CMSC/Math 456: Cryptography (Fall 2023)

Lecture 7

Daniel Gottesman

Administrative

Problem Set #3 is due on Thursday Sep. 21 at noon.

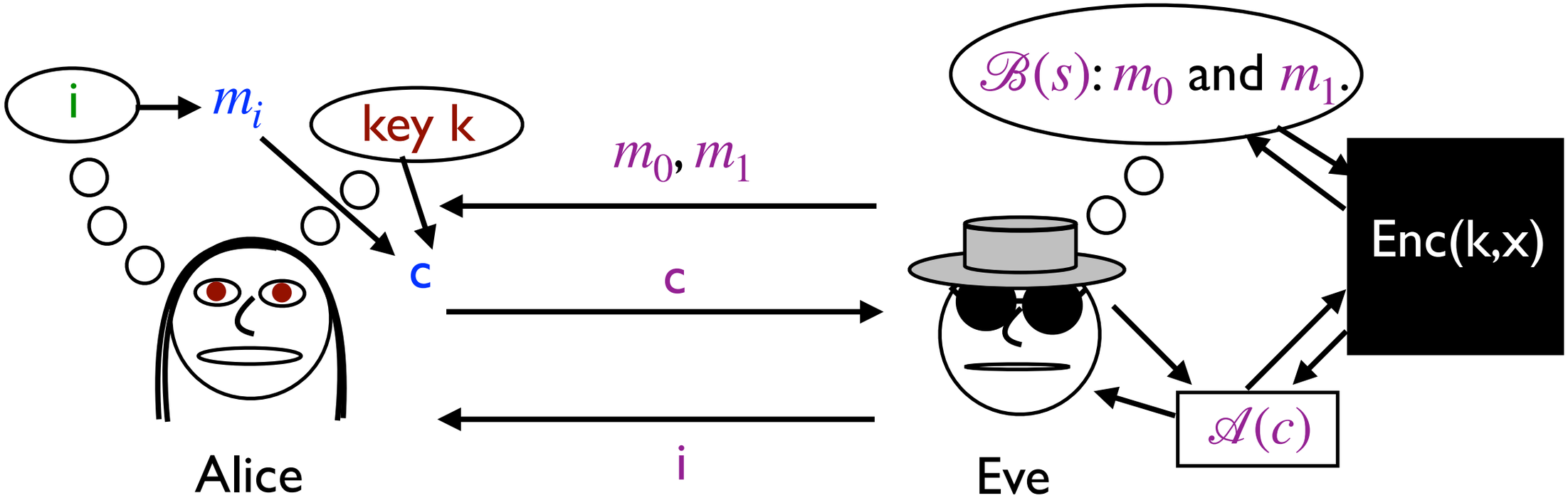
This class is being recorded

Definition of CPA Security

Definition: (Enc, Dec) with security parameter s is CPA-secure if, for any pair of messages m_0 and m_1 chosen by the adversary (using $\mathcal{B}(s)$ and oracle access to Enc(k,x)) and for any efficient attack $\mathcal{A}(c)$ (also with oracle access to Enc(k,x))

$$|\Pr_k(\mathcal{A}(Enc(k, m_0)) = 1) - \Pr_k(\mathcal{A}(Enc(k, m_1)) = 1)| \leq \epsilon(s)$$

for negligible $\epsilon(s)$ and probability taken over k and randomness of Enc.



This class is being recorded

CPA Security With Stream Ciphers

Suppose we have a stream cipher that takes an IV. Suppose on initial value IV and key k, the stream cipher outputs the string

$$x_0, x_1, x_2, x_3, x_4, \ldots$$

Then if we have the message $(m_0, m_1, m_2, m_3, \ldots)$, we encrypt with the ciphertext $(IV, m_0 \oplus x_0, m_1 \oplus x_1, m_2 \oplus x_2, \ldots)$.

Enc takes as input k and m. It generates a random IV, uses IV and k to generate the x string, and then encrypts as above.

Since Bob gets IV as part of the ciphertext and knows k, he can also generate the bit stream x and decrypt.

> But this is CPA-secure: because the IV is very unlikely to repeat (assuming it is long enough), there will be a new stream each time and Eve has no way to predict it without knowing the key.

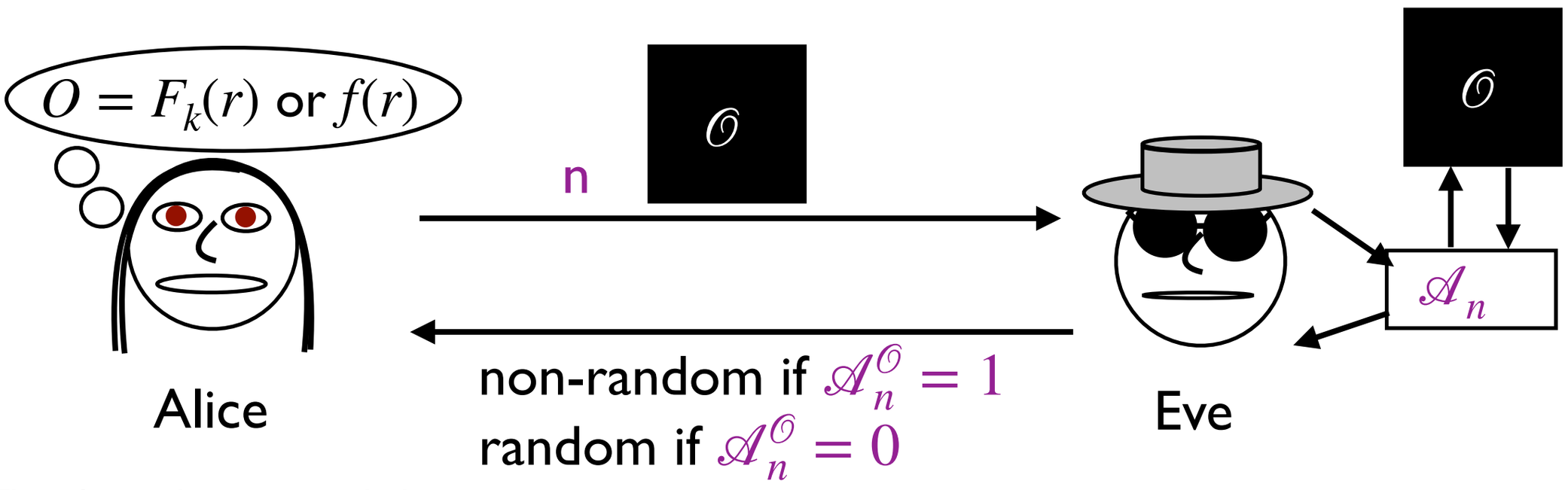
This class is being recorded

Pseudorandom Functions

Definition: Let $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a *deterministic efficiently computable* function with $|F_k(r)| = |r| = n$ and $|k| = s = \text{poly}(n)$. Then $F_k(r)$ is a pseudorandom function if, for any efficient attack $\mathcal{A}_n$ that accesses an oracle and outputs a bit,

$$|\Pr_k(\mathcal{A}_n^{F_k(r)} = 1) - \Pr_f(\mathcal{A}_n^f = 1)| \leq \epsilon(s)$$

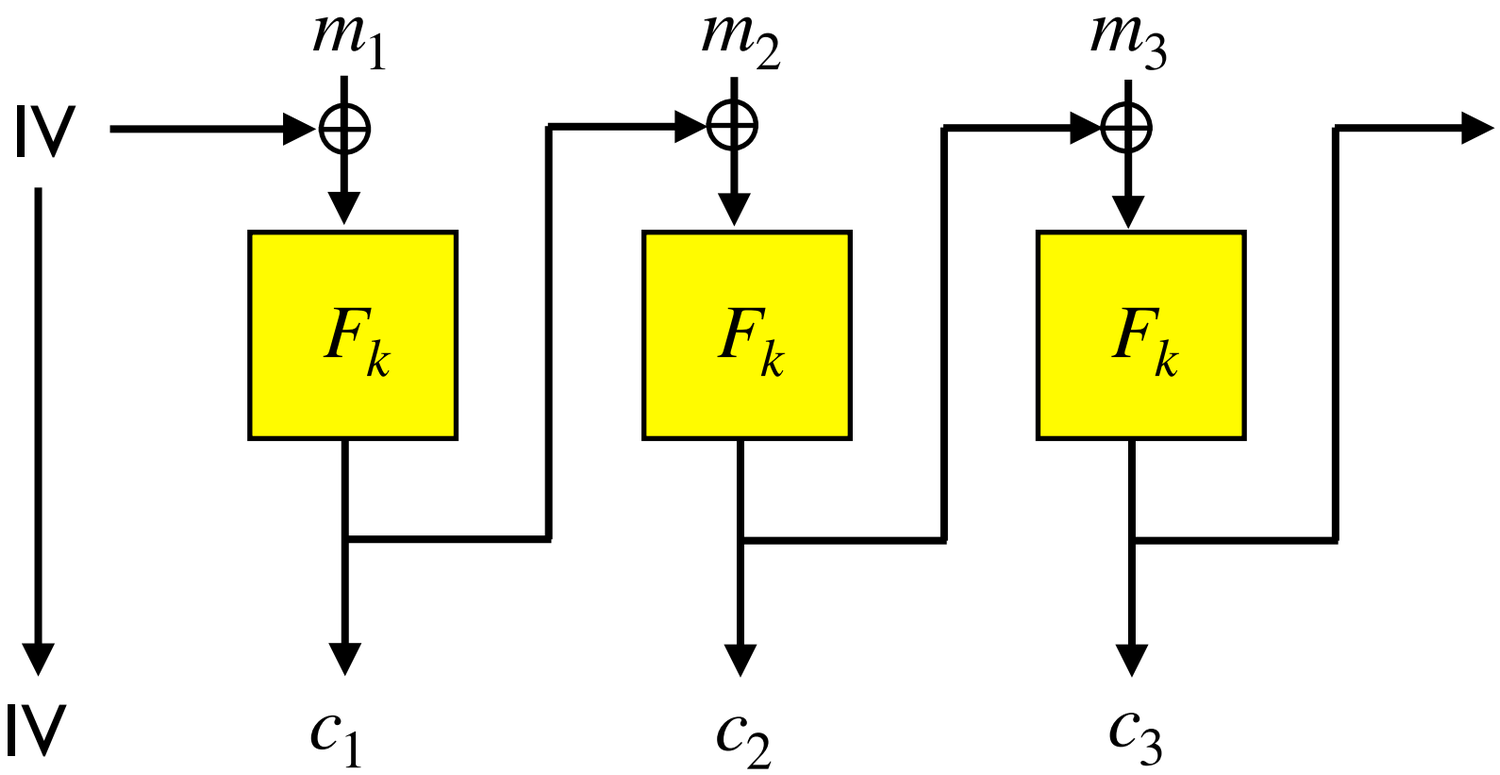
with $\epsilon(s)$ a negligible function and probabilities averaged over randomness of $\mathcal{A}$, as well as over uniformly random keys k (left probability) and truly random functions f (right probability).



This class is being recorded

Block Ciphers

Recall that we were discussing block ciphers that can be secure against a chosen plaintext attack. For example, CBC mode builds on smaller units, each of which is based on a pseudorandom permutation. DES and AES are examples of pseudorandom permutations.



To decrypt, Bob must invert F_k on block i and XOR with c_{i-1}.

This class is being recorded

Block Cipher vs. Random Function

Let's play the game from the definition of pseudorandom function. I will have a function which is either a random permutation or a candidate pseudorandom permutation, using the following rule:

Key: A pair of numbers k = (A,B) (non-negative integers less than 32).
Function:

$$F_k(x) = (Ax + B) \bmod 32$$

(When we study modular arithmetic, you will see that this is a permutation when A is relatively prime to 32.)

Try it: Ask me queries and try to determine which it is, using a limited number of queries.

This class is being recorded

Failure of Test Case

Why is this a bad candidate for pseudorandom permutation?

$$F_k(x) = (Ax + B) \bmod 32$$

One problem is that certain inputs directly expose part of the key:

$$F_k(0) = B$$

Another problem is that a change in the input produces a very predictable change in the output:

$$F_k(x + 1) = (F_k(x) + A) \bmod 32$$

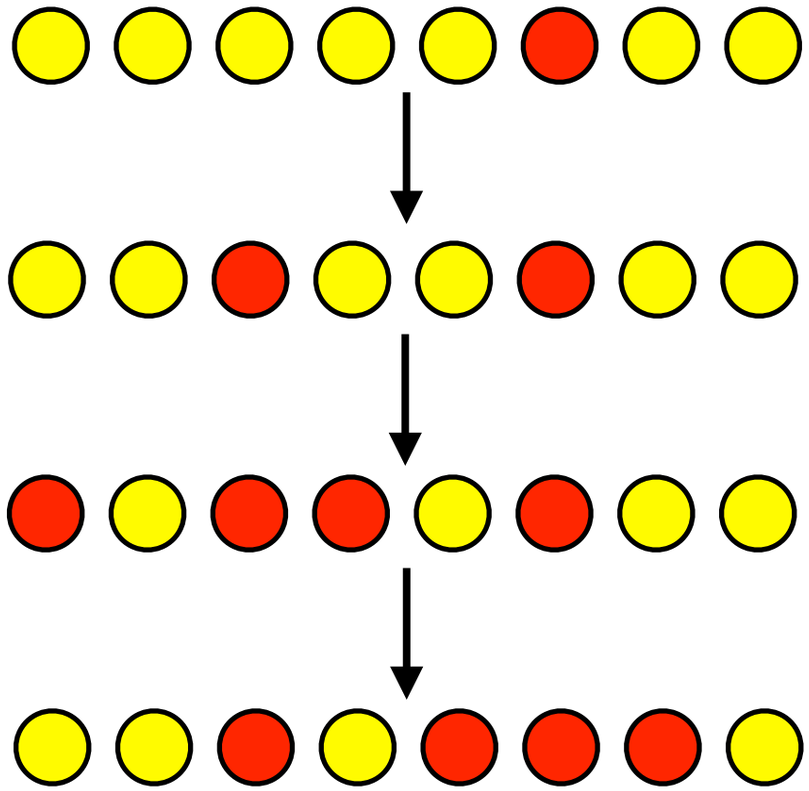
We want to avoid by designing the functions so that related inputs have very different outputs.

This class is being recorded

Goals of Block Cipher Design

- Must be invertible to use with CBC mode (i.e., pseudorandom permutation rather than pseudorandom function).
- Even when the inputs are related, the outputs should be very different.

In particular, the change of even a single bit of the input should result in a totally different output. This is known as the "avalanche effect." It is often achieved by having multiple rounds, each of which magnifies small changes.

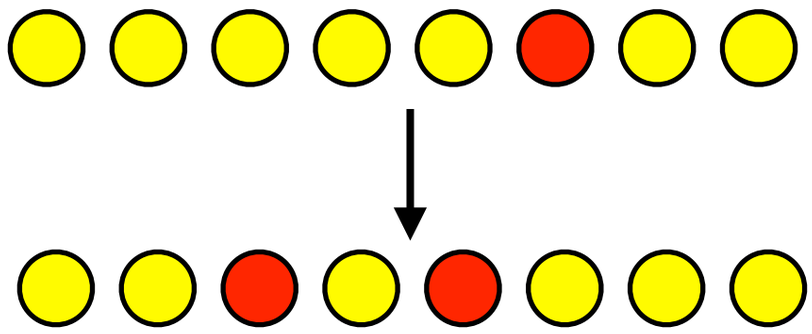


This class is being recorded

Avalanche Effect vs. Breaking Cipher

Why does the avalanche effect contribute to making a cipher secure against chosen plaintext attacks?

Because we are focusing on permutations, Eve can try to trace a ciphertext backwards to the input. Because she doesn't know the key, she has limited ability to do this.

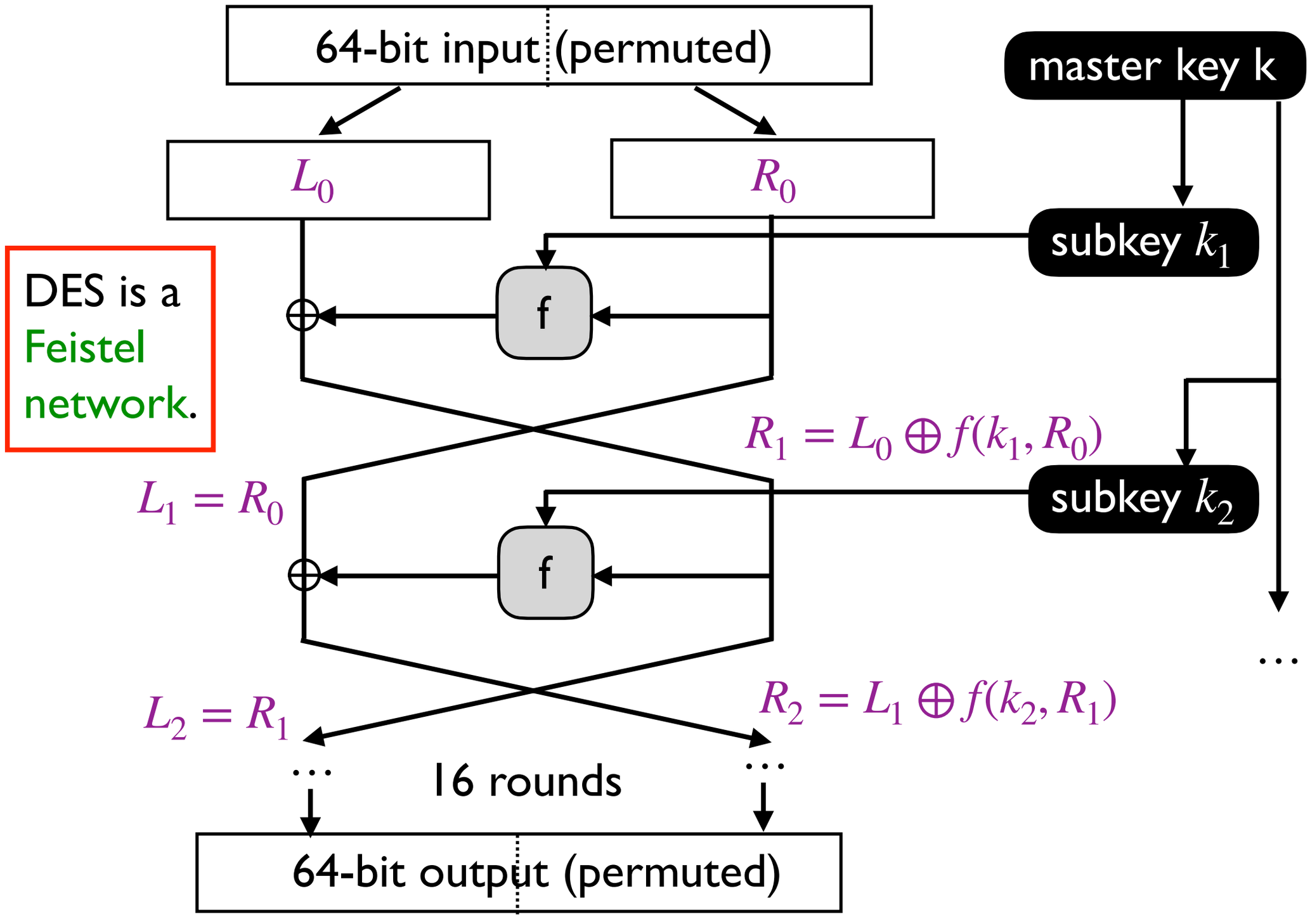


However, if there is no avalanche effect and Eve gets two (plaintext, ciphertext) pairs that differ in a small fraction of bits of the plaintext, the ciphertexts will also only differ in a limited set of places.

She can then focus on deducing the segment of the key that is relevant to those locations. This is a much smaller search than searching over all possible key values.

This class is being recorded

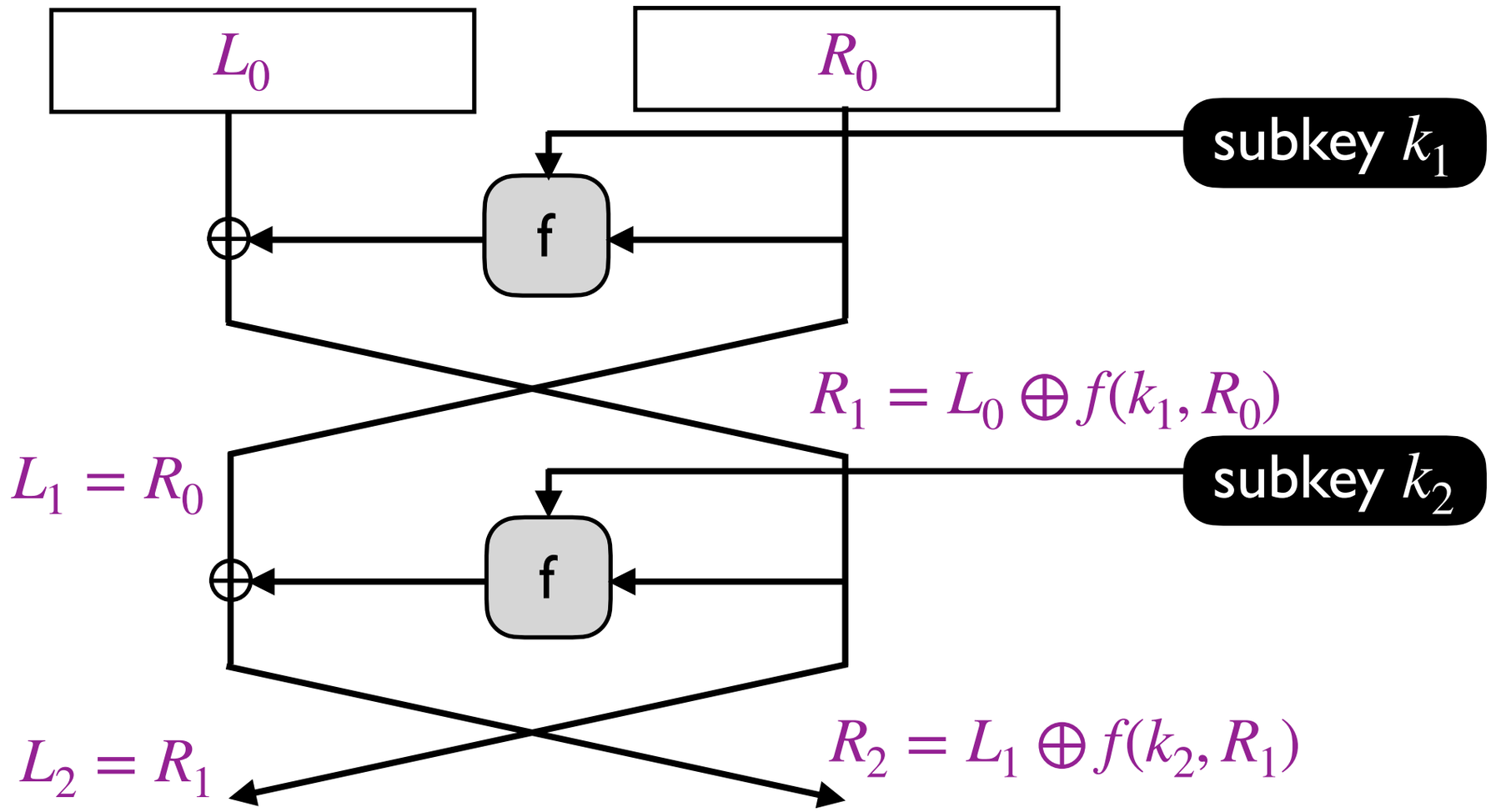
DES Overview



This class is being recorded

Feistel Network

A Feistel network consists of a sequence of rounds sequentially acting on the message, which is split into a left and right half.



In each round, the current right half is fed into a round function f with a key for the round and then XORed with the left half. The modified left half and old right half are then switched.

This class is being recorded

Inverse of Feistal Network

A Feistal network is automatically a permutation. (Permutation here means we are permuting the strings and not just the bits.) In particular, it has a straightforward inverse for someone with access to the key: Simply run the network backwards.

Suppose we know R_i and L_i. Since $L_i = R_{i-1}$, we can always determine what the input to the f function in round i was. Then we can calculate

$$L_{i-1} = R_i \oplus f(k_{i-1}, R_{i-1})$$

which gives us R_{i-1} and L_{i-1}. We can then work our back to the beginning.

Notice that f does not have to be invertible itself.

This class is being recorded

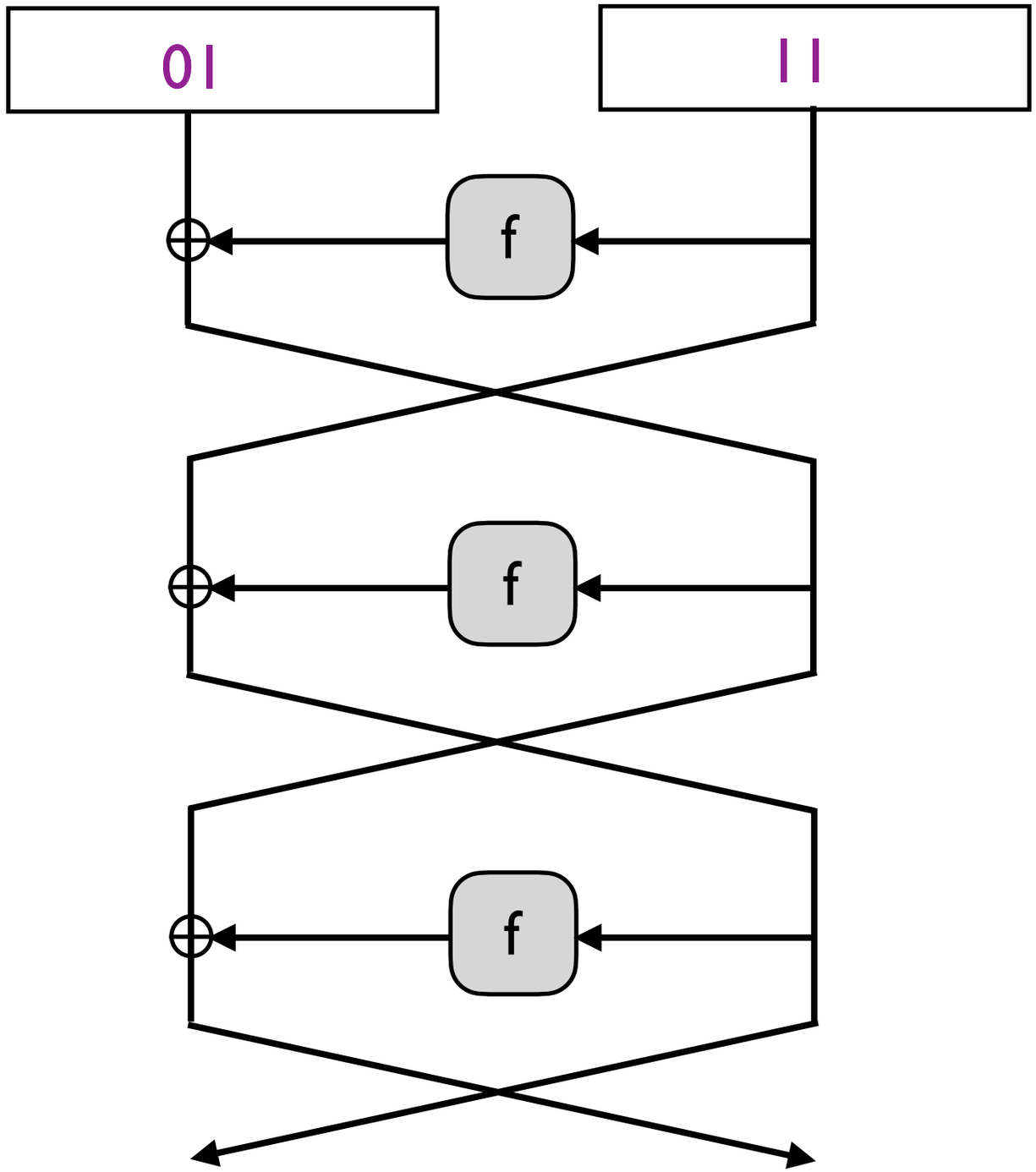
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

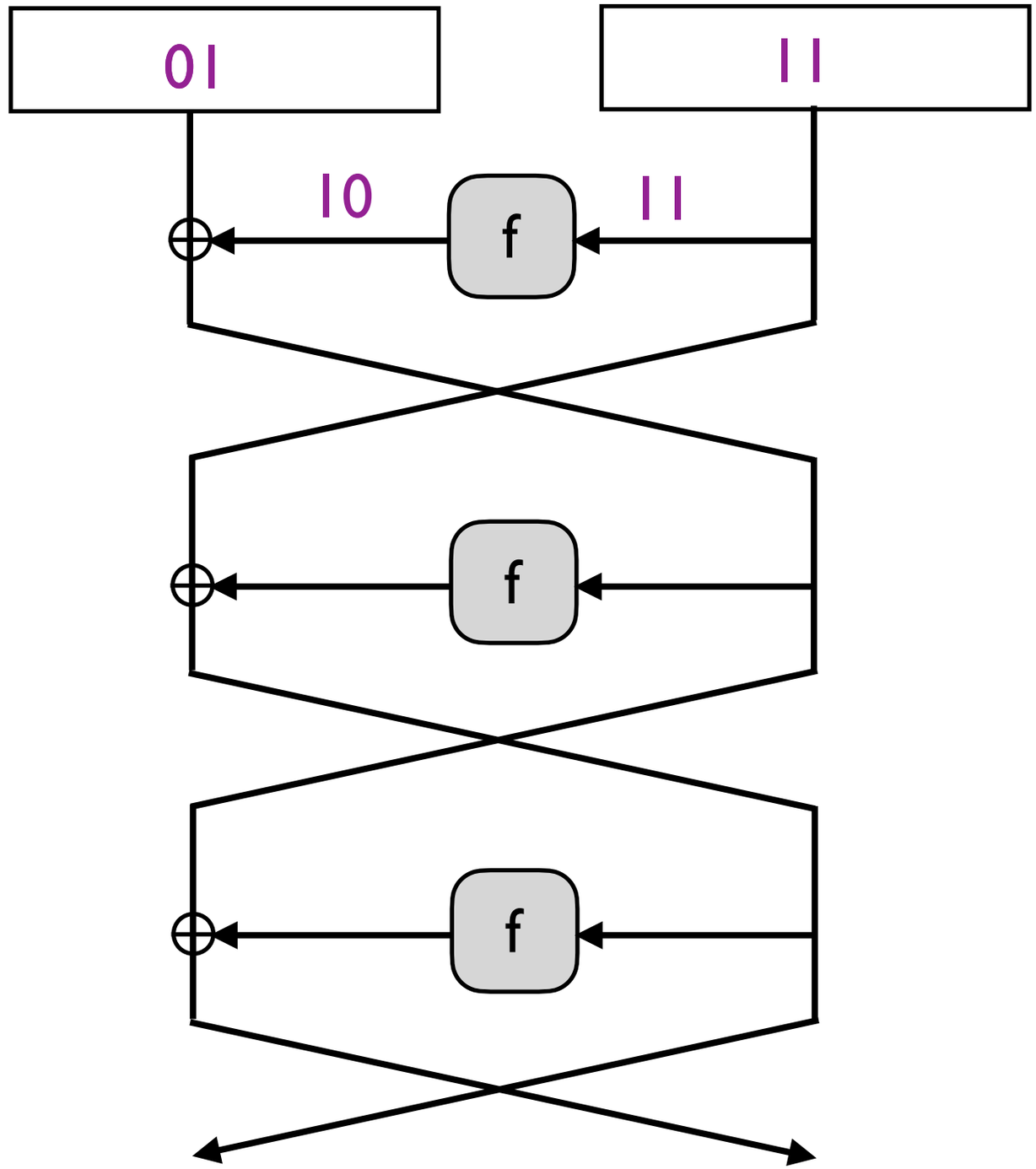
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

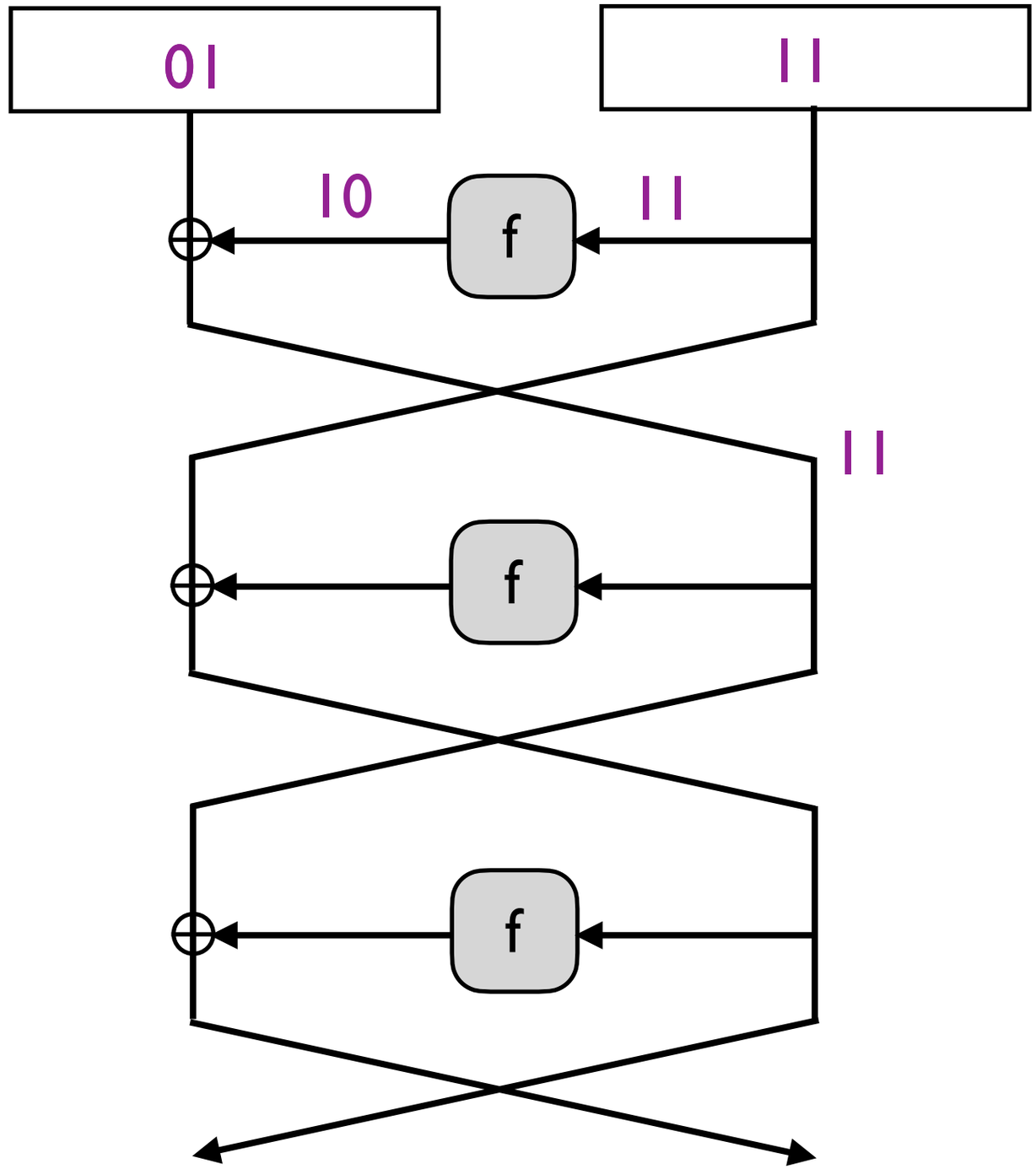
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



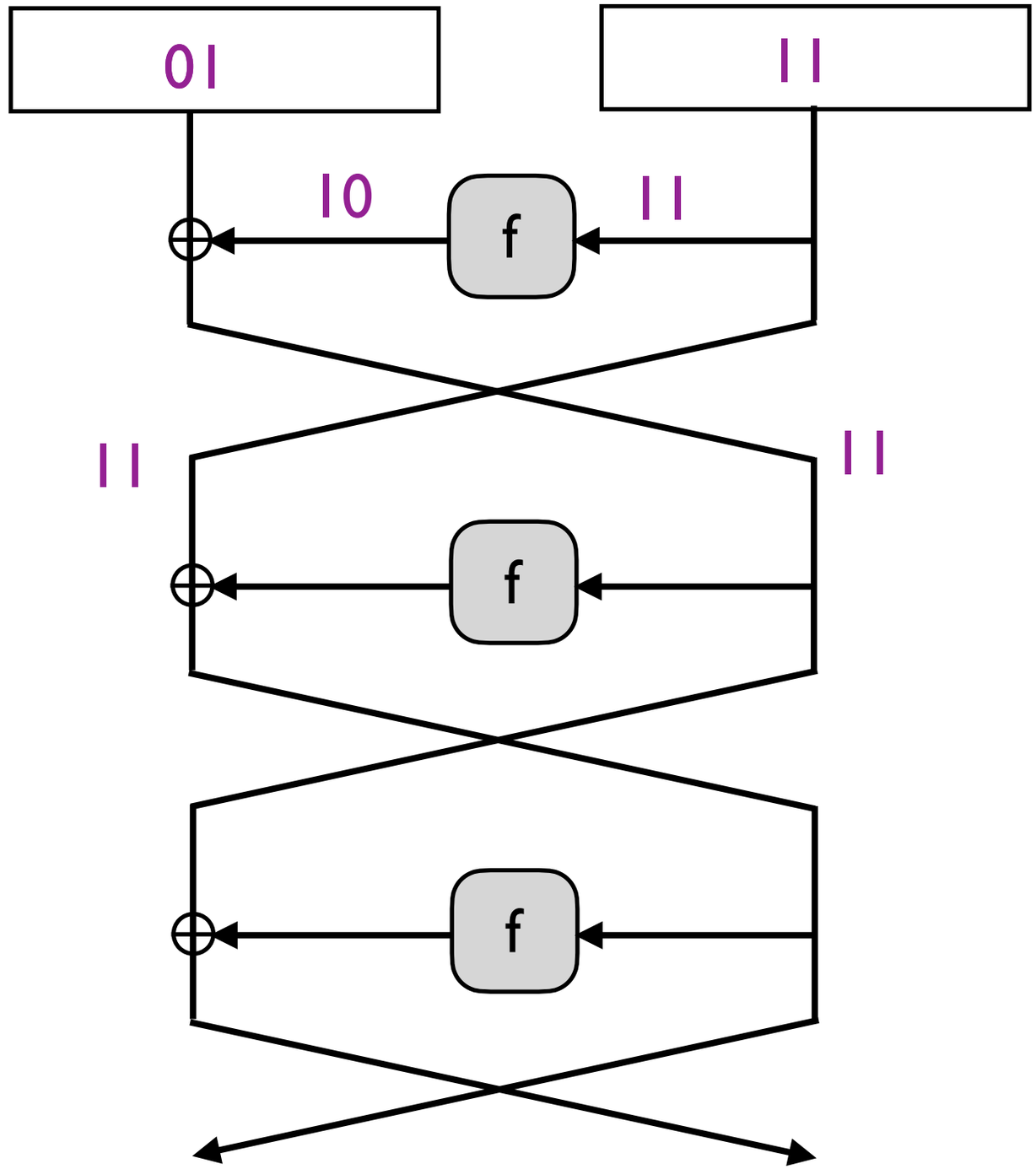
This class is being recorded

Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:
$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



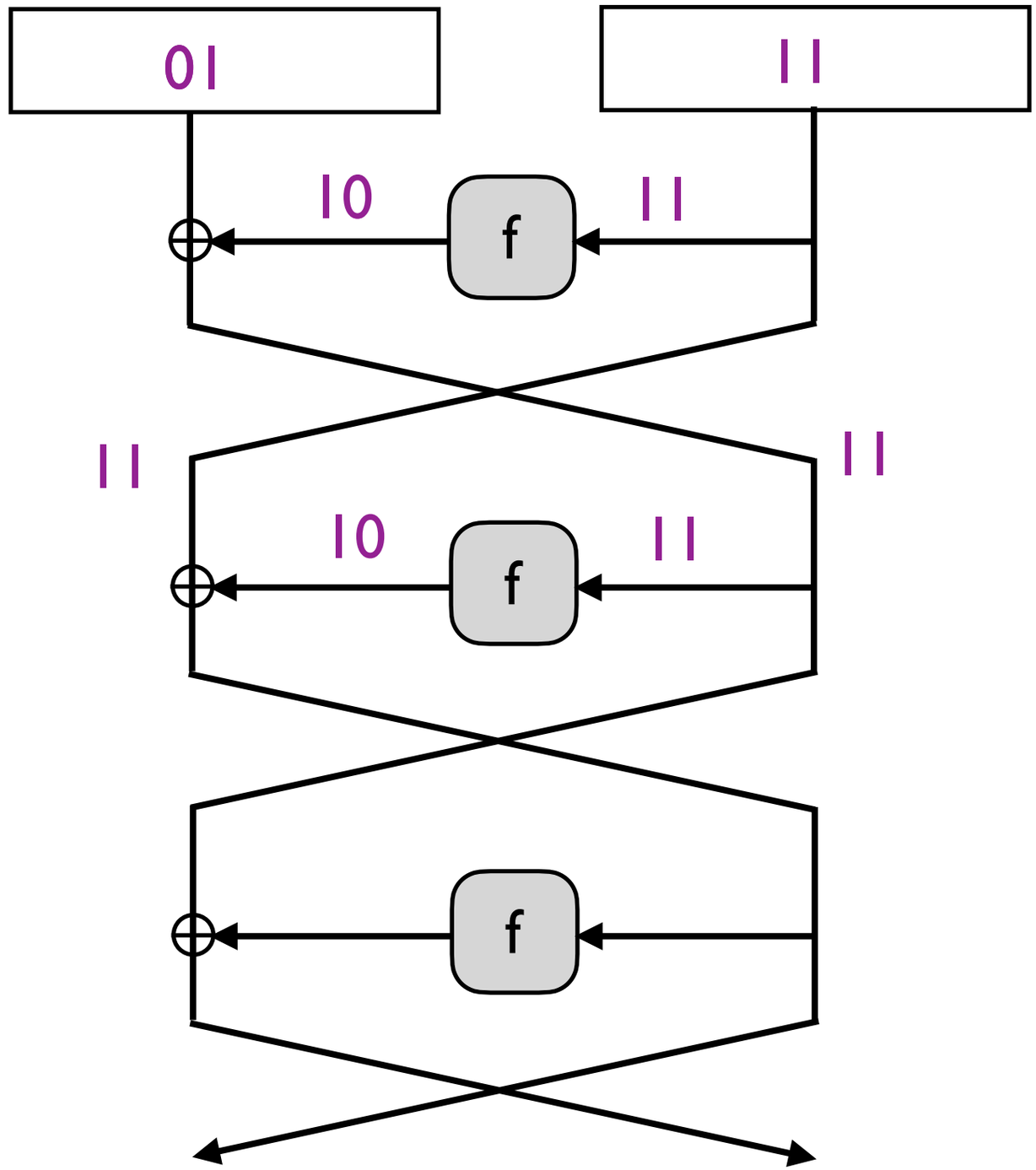
This class is being recorded

Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:
$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

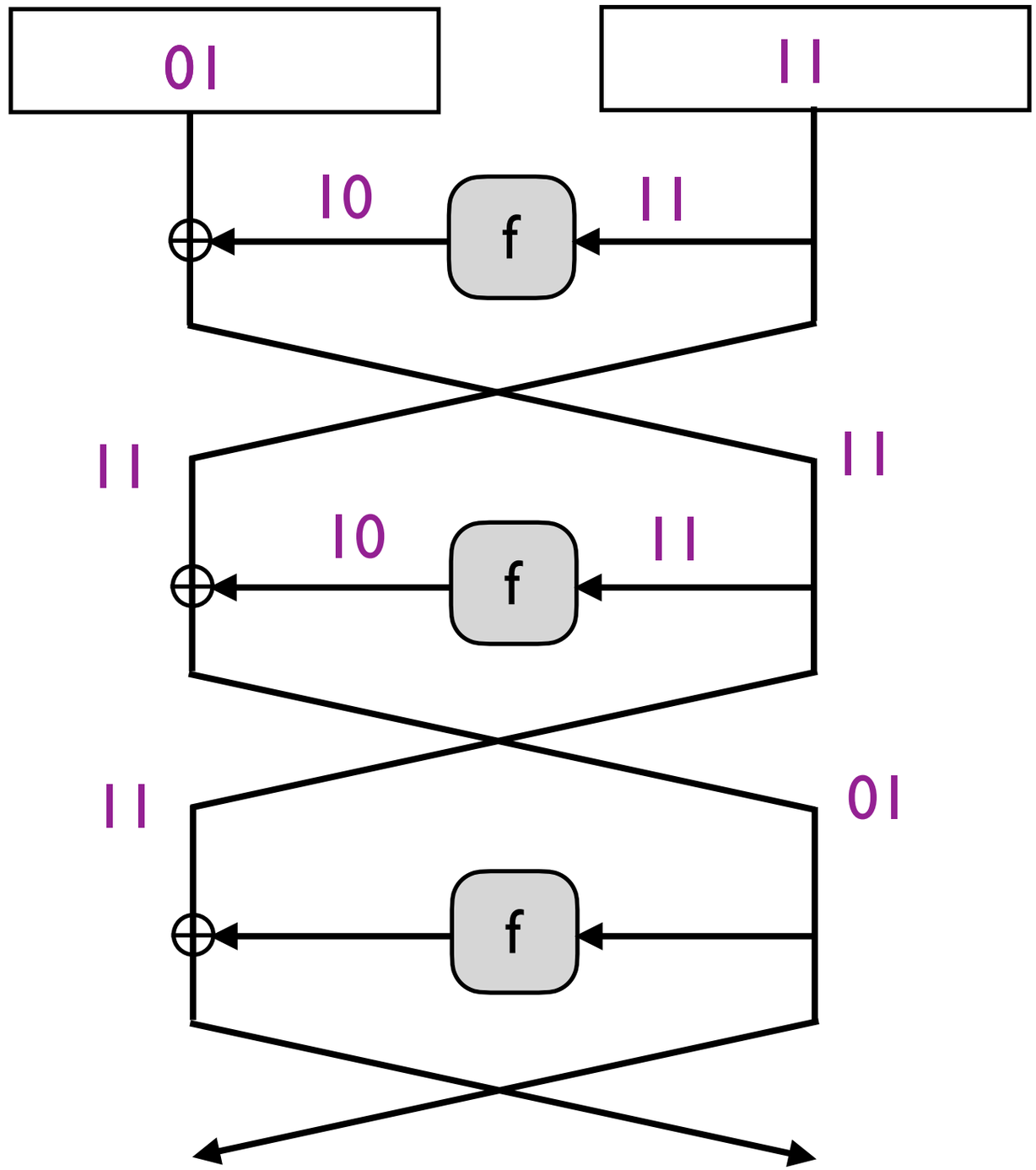
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

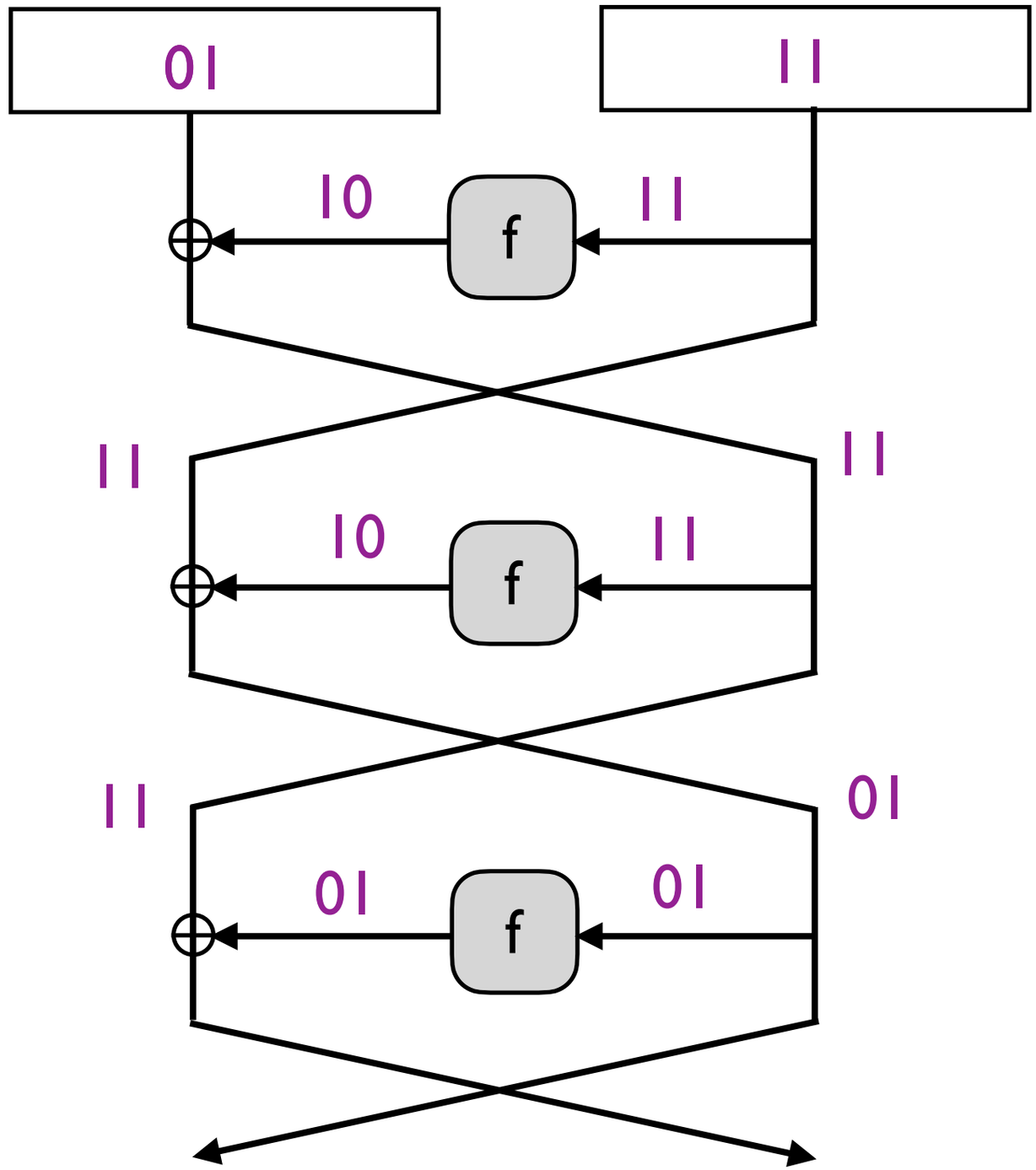
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

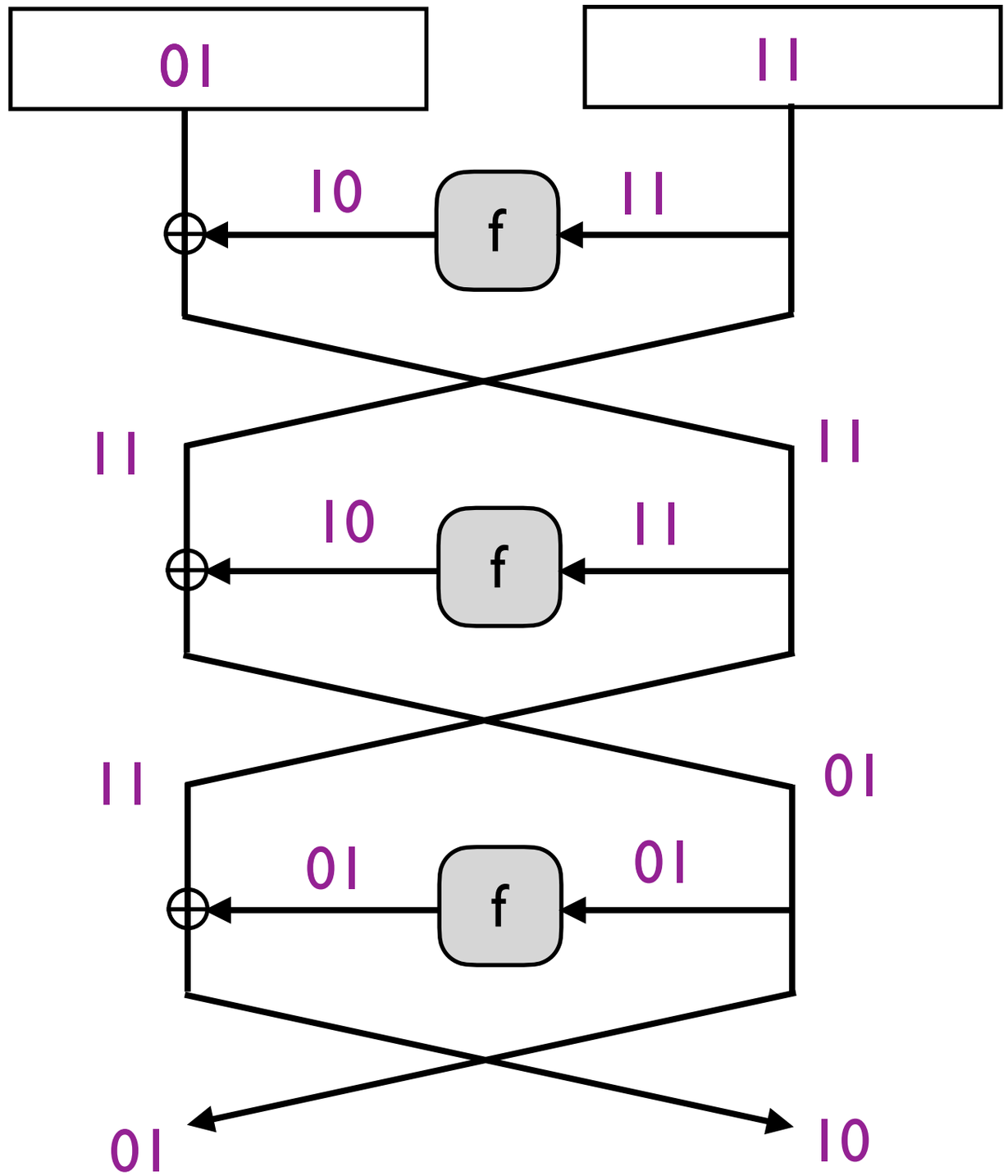
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

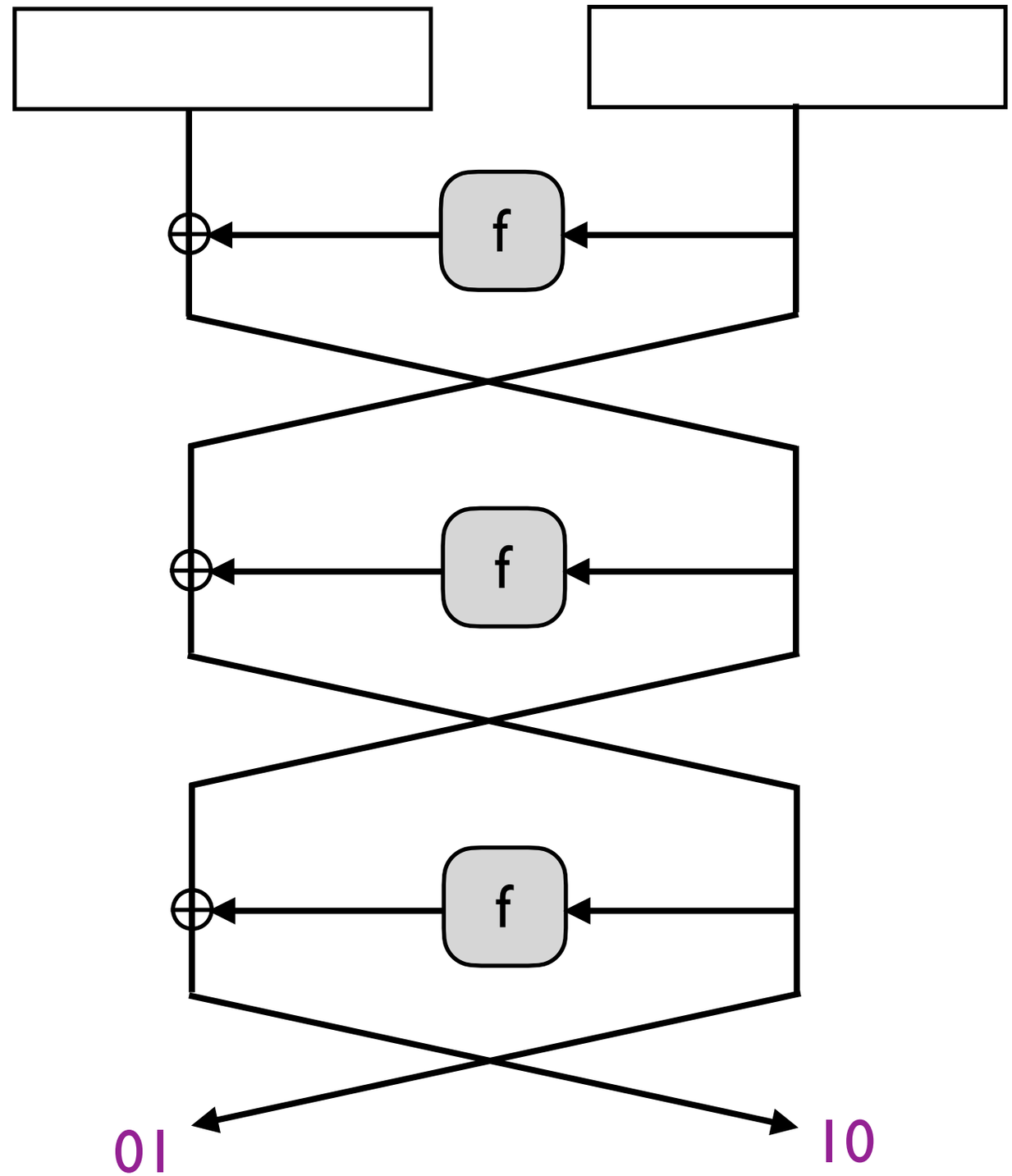
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

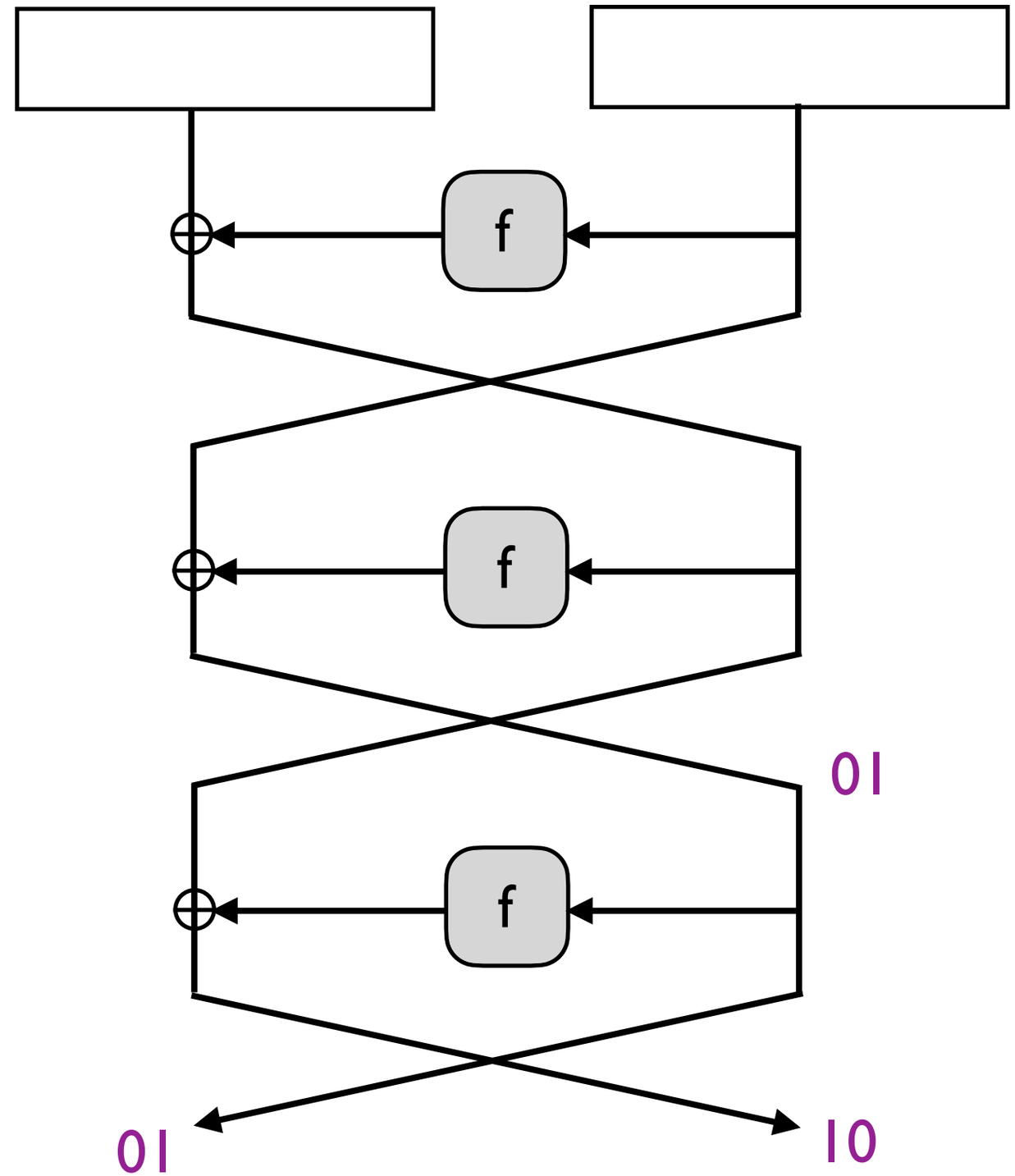
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

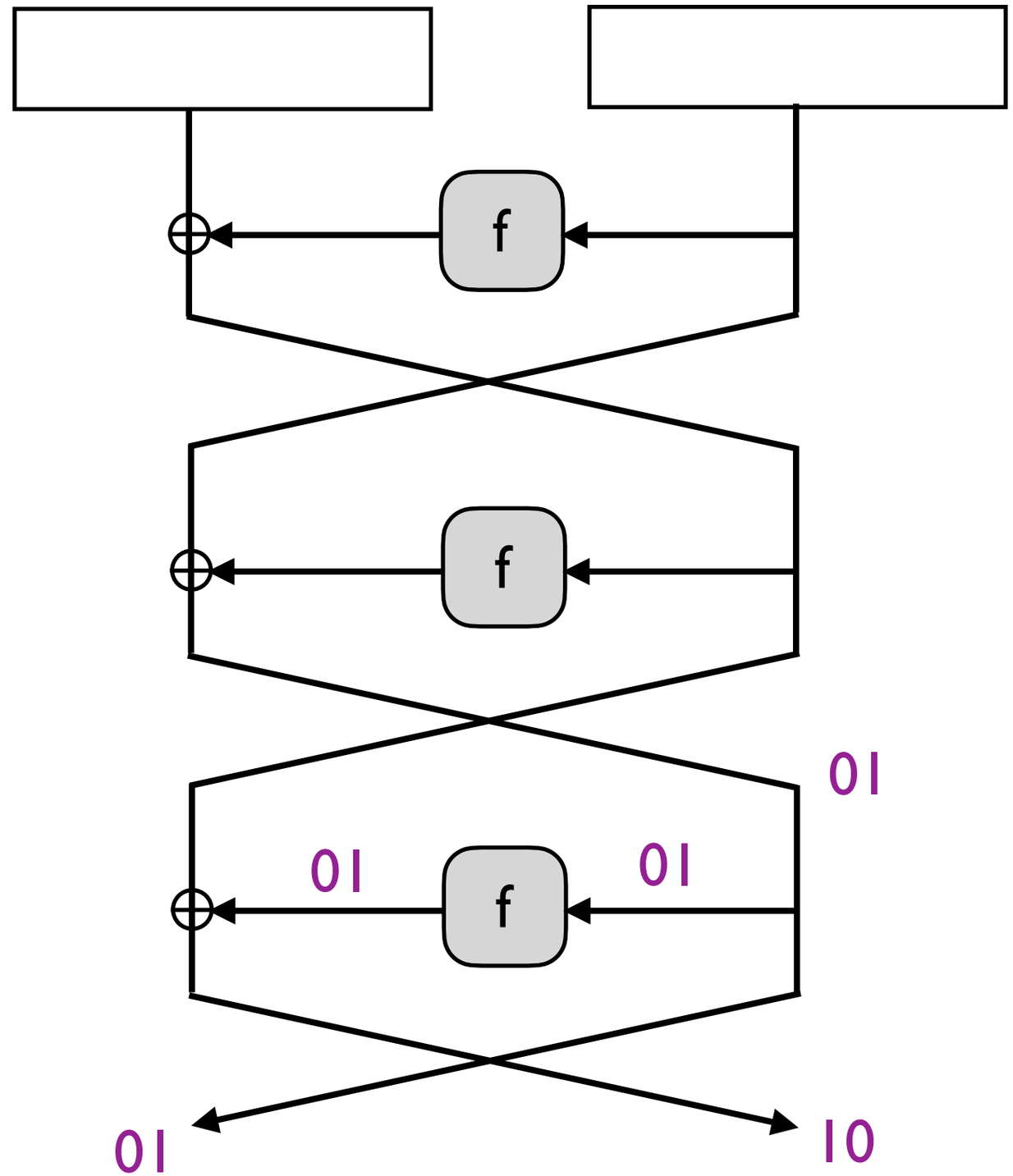
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

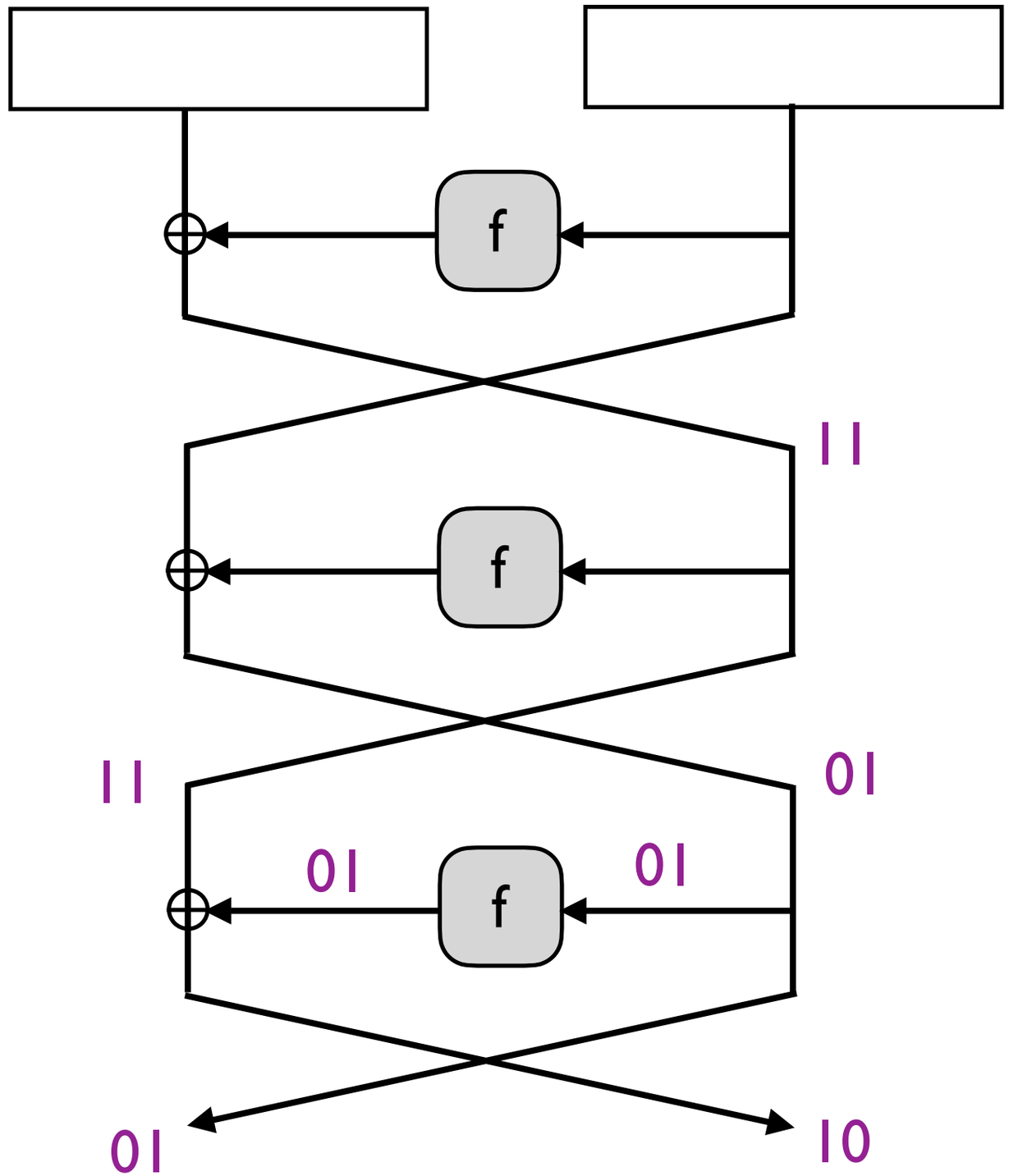
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

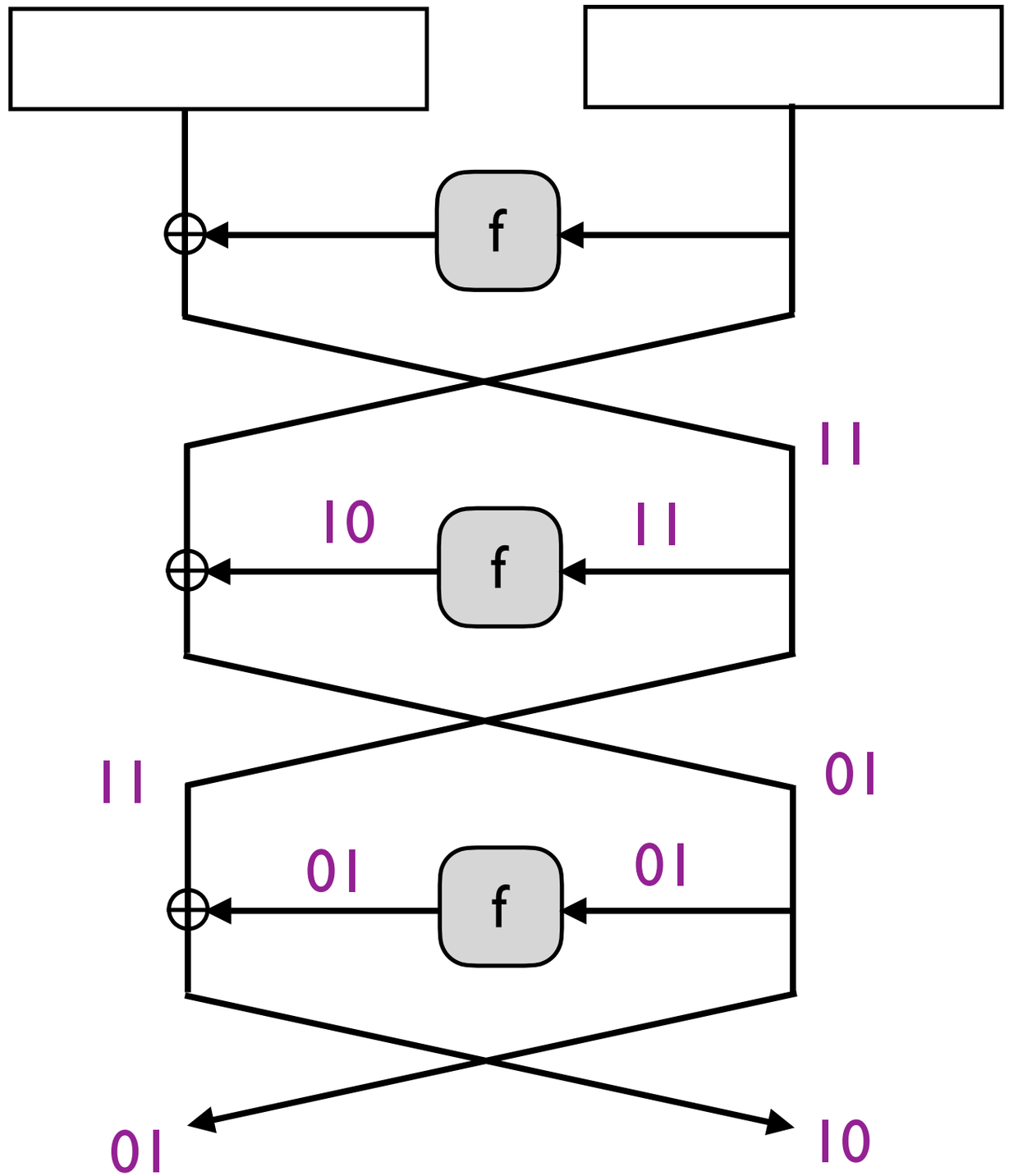
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

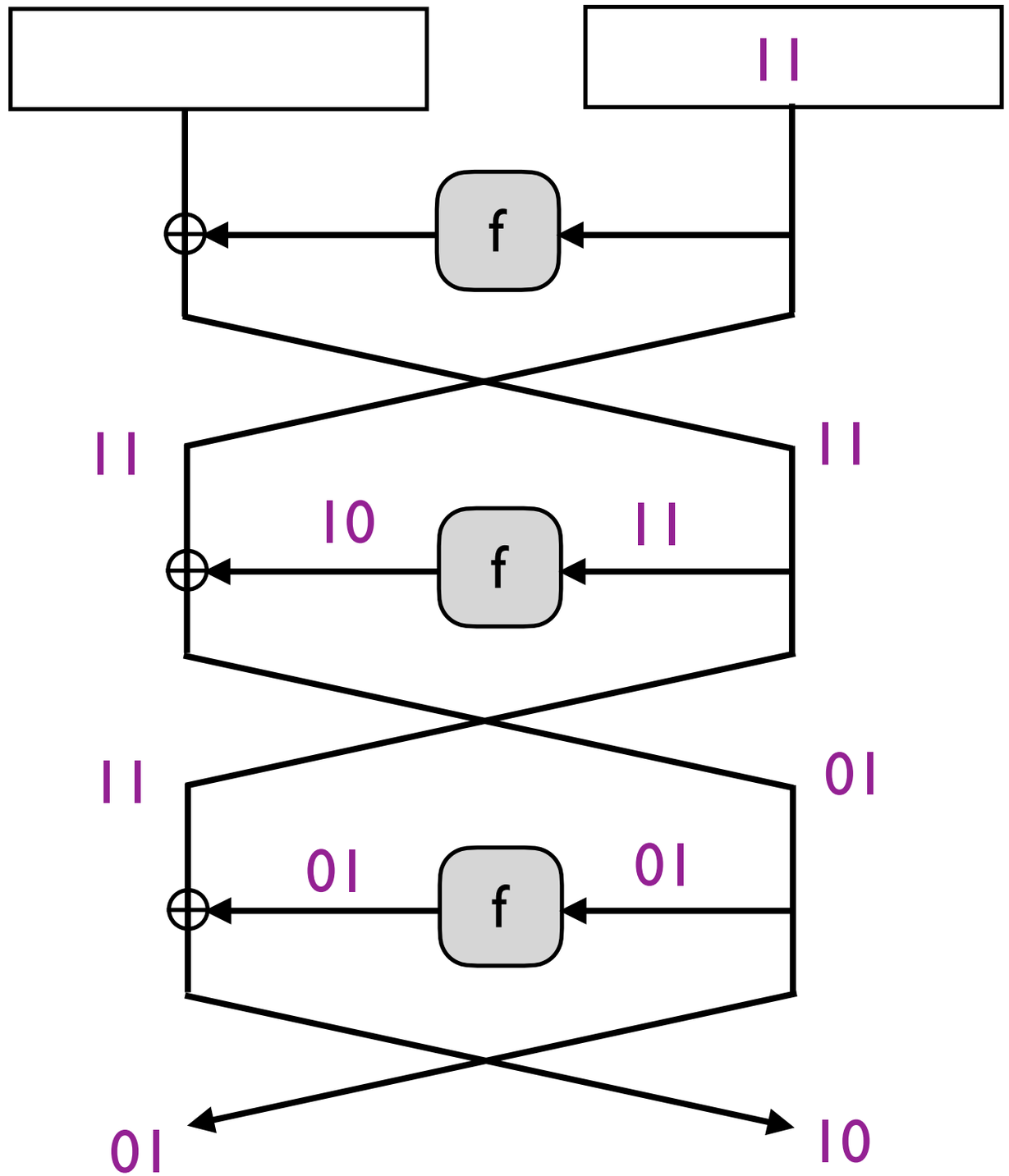
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

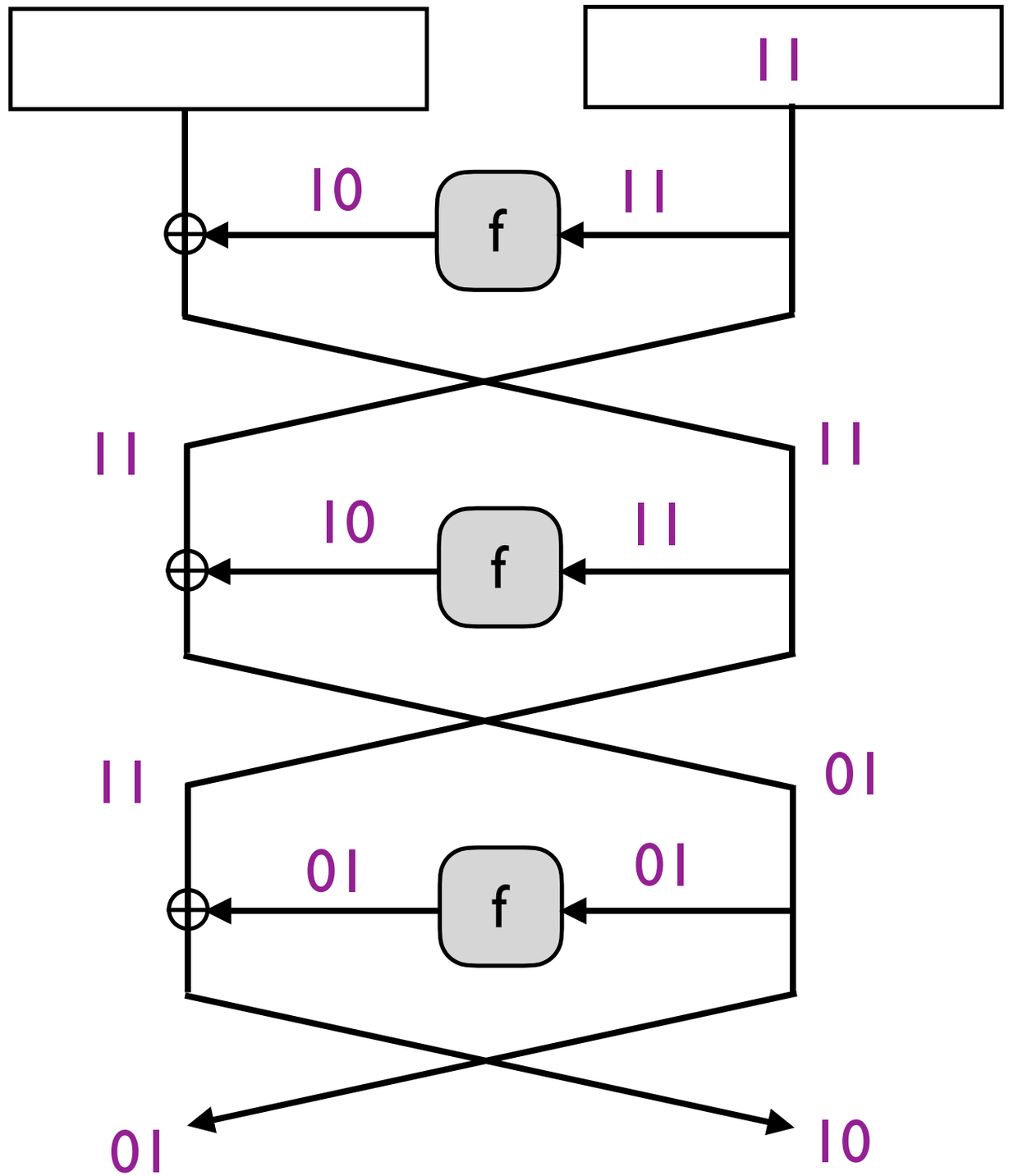
Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:

$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



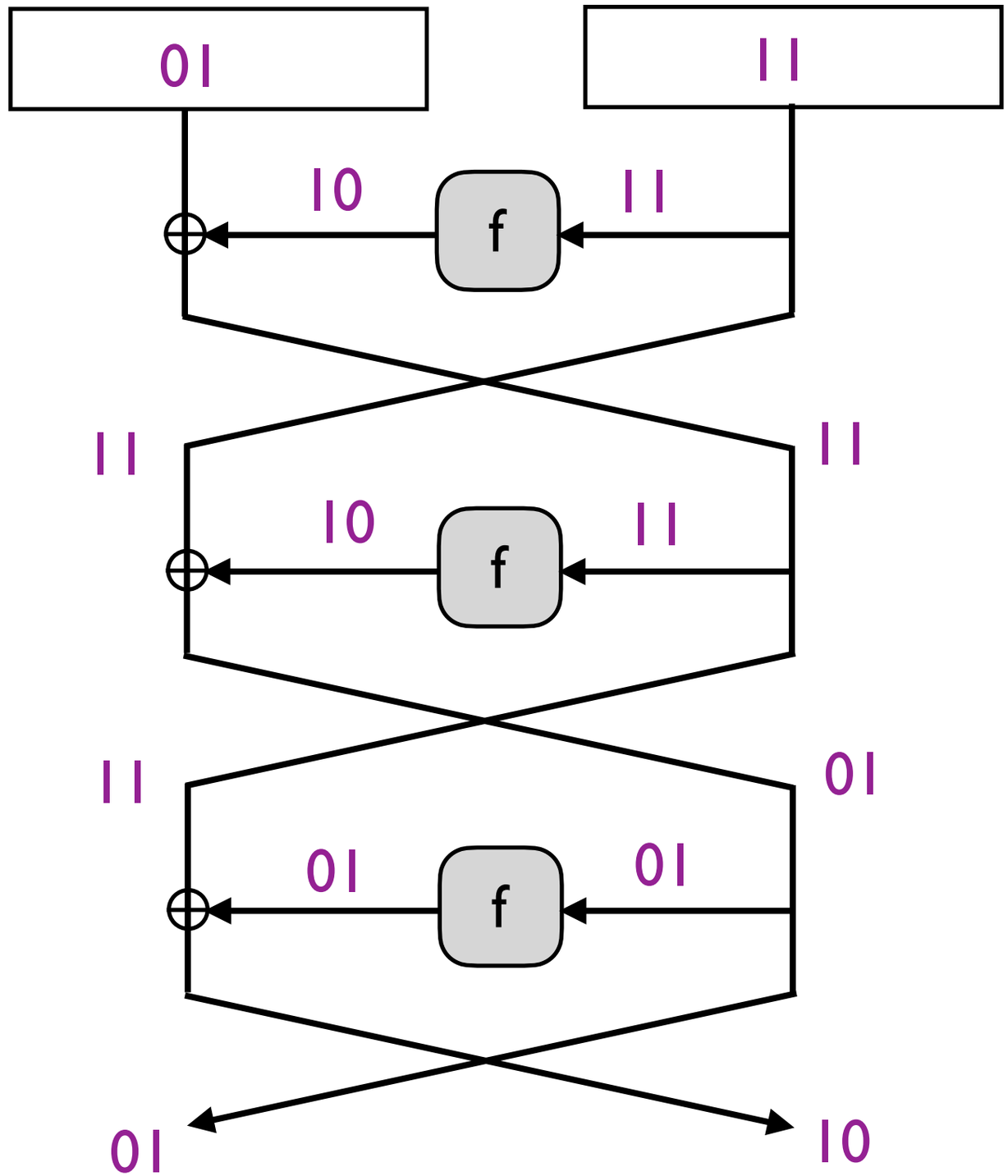
This class is being recorded

Feistel Network Example

Let's try an example, using 3 rounds on a 4-bit input (2 bits on each side).

The round function f will always be the same:
$f(x, y) = (x \text{ AND } y, 1 \oplus x)$

Input	Output
(0,0)	(0,1)
(0,1)	(0,1)
(1,0)	(0,0)
(1,1)	(1,0)



This class is being recorded

DES Subkeys

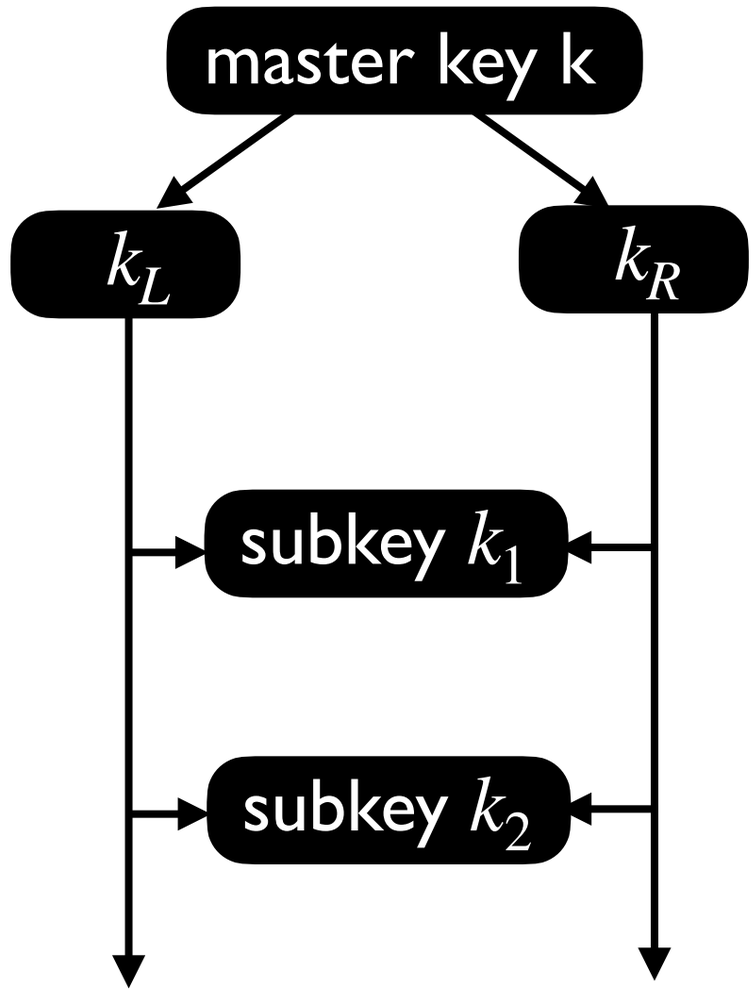
In DES, the master key k is 56 bits.

Each subkey k_i is derived from k via a key schedule:

The master key is then split into two 28-bit halves k_L and k_R.

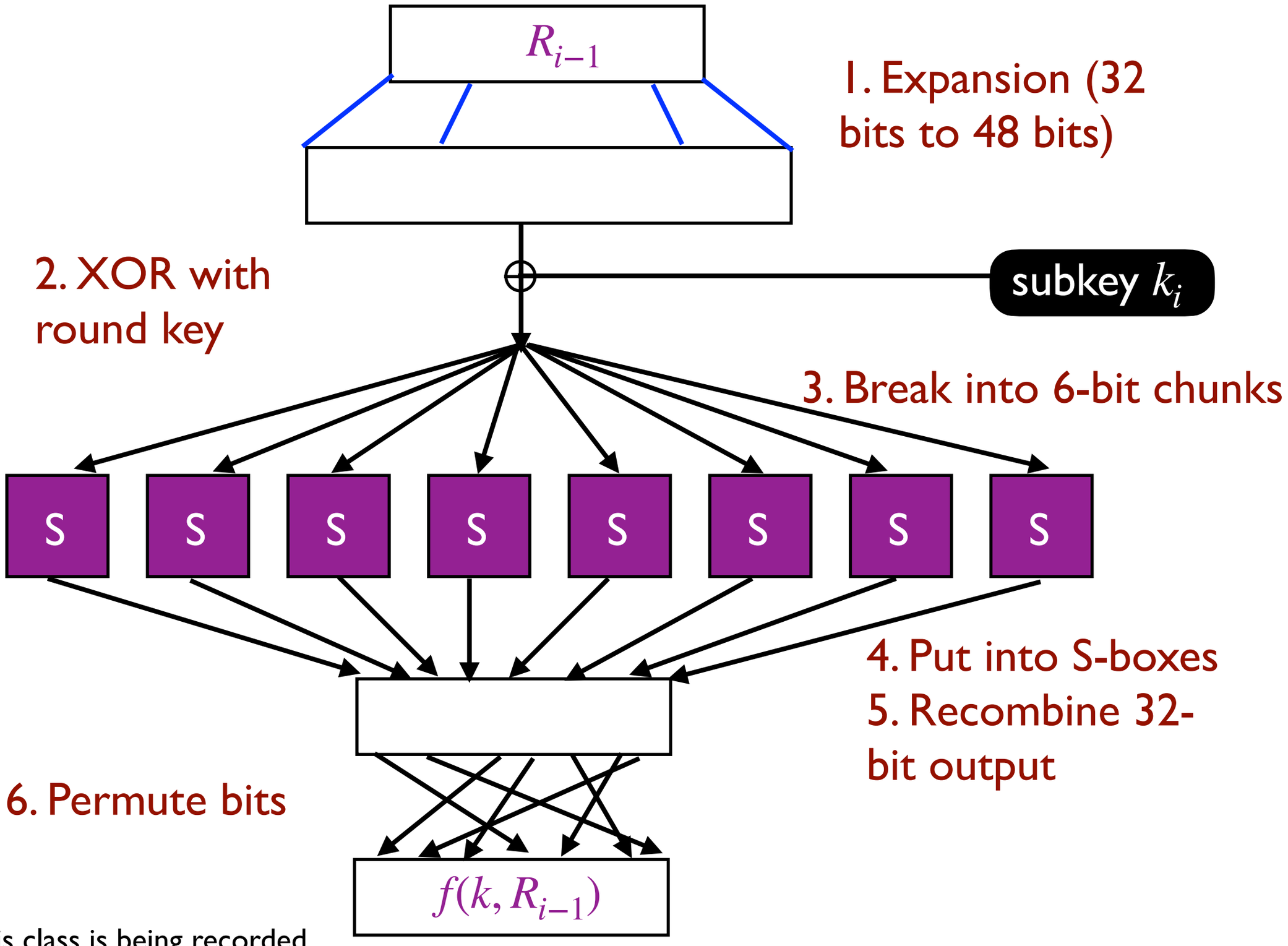
Each subkey k_i is then a permutation of 24 bits from k_L and 24 bits from k_R.

The choice of key bits used is rotated each round so that all key bits are used about the same number of times.



This class is being recorded

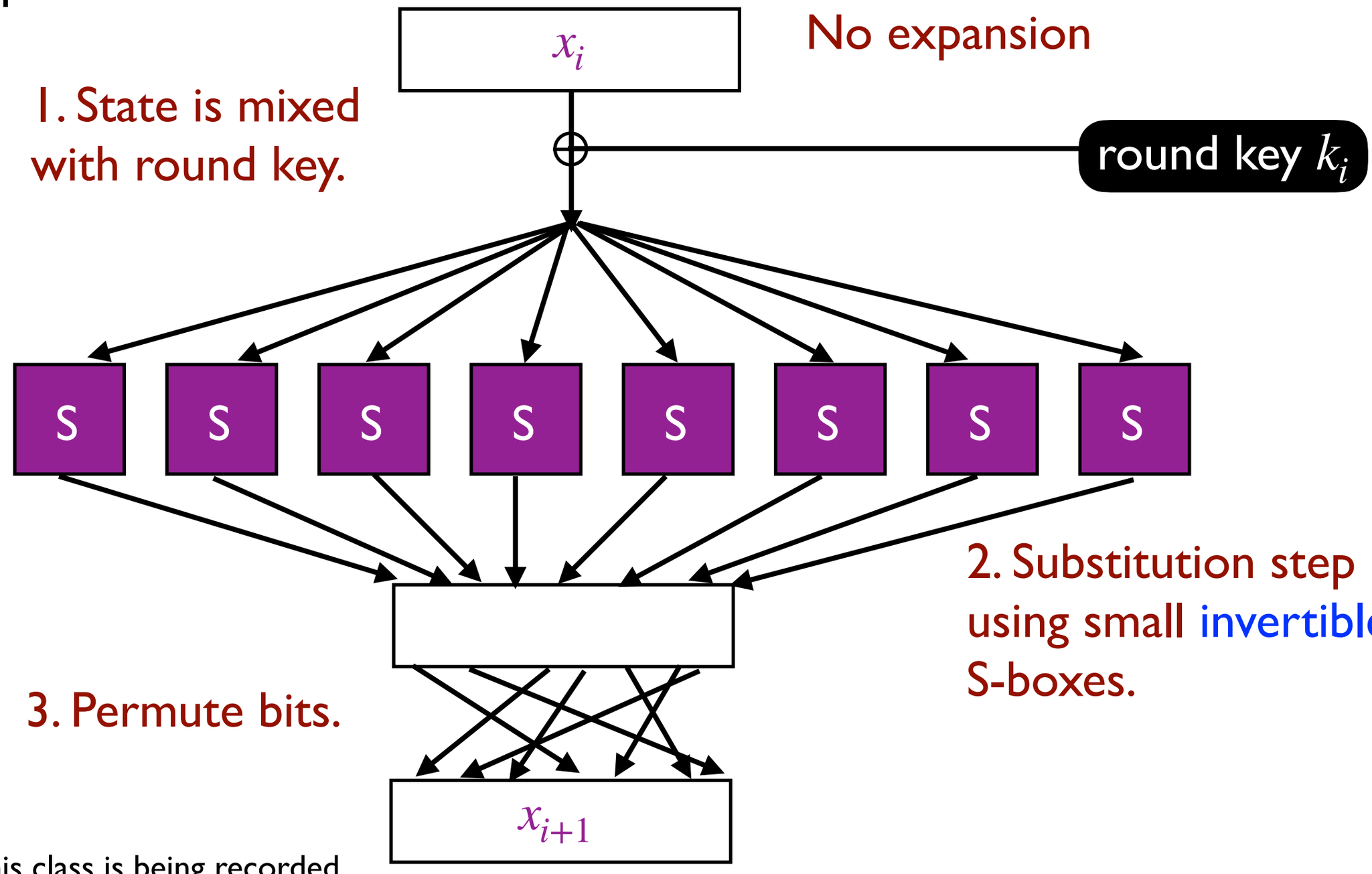
DES Mangler Function



This class is being recorded

Substitution-Permutation Networks

The DES mangler function is a variant of a substitution-permutation network, a design paradigm for pseudorandom permutations.



This class is being recorded

Confusion-Diffusion

The S-boxes introduce confusion: They change their inputs into totally different strings and magnify single-bit changes. However, the S-box is small and acts on only a few bits, so the confusion is only local.

Then the permutation step causes diffusion: whatever local confusion was introduced by the S-boxes spreads out to many different locations.

Multiple rounds of substitution and permutation cause the confusion to be magnified further and continue to spread around.

We need both to get an avalanche effect.

You also need key mixing: This is a permutation, and without the key, Eve can just trace the permutation backwards to get the input.

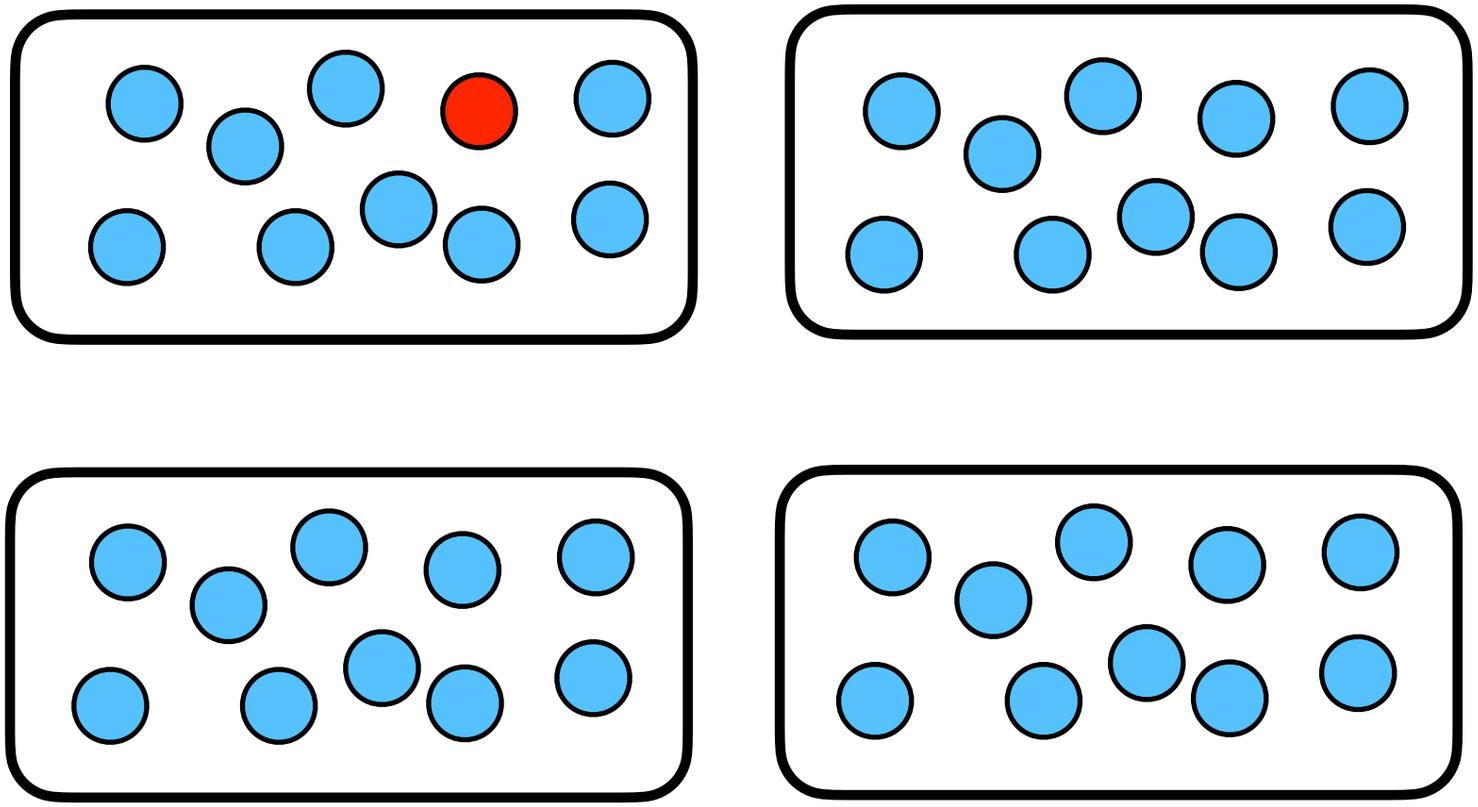
This class is being recorded

Disease Confusion-Diffusion

Imagine you have a disease spreading. It starts with one patient.

Confusion: The disease infects additional people in the same city as someone who is sick.

Diffusion: Some people travel to different cities.

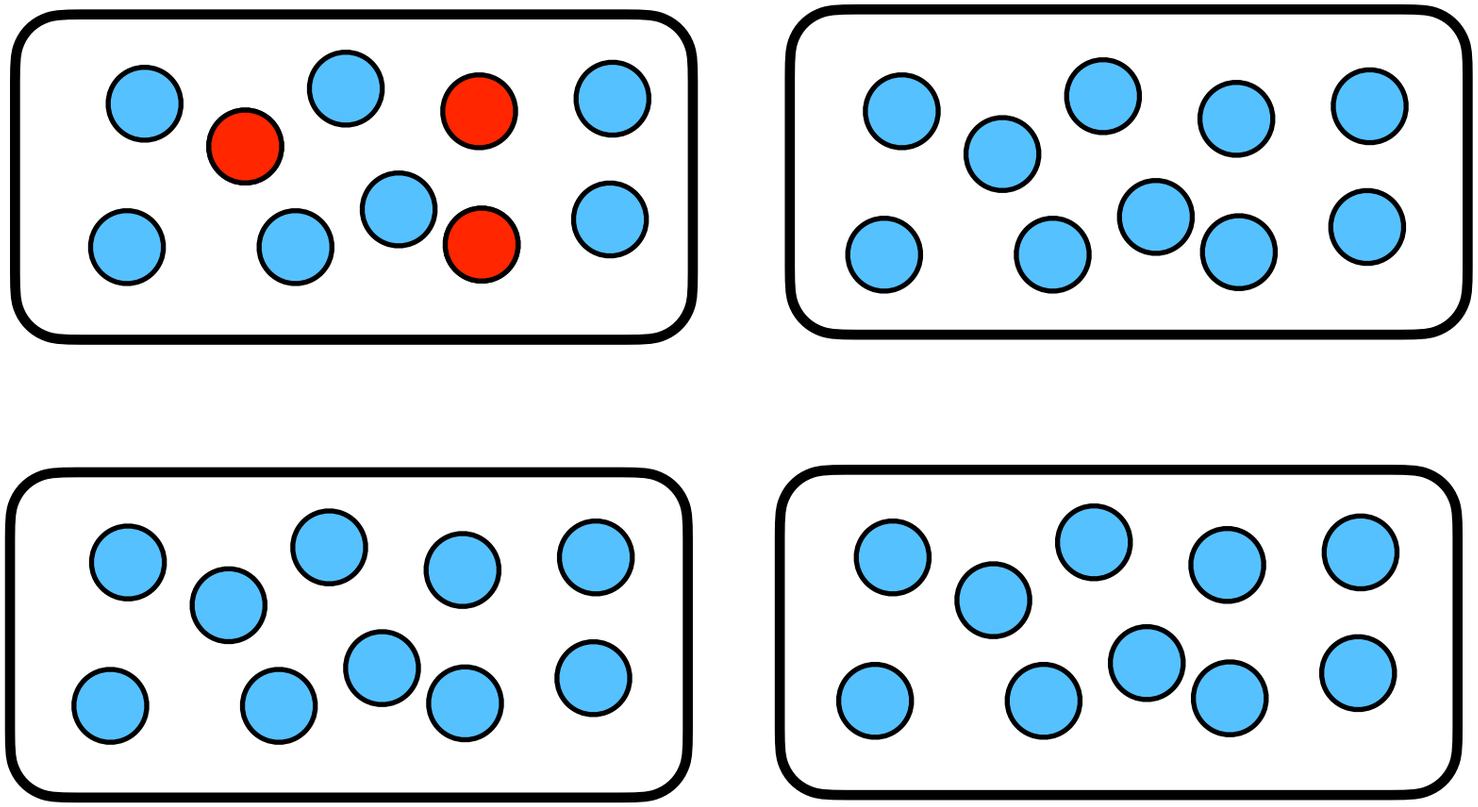


This class is being recorded

Disease Confusion-Diffusion

Imagine you have a disease spreading. It starts with one patient.

Confusion: The disease infects additional people in the same city as someone who is sick.
Diffusion: Some people travel to different cities.



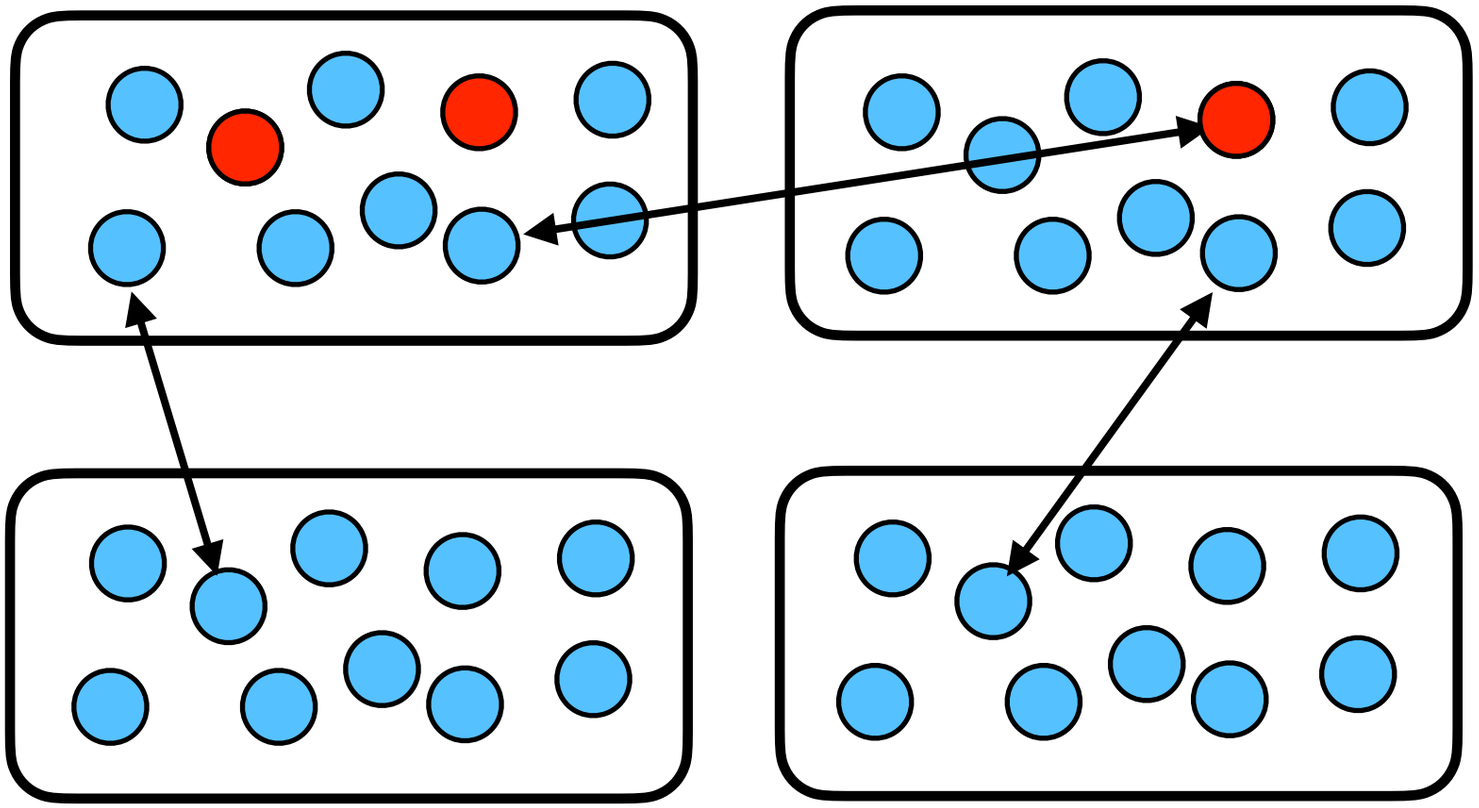
1. Confusion

This class is being recorded

Disease Confusion-Diffusion

Imagine you have a disease spreading. It starts with one patient.

Confusion: The disease infects additional people in the same city as someone who is sick.
Diffusion: Some people travel to different cities.



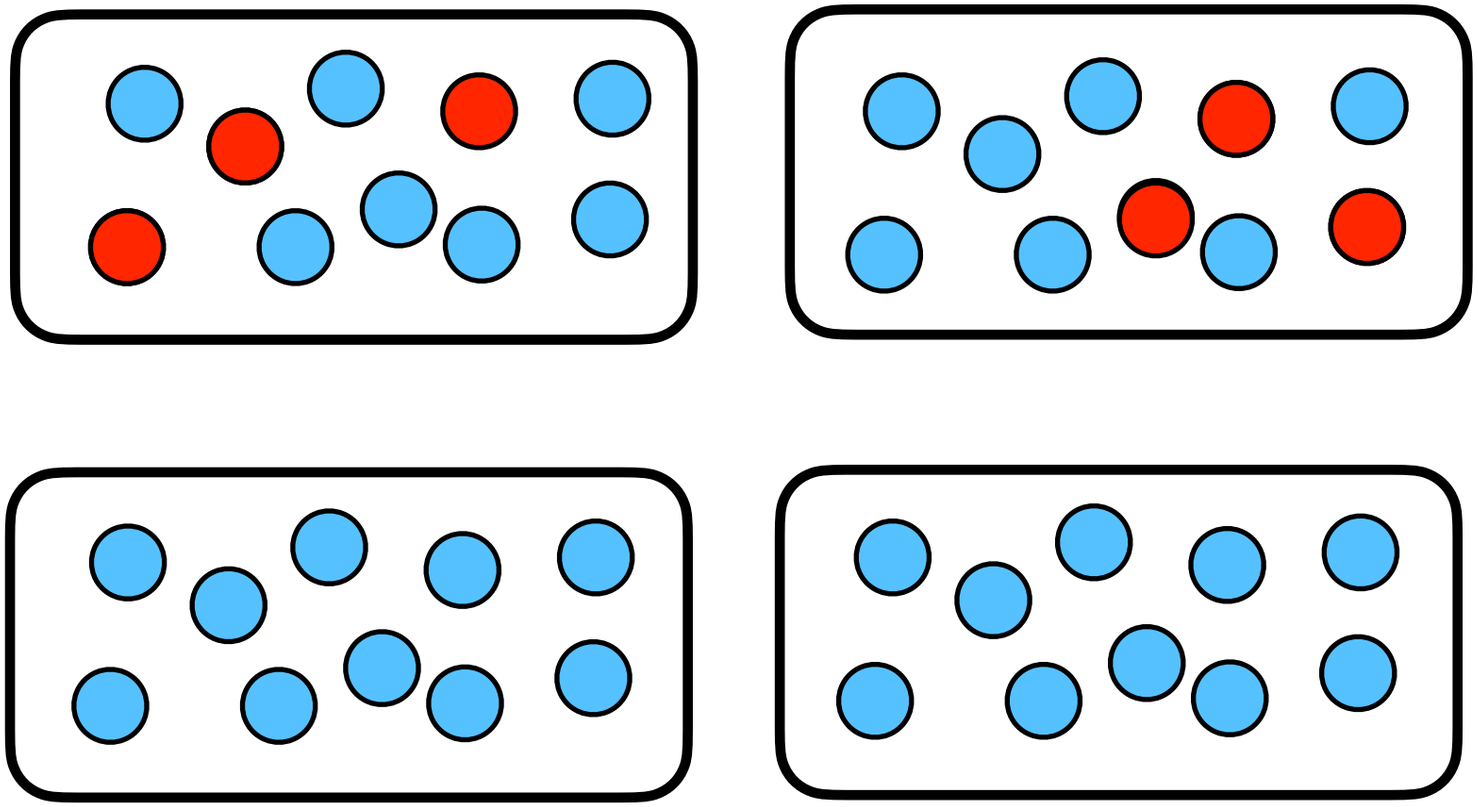
1. Confusion
2. Diffusion

This class is being recorded

Disease Confusion-Diffusion

Imagine you have a disease spreading. It starts with one patient.

Confusion: The disease infects additional people in the same city as someone who is sick.
Diffusion: Some people travel to different cities.



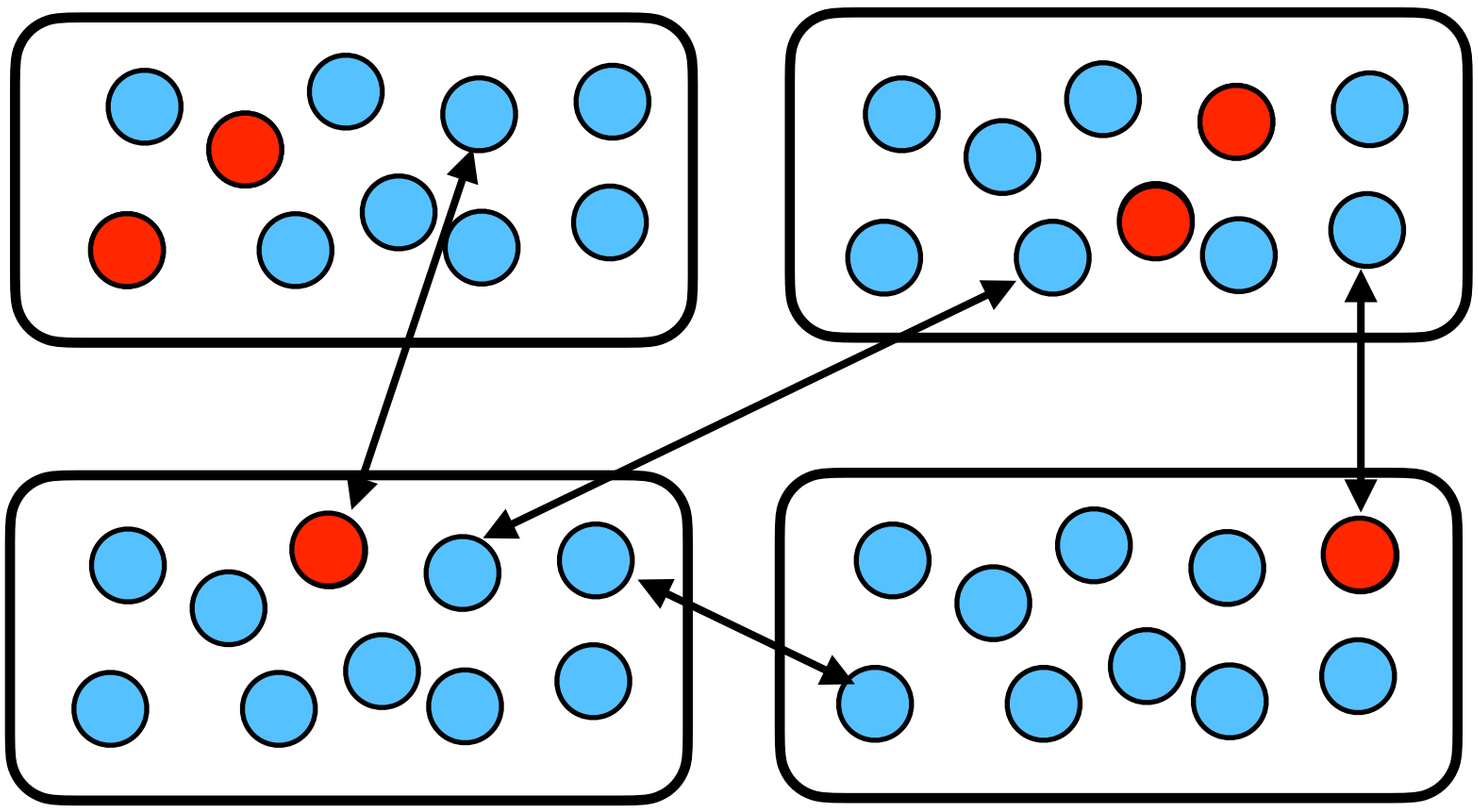
1. Confusion
2. Diffusion
3. Confusion

This class is being recorded

Disease Confusion-Diffusion

Imagine you have a disease spreading. It starts with one patient.

Confusion: The disease infects additional people in the same city as someone who is sick.
Diffusion: Some people travel to different cities.

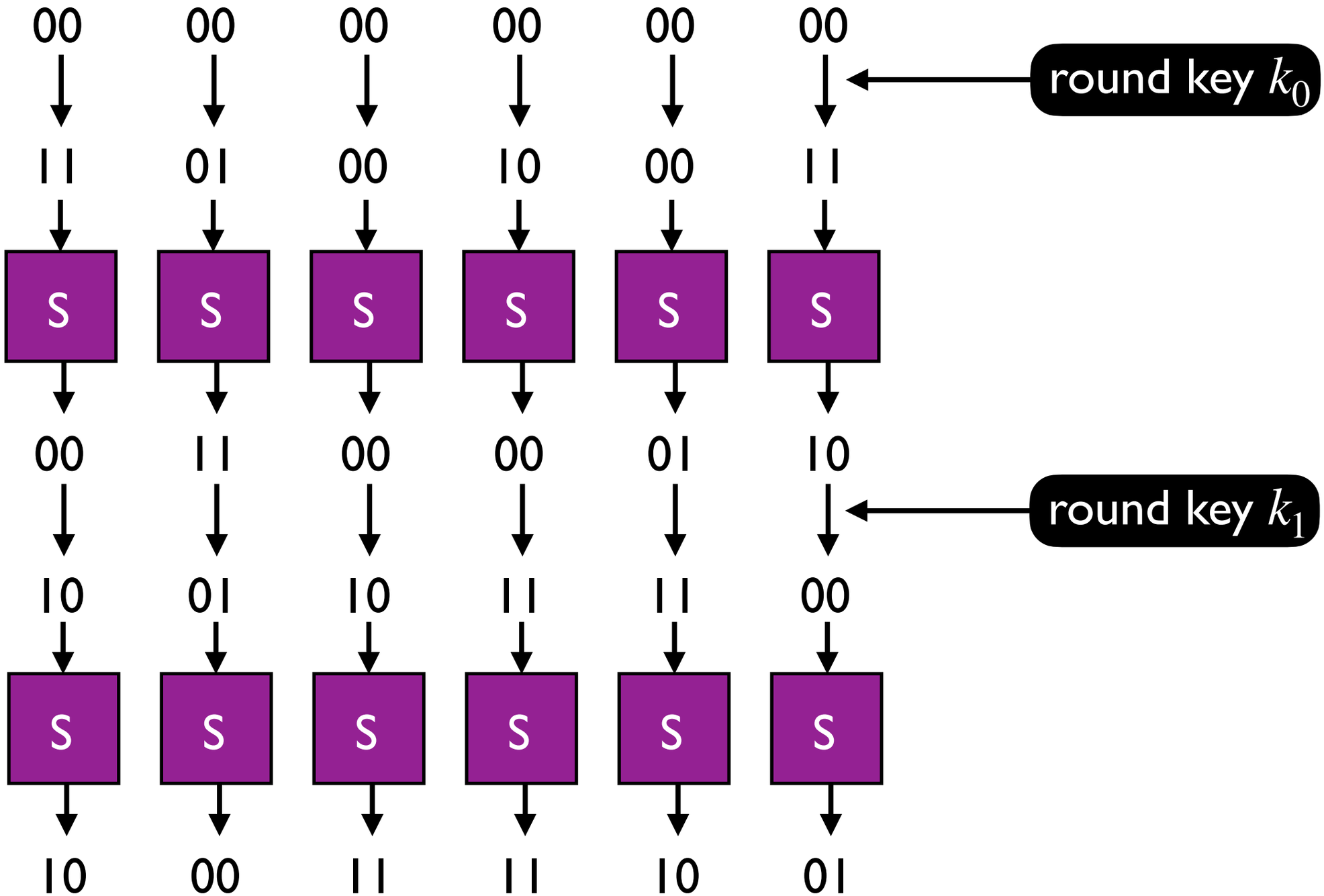


1. Confusion
2. Diffusion
3. Confusion
4. Diffusion

This class is being recorded

Substitution Only

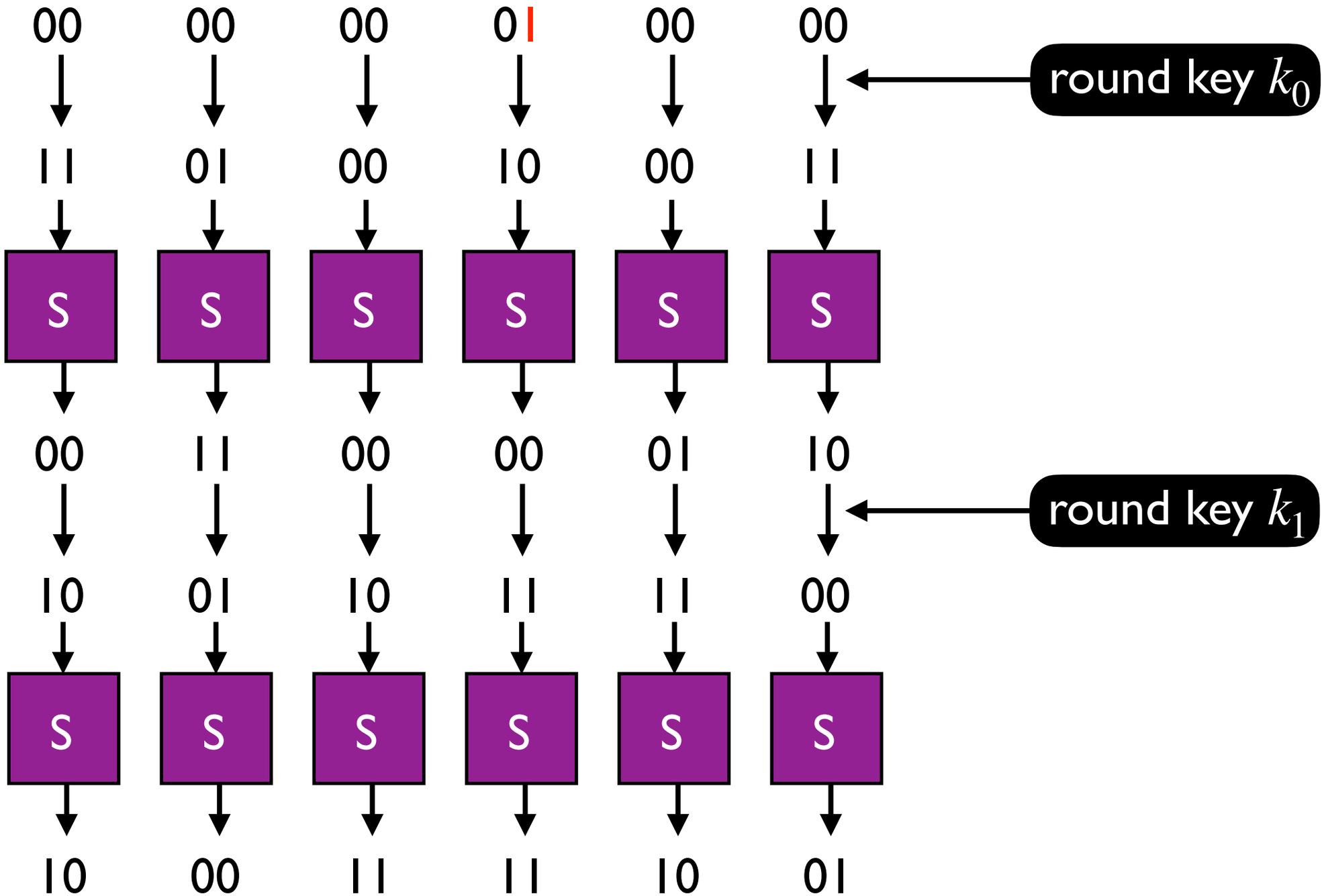
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

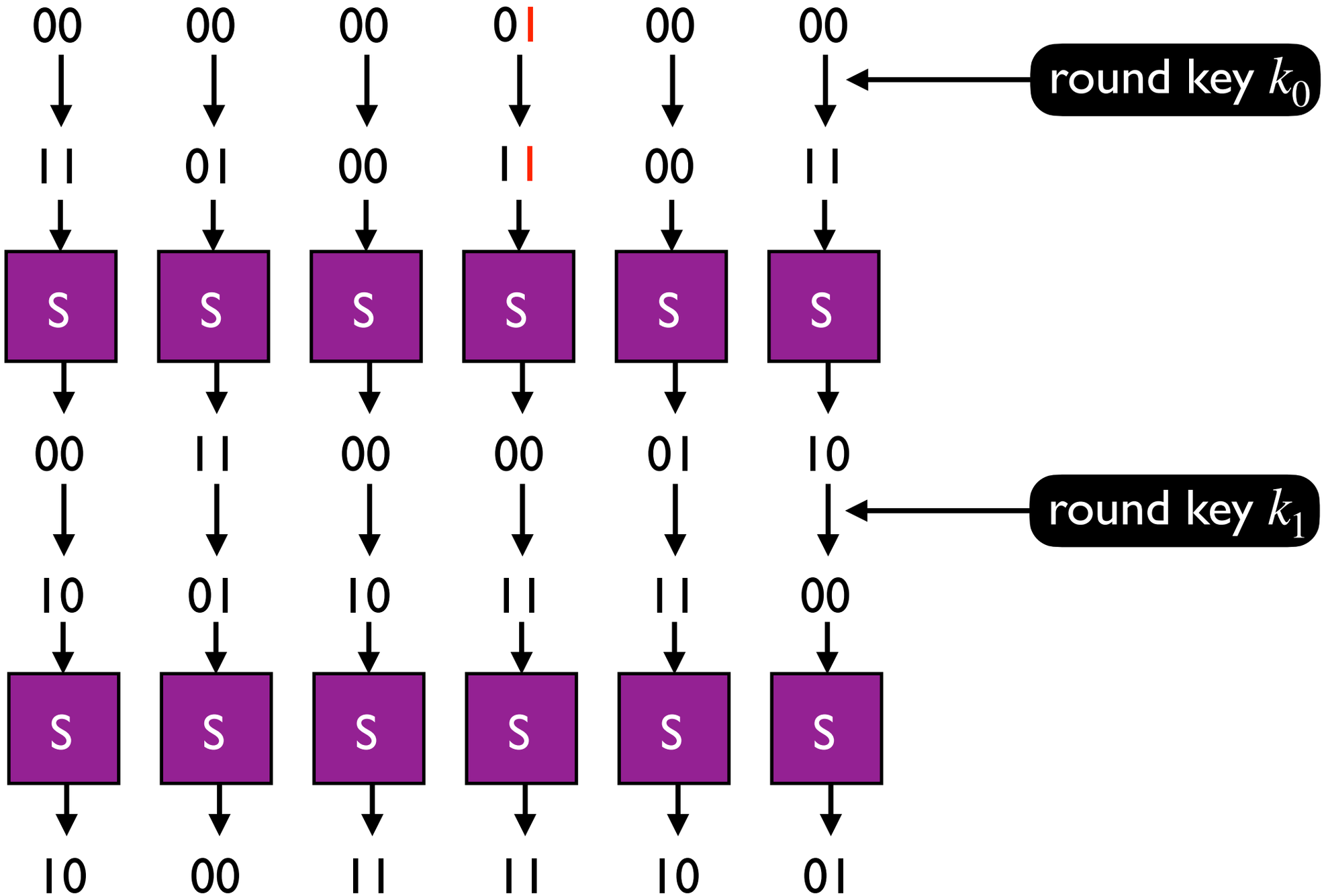
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

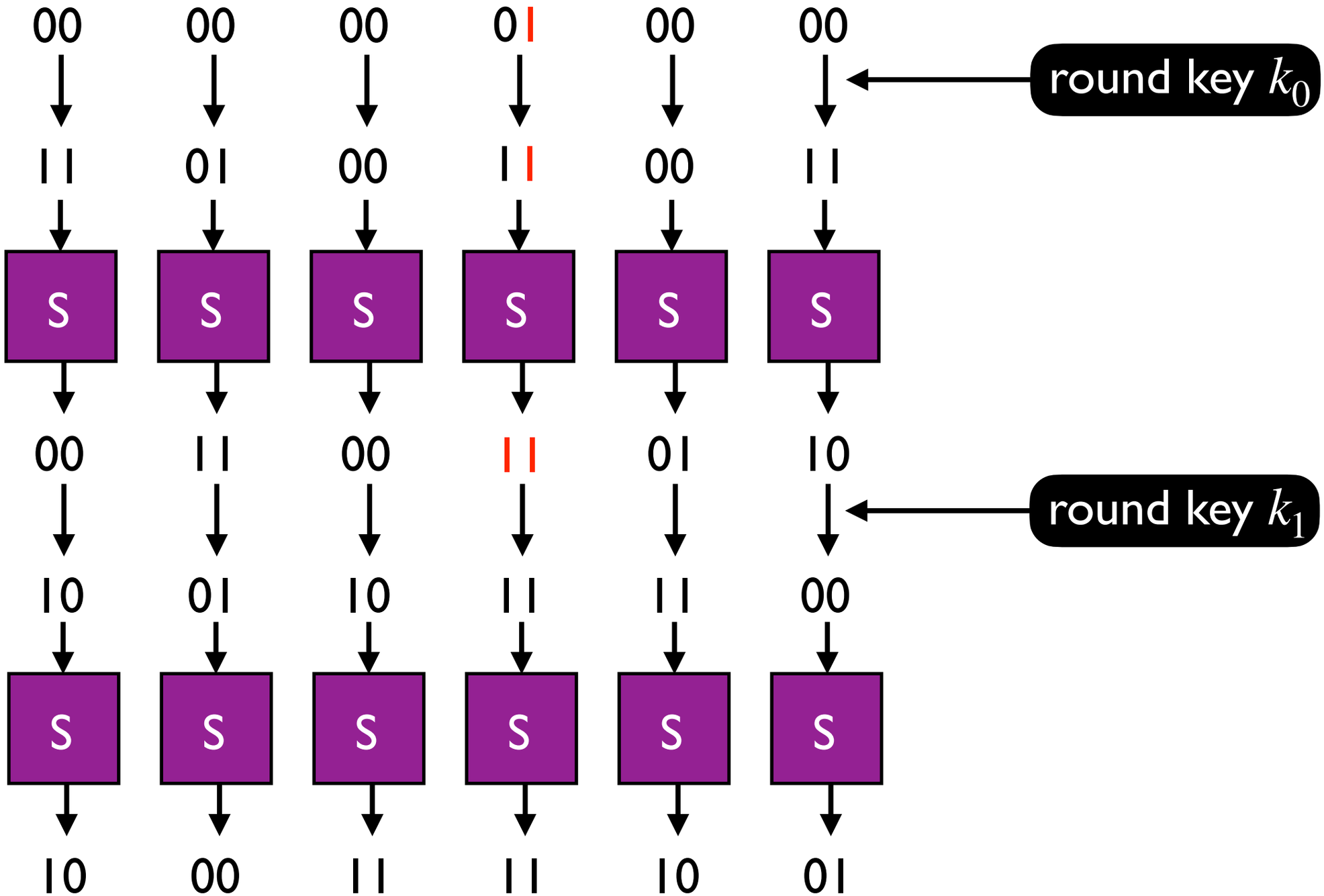
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

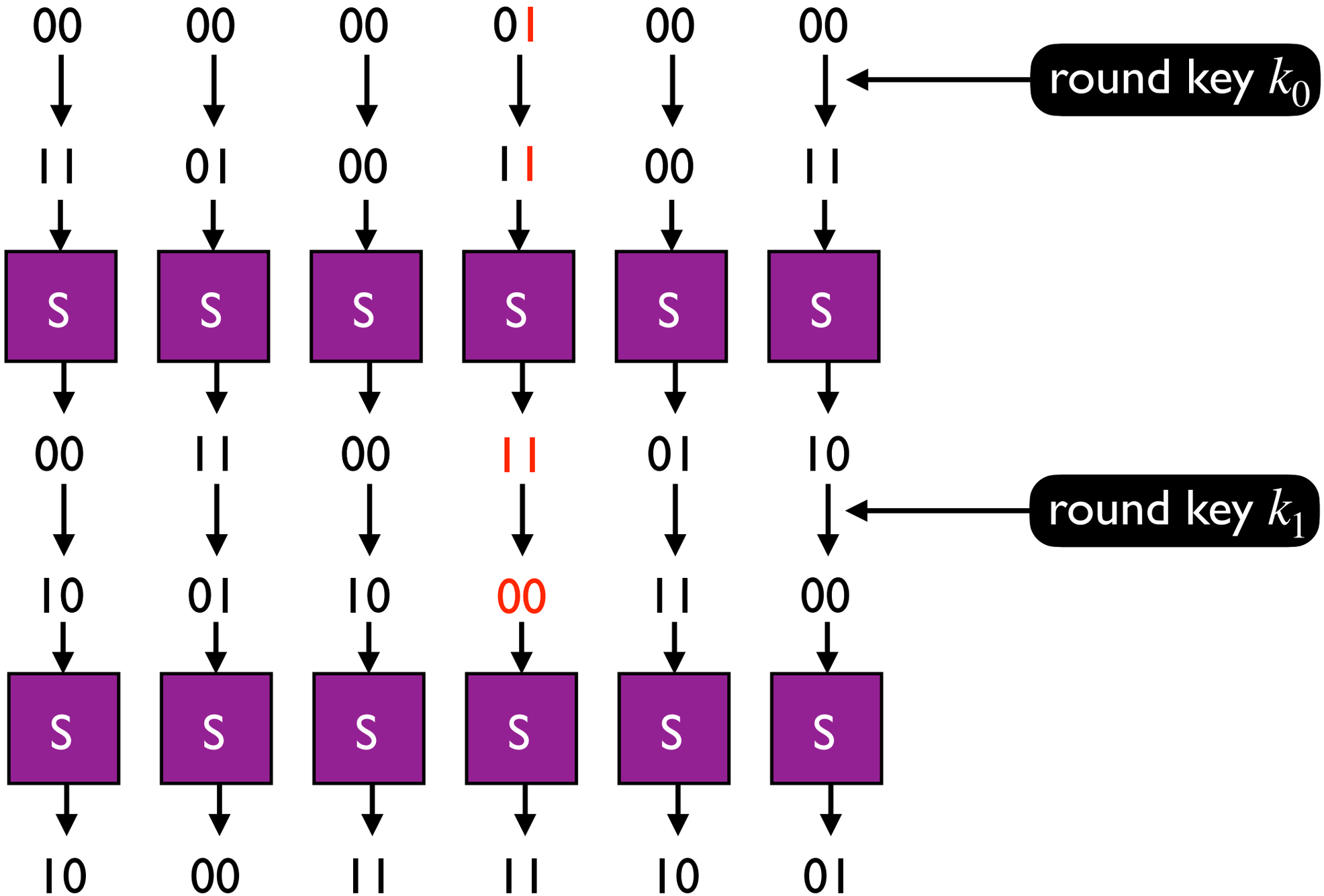
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

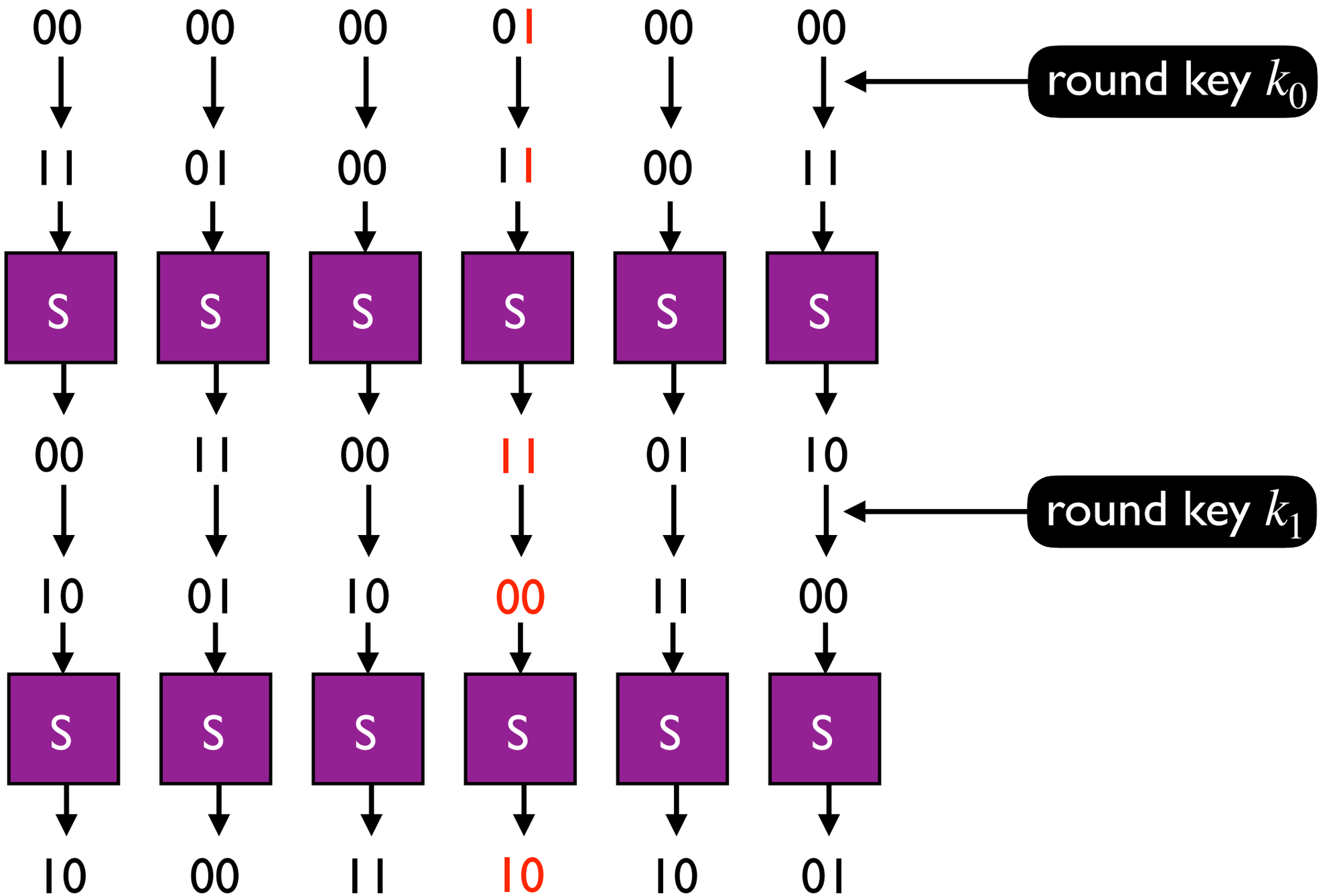
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

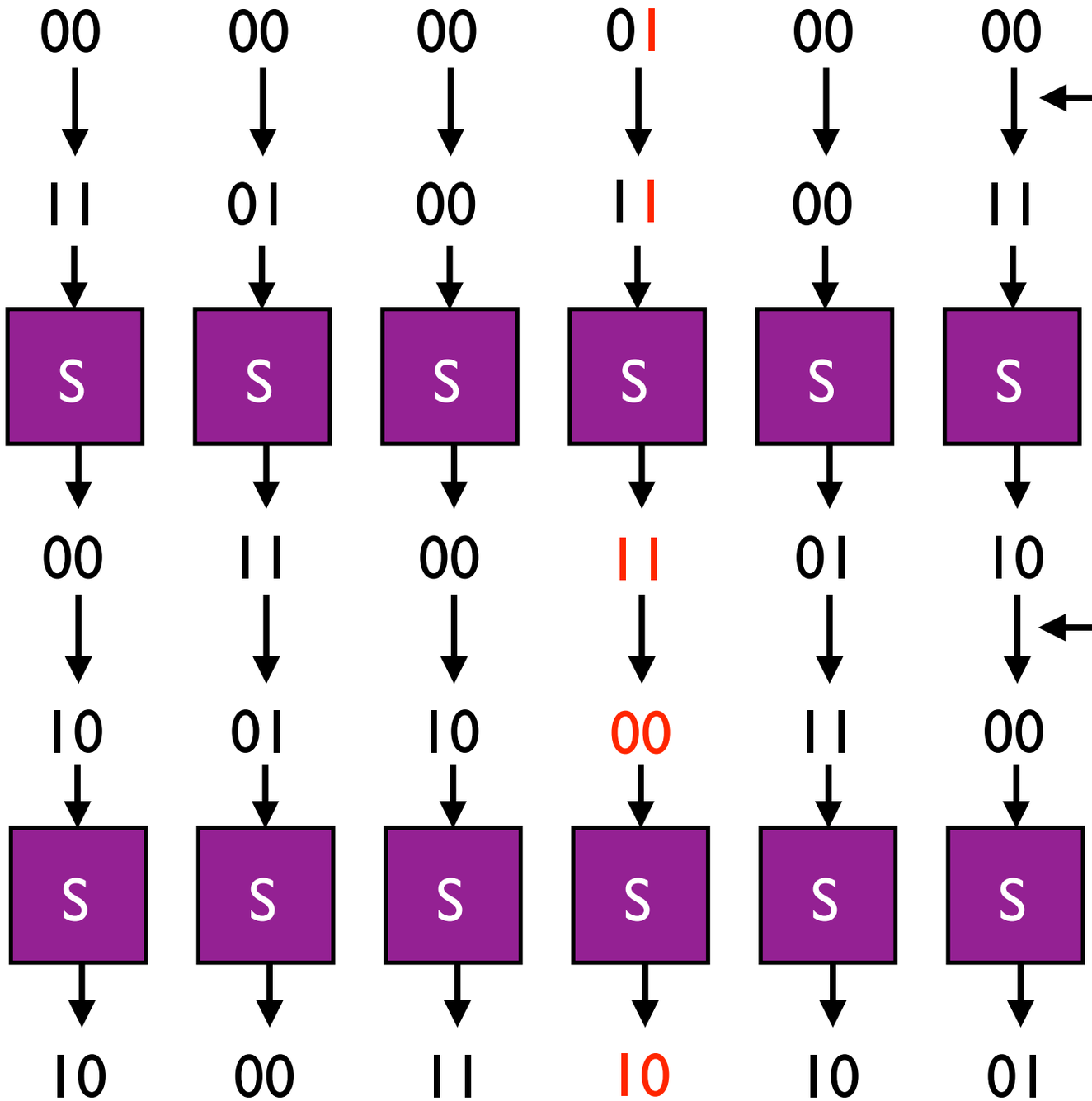
Suppose we have S-boxes but no permutation.



This class is being recorded

Substitution Only

Suppose we have S-boxes but no permutation.

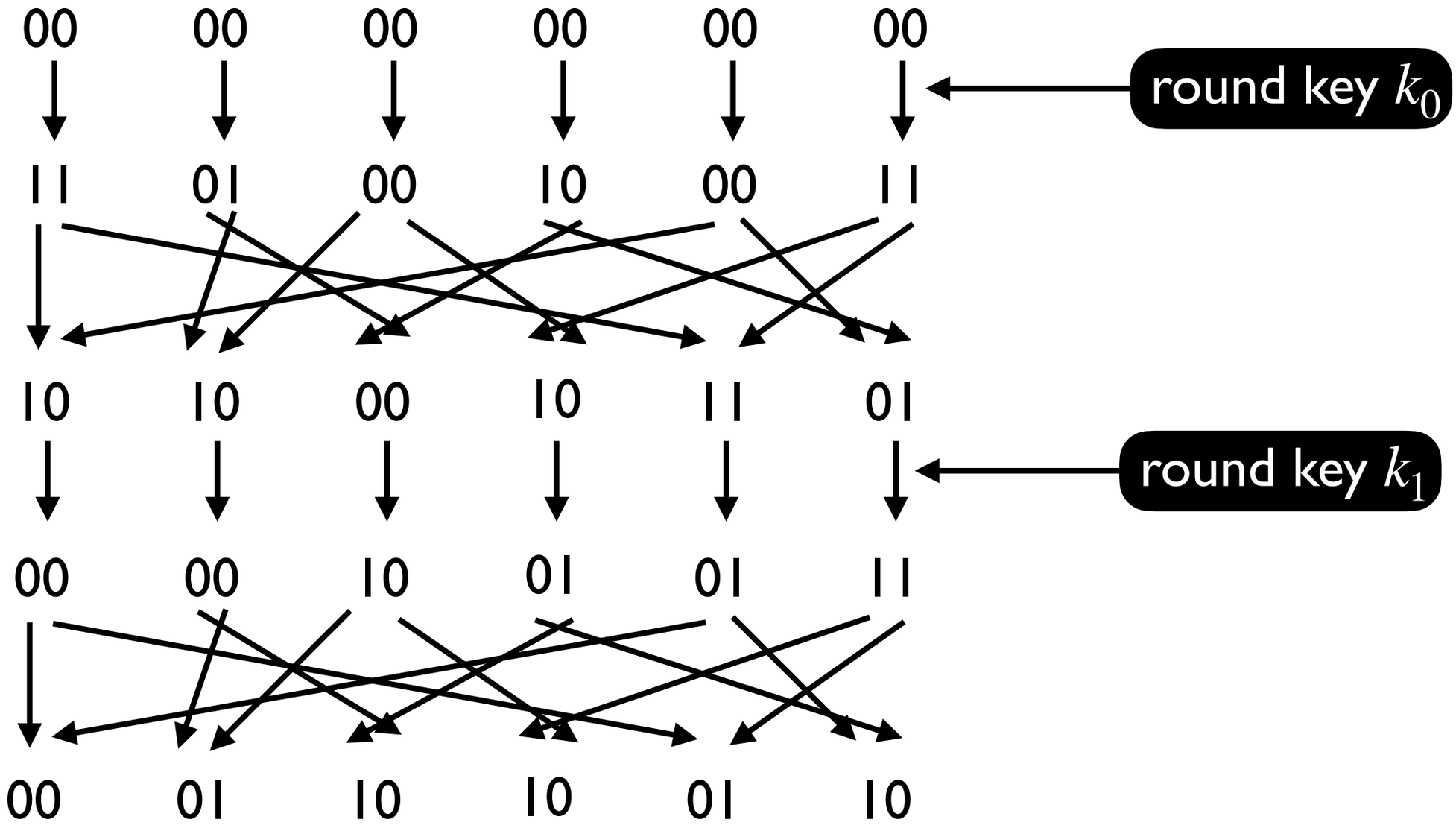


No avalanche effect!

Eve can focus on the 2 bits of key for just this S-box.

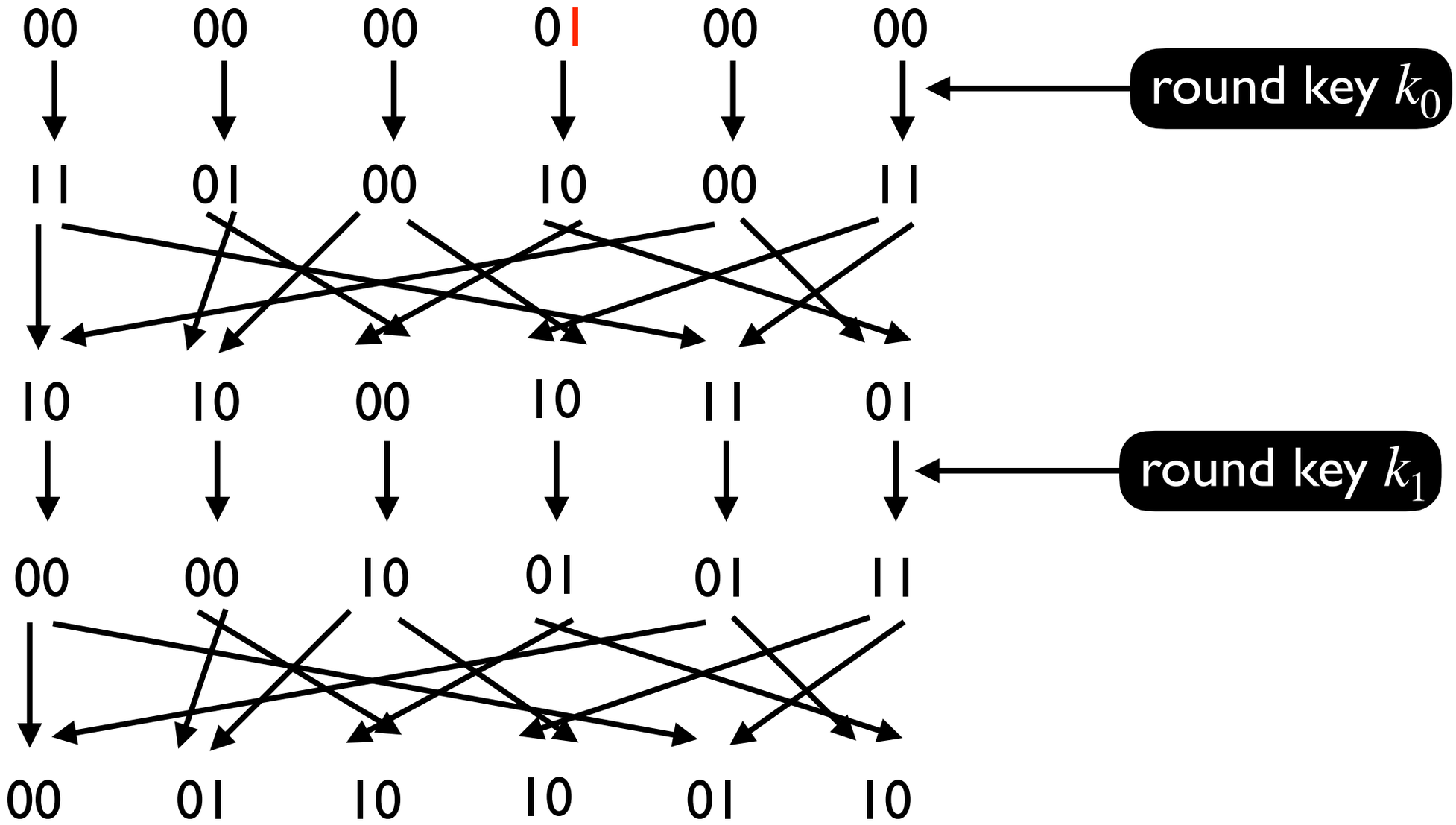
This class is being recorded

Permutation Only



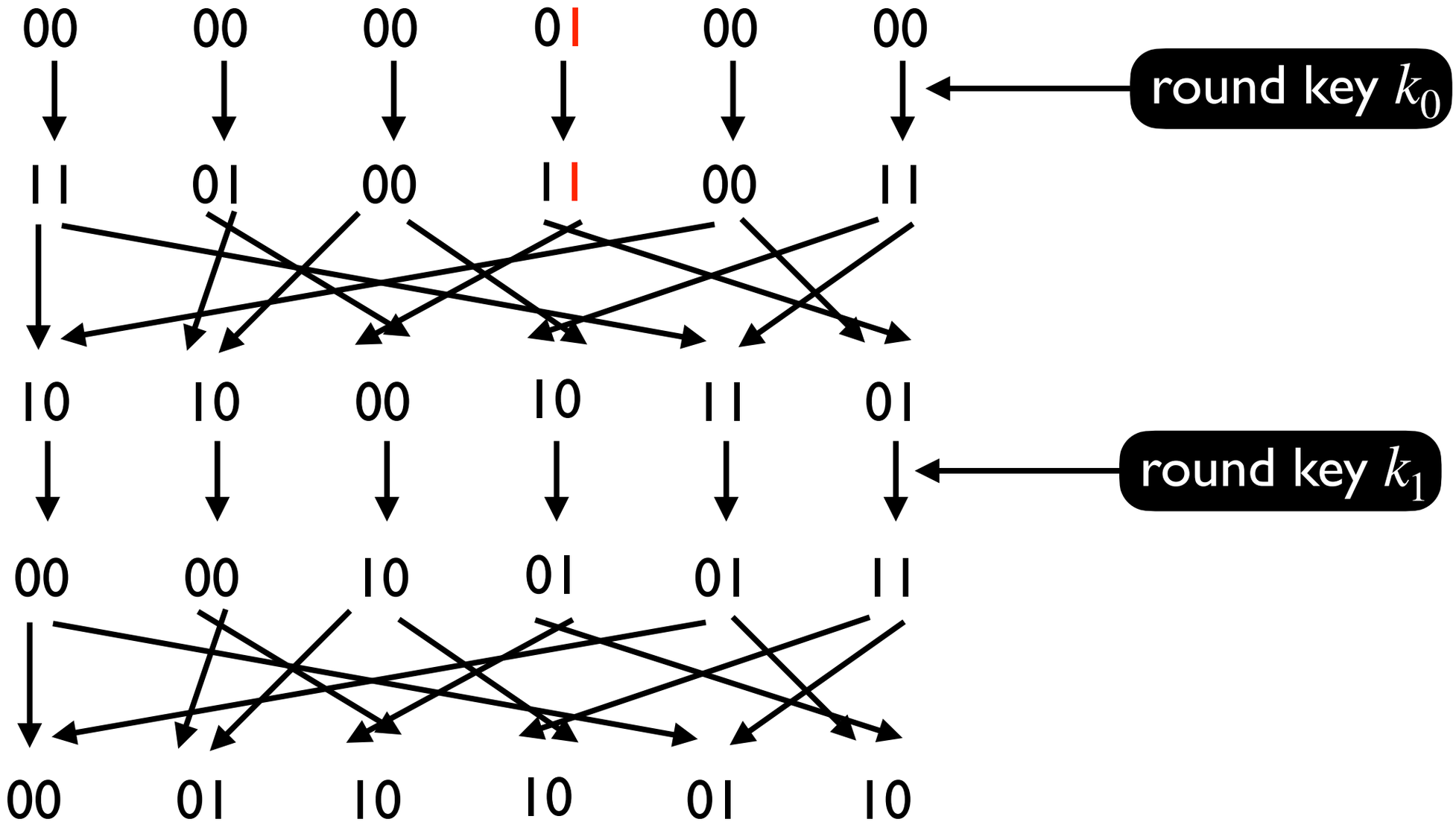
This class is being recorded

Permutation Only



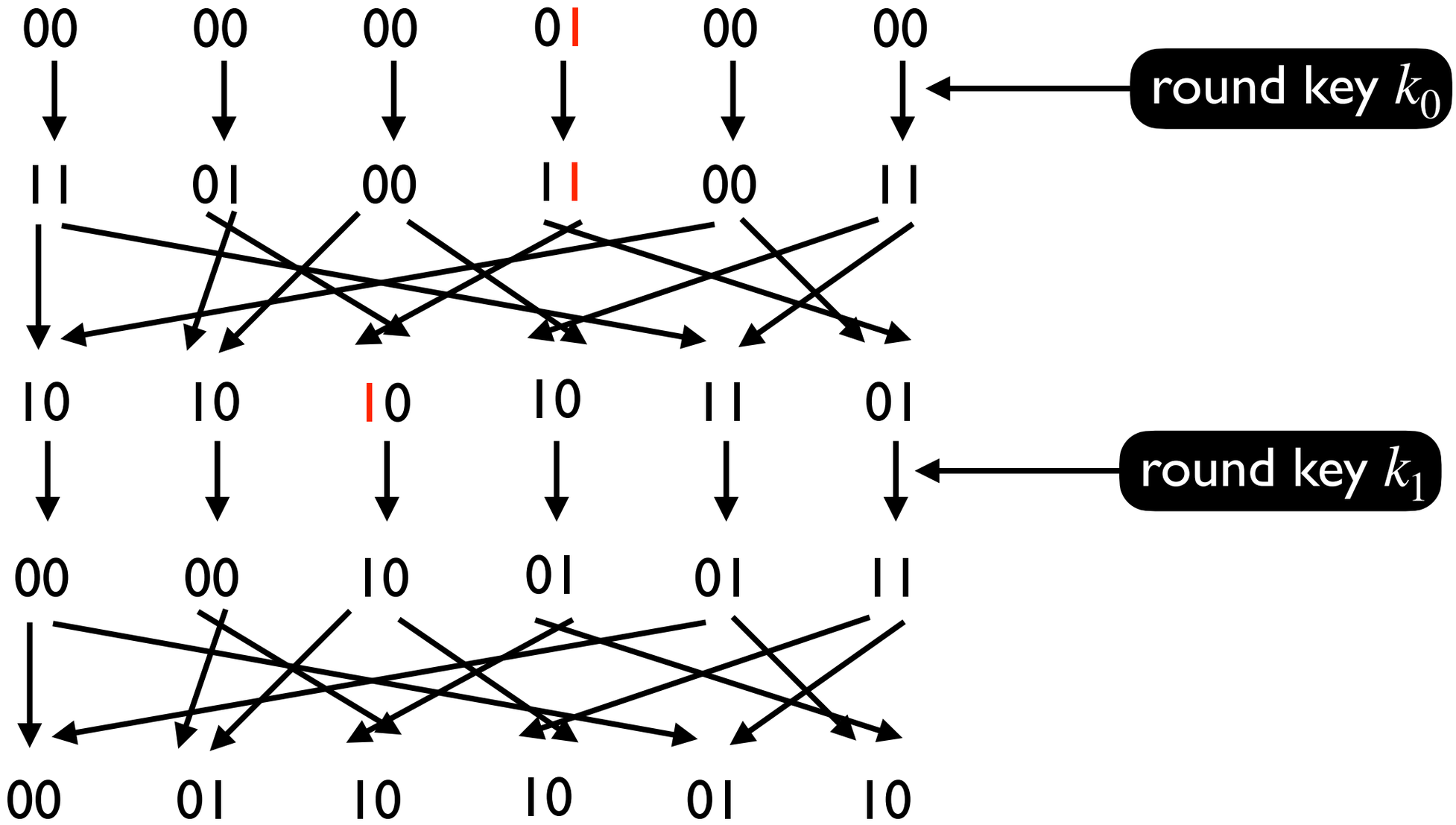
This class is being recorded

Permutation Only



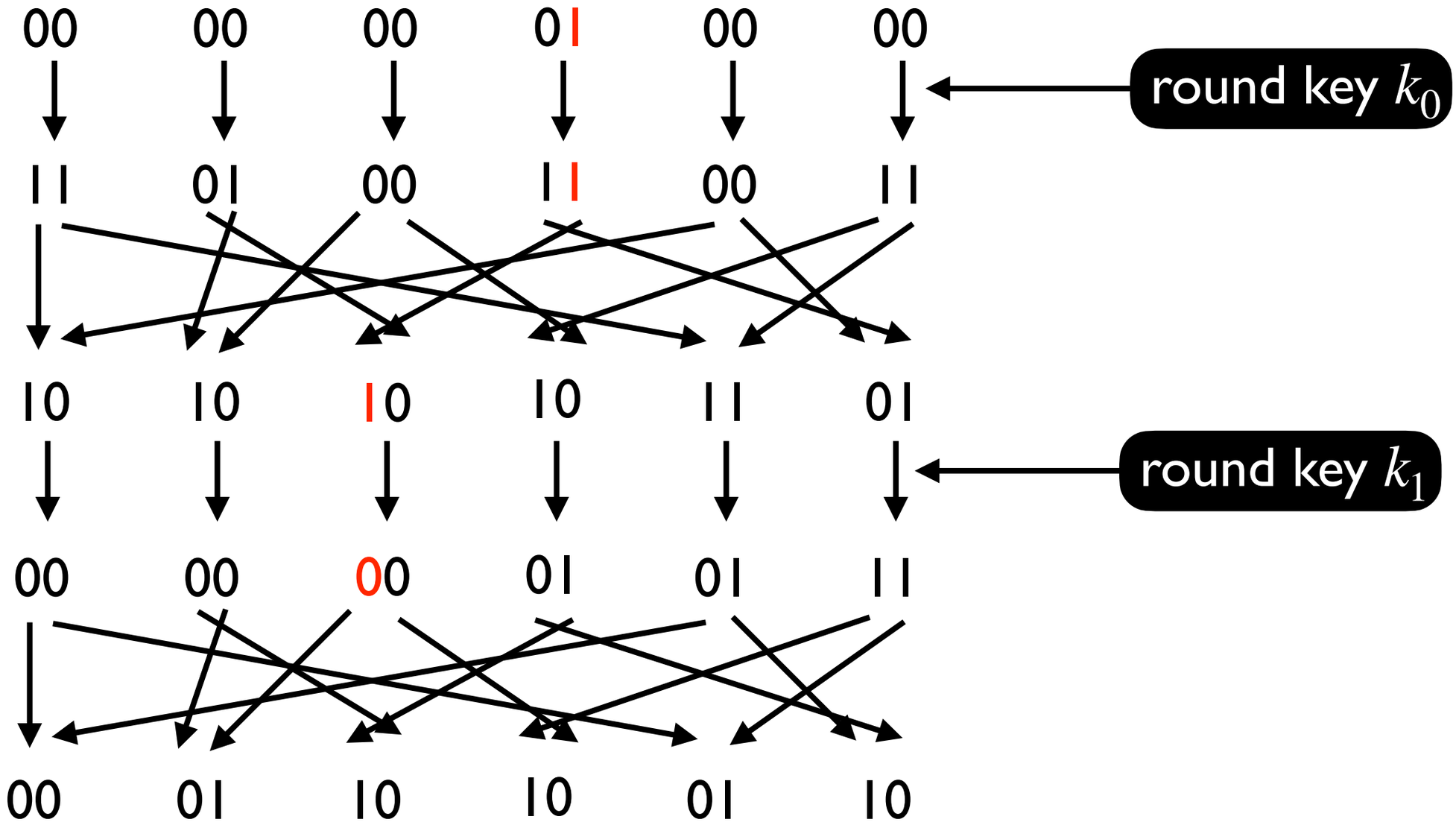
This class is being recorded

Permutation Only



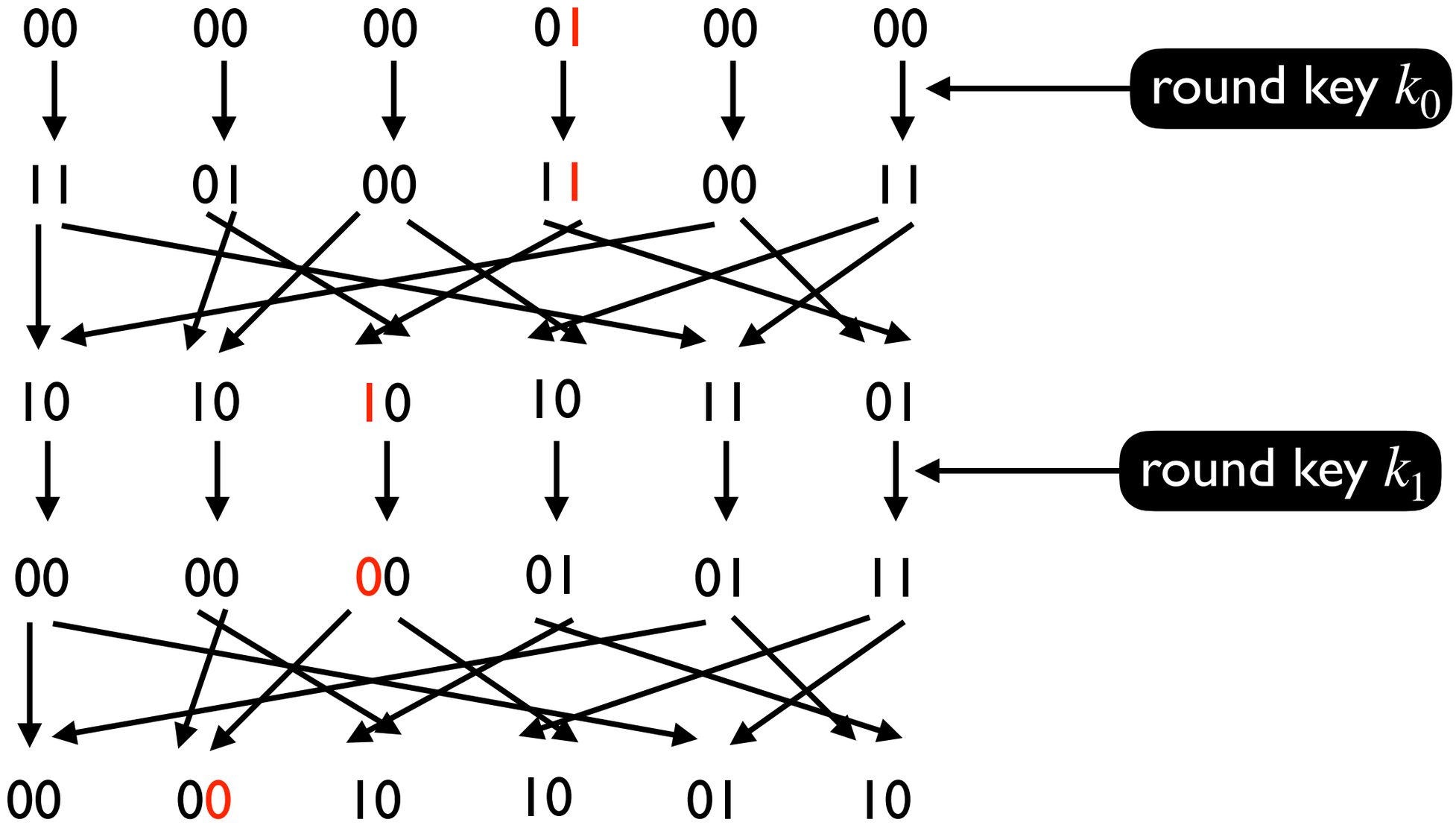
This class is being recorded

Permutation Only



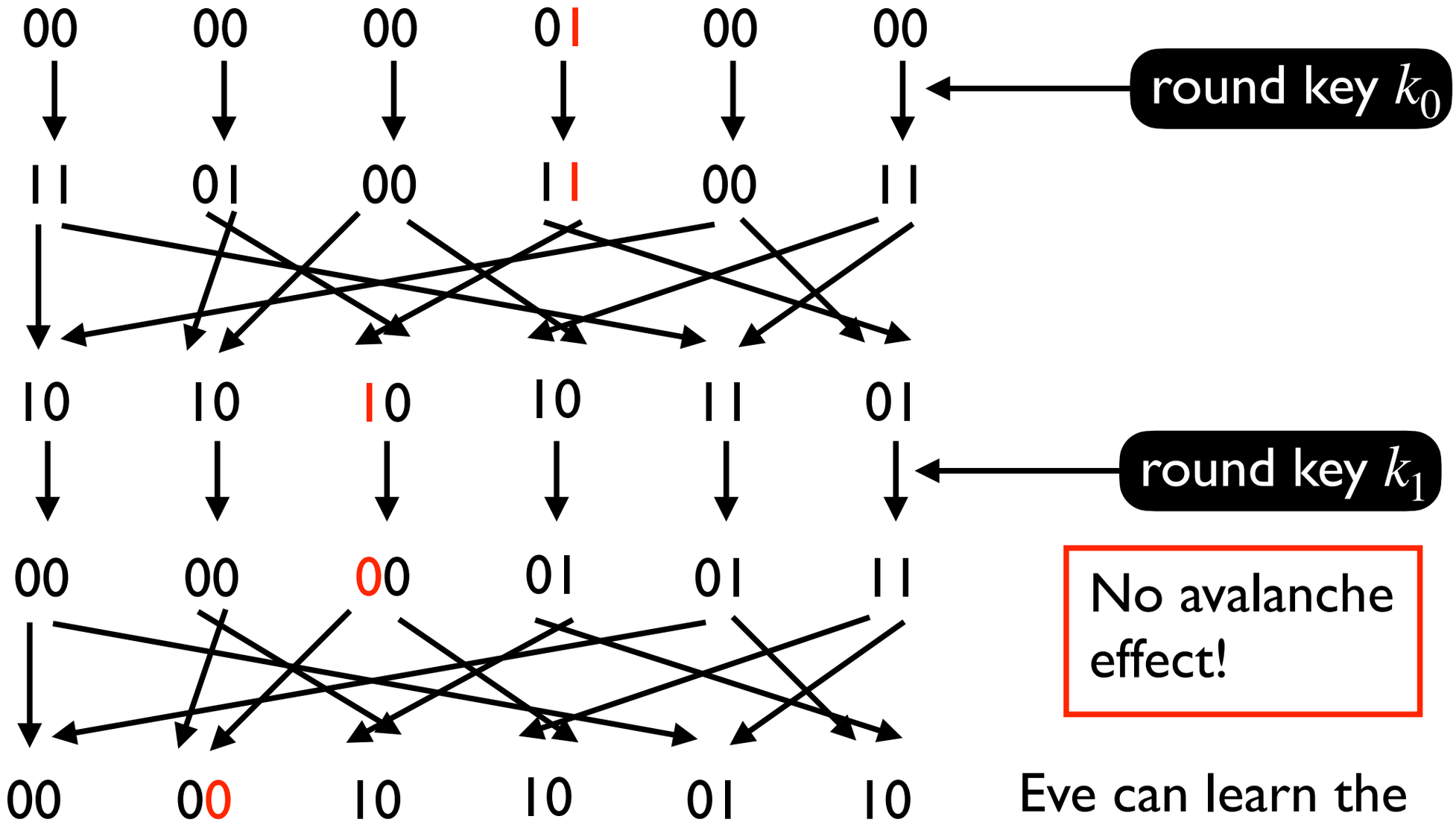
This class is being recorded

Permutation Only



This class is being recorded

Permutation Only



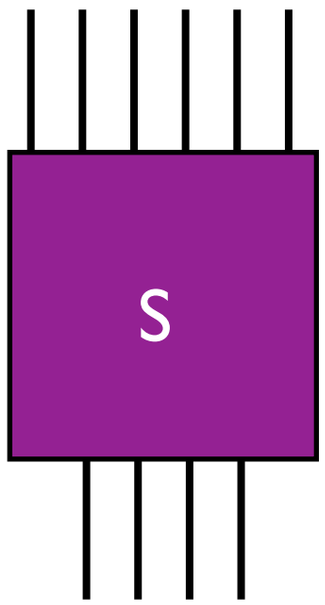
No avalanche effect!

Eve can learn the XOR of the bits of key corresponding to this path.

This class is being recorded

DES S-Boxes

Each of the 8 S-boxes in a single mangler function are different, but the set of S-boxes is the same from round to round (as are the expansion function and the final permutation).



The S-box takes 6 bits as inputs and outputs 4 bits, so the 8 S-boxes together reduce the expanded 48-bit string back to the original 32 bits.

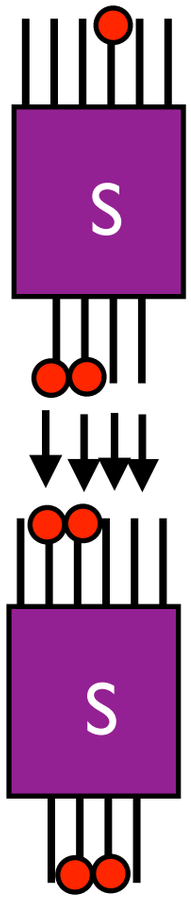
The S-boxes are carefully designed complicated functions defined by lookup tables.

They have the property that all possible outputs are equally likely on a random input (so each possible output can be reached with 4 possible inputs) and that if two possible inputs x_1 and x_2 differ in only one position, the outputs $S(x_1)$ and $S(x_2)$ differ in at least two positions.

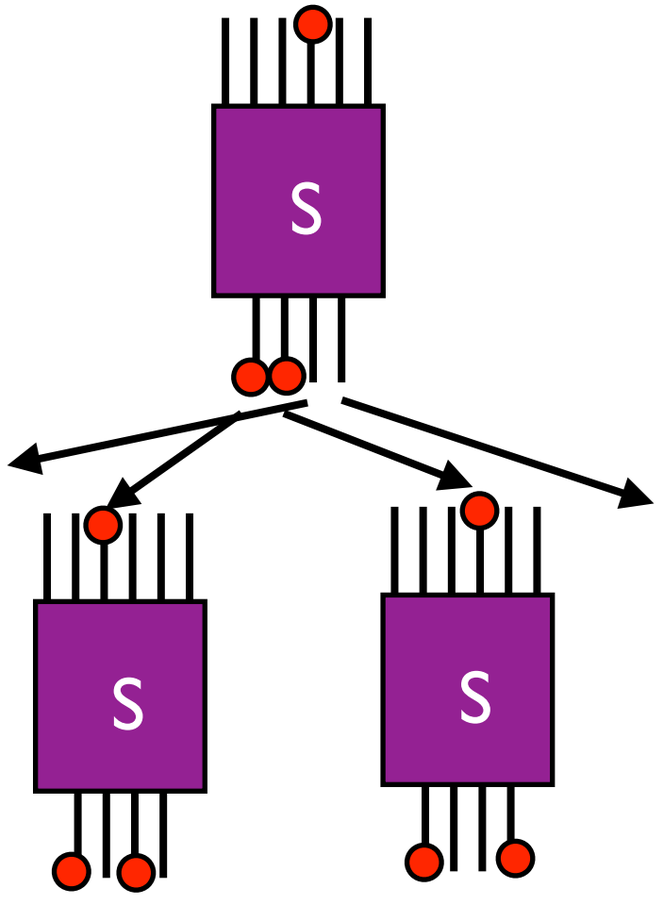
This class is being recorded

Avalanche Effect for Mangler

The S-boxes have the property that they double the size of single-bit disturbances. The expansion step may or may not increase the size of disturbances. The permutation cannot increase the size of a disturbance, but it is still important.



Any disturbance magnified in one round will return to be magnified more in future rounds, but without the permutation, the 2 flipped bits will likely end up in the same S-box and won't be further magnified. The permutation makes sure they hit different S-boxes, where each can be magnified again.

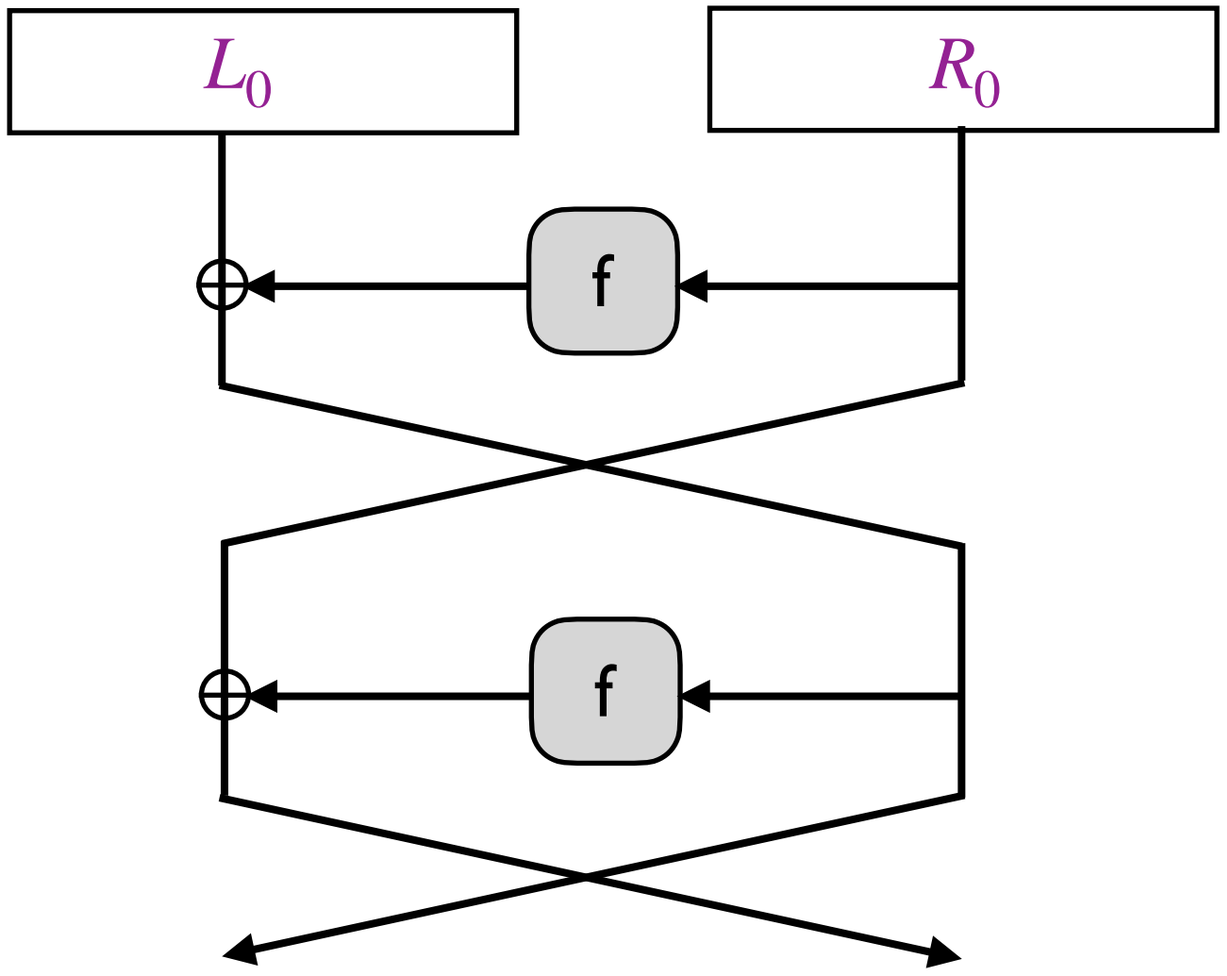


This class is being recorded

Avalanche Effect for Feistel Network

The multiple rounds of a Feistal network facilitate the avalanche effect provided the f functions magnify changes in their inputs.

Since f has the property that a single bit flipped in the input leads to 2 or more bit flips in the output, then a bit flip in R_0 leads to two bit flips in R_1, which in turn leads to 4 bit flips in R_2, and so on. 16 rounds is plenty to make sure a single bit change in the input leads to totally different outputs.



This class is being recorded

Breaking DES

DES was created in the 1970s, and at that time the key length of 56 bits was adequate (although still a bit short). However, by the 1990s, it was possible to break DES via a brute force attack on specialized hardware.

> (There are faster theoretical attacks but they use too many chosen plaintexts to be practical.)

Since DES was widespread, this problem has been addressed with 3DES, which simply runs DES 3 times with different keys.

> There is a "meet-in-the-middle" attack against 3DES which uses time more like 2^{2*56} rather than the brute force time of 2^{3*56}. However, this is still secure in practice.

However, AES is a more modern encryption protocol and is better.

This class is being recorded