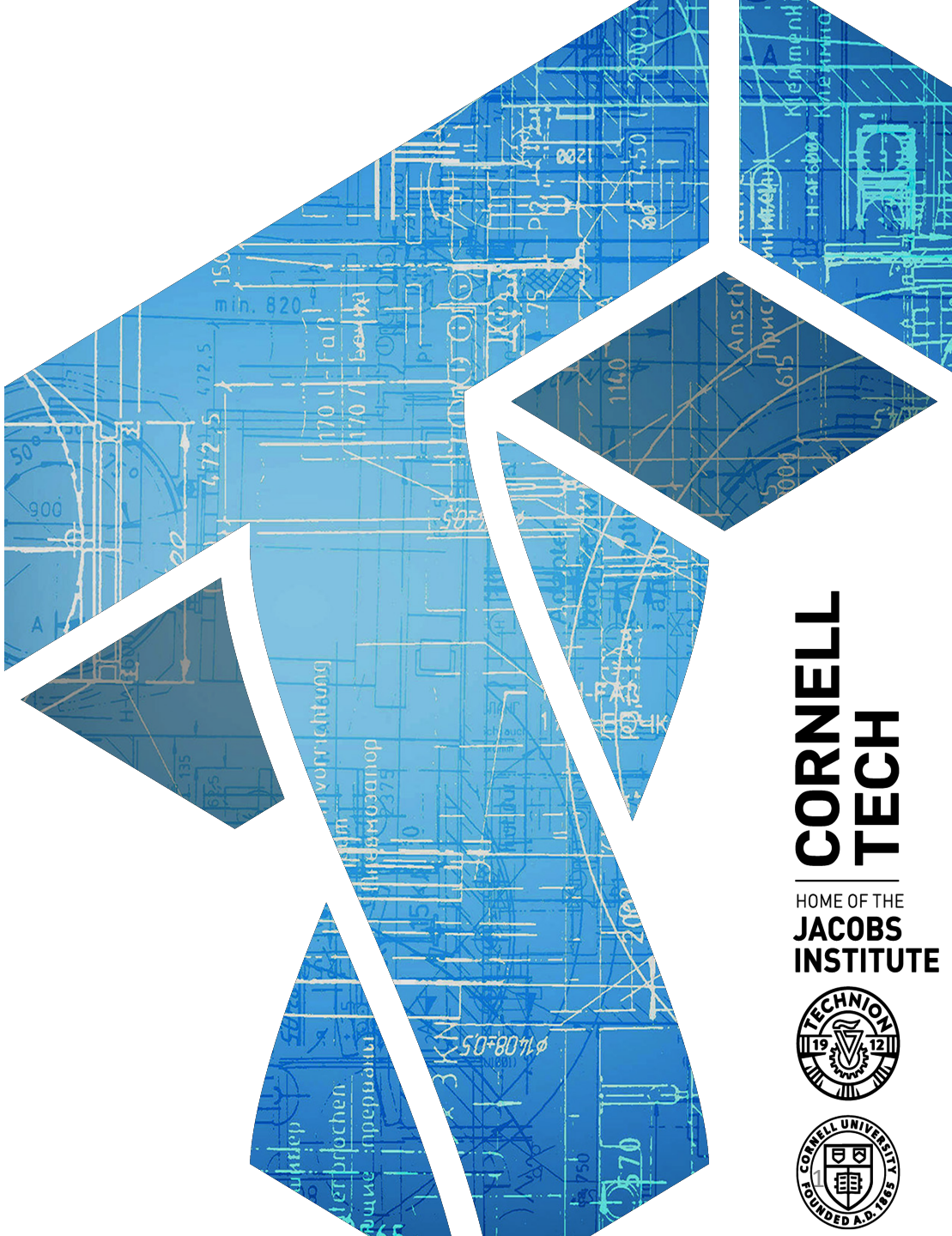
CS 5830
Cryptography

Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Administrivia

- Extra credit projects (up to 10% extra credit)
  - Some students expressed interest in doing semester project for extra credit
  - Potential ideas:
    - Extend studio project with some crypto in a meaningful way
    - Pick recent academic research paper(s) reimplement results and perform performance experiments;
    - Extend recent research results in some way (show new attack, explore new use cases, etc.);
    - Audit or extend some open source crypto project
    - …
  - Send project proposal to me by end of February
  - Deliverable would be presentation to class on what you did last week of semester (amount of time TBD)
- Homework 1 questions?

Recap and where we're at

- Stream ciphers as computationally secure one-time pads
  - RC4 example of custom-built stream cipher
    - More modern examples: ChaCha, Xsalsa, Trivium, etc.
- Can build stream ciphers from blockciphers
  - CTR mode encryption
- Today: more on blockciphers

Block ciphers

Family of permutations, one permutation for each key

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

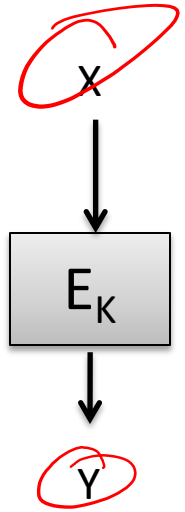
Use notation $E(K,X) = E_K(X) = Y$

Define inverse $D(K,Y) = D_K(Y) = X$

$$\text{s.t. } D(K,E(K,X)) = X$$

E,D must be efficiently computable

Pick K uniformly at random from $\{0,1\}^k$



K_1

X	Y
00	10
01	11
10	01
11	00

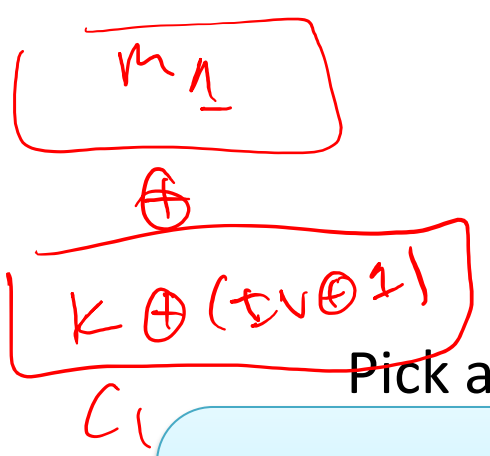
K_2

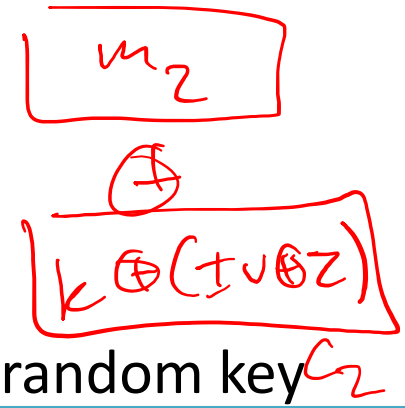
X	Y
00	11
01	10
10	00
11	01

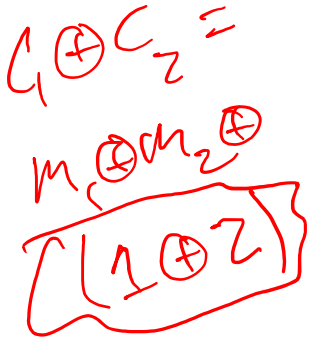
K_3

X	Y
00	11
01	10
10	00
11	01

CTR-mode SE scheme







Kg():
K <-$ {0,1}^{k}

Enc(K,M):
L <- |M| ; m <- ceil(L/n)
IV <-$ {0,1}$^{n/2}$
P <- E_K(IV || 1) || ⋯ || trunc(E_K(IV || m))
Return (IV, P ⊕ M)

Dec(K,(IV,C)):
L <- |C| ; m <- ceil(L/n)
P <- E_K(IV || 1) || ⋯ || trunc(E_K(IV || m))
Return (IV, P ⊕ C)

Pick a random key

Should be able to call Enc with same K for many messages. What could go wrong?

What security properties do we need from the block cipher?

Assume ciphertext can be parsed into IV and remaining ciphertext bits

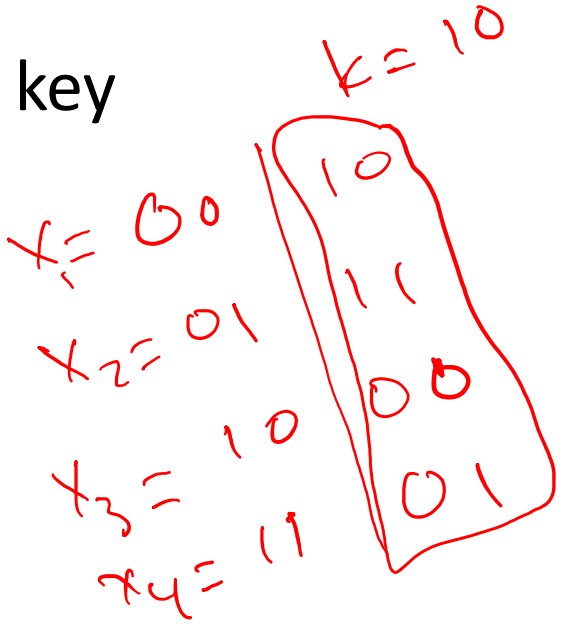
One-time pad as a blockcipher

Family of permutations, one permutation for each key

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

Let $E_K(X) = X \oplus K$

Then $D_K(Y) = Y \oplus K$



This defines a family of permutations, one for each key.

Efficient to compute

So this can be considered a blockcipher, functionally speaking

But is this secure?

Blockcipher security: key recovery

We would want a blockcipher to *at least* hide the key

Key recovery under chosen-message attack (KR-CMA):

- Adversary can see plaintext/ciphertext pair
- Goal is to recover secret key K

KR-CMA(SE, $\mathcal{A}$):
X <-$ $\mathcal{A}$
K <-$ $\{0,1\}^k$
K' <-$ $\mathcal{A}(E_K(X))$
Return (K = K')

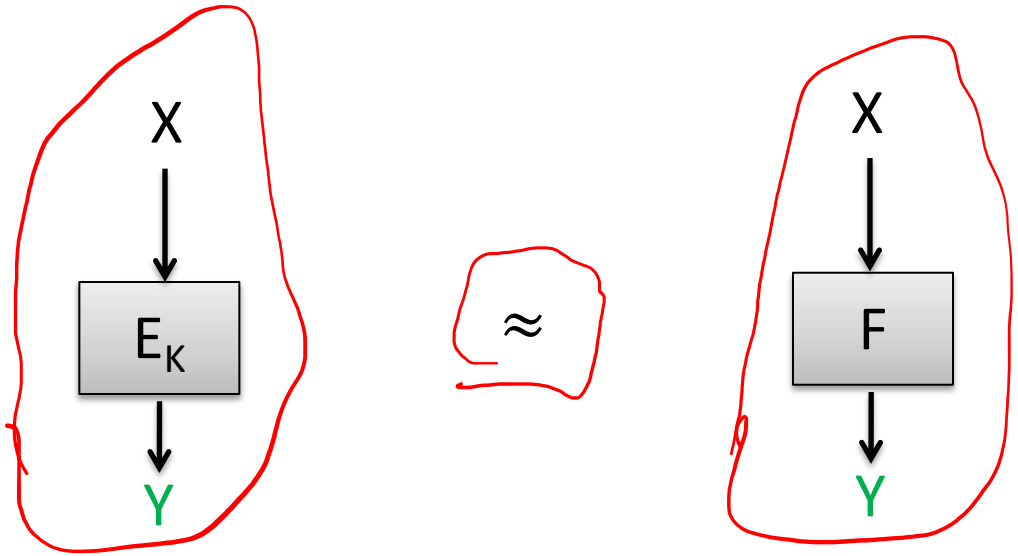
Can you give adversary that breaks OTP as a blockcipher?

$E_K(X) = X$

Can you give blockcipher for which key can't be recovered?
(Hint: just focus on correctness) idea: ignore the key!

KR security is necessary, but is it sufficient?

Pseudorandom function (PRF) security



F is a random function:

X	Y
00	10
01	11
10	10
11	00

Choose each Y value at random, with replacement

No efficient adversary can distinguish between E_K and random function

- Even given chosen-messages attack: can query X of choosing and get Y, many times

Pseudorandom function (PRF) security

Func(n) is set of all functions
$\{0,1\}^n \rightarrow \{0,1\}^n$

```
PRF(E, C):
K <-$ {0,1}^k
F <-$ Func(n)
b <-$ {0,1}
b' <-$ C^O()
Return (b = b')

O(X):
If b = 1 then
   Return E_K(X)
Return F(X)
```

(t,q,ϵ)-pseudorandom function:
no attacker $\mathcal{C}$ limited to time t and q queries to O can distinguish between E_K and random function with advantage greater than ϵ

O is called an ***oracle***.
A subroutine that adversary can make calls to

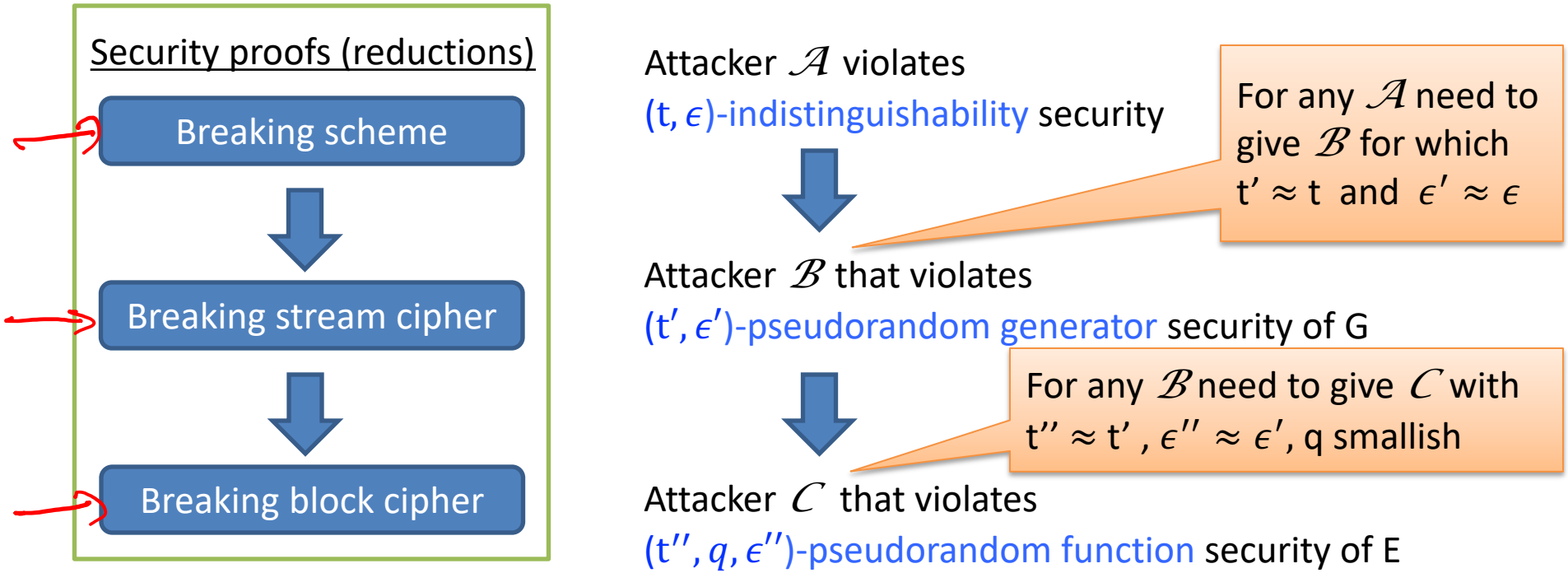
Measuring advantage of adversary:

$$\Pr[\text{PRF}(G,L,\mathcal{C}) = 1] \leq 1/2 + \epsilon$$

Give an adversary $\mathcal{C}$ that achieves high advantage against E = OTP

Reduction-based security analysis

Goal: show that if blockcipher is secure, then CTR encryption is secure

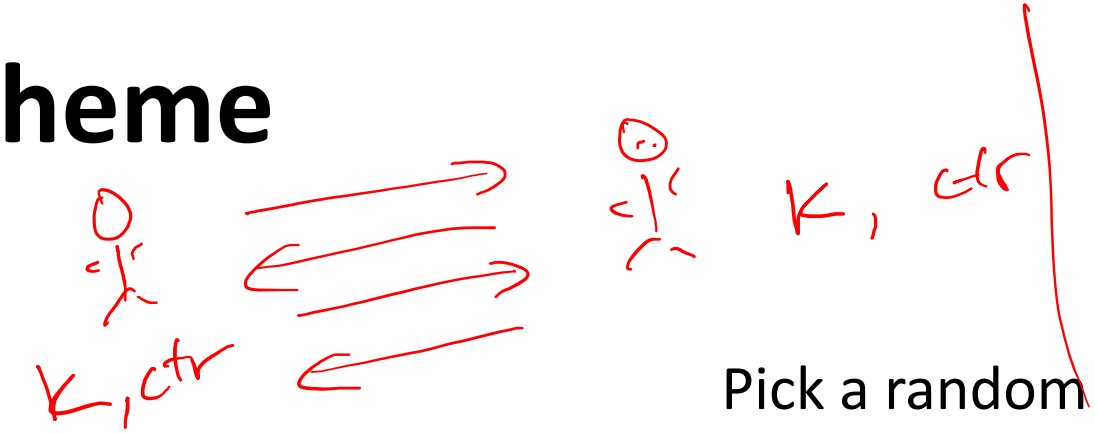


Reduces security analysis task to analyzing block cipher

Confidence in block cipher security gives confidence in scheme's security

This is a bit simplistic: in fact want multi-message security

CTR-mode SE scheme



Kg():
K <-$ $\{0,1\}^k$

Pick a random key

Enc(K,M):
L <- |M| ; m <- ceil(L/n)
IV <-$ $\{0,1\}^n$
P <- $E_K(IV \oplus 1) \| \cdots \| \text{trunc}(E_K(IV \oplus m))$
Return (IV, P $\oplus$ M)

Should be able to call Enc with same K for many messages. What could go wrong?

Dec(K,(IV,C)):
L <- |C| ; m <- ceil(L/n)
P <- $E_K(IV \oplus 1) \| \cdots \| \text{trunc}(E_K(IV \oplus m))$
Return (IV, P $\oplus$ C)

Assume ciphertext can be parsed into IV and remaining ciphertext bits

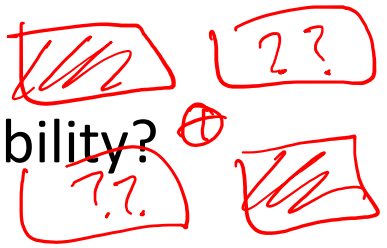
Many-message attack against CTR

Adversary obtains q CTR encryptions (IV_1,C_1) , (IV_2,C_2) , … , (IV_q,C_q) under same key K
Look for $i \neq j$ such that $IV_i = IV_j$
Output $C_i \oplus C_j = M_i \oplus M_j$

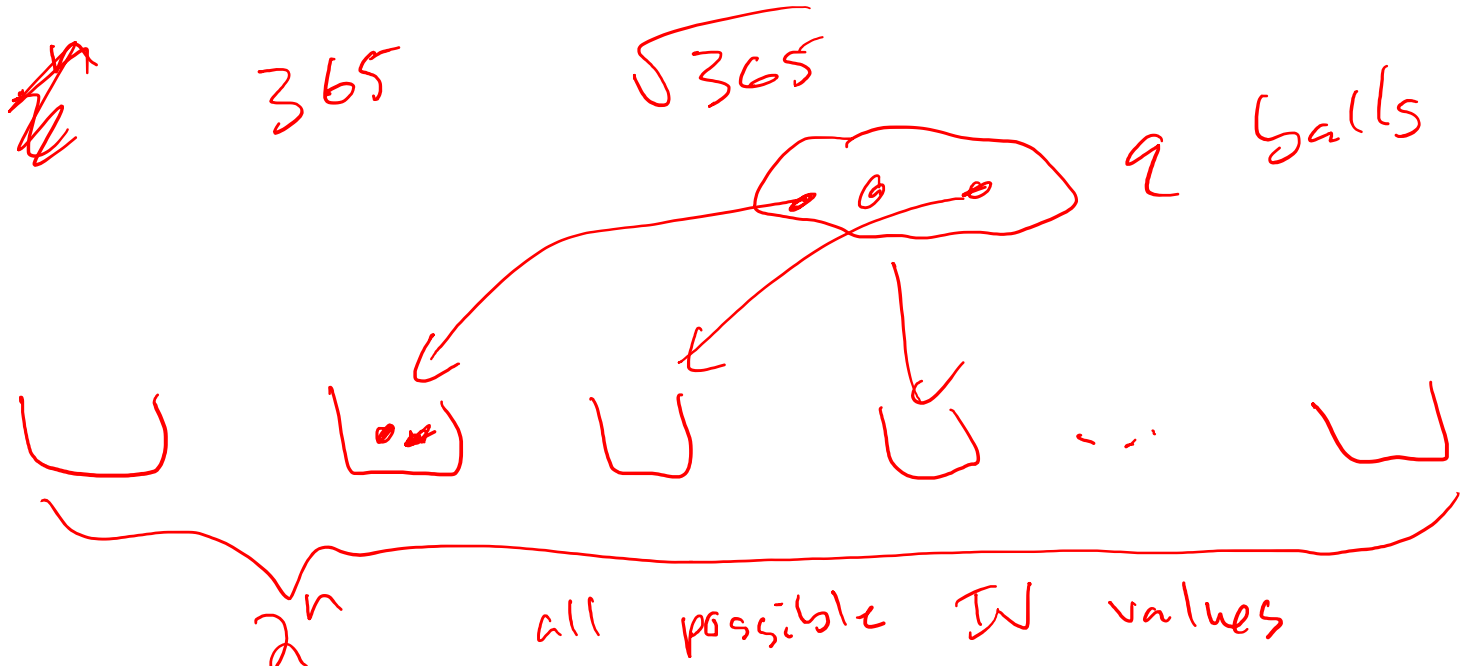
$IV_i \xleftarrow{\$} \{0,1\}^n$

First half of M_i | Second half of M_j

This reveals a lot of partial information about plaintext



How big does q need to get before attack succeeds with good probability?



The Birthday Bound

Throw q balls randomly into 2^n bins. Let Coll(2^n,q) be event that some bin has (at least) two balls. Then:

$$0.3q(q-1) / 2^n \leq \Pr[\text{Coll}(2^n,q)] \leq q(q-1) / 2^{n+1}$$

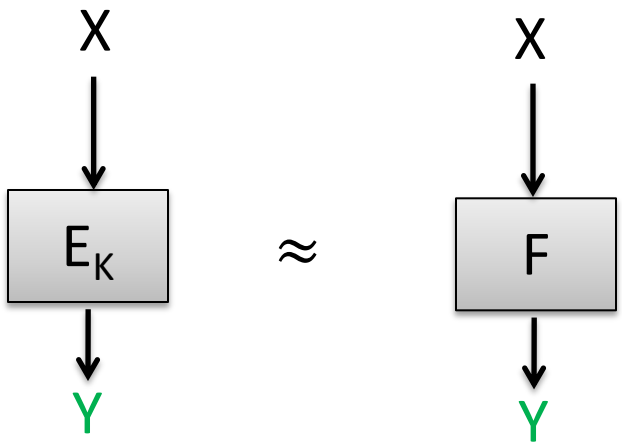
$q^2 / 2^n \to 1$ when $q = 2^{n/2}$

See Appendix A of: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

Birthday PRF attack against *any* block cipher

E_K is a ***permutation***

X	Y
00	10
01	11
10	01
11	00



F is a random ***function***:

X	Y
00	10
01	11
10	10
11	00

Choose each Y value at random, with replacement

For any sequence of queries by adversary what is guaranteed to ***never*** happen when interacting with E_K that could happen when interacting with F?

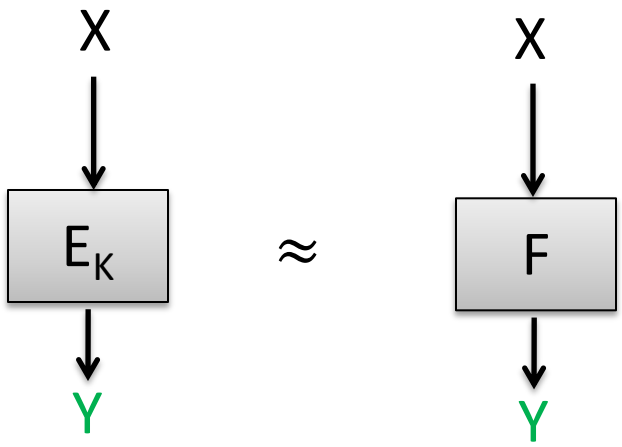
If n is bit length of Y values, how many outputs of F do we have to see before we are likely to see repeat?

$2^{n/2}$

Birthday PRF attack against *any* block cipher

E_K is a ***permutation***

X	Y
00	10
01	11
10	01
11	00



F is a random ***function***:

X	Y
00	10
01	11
10	10
11	00

Choose each Y value at random, with replacement

Generic PRF adversary: query oracle O on q distinct inputs $X_1, \ldots, X_q$ to get outputs $Y_1, \ldots, Y_q$. If exists $Y_i = Y_j$ guess that you are interacting with F.

Succeeds if $q \approx 2^{n/2}$

DES $n = 64$

AES $n = 128$

This means we need n to be large enough to make $2^{n/2}$ intractably large

More on block cipher security

- Need key length k to make 2^k computationally intractable
- Need block length n to make $2^{n/2}$ computationally intractable

- Pseudorandom permutation (PRP) security similar to PRF, but indistinguishability from random permutation
  - We want PRF security in CTR application, so focused on that
  - Birthday bound proof equivalence of PRP/PRF notions for $q \ll 2^{n/2}$
- Chosen-ciphertext attack variants
  - Strong PRF and PRP security gives access to inverse oracle to which Y values of adversary's choosing can be queried

Block cipher design

- A big topic with ~50 year history
- Data Encryption Standard (DES) designed in 1970s
  - Uses Feistel network structure
- Advanced Encryption Standard (AES) designed in 1990s
  - Uses substitution-permutation network structure

Data encryption standard (DES)

Originally called Lucifer
- team at IBM
- input from NSA
- standardized by NIST in 1976

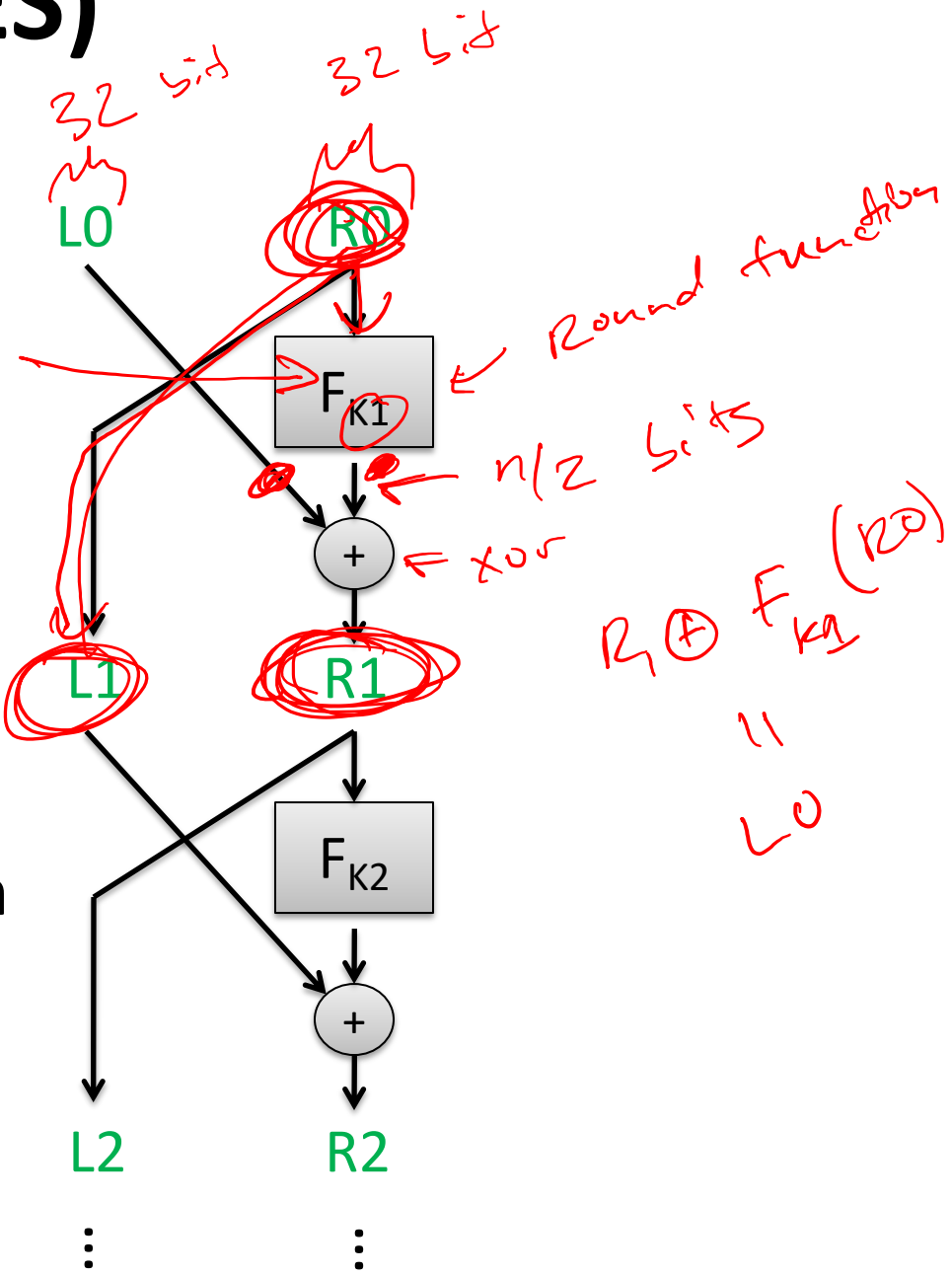
n = 64
k = 56

Number of keys:
72,057,594,037,927,936

Split 64-bit input into L0,R0 of 32 bits each

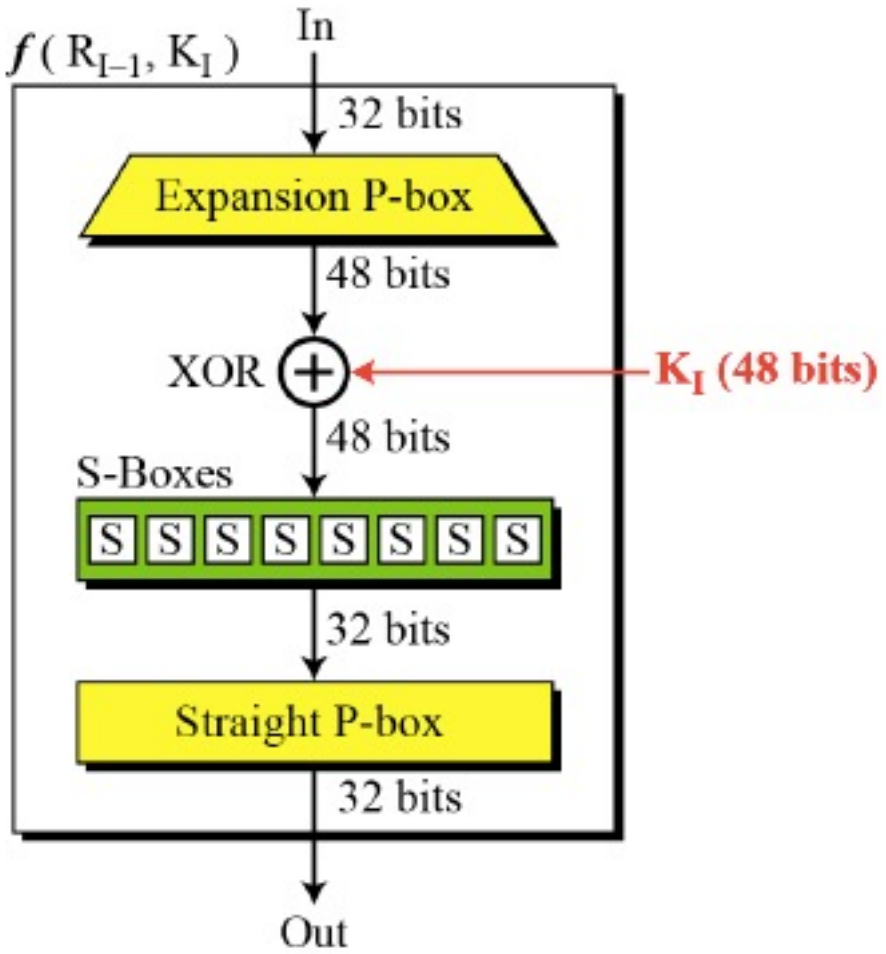
Repeat Feistel round 16 times

Each round applies function F using separate round key



DES round functions

- P-box expands 32 bits to 48 bits and permutes
- S-boxes: 6-bit to 4-bit lookup tables
- XOR in round key
  - 16 48-bit round keys derived via key schedule from 56 bit key deterministically
- How S-boxes chosen? Why particular permutations?

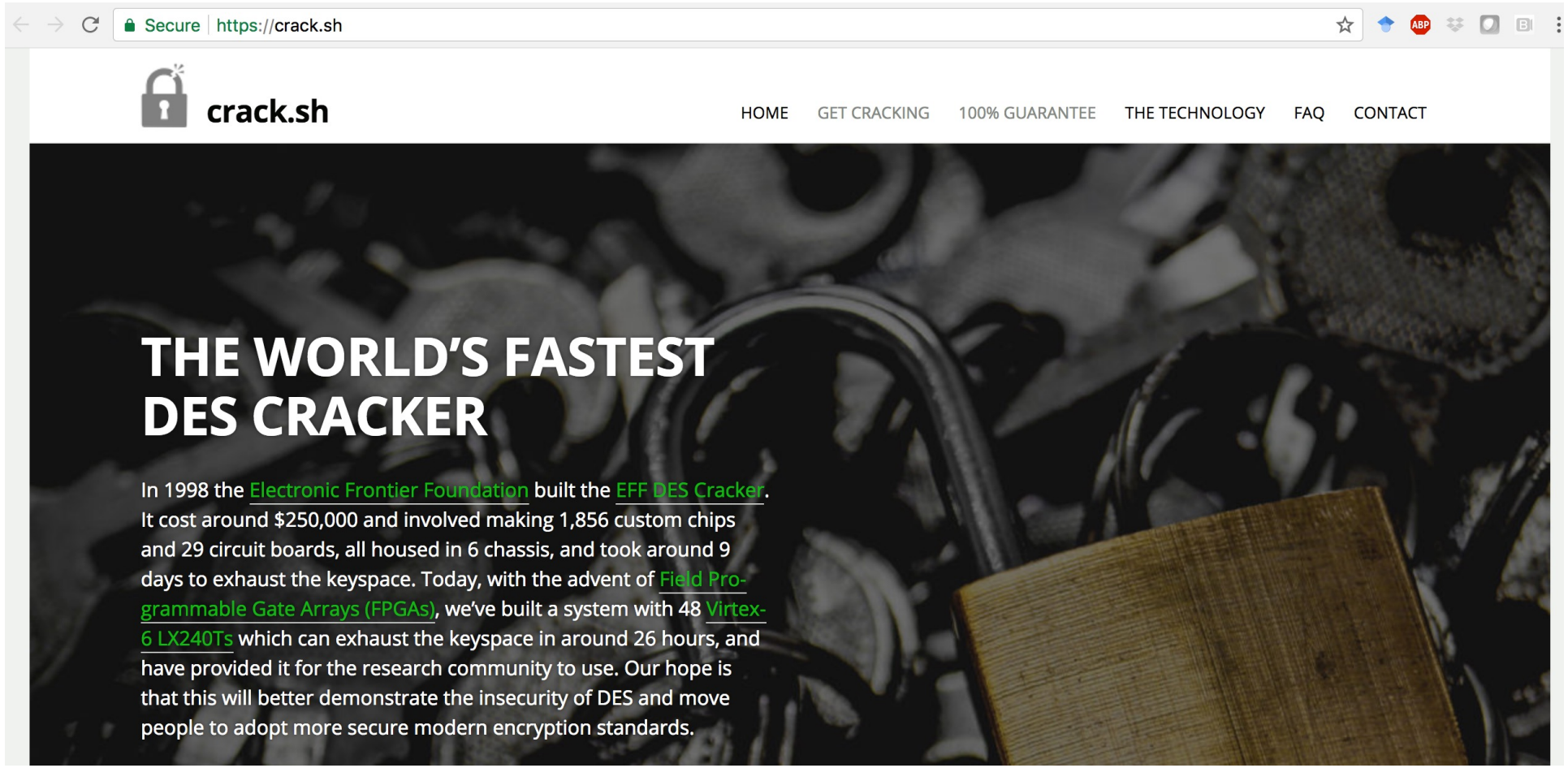


Best attacks against DES

Attack	Attack type	Complexity	Year
Biham, Shamir	Chosen plaintexts, recovers key	2^{47} plaintext, ciphertext pairs	1992
Matsui	Known plaintexts, recovers key	2^{47} plaintext, ciphertext pairs	1993
DESCHALL	Brute-force attack	$2^{56/4}$ DES computations 41 days	1997
EFF Deepcrack	Brute-force attack	~4.5 days	1998
Deepcrack + DESCHALL	Brute-force attack	22 hours	1999

DES is still used in some places

3DES (use DES 3 times in a row with more keys) expands keyspace and still used widely in practice

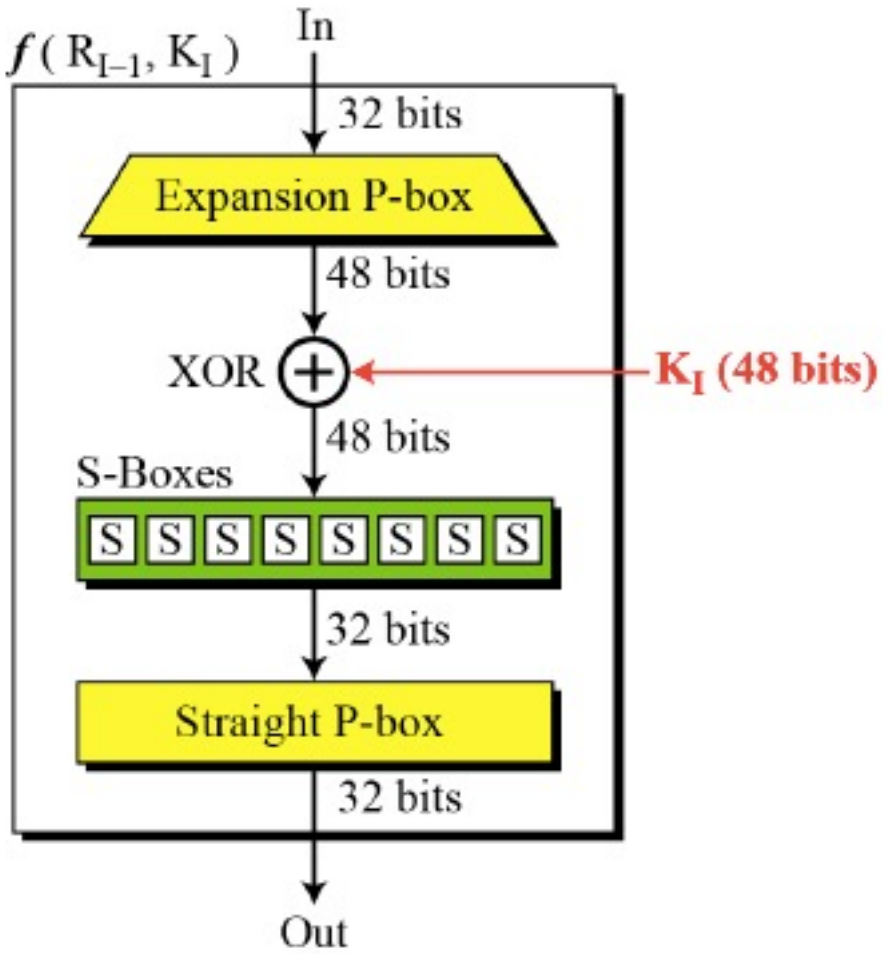

Secure https://crack.sh
crack.sh
HOME
GET CRACKING
100% GUARANTEE
THE TECHNOLOGY
FAQ
CONTACT
THE WORLD'S FASTEST DES CRACKER
In 1998 the Electronic Frontier Foundation built the EFF DES Cracker. It cost around $250,000 and involved making 1,856 custom chips and 29 circuit boards, all housed in 6 chassis, and took around 9 days to exhaust the keyspace. Today, with the advent of Field Programmable Gate Arrays (FPGAs), we've built a system with 48 Virtex-6 LX240Ts which can exhaust the keyspace in around 26 hours, and have provided it for the research community to use. Our hope is that this will better demonstrate the insecurity of DES and move people to adopt more secure modern encryption standards.

The History

- DES (under name Lucifer) designed by IBM in 1970s
- NIST standardized it
  - NSA evaluated it and made suggested changes to shorten key length to 56 bits and changes to S-boxes
  - Many public criticisms of these changes, though S-boxes change actually strengthened DES
- AES competition run by NIST (1997-2000)
  - Many good submissions (15 total submissions)
  - AES chosen as winner

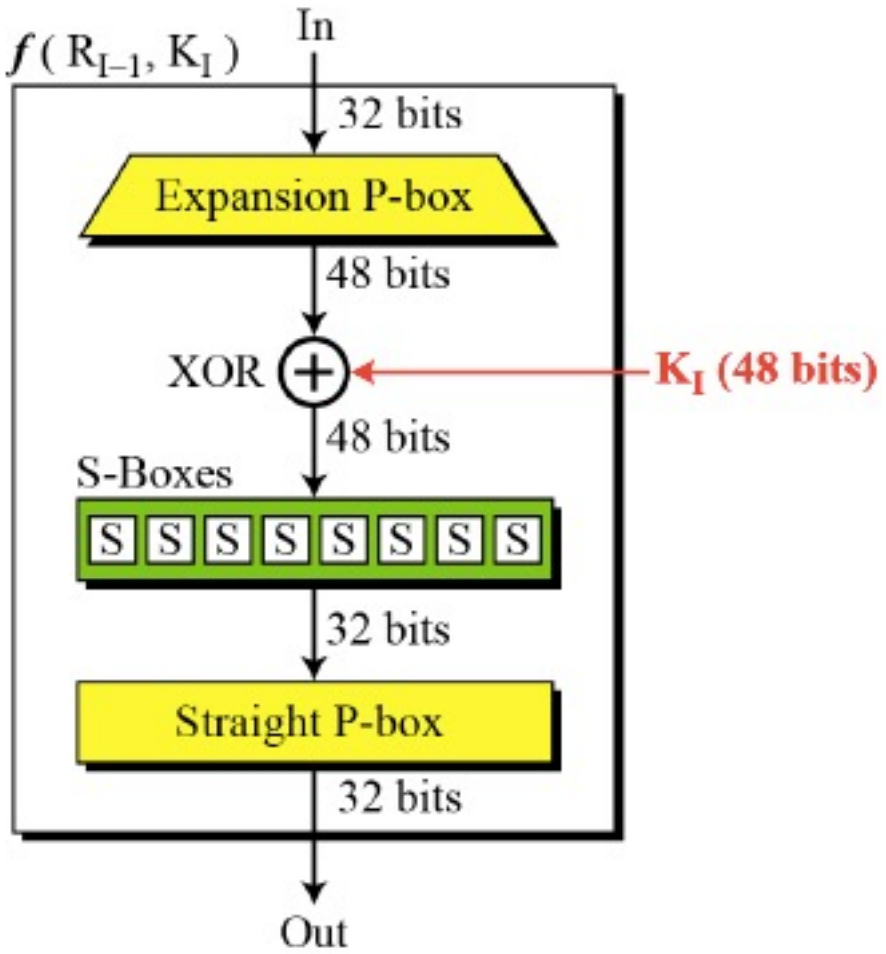
Differential cryptanalysis

- Differential cryptanalysis: [Biham, Shamir 1992]
  - Analyze S-boxes to find differentials Δ_x, Δ_y such that
    - $\Delta_y = Sbox(X + \Delta_x) + Sbox(X)$
  - holds with high probability over random choice of X
  - Piece together such differentials across rounds, providing a differential trail
  - Use to reduce key search space significantly, given lots of encryptions of chosen messages



Linear cryptanalysis

- Linear cryptanalysis: [Matsui 1993]
  - Approximate S-box behavior by linear functions, e.g.,:
    - X1 + X2 + X6 = Y1 + Y2 + Y4
  - S-boxes exhibit some biases, meaning some linear functions are satisfied with higher probability (over uniform inputs)
  - Can carefully "pile up" linear approximations across multiple DES rounds
  - Use linear approximations to efficiently narrow down search for key, given lots of encryptions of uniform messages



http://www.cs.bc.edu/~straubin/crypto2017/heys.pdf

Next up

- Building & analyzing block ciphers
  - Feistel networks and DES
  - AES cipher
  - Cryptanalysis of block ciphers