COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

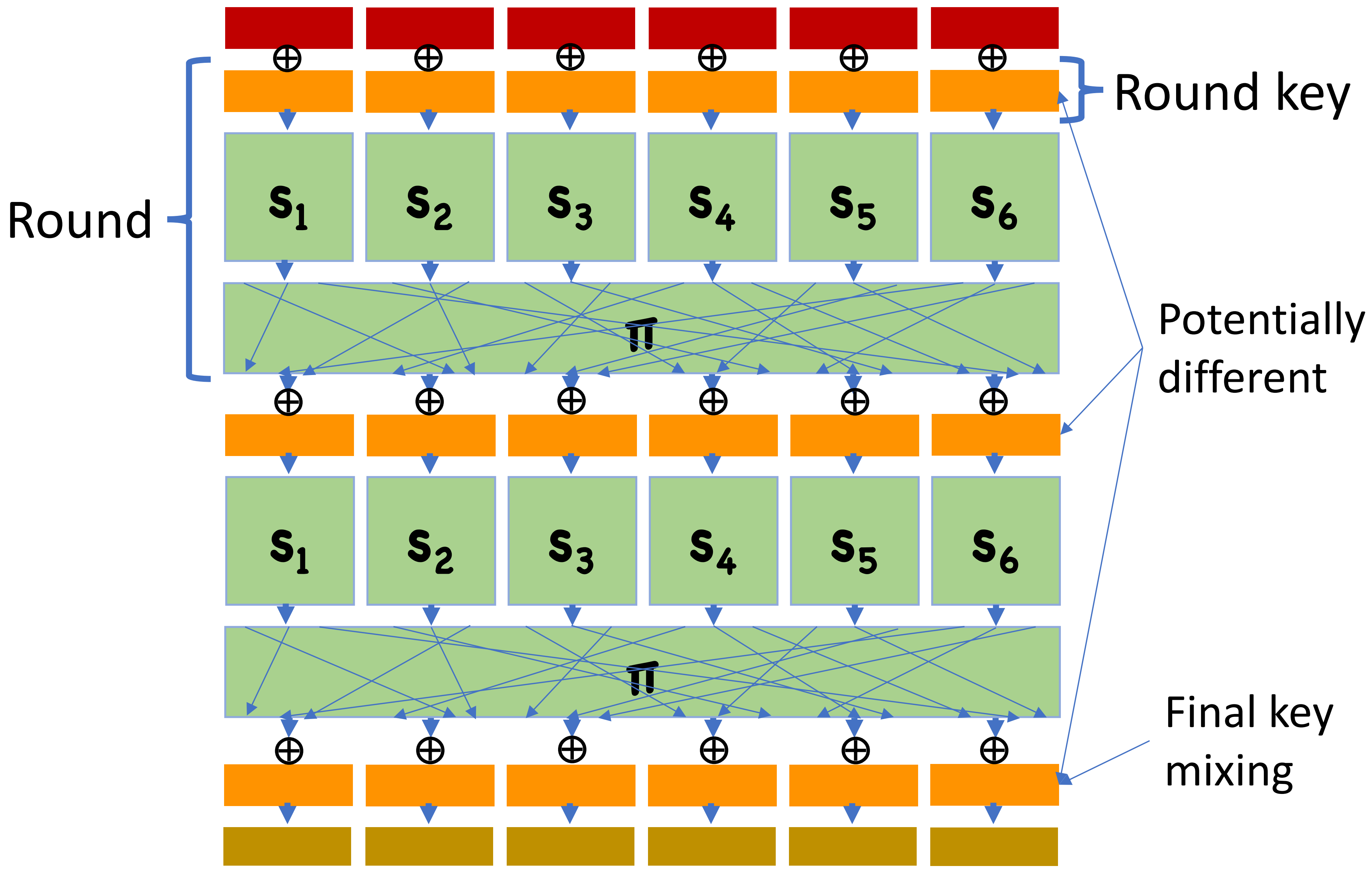
Announcements

HW 4 Due Tomorrow

No homework this week

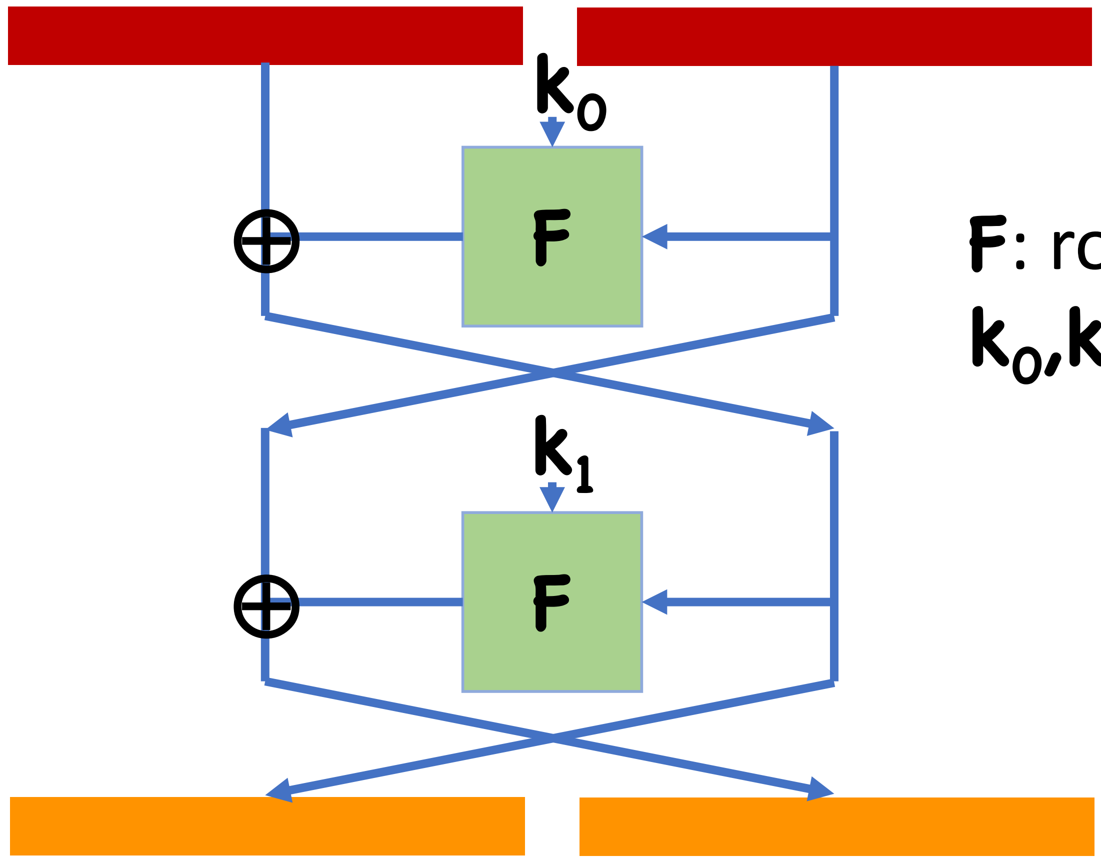
- Instead, prep for take-home midterm

Recap: SPN Netoworks



Recap: Feistel Network

Convert functions into permutations



F: round function

$\mathbf{k_0,k_1}$: round keys

Today

Attacks on Block Ciphers

Brute Force Attacks

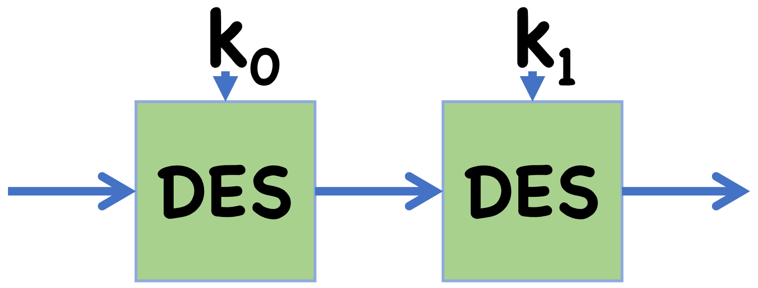
Suppose attacker is given an input/output pair

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

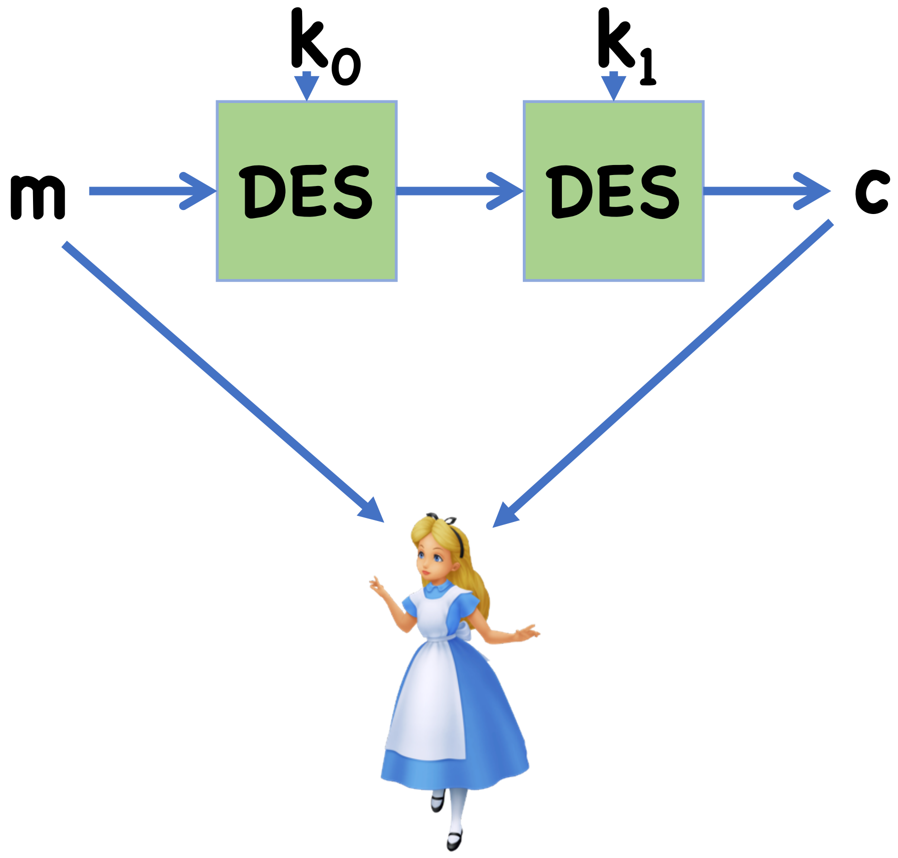
2DES key length: 112 bits

Brute force attack running time: 2^{112}

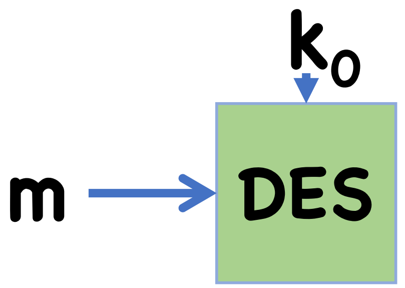
Meet In The Middle Attacks

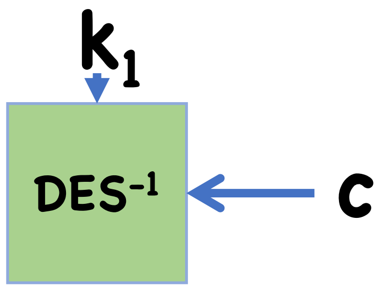
For 2DES, can actually find key in 2^{56} time

- 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

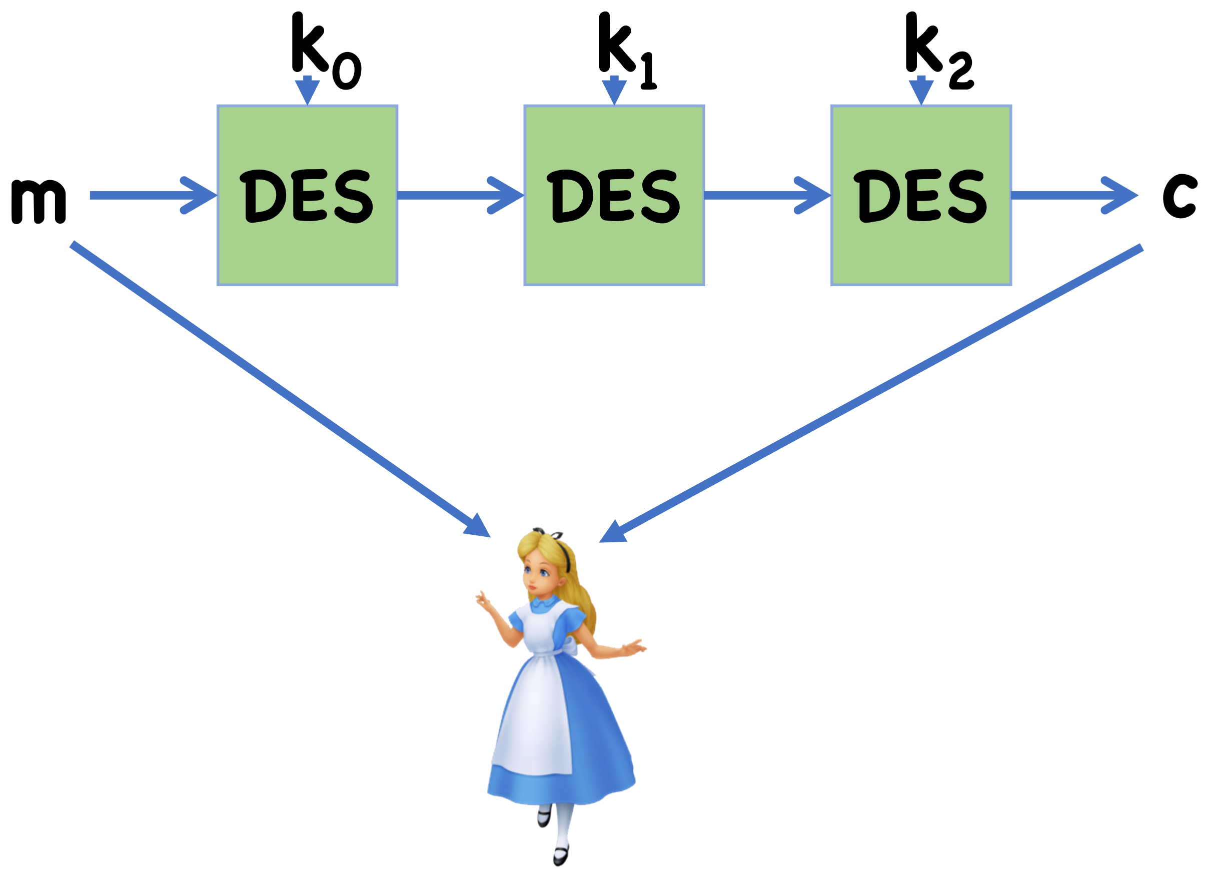
Complexity of meet in the middle attack:

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

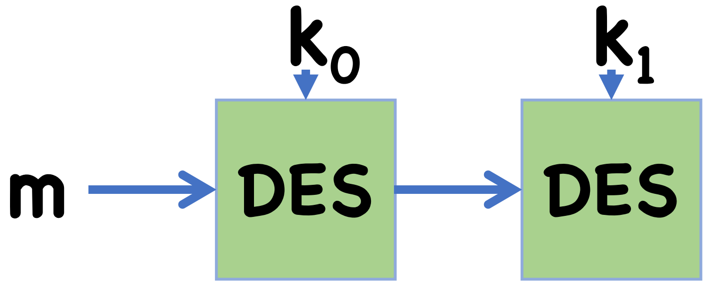
On 2DES, roughly same time complexity as brute force on DES

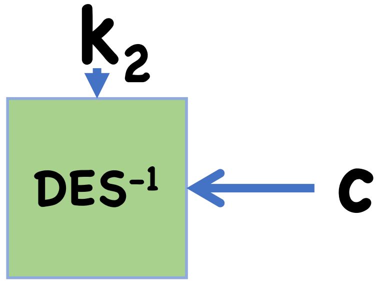
MITM Attacks on 3DES

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

Generating MITM

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

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

Brute Force vs. Generic Attacks

MITM attacks on iterated block ciphers are *generic*

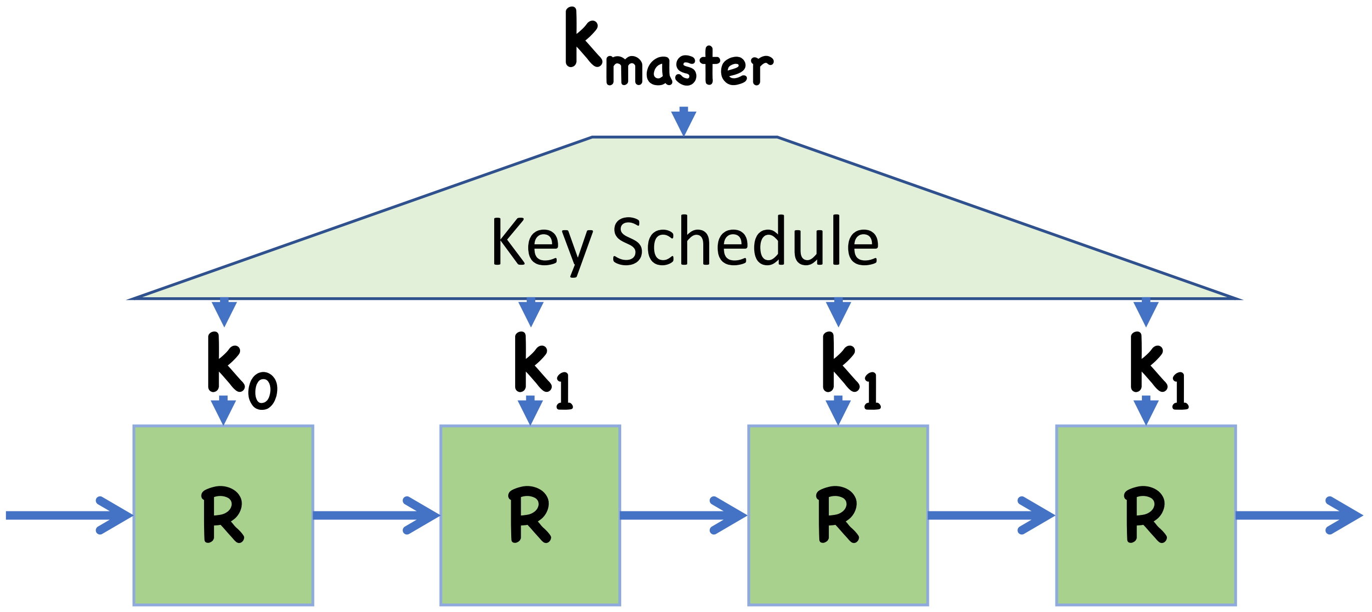
- Attack exists independent of implementation details of block cipher

However, still beats a *brute force*

- Doesn’t simply try every key

MITM Attacks

MITM attacks can also be applied to plain single block ciphers



Can yield reasonable attacks if the key schedule produces highly independent round keys

Time-Space Tradeoffs

MITM attack requires significant space

In contrast, brute force requires essentially no space, but runs slower

Known as a time-space tradeoff

Another Time-Space Trade-off Example

Given $y=F(k,x)$, find x

- Allowed many queries to $F(k,x)$ oracle
  (That is, standard block cipher oracle)
- Assume $|k| \gg |x|$

Option 1:

- Brute force search over entire domain looking for x
- Time: 2^l
- Space: 1

Another Time-Space Trade-off Example

Given **y=F(k,x)**, find **x**

- Allowed many queries to **F(k,x)** oracle
  (That is, standard block cipher oracle)
- Assume **|k| >> |x|**

Option 2: Preprocessing

- Before seeing **y**, compute giant table of **(x,F(k,x))** pairs, sorted by **F(k,x)**
- Preprocessing Time: 2^l
- Space: 2^l
- Online time: ???

Option 3: Hellman's Attack

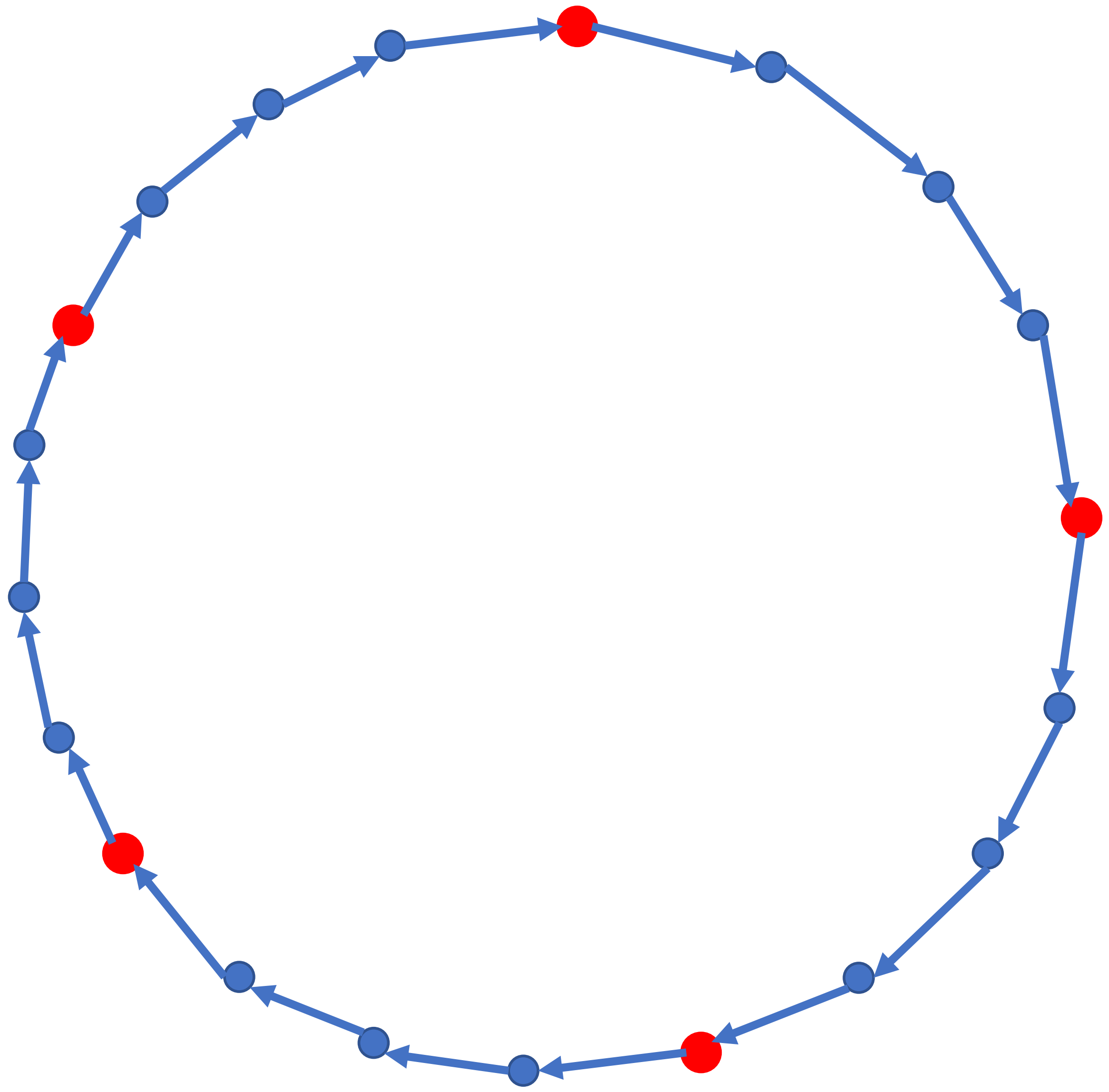
For simplicity, assume $F(k,\bullet)$ forms a cycle covering entire domain

- $\{0, F(k,0), F(k, F(k,0)), F(k, F(k, F(k,0))),\ldots\} = X$

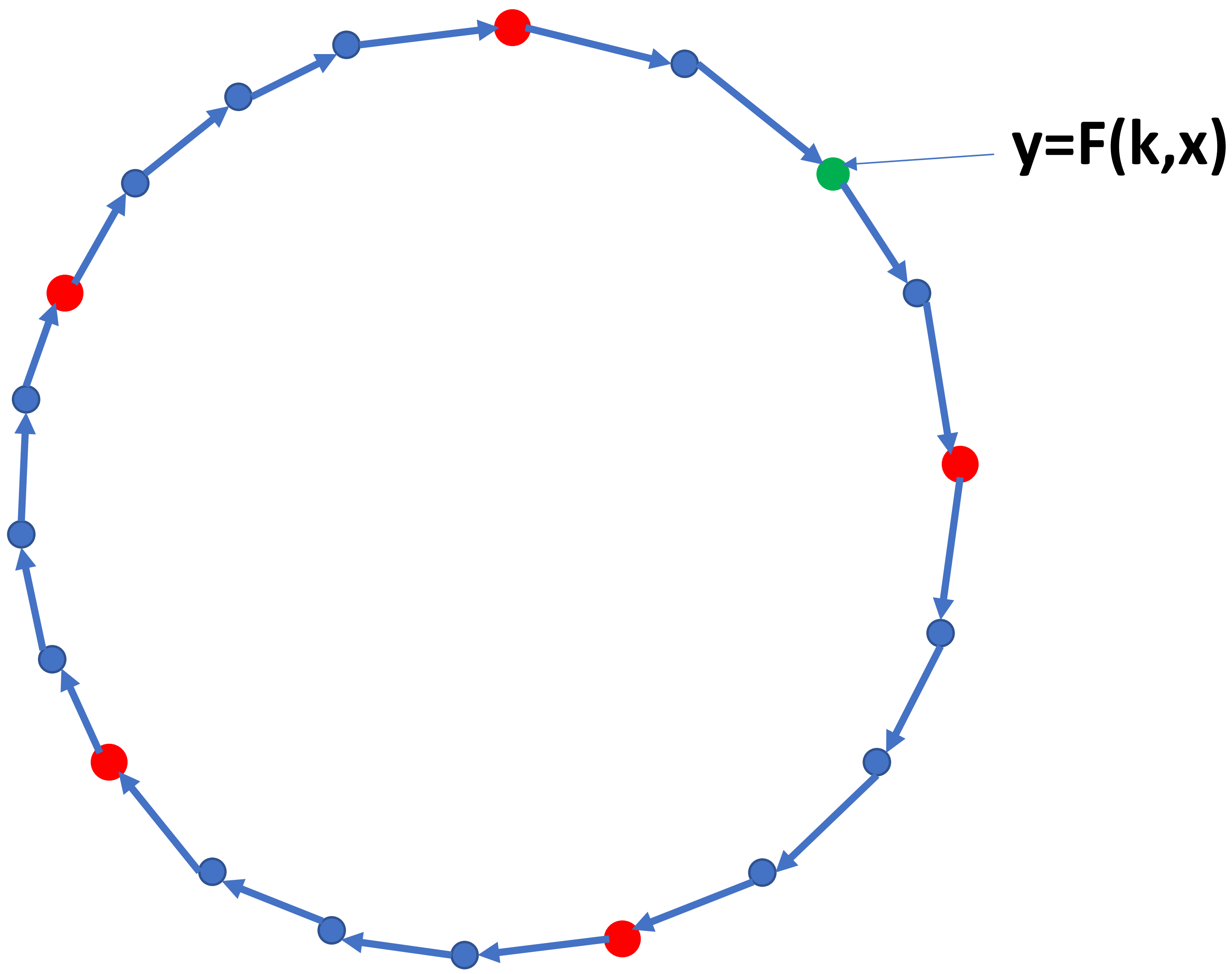
In preprocessing stage:

- Attacker iterates over entire cycle, saving every t^{th} term in a table $(x_1,\ldots,x_{N/t})$ where $N=2^l$

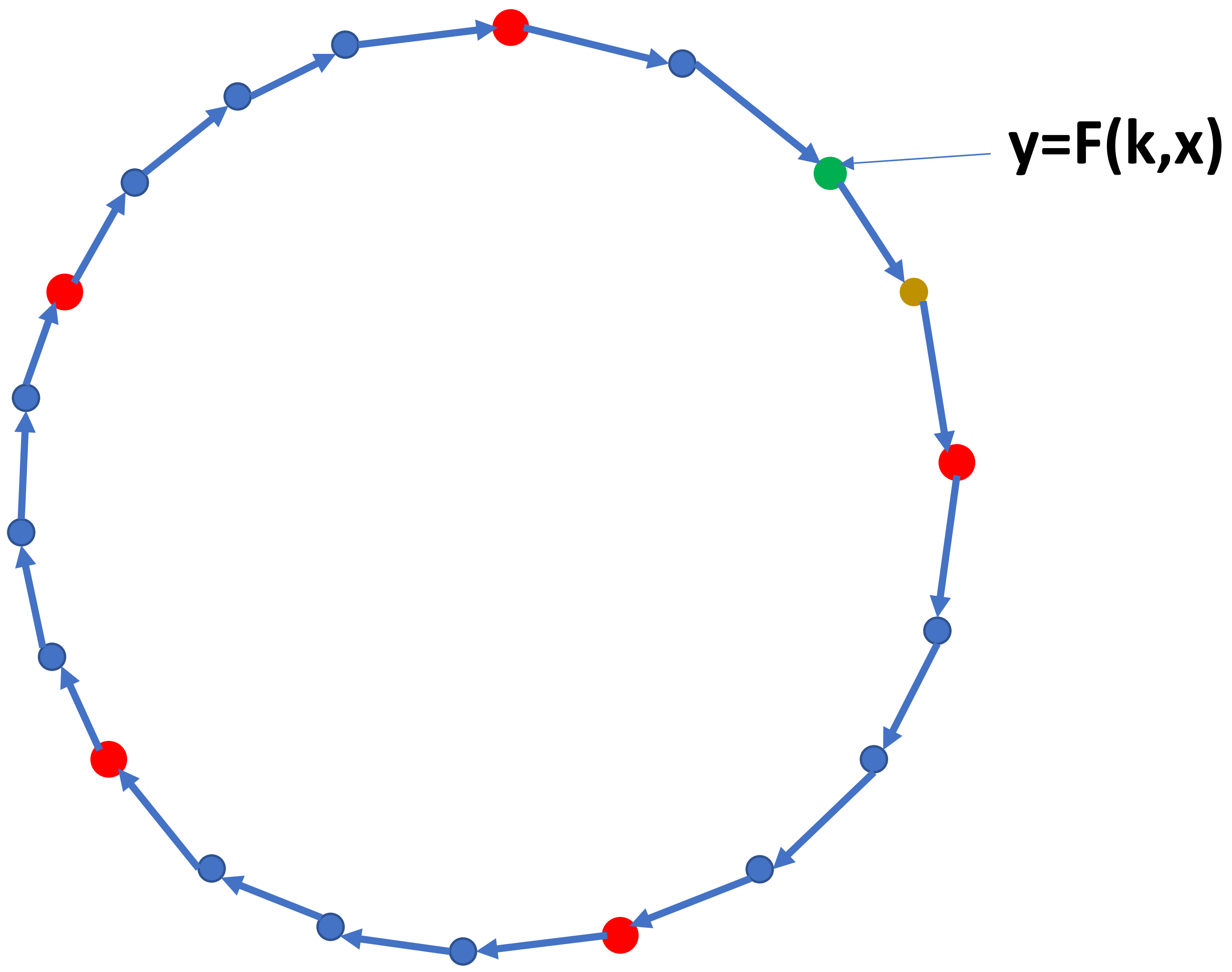
Option 3: Hellman’s Attack



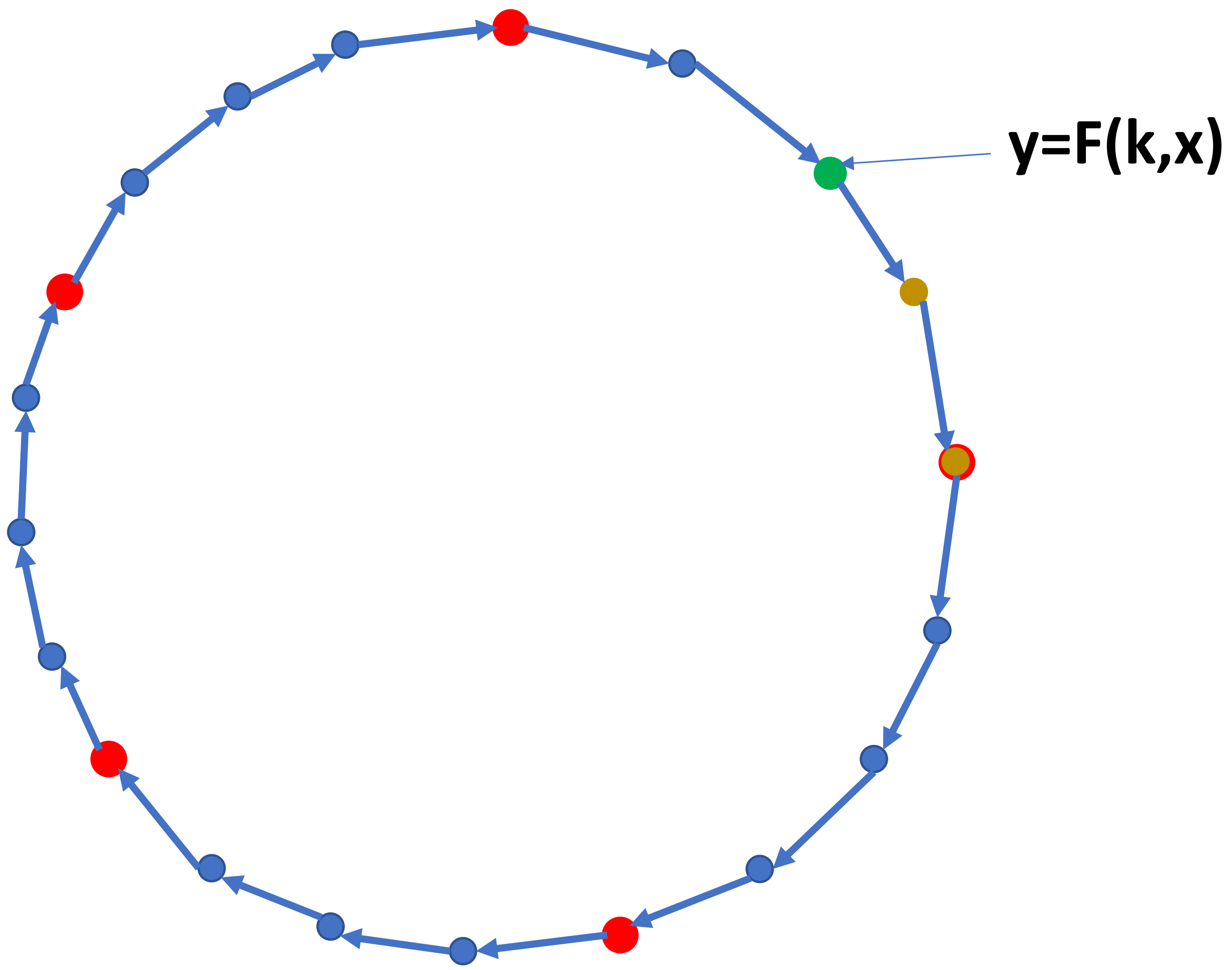
Option 3: Hellman’s Attack



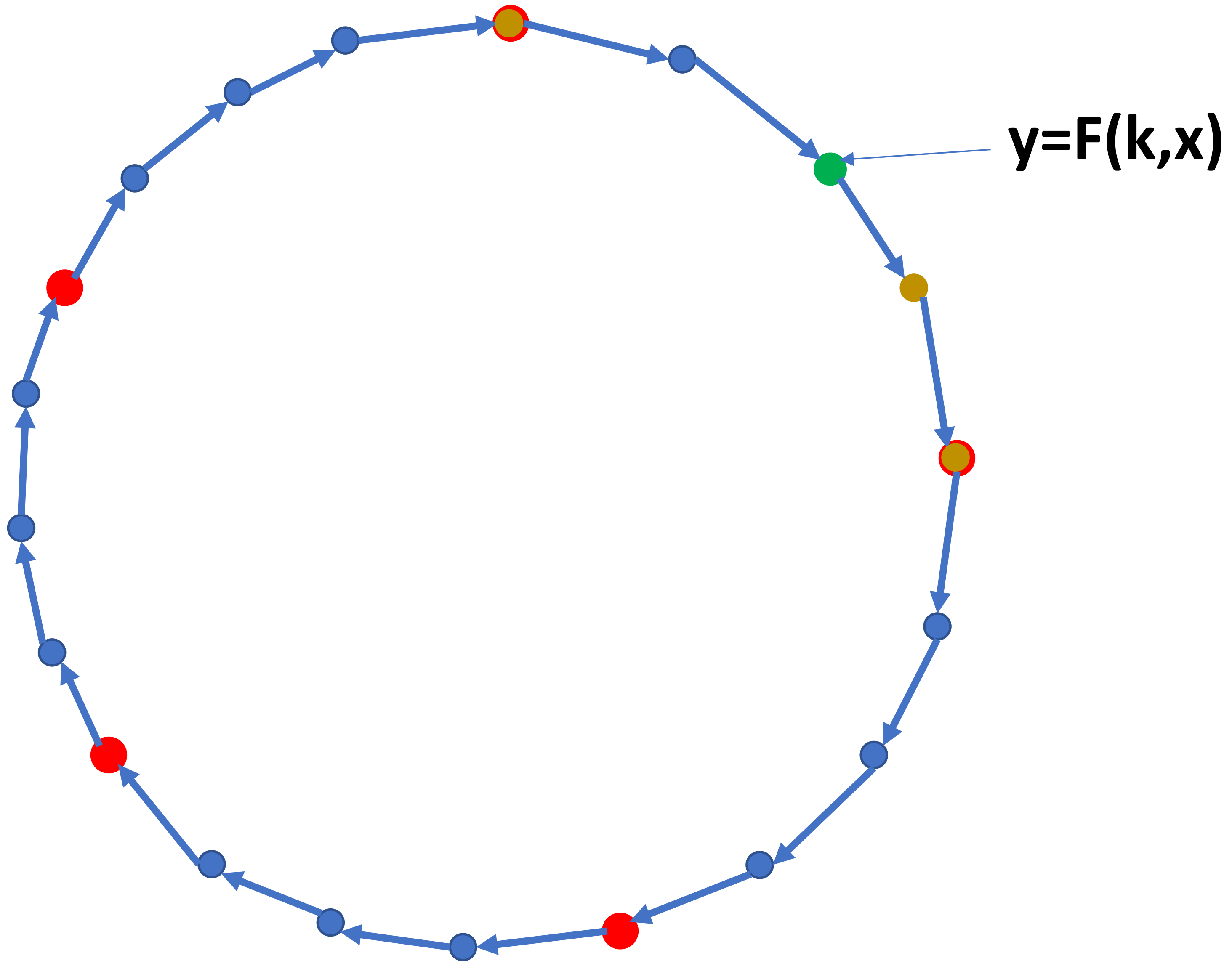
Option 3: Hellman’s Attack



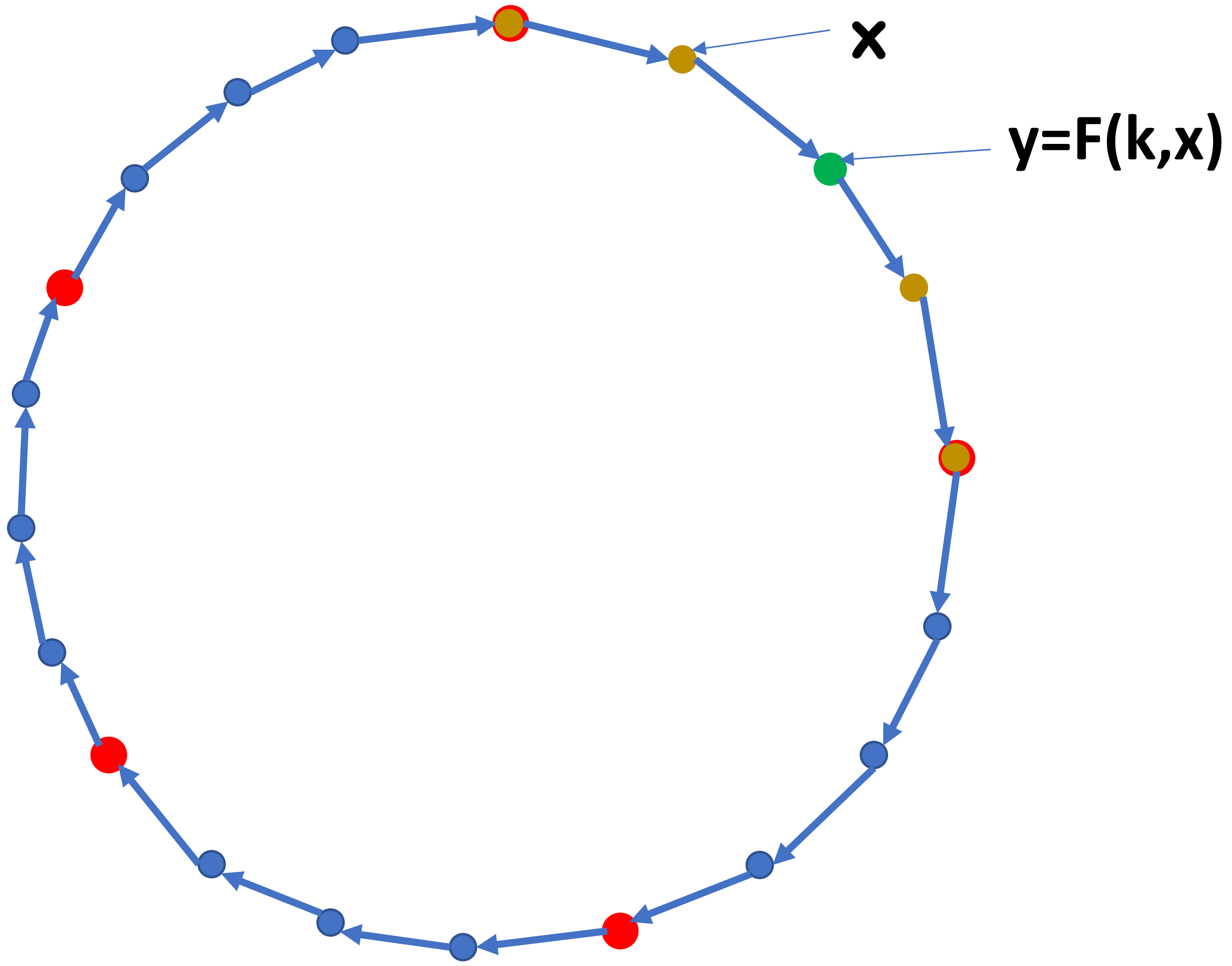
Option 3: Hellman’s Attack



Option 3: Hellman's Attack



Option 3: Hellman's Attack



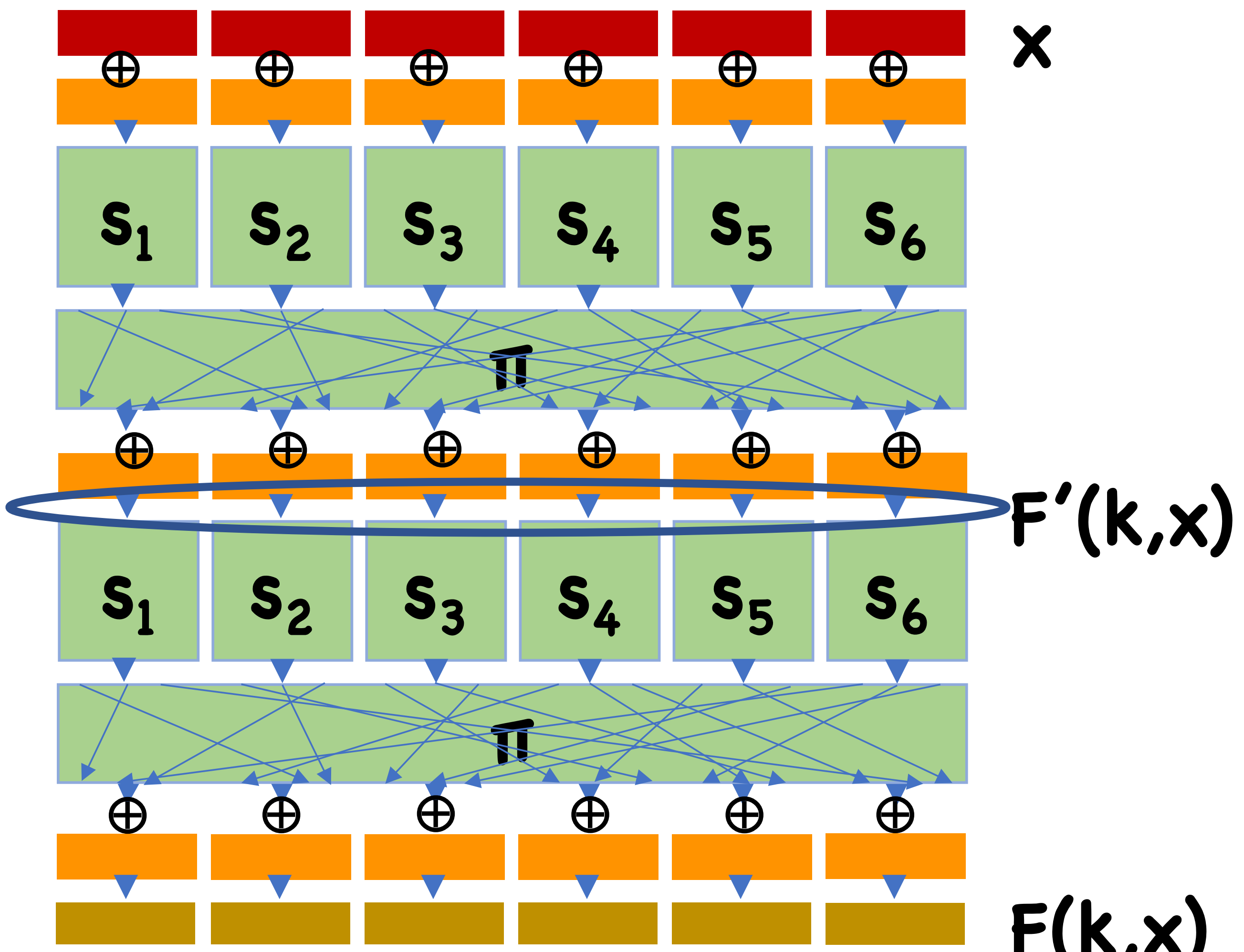
Option 3: Hellman's Attack

Preprocessing Time:	$N=2^l$
Space:	N/t
Online Time:	t

Time-space tradeoff: **space × online time ≈** N

For non-cycles, attack is a bit harder, but nonetheless possible

Differential Cryptanalysis



Differential Cryptanalysis

Suppose there were Δ_x, Δ_z such that, for random key k and random x_1, x_2 where $x_1 \oplus x_2 = \Delta_z$, $F'(k,x_1) \oplus F'(k,x_2) = \Delta_z$ with probability $p \gg 2^{-l}$

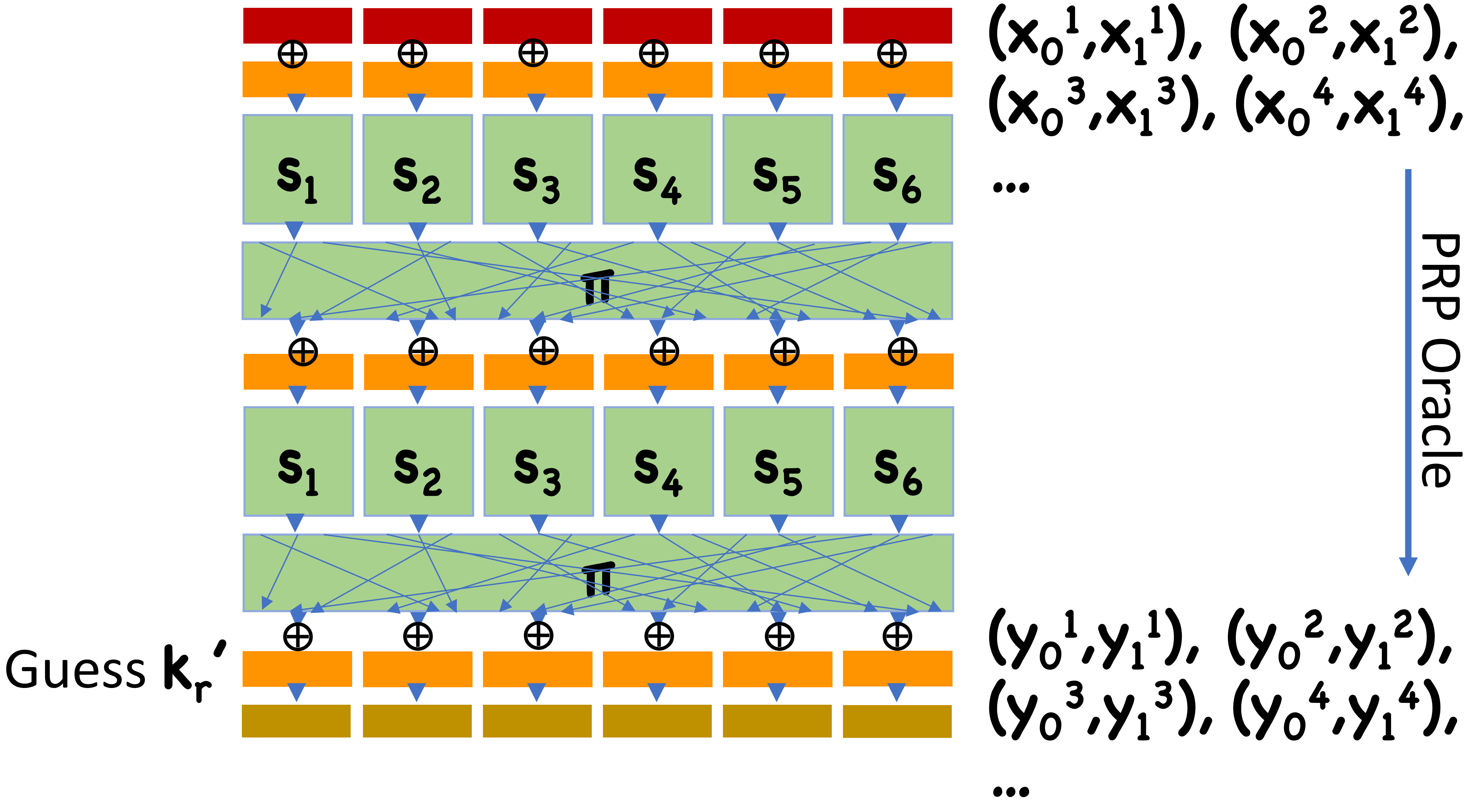
- Call (Δ_x, Δ_z) a differential
- p is probability of differential
- 2^{-l} is probability for random permutation

Differential Cryptanalysis

Attack:

- Choose many random pairs (x_1,x_2) s.t. $x_1 \oplus x_2 = \Delta_y$
- Make queries on each $x_1,\ x_2$, obtaining y_1,y_2
- For each round key guess k_r',
  - Use differentials to determine if guess was correct

Differential Cryptanalysis



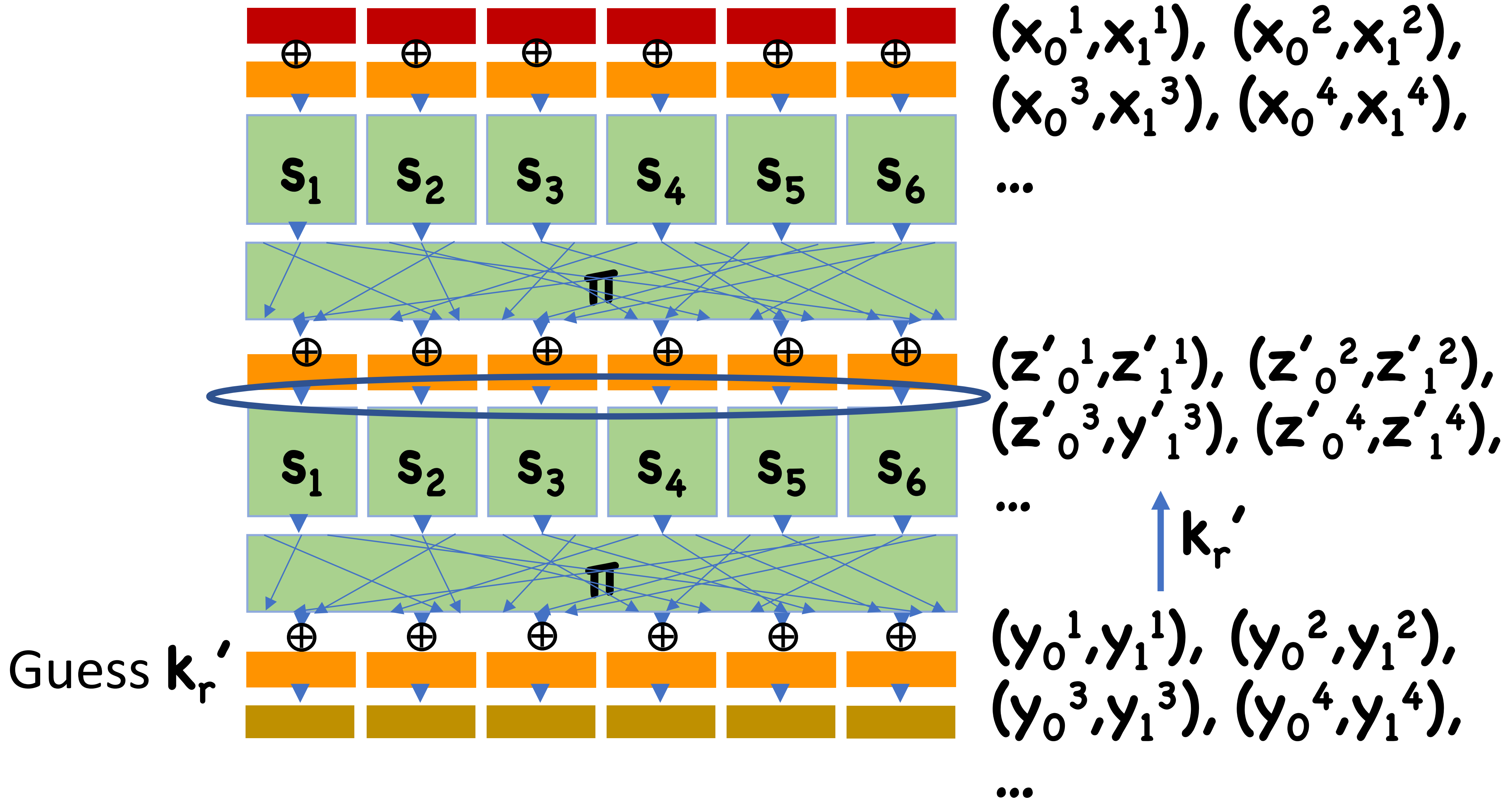
(x_0^1, x_1^1), (x_0^2, x_1^2), (x_0^3, x_1^3), (x_0^4, x_1^4), …
PRP Oracle
(y_0^1, y_1^1), (y_0^2, y_1^2), (y_0^3, y_1^3), (y_0^4, y_1^4), …
Guess $\mathbf{k}_r'$
S_1 S_2 S_3 S_4 S_5 S_6
π
$\oplus$

Differential Cryptanalysis

Attack:

- Choose many random pairs (x_1,x_2) s.t. $x_1 \oplus x_2 = \Delta_x$
- Make queries on each x_1, x_2, obtaining y_1,y_2
- For each round key guess k_r',
    - Undo last round assuming k_r', obtaining (z_1',z_2')
    - Look for $z_1' \oplus z_2' = \Delta_z$
    - If right guess, expect $\approx$ p fraction
    - If wrong guess, expect $\approx$ 2^{-l} fraction

Differential Cryptanalysis


S_1 S_2 S_3 S_4 S_5 S_6
π
S_1 S_2 S_3 S_4 S_5 S_6
π
$\oplus$
Guess $\mathbf{k_r}'$
(x_0^1,x_1^1), (x_0^2,x_1^2), (x_0^3,x_1^3), (x_0^4,x_1^4), …
$(z'_0{}^1,z'_1{}^1)$, $(z'_0{}^2,z'_1{}^2)$, $(z'_0{}^3,y'_1{}^3)$, $(z'_0{}^4,z'_1{}^4)$, …
k_r'
(y_0^1,y_1^1), (y_0^2,y_1^2), (y_0^3,y_1^3), (y_0^4,y_1^4), …

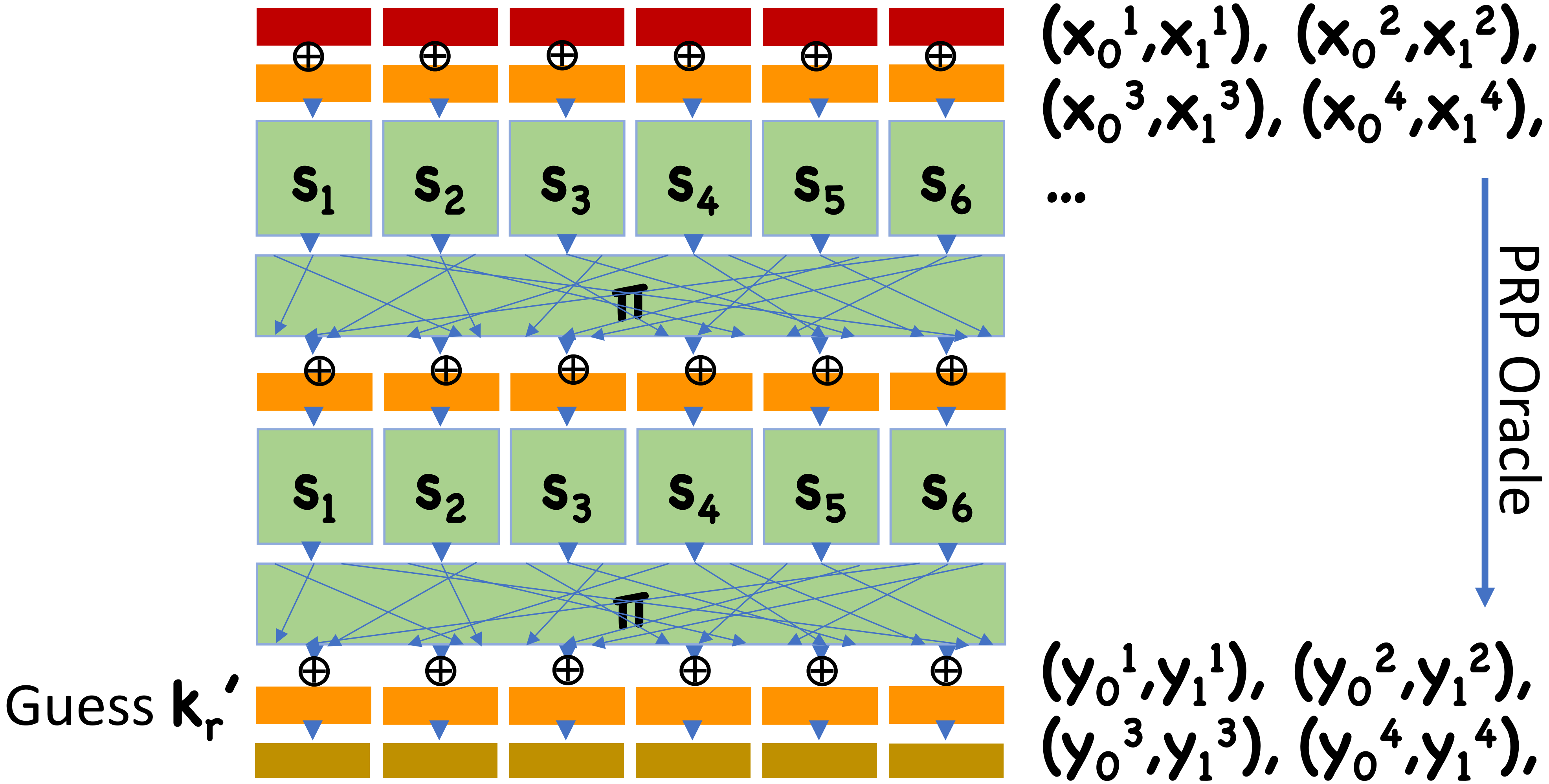
Differential Cryptanalysis

So far, inefficient since we have to iterate over all $\mathbf{2^l}$ possible round keys

Instead, we can learn $\mathbf{k_r}$ byte by byte

- Guess 8 bits of $\mathbf{k_r}$ at a time
- Iterate through all 2^8 possible values for those 8 bits
    - Compute 8 bits of $\mathbf{z_1', z_2'}$, look for (portion of) differential
- Which bits to choose?

Differential Cryptanalysis



(x_0^1, x_1^1), (x_0^2, x_1^2), (x_0^3, x_1^3), (x_0^4, x_1^4), …

(y_0^1, y_1^1), (y_0^2, y_1^2), (y_0^3, y_1^3), (y_0^4, y_1^4), …

Differential Cryptanalysis

Extending to further levels:

- One $\mathbf{k_r}$ is known, can uncompute last layer
- Now perform same attack on round-reduced cipher
- Repeat until all round keys have been found

Finding Differentials

So far, assumed differential given

How do we find it?
- Can’t simply brute force all possible differentials

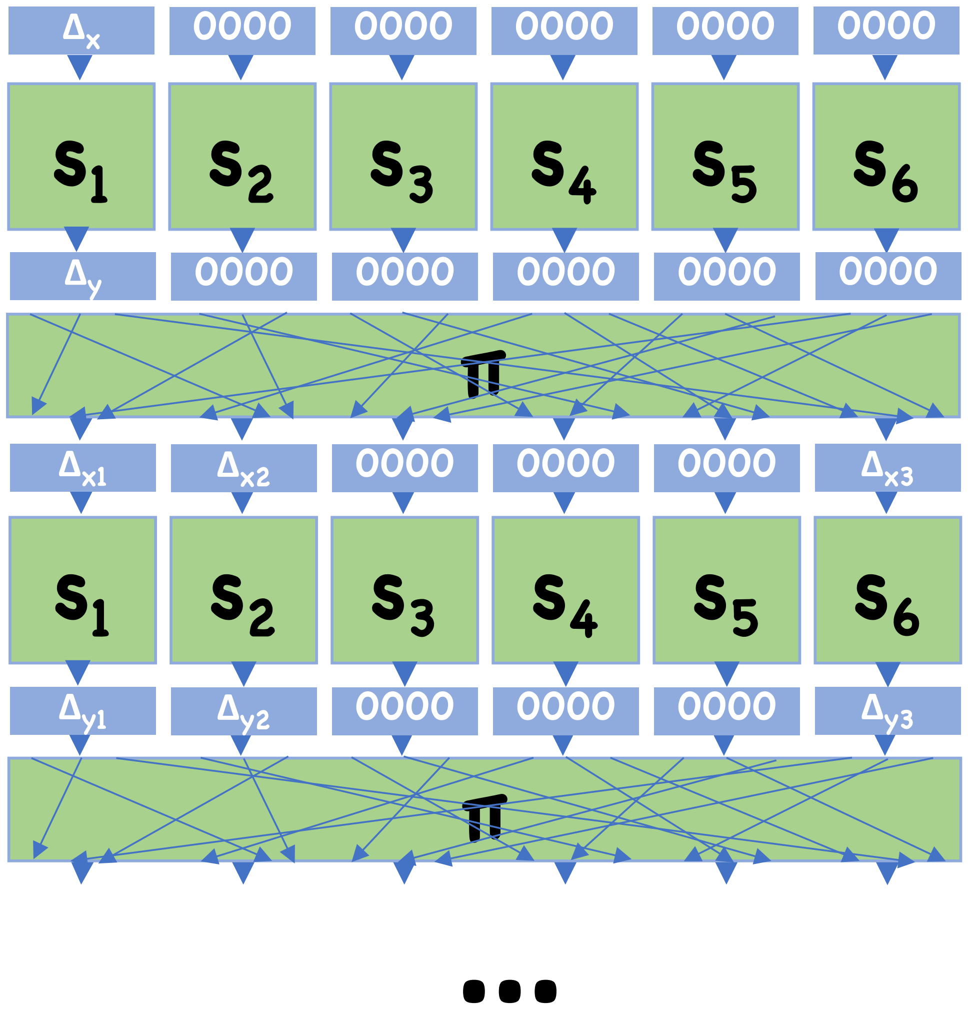
Finding Differentials

Solution: look for differentials in S-boxes

- Only 2^8 possible differences, so we can actually look for all possible differentials

- Then trace differentials through the evaluation
  - Key mixing does not affect differentials
  - Diffusion steps just shuffle differential bits

Differential Cryptanalysis



Differential Cryptanalysis in Practice

Used to attack real ciphers

- FEAL-8, proposed as alternative to DES in 1987
  - requires just 1000 chosen input/output pairs, 2 minutes computation time in 1990’s

- Also theoretical attacks on DES
  - Requires 2^{47} chosen input/output pairs
  - Very difficult to obtain in real world applications
  - However, small changes to S-boxes in DES lead to much better differential attacks

Linear Cryptanalysis

High level idea: look for linear relationships that hold with too-high a probability

- E.g. $x_1 \oplus x_5 \oplus x_{17} \oplus y_3 \oplus y_6 \oplus y_{12} \oplus y_{21} = 0$

Can show that if happen with too-high probability, can completely recover key

Important feature: only requires *known* plaintext as opposed to *chosen* plaintext

- Much easier to carry out in practice
- Ex: DES can be broken with 2^{43} input/output pairs

Block Cipher Design

S-boxes are designed to minimize differential and linear cryptanalysis

- Cannot completely remove differentials/linear relations, but can minimize their probability

Increasing number of rounds helps

- Likelihood of differential decreases each round

Side-Channel Attacks

Cache Timing Attacks

AES evaluation performs many table lookups

- E.g. for S-box evaluation

Recall multiple levels of memory hierarchy:

- CPU registers, L0, L1, L2 cache, etc

Access to different levels require orders of magnitude different access times

Not all of AES lookup tables fix in CPU registers, so there will be frequent cache misses

- Cache misses will depend on data (e.g. **k**)

Power Analysis

Example: smart card has a DES key hardwired inside

- Gain physical access to card
- Assume “tamper proof” – cannot just break open to learn key
- However, interrogate card on various inputs
- Can also look at power usage

Power Analysis

DES’s key schedule involves rotating the master key

Possible way to rotate digits:

- Check if last bit is 1
- If so, shift to right and prepend with 1
- If not, shift to left and prepend with 0

These two operations have different power draw

- By looking at power consumption, can learn if last bit is 0 or 1

Side-Channel Attacks

Side-channel attacks even apply assuming block cipher is “secure”

- Problem: not modeled by our security experiment

Real life solution:

- Careful implementations to thwart side channels

Academic Solution:

- Try to add side-channel attacks to model

Leakage Resilient Cryptography

Attempts to provide security definitions that model real-life side channel attacks on cryptosystems

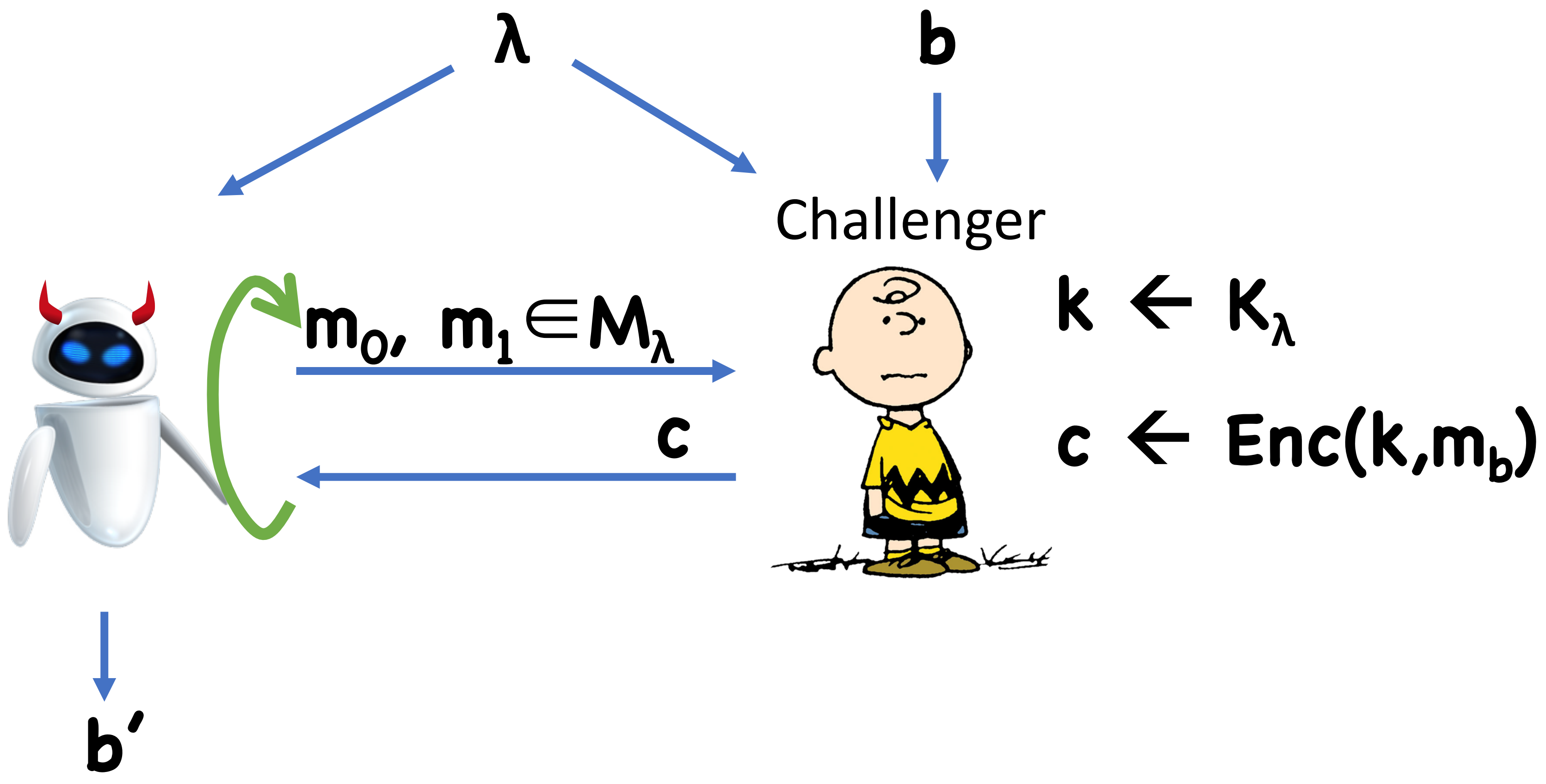
- E.g. adversary is allowed to learn certain functions of the secret key at various time steps

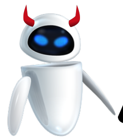
Goal: design crypto to be secure even in presence of such attacks

- Beyond scope of this course

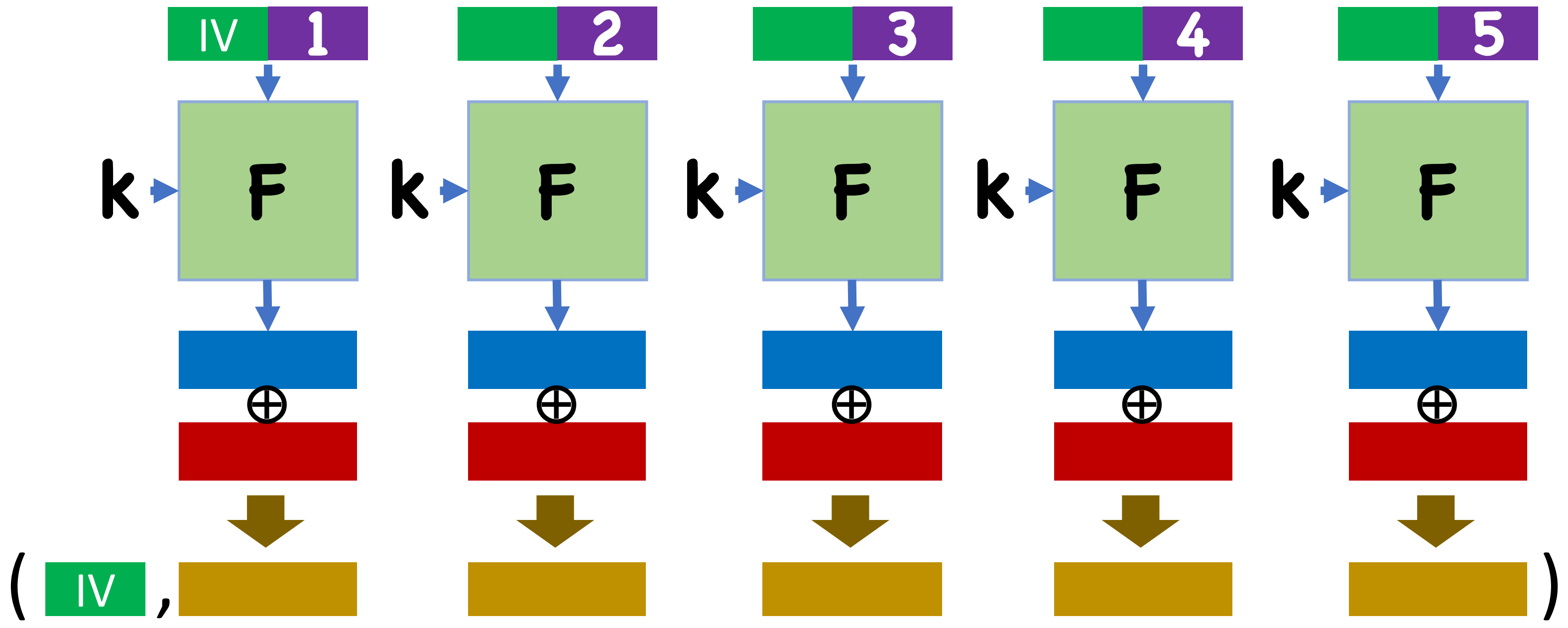
Message Integrity

Recall: CPA Security



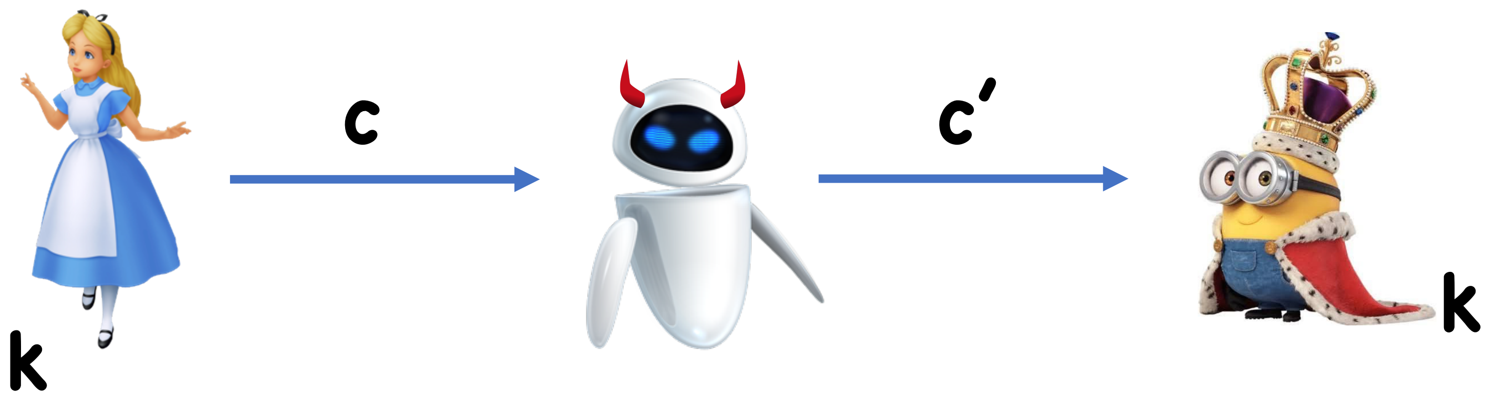
$\text{LoR-Exp}_b($  $, \lambda)$

Recall: Counter Mode (CTR)



Limitations of CPA security

attackatdawn



attackatdusk

How?

Malleability

Some encryption schemes of operation are *malleable*

- Can modify ciphertext to cause predictable changes to plaintext

Examples: basically everything we’ve seen so far

- Stream ciphers
- CTR
- CBC
- ECB
- …

Message Integrity

We cannot stop adversary from changing the message in route to Bob

However, we can hope to have Bob perform some check on the message he receives to ensure it was sent by Alice

- If check fails, Bob rejects the message

For now, we won’t care about message secrecy

- We will add it back in later

Message Authentication Codes

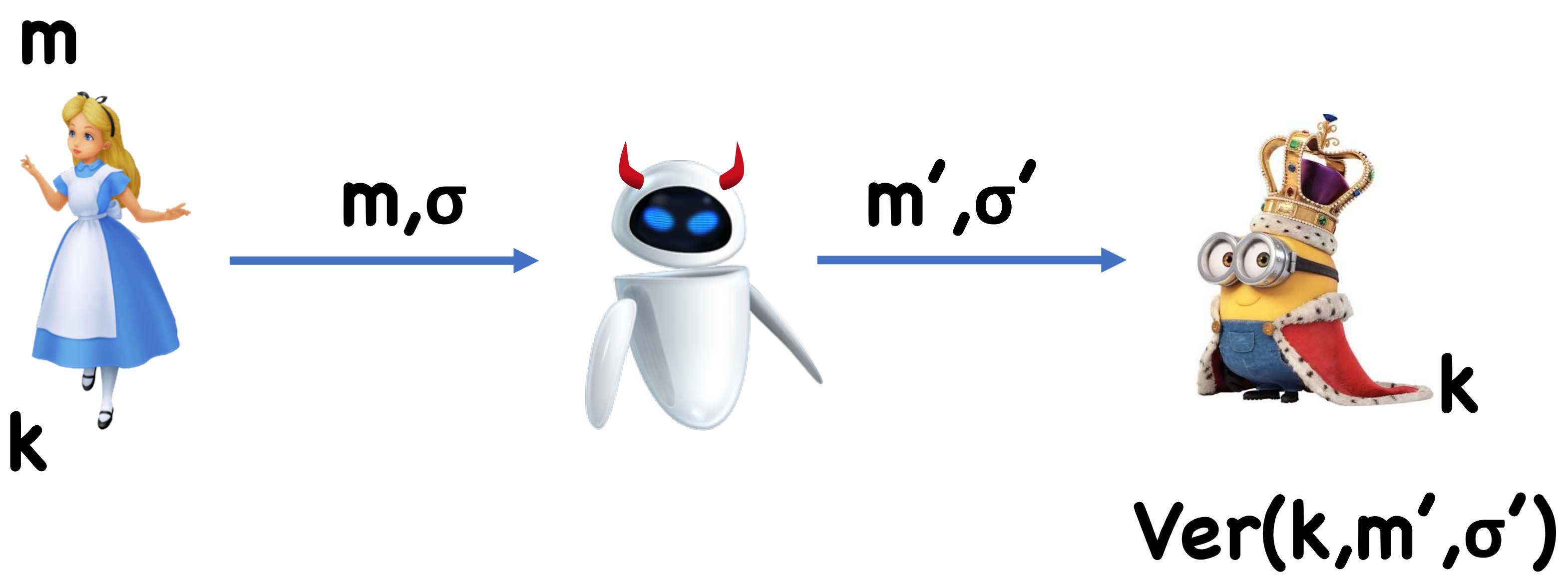
Syntax:

- Key space $\mathbf{K_\lambda}$
- Message space $\mathbf{M}$
- Tag space $\mathbf{T_\lambda}$
- $\mathbf{MAC(k,m) \rightarrow \sigma}$
- $\mathbf{Ver(k,m,\sigma) \rightarrow 0/1}$

Correctness:

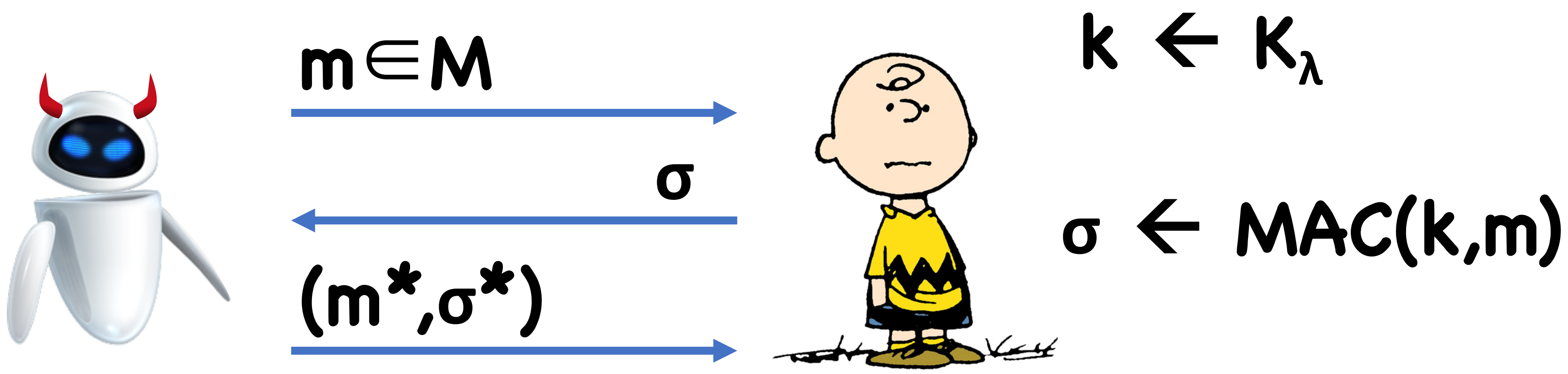
- $\forall \mathbf{m,k,\ Ver(k,m,\ MAC(k,m)\ ) = 1}$

Limitations of CPA security



Goal: If Eve changed **m**, Bob should reject

1-time Security For MACs



$k \leftarrow K_\lambda$

$m \in M$

σ

$\sigma \leftarrow MAC(k,m)$

(m^*,σ^*)

Output 1 iff:

- $m^* \neq m$
- $Ver(k,m^*,\sigma^*) = 1$

1CMA-Adv(, λ) = Pr[outputs 1]

Definition: (MAC,Ver) is 1-time secure under a chosen message attack **(1CMA-secure)** if, for all PPT , there exists a negligible ε such that

$$\textbf{1CMA-Adv}(\ ,\ \lambda) \leq \varepsilon(\lambda)$$

A Simple 1-time MAC

Suppose $\mathbf{H_\lambda}$ is a family of pairwise independent functions from $\mathbf{M}$ to $\mathbf{T_\lambda}$

For any $\mathbf{m_0 \neq m_1 \in M,\ \sigma_0, \sigma_1 \in T_\lambda}$

$$\Pr_{h \leftarrow H_\lambda}[\ h(m_0)=\sigma_0 \wedge h(m_1)=\sigma_1] = 1/|T_\lambda|^2$$

$\mathbf{K = H_\lambda}$

$\mathbf{MAC(h, m) = h(m)}$

$\mathbf{Ver(h,m,\sigma) = (h(m) == \sigma)}$

Theorem: (MAC,Ver) is 1-time secure, provided $\mathbf{T_\lambda}$ is large enough. In particular, for any (not necessarily PPT) ,

1CMA-Adv(, λ) = $1/|T_\lambda|$

So to have security, just need $\mathbf{|T_\lambda|}$ to be superpolynomial

- Ex: $\mathbf{T_\lambda = \{0,1\}^\lambda}$

Next Time

Many-time MACs

Constructing MACs