COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

Previously on COS 433…

Pseudorandom Permutations

(also known as block ciphers)

Functions that "look like" random **permutations**

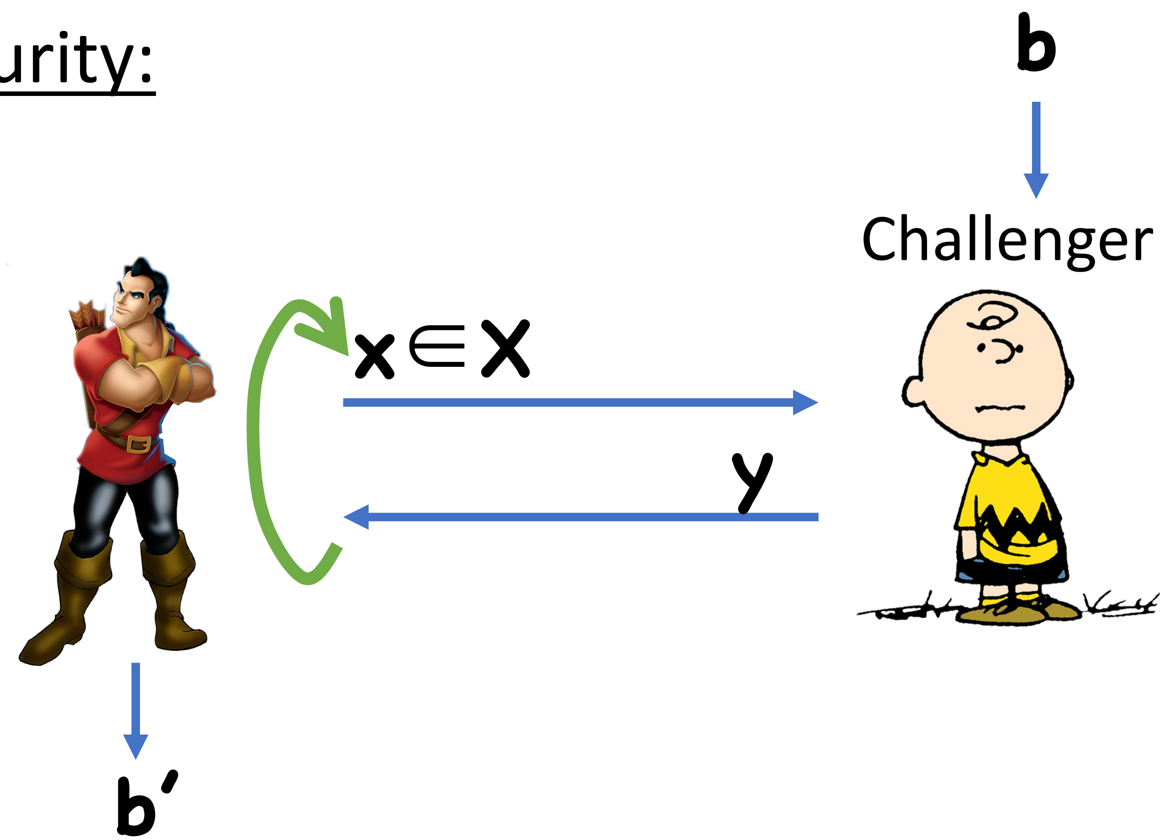
Syntax:

- Key space **K** (usually $\{0,1\}^\lambda$)
- Domain=Range= **X** (usually $\{0,1\}^n$)
- Function $F: K \times X \rightarrow X$
- Function $F^{-1}: K \times X \rightarrow X$

Correctness: $\forall k, x,\ F^{-1}(k, F(k, x)) = x$

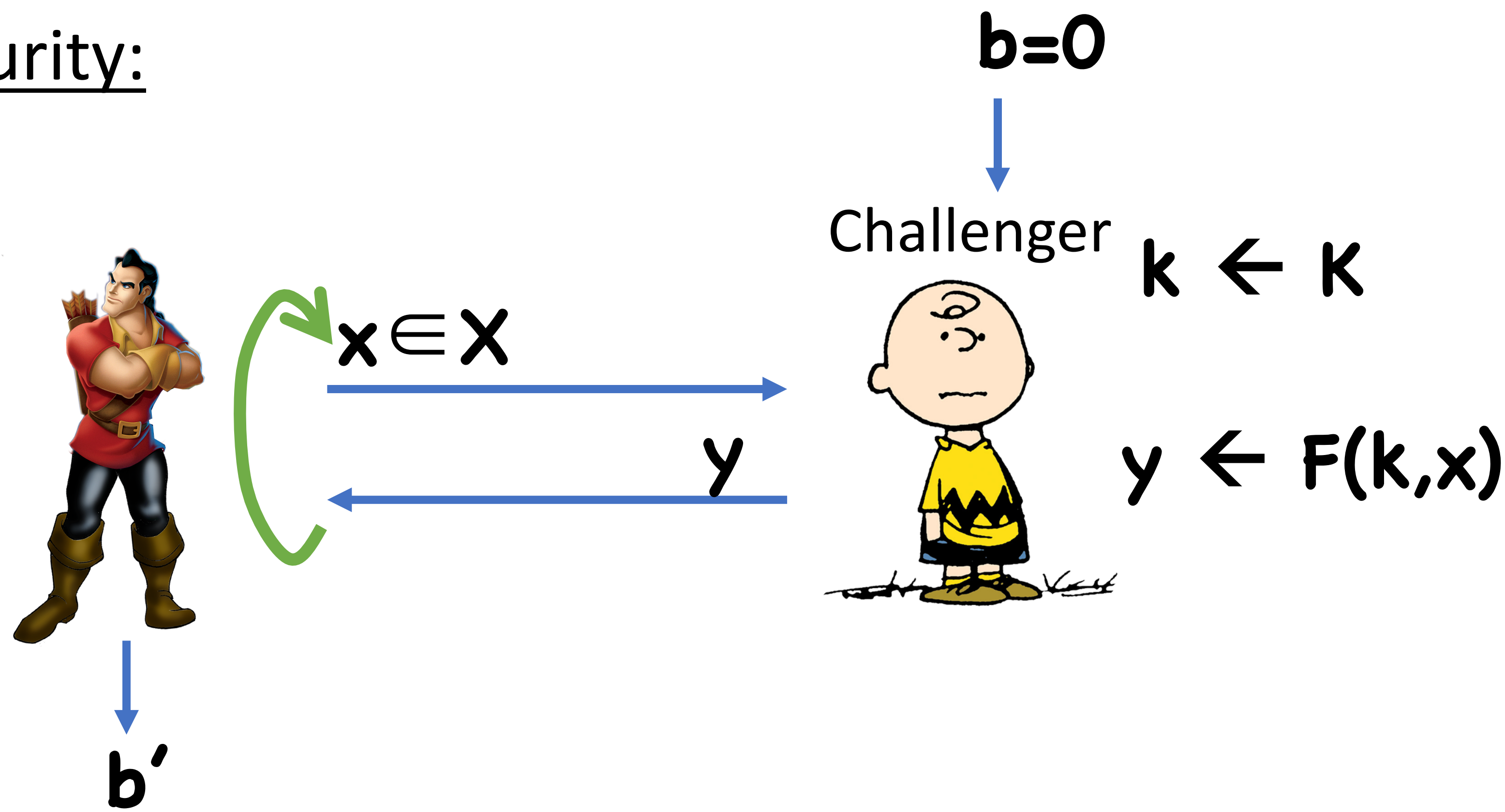
Pseudorandom Permutations

<u>Security:</u>



Pseudorandom Permutations

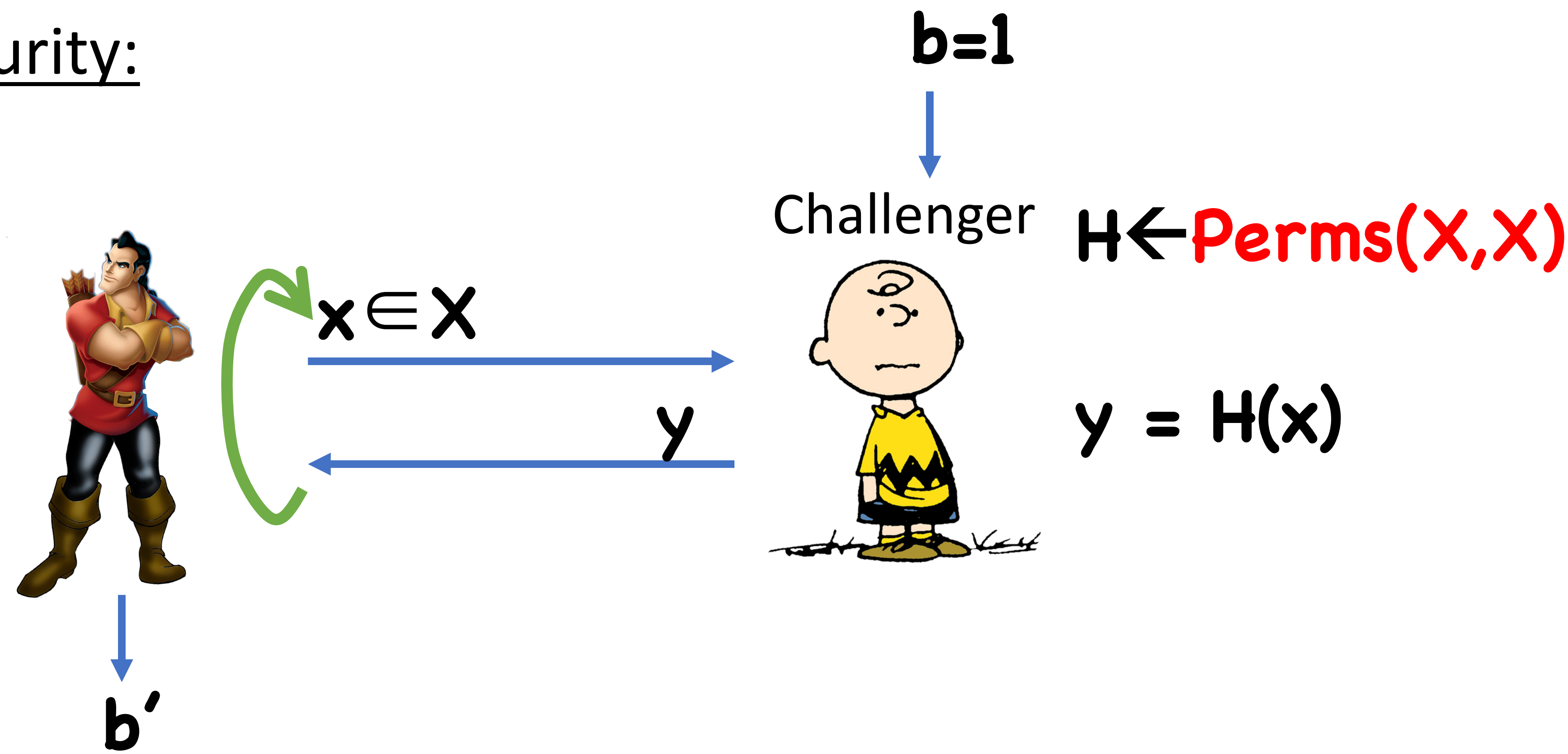
<u>Security:</u>



PRF-Exp$_0$( **)**

Pseudorandom Permutations

<u>Security:</u>



PRF-Exp$_1$( **)**

PRF Security Definition

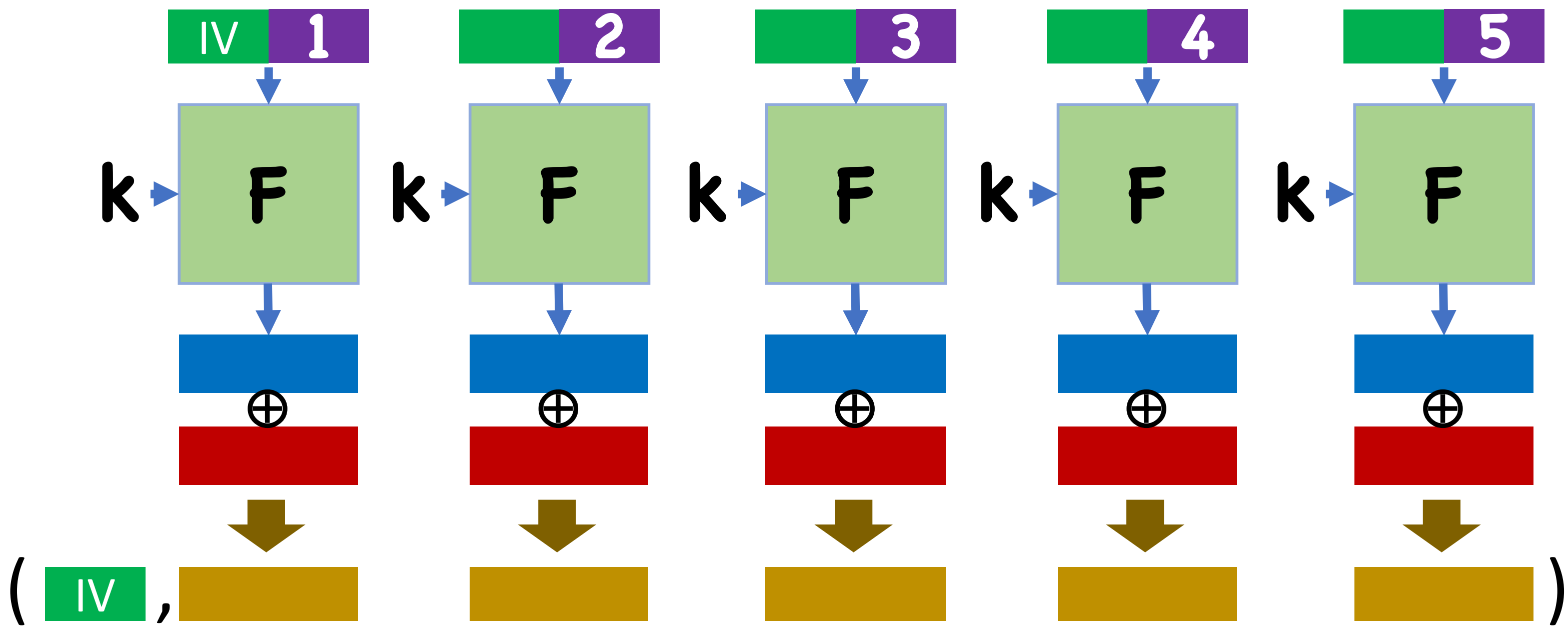
Definition: F is a **(t,q,ε)-**secure PRP if, for all  running in time at most **t** and making at most **q** queries,

| Pr[1←PRF-Exp$_0$( **)]**

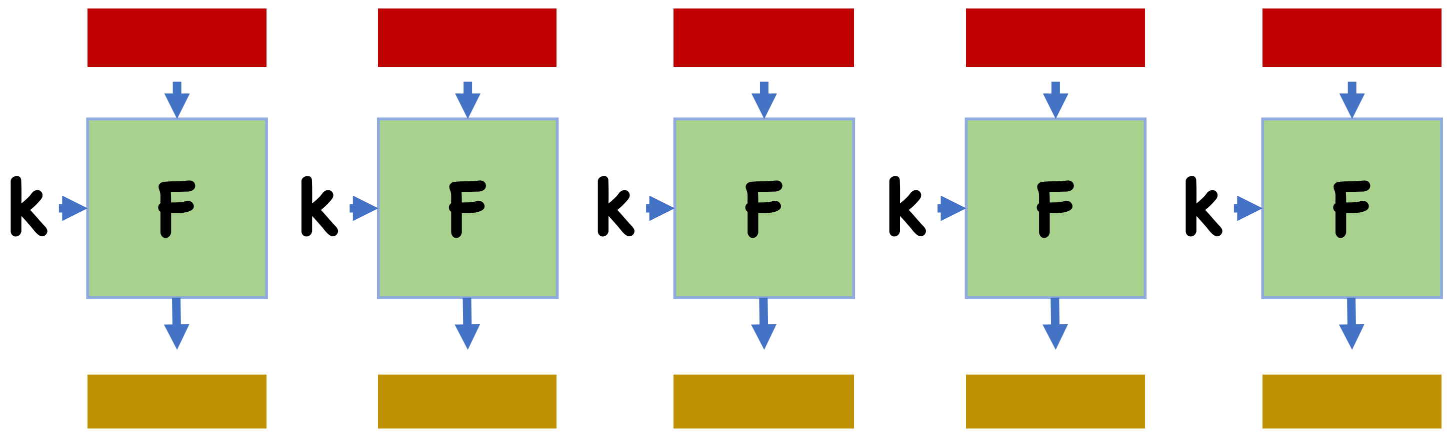
− Pr[1←PRF-Exp$_1$( **)] | ≤ ε**

Theorem: A PRP (F,F^{-1}) is (t,q,ε)-secure iff F is $(t,q,\varepsilon+q^2/2|X|)$-secure as a PRF

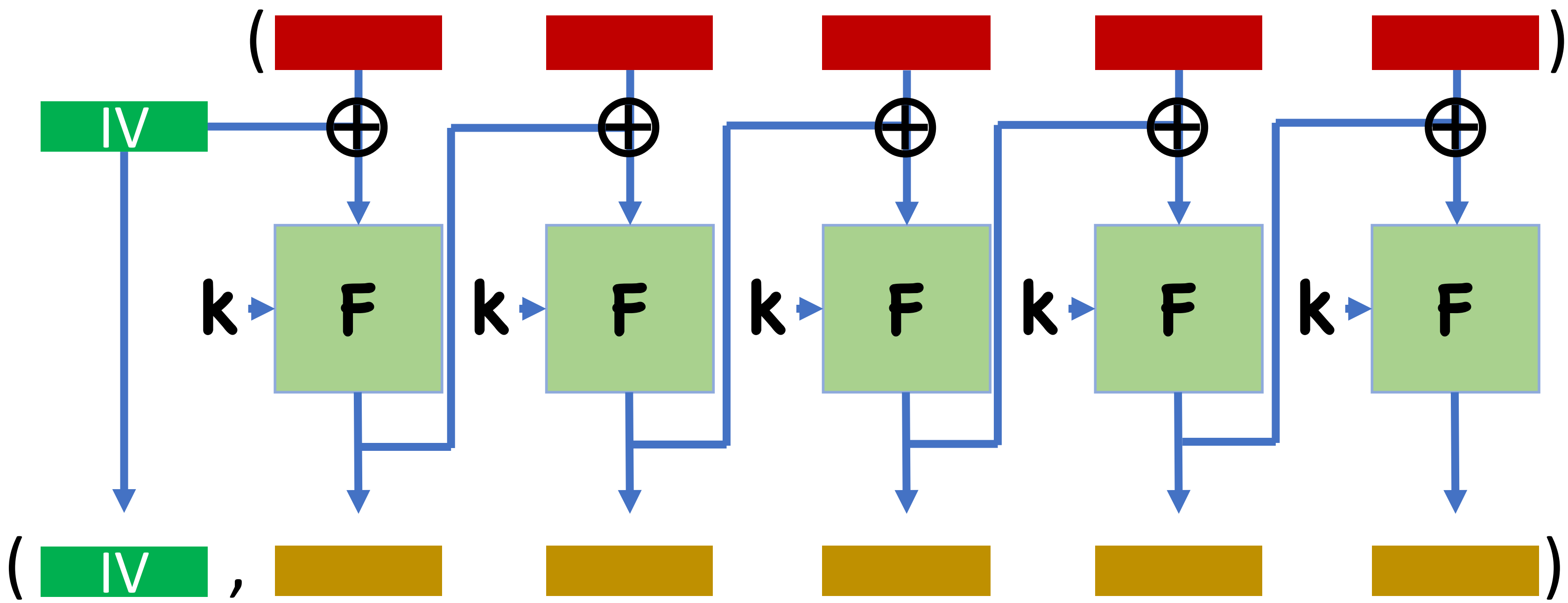
Counter Mode (CTR)



Electronic Code Book (ECB)



Cipher Block Chaining (CBC) Mode



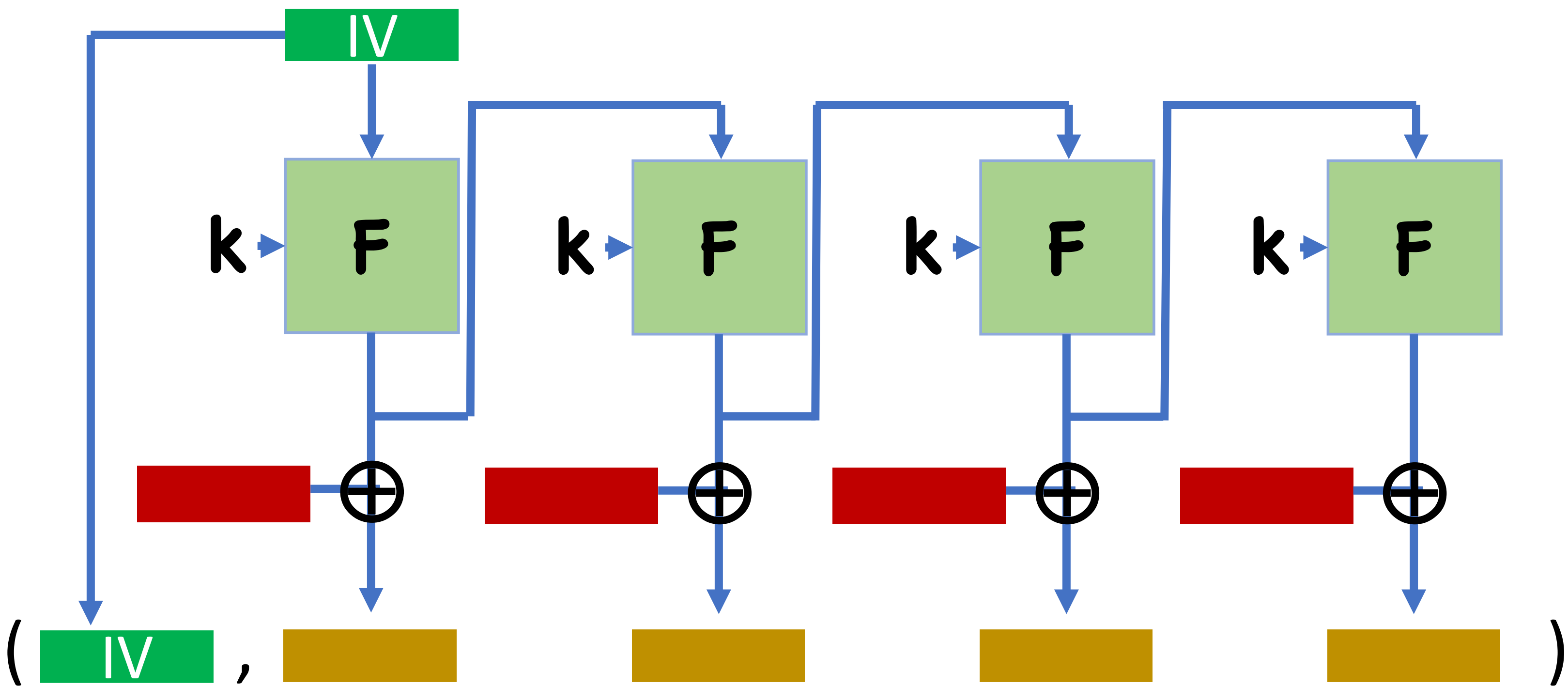
(For now, assume all messages are multiples of the block length)

Today

A few more modes of operation

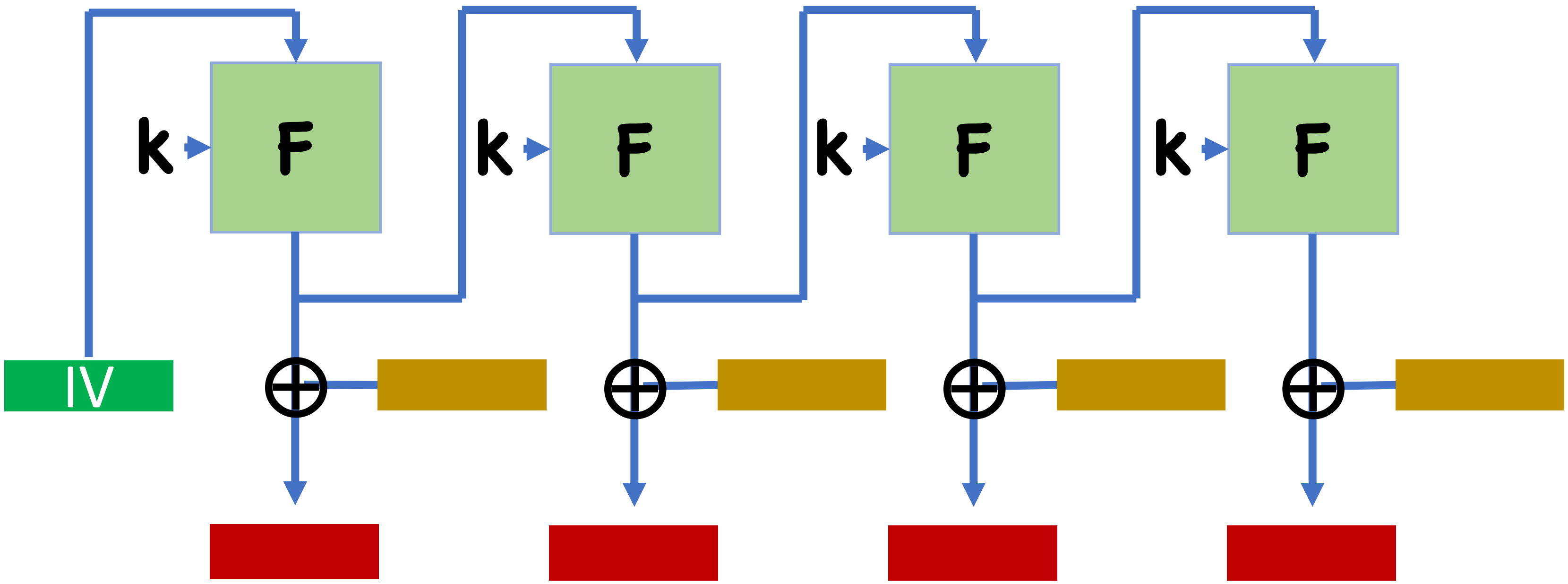
How to construct block ciphers

Output Feedback Mode (OFB)



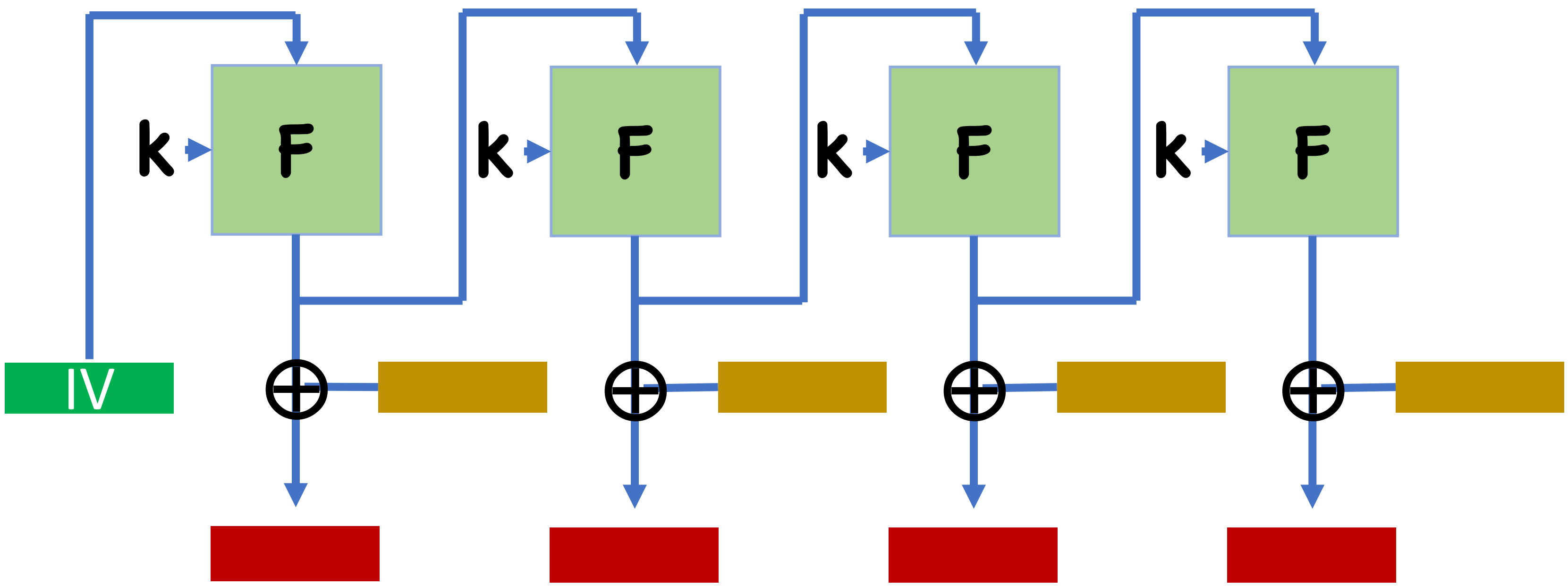
Turn block cipher into stream cipher

OFB Decryption



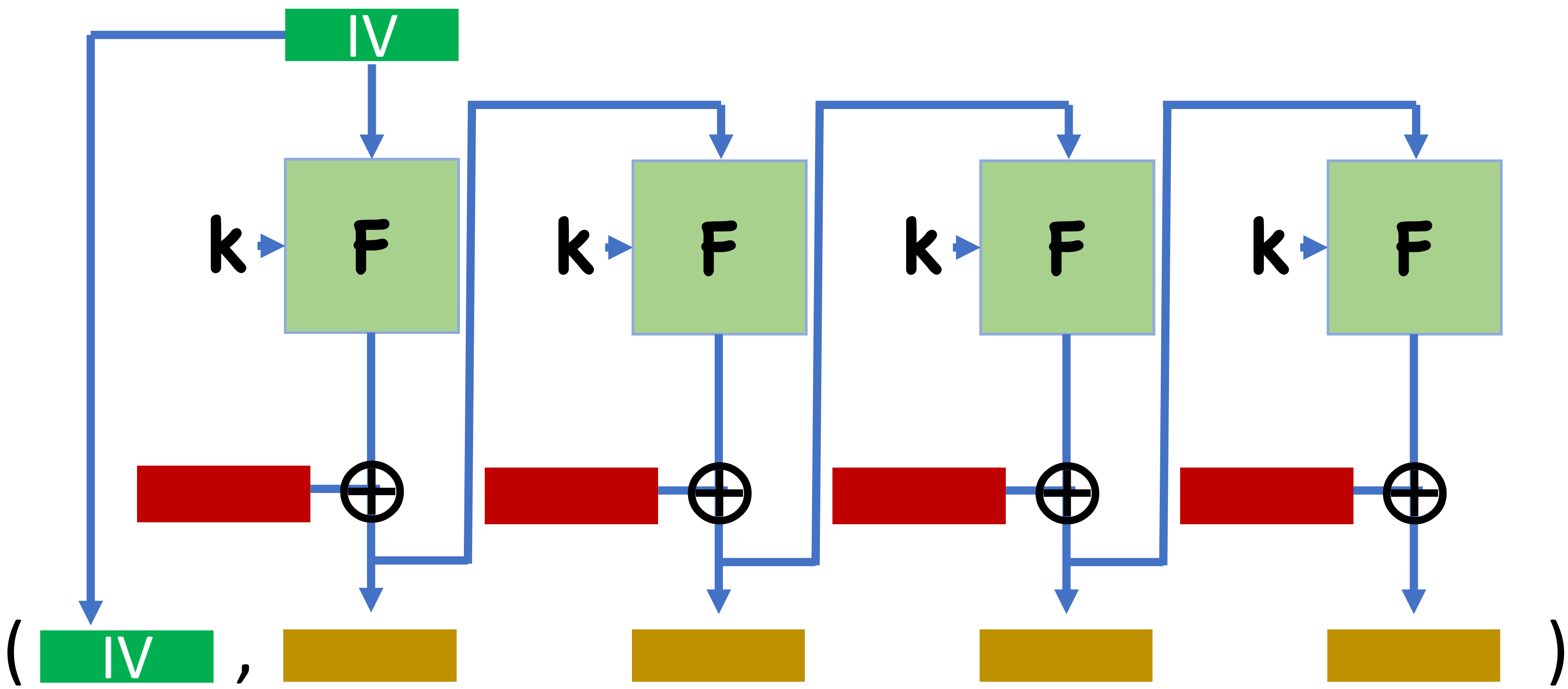
What happens if a block is lost in transmission?

OFB decryption:



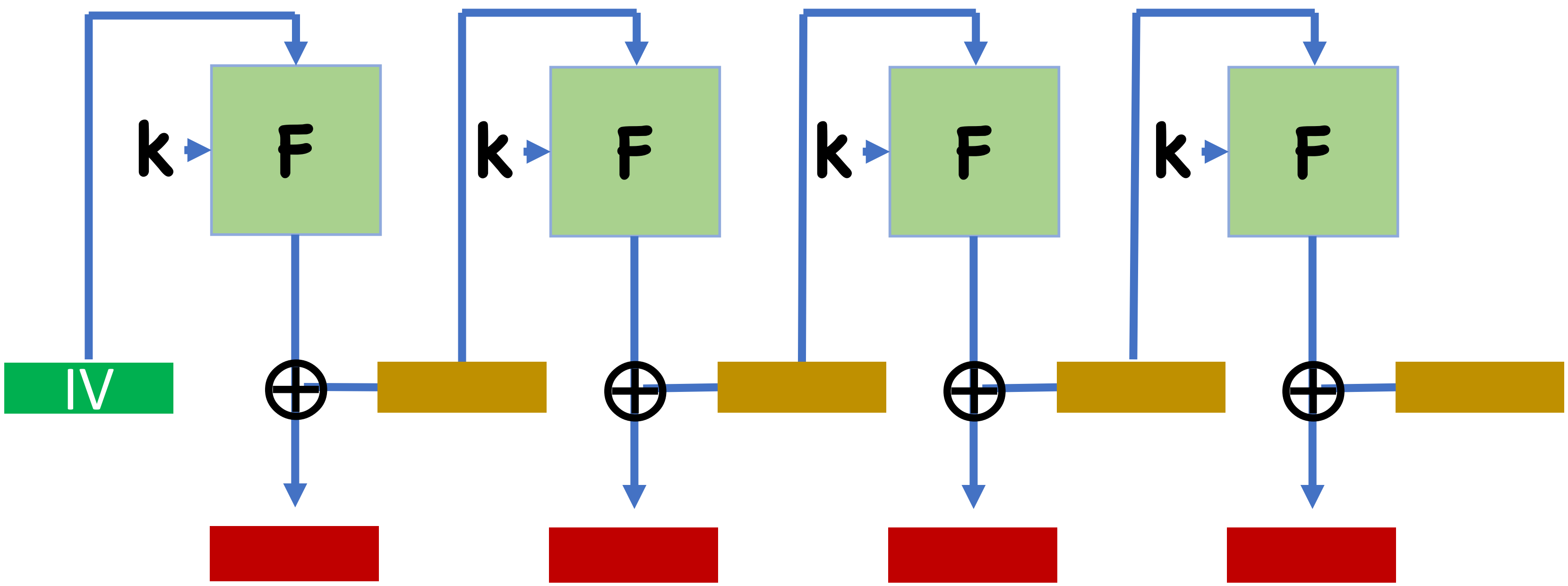
Same goes for CTR mode

Cipher Feedback (CFB)



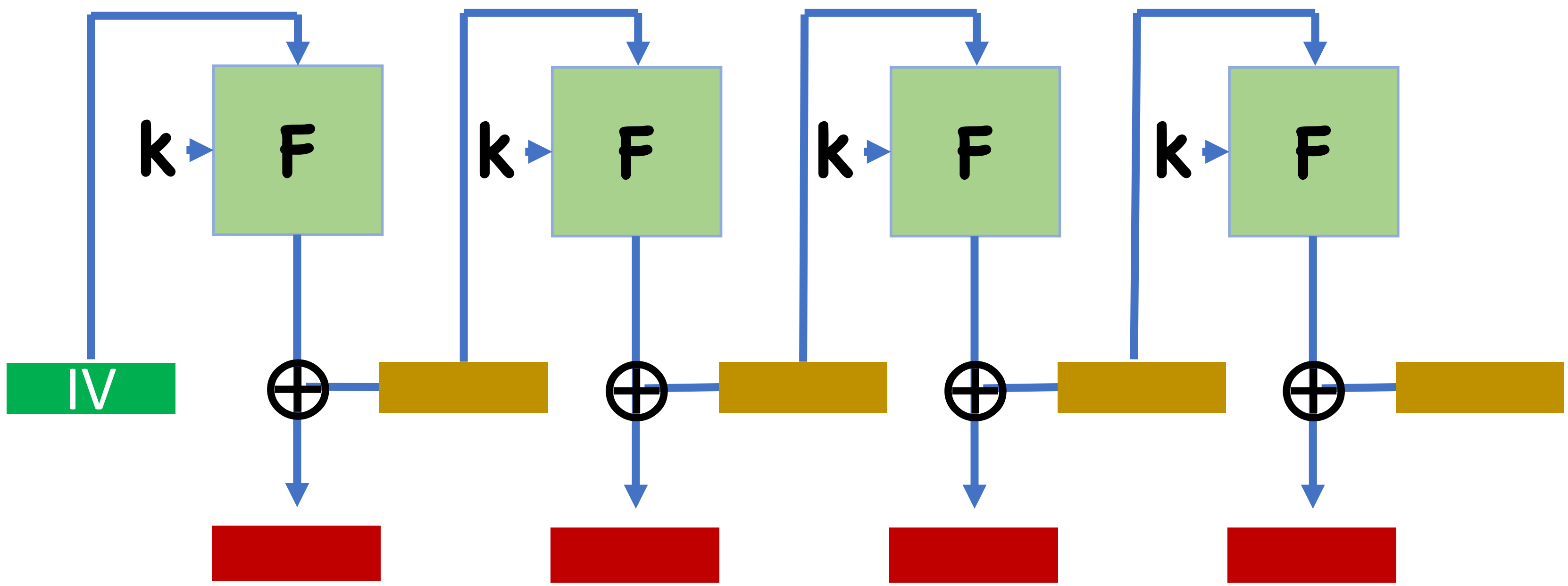
Turn block cipher into **self-synchronizing** stream cipher

CFB Decryption



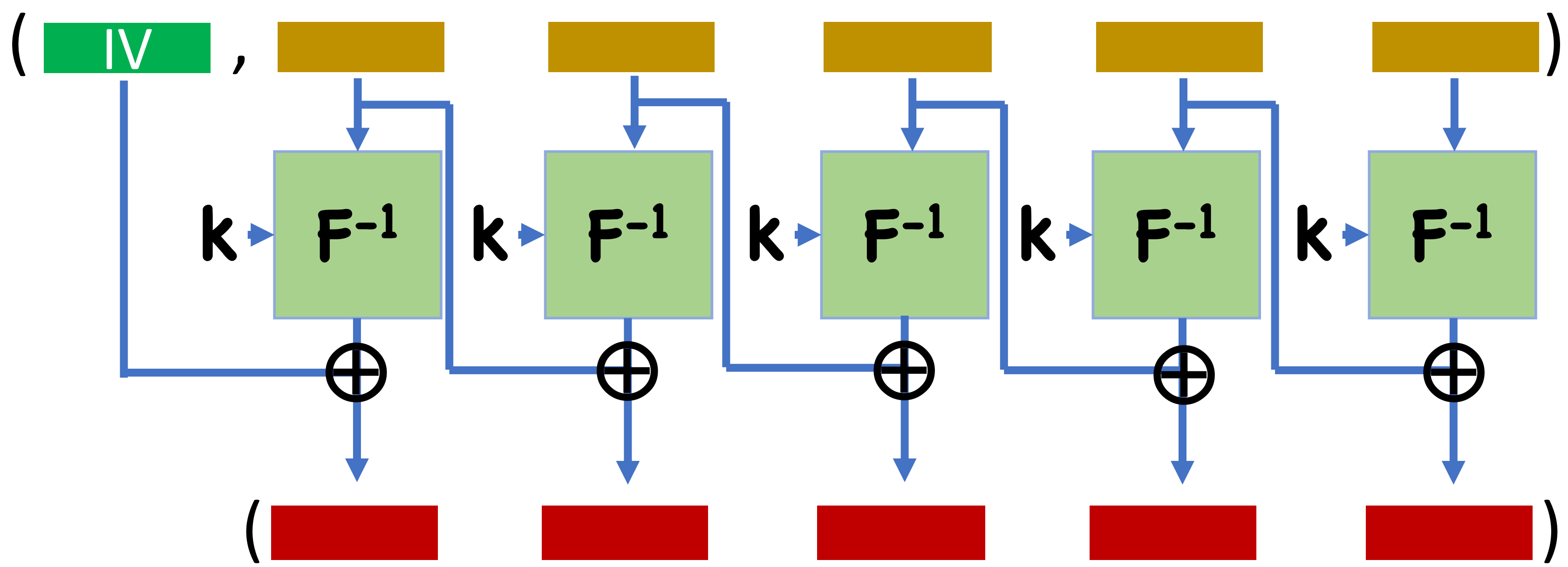
What happens if a block is lost in transmission?

CFB decryption:



What happens if a block is lost in transmission?

What about CBC?



Security of OFB, CFB modes

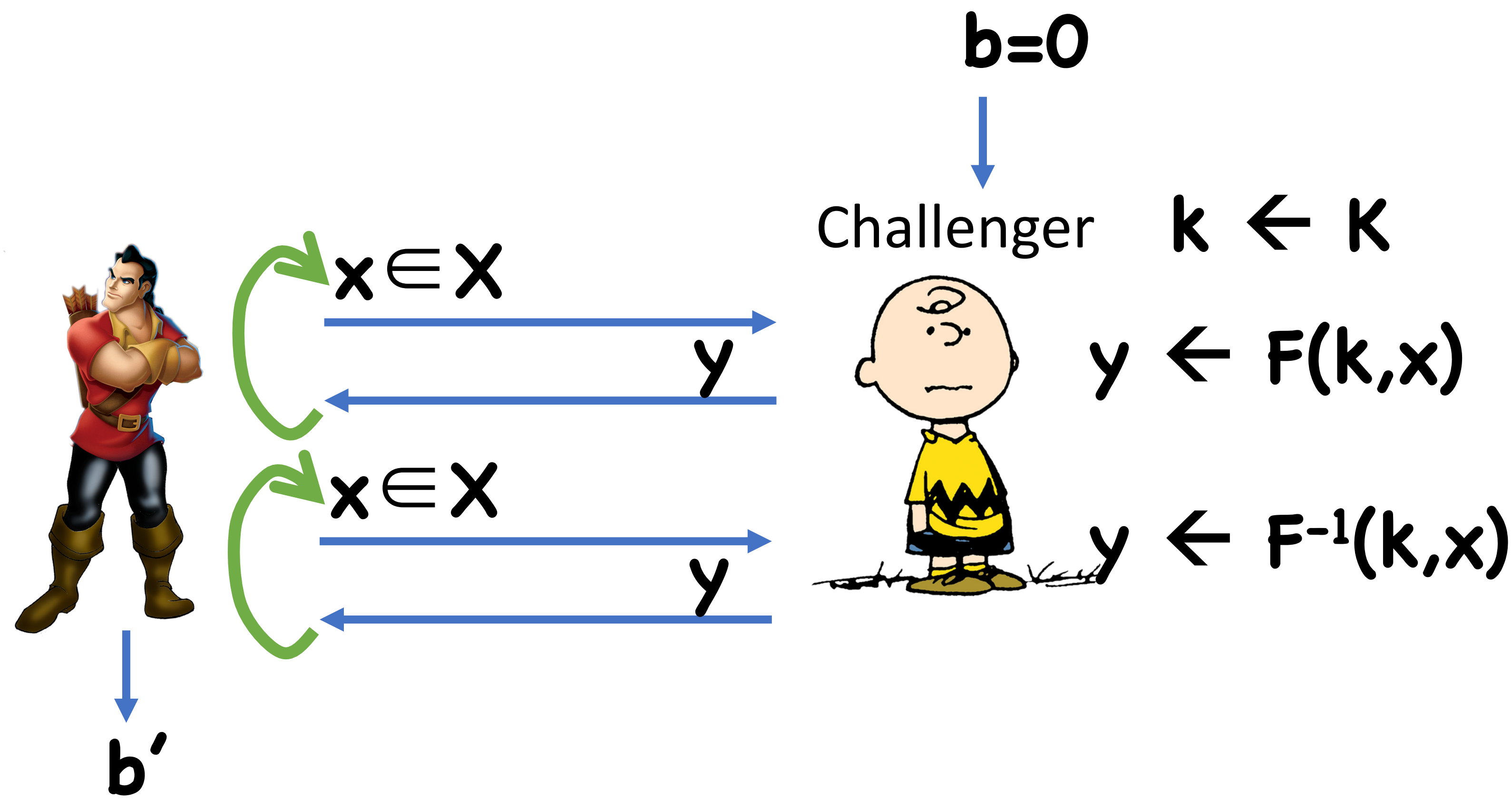
Security very similar to CBC

Define 4 hybrids
- 0: encrypt left messages
- 1: replace PRP with random permutation
- 2: encrypt right messages
- 3: replace random permutation with PRP

0,1 and 2,3 are indistinguishable by PRP security

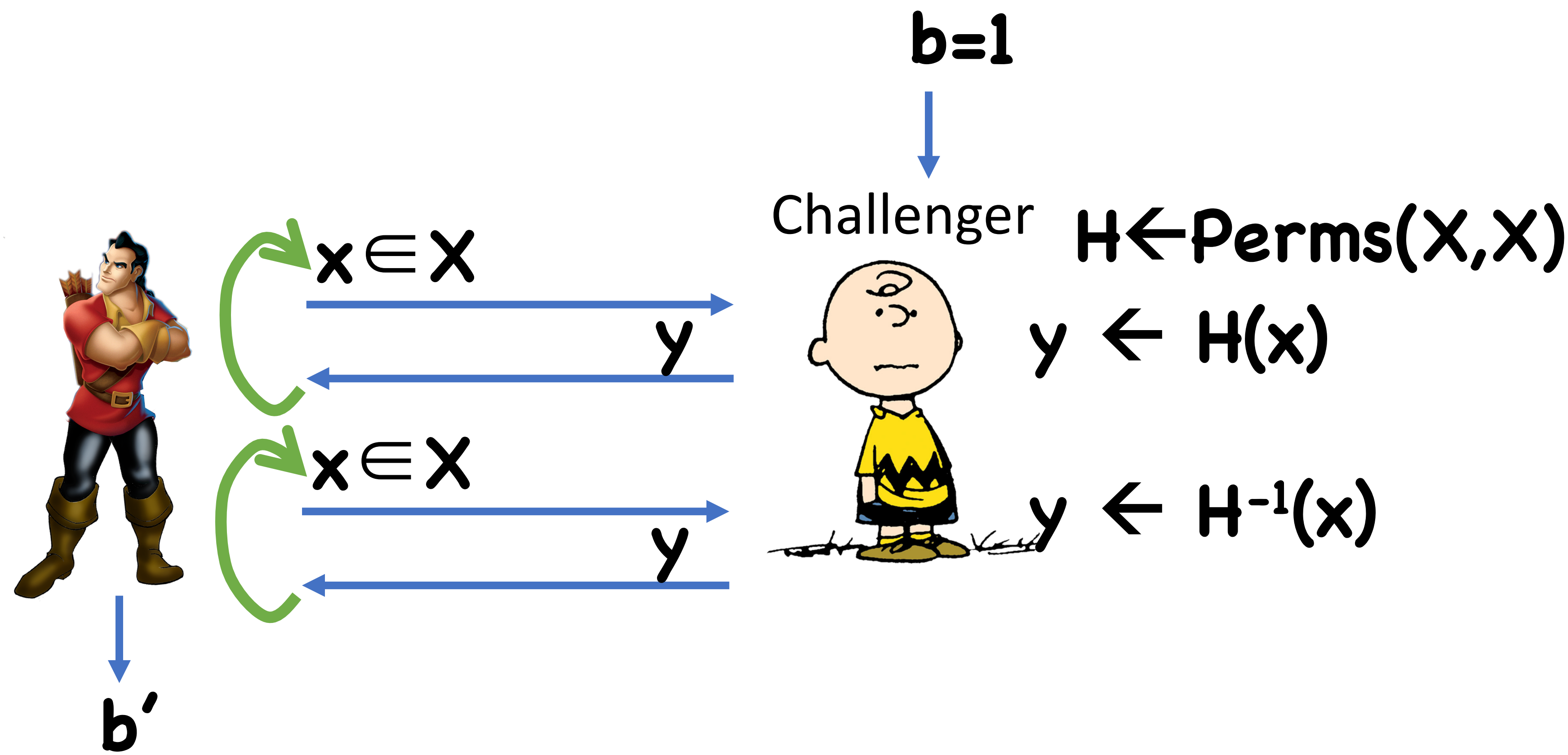
1,2 are indistinguishable since ciphertexts are essentially random

Strong PRPs



PRF-Exp$_0$( **)**

Strong PRPs



PRF-Exp$_1$()

Theorem: If (F, F^{-1}) is a strong PRP, then so is (F^{-1}, F)

PRPs vs PRFs

In practice, PRPs are the central building block of most crypto

- Also PRFs
- Can build PRGs
- Very versatile

Constructing block ciphers

Difficulties

$2^n!$ Permutations on **n**-bit blocks

$\Rightarrow$ $\approx n2^n$ bits to write down random perm.

Reasonable for very small **n** (e.g. **n<20**), but totally infeasible for large **n** (e.g. **n=128**)

Challenge:

- Design permutations with small description that "behave like" random permutations

Difficulties

For a random permutation **H**, **H(x)** and **H(x′)** are (essentially) independent random strings

- Even if **x** and **x′** differ by just a single bit

Therefore, for a random key **k**, changing a single bit of **x** should "affect" all output bits of **F(k,x)**

Definition: For a function $\mathbf{H:\{0,1\}^n \rightarrow \{0,1\}^n}$, we say that bit $\mathbf{i}$ of the input affects bit $\mathbf{j}$ of the output if:

For a random $\mathbf{x_1,\ldots,x_{i-1},x_{i+1}, \ldots, x_n}$, if we let
$\mathbf{y=H(x_1\ldots x_{i-1}0x_{i+1}\ldots x_n)}$ and
$\mathbf{z=H(x_1\ldots x_{i-1}1x_{i+1}\ldots x_n)}$
Then $\mathbf{y_j \neq z_j}$ with probability $\mathbf{\approx 1/2}$

Theorem: If $(\mathbf{F},\mathbf{F}^{-1})$ is a secure PRP, then with (with "high" probability over the key **k**), for the function **F(k,•)**, every bit of input affects every bit of output

Proof:

- For random permutations this is true
- If bit **i** did not affect bit **j**, we can construct an adversary that distinguishes **F** from random

Confusion/Diffusion Paradigm

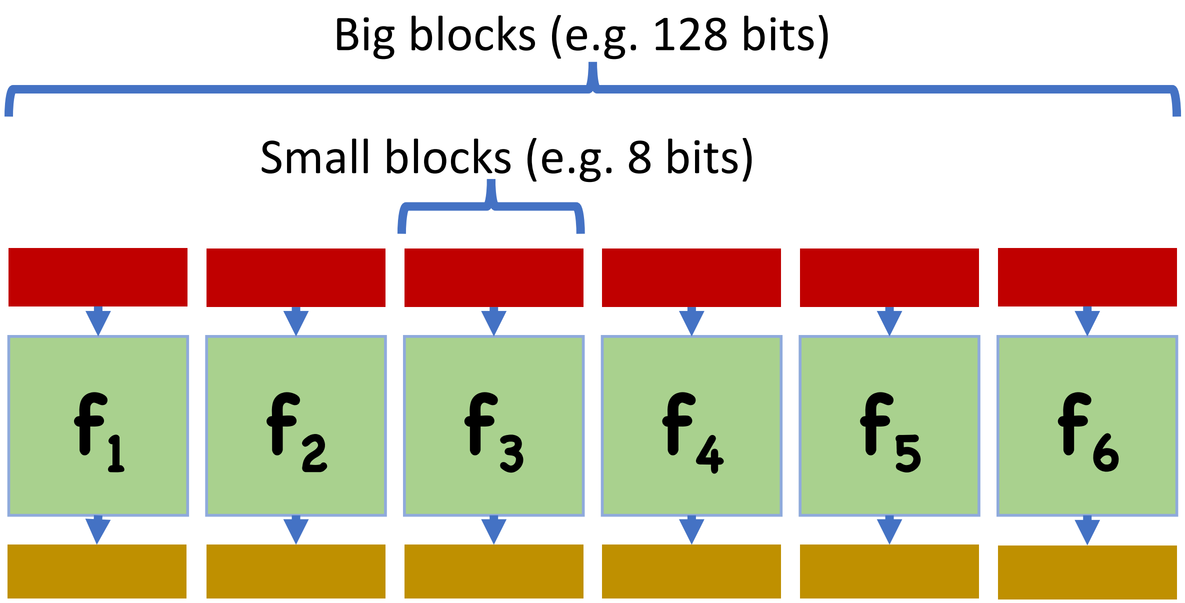
Confusion/Diffusion Paradigm

Goal: build permutation for large blocks from permutations for small blocks

- Small block perms can be made truly random
- Hopefully result is pseudorandom

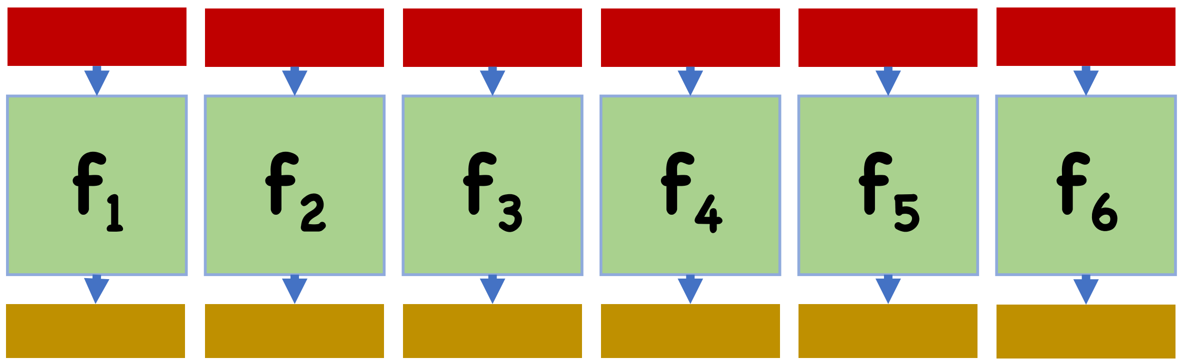
Confusion/Diffusion Paradigm

First attempt: break blocks into smaller blocks, apply smaller permutation blockwise



Key: description of f_1, f_2,…

Confusion/Diffusion Paradigm

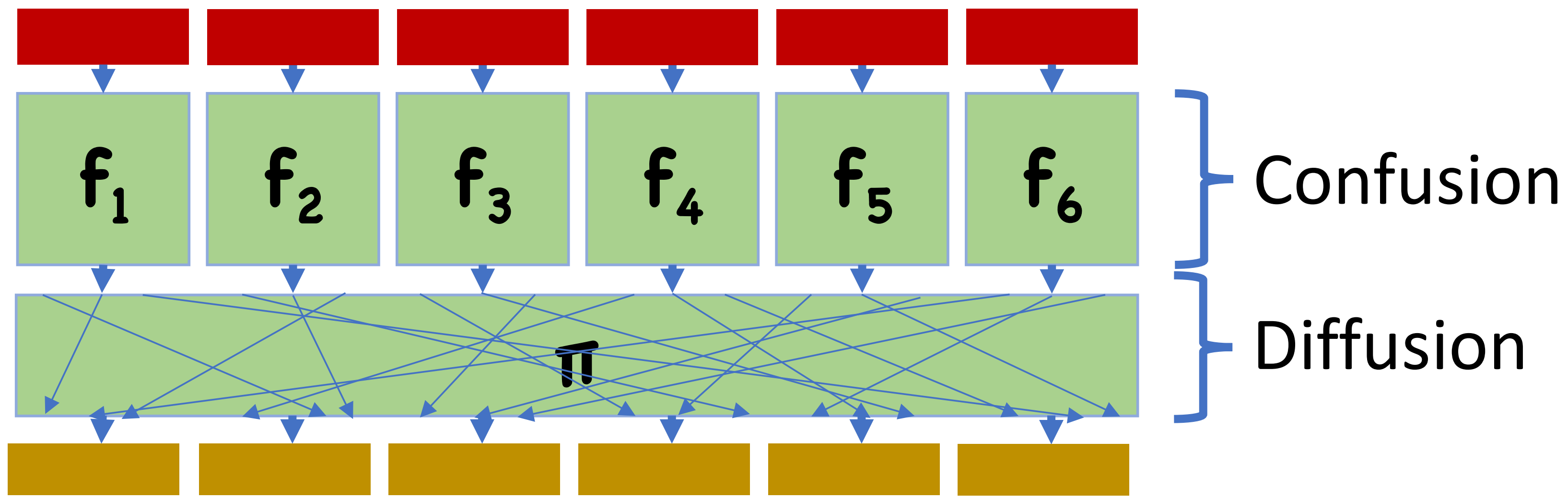


Is this a secure PRP?

- Key size: **$\approx(8\times2^8)\times(128/8) = 2^{15}$**, so reasonable
- Running time: a few table lookups, so efficient
- Security?

Confusion/Diffusion Paradigm

Second attempt: shuffle output bits

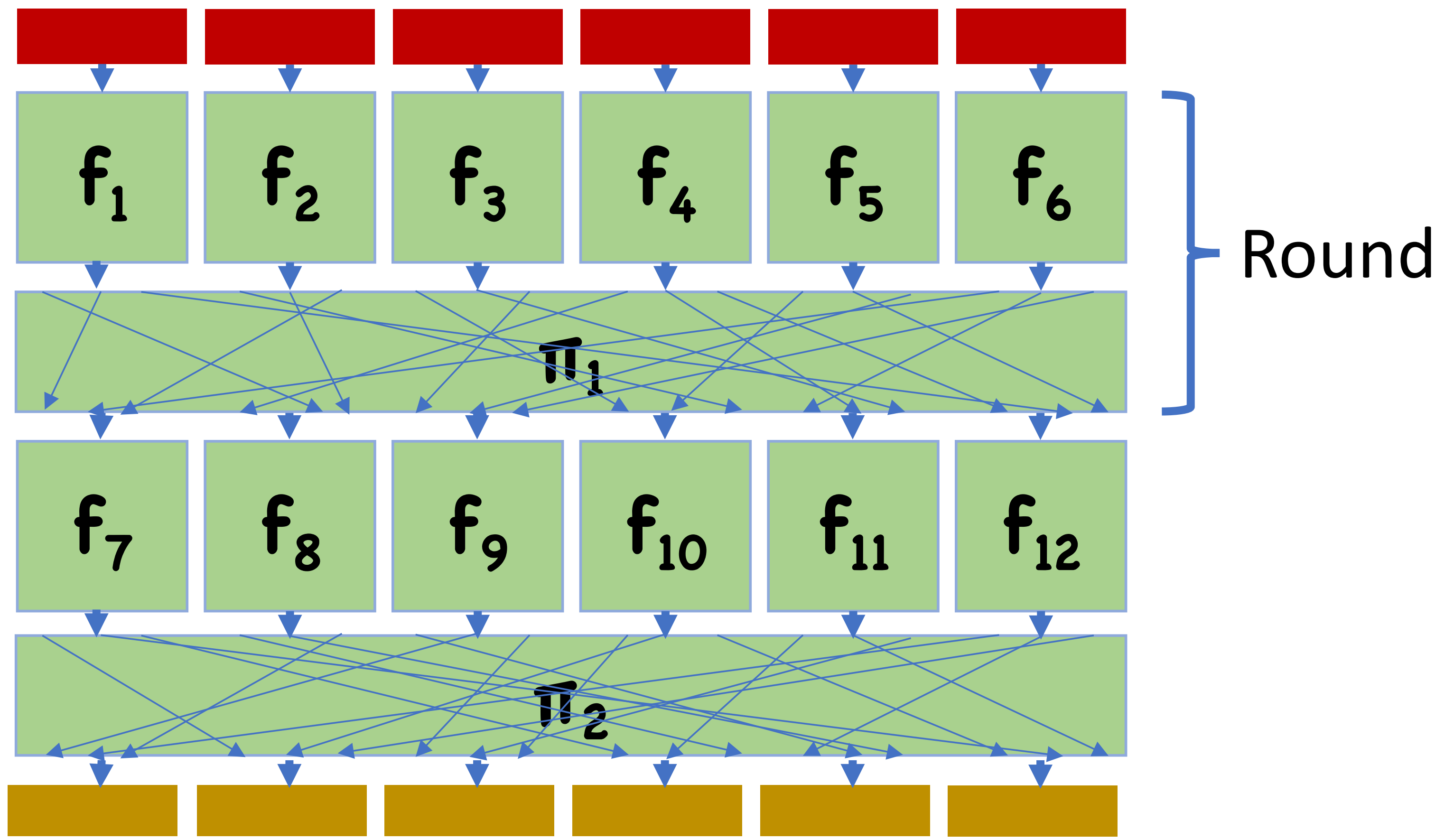


Is this a secure PRP?

- Key size: $\approx 2^{15}$ + 128×Log 128 $\approx 2^{15}$
- Running time: a few table lookups
- Security?

Confusion/Diffusion Paradigm

Third Attempt: Repeat multiple times!



Confusion/Diffusion Paradigm

While single round is insecure, we’ve made progress

- Each bit affects 8 output bits

With repetition, hopefully we will make more and more progress

Confusion/Diffusion Paradigm

With 2 rounds,

- Each bit affects 64 output bits

With 3 rounds, all 128 bits are affected

Repeat a few more times for good measure

Limitations

Describing subs/perms requires many bits

- Key size for **r** rounds is approximately $2^{15}\times r$
- Ideally want key size to be **128** (or **256**)

Idea: instead, fix subs/perms

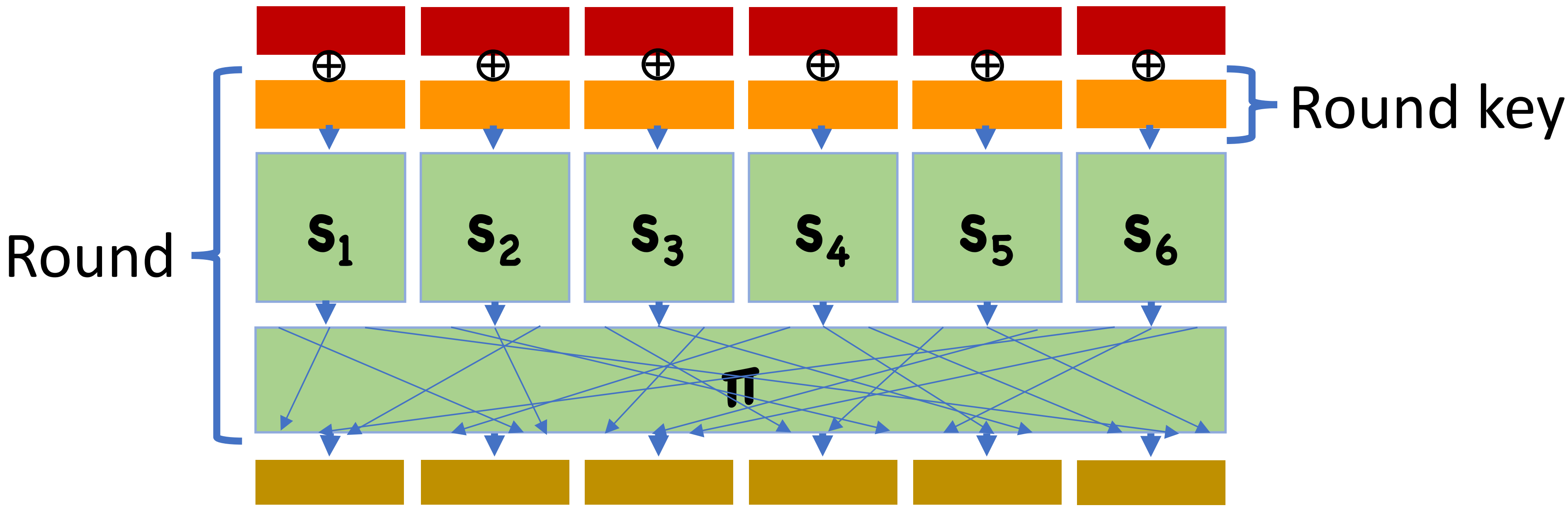
- But then what's the key?

Substitution Permutation Networks

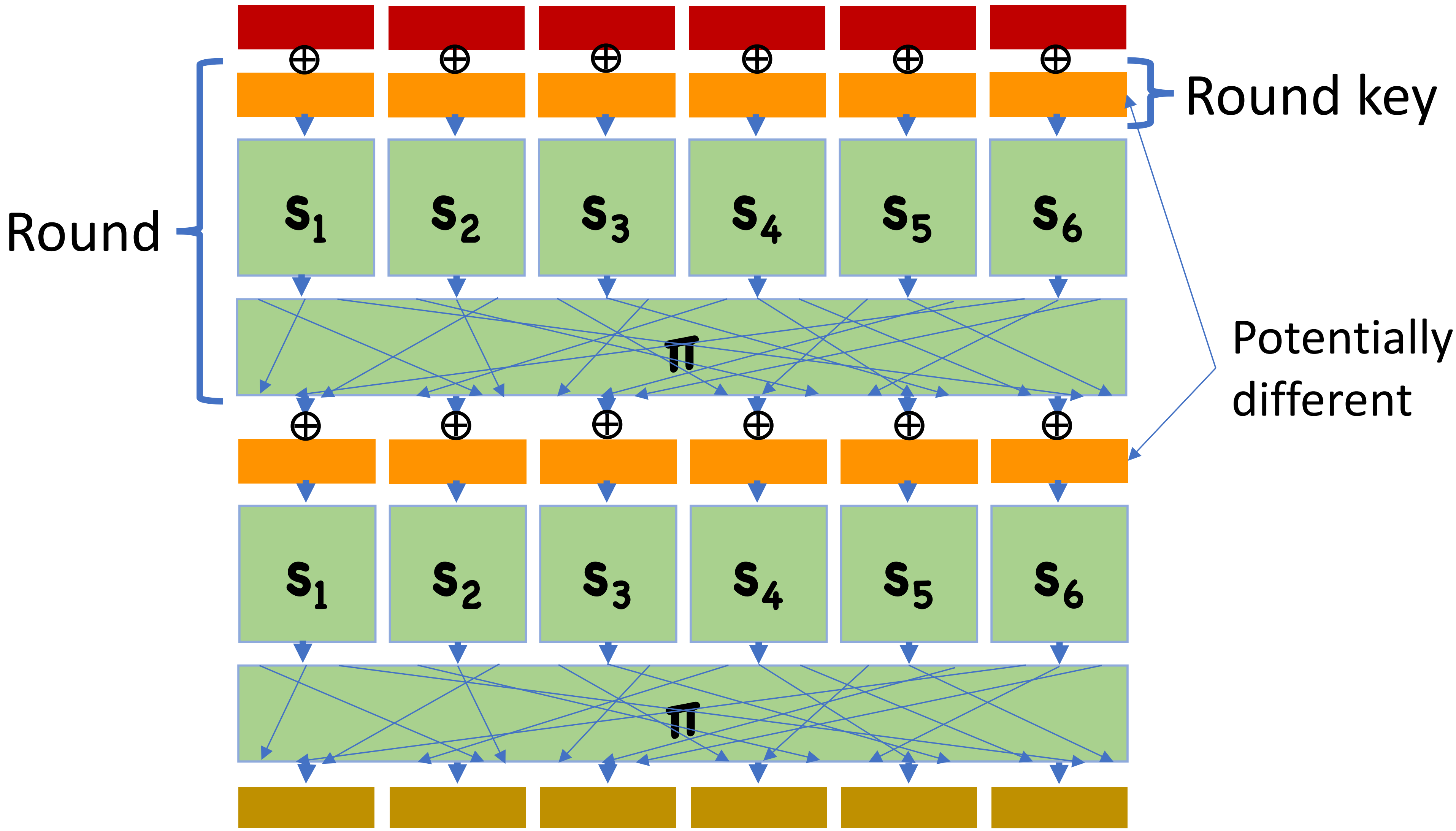
Variant of previous construction

- Fixed public permutations for confusion (called a substitution box, or S-box)
- Fixed public permutation for diffusion (called a permutation box, or P-box)
- XOR "round key" at beginning of each round

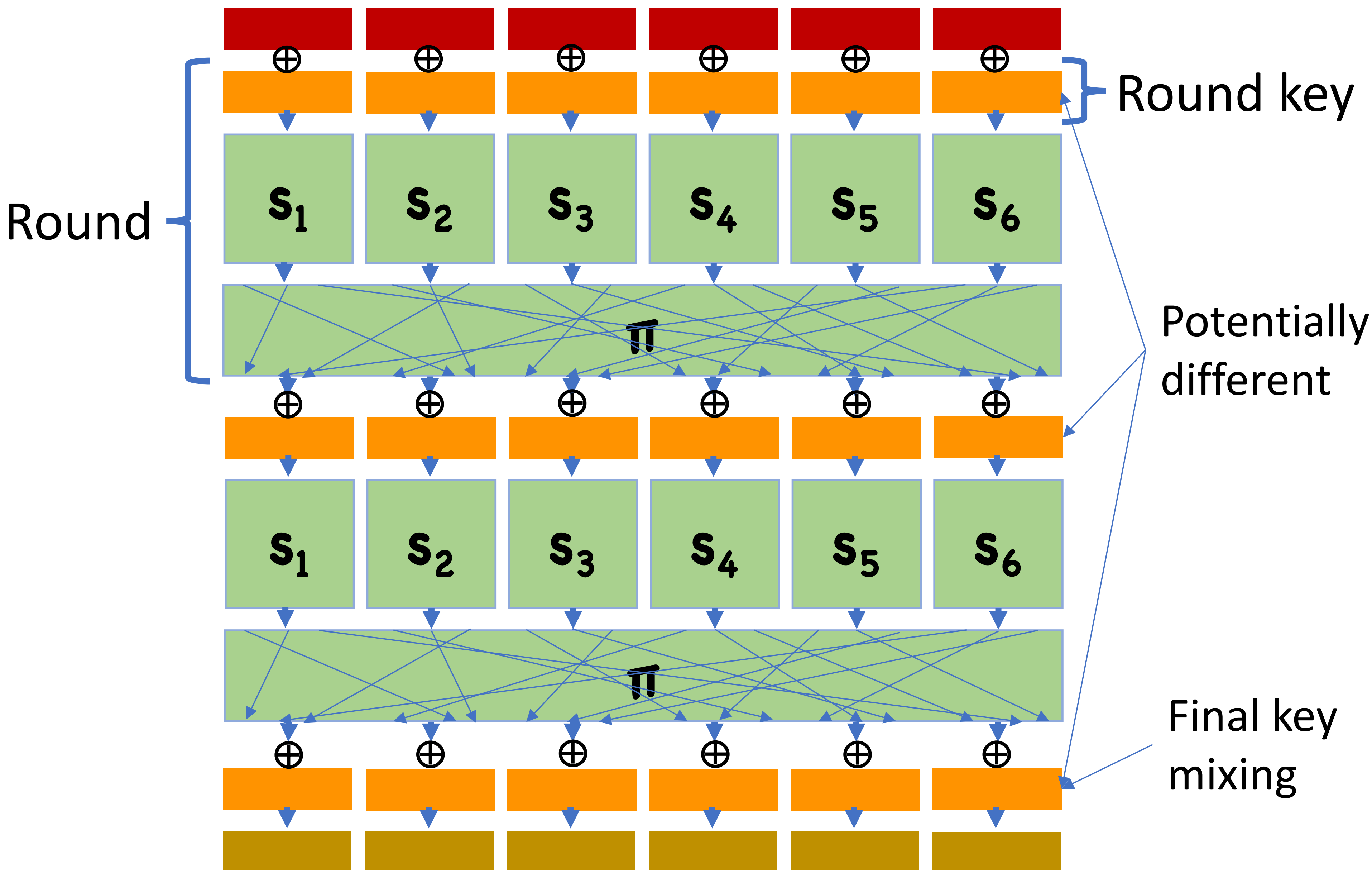
Substitution Permutation Networks



Substitution Permutation Networks



Substitution Permutation Networks



Substitution Permutation Networks

To specify a network, must:

- Specify S-boxes
- Specify P-box
- Specify key schedule (how round keys are derived from master)

Choice of parameters can greatly affect security

Designing SPNs

Avalanche Affect:

- Need S-boxes and mixing permutations to cause every input bit to ”affect” every output bit

One way to guarantee this:

- Changing any bit of S-box input causes at least 2 bits of output to change
- Mixing permutations send outputs of S-boxes into at least 2 different S-boxes for next round
- Sufficiently many rounds are used
- At least how many rounds should be used?

Designing SPNs

For strong PRPs, need avalanche in reverse too

- Changing one bit of output of S box changes at least 2 bits of input
- Mixing permutations take inputs for next round from at least two different S-box outputs

Designing S-Boxes

Random?

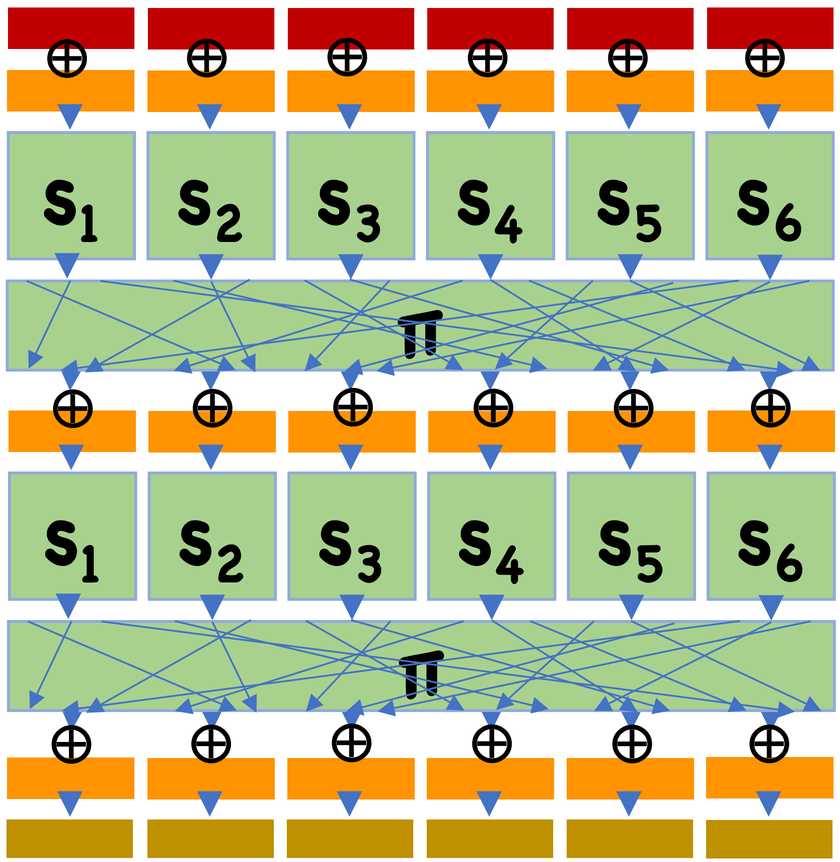
- Let x,x' be two distinct 8-bit values
- $\Pr[S(x)$ and $S(x')$ differ on a single bit$] = 8/255$
- Call such x,x' "bad"
- $\Pr[\exists \text{ bad } x,x'] = (1-8/255)\char`^\binom{256}{2} \approx 1$
- Very high probability that some pair of inputs will have outputs that differ on a single bit

Therefore, must carefully design S-boxes rather than choose at random

Linearity?

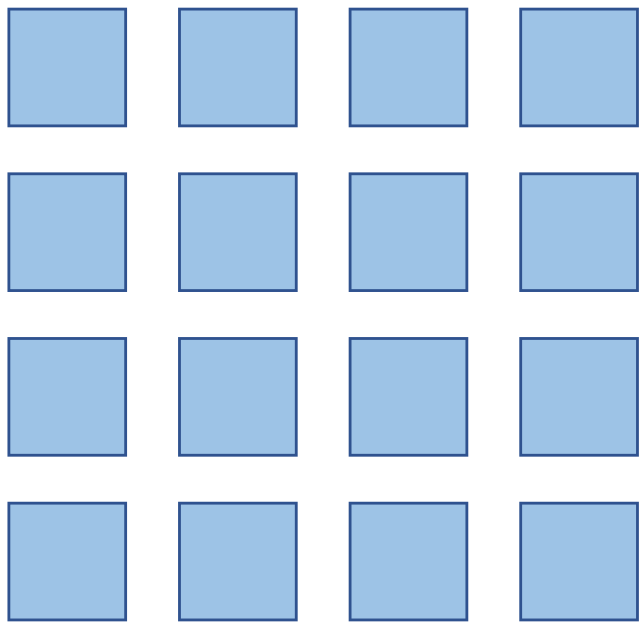
Can S-Boxes be linear?

- That is, $\mathbf{S(x_0)} \oplus \mathbf{S(x_1)} = \mathbf{S(x_0 \oplus x_1)}$?



AES

State = **4×4** grid of bytes



AES

One fixed S-box, applied to each byte

- Step 1: multiplicative inverse over finite field $\mathbb{F}_8$
- Step 2: fixed affine transformation
- Implemented as a simple lookup table

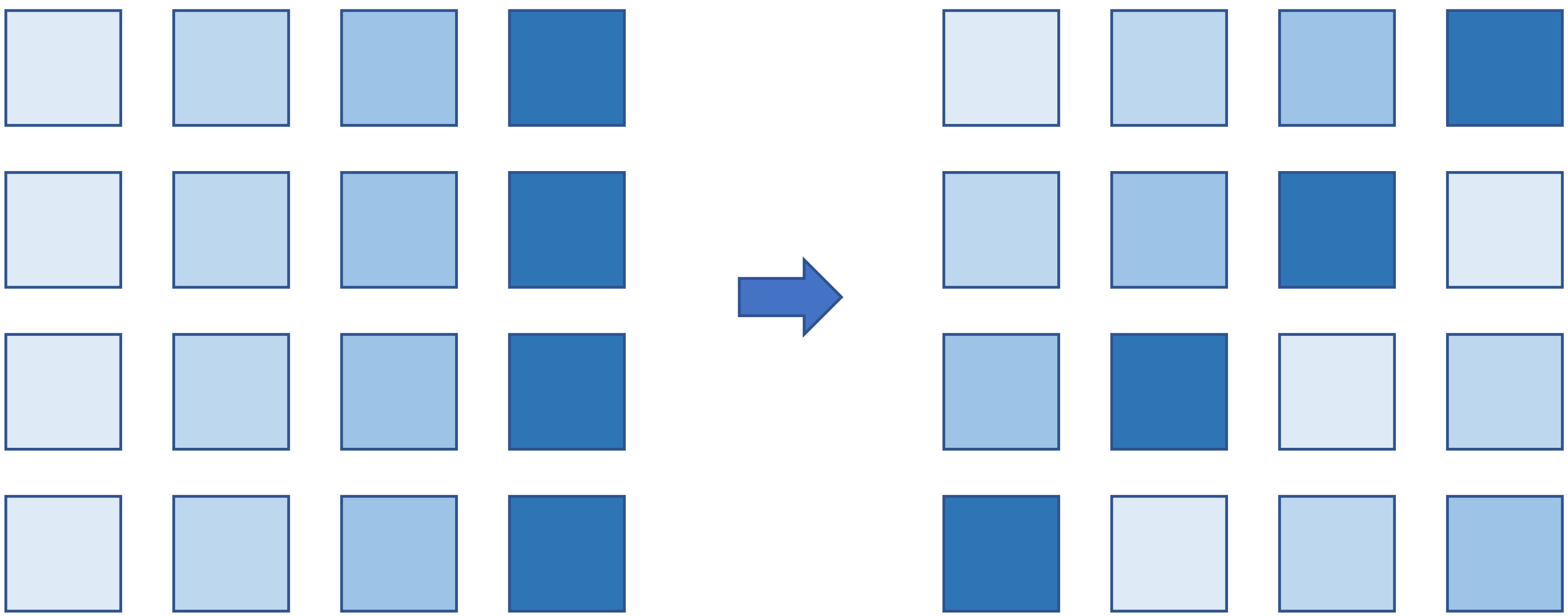
AES

Diffusion (not exactly a P-box):

- Step 1: shift rows
- Step 2: mix columns

AES

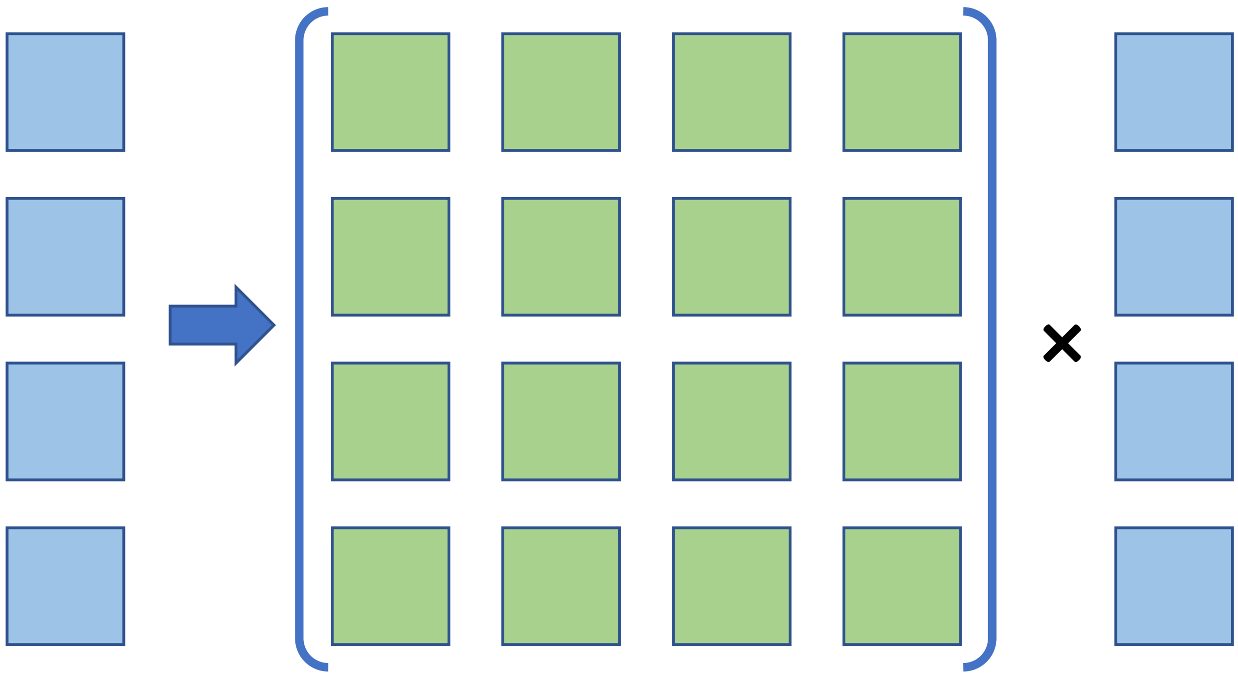
Shift Rows:



AES

Mix Columns

- Each byte interpreted as element of $\mathbb{F}_8$
- Each column is then a length-4 vector
- Apply fixed linear transformation to each column



AES

Number of rounds depends on key size

- 128-bit keys: 10 rounds
- 192-bit keys: 12 rounds
- 256-bit keys: 14 rounds

Key schedule:

- Won’t describe here, but involves more shifting, S-boxes, etc
- Can think of key schedule as a weak PRG

Feistel Networks

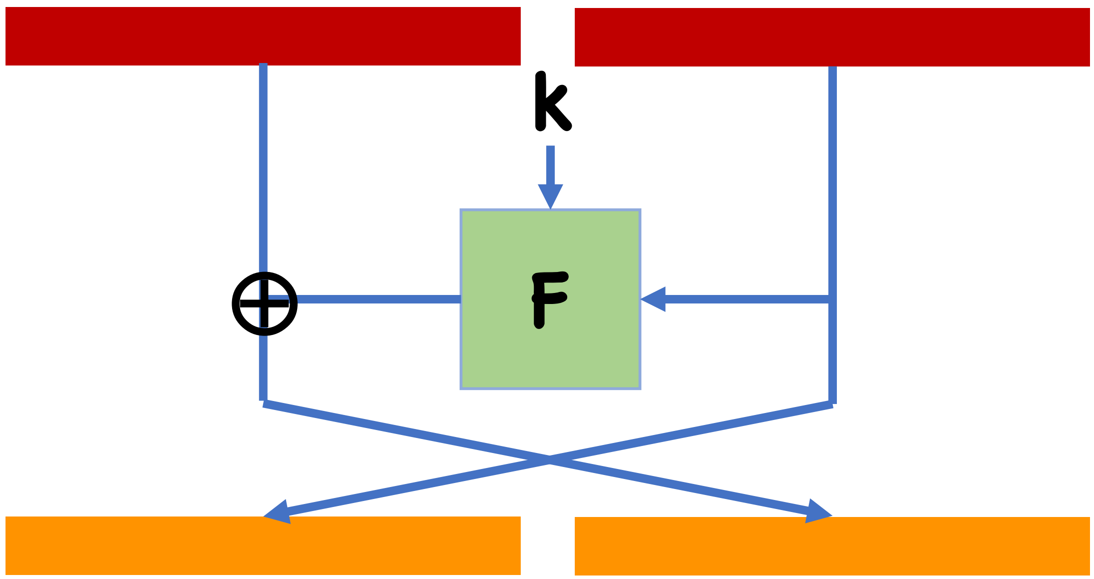
Feistel Networks

Designing permutations with good security properties is hard

What if instead we could built a good permutation from a function with good security properties…

Feistel Network

Convert functions into permutations

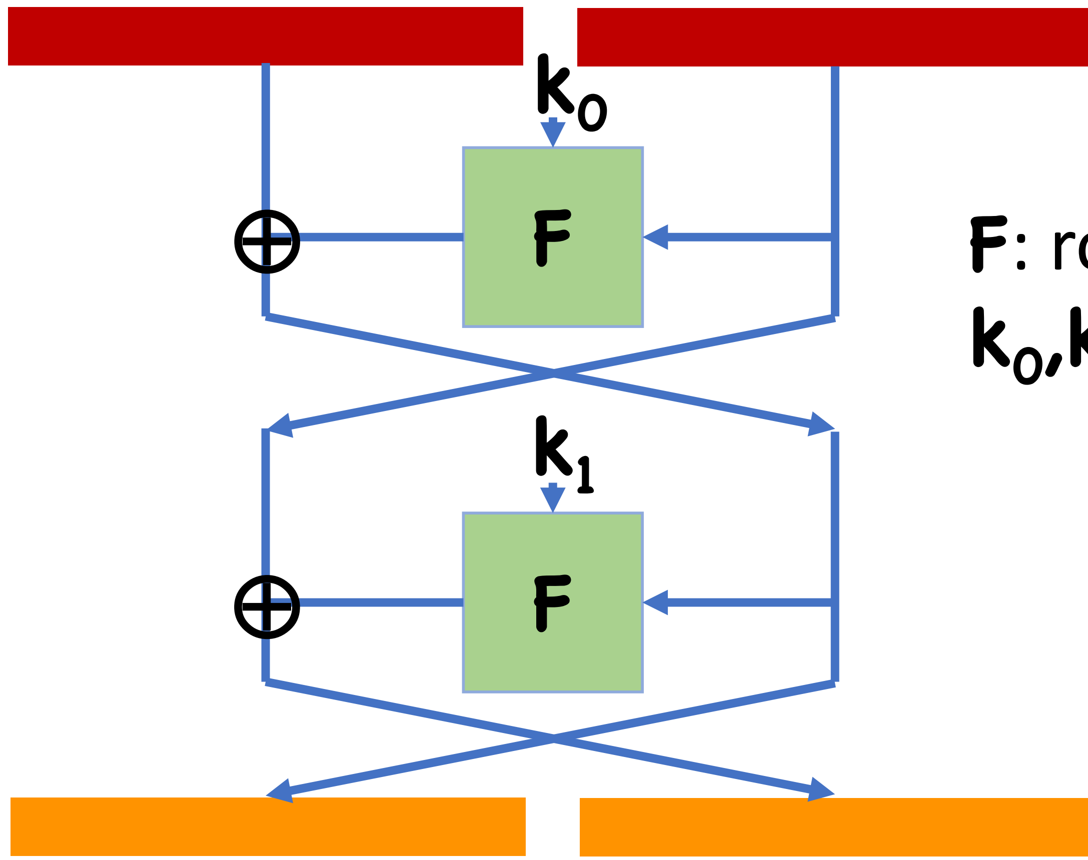


F public
k secret

Can this possibly give a secure PRP?

Feistel Network

Convert functions into permutations



F: round function

k_0,k_1: round keys

Feistel Network

Depending on specifics of round function, different number of rounds may be necessary

- Number of rounds must always be at least 3
- (Need at least 4 for a strong PRP)
- Maybe need even more for weaker round functions

Luby-Rackoff

3- or 4-round Feistel where round function is a PRF

Theorem: If F is a secure PRF, then 3 rounds of Feistel (with independent round keys) give secure PRP. 4 rounds give a strong PRP

- Proof non-trivial, won't be covered in this class

Limitations of Feistel Networks

Turns out Feistel requires block size to be large

- If number of queries ~$2^{\text{block size}/2}$, can attack

Format preserving encryption:

- Encrypted data has same form as original
- E.g. encrypted SSN is an SSN
- Useful for encrypting legacy databases

Sometimes, want a very small block size

Constructing Round Functions

Ideally, ”random looking” functions

Similar ideas to constructing PRPs

- Confusion/diffusion
- SPNs, S-boxes, etc

Key advantage is that we no longer need the functions to be permutations

- S-boxes can be non-permutations

DES

Block size: 64 bits

Key size: 56 bits

Rounds: 16

!!!

DES

Key Schedule:

- Round keys are just 48-bit subsets of master key

Round function:

- Essentially an SPN network

DES S-Boxes

8 different S-boxes, each

- 6-bit input, 4-bit output
- Table lookup: 2 bits specify row, 4 specify column

- Each row contains every possible 4-bit output
- Changing one bit of input changes at least 2 bits of output

DES History

Designed in the 1970's

- At IBM, with the help of the NSA
- At the time, many in academia were suspicious of NSA's involvement
  - Mysterious S-boxes
  - Short key length
- Turns out, S-box probably designed well
  - Resistant to "differential cryptanalysis"
  - Known to IBM and NSA in 1970's, but kept secret
- Essentially only weakness is the short key length
  - Maybe secure in the 1970's, definitely not today

DES Security Today

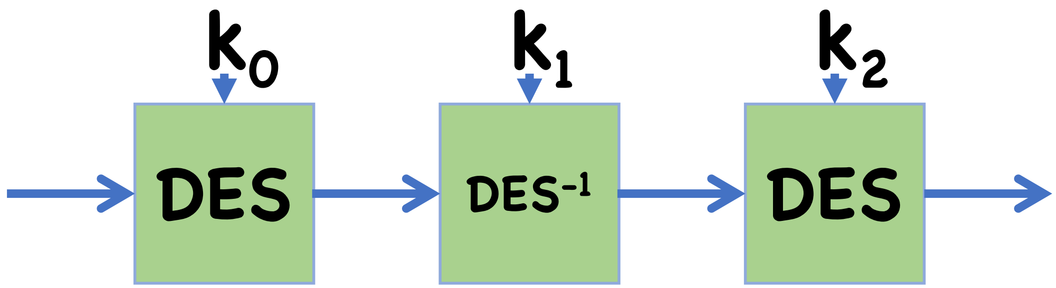
Seems like a good cipher, except for its key length and block size

What's wrong with a small block size?

- Remember for e.g. CTR mode, IV is one block
- If two identical IV's seen, attack possible
- After seeing q ciphertext, probability of repeat IV is roughly $\mathbf{q^2/2^{\text{block length}}}$
- Attack after seeing ≈ billion messages

3DES: Increasing Key Length

3DES key = Apply DES three times with different keys



Why three times?

- Next time: “meet in the middle attack” renders 2DES no more secure than 3DES

Why inverted second permutation?

Attacks on block ciphers

Brute Force Attacks

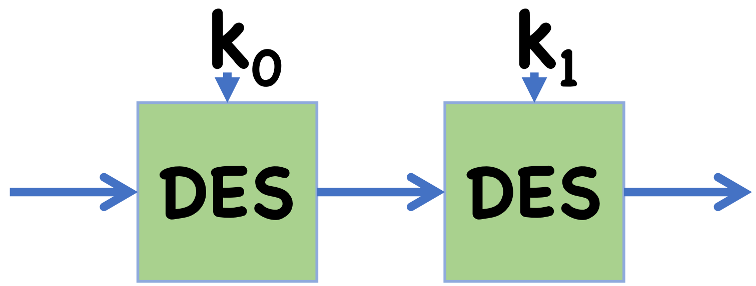
Suppose attacker is given a few input/output pairs

Likely only one key could be consistent with this input/output

Brute force search: try every key in the key space, and check for consistency

Attack time: $2^{\text{key length}}$

Insecurity of 2DES



DES key length: 56 bits

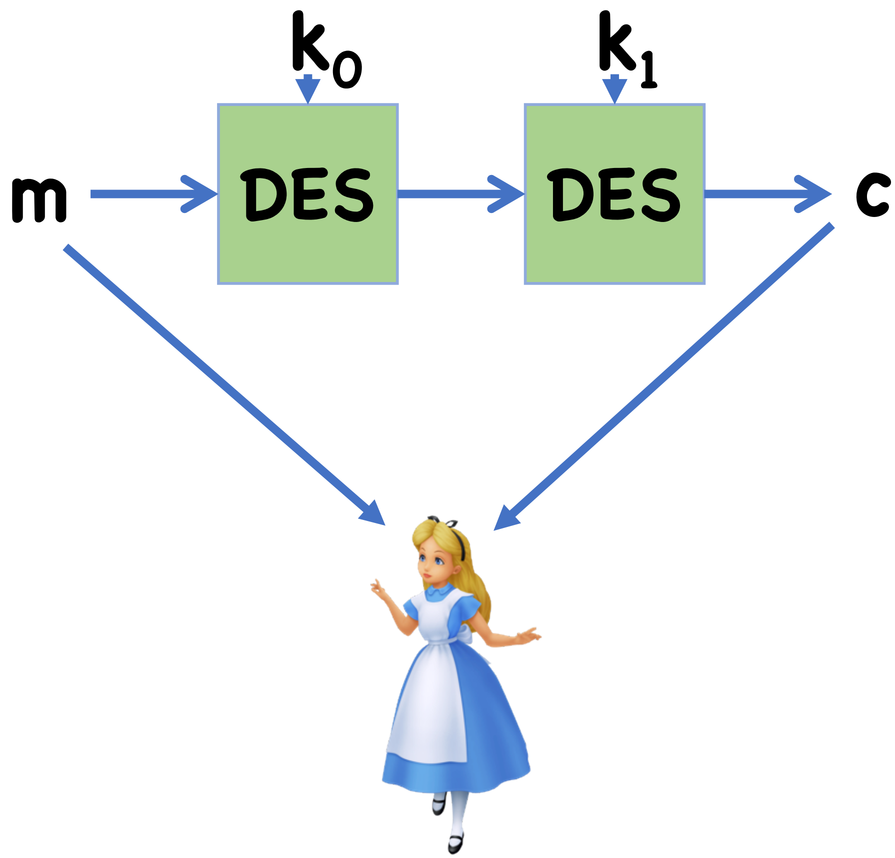
2DES key length: 112 bits

Brute force attack running time: 2^{112}

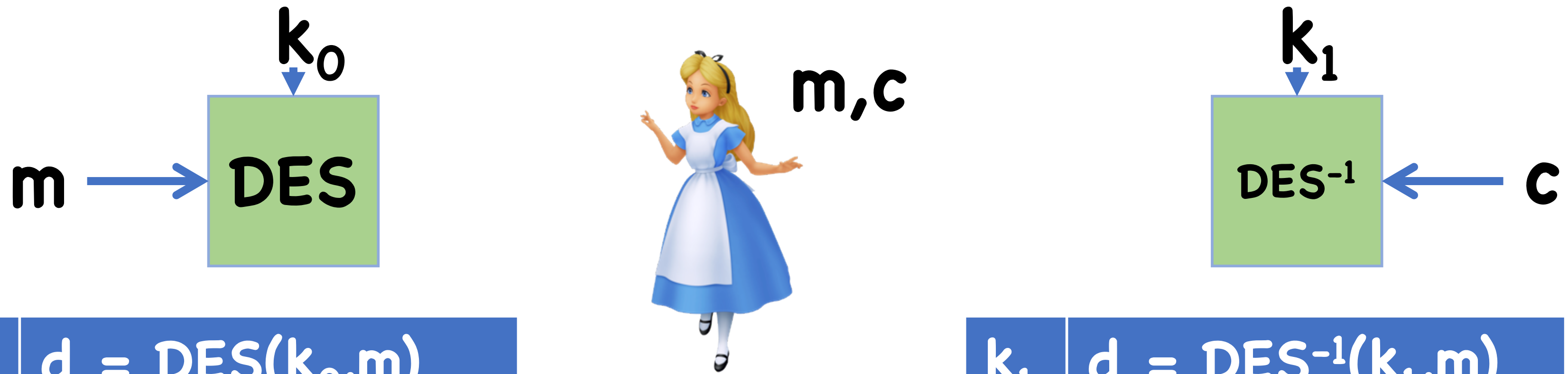
Meet In The Middle Attacks

For 2DES, can actually find key in 2^{56} time

- Also $\approx 2^{56}$ space



Meet In The Middle Attacks



k_0	$d = DES(k_0,m)$
0	52
1	93
2	03
3	96
4	20
5	49
…	…

k_1	$d = DES^{-1}(k_1,m)$
0	69
1	10
2	86
3	49
4	99
5	08
…	…

Meet In The Middle Attacks

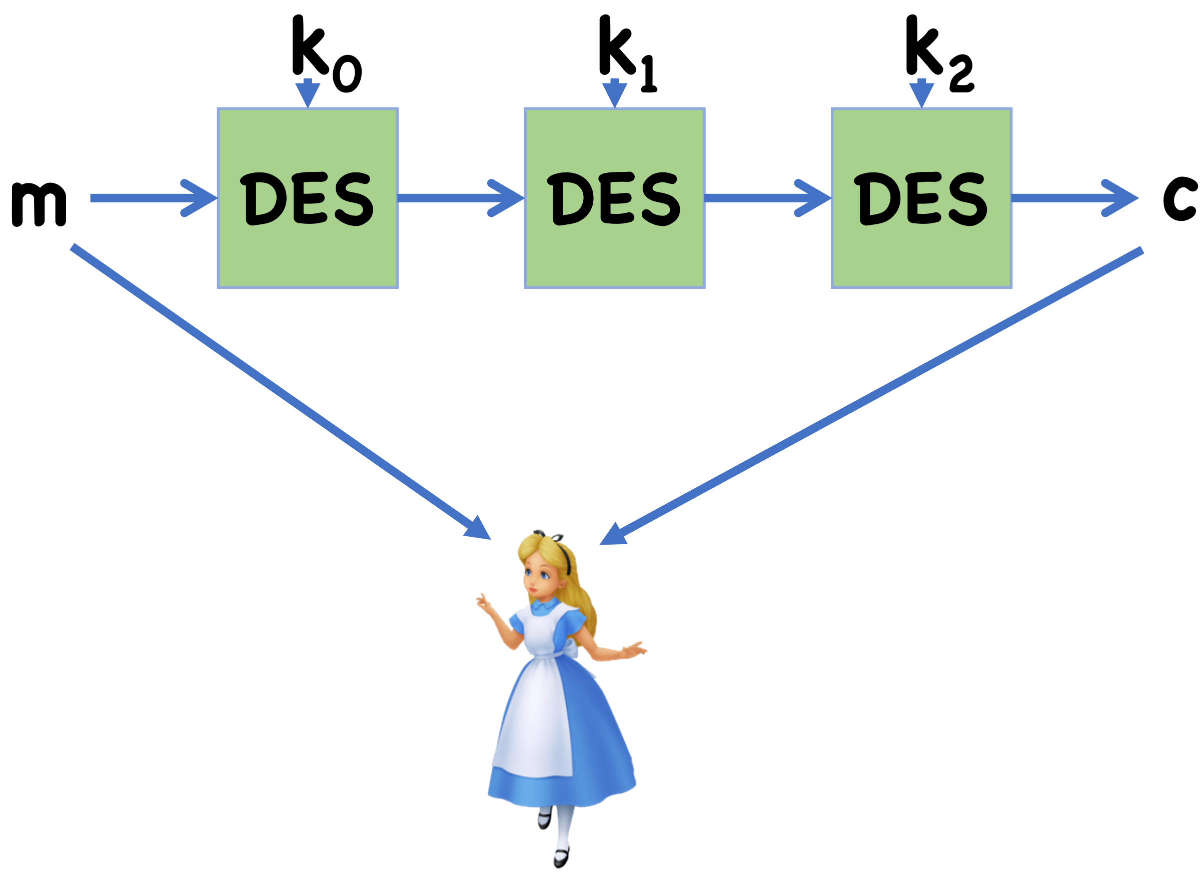
Complexity of meet in the middle attack:

- Computing two tables: time, space $2 \times 2^{\text{key length}}$
- Slight optimization: don't need to actually store second table

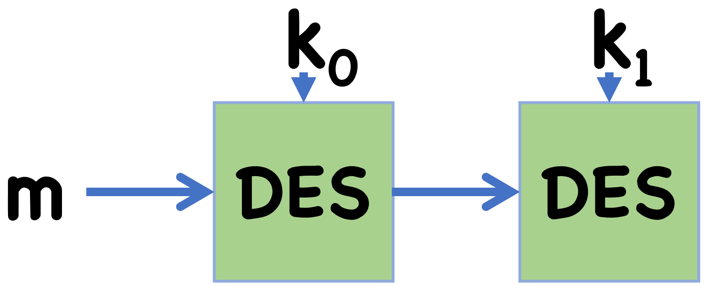
On 2DES, roughly same time complexity as brute force on DES

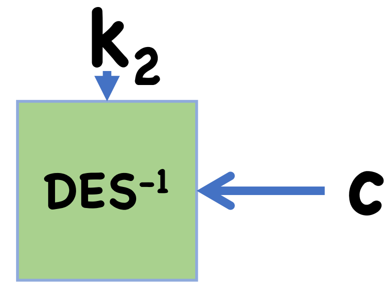
MITM Attacks on 3DES

MITM attacks also apply to 3DES…



MITM for 3DES





k_0	k_1	$d = DES(k_0,m)$
0	0	52
0	1	93
…	…	03
5	6	96
5	7	20
5	8	49
…		…

k_2	$d = DES^{-1}(k_2,m)$
0	69
1	10
2	86
3	49
4	99
5	08
…	…

MITM for 3DES

No matter where “middle” is, need to have two keys on one side

- Must go over 2^{112} different keys

Space?

While 3DES has 168 bit keys, effective security is 112 bits

Generalizing MITM

In general, given r rounds of a block cipher with t-bit keys,

- Attack time: $2^{t\lceil r/2 \rceil}$
- Attack space: $2^{t\lfloor r/2 \rfloor}$

Brute Force vs. Generic Attacks

MITM attacks on iterated block ciphers are *generic*

- Attack exists independent of implementation details of block cipher

However, still beats a *brute force*

- Doesn’t simply try every key

Next time

More attacks on block ciphers

Reminders

HW2 due **tomorrow**

Project 1 due next week