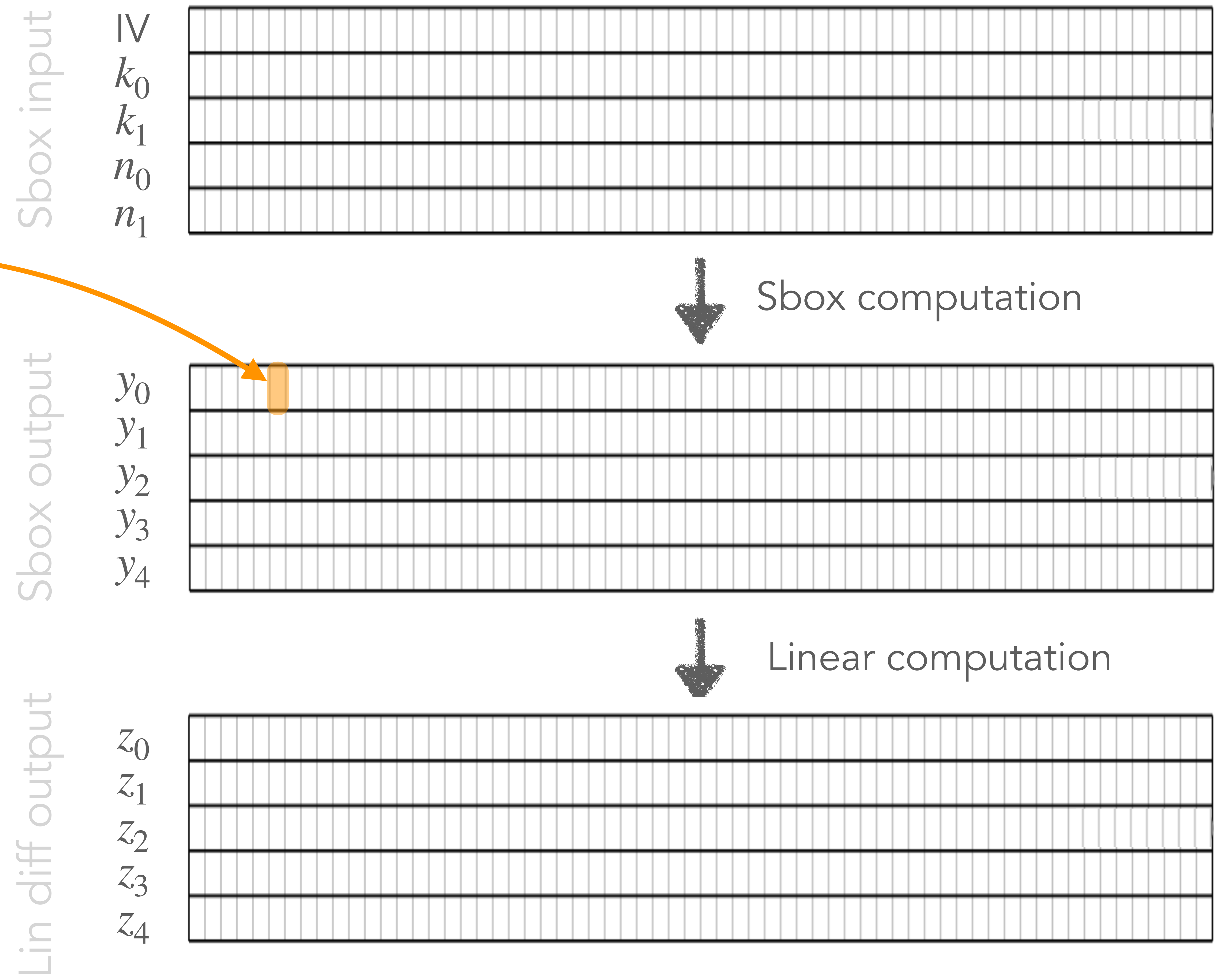


Previous CPA (1)

Ramezanpour et al., 2020

Choose Sbox output as attack point

- Intermediate variable: y_0^j



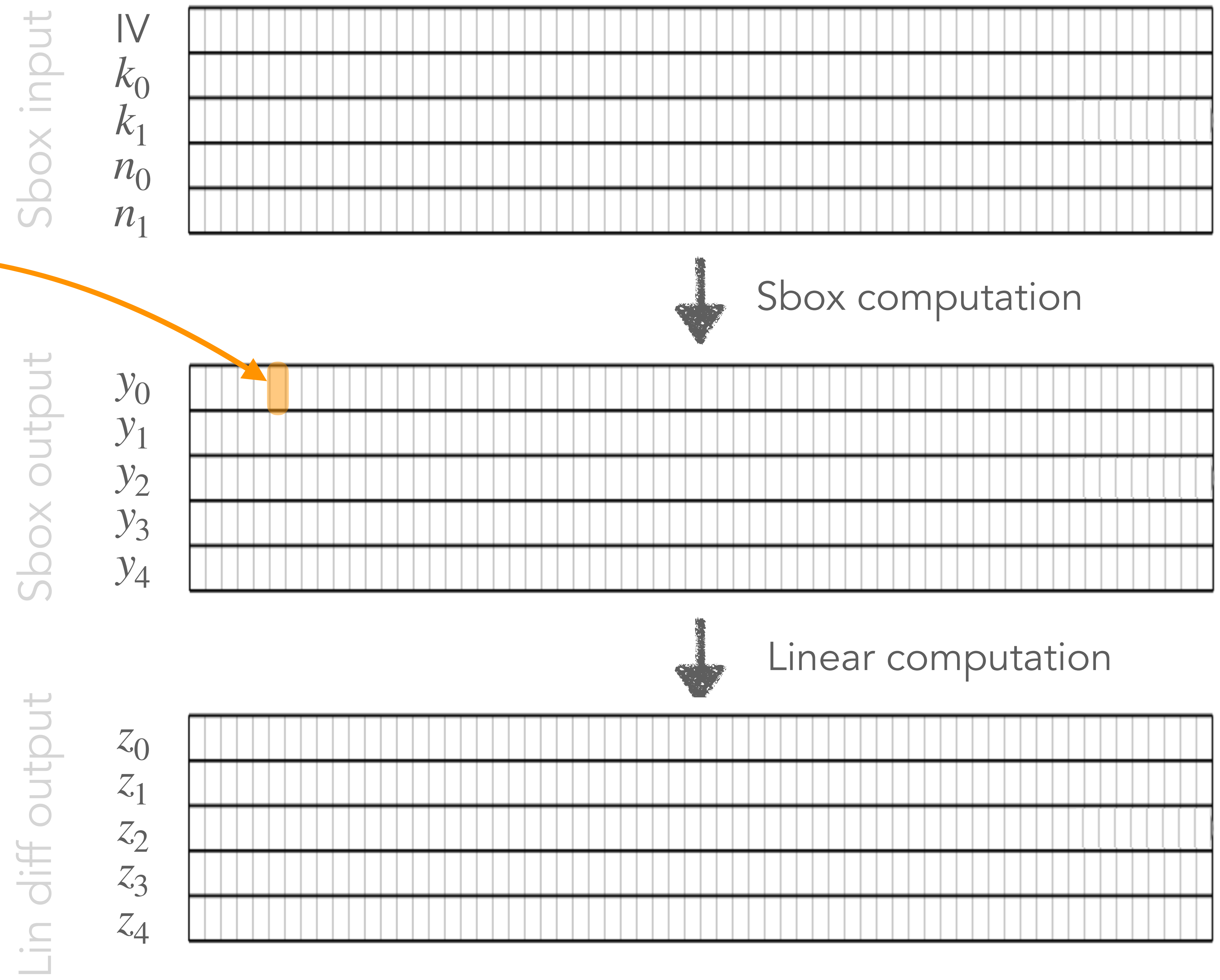
Previous CPA (1)

Ramezanpour et al., 2020

Choose Sbox output as attack point

- Intermediate variable: y_0^j

  → Failed (with 40K traces)



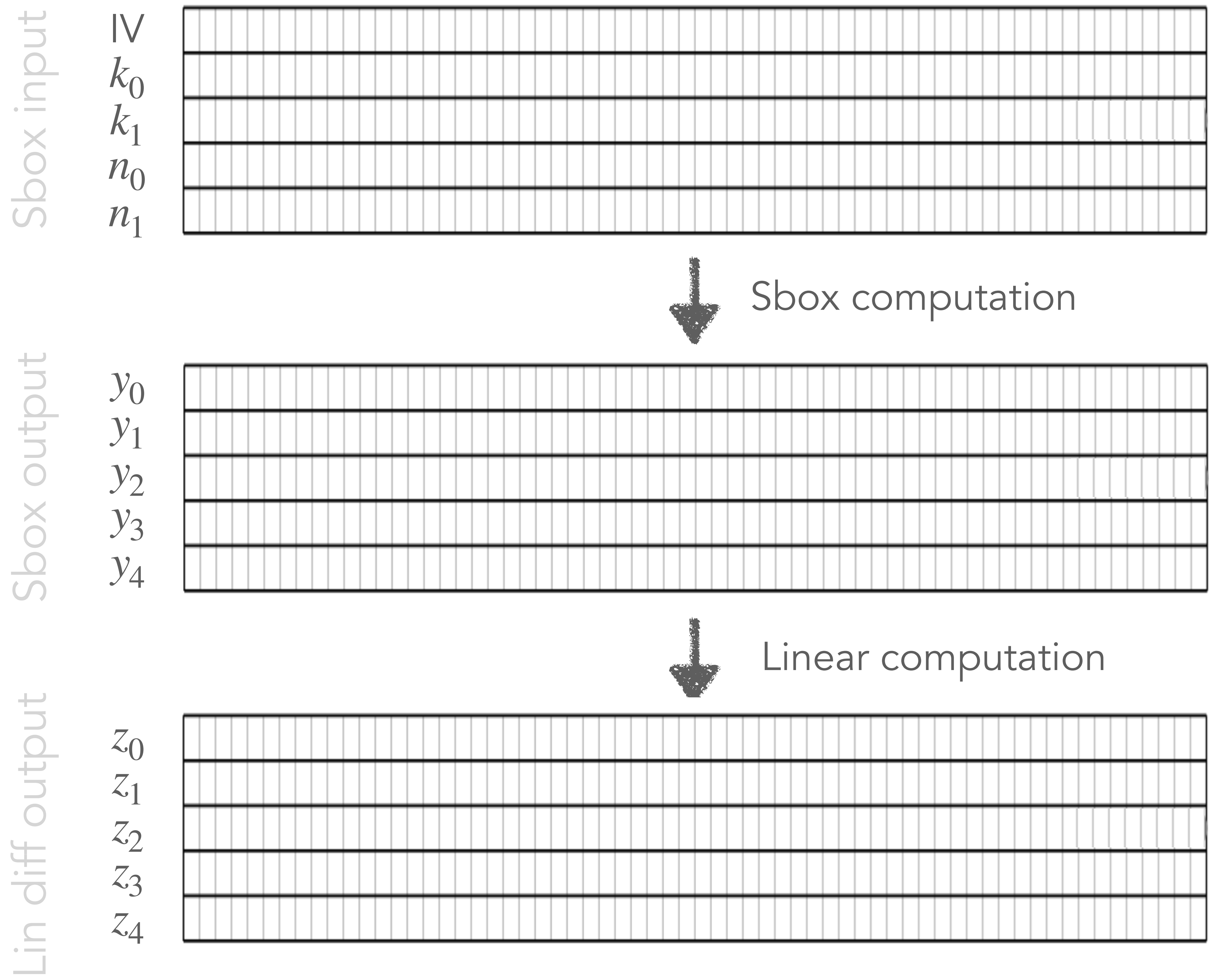
Previous CPA (1)

Ramezanpour et al., 2020

Choose Sbox output as attack point

- Intermediate variable: y_0^j

  → Failed (with 40K traces)



Previous CPA (1)

Ramezanpour et al., 2020

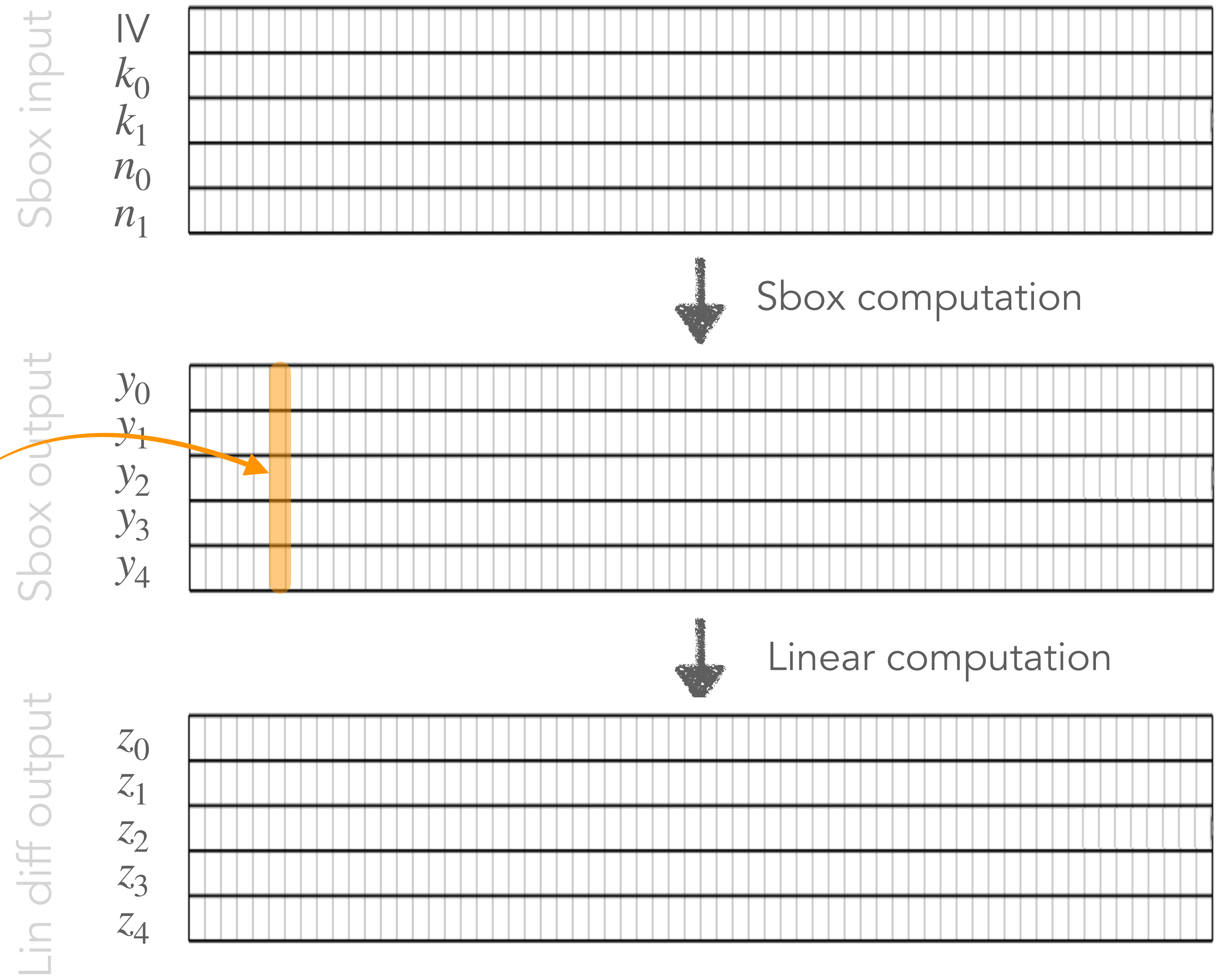
Choose Sbox output as attack point

- Intermediate variable: y_0^j

  → Failed (with 40K traces)

- Intermediate variable: $(y_0^j | y_1^j | y_2^j | y_3^j | y_4^j)$

  HW of Sbox output



Previous CPA (1)

Ramezanpour et al., 2020

Choose Sbox output as attack point

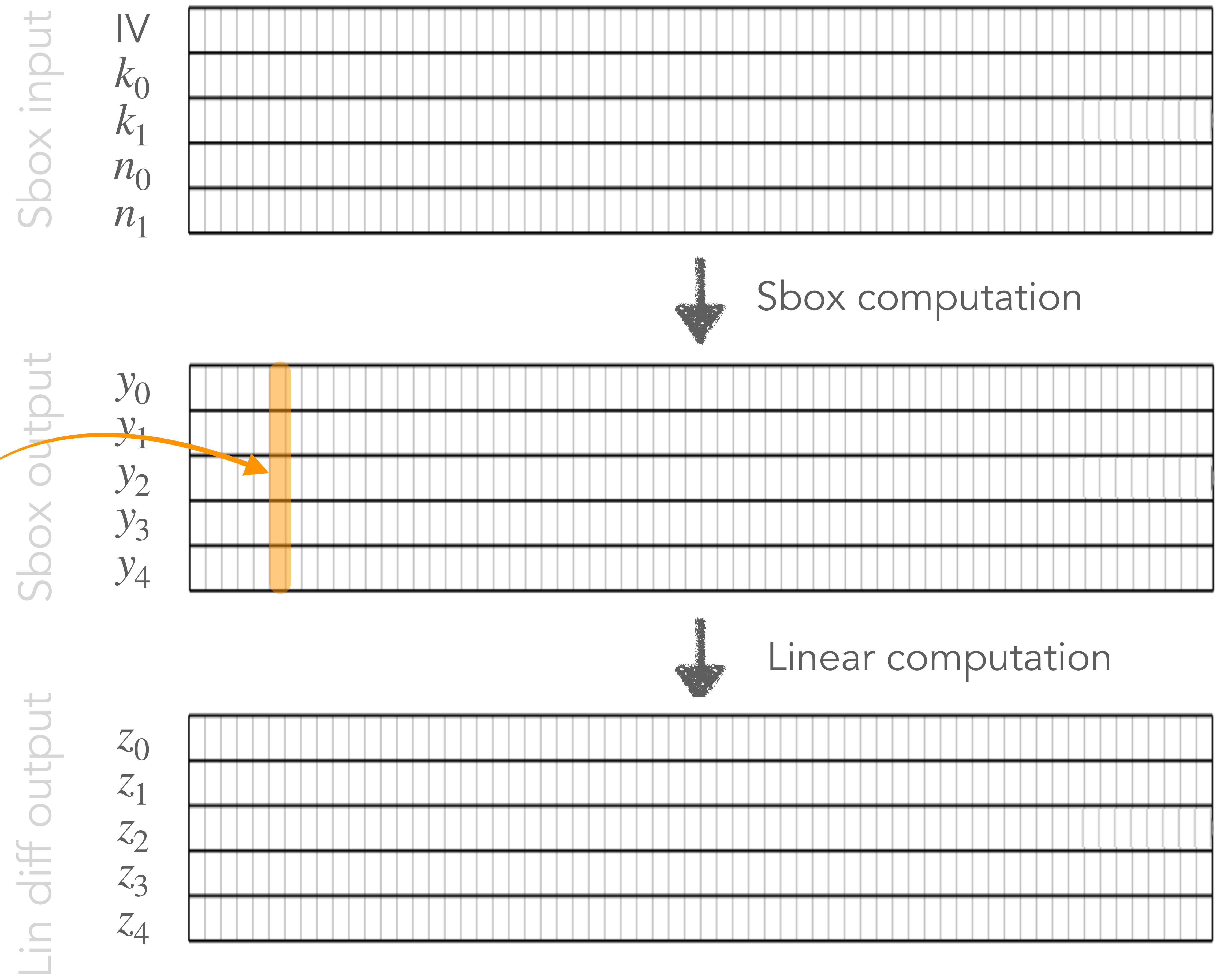
- Intermediate variable: y_0^j

  → Failed (with 40K traces)

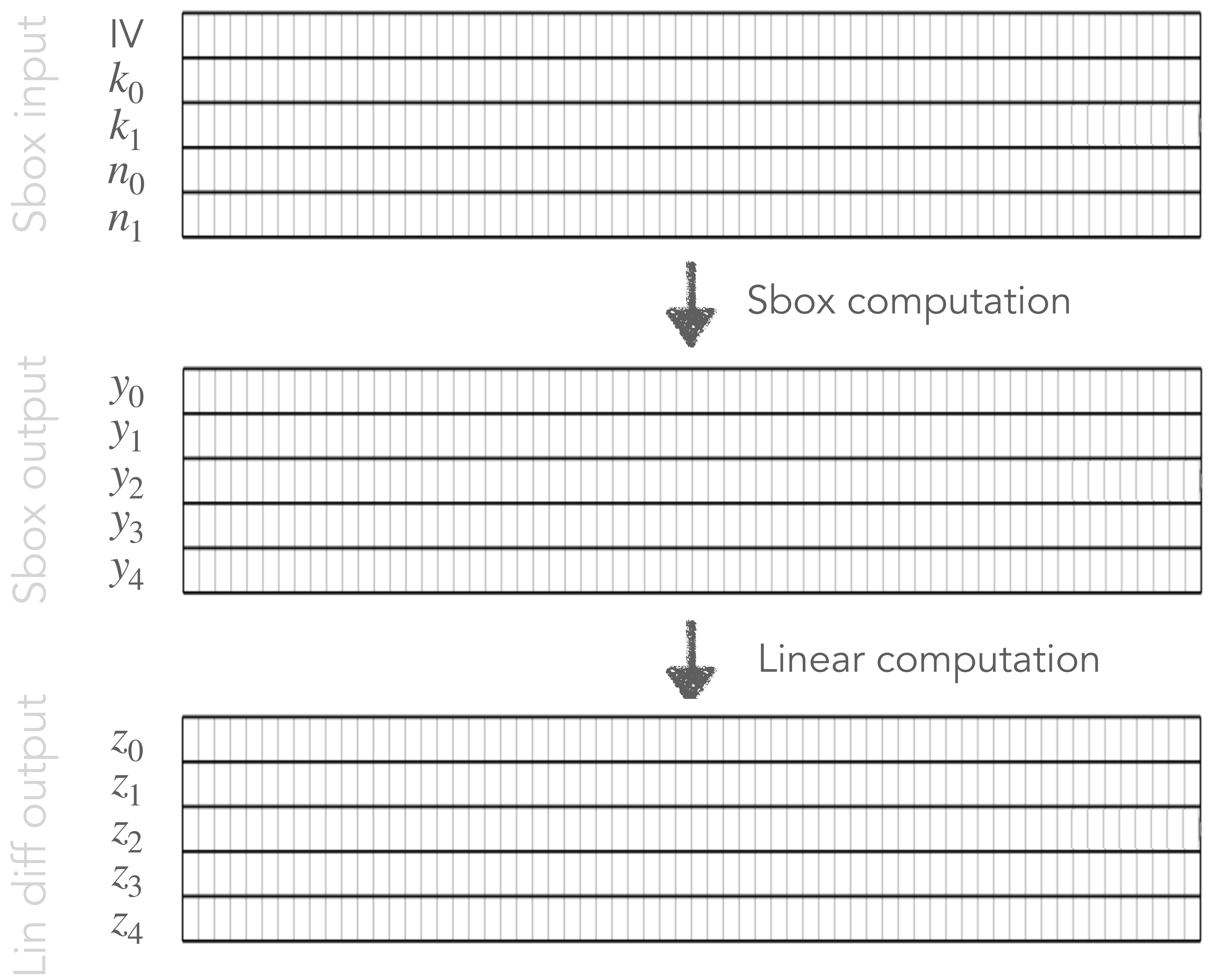
- Intermediate variable: $(y_0^j | y_1^j | y_2^j | y_3^j | y_4^j)$

  HW of Sbox output

  → Failed (with 40K traces)



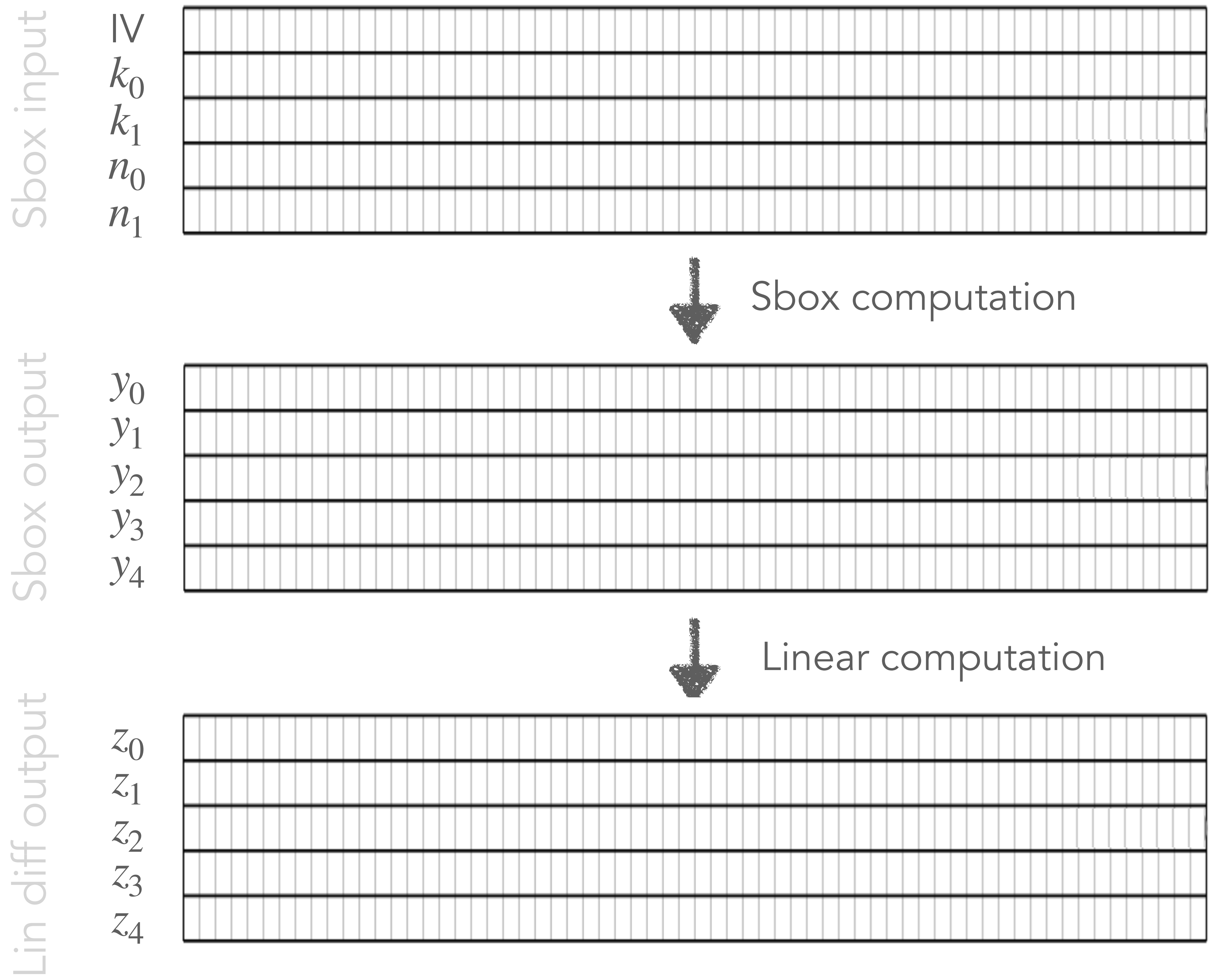
Previous CPA (2)



Previous CPA (2)

Samwel and Daemen, 2018

Choose linear diffusion output as attack point

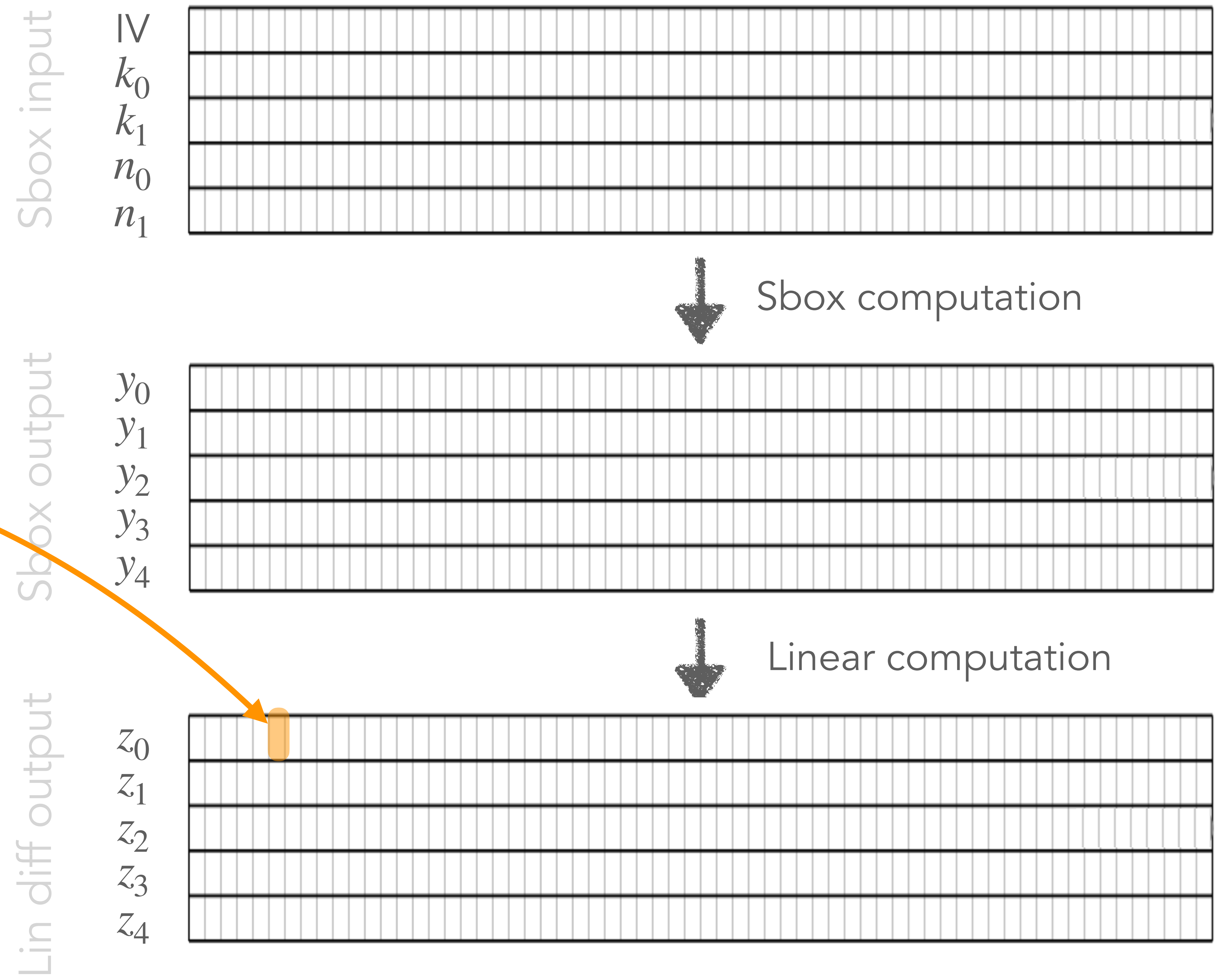


Previous CPA (2)

Samwel and Daemen, 2018

Choose linear diffusion output as attack point

Intermediate variable: z_0^j



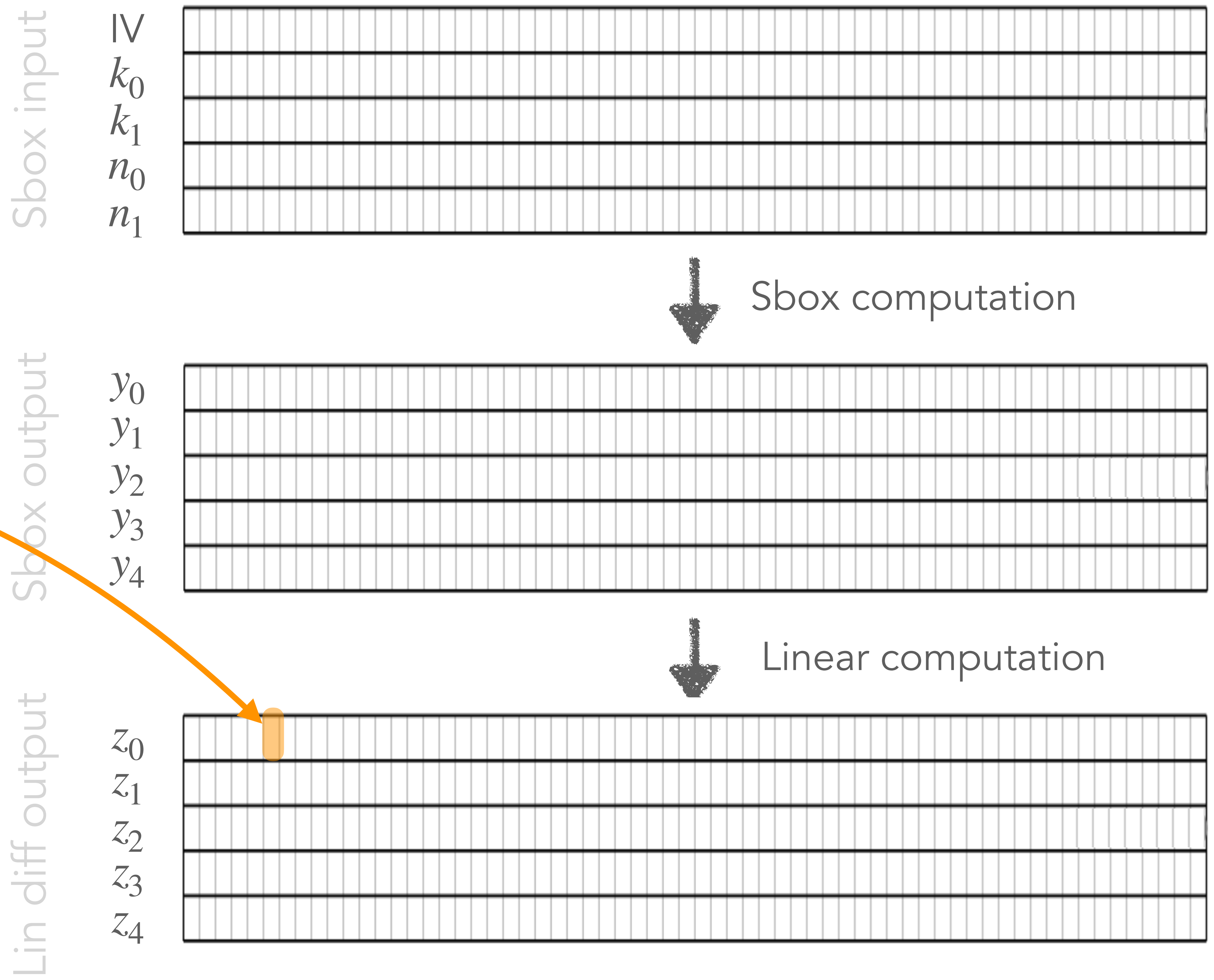
Previous CPA (2)

Samwel and Daemen, 2018

Choose linear diffusion output as attack point

Intermediate variable: z_0^j

Fine-tune computation: $z_0^j \rightarrow \tilde{z}_0^j$



Previous CPA (2)

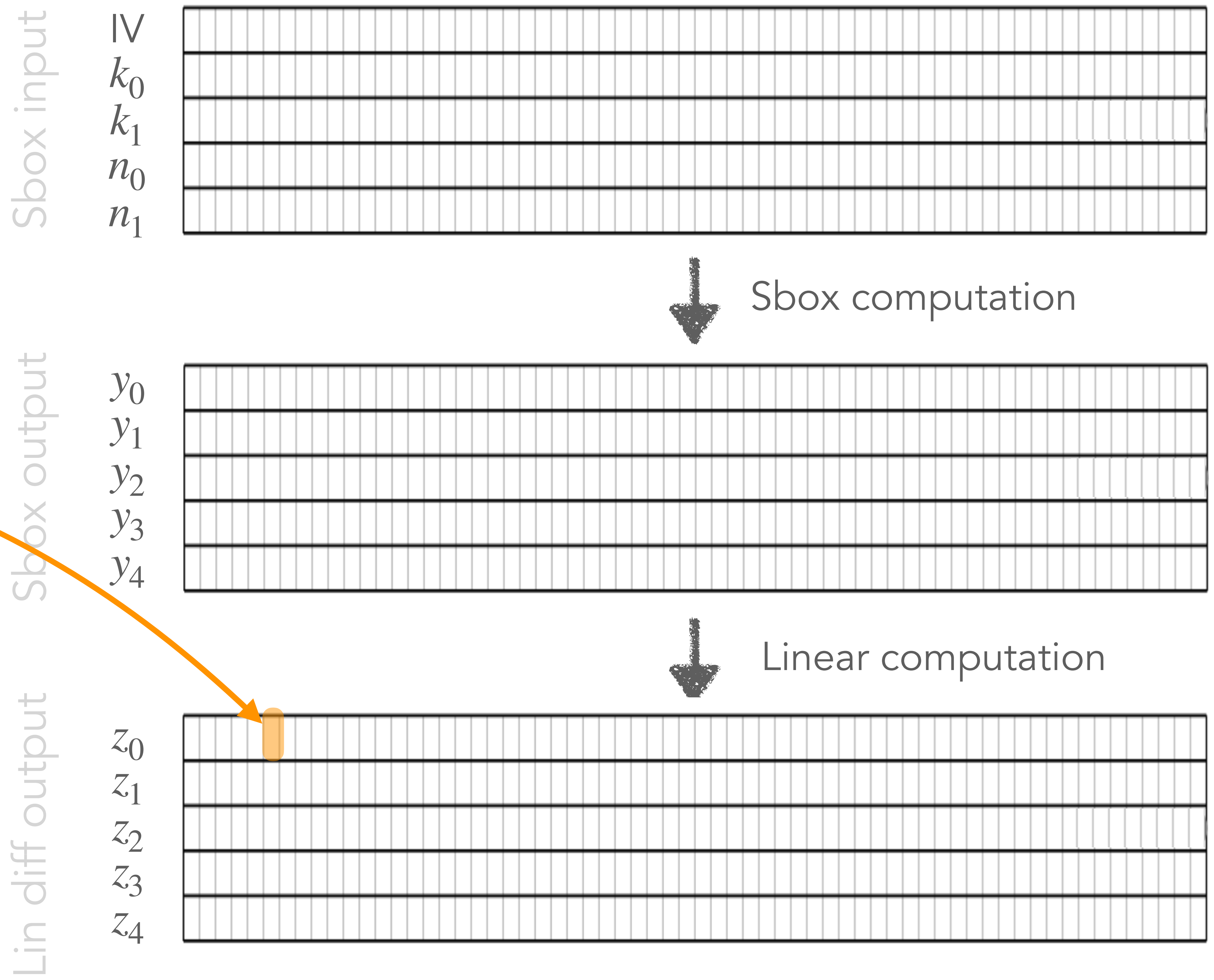
Samwel and Daemen, 2018

Choose linear diffusion output as attack point

Intermediate variable: z_0^j

Fine-tune computation: $z_0^j \rightarrow \tilde{z}_0^j$

→ Succeeded

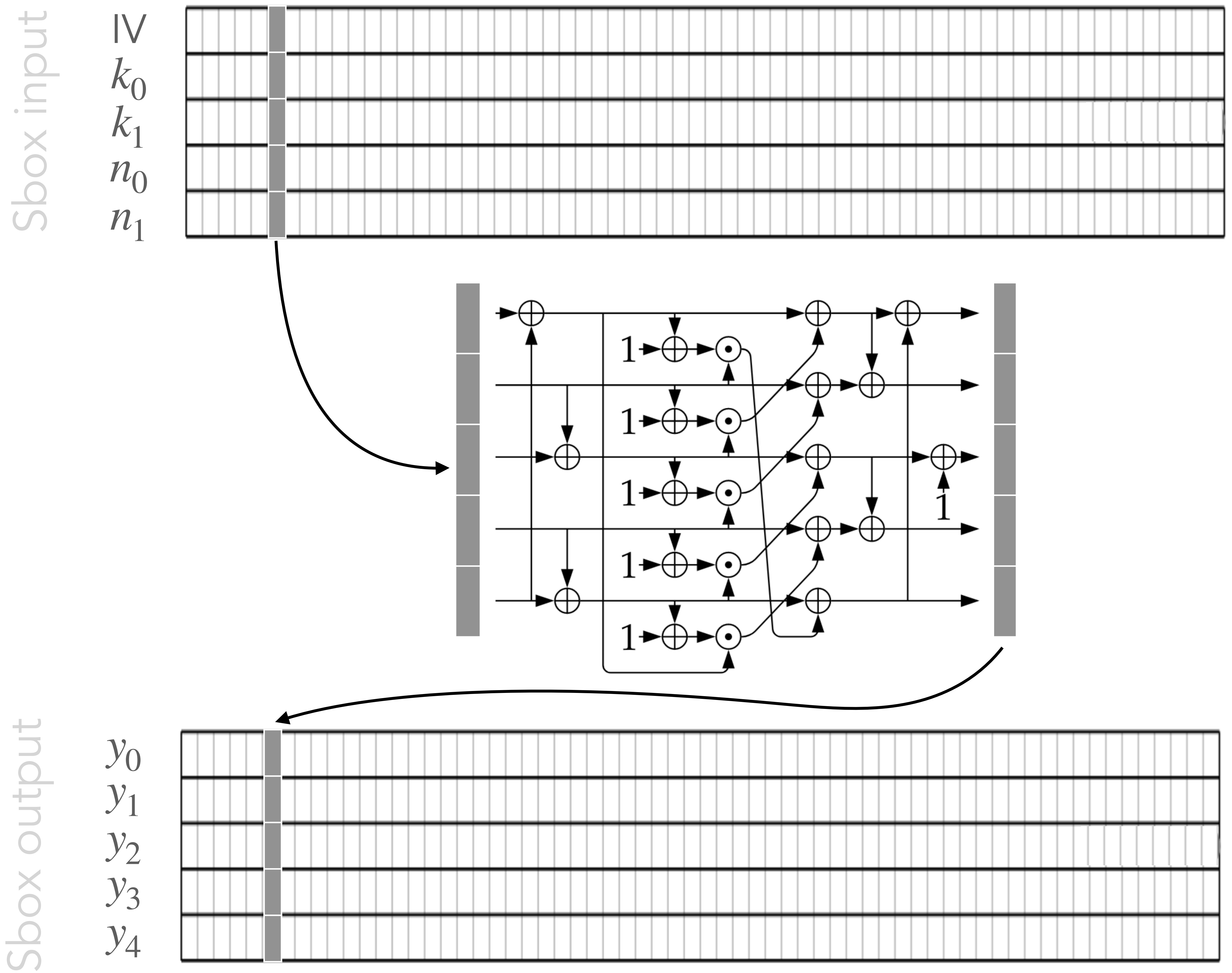


🤔

What are the reasons of this difference ?

(1) Sbox output as attack point

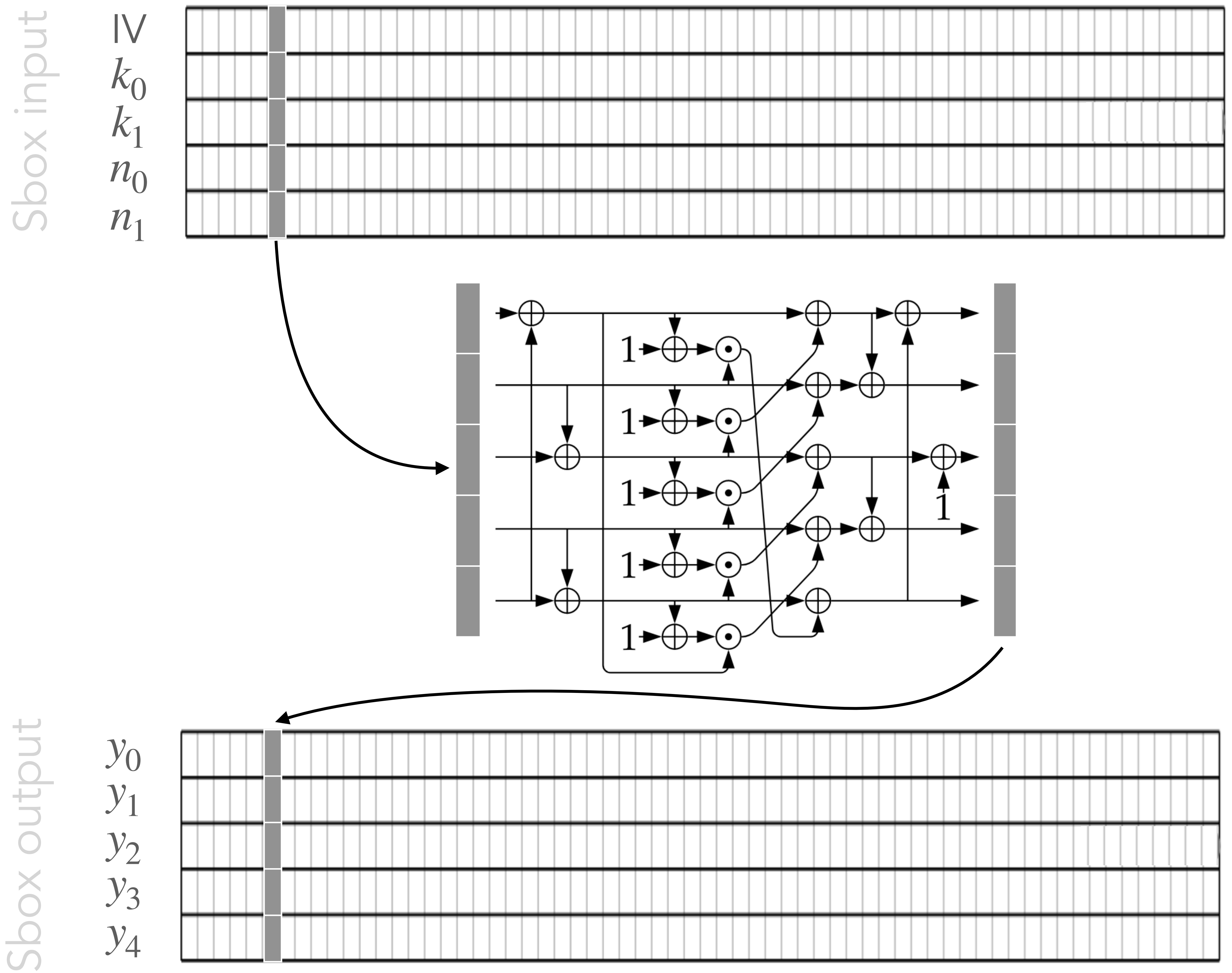
(1) Sbox output as attack point



(1) Sbox output as attack point

In an Sbox computation y_0^j:

- 2 bits of key : (k_0^j, k_1^j)
- 2 bits of nonce : (n_0^j, n_1^j)

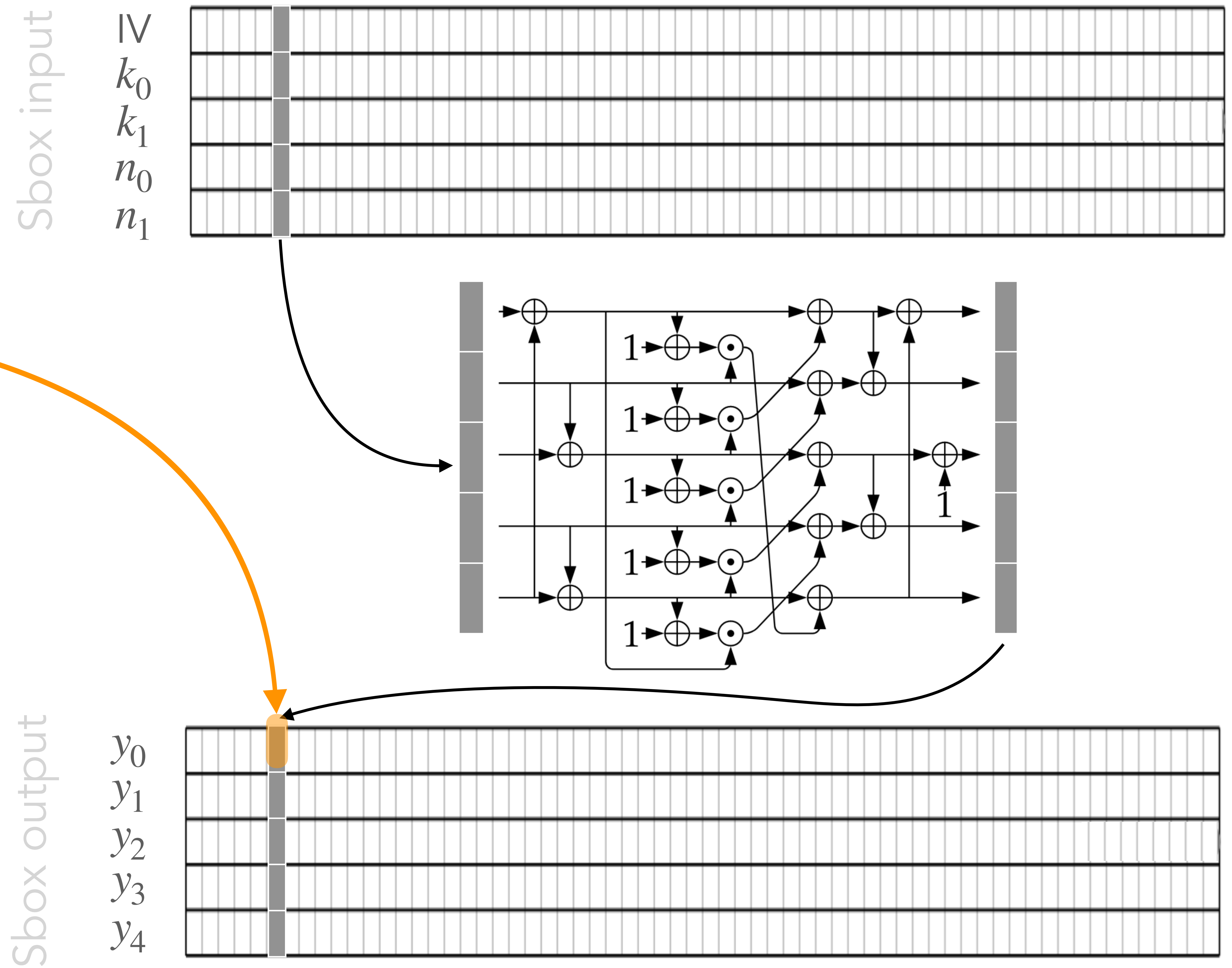


(1) Sbox output as attack point

In an Sbox computation y_0^j:

- 2 bits of key : (k_0^j, k_1^j)
- 2 bits of nonce : (n_0^j, n_1^j)

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j$$



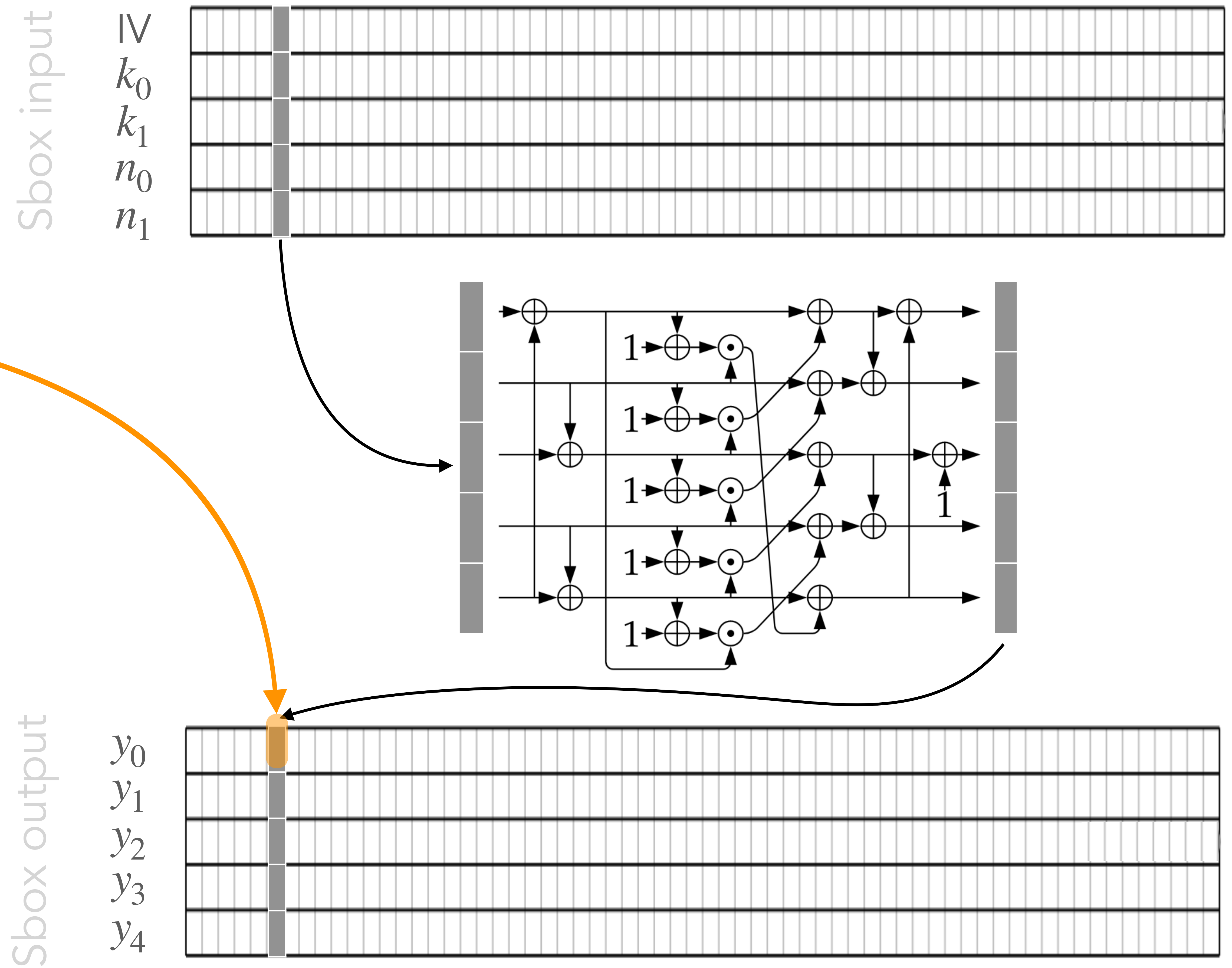
(1) Sbox output as attack point

In an Sbox computation y_0^j:

- 2 bits of key : (k_0^j, k_1^j)
- 2 bits of nonce : (n_0^j, n_1^j)

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j$$

	(k_0^j, k_1^j)			
(n_0^j, n_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	0	1	1	1
(0,1)	0	1	0	0
(1,0)	1	0	0	0
(1,1)	1	0	1	1
Correlation	-1		1	



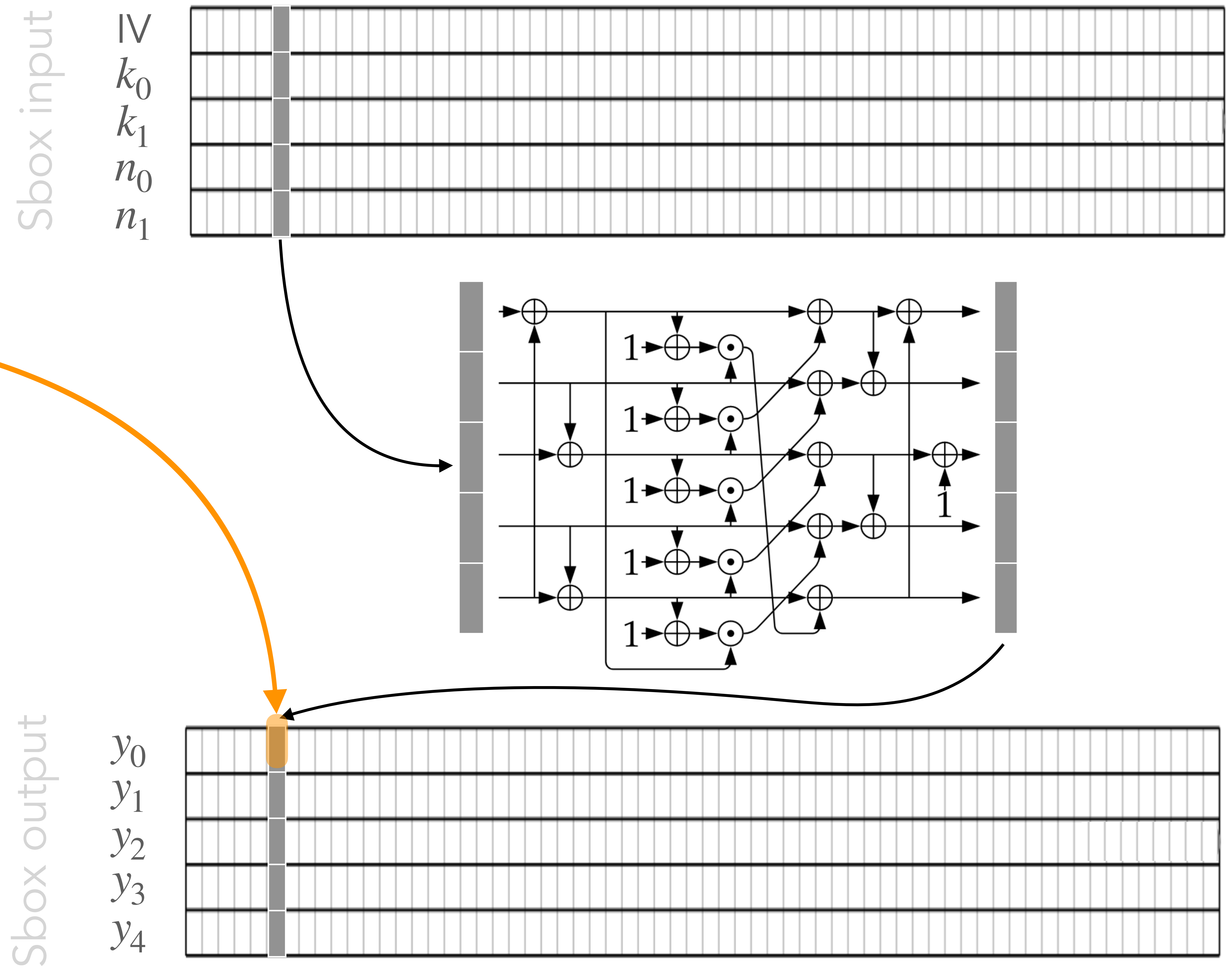
(1) Sbox output as attack point

In an Sbox computation y_0^j:

- 2 bits of key : (k_0^j, k_1^j)
- 2 bits of nonce : (n_0^j, n_1^j)

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j$$

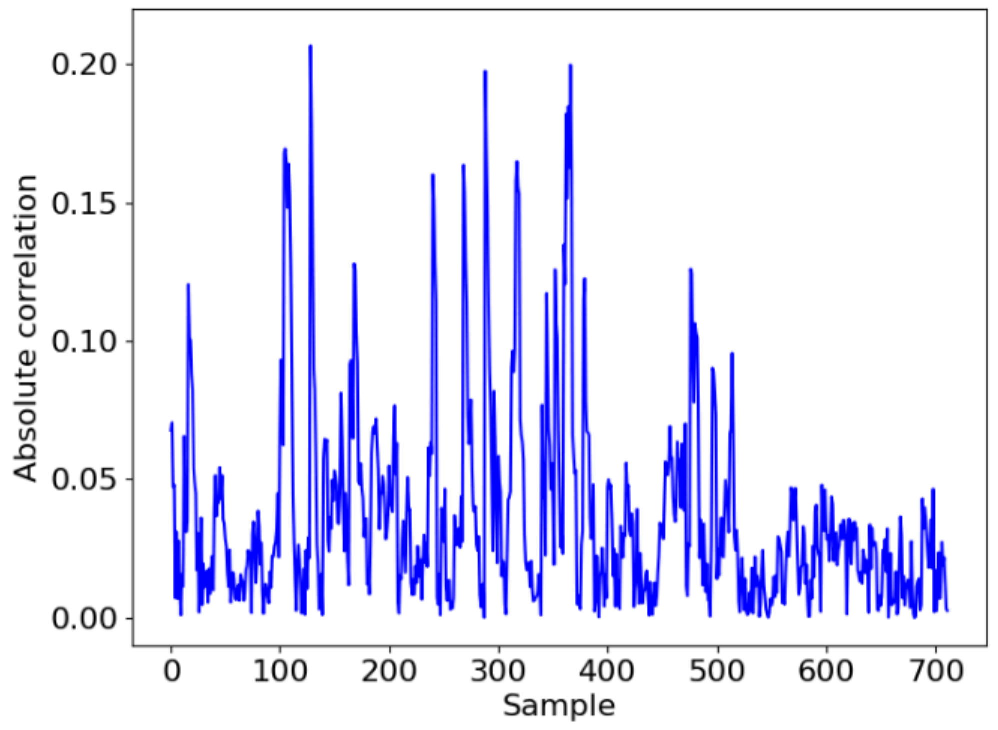
	(k_0^j, k_1^j)			
(n_0^j, n_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	0	1	1	1
(0,1)	0	1	0	0
(1,0)	1	0	0	0
(1,1)	1	0	1	1
Correlation	-1		1	



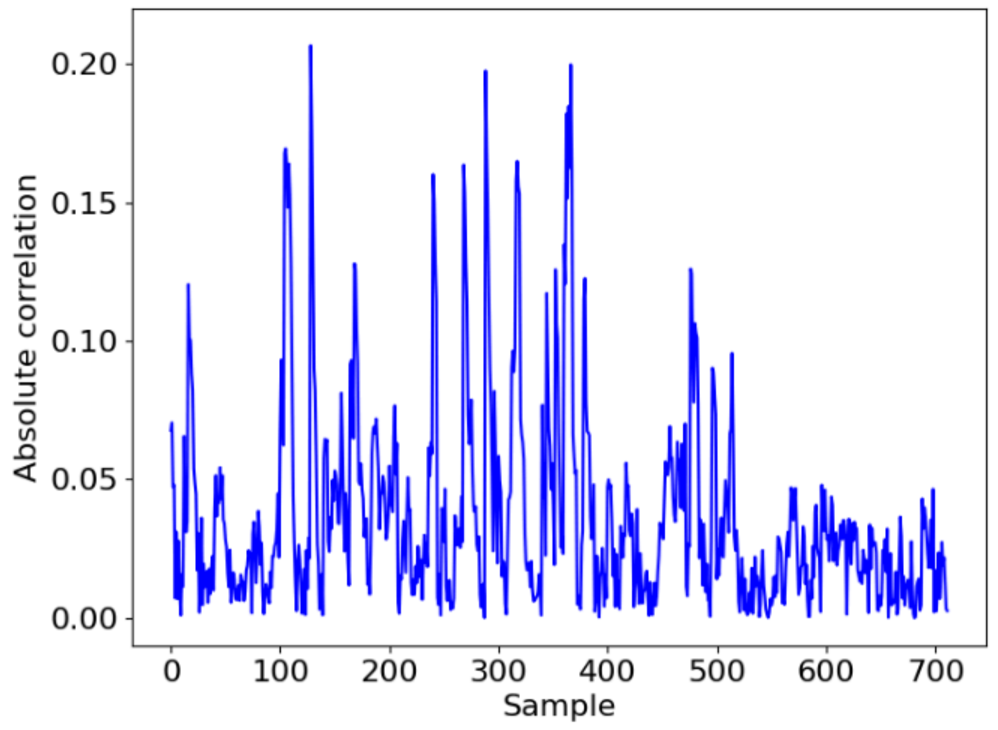
→ We cannot obtain unique (correct) key candidate

(1) Sbox output as attack point

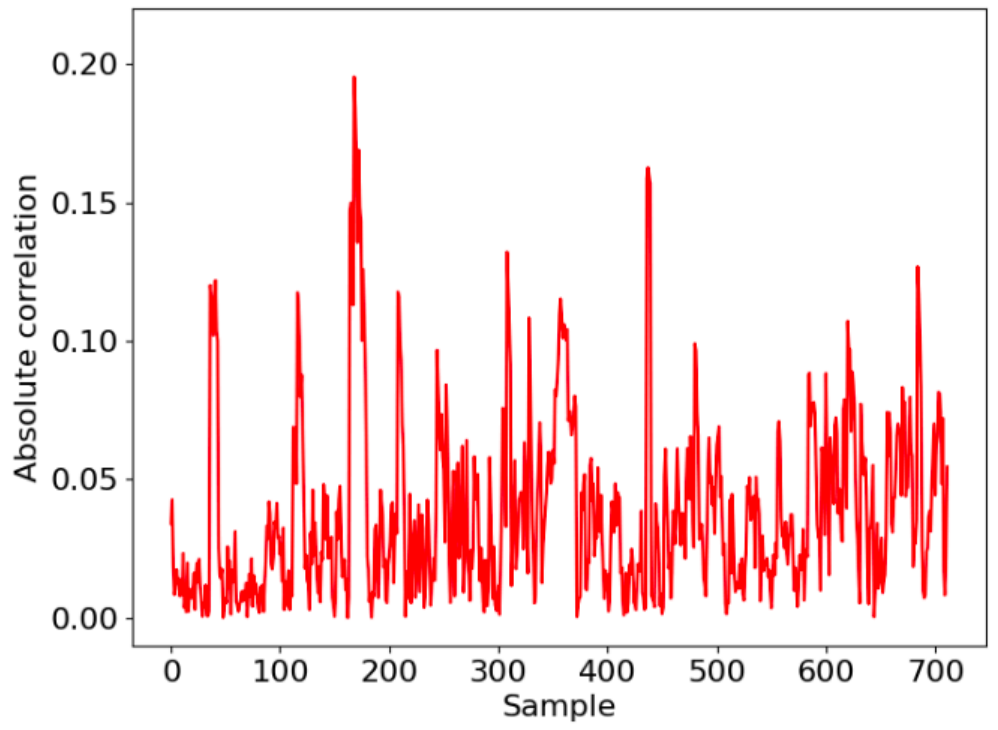
Correlation traces for all key candidates



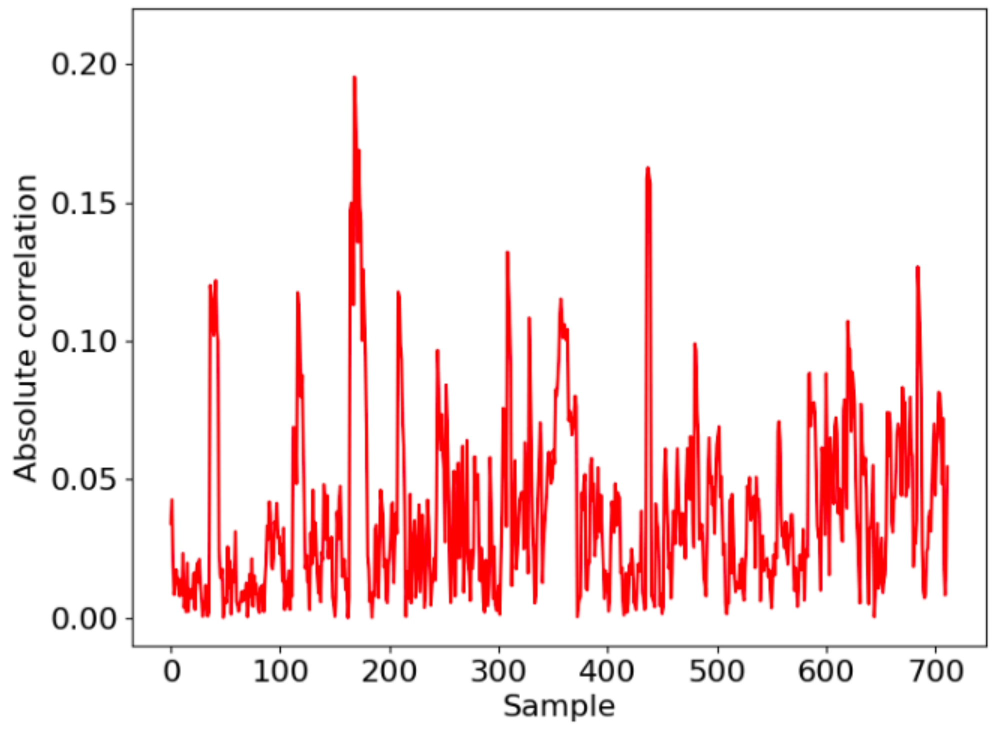
(a) $(k_0^j, k_1^j) = (0, 0)$



(b) $(k_0^j, k_1^j) = (0, 1)$



(c) $(k_0^j, k_1^j) = (1, 0)$



(d) $(k_0^j, k_1^j) = (1, 1)$

(1) Sbox output as attack point

Correlations of distributions associated to all key pairs

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1	1	-	-
(0,1)	1	1	-	-
(1,0)	-	-	1	1
(1,1)	-	-	1	1

y_0^j

(1) Sbox output as attack point

Correlations of distributions associated to all key pairs

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1	1	-	-
(0,1)	1	1	-	-
(1,0)	-	-	1	1
(1,1)	-	-	1	1

y_0^j

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1	-	-	1
(0,1)	-	1	1	-
(1,0)	-	1	1	-
(1,1)	1	-	-	1

y_1^j

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1	1	1	1
(0,1)	1	1	1	1
(1,0)	1	1	1	1
(1,1)	1	1	1	1

y_2^j and y_3^j

k_0^j	0	1
0	1	0
1	0	1

y_4^j

(1) Sbox output as attack point

Hamming weight of Sbox output: HW($y_0^j \,|\, y_1^j \,|\, y_2^j \,|\, y_3^j \,|\, y_4^j$)

(1) Sbox output as attack point

Hamming weight of Sbox output: HW($y_0^j \mid y_1^j \mid y_2^j \mid y_3^j \mid y_4^j$)

Correlations of distributions associated to all key pairs

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1.00	0.15	0.89	0.87
(0,1)	0.15	1.00	0.48	0.09
(1,0)	0.89	0.48	1.00	0.90
(1,1)	0.87	0.09	0.90	1.00

(1) Sbox output as attack point

Hamming weight of Sbox output: HW($y_0^j \,|\, y_1^j \,|\, y_2^j \,|\, y_3^j \,|\, y_4^j$)

Correlations of distributions associated to all key pairs

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1.00	0.15	0.89	0.87
(0,1)	0.15	1.00	0.48	0.09
(1,0)	0.89	0.48	1.00	0.90
(1,1)	0.87	0.09	0.90	1.00

(1) Sbox output as attack point

Hamming weight of Sbox output: HW($y_0^j \,|\, y_1^j \,|\, y_2^j \,|\, y_3^j \,|\, y_4^j$)

Correlations of distributions associated to all key pairs

(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1.00	0.15	0.89	0.87
(0,1)	0.15	1.00	0.48	0.09
(1,0)	0.89	0.48	1.00	0.90
(1,1)	0.87	0.09	0.90	1.00

→ Not effective for CPA attacks

(1) Sbox output as attack point

Hamming weight of Sbox output: HW($y_0^j \mid y_1^j \mid y_2^j \mid y_3^j \mid y_4^j$)

Correlations of distributions associated to all key pairs

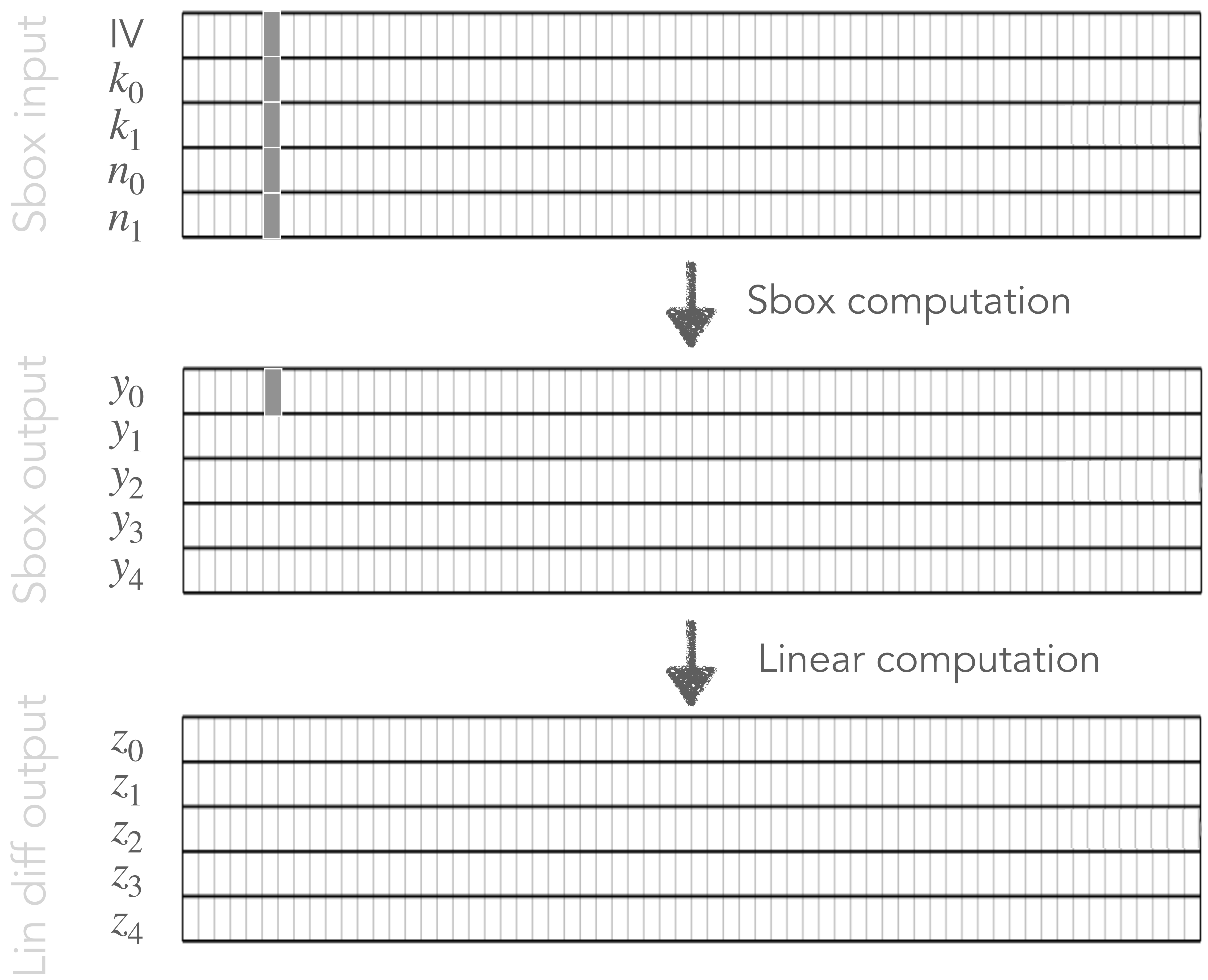
(k_0^j, k_1^j)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	1.00	0.15	0.89	0.87
(0,1)	0.15	1.00	0.48	0.09
(1,0)	0.89	0.48	1.00	0.90
(1,1)	0.87	0.09	0.90	1.00

→ Not effective for CPA attacks

This may explain why Ramezanpour et al., 2020 failed

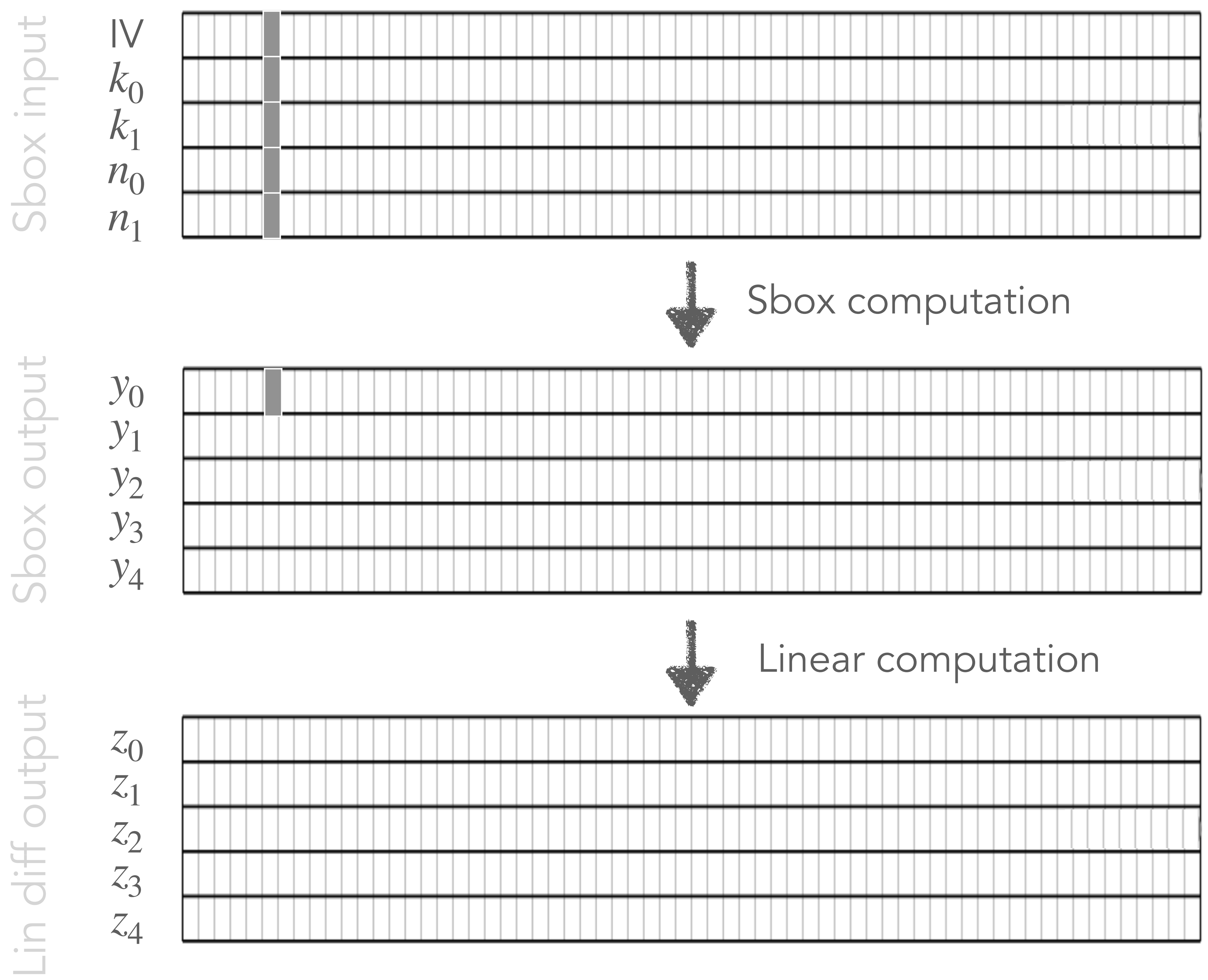
(2) Fine-tuned linear diffusion output as attack point

(2) Fine-tuned linear diffusion output as attack point



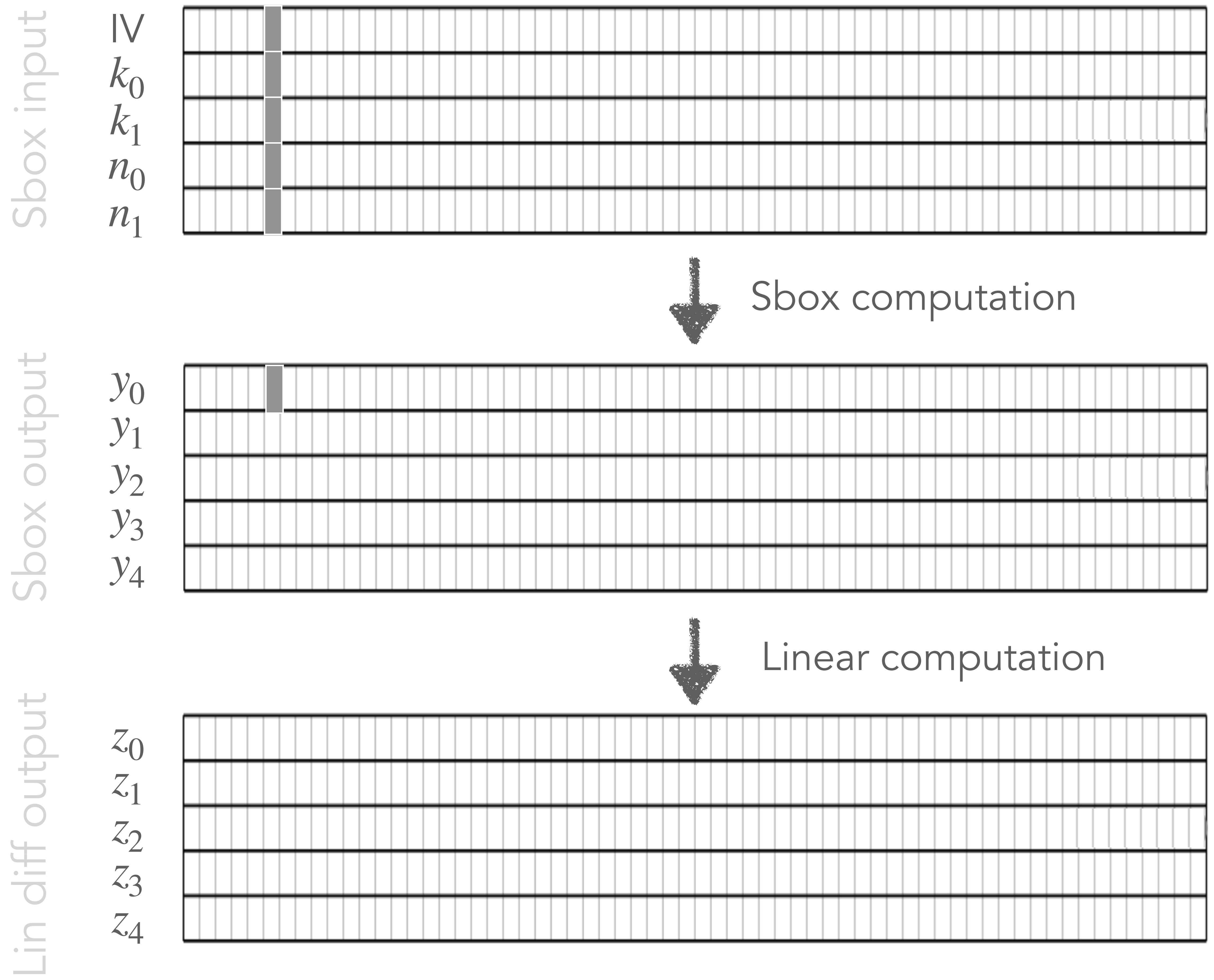
(2) Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \oplus k_1^j k_0^j \oplus k_0^j \mathsf{IV}^j \oplus k_1^j \oplus \mathsf{IV}^j$



(2) Fine-tuned linear diffusion output as attack point

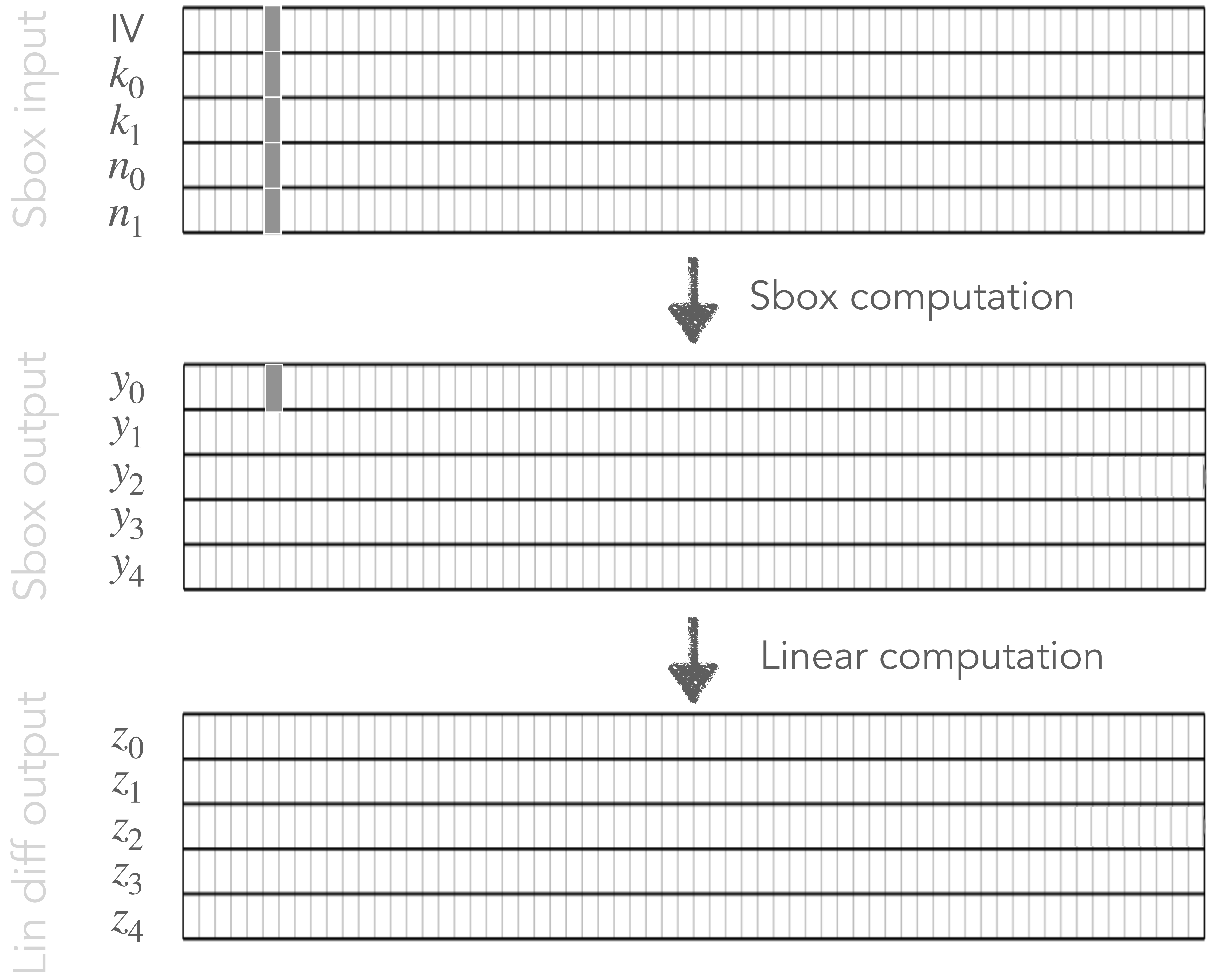
$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$



(2) Fine-tuned linear diffusion output as attack point

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathsf{IV}^j \oplus k_1^j \oplus \mathsf{IV}^j}$$

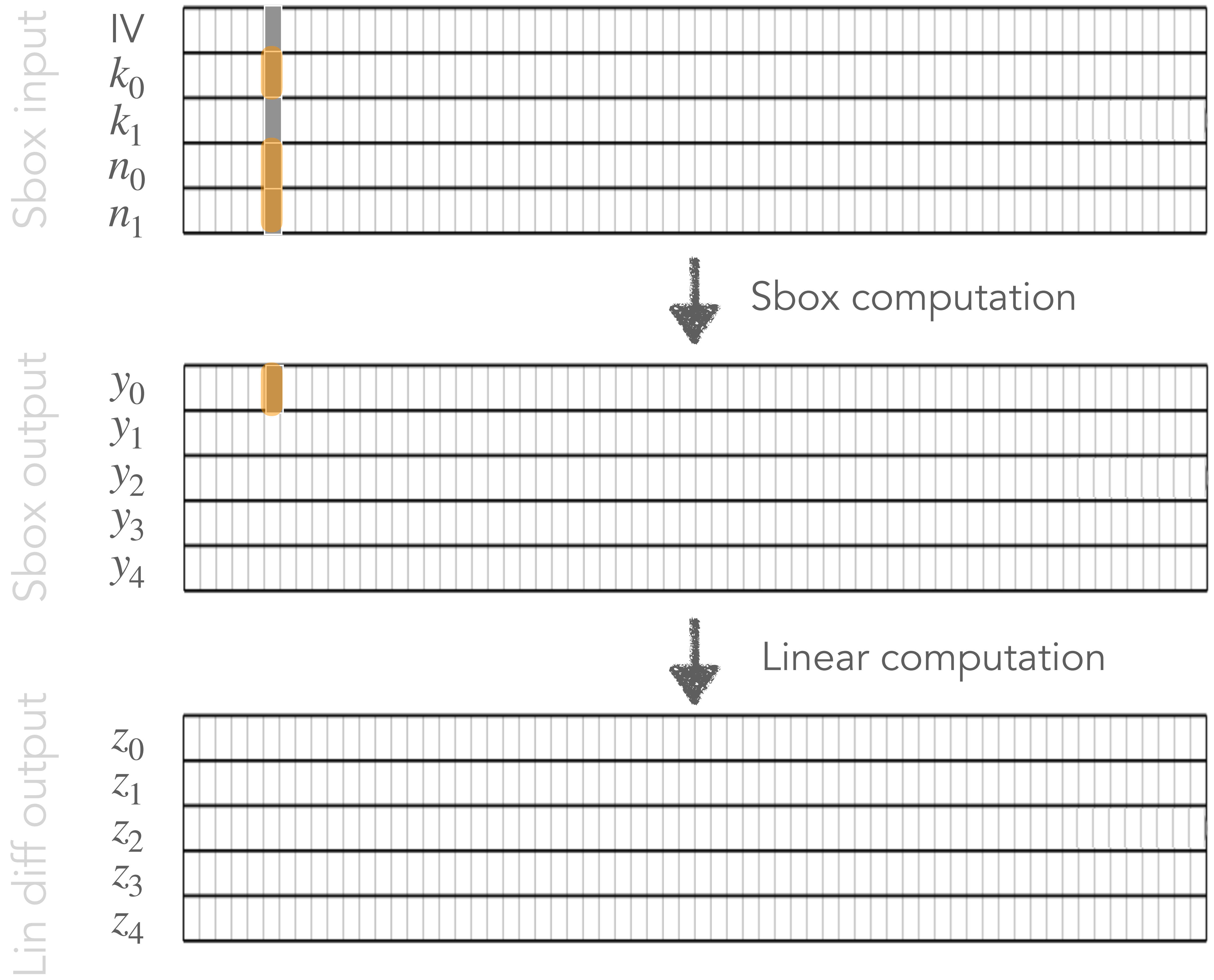
$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$



(2) Fine-tuned linear diffusion output as attack point

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

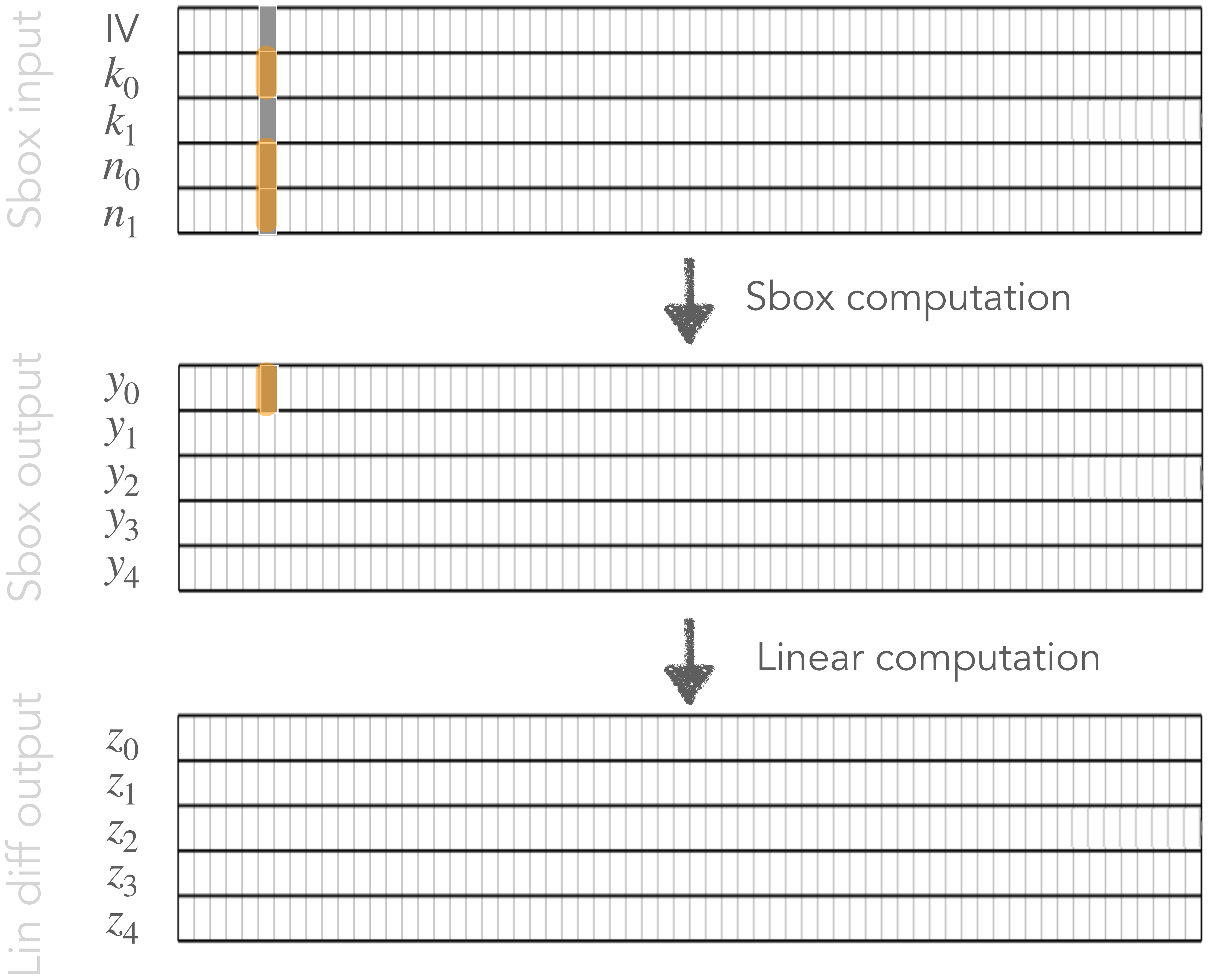


(2) Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

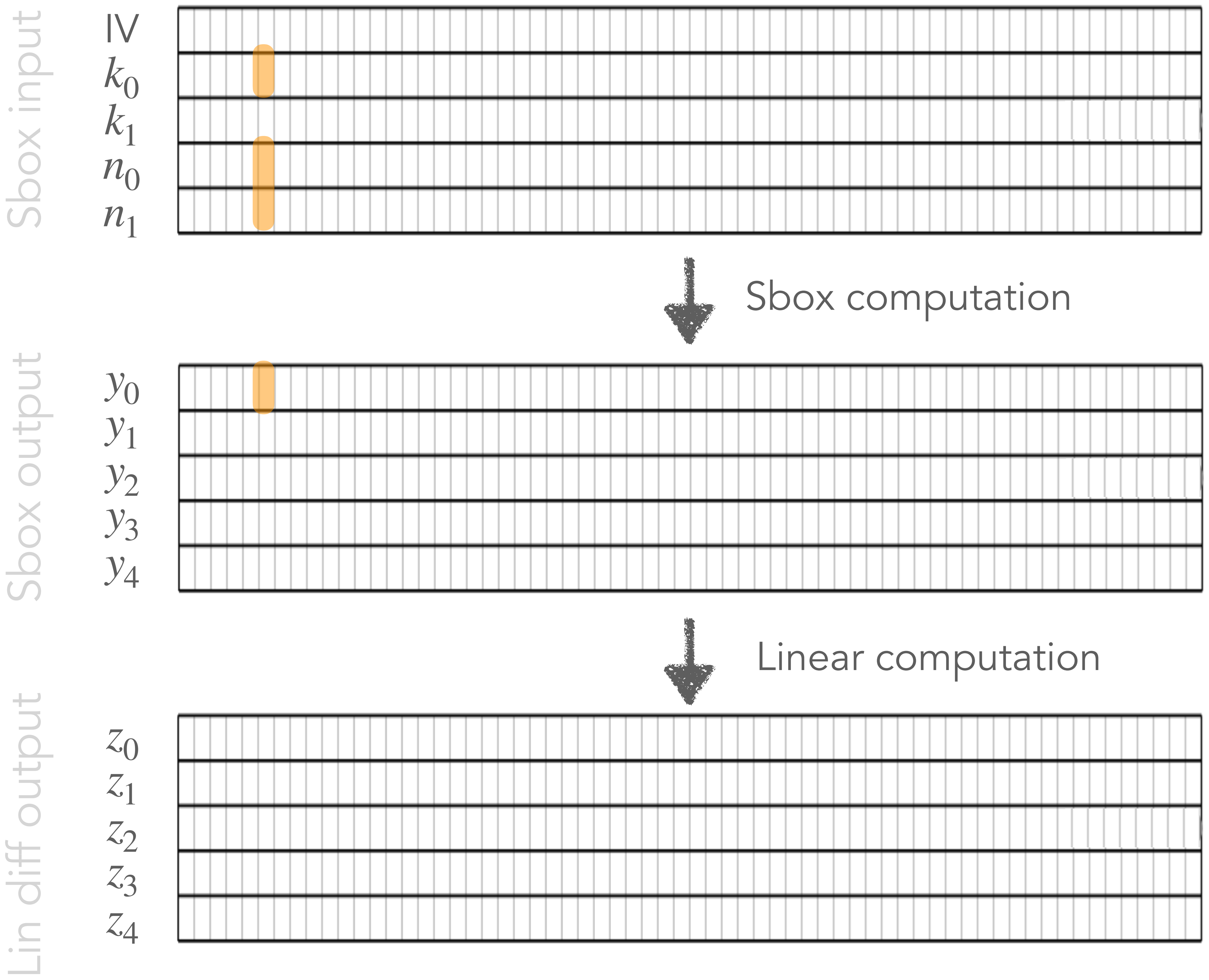
	k_0^j	
(n_0^j, n_1^j)	0	1
(0,0)	0	1
(0,1)	0	0
(1,0)	1	0
(1,1)	1	1
Correlation	0	



Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$



Fine-tuned linear diffusion output as attack point

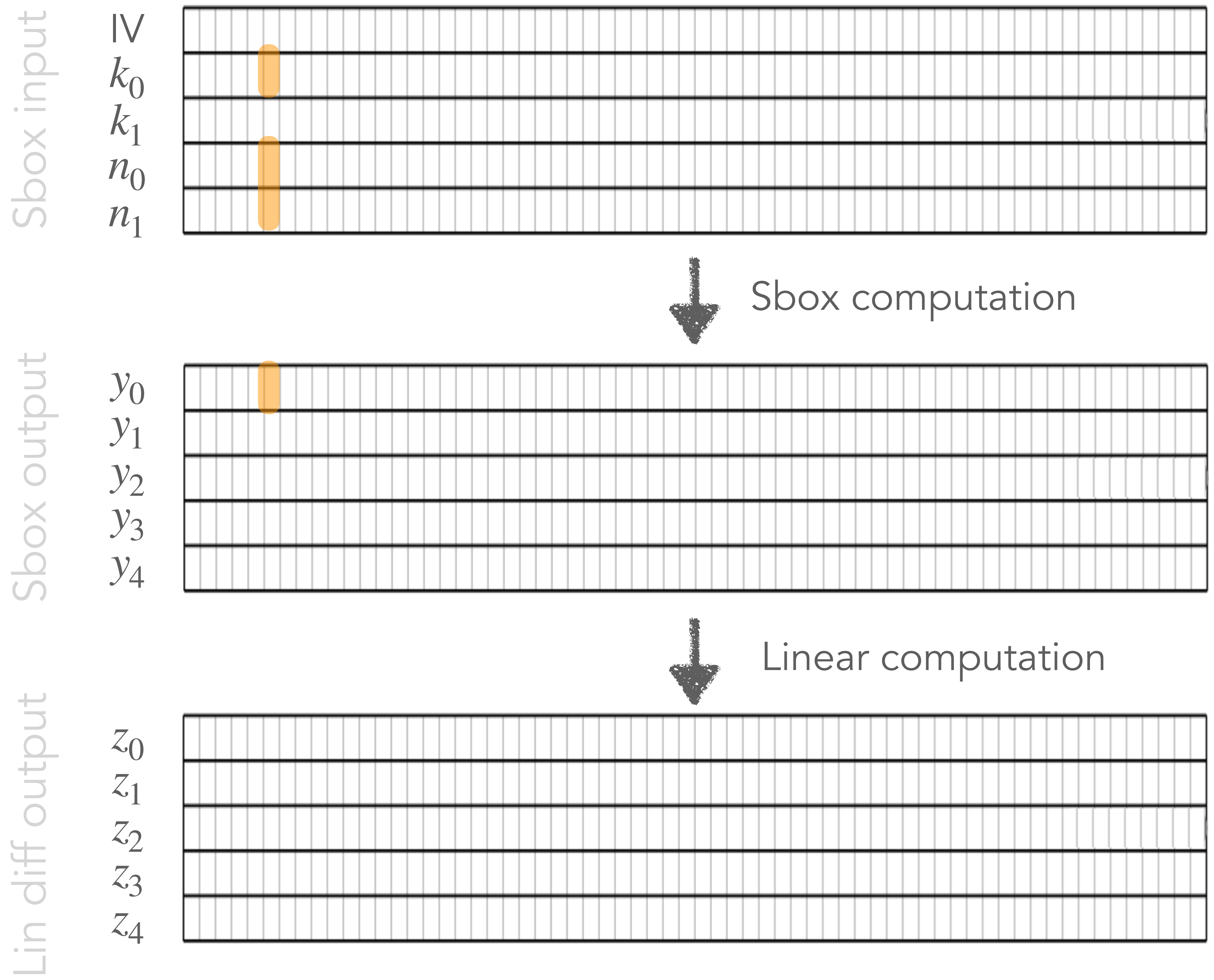
$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

Samwel and Daemen, 2018

Hardware implementation

with register at linear layer output (z_0^j)



Fine-tuned linear diffusion output as attack point

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$

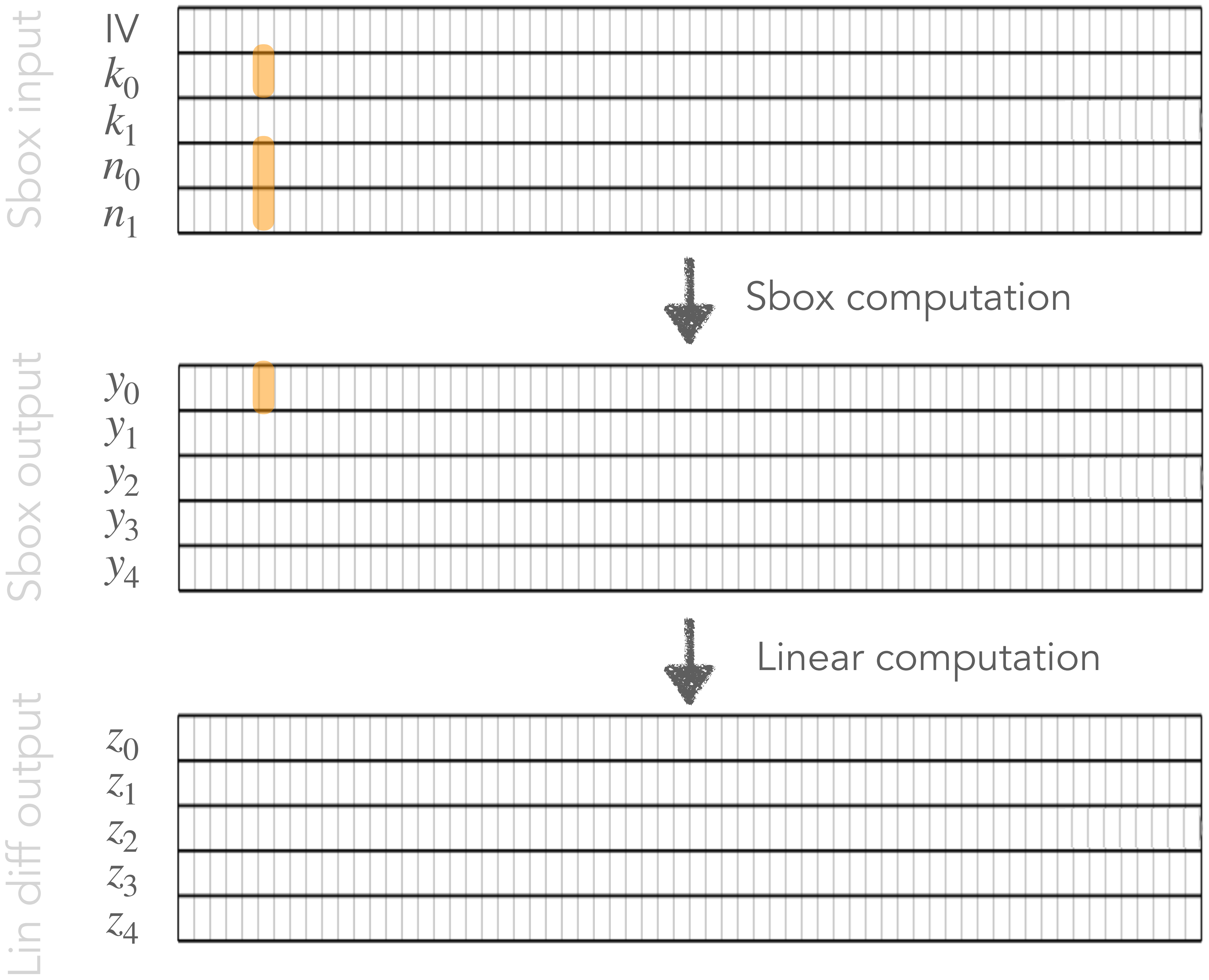
$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Similarly:

$$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$$



Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

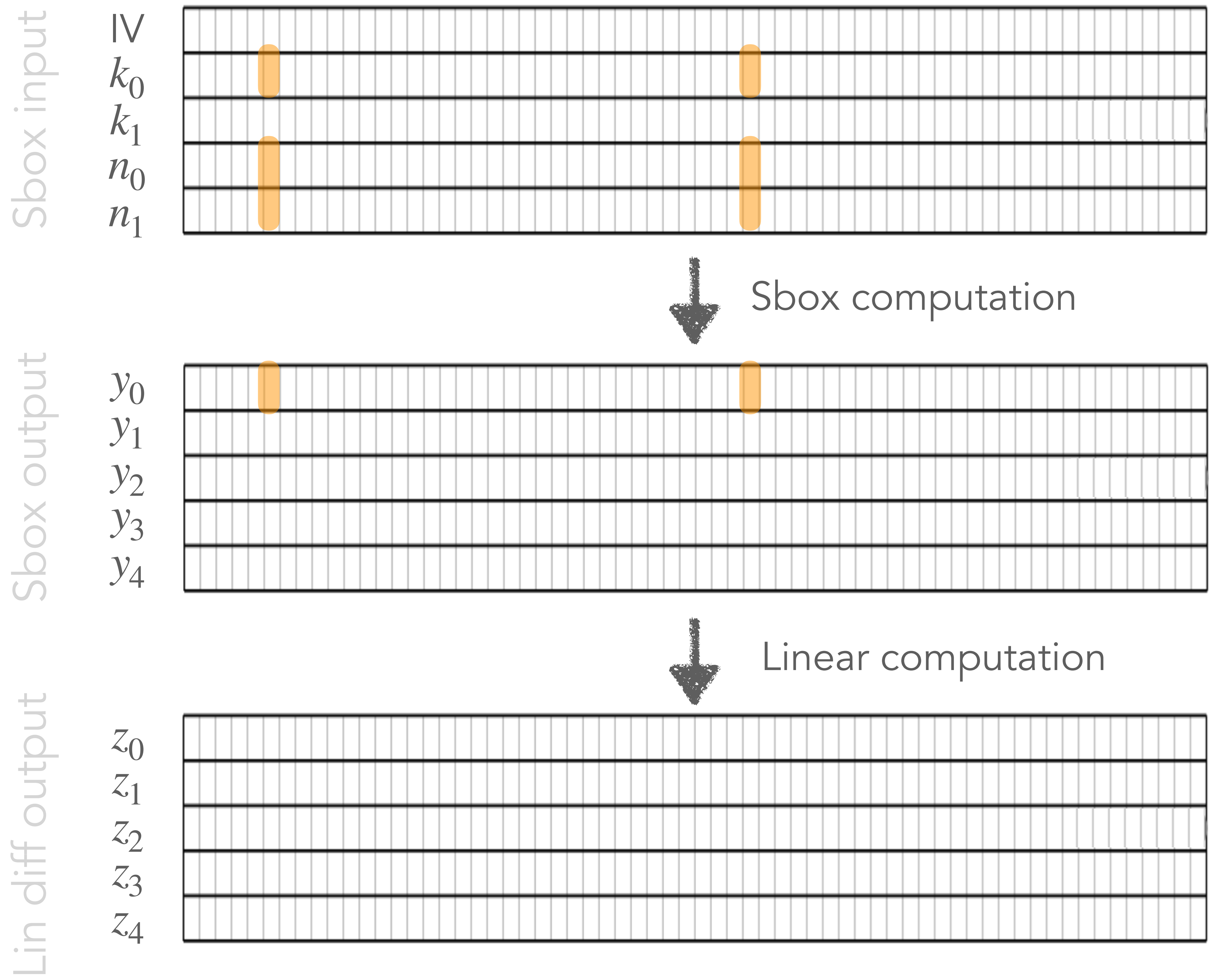
$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Similarly:

$$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$$



Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

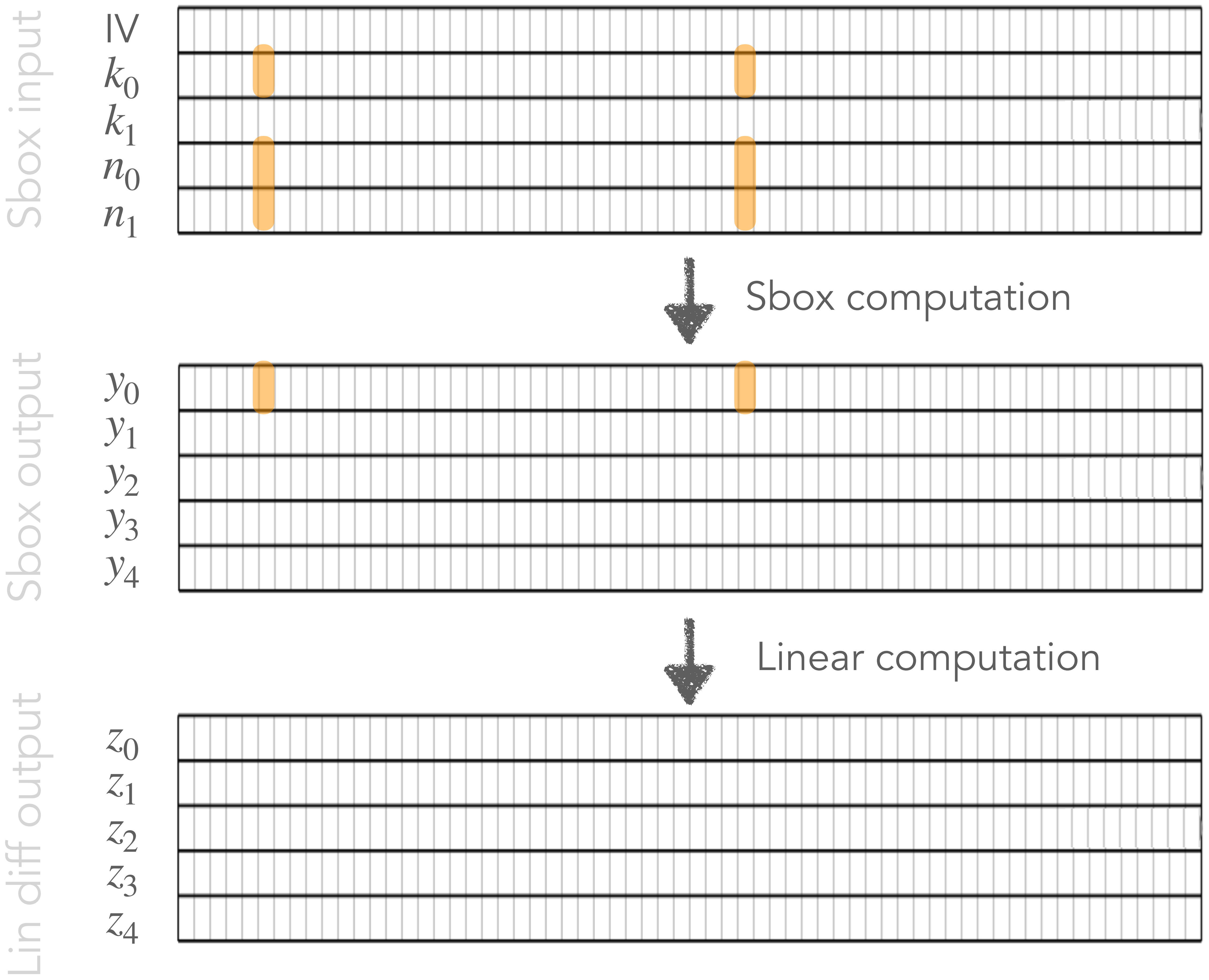
Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Similarly:

$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$

$\tilde{y}_0^{j+45} = k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}$



Fine-tuned linear diffusion output as attack point

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

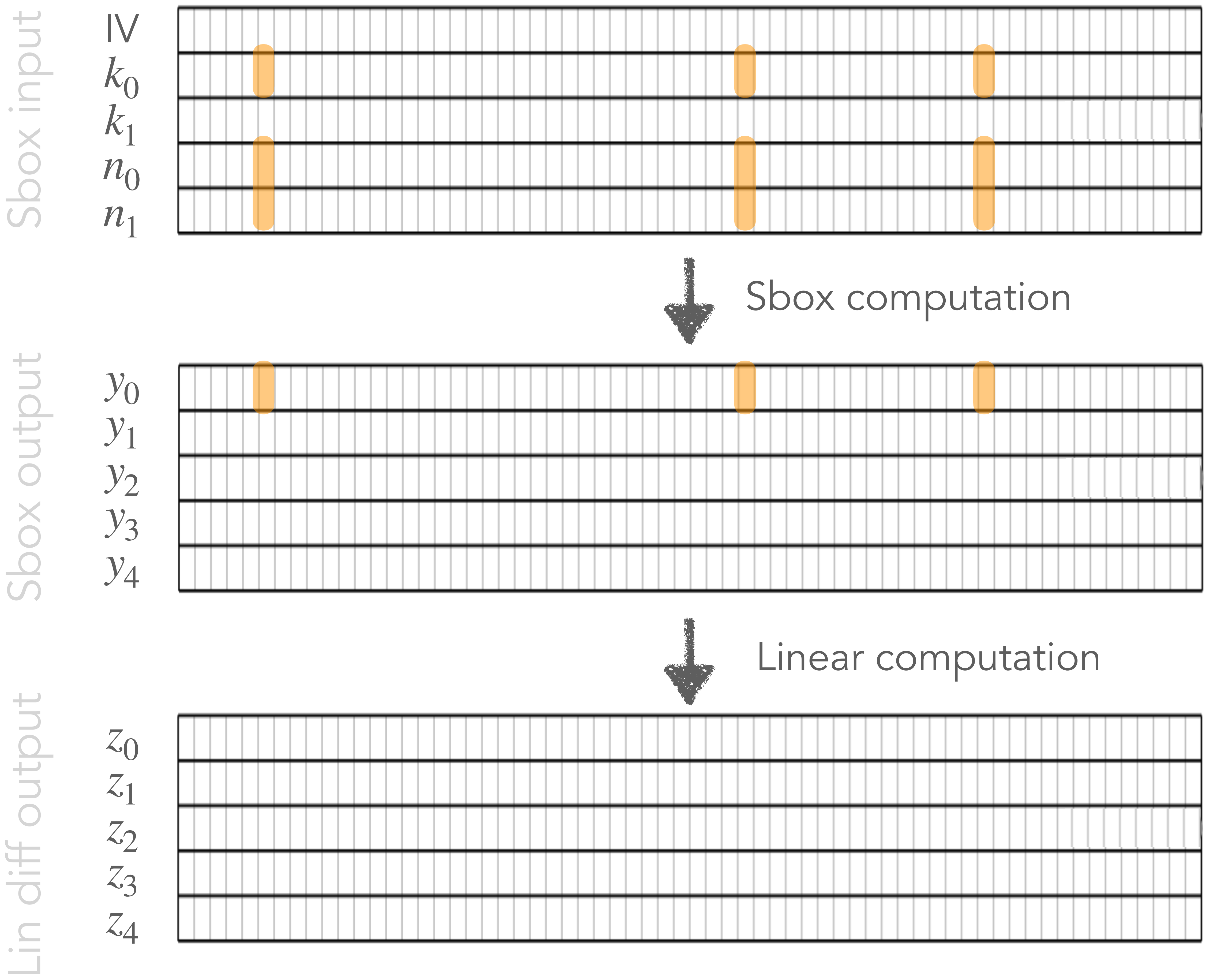
$\boxed{\text{Samwel and Daemen, 2018}}$

Hardware implementation
with register at linear layer output (z_0^j)

Similarly:

$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$

$\tilde{y}_0^{j+45} = k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}$



Fine-tuned linear diffusion output as attack point

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

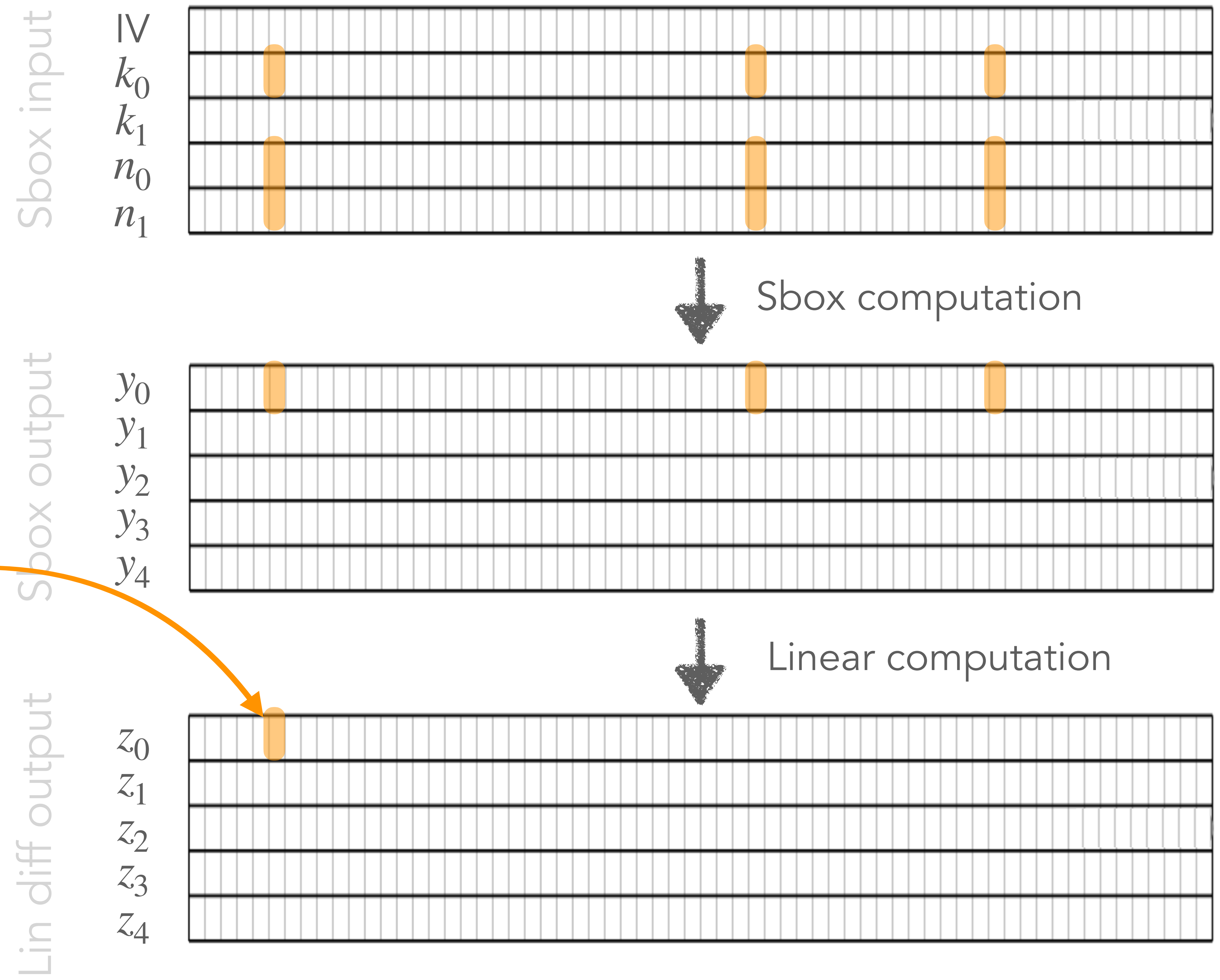
Similarly:

$$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$$

$$\tilde{y}_0^{j+45} = k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}$$

Linear computation:

$$\tilde{z}_0^j = \tilde{y}_0^j \oplus \tilde{y}_0^{j+36} \oplus \tilde{y}_0^{j+45}$$



Fine-tuned linear diffusion output as attack point

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Similarly:

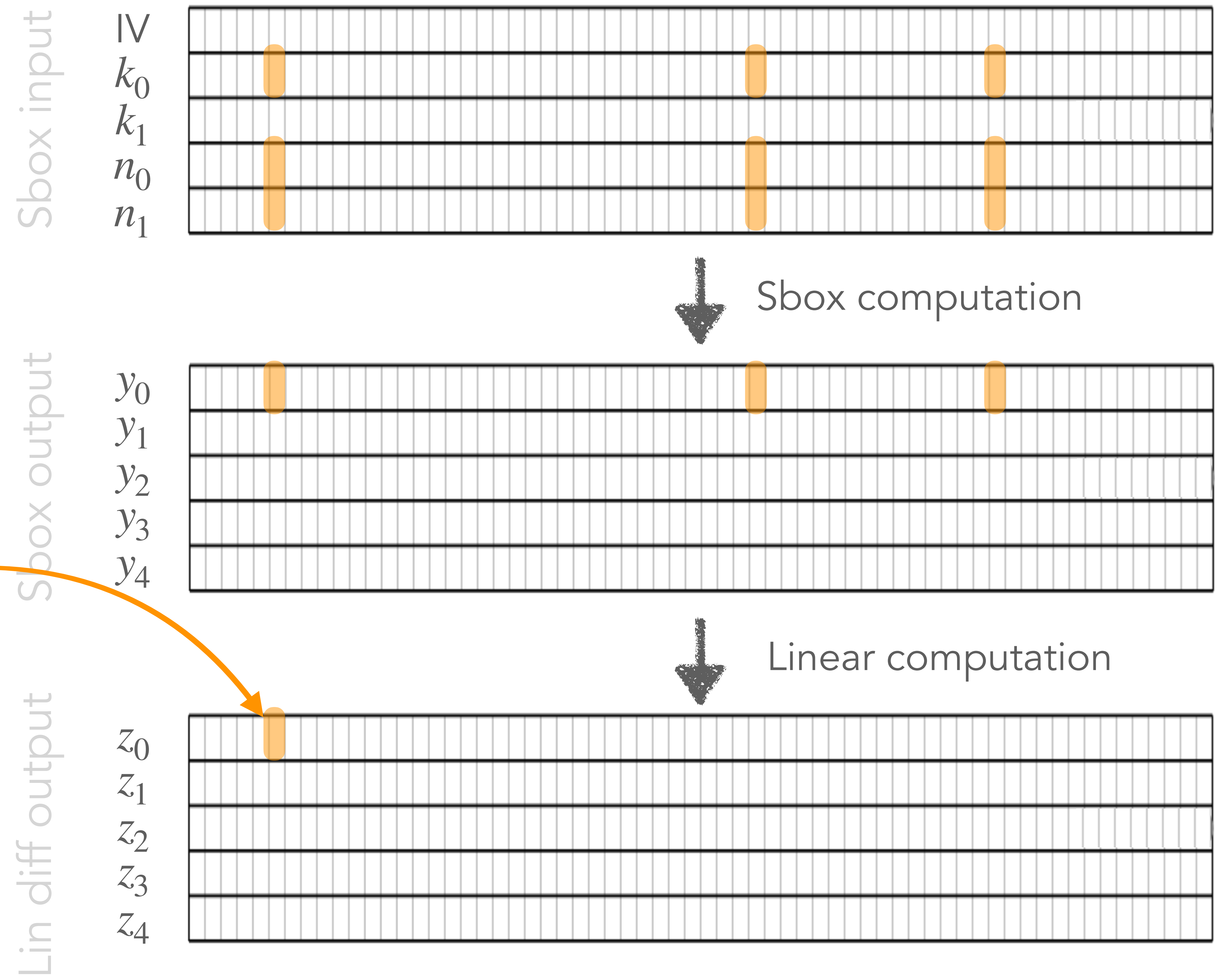
$$\tilde{y}_0^{j+36} = k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36}$$

$$\tilde{y}_0^{j+45} = k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}$$

Linear computation:

$$\tilde{z}_0^j = \tilde{y}_0^j \oplus \tilde{y}_0^{j+36} \oplus \tilde{y}_0^{j+45}$$

$$\begin{aligned}\tilde{z}_0^j = {} & k_0^j(n_1^j \oplus 1) \oplus n_0^j \\ & \oplus k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36} \\ & \oplus k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}\end{aligned}$$



Fine-tuned linear diffusion output as attack point

$$
\begin{aligned}
\tilde{z}_0^j = {} & k_0^j(n_1^j \oplus 1) \oplus n_0^j \\
& \oplus k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36} \\
& \oplus k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}
\end{aligned}
$$

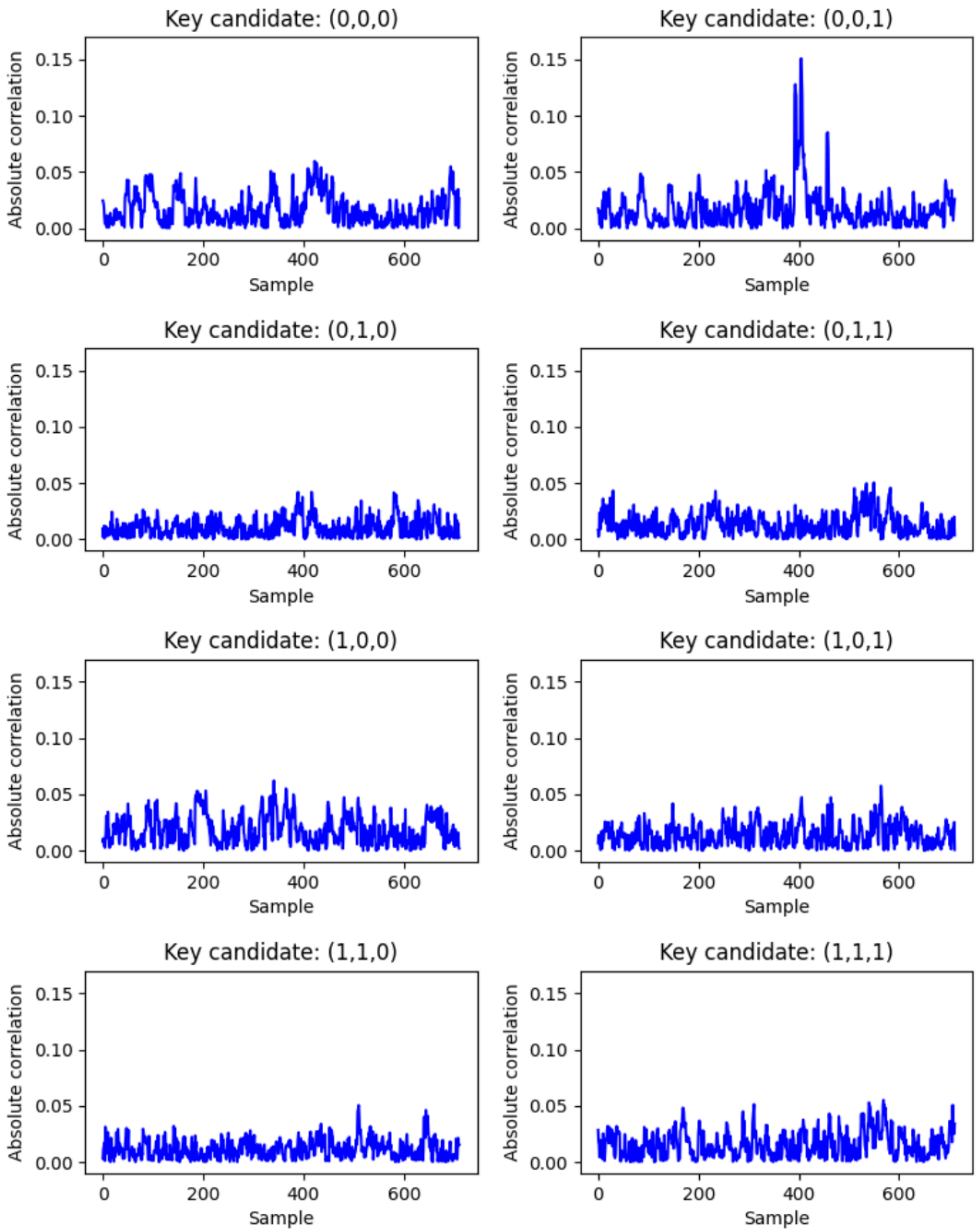
Fine-tuned linear diffusion output as attack point

Correlations of distributions associated to all key pairs

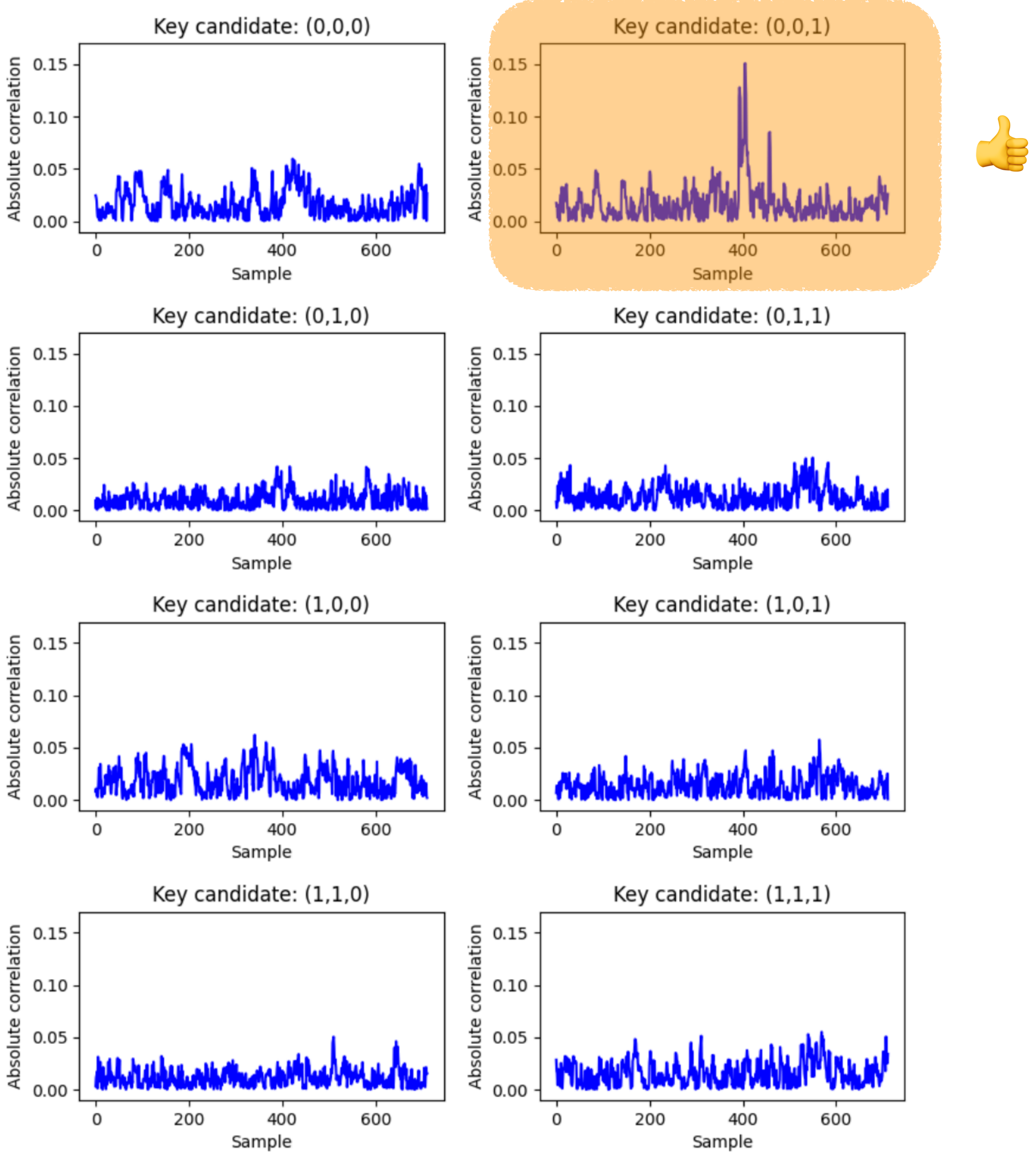
$(k_0^j, k_0^{j+36}, k_0^{j+45})$	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
(0,0,0)	1	-	-	-	-	-	-	-
(0,0,1)	-	1	-	-	-	-	-	-
(0,1,0)	-	-	1	-	-	-	-	-
(0,1,1)	-	-	-	1	-	-	-	-
(1,0,0)	-	-	-	-	1	-	-	-
(1,0,1)	-	-	-	-	-	1	-	-
(1,1,0)	-	-	-	-	-	-	1	-
(1,1,1)	-	-	-	-	-	-	-	1

$$
\begin{aligned}
\tilde{z}_0^j = {} & k_0^j(n_1^j \oplus 1) \oplus n_0^j \\
& \oplus k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36} \\
& \oplus k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}
\end{aligned}
$$

Fine-tuned linear diffusion output as attack point



Fine-tuned linear diffusion output as attack point



$\tilde{y}^j_0$ as attack point ?

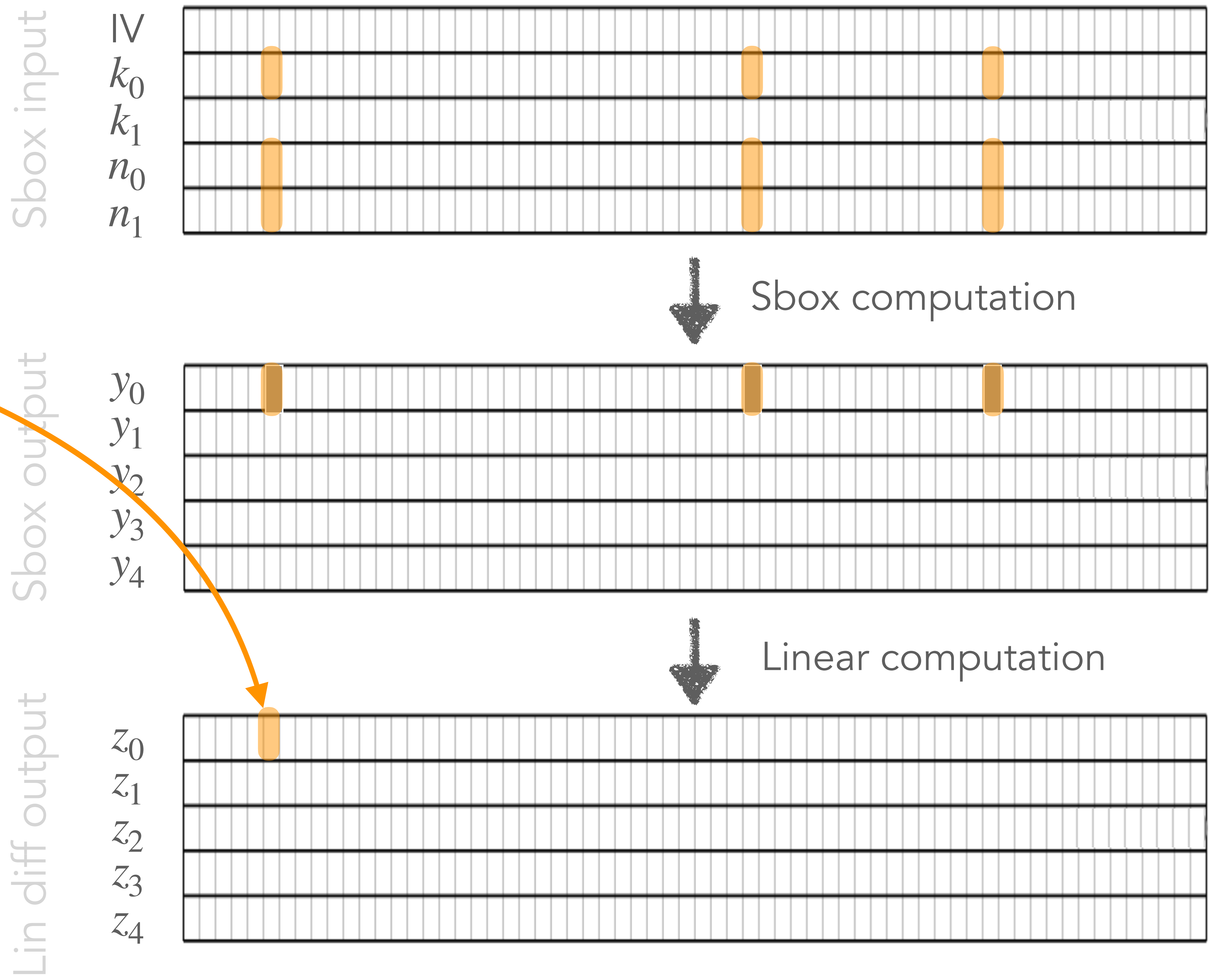
$y^j_0 = k^j_0(n^j_1 \oplus 1) \oplus n^j_0 \cancel{\oplus k^j_1 k^j_0 \oplus k^j_0 \mathrm{IV}^j \oplus k^j_1 \oplus \mathrm{IV}^j}$

$\tilde{y}^j_0 = k^j_0(n^j_1 \oplus 1) \oplus n^j_0$

Samwel and Daemen, 2018

Hardware implementation

with register at linear layer output (z^j_0)



$\tilde{y}_0^j$ as attack point ?

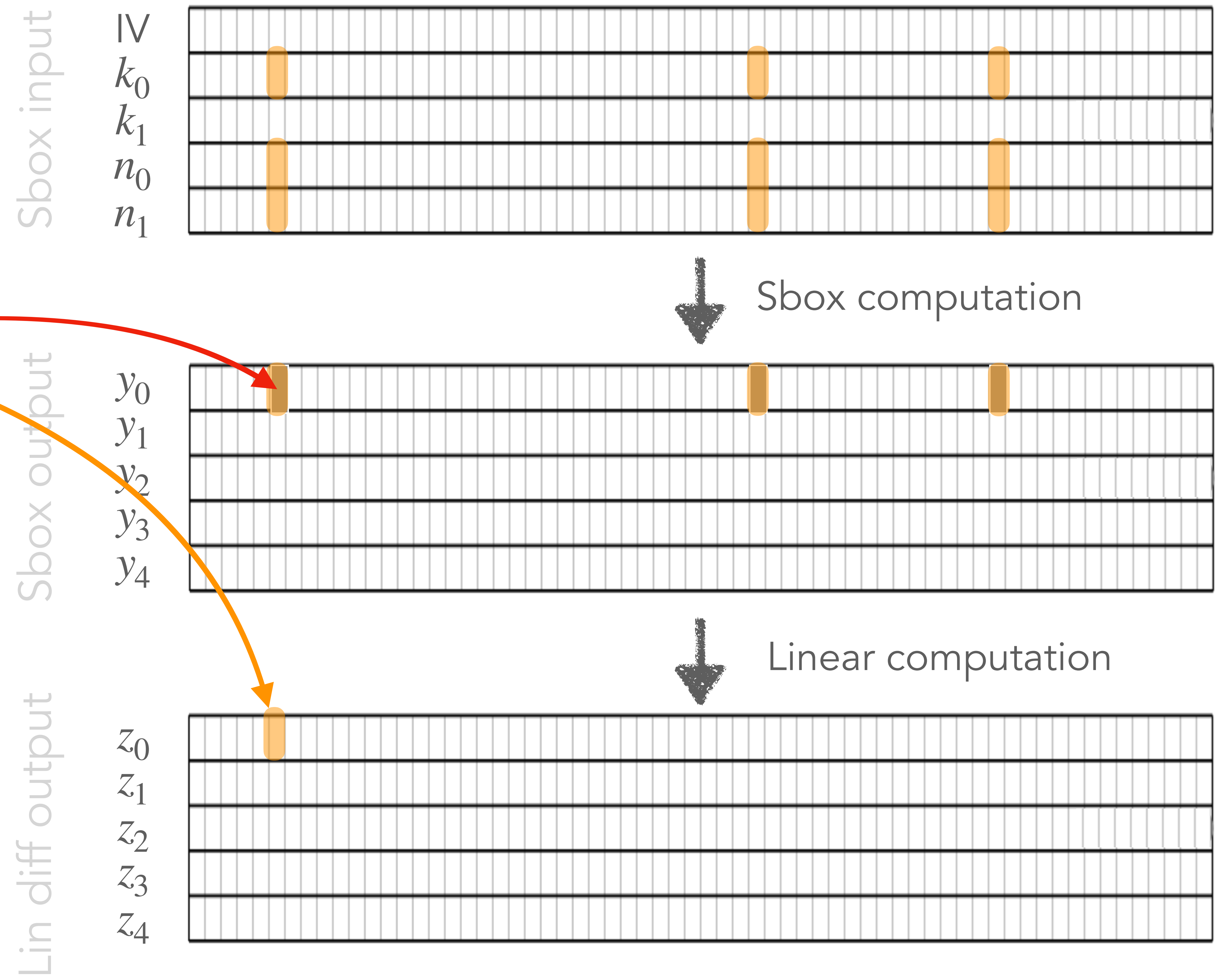
$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?



$\tilde{y}_0^j$ as attack point ?

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

$\boxed{\text{Samwel and Daemen, 2018}}$

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?

$\tilde{y}_0^j$ as attack point ?

$$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j = \begin{cases} n_0^j & \text{if } k_0^j = 0, \\ n_0^j \oplus n_1^j \oplus 1 & \text{if } k_0^j = 1. \end{cases}$$

$\tilde{y}_0^j$ as attack point ?

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j \mathrm{IV}^j \oplus k_1^j \oplus \mathrm{IV}^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j = \begin{cases} n_0^j & \text{if } k_0^j = 0, \\ n_0^j \oplus n_1^j \oplus 1 & \text{if } k_0^j = 1. \end{cases}$$

→ Correlated with activity of n_0

$\tilde{y}_0^j$ as attack point ?

$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j = \begin{cases} n_0^j & \text{if } k_0^j = 0, \\ n_0^j \oplus n_1^j \oplus 1 & \text{if } k_0^j = 1. \end{cases}$$

→ Correlated with activity of n_0

→ Correlated with activity of $n_0 \oplus n_1$

$\tilde{y}_0^j$ as attack point ?

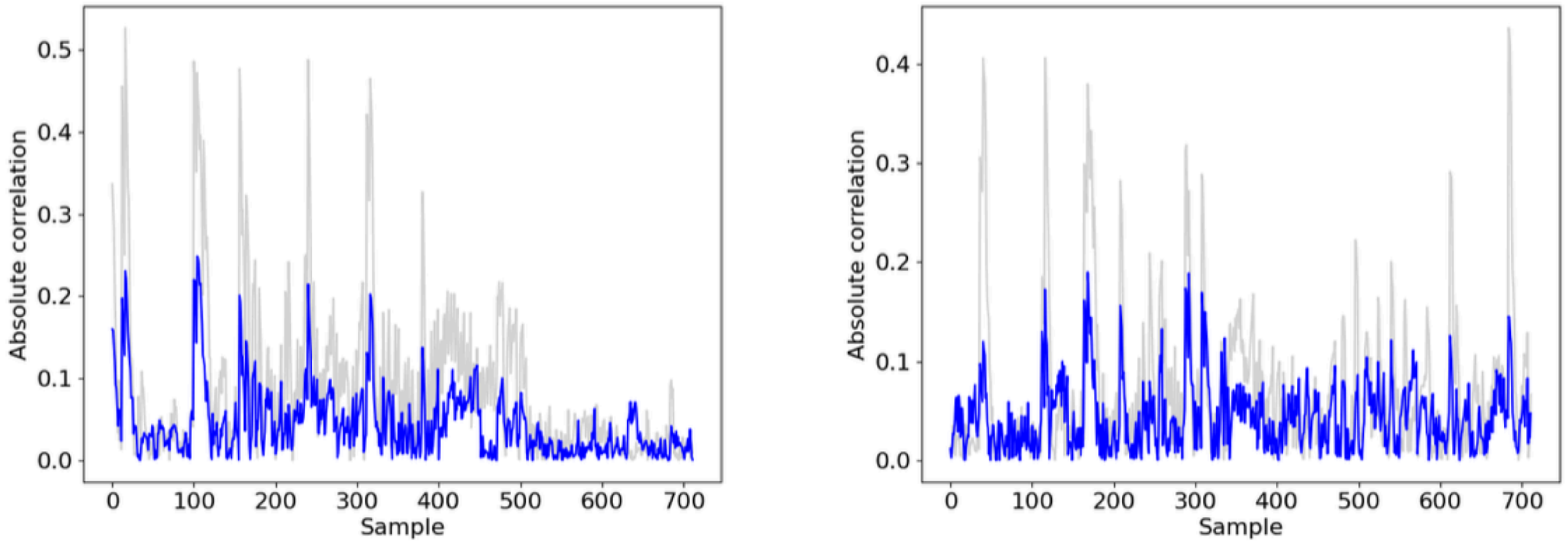
$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?



(a) $k_0^j = 0$ (b) $k_0^j = 1$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j = \begin{cases} n_0^j & \text{if } k_0^j = 0, \\ n_0^j \oplus n_1^j \oplus 1 & \text{if } k_0^j = 1. \end{cases}$$

→ Correlated with activity of n_0

→ Correlated with activity of $n_0 \oplus n_1$

$\tilde{y}_0^j$ as attack point ?

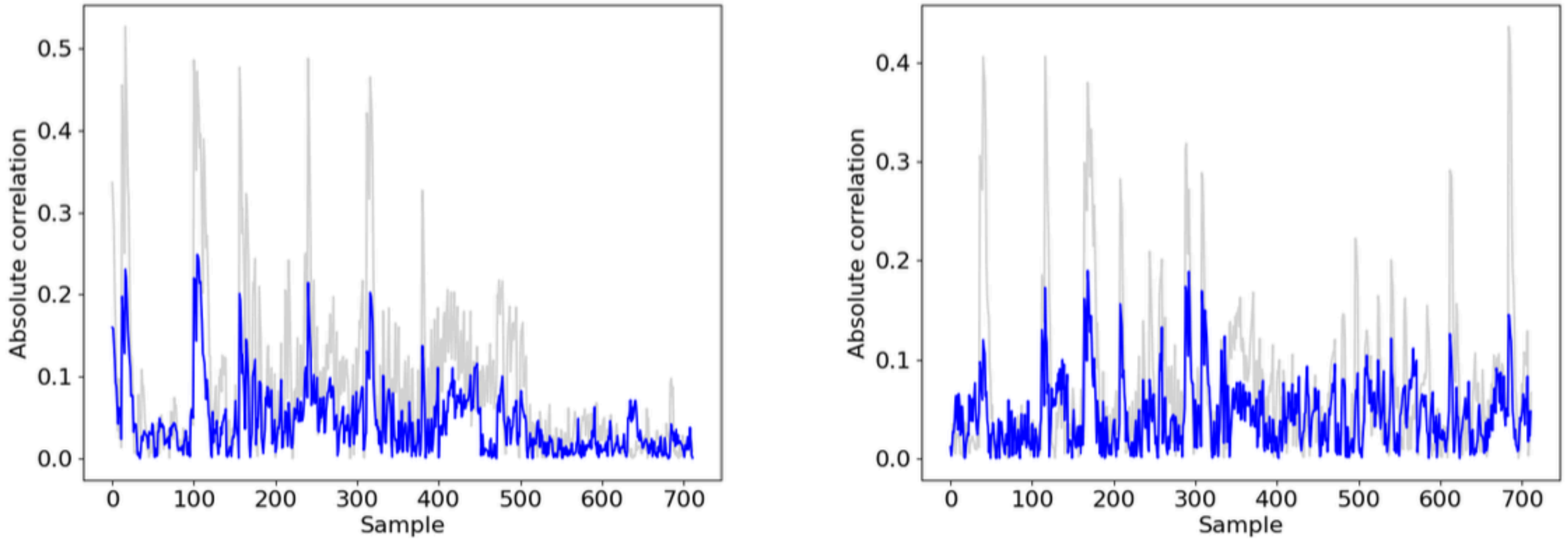
$y_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j \cancel{\oplus k_1^j k_0^j \oplus k_0^j IV^j \oplus k_1^j \oplus IV^j}$

$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j$

Samwel and Daemen, 2018

Hardware implementation
with register at linear layer output (z_0^j)

Software implementation:
Can we use $\tilde{y}_0^j$ as attack point ?



(a) $k_0^j = 0$ (b) $k_0^j = 1$

$$\tilde{y}_0^j = k_0^j(n_1^j \oplus 1) \oplus n_0^j = \begin{cases} n_0^j & \text{if } k_0^j = 0, \\ n_0^j \oplus n_1^j \oplus 1 & \text{if } k_0^j = 1. \end{cases}$$

→ Correlated with activity of n_0

→ Correlated with activity of $n_0 \oplus n_1$

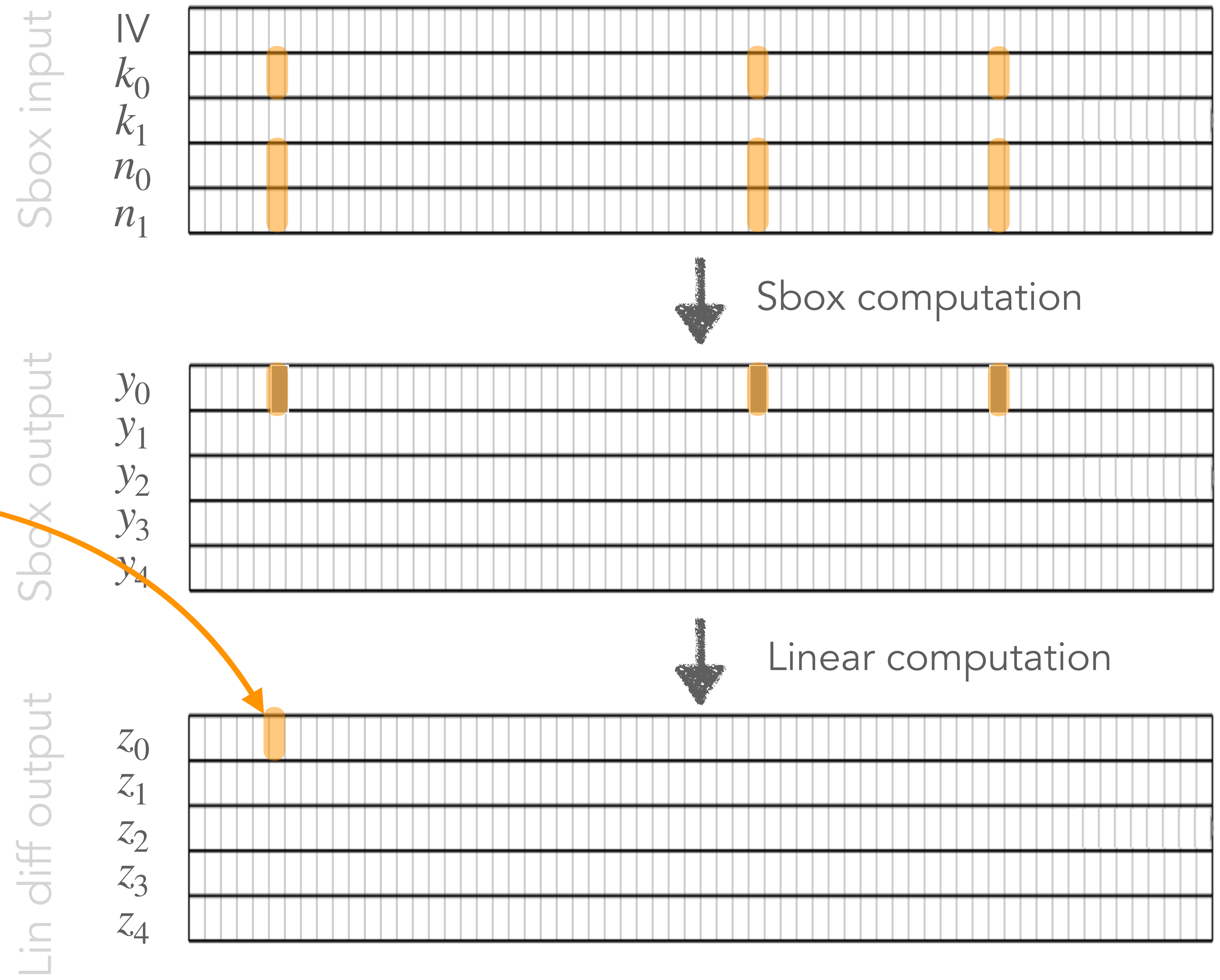
→ Not effective for CPA attacks

The best choice

for both hardware and software implementations

Samwel and Daemen, 2018

$$
\begin{aligned}
\tilde{z}_0^j = {} & k_0^j(n_1^j \oplus 1) \oplus n_0^j \\
& \oplus k_0^{j+36}(n_1^{j+36} \oplus 1) \oplus n_0^{j+36} \\
& \oplus k_0^{j+45}(n_1^{j+45} \oplus 1) \oplus n_0^{j+45}
\end{aligned}
$$

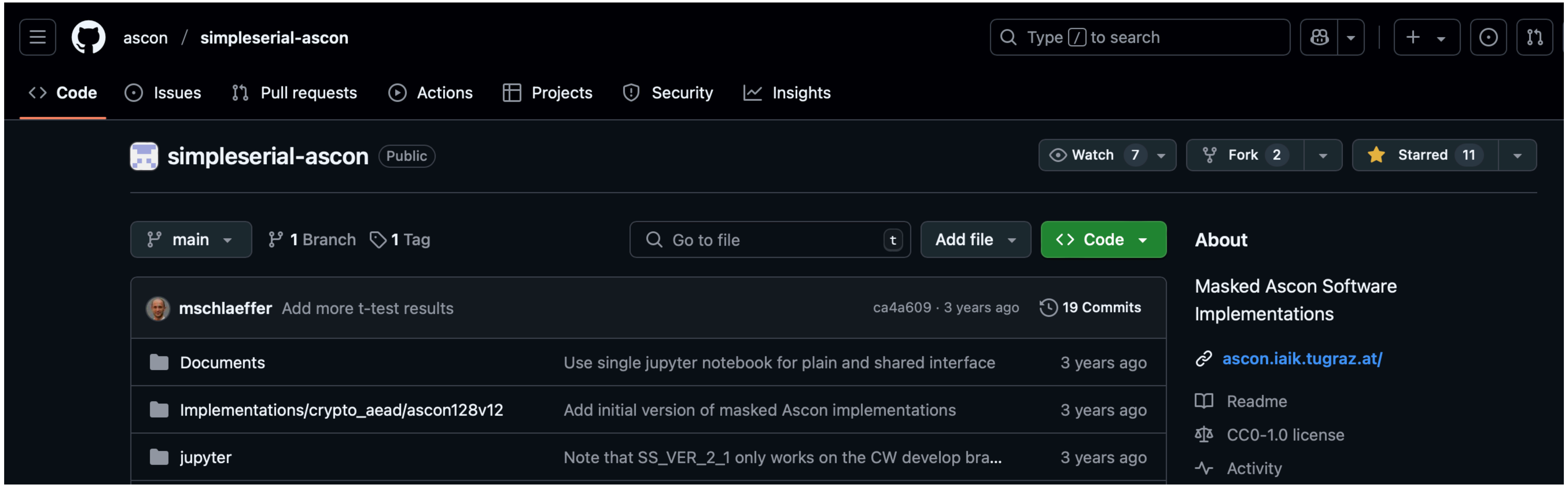


Second-Order CPA attack

Target implementation

Masked software implementations with 2 shares
by Ascon team

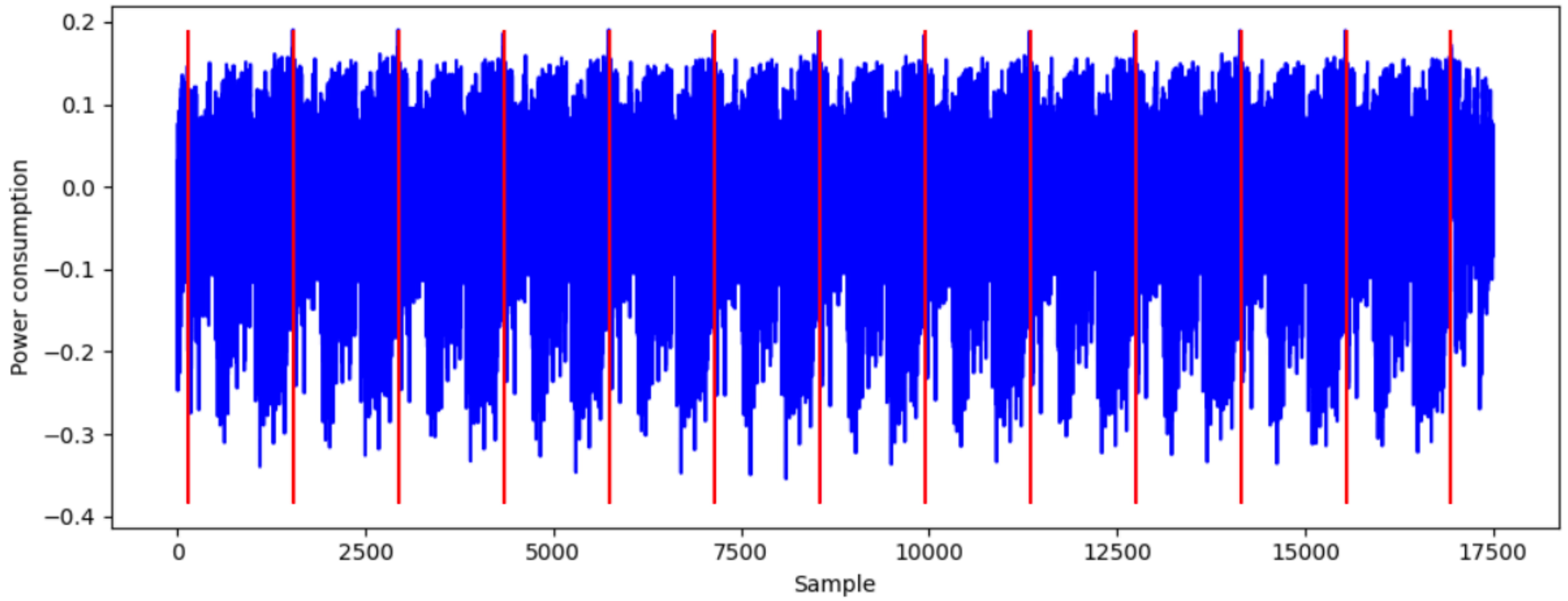
https://github.com/ascon/simpleserial-ascon



Target implementation

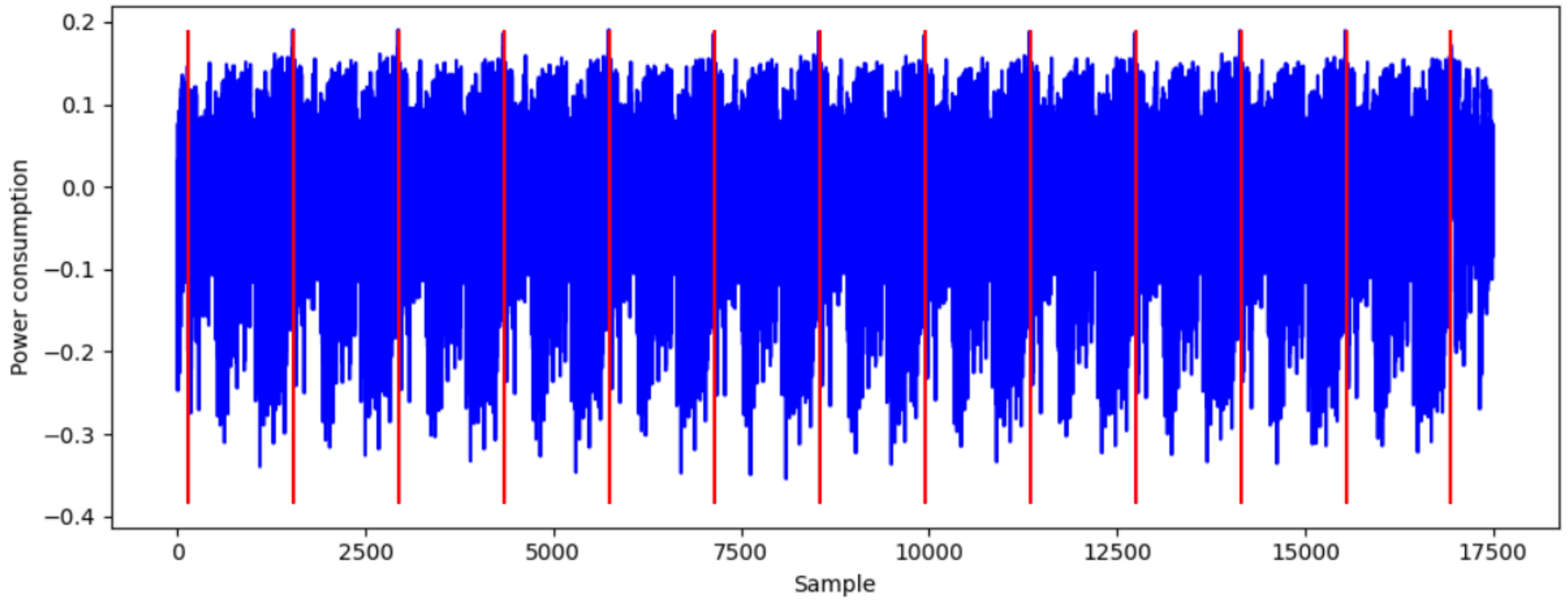
Target implementation

Power consumption of the first 12 rounds from ChipWhisperer ARM with Cortex-M3 core:

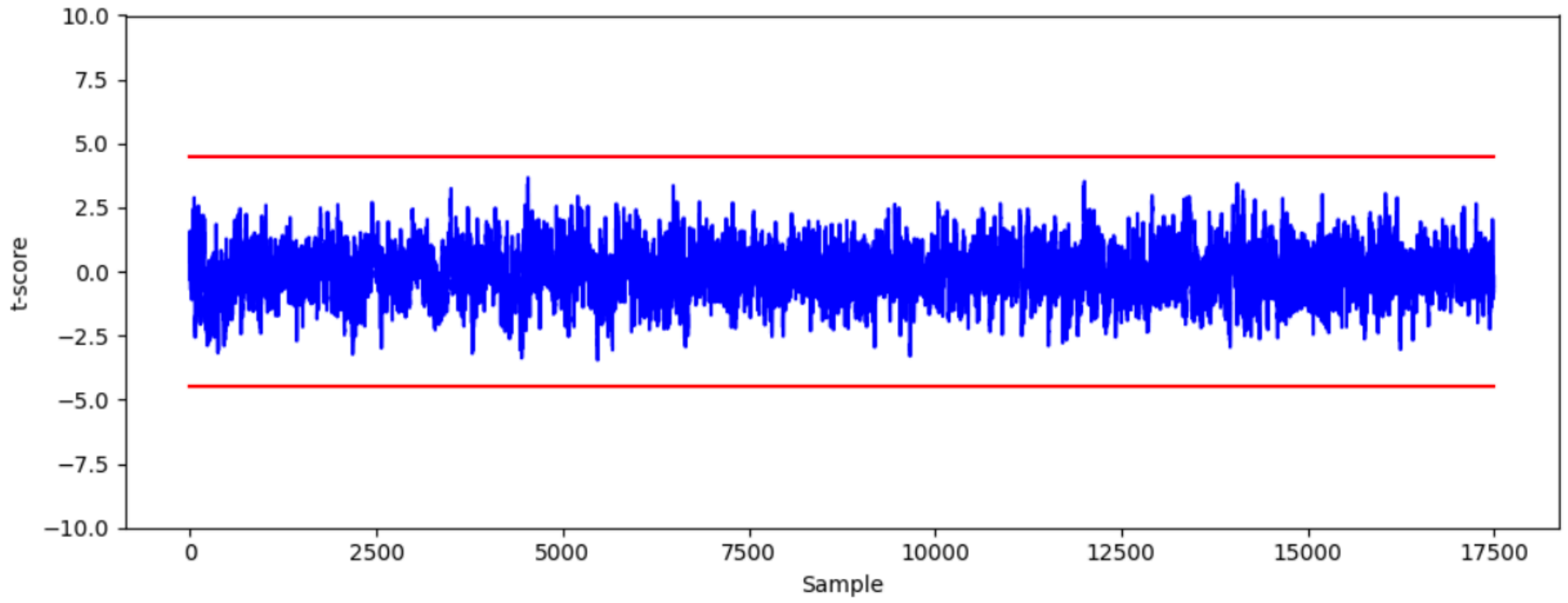


Target implementation

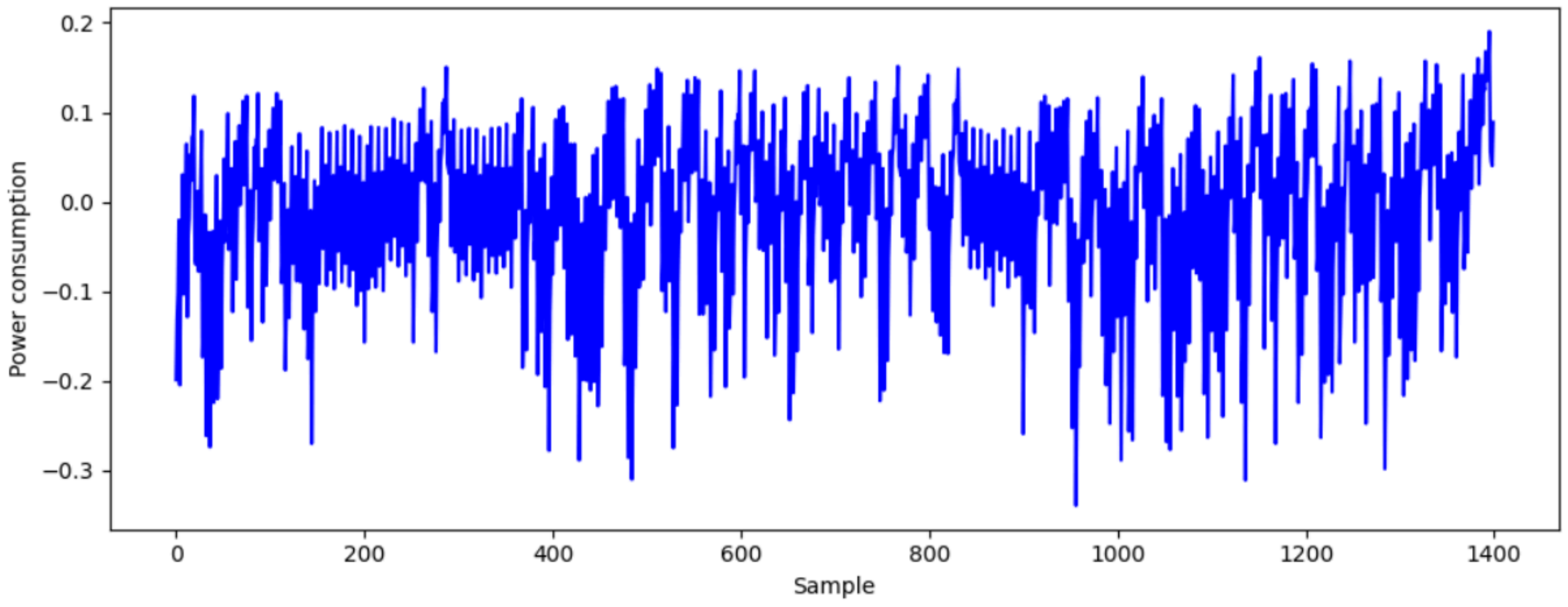
Power consumption of the first 12 rounds from ChipWhisperer ARM with Cortex-M3 core:



Verify first-order leakage by TVLA:

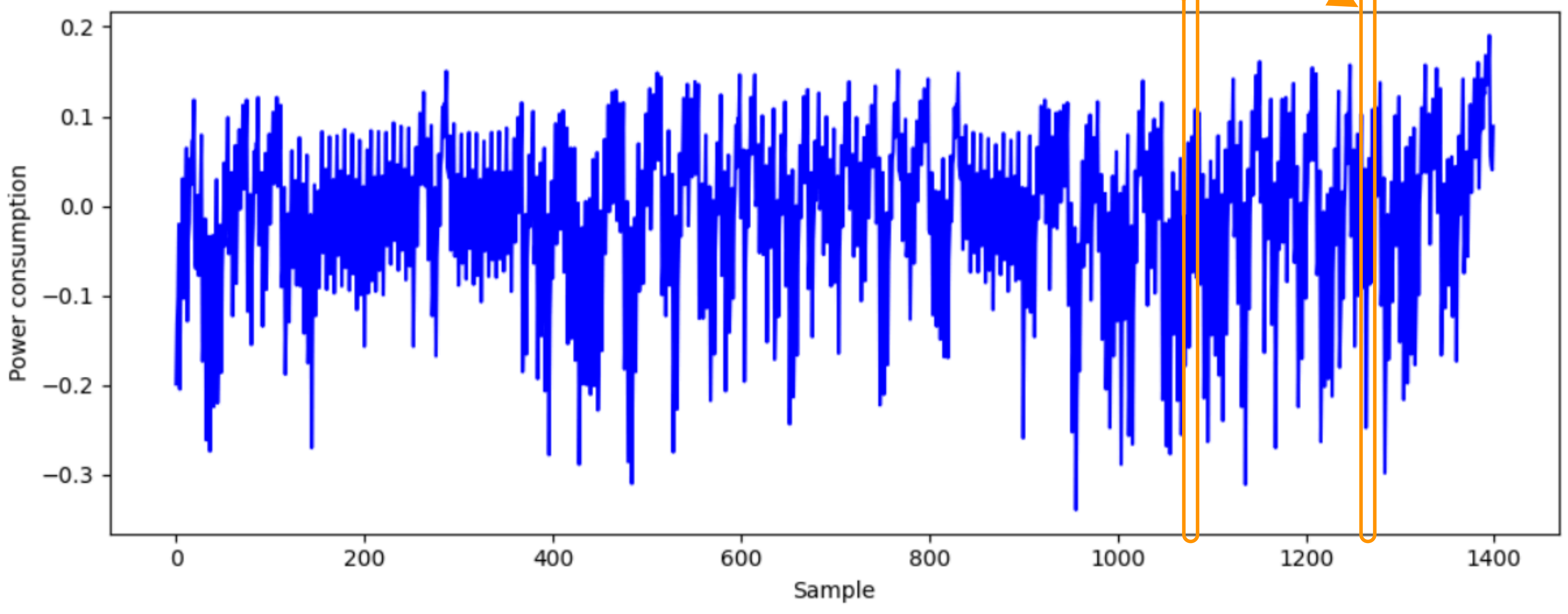


Second-Order CPA



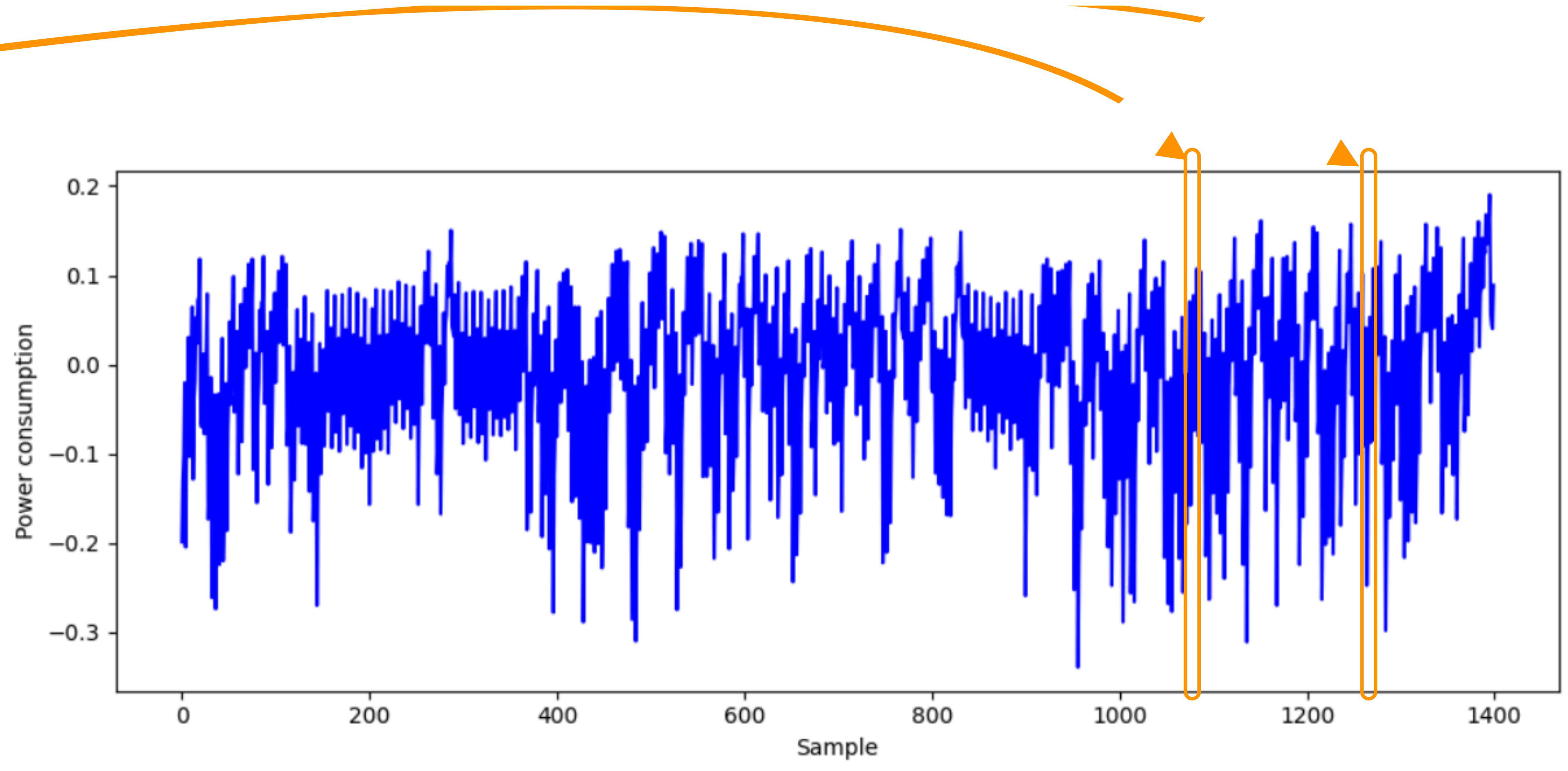
Second-Order CPA

Combine two points on the trace
(by normalized product)



Second-Order CPA

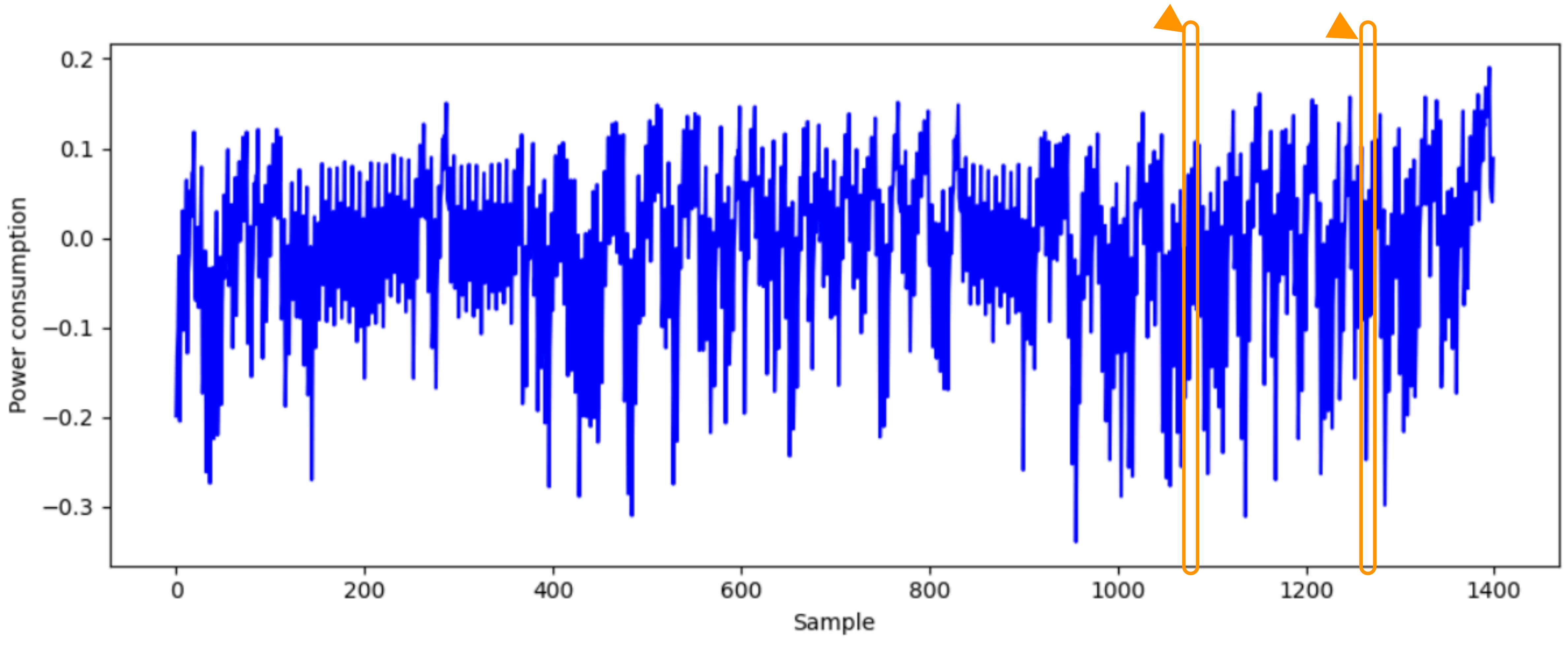
Combine two points on the trace
(by normalized product)



Optimizations:

Second-Order CPA

Combine two points on the trace (by normalized product)

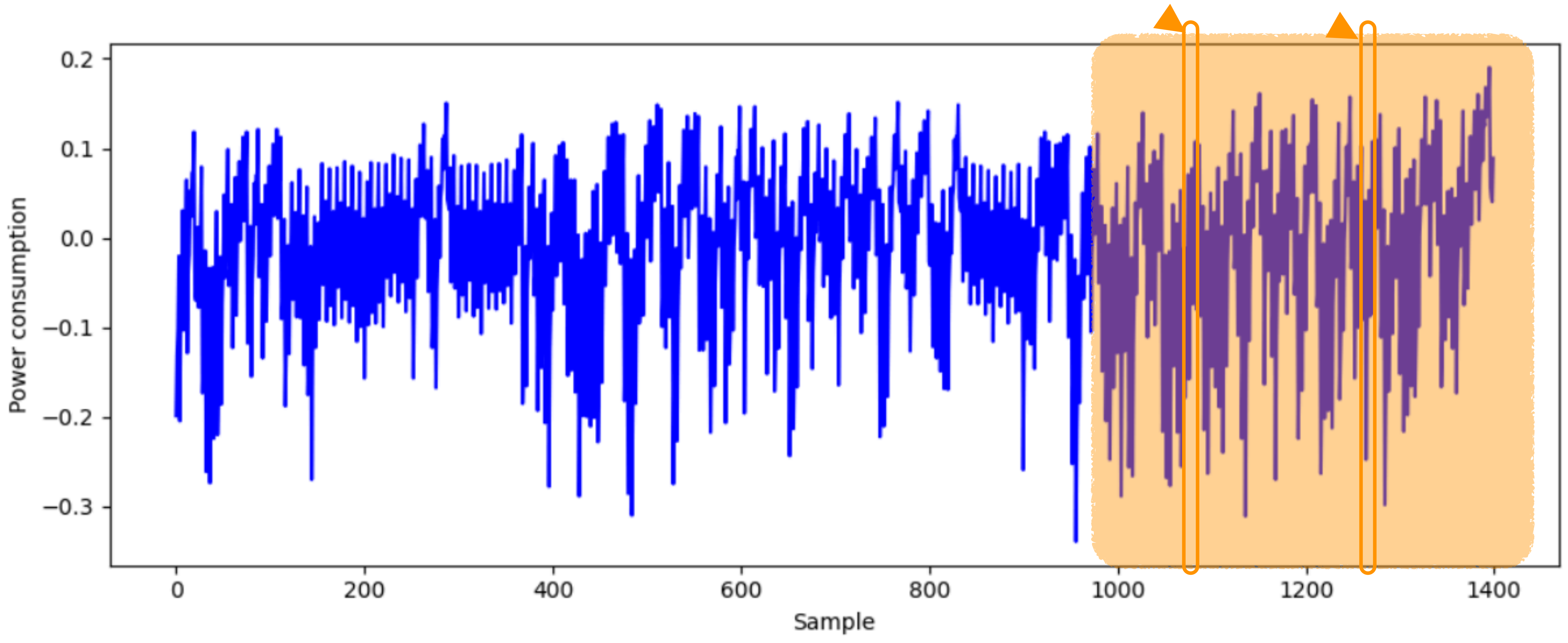


Optimizations:

- Attack point at linear layer output

Second-Order CPA

Combine two points on the trace
(by normalized product)

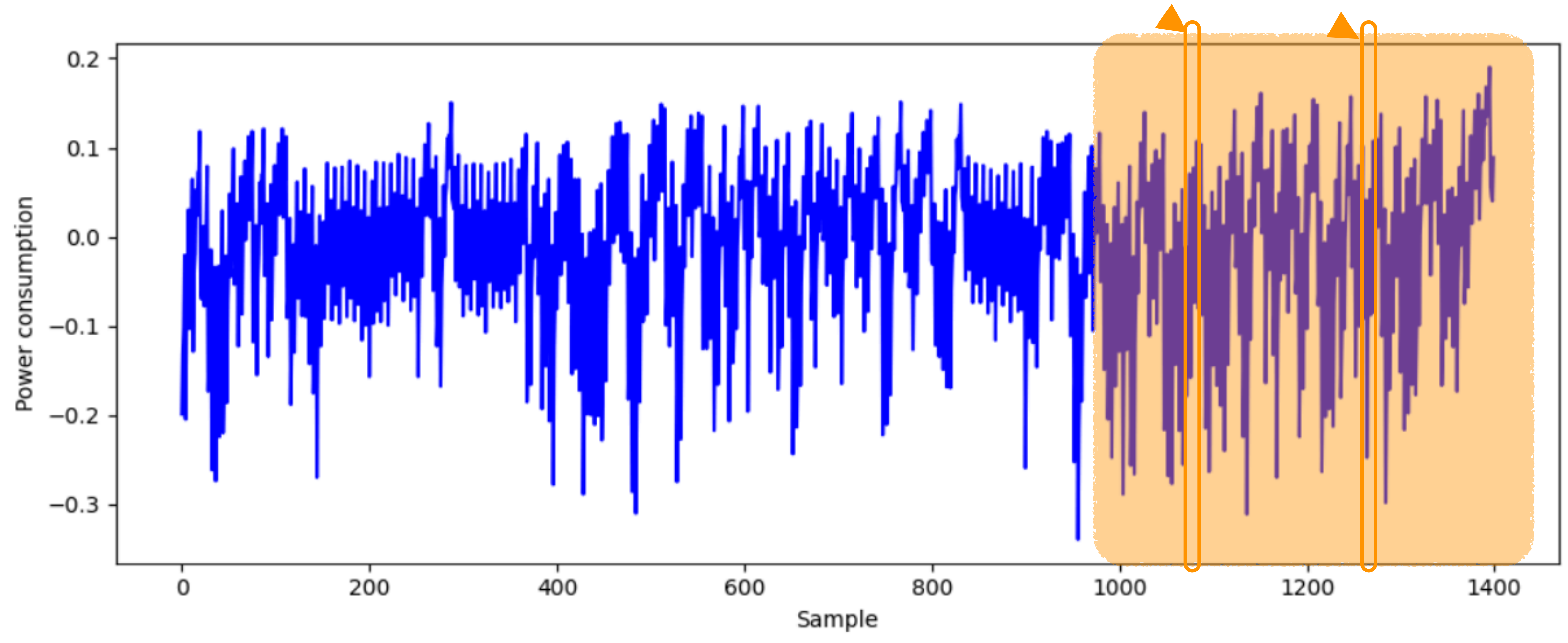


Optimizations:

- Attack point at linear layer output

Second-Order CPA

Combine two points on the trace
(by normalized product)



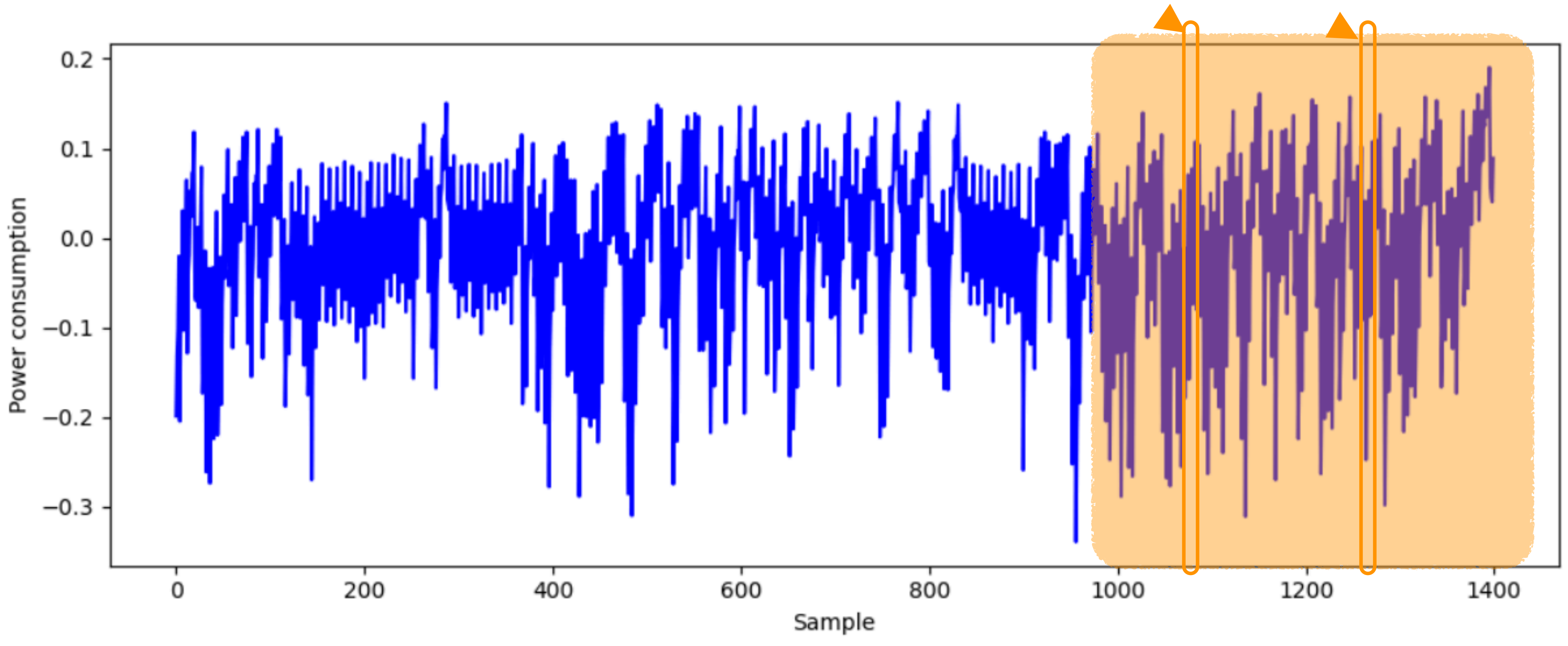
Optimizations:

- Attack point at linear layer output

  → focus on the right part of the trace

Second-Order CPA

Combine two points on the trace
(by normalized product)



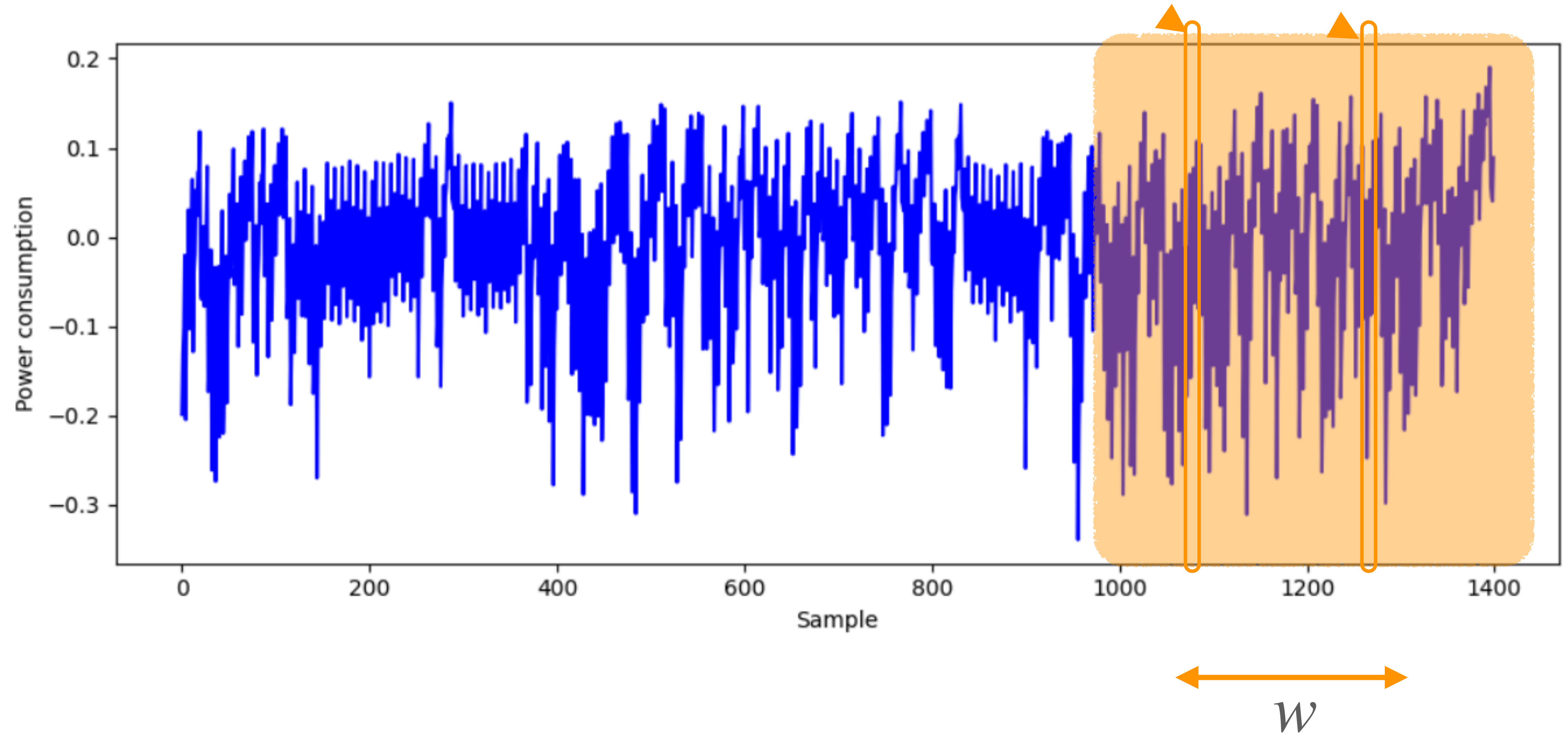
Optimizations:

- Attack point at linear layer output
  → focus on the right part of the trace

- Two shares occur in a time span

Second-Order CPA

Combine two points on the trace
(by normalized product)



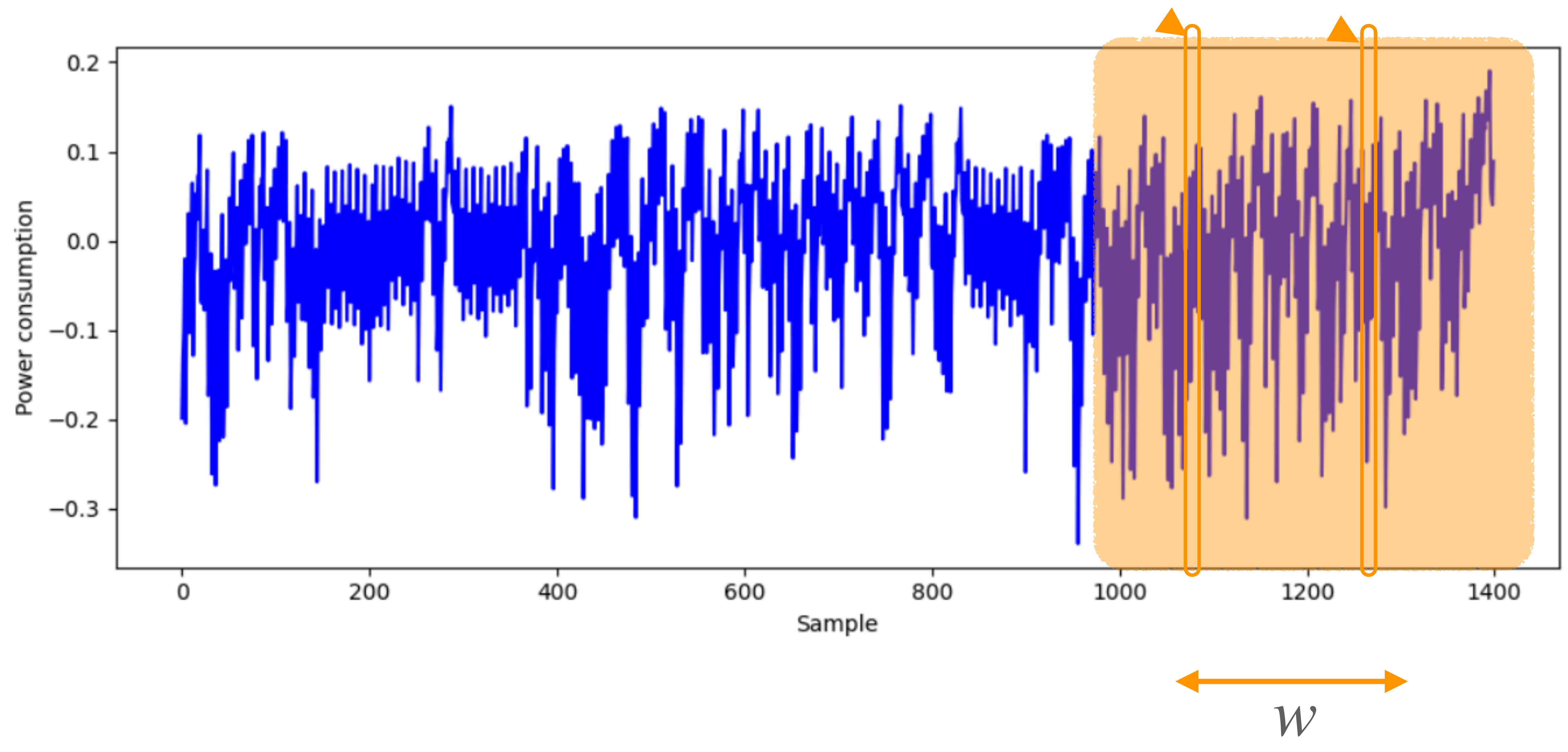
Optimizations:

- Attack point at linear layer output
  → focus on the right part of the trace

- Two shares occur in a time span

Second-Order CPA

Combine two points on the trace
(by normalized product)



Optimizations:

- Attack point at linear layer output

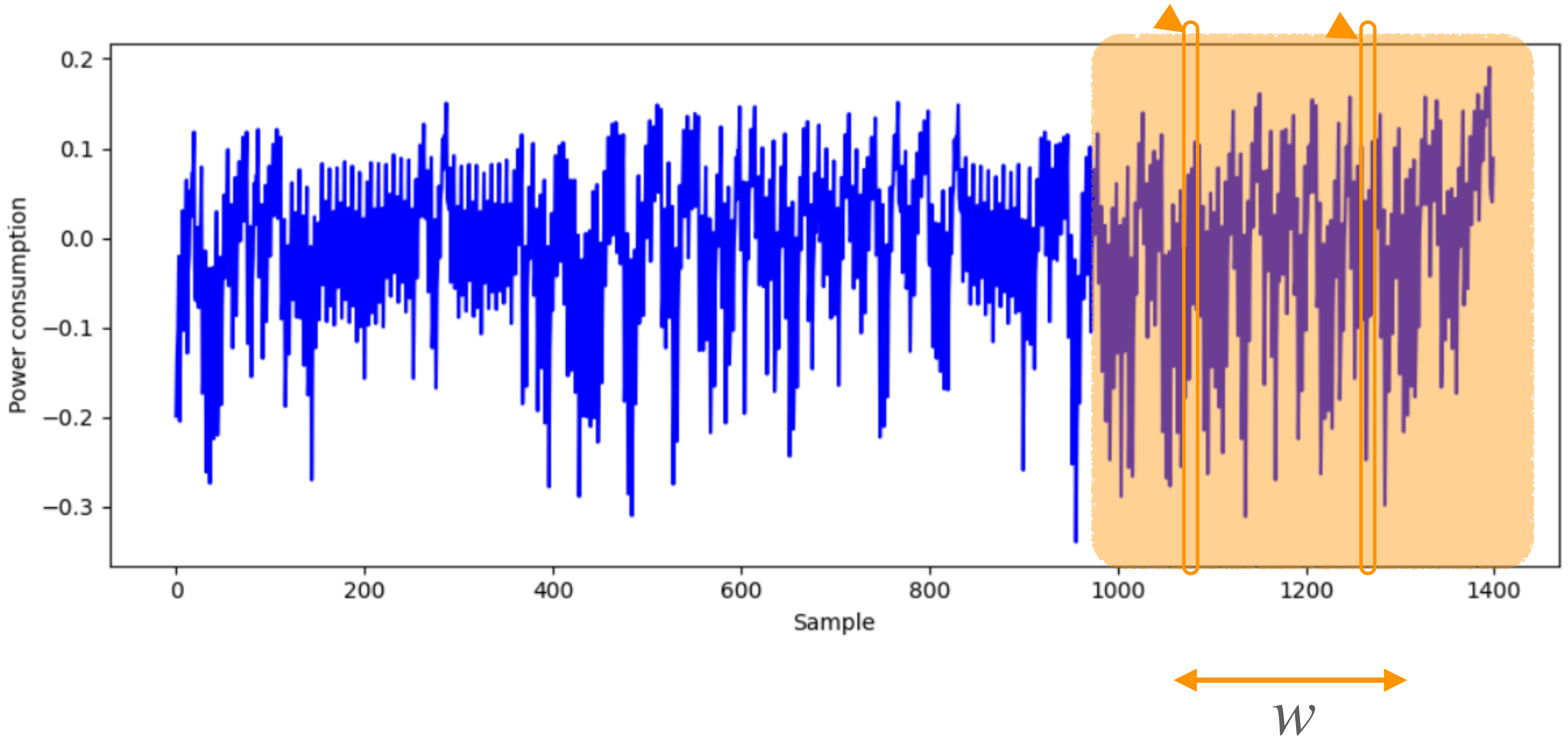
  → focus on the right part of the trace

- Two shares occur in a time span

  → parameter window w

Second-Order CPA

Combine two points on the trace
(by normalized product)



Optimizations:

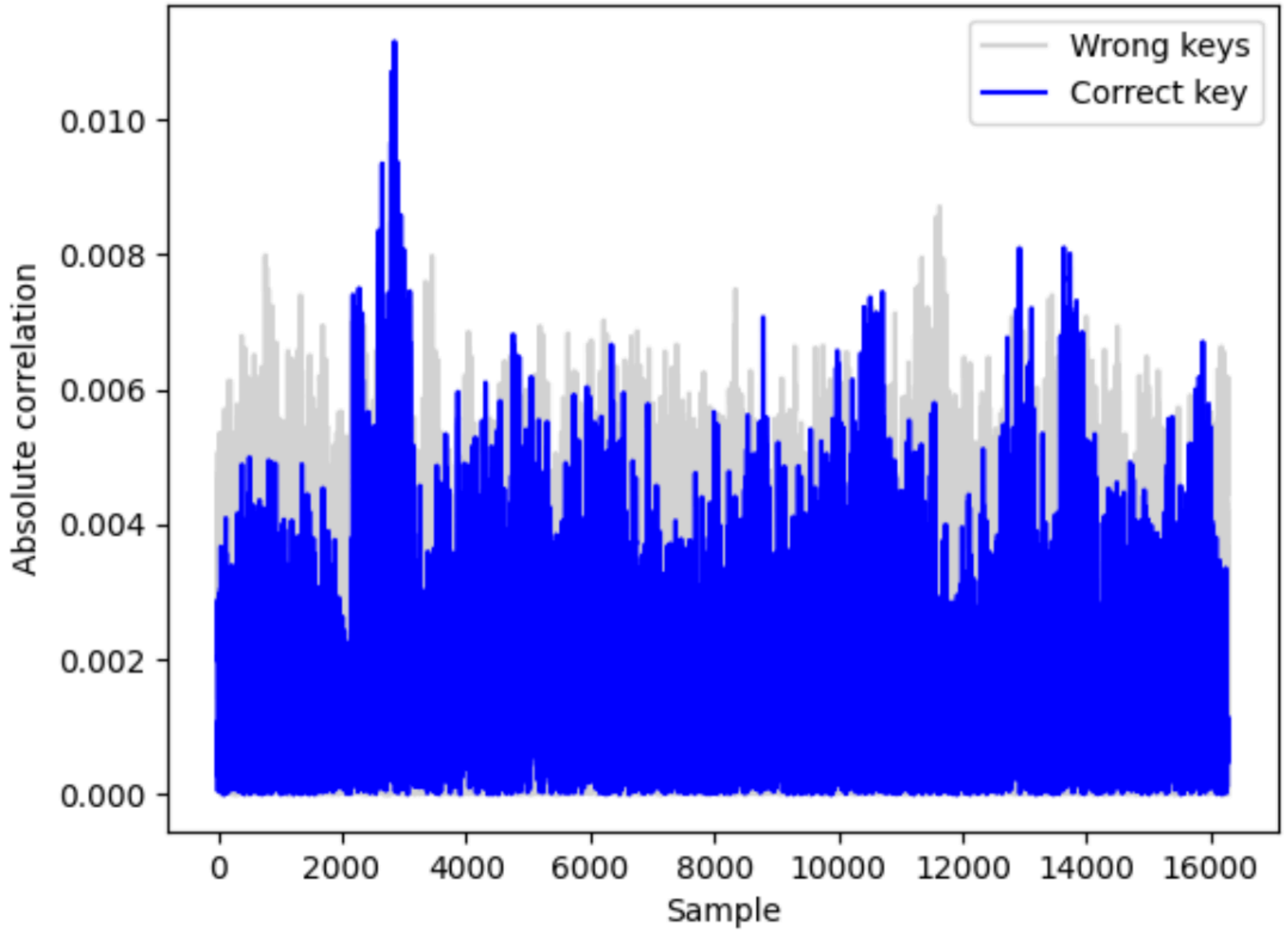
- Attack point at linear layer output
  → focus on the right part of the trace
- Two shares occur in a time span
  → parameter window w

Our attack considers the last 350 samples
$w = 50$

Results

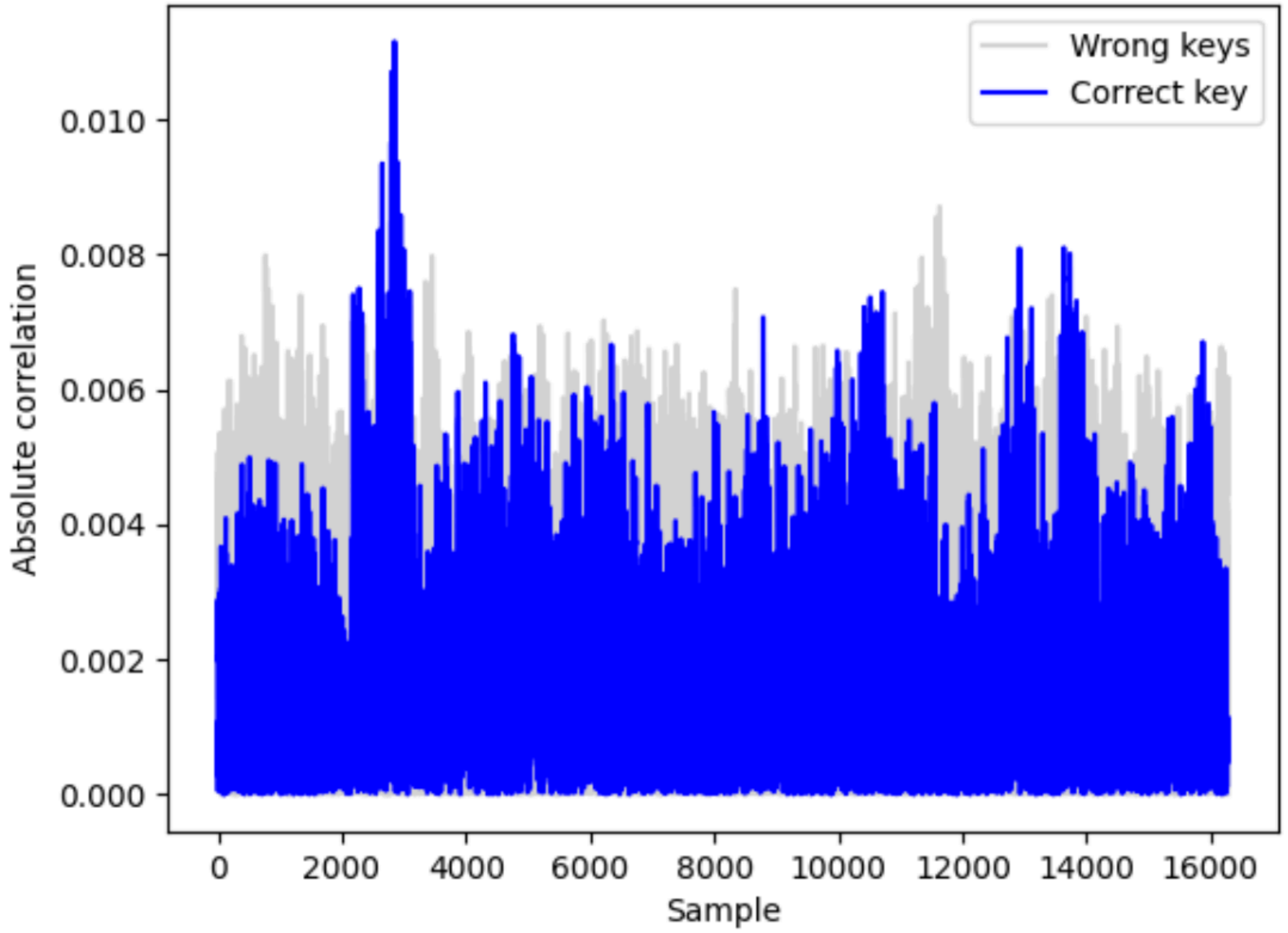
Results

Correlation traces

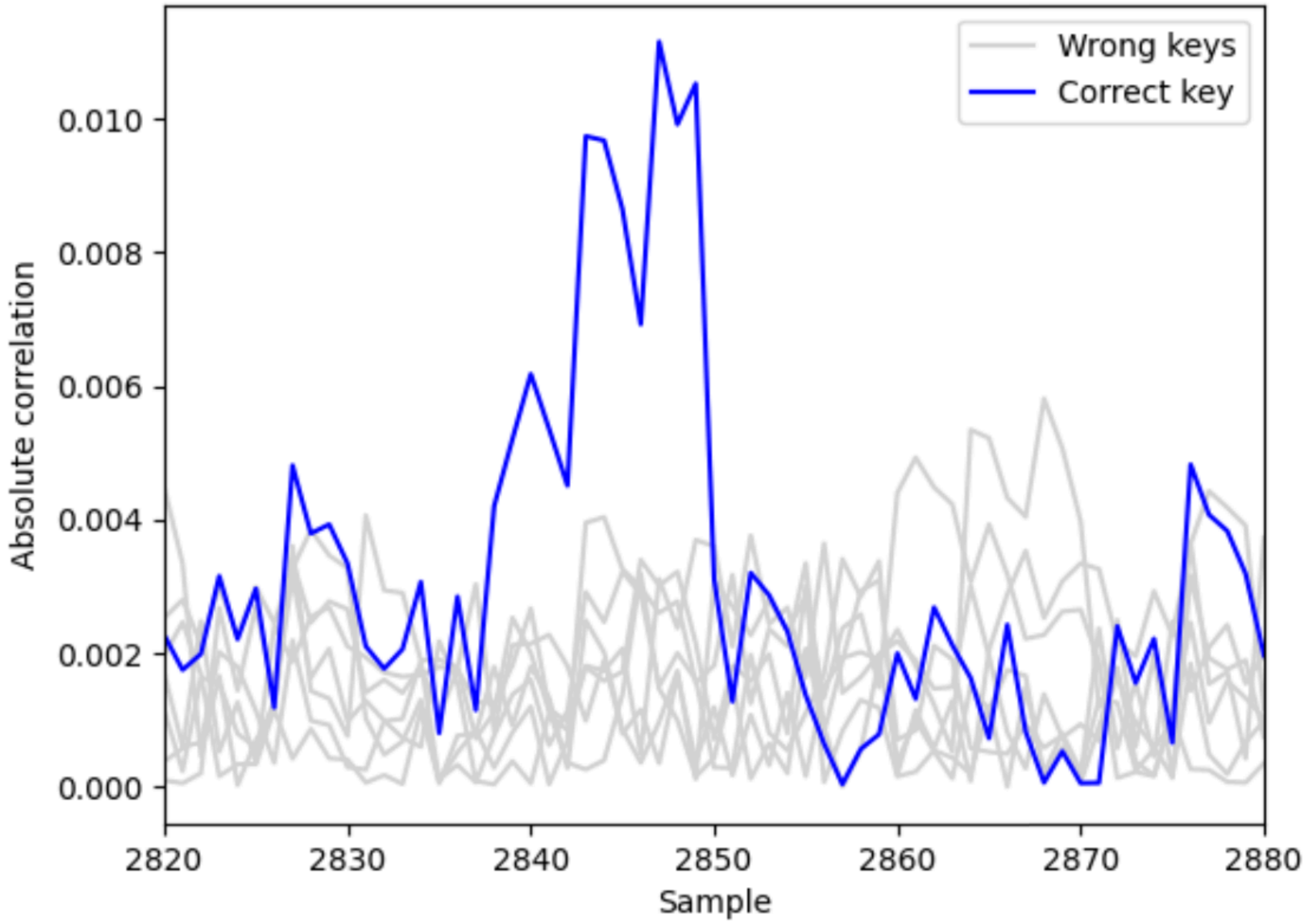


Results

Correlation traces



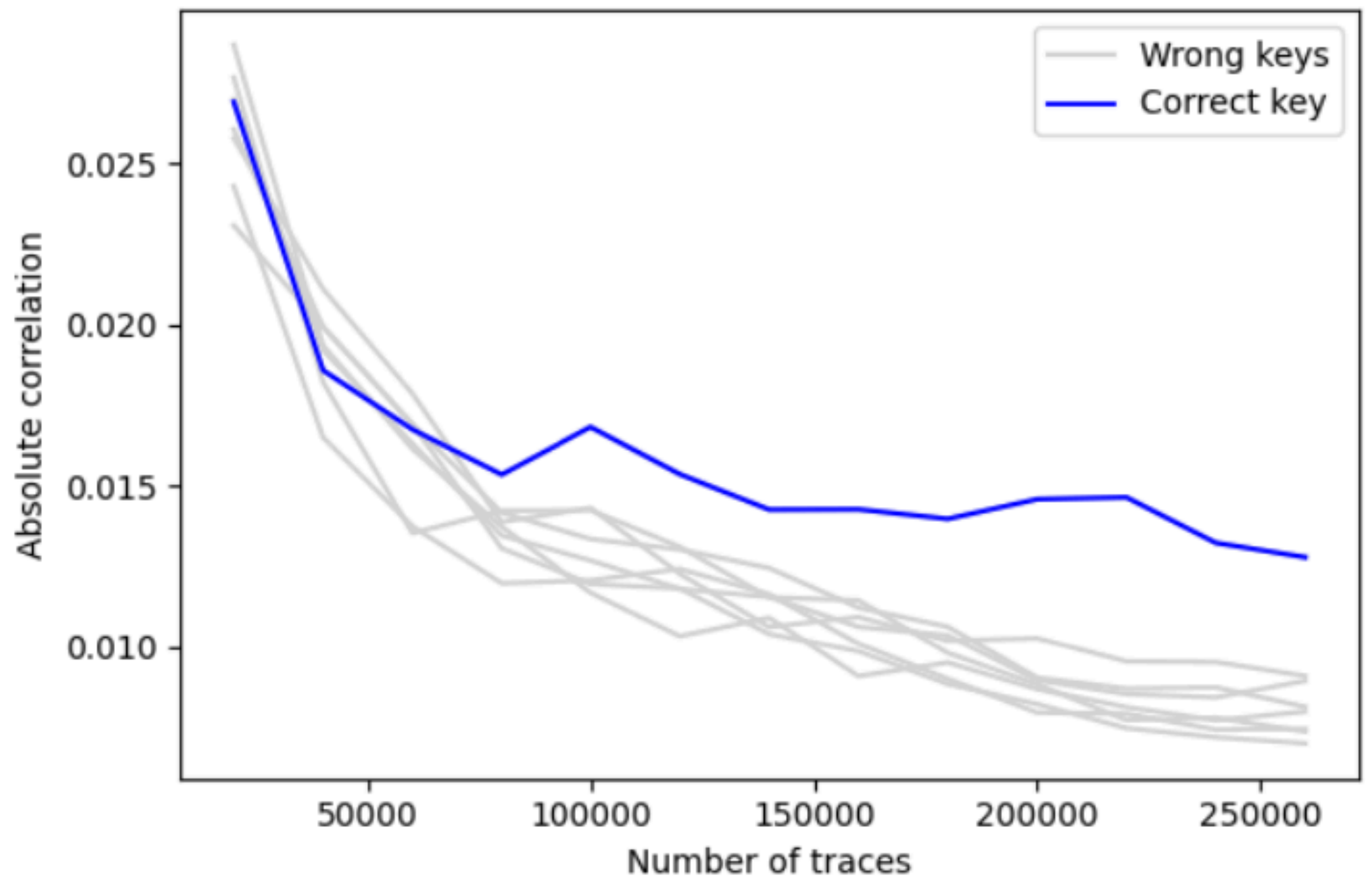
Correlation peak



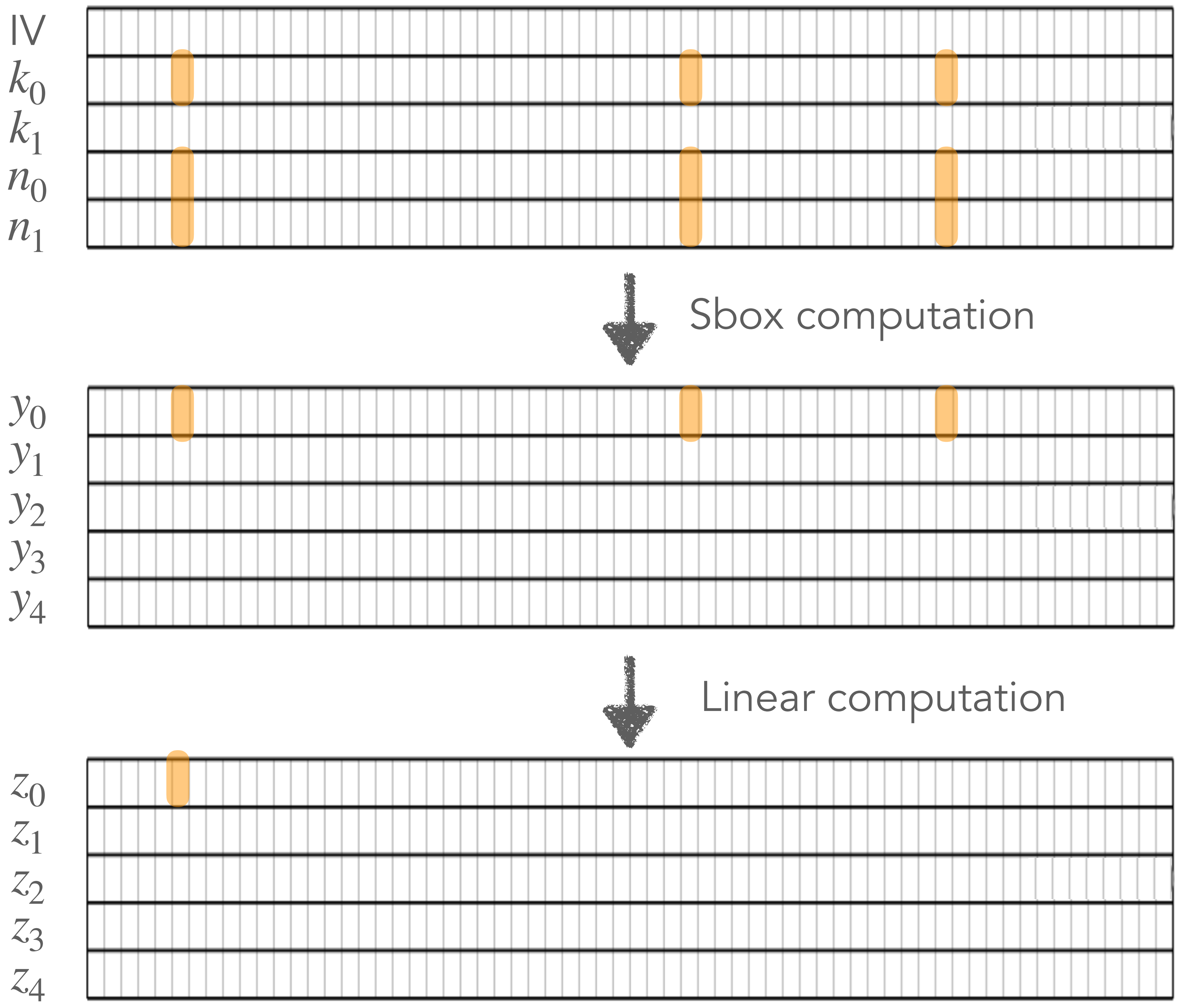
Results

Results

Correlation with increasing number of traces

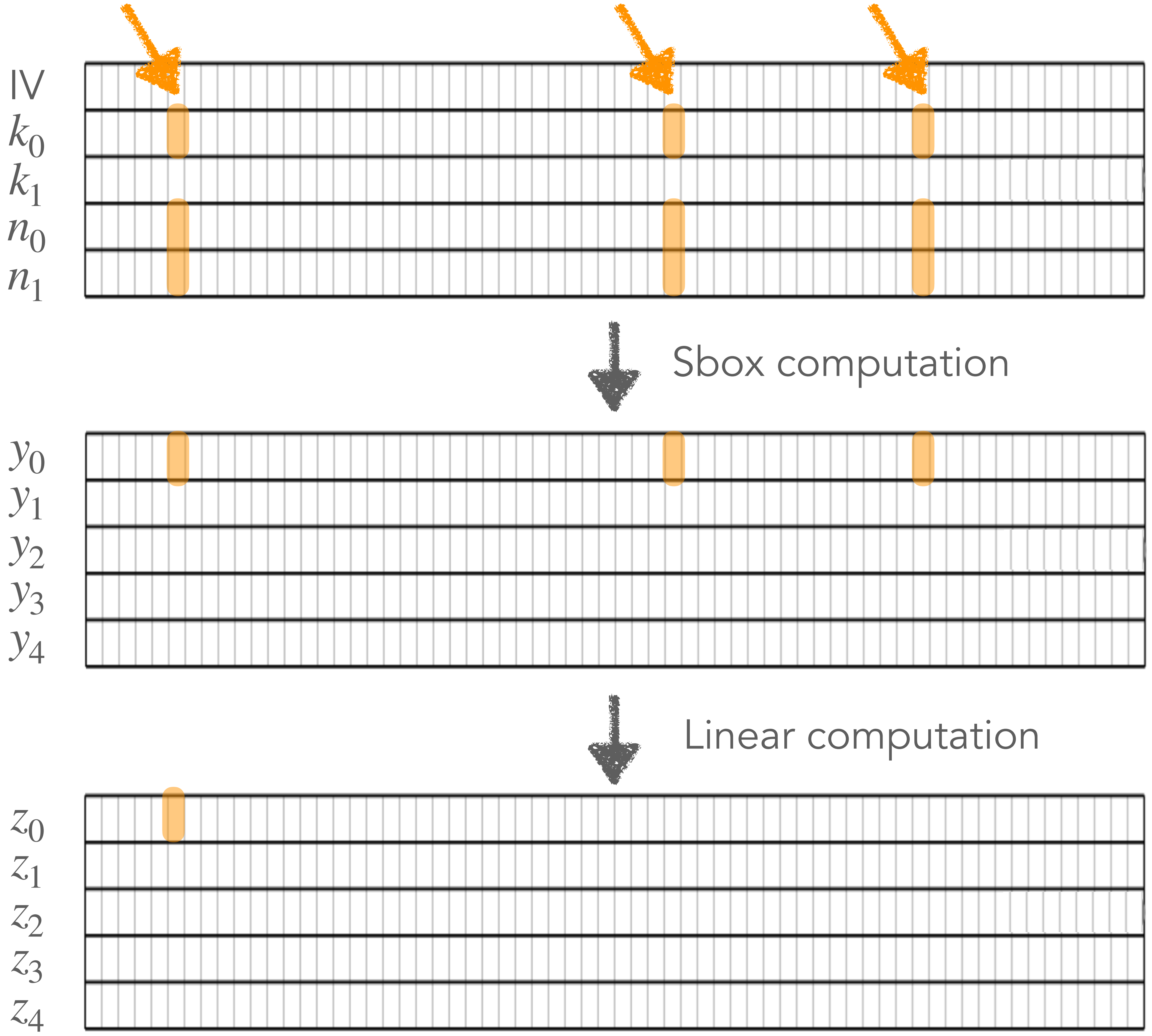


CPA runs for full-key recovery



CPA runs for full-key recovery

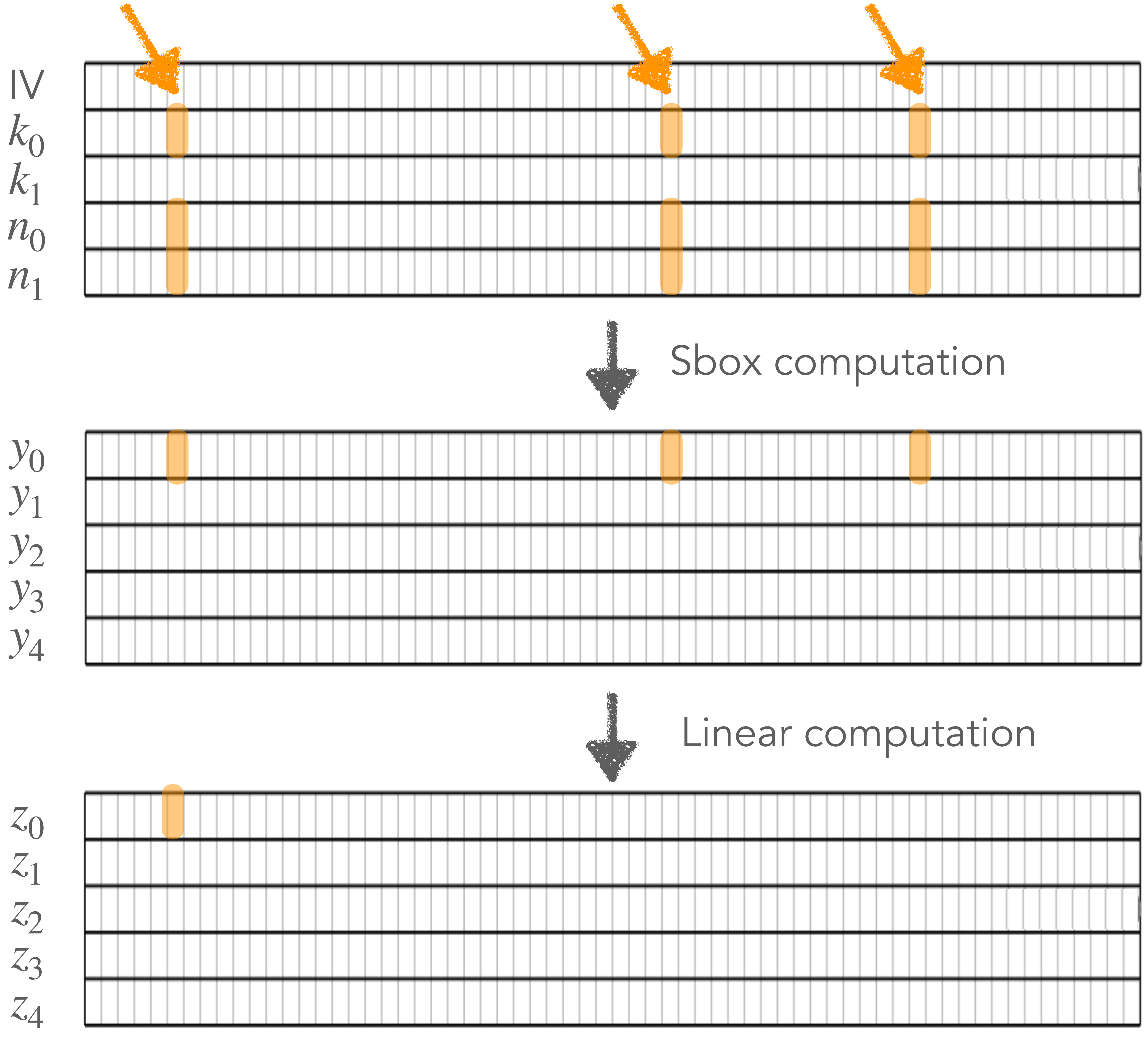
Each CPA run recovers 3 key bits



CPA runs for full-key recovery

Each CPA run recovers 3 key bits

How many CPA runs to recover 128 key bits?

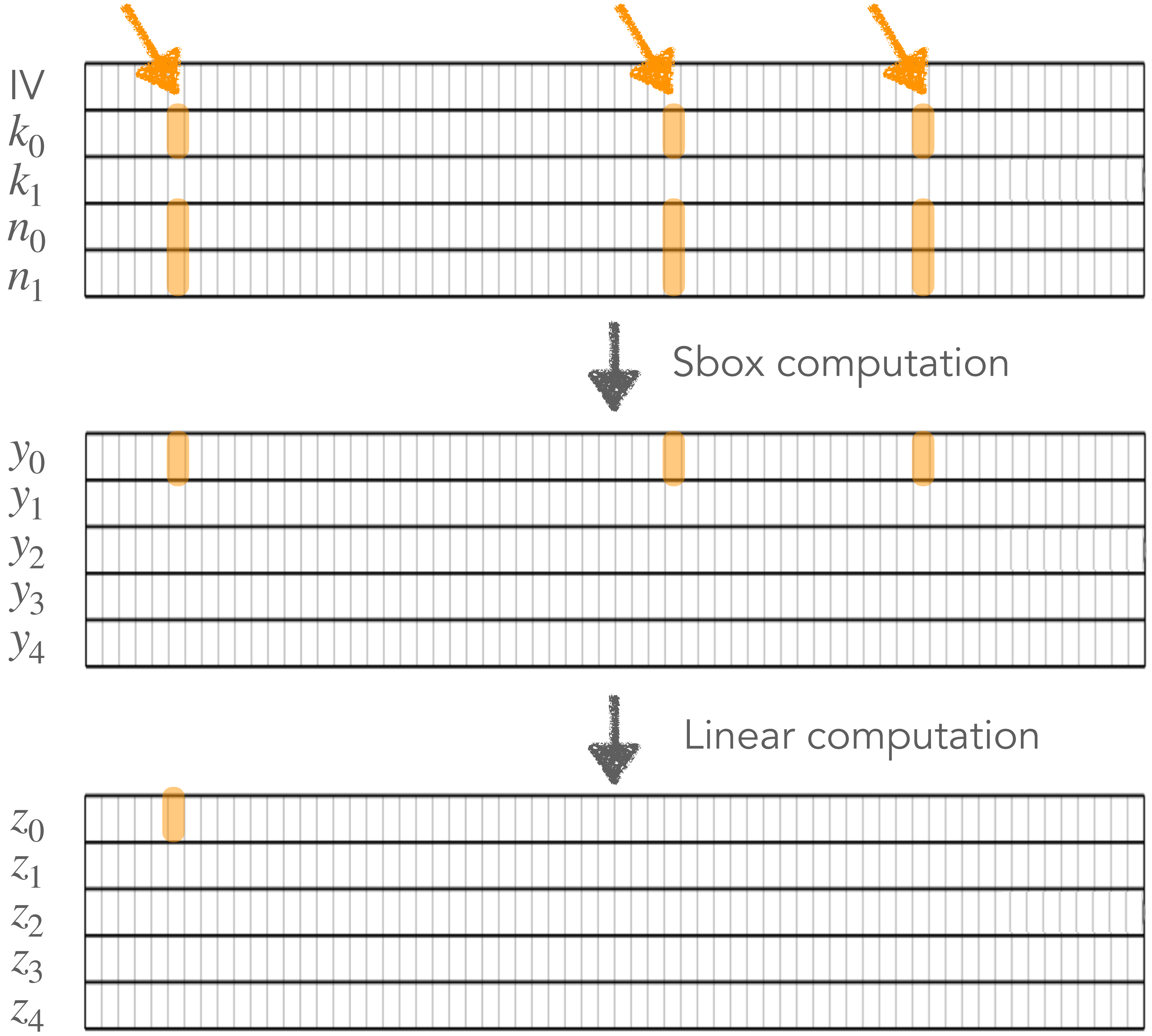


CPA runs for full-key recovery

Each CPA run recovers 3 key bits

How many CPA runs to recover 128 key bits?

Weissbart and Picek, 2023 — 63 CPA runs



CPA runs for full-key recovery

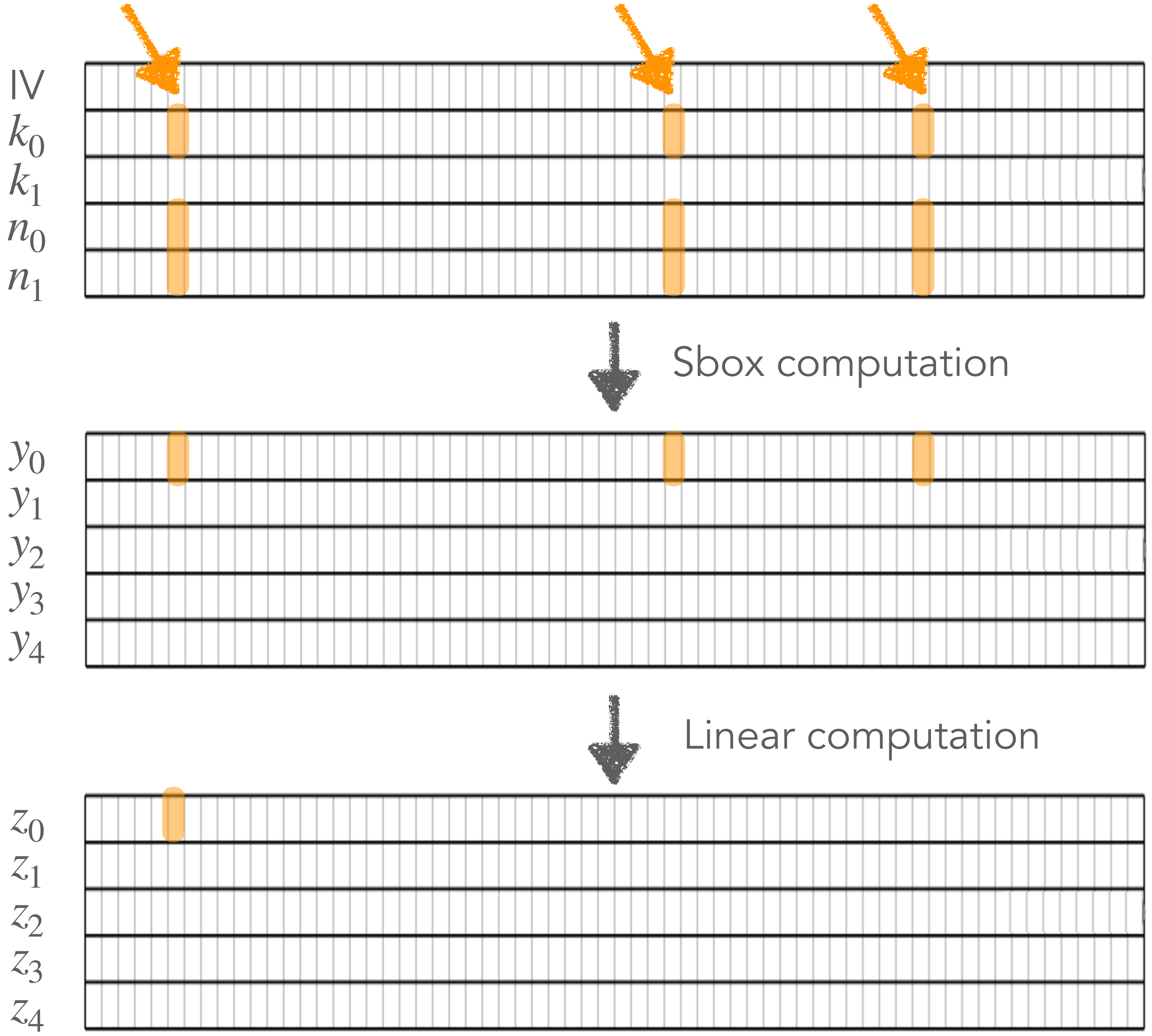
Each CPA run recovers 3 key bits

How many CPA runs to recover 128 key bits?

Weissbart and Picek, 2023 — 63 CPA runs

This work:

- formalizes a *set cover problem*
- uses a SAT solver



CPA runs for full-key recovery

Each CPA run recovers 3 key bits

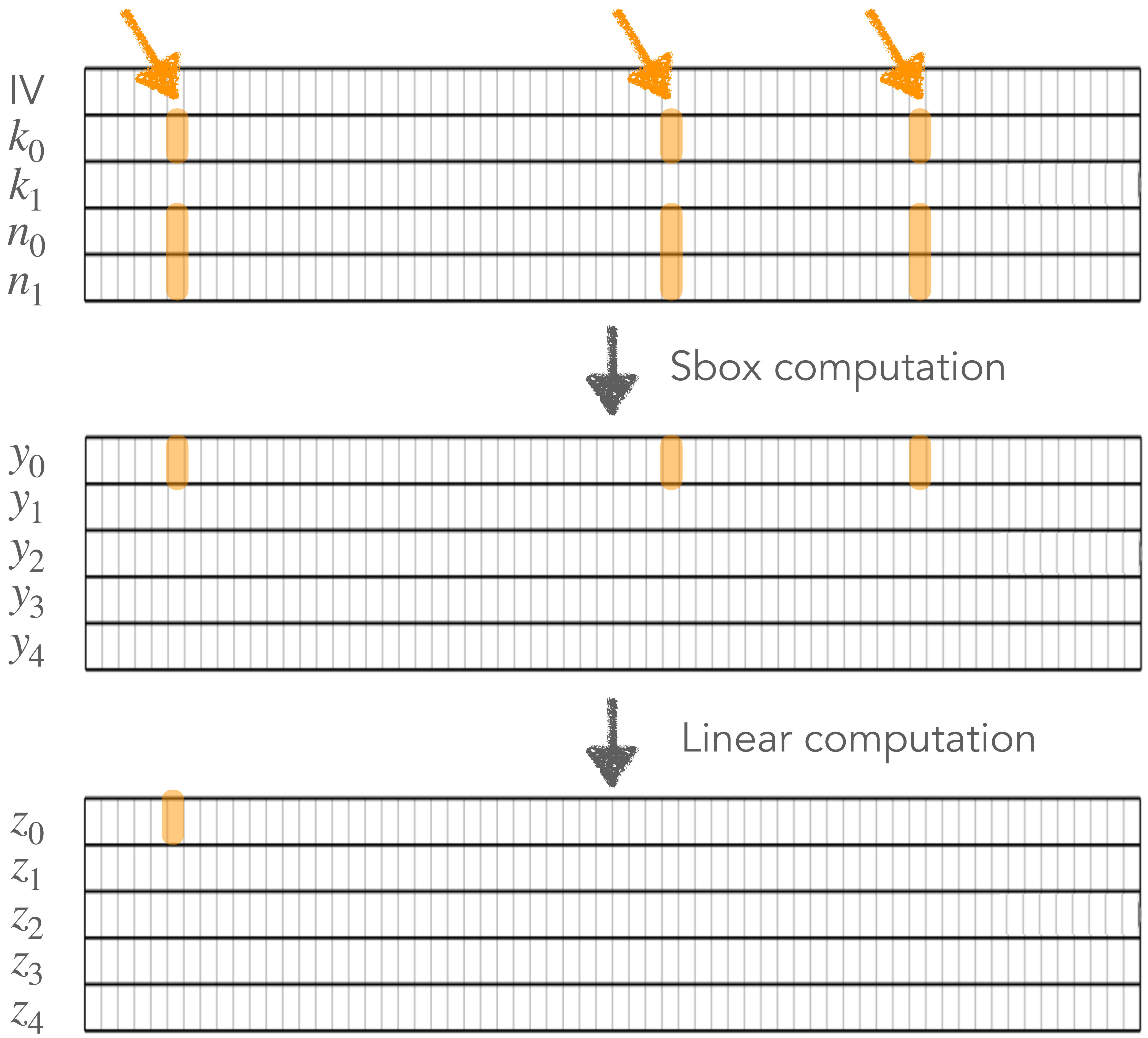
How many CPA runs to recover 128 key bits?

Weissbart and Picek, 2023 — 63 CPA runs

This work:

- formalizes a *set cover problem*
- uses a SAT solver

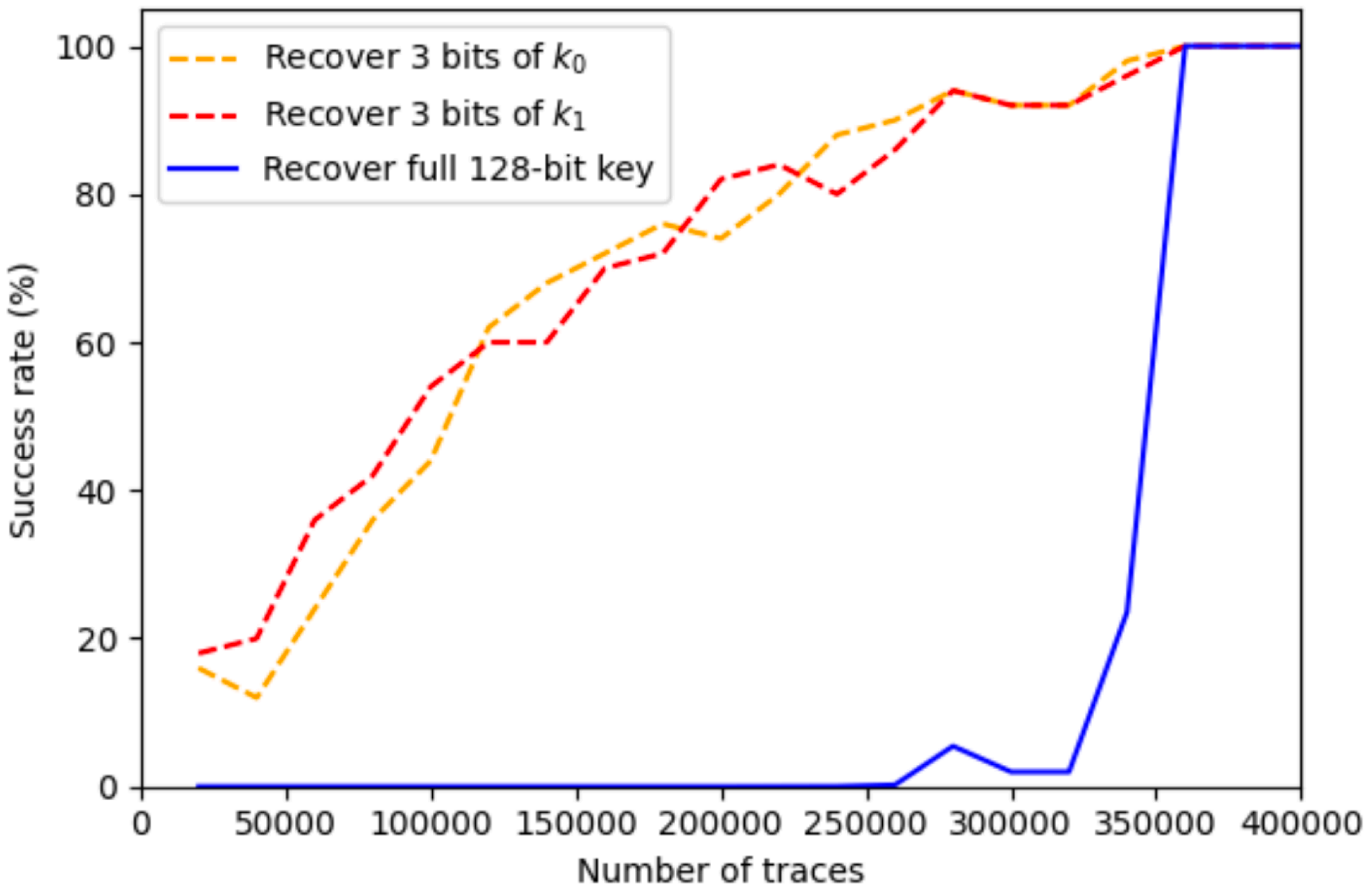
→ 47 CPA runs (optimal)



Full-key recovery

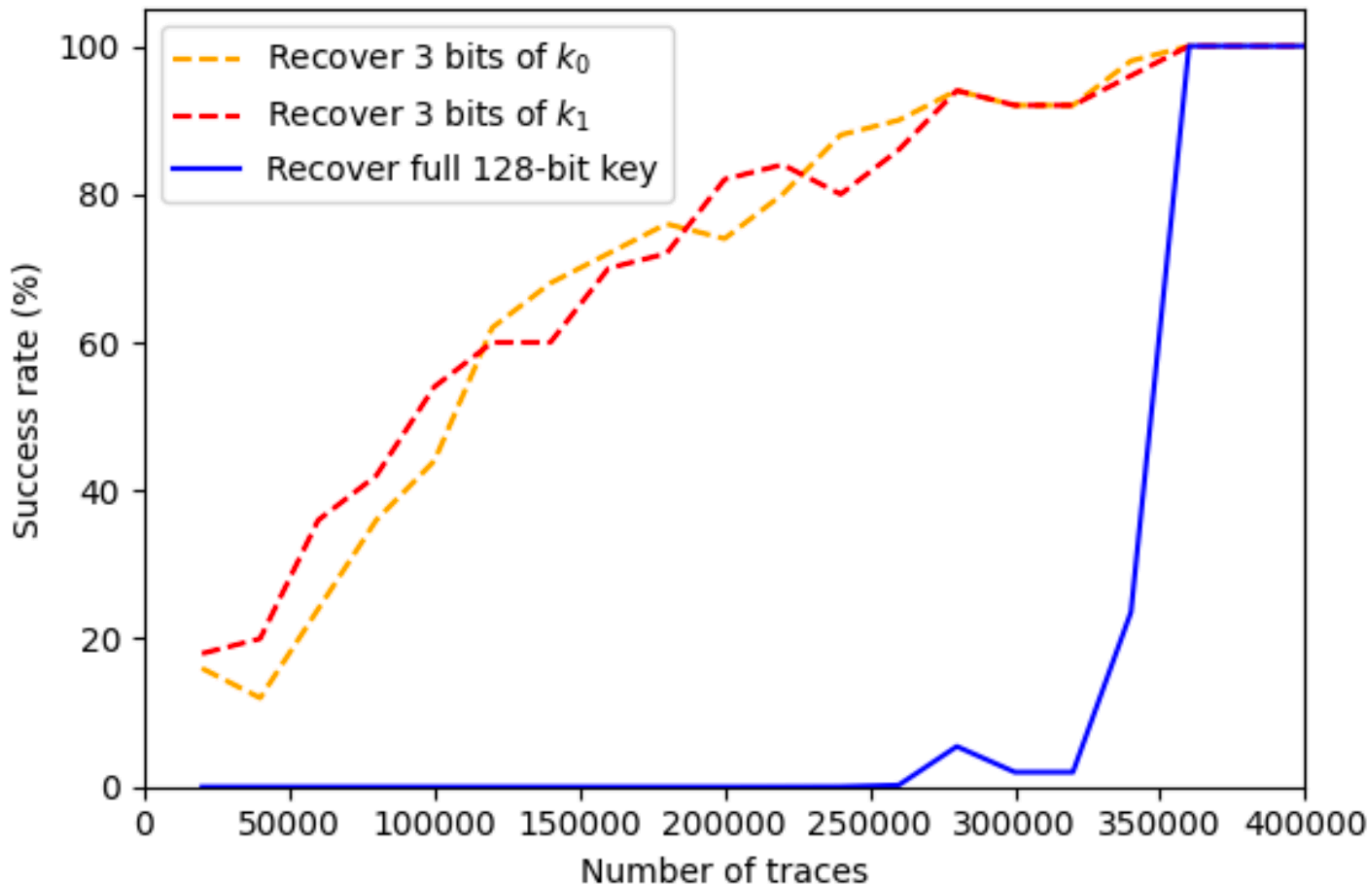
Full-key recovery

Success rates



Full-key recovery

Success rates



360K traces ensure 100% success rates

Recover full key in 4.7 hours

Summary

CPA attack on Ascon

CPA attack on Ascon

- Quadratic boolean Sbox function
  - choose the selection function carefully

CPA attack on Ascon

- Quadratic boolean Sbox function
  - choose the selection function carefully
- Optimal number of CPA runs for full-key recovery: 47

CPA attack on Ascon

- ✦ Quadratic boolean Sbox function
  - choose the selection function carefully
- ✦ Optimal number of CPA runs for full-key recovery: 47
- ✦ First results on practical second-order CPA

Practical Second-Order CPA Attack on Ascon with Proper Selection Function

Viet-Sang Nguyen

joint work with Vincent Grosso and Pierre-Louis Cayrel

CASCADE Conference

Saint-Etienne, 2 April, 2025

UMR ▪ CNRS ▪ 5516 ▪ Saint-Étienne

PROPHY ANR-22-CE39-0008-01