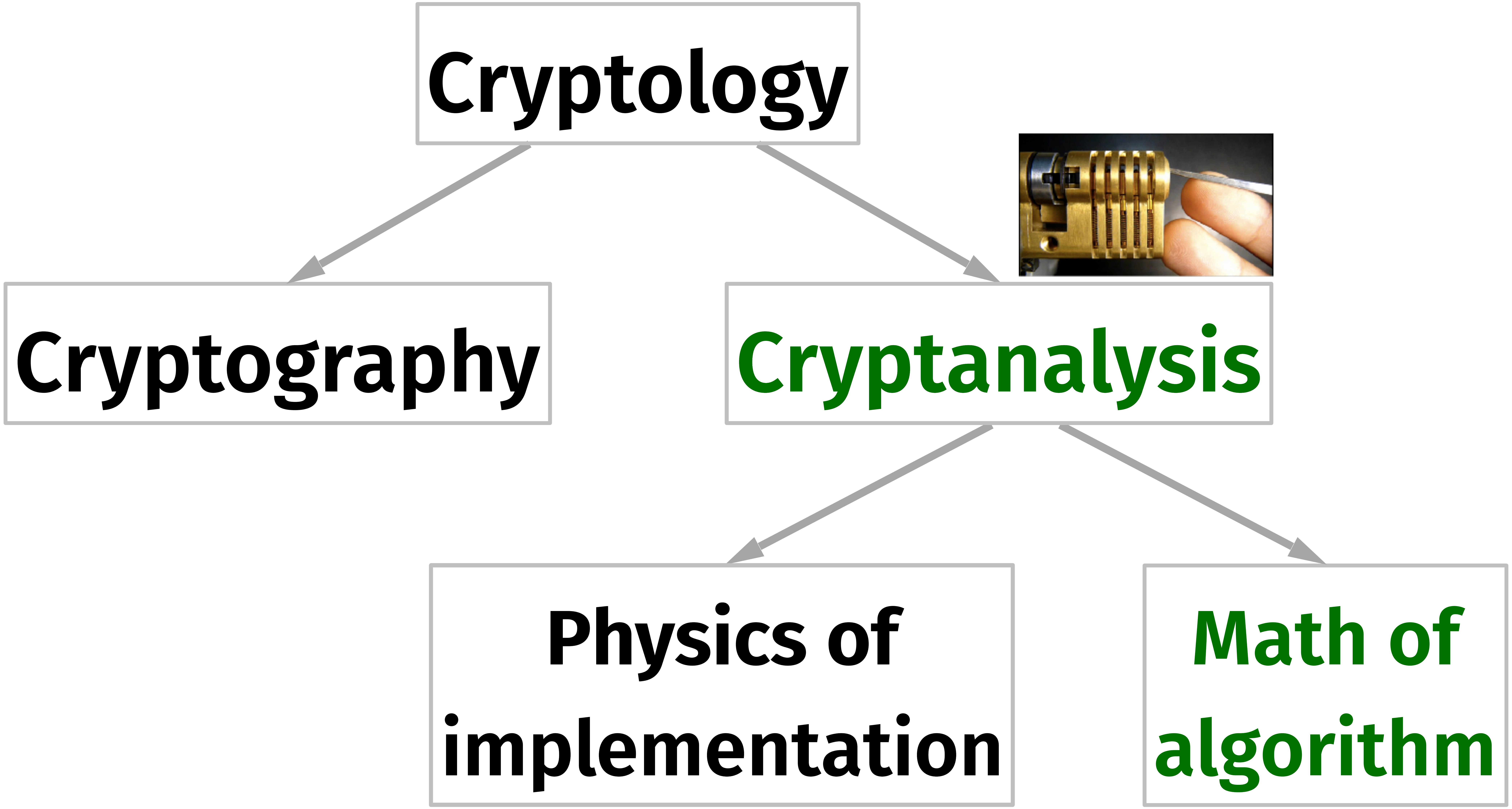
Lecture 20: Protecting data while computing

- Lab 11 will be posted soon, due Wednesday 5/1
- Online course evaluation is live
- Final exam
    - Scope: all topics covered in lectures, recitations, and labs (except law/policy)
    - Format: similar to the midterm
    - Sample exam: will post on Piazza soon
    - Review session: respond to Piazza poll by Saturday 4/27 with your availability



Cryptology

Cryptography

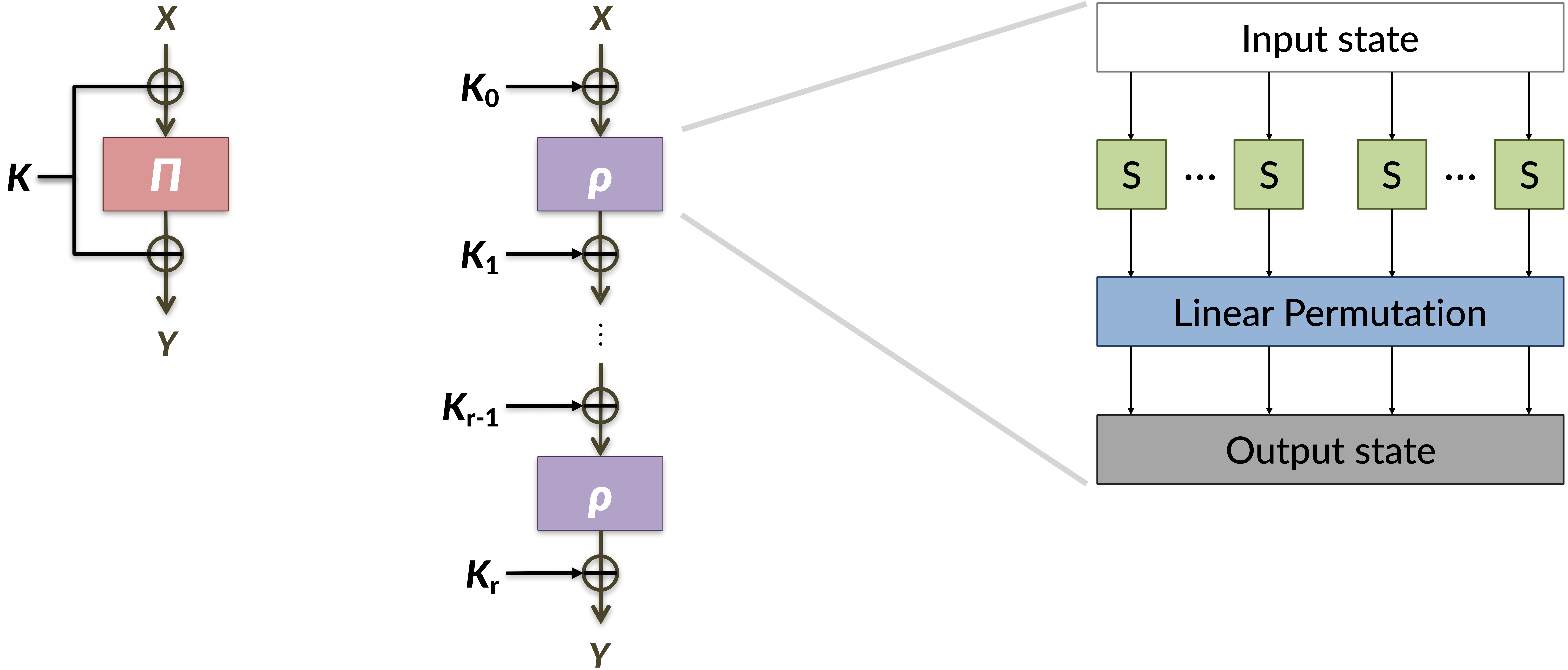
Cryptanalysis

Physics of implementation

Math of algorithm

Refresher: block cipher design

Key alternation, over several rounds, each w/ substitution & permutation



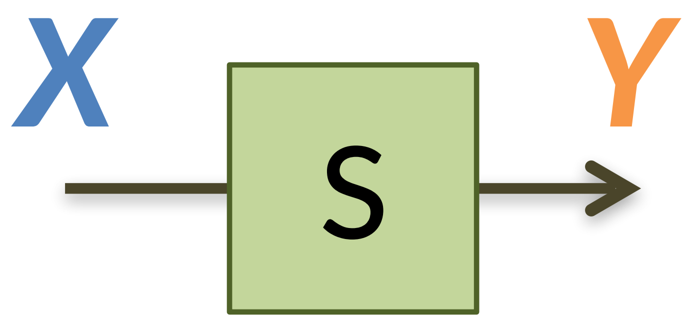
Question: what if S is 'too linear'?

Confusion

- Uncertain K → can't correlate X, Y
- Ideal: Prob[correlation] so small that attacker prefers a brute force attack

Linear cryptanalysis

Exploits the fact that S may behave 'similarly' to a linear function

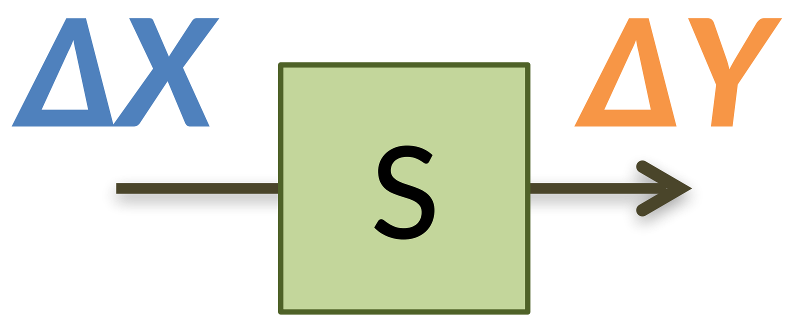


Diffusion

- 1 bit ΔX → huge ΔY
- Ideal: each output bit depends on all input bits (2 rounds in AES)

Differential cryptanalysis *(our focus)*

Exploits the fact that *differences* in inputs + outputs may be correlated



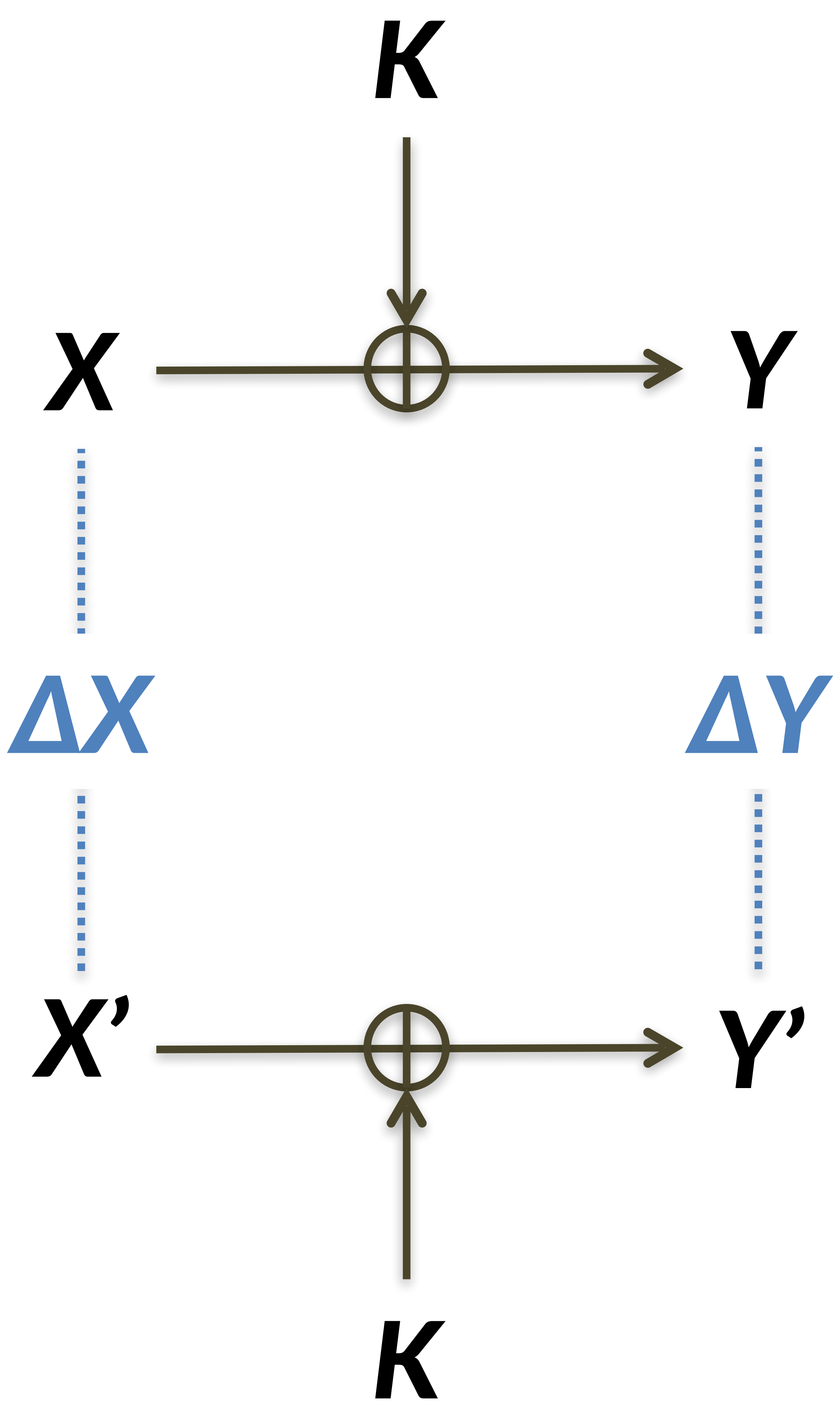
Our first differential cryptanalysis

Consider a one-time pad

- Claude Shannon (and others) showed that it is ‘perfectly hiding’
- Concretely: if you don’t know K, then it is impossible to correlate X and Y

What about a two-time pad?

- Suppose attacker has two X/Y pairs
- Confusion disappears!
- Concretely: *even without knowing* K, we can say for sure that ΔX = ΔY
  - ΔX = X ⊕ X’
  - ΔY = Y ⊕ Y’



The TOY cipher

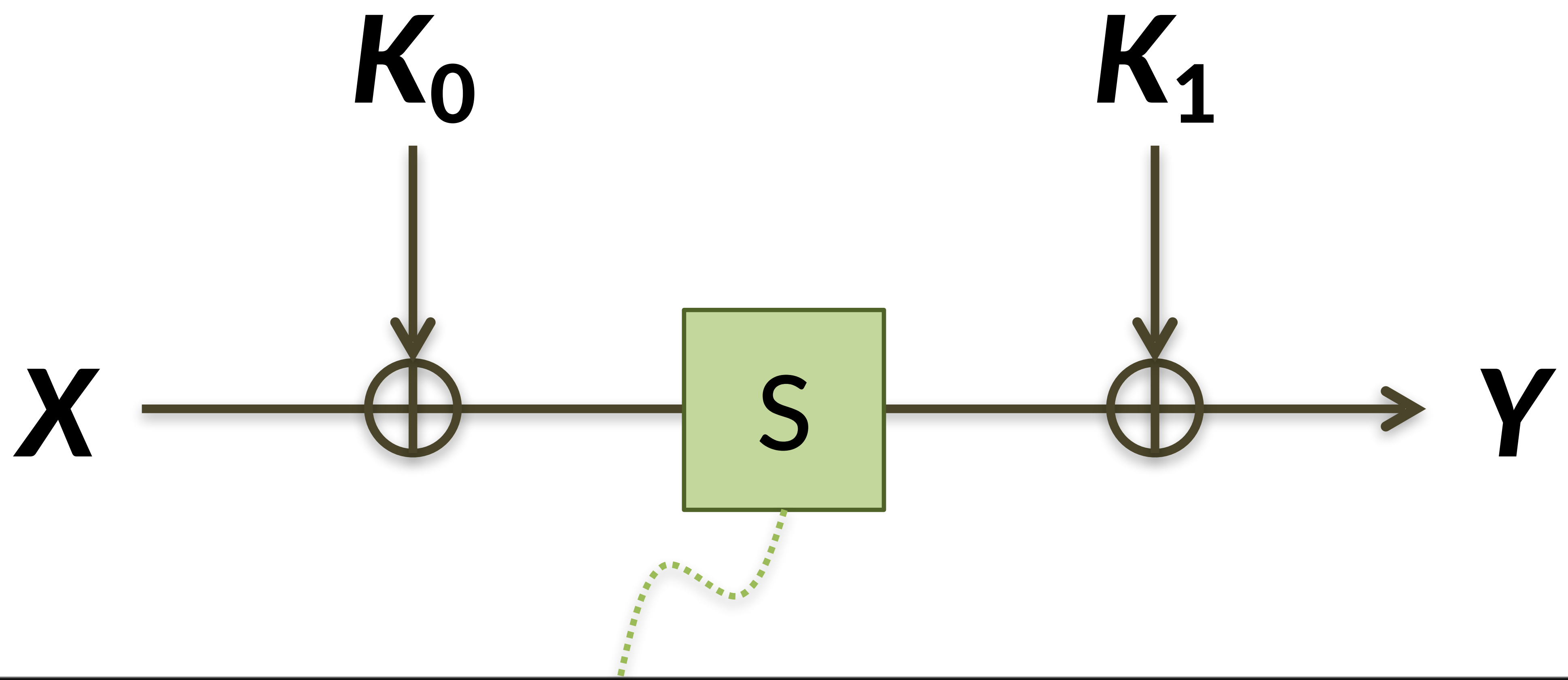
TOY cipher design = an S-box sandwiched by one-time pads

Concrete sizes

- 4-bit input X and output Y
- 8-bit total key
- S-box has $2^4 = 16$ total inputs/outputs

Hope: cannot break TOY faster than a brute-force search of $2^8 = 256$ keys

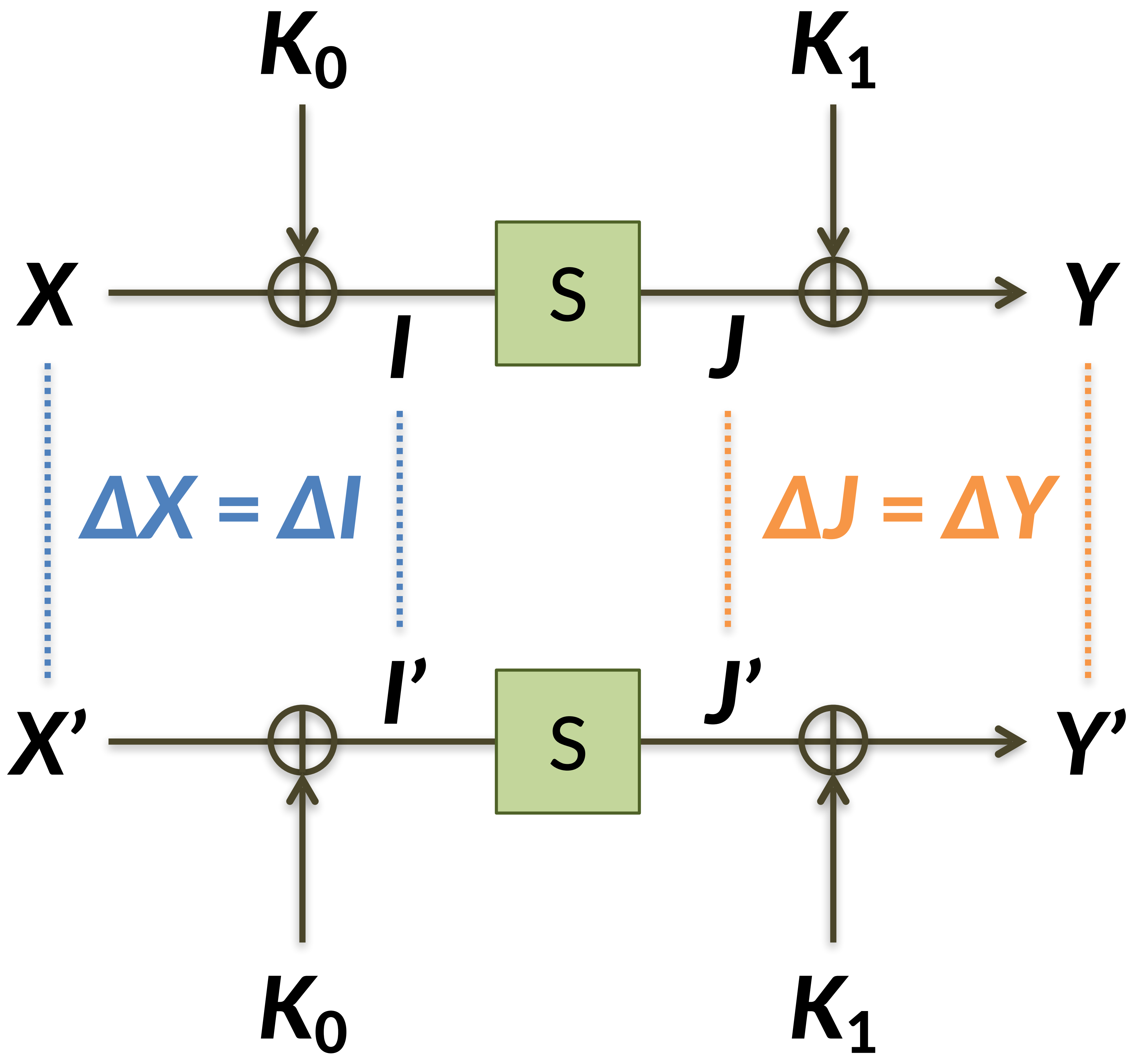
Sadly, this hope is false



x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S[x]$	6	4	c	5	0	7	2	e	1	f	3	d	8	a	9	b

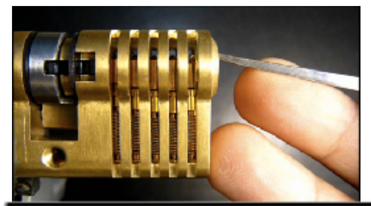
Differential cryptanalysis of TOY

- Consider two input/output pairs
- What do we know about differences?
- ΔX = ΔI and ΔJ = ΔY, indep of key
- This doesn't directly relate ΔX and ΔY… but, at least we learned that it suffices to connect ΔI with ΔJ
- Remember: ΔJ = J ⊕ J' = S[I] ⊕ S[I']
- New plan: try all pairs I, I' that differ by ΔI, see which yields a difference of ΔJ on the other side of the S-box

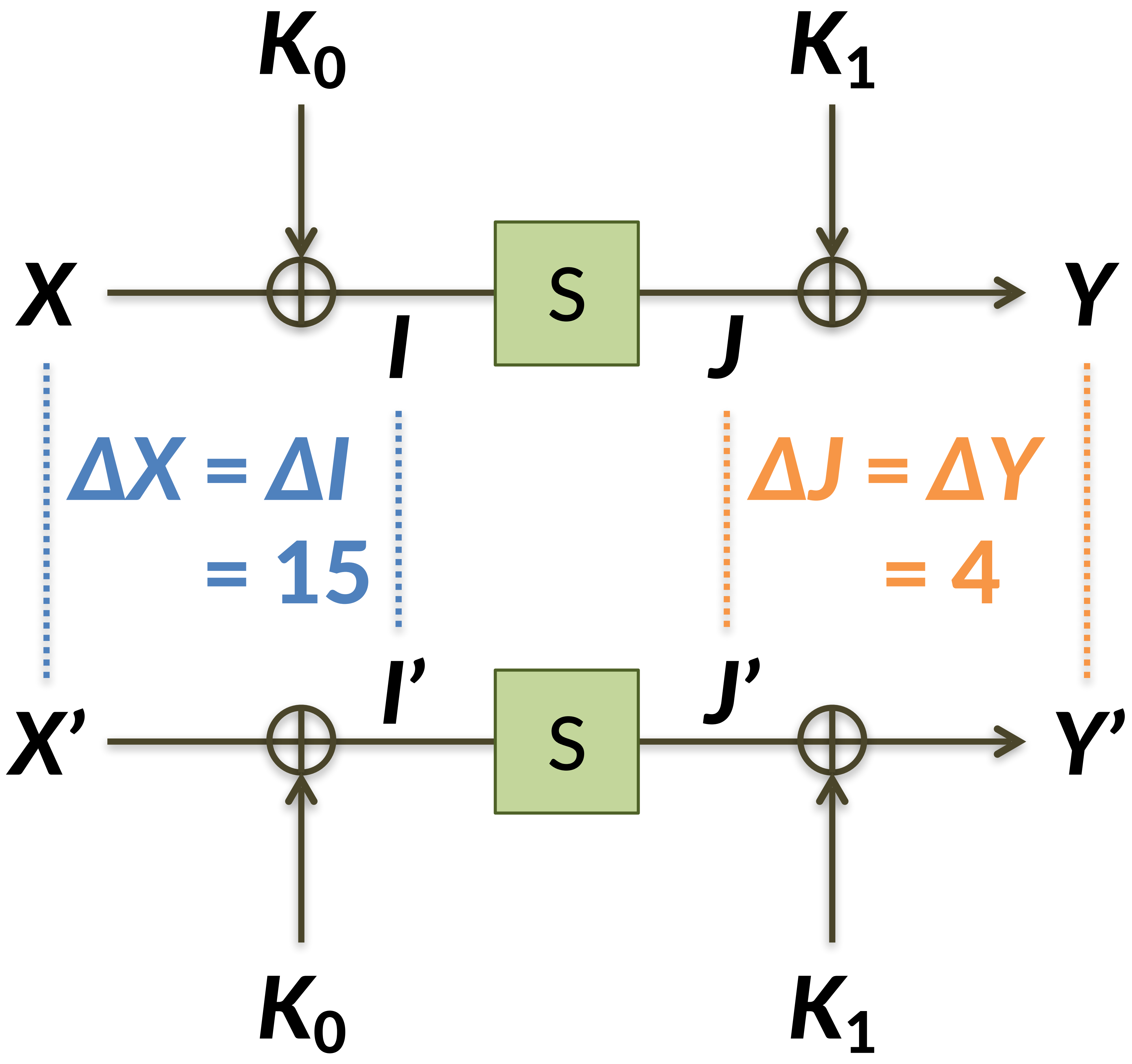


Concrete example

- Input $X = 0$ maps to output $Y = 11$ (i.e., 0xB)
- Input $X' = 15$ maps to output $Y' = 15$ (i.e., 0xF)



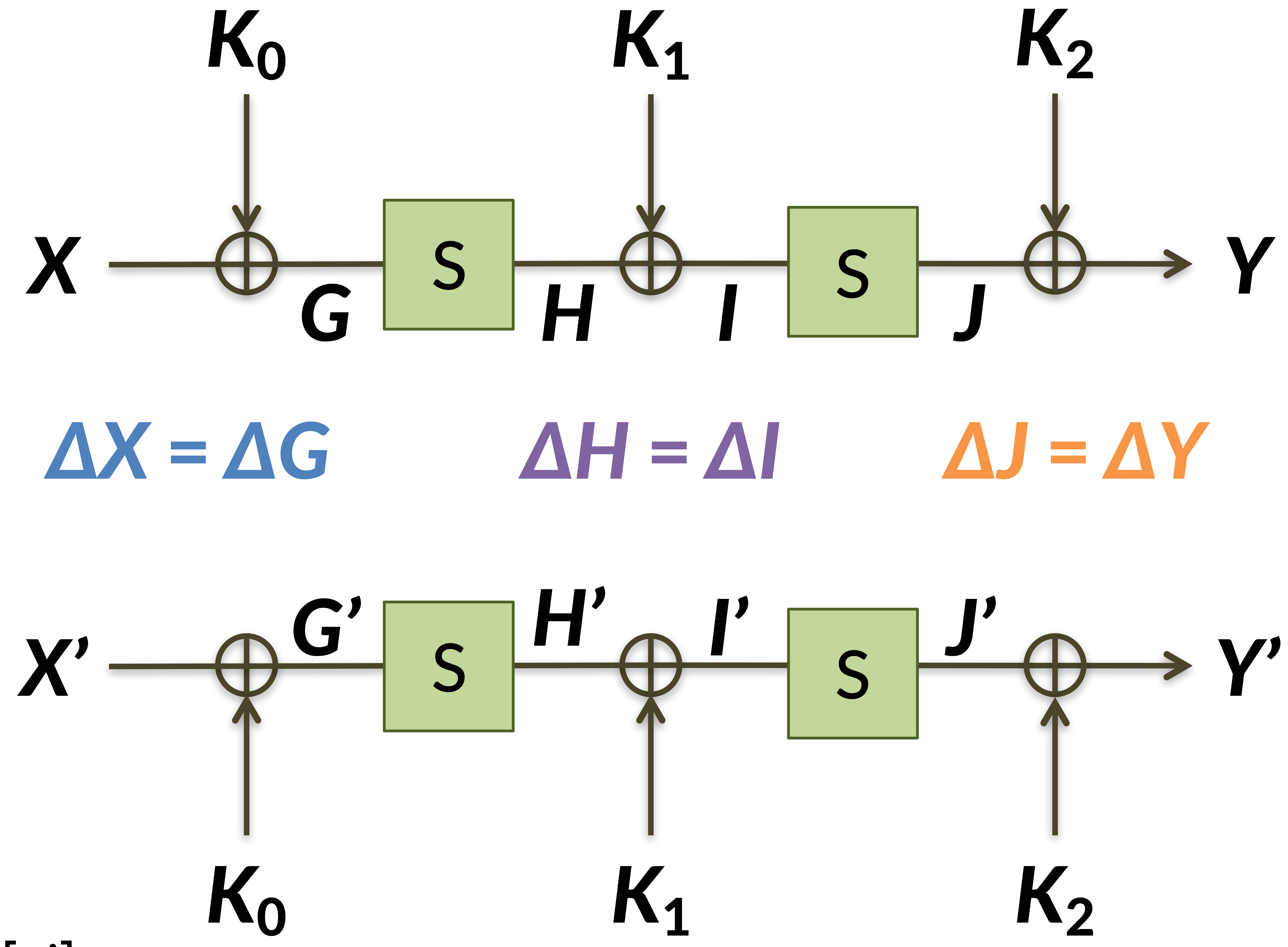
$K_0 = I$	I'	$S[I]$	$S[I']$	$S[I] \oplus S[I']$
0	f	6	b	d
1	e	4	9	d
2	d	c	a	6
3	c	5	8	d
4	b	0	d	d
5	a	7	3	4
6	9	2	f	d
7	8	e	1	f
8	7	1	e	f
9	6	f	2	d
a	5	3	7	4
b	4	d	0	d
c	3	8	5	d
d	2	a	c	6
e	1	9	4	d
f	0	b	6	d



Two possible keys: (5,C) and (A,8)

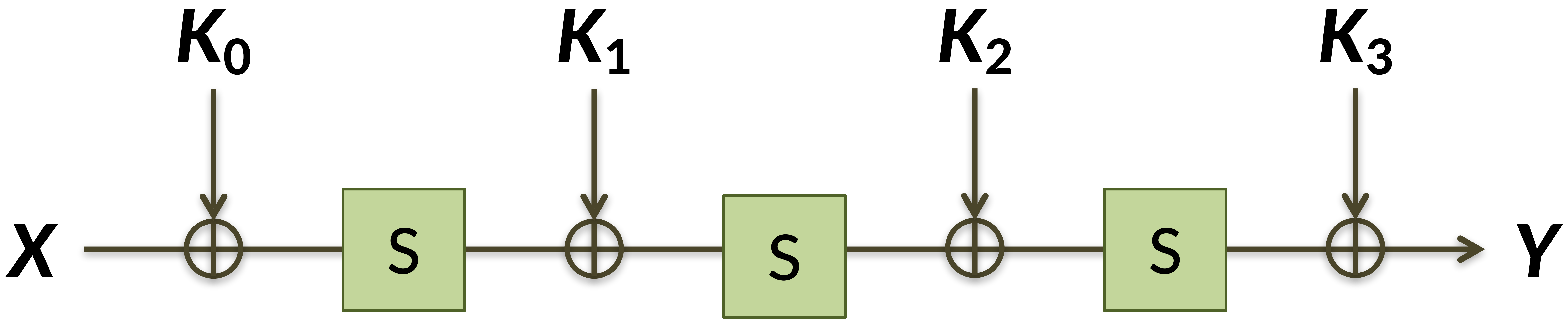
Differential cryptanalysis of 2TOY

- Main rule of cipher design: if the cipher breaks, simply add more rounds
- Now we don't know all differences
- But if we *did* know $\Delta H = \Delta I$ then we would be back to TOY's analysis
- Let's see if we can fake it!
  - Suppose ΔX = 0xF just as before
  - Then ΔI = 0xD with prob 10/16
  - Simply assume that's the case, and conduct the TOY cryptanalysis attack
  - Find values of K_2 consistent with $\Delta I = S^{-1}[Y] + S^{-1}[Y']$
- If Pr[guess] is high enough, then will often get the right answer



$\Delta X = \Delta G$ $\Delta H = \Delta I$ $\Delta J = \Delta Y$

Differential trails through 3TOY



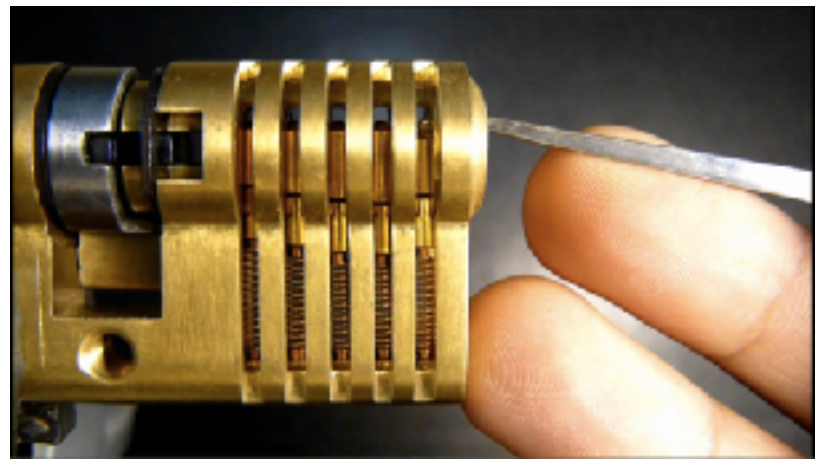
Y ?

Differential trail: $\Delta X \rightarrow \Delta I_1 \rightarrow \Delta I_2 \rightarrow \Delta Y$

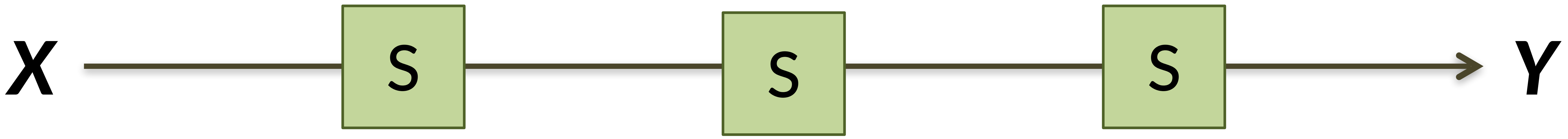
Y' ?

Two central themes of differential cryptanalysis

1. Internal variables might depend on the key, but *differences* between them may not!
2. Narrow key space by testing when (parts of) the key are consistent with known Δs



Differential trails through 3TOY



Differential trail: $\Delta X \dashrightarrow \Delta I_1 \dashrightarrow \Delta I_2 \dashrightarrow \Delta Y$

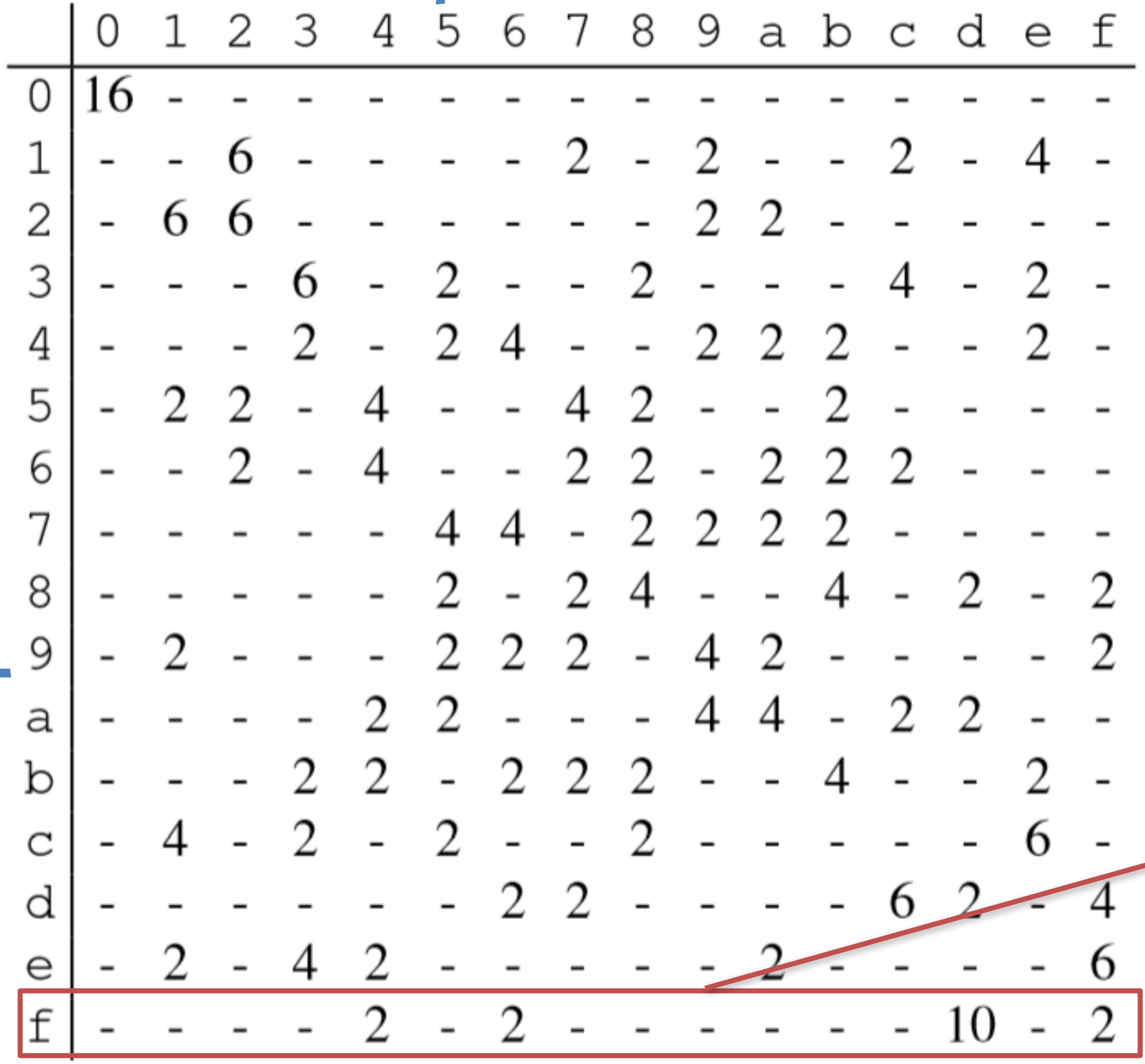
Example: $F \dashrightarrow D \dashrightarrow 6 \dashrightarrow 4$

Question: What is the probability of this trail occurring?

Difference propagation table

Output difference

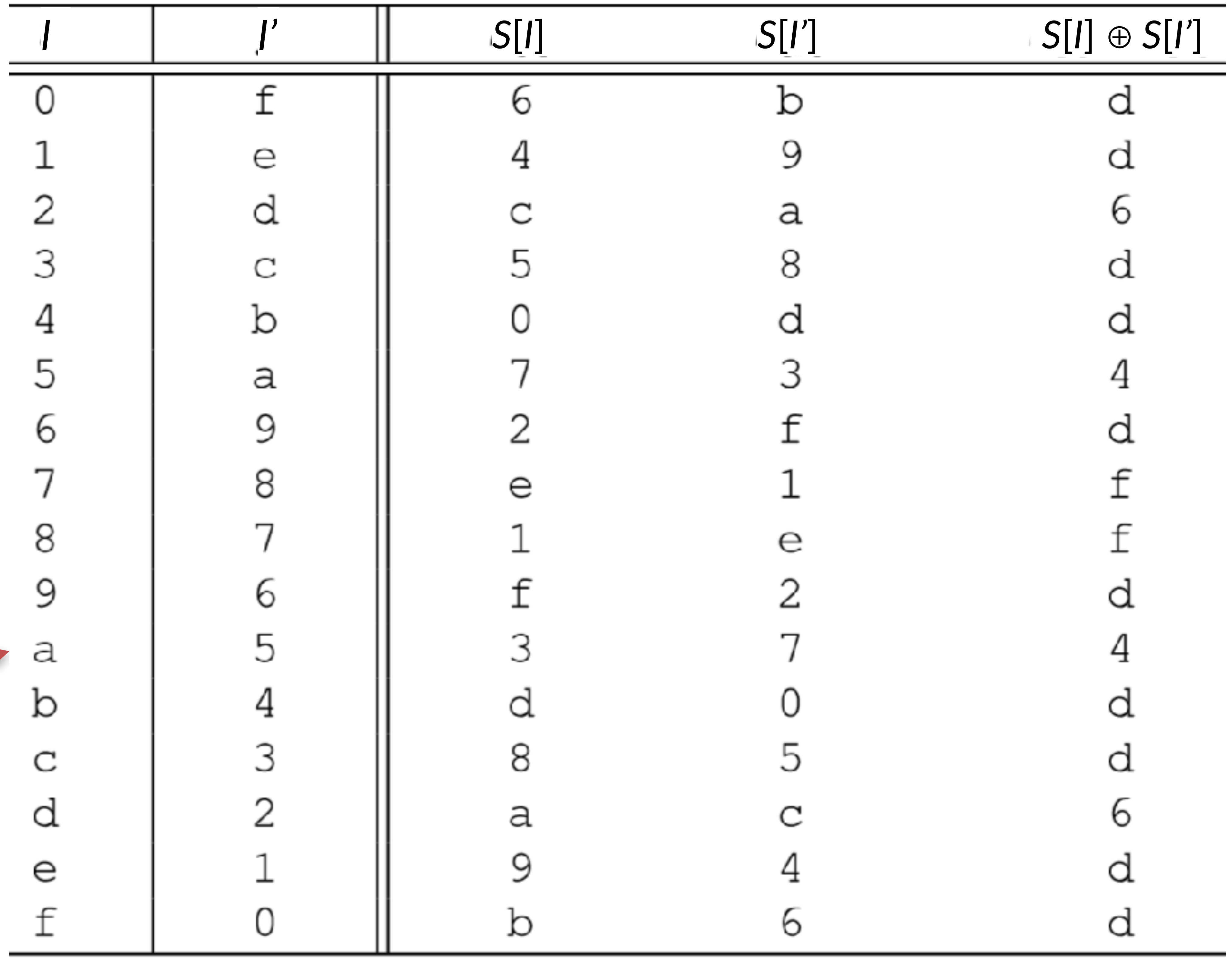
Input difference



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
a	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
b	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
c	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
d	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
e	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
f	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

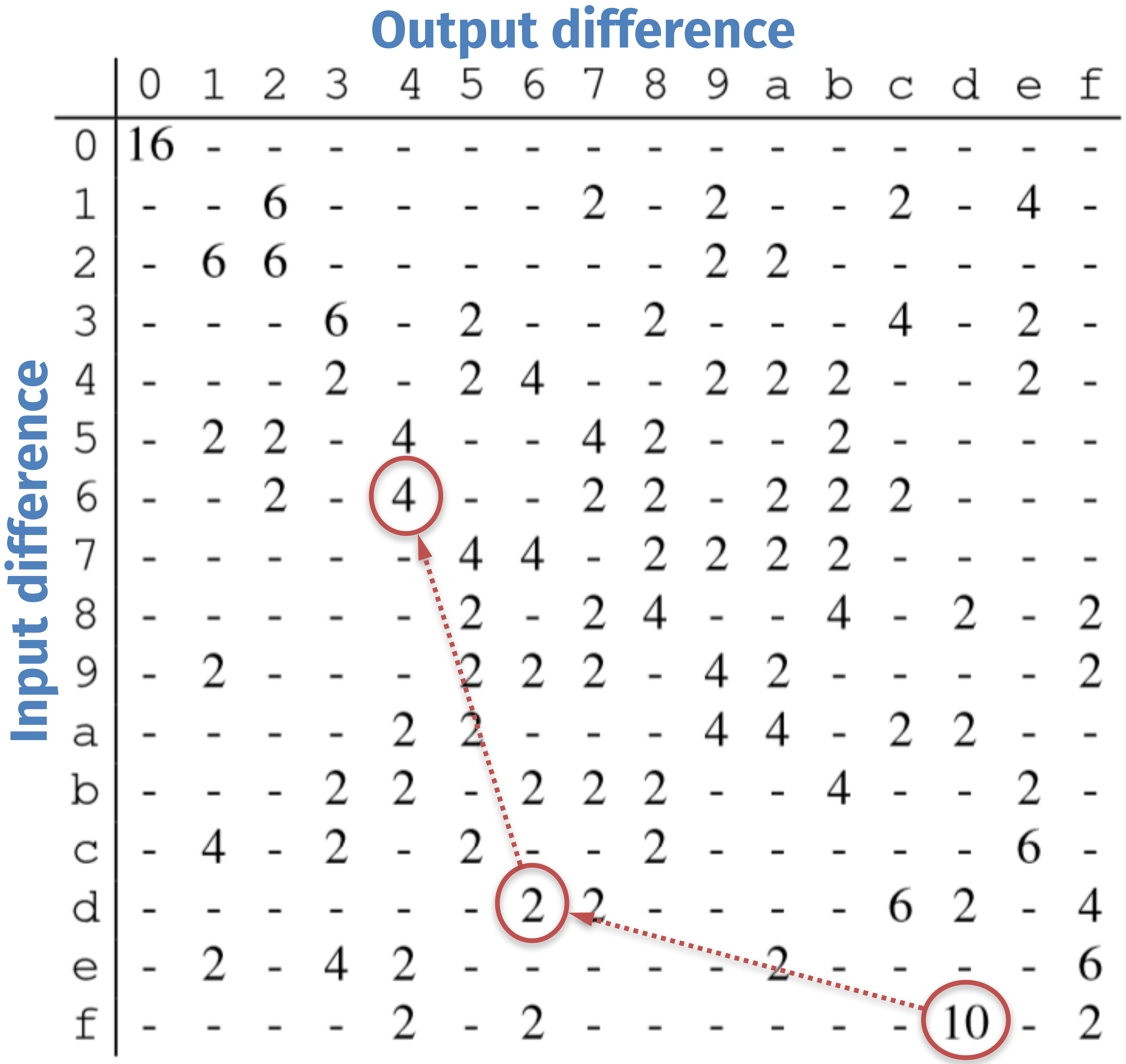
Table is based on S-box alone

Try all inputs differing by *row value*, see how often their outputs differ by *column value*



I	I'	$S[I]$	$S[I']$	$S[I] \oplus S[I']$
0	f	6	b	d
1	e	4	9	d
2	d	c	a	6
3	c	5	8	d
4	b	0	d	d
5	a	7	3	4
6	9	2	f	d
7	8	e	1	f
8	7	1	e	f
9	6	f	2	d
a	5	3	7	4
b	4	d	0	d
c	3	8	5	d
d	2	a	c	6
e	1	9	4	d
f	0	b	6	d

Difference propagation table


Output difference
Input difference

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
a	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
b	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
c	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
d	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
e	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
f	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

Table is based on S-box alone

Try all inputs differing by *row value*, see how often their outputs differ by *column value*

Computing Pr[trail]

Look up probability of each link, and multiply them together

Pr[F → D → 6 → 4]

≈ Pr[F → D] · Pr[D → 6] · Pr[6 → 4]

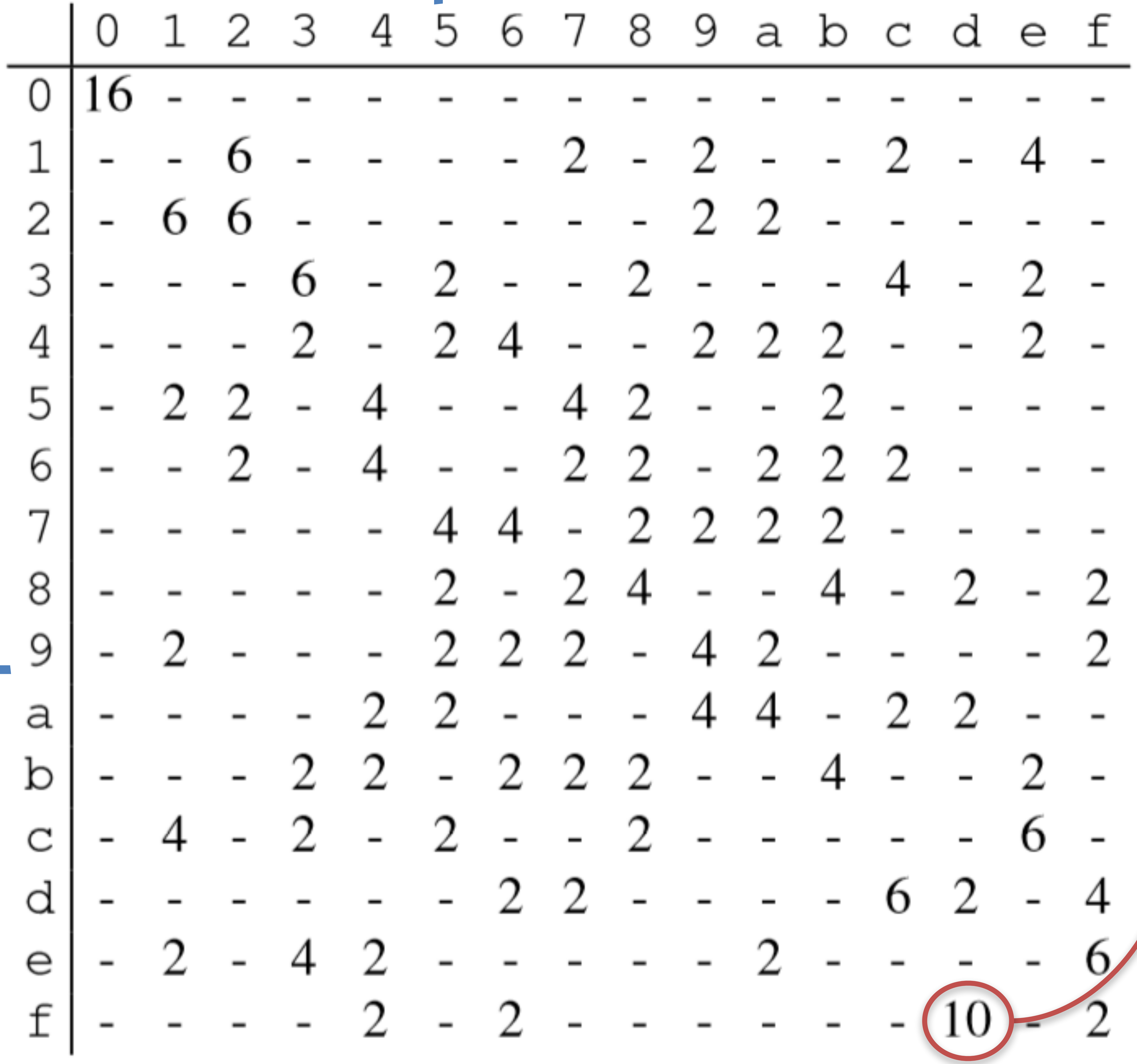
= 10/16 · 2/16 · 4/16 = 5/64

(Actually, the probabilities are not independent, whoops. But it tends to yield a value close to the right answer.)

Difference propagation table

Output difference

Input difference

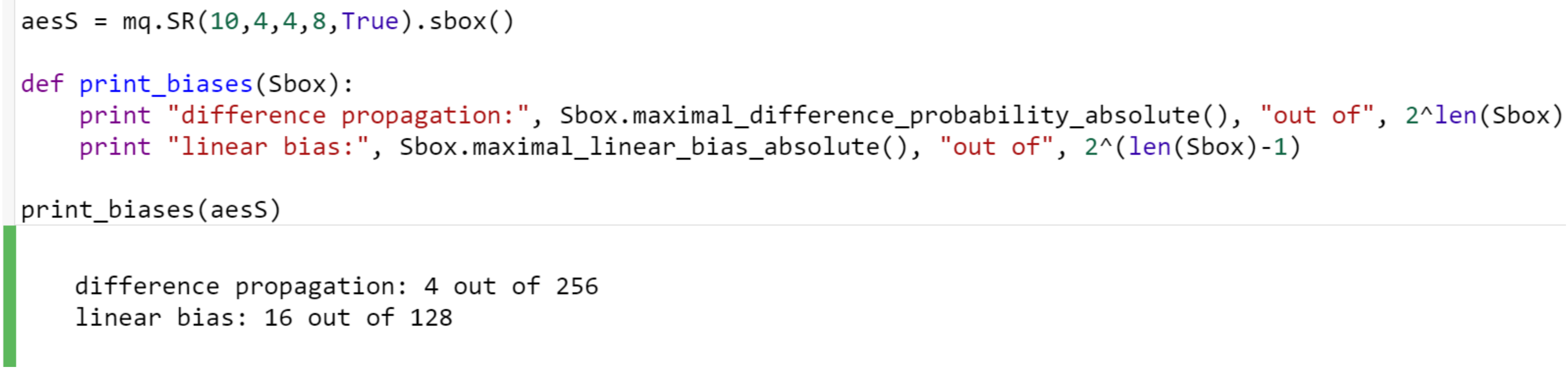


	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	-	6	-	-	-	-	2	-	2	-	-	2	-	4	-
2	-	6	6	-	-	-	-	-	-	2	2	-	-	-	-	-
3	-	-	-	6	-	2	-	-	2	-	-	-	4	-	2	-
4	-	-	-	2	-	2	4	-	-	2	2	2	-	-	2	-
5	-	2	2	-	4	-	-	4	2	-	-	2	-	-	-	-
6	-	-	2	-	4	-	-	2	2	-	2	2	2	-	-	-
7	-	-	-	-	-	4	4	-	2	2	2	2	-	-	-	-
8	-	-	-	-	-	2	-	2	4	-	-	4	-	2	-	2
9	-	2	-	-	-	2	2	2	-	4	2	-	-	-	-	2
a	-	-	-	-	2	2	-	-	-	4	4	-	2	2	-	-
b	-	-	-	2	2	-	2	2	2	-	-	4	-	-	2	-
c	-	4	-	2	-	2	-	-	2	-	-	-	-	-	6	-
d	-	-	-	-	-	-	2	2	-	-	-	-	6	2	-	4
e	-	2	-	4	2	-	-	-	-	-	2	-	-	-	-	6
f	-	-	-	-	2	-	2	-	-	-	-	-	-	10	-	2

Def. *Max difference propagation*

Largest one-round difference propagation in the entire table

Max difference propagation in the AES S-box

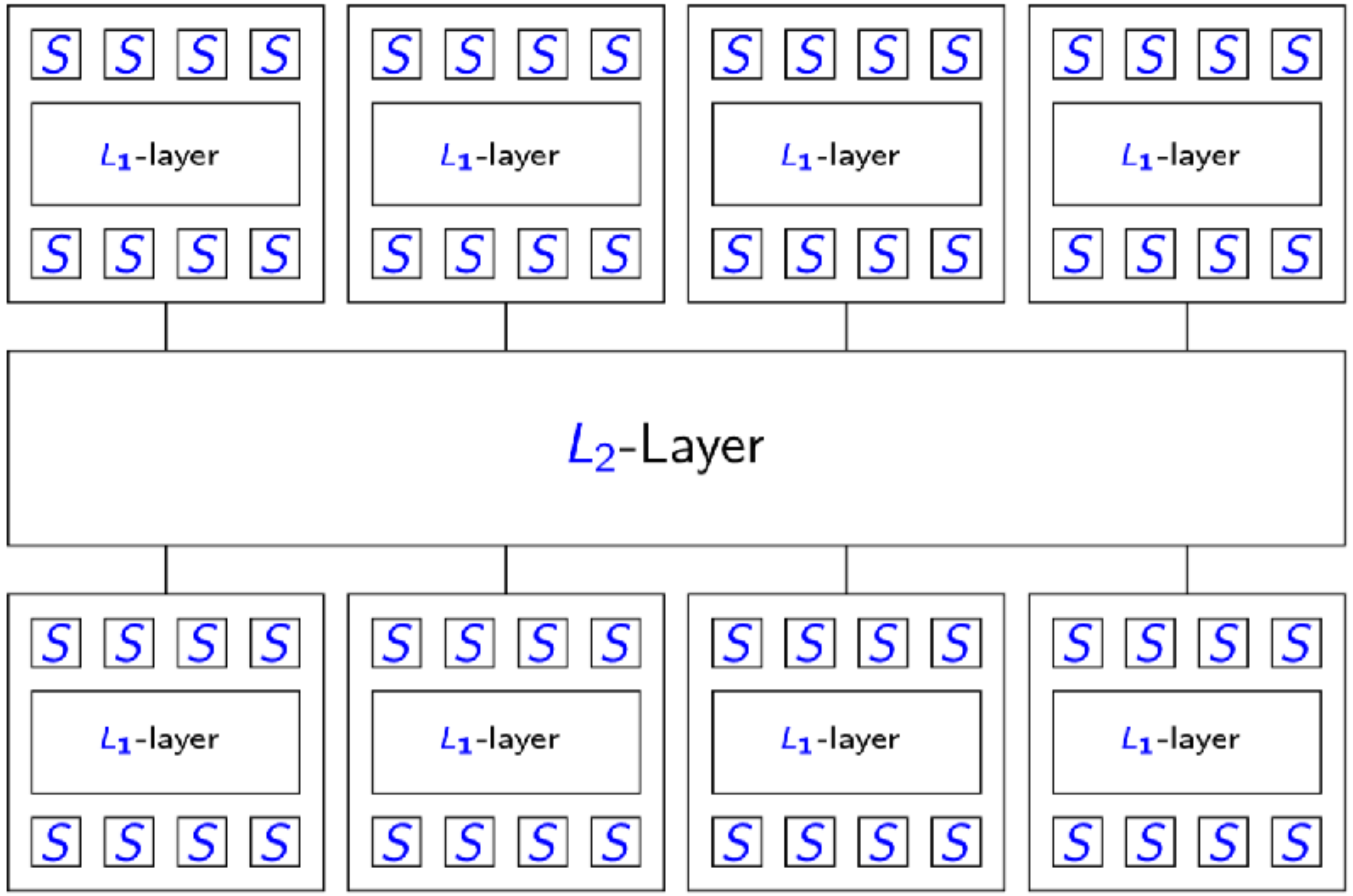


```
aesS = mq.SR(10,4,4,8,True).sbox()

def print_biases(Sbox):
    print "difference propagation:", Sbox.maximal_difference_probability_absolute(), "out of", 2^len(Sbox)
    print "linear bias:", Sbox.maximal_linear_bias_absolute(), "out of", 2^(len(Sbox)-1)

print_biases(aesS)
```

```
difference propagation: 4 out of 256
linear bias: 16 out of 128
```

Cryptanalysis of AES: Wide trail strategy through 4 rounds

- Picture depicts 4 rounds of AES
  - ≥ 25 active S-boxes in 4 rounds
  - Each has max diff propagation of 2^{-6}
- So Pr [four-round trail] ≈ 2^{-150}
  - An 8-round trail has C < 2^{-300}
  - A 12-round trail has C < 2^{-450}
- Brute force search is better



“Instead of spending most of its resources on large S-boxes, the wide trail strategy aims at designing the round transformations such that there are no [linear or differential] trails/characteristics of low weight”

Bounds for differential trails in Keccak-$f[1600]$

Rounds	Lower bound		Best known	
1	2		2	
2	8		8	
3	32	[Keccak team]	32	[Duc et al.]
4			134	[Keccak team]
5			510	[Naya-Plasencia et al.]
6	74	[Keccak team]	1360	[Keccak team]
24	296		???	

Slide from
Keccak team

New topic: Protecting data while computing

- We saw our first example of protecting data while computing last week, when we built an “Oblivious PRF” as a building block toward PAKE
  - Punchline: Alice and Bob worked together even while they viewed each other as ‘adversaries’ trying to learn their sensitive input data
- Now let’s protect our sensitive data even while performing an *arbitrary* calculation over our joint inputs
- Credit: the slides in this portion of the lecture were created by Mike Rosulek at Oregon State (web.engr.oregonstate.edu/~rosulekm/crypto)