Introduction to Cryptography

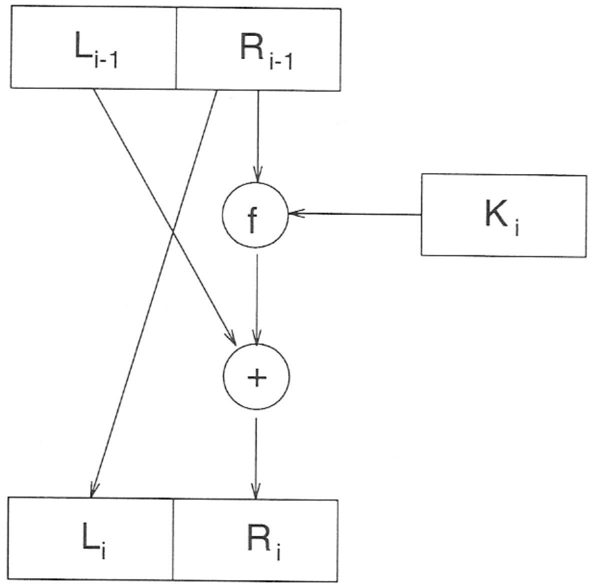
Lecture 4
Differential cryptanalysis of DES, message authentication

Benny Pinkas

Feistel Networks

- Encryption:
- *Input:* $P = L_{i-1} \mid R_{i-1}$. $|L_{i-1}|=|R_{i-1}|$
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(K_i, R_{i-1})$
- Decryption?

- No matter which function is used as F, we obtain a permutation (i.e., F is reversible even if *f* is not).
- The same code/circuit, with keys in reverse order, can be used for decryption.
- Theoretical result [LubRac]: If *f* is a pseudo-random *function* then a 4 rounds Feistel network gives a pseudo-random *permutation*

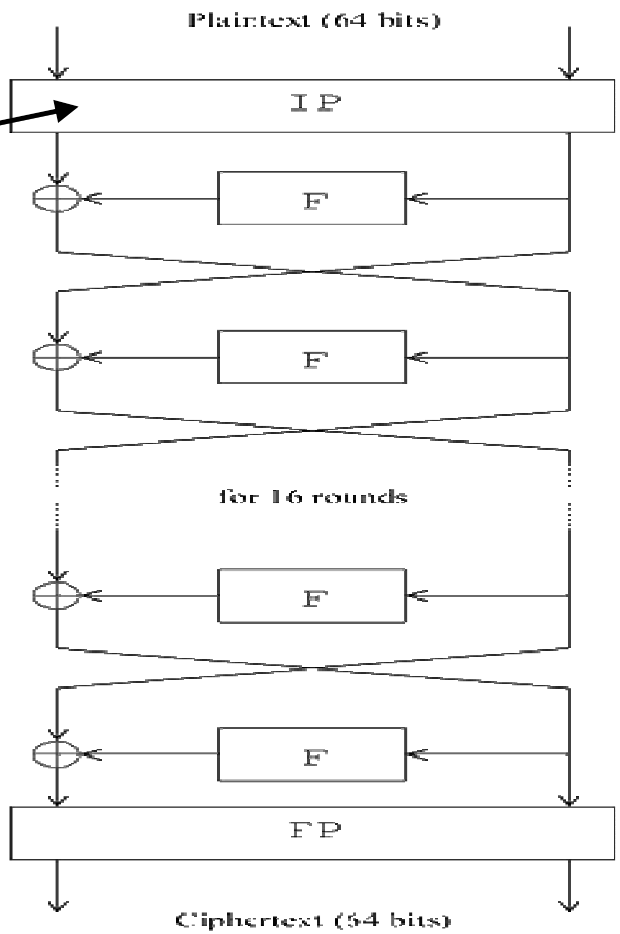


DES (Data Encryption Standard)

- A Feistel network encryption algorithm:
  - How many rounds?
  - How are the round keys generated?
  - What is F?

- DES (Data Encryption Standard)
  - Designed by IBM and the NSA, 1977.
  - 64 bit input and output
  - 56 bit key
  - 16 round Feistel network
  - Each round key is a 48 bit subset of the key
- Throughput ≈ software: 10Mb/sec, hardware: 1Gb/sec (in 1991!).

DES

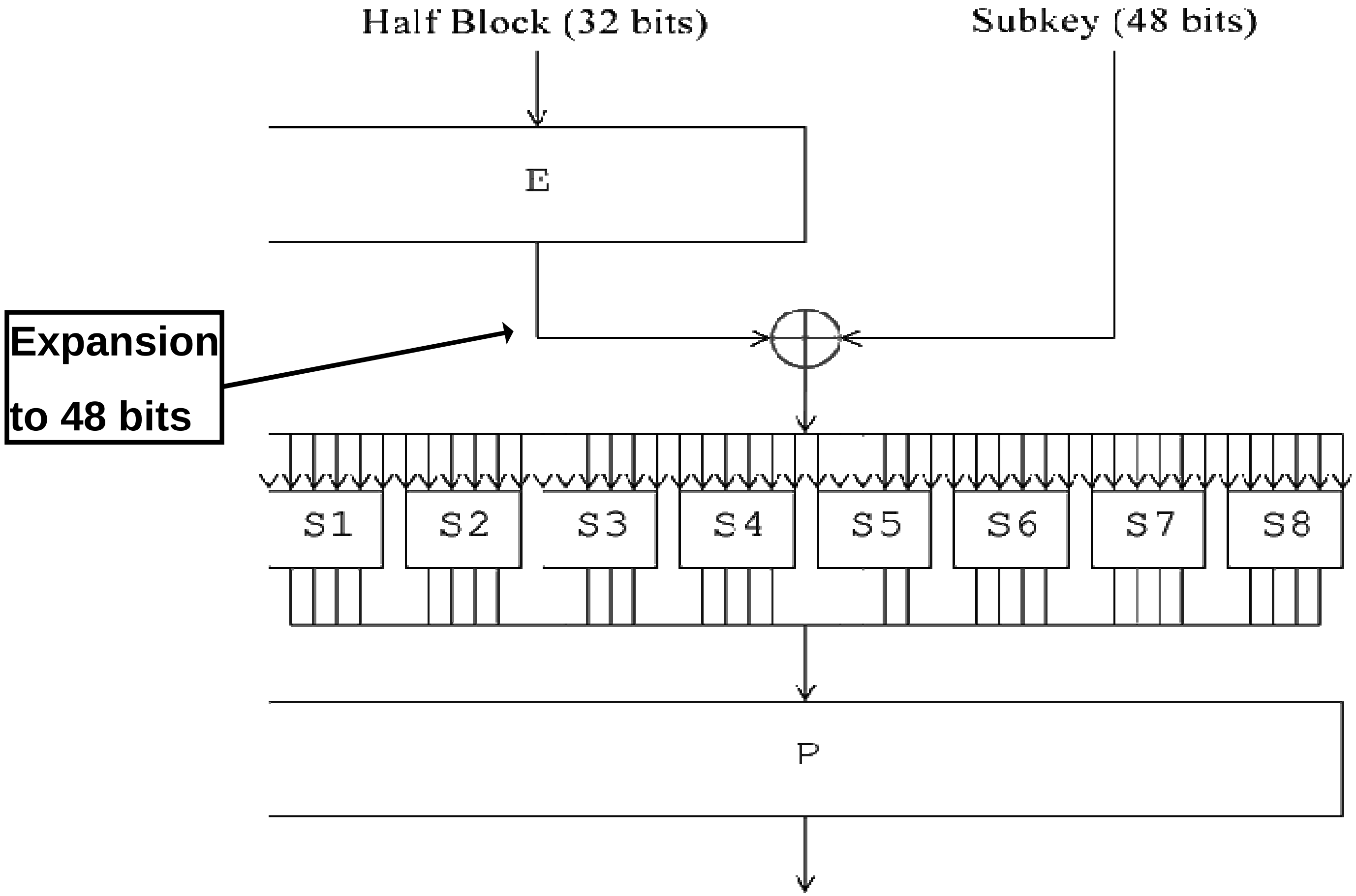


Initial permutation of bit locations:

- not secret

- makes implementations in software less efficient

DES F functions



The S-boxes

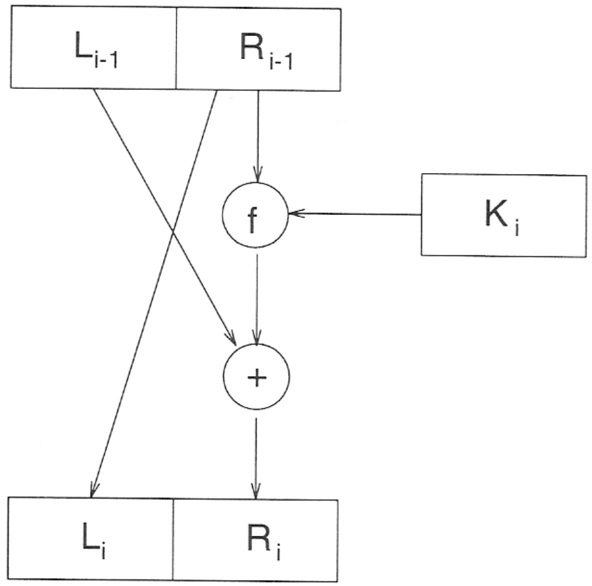
- Very careful design (it is now known that random choices for the S-boxes result in weak encryption).

- Each s-box maps 6 bits to 4 bits:
  - A 4×16 table of 4-bit entries.
  - Bits 1 and 6 choose the row, and bits 2-5 choose column.
  - Each row is a *permutation* of the values 0,1,…,15.
    - Therefore, given an output there are exactly 4 options for the input
  - Changing one input bit changes at least two output bits $\Rightarrow$ avalanche effect.

A Linear F in a Feistel Network?

- Suppose $F(R_{i-1},K_i) = R_{i-1} \oplus K_i$
  - Namely, that F is linear
- Then $R_i = L_{i-1} \oplus R_{i-1} \oplus K_i$

  $L_i = R_{i-1}$
- Write L_{16}, R_{16} as linear functions of L_0, R_0 and K.
  - Given L_0R_0 and $L_{16}R_{16}$ Solve and find K.
- F must therefore be non-linear.
- F is the only source of non-linearity in DES.



Differential Cryptanalysis of DES [Biham-Shamir 1990]

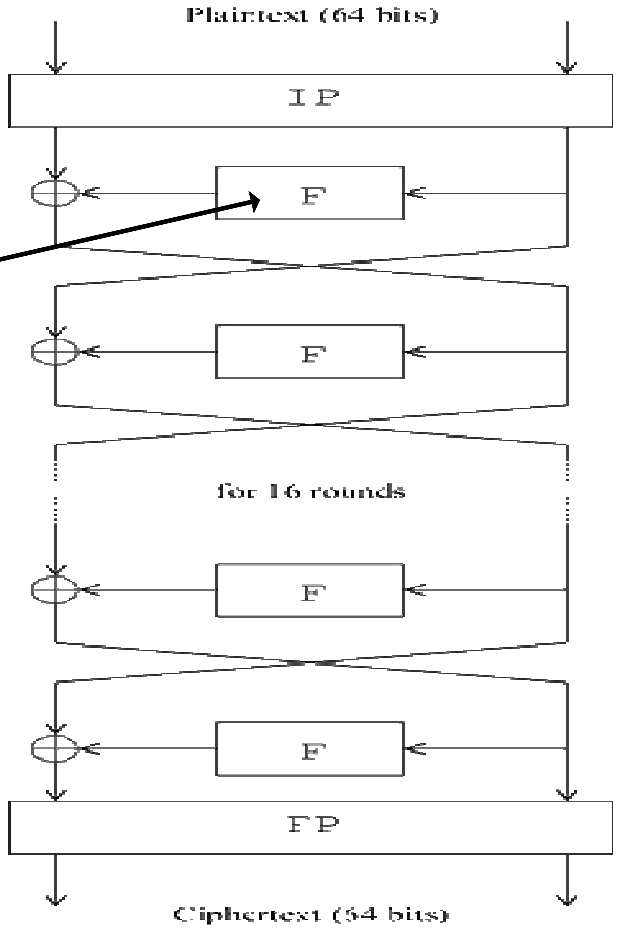
- The first attack to reduce the overhead of breaking DES to below exhaustive search
- Very powerful when applied to other encryption algorithms

- Depends on the structure of the encryption algorithm
- Observation: all operations except for the s-boxes are linear
- Linear operations:
  - $a = b \oplus c$
  - a = the bits of b in (known) permuted order
- Linear relations can be exposed by solving a system of linear equations

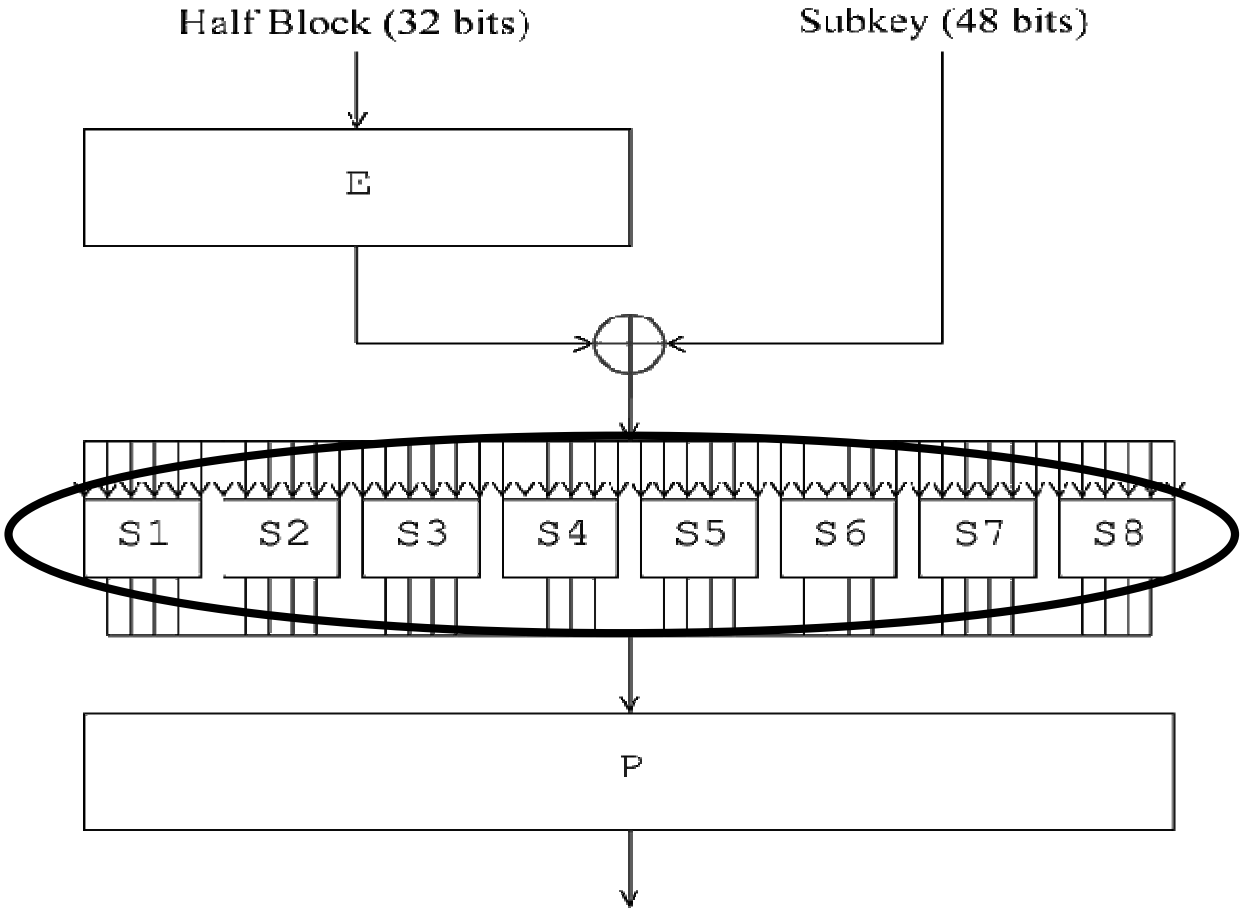
Differential Cryptanalysis of DES

DES diagram:

S-boxes



DES F functions



Source of non-linearity

Differential Cryptanalysis

- The S-boxes are non-linear
- We study the differences (XOR) of two encryptions of two different plaintexts

- Notation:
  - The plaintexts are P and P*
  - Their difference (XOR) is $dP = P \oplus P^*$

  - Let X and X* be two intermediate values, for P and P*, respectively, in the encryption process.
  - Their difference is $dX = X \oplus X^*$
    - Namely, dX is always the result of two inputs

Differences and S-boxes

- S-box: a function (table) from 6 bit inputs to 4 bit output
- X and X* are inputs to the same S-box. We can compute their difference $dX = X \oplus X^*$.
- Y = S(X)
- When dX = X xor X* = 0, then X=X*, and therefore $Y=S(X)=S(X^*)=Y^*$, and dY=0.
- When $dX \neq 0$, $X \neq X^*$ and we don't know dY for sure, but we can investigate its distribution.
- For example,

Distribution of Y' for S1

- dX=110100
- There are 2^6=64 input pairs with this difference, { (000000,110100), (000001,110101),…}
- For each pair we can compute the xor of outputs of S1
- E.g., S1(000000)=1110, S1(110100)=1001. dY=0111.
- Table of frequencies of each dY:

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12
1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

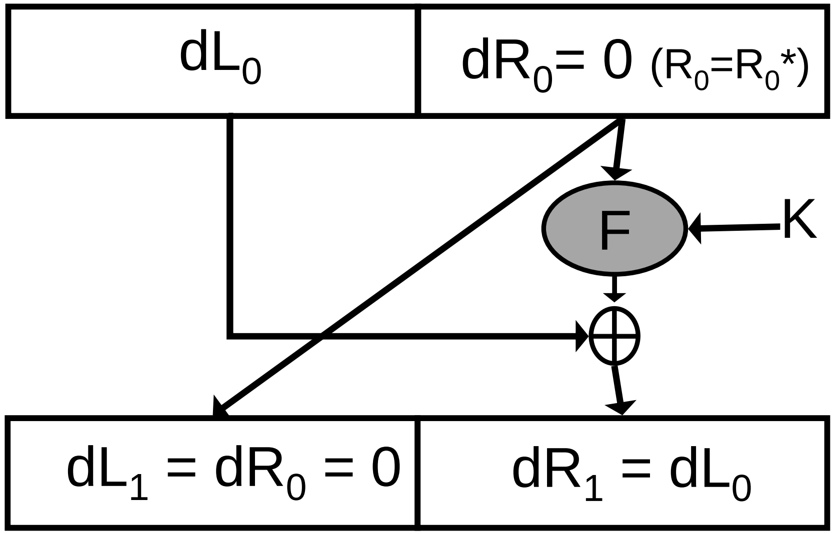
Differential Probabilities

- The probability of dX $\Rightarrow$ dY is the probability that a pair of inputs whose xor is dX, results in a pair of outputs whose xor is dY (for a given S-box).
- Namely, for dX=110100 these are the entries in the table divided by 64.

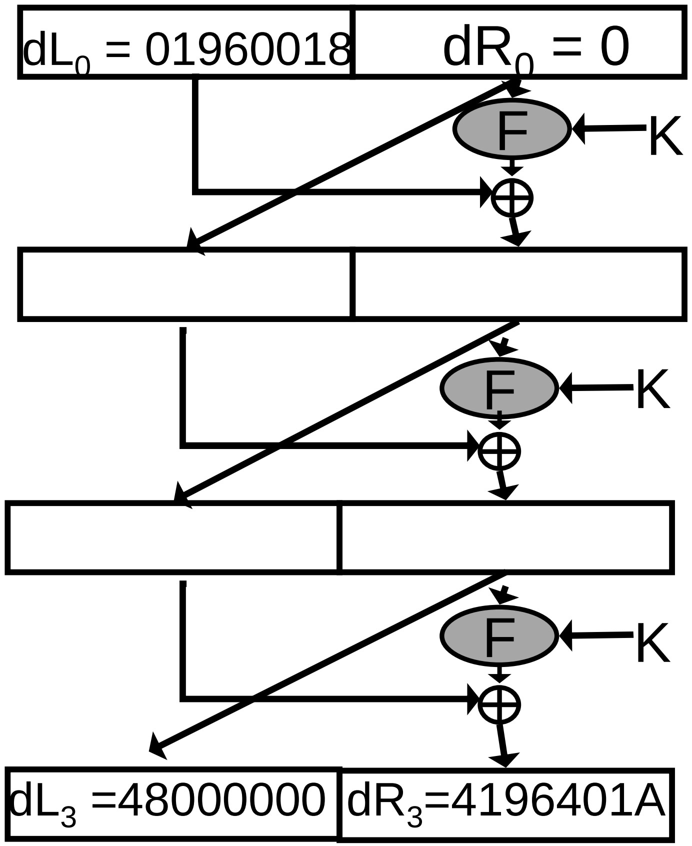
- Differential cryptanalysis uses entries with large values
  - dX=0 $\Rightarrow$ dY=0
  - Entries with value 16/64
  - (Recall that the outputs of the S-box are uniformly distributed, so the attacker gains a lot by looking at differentials rather than the original values.)

Warmup

Inputs: L_0R_0, $L_0^*R_0^*$, s.t. $R_0=R_0^*$.
Namely, inputs whose xor is dL_0 0

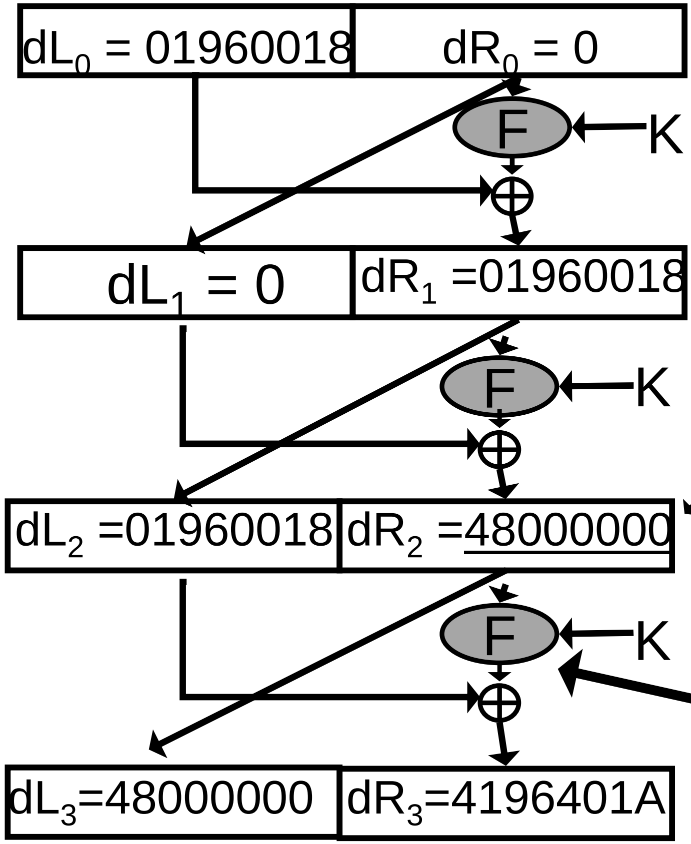


3 Round DES



The attacker knows the two plaintext/ciphertext pairs, and therefore also their differences

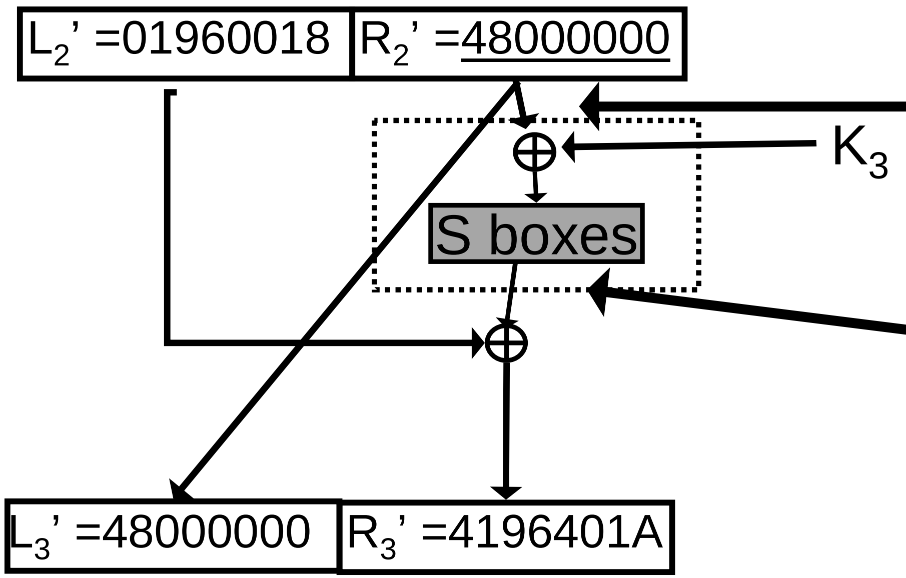
Intermediate differences equal to plaintext/ciphertext differences



Note that here the adversary also knows the actual two values

dF = 4196401A
⊕ 01960018
= <u>40004002</u>

Finding K



The actual two inputs to F are known

Output xor of F (i.e., S boxes) is 40004002

⇒Table enumerates options for the pairs of inputs to S box

Find which K_3 maps the inputs to an s-box input pair that results in the output pair!

DES with more than 3 rounds

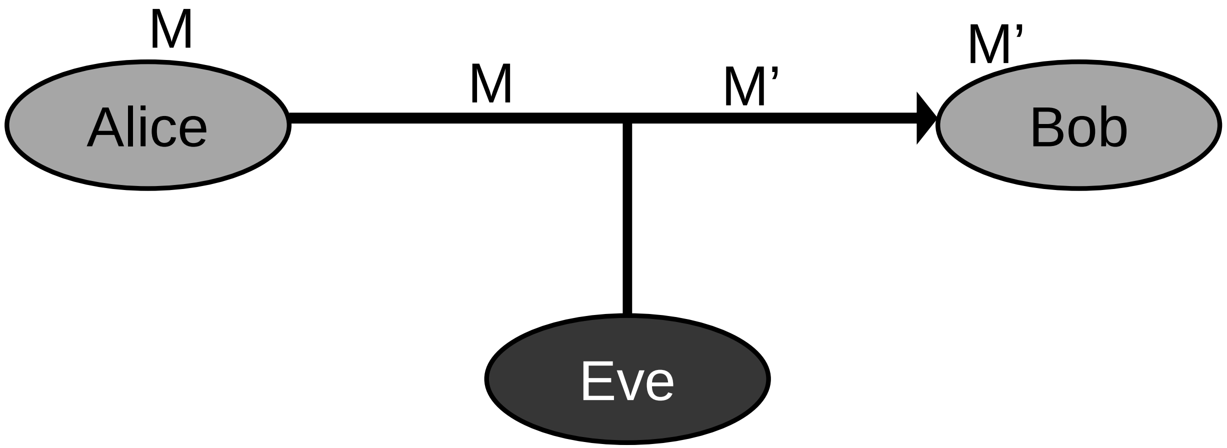
- Carefully choose pairs of plaintexts with specific xor, and determine xor of pairs of intermediate values at various rounds.
- E.g., if $dL_0=40080000_x$, $dR_0=04000000_x$
  Then, with probability ¼, $dL_3=04000000_x$, $dR_3=4008000_x$

- 8 round DES is broken given 2^{14} chosen plaintexts.
- 16 round DES is broken given 2^{47} chosen plaintexts...

Message Authentication

Data Integrity, Message Authentication

- Risk: an *active* adversary might change messages exchanged between Alice and Bob



- Authentication is orthogonal to secrecy. It is a relevant challenge regardless of whether encryption is applied.

One Time Pad

- OTP is a perfect cipher, yet provides no authentication
  - Plaintext $x_1x_2\ldots x_n$
  - Key $k_{1k2}\ldots k_n$
  - Ciphertext $c_1=x_1\oplus k_{1,}$ $c_2=x_2\oplus k_2,\ldots,c_n=x_n\oplus k_n$

- Adversary changes, e.g., c_2 to $1\oplus c_2$
- User decrypts $1\oplus x_2$

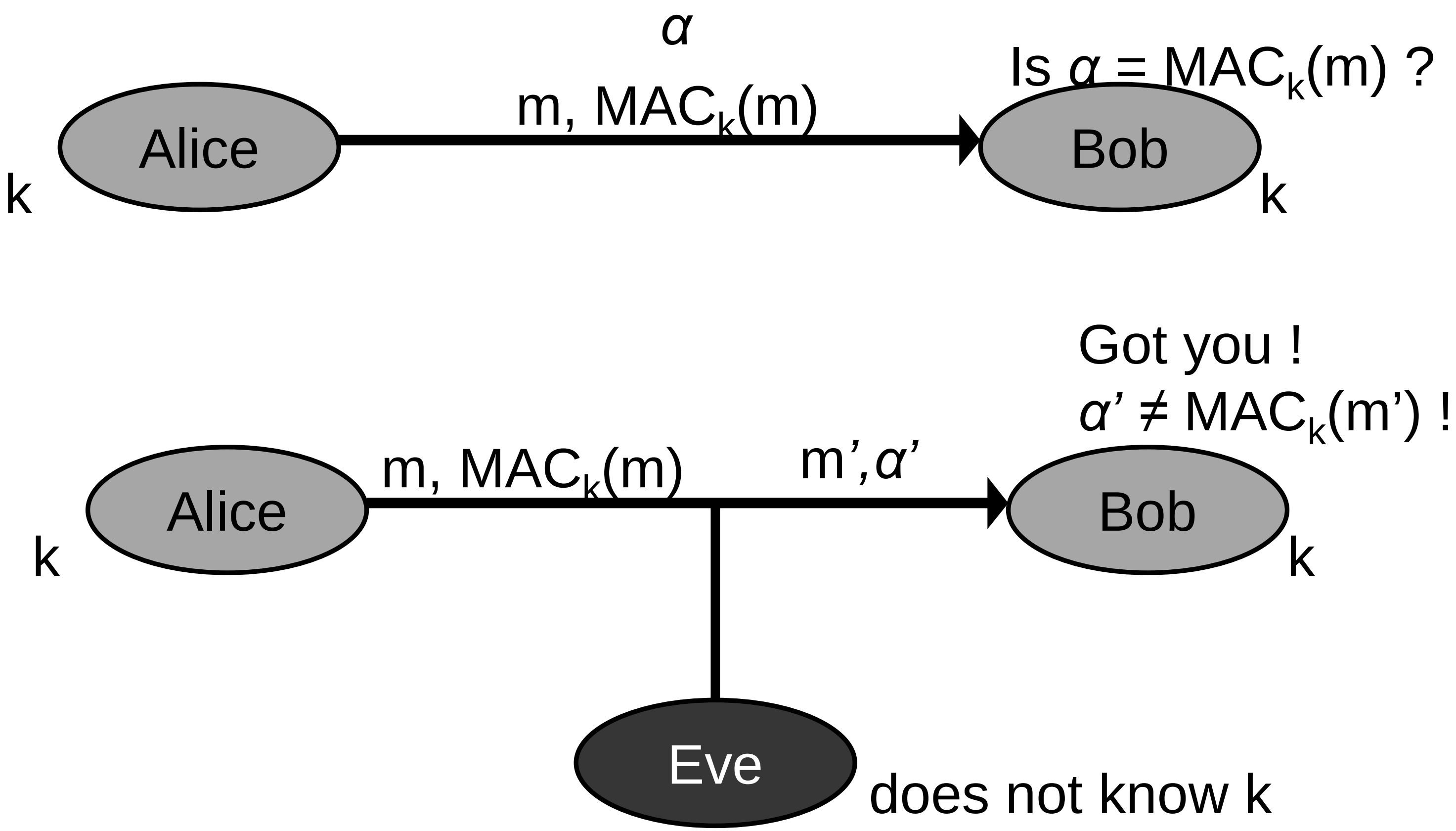
- Error-detection codes are insufficient. (For example, linear codes can be changed by the adversary, even if encrypted.)
  - They were not designed to withstand adversarial behavior.

Definitions

- Scenario: Alice and Bob share a secret key K.
- Authentication algorithm:
  - Compute a Message Authentication Code: $\alpha = MAC_K(m)$.
  - Send m and α
- Verification algorithm: $V_K(m, \alpha)$.
  - $V_K(m, MAC_K(m)) = accept.$
  - For $\alpha \neq MAC_K(m)$, $V_K(m, \alpha) = reject.$

- How does $V_k(m)$ work?
  - Receiver knows k. Receives m and α.
  - Receiver uses k to compute $MAC_K(m)$.
  - $V_K(m, \alpha) = 1$ iff $MAC_K(m) = \alpha$.

Common Usage of MACs for message authentication



Requirements

- Security: The adversary,
  - Knows the MAC algorithm (but not K).
  - Is given many pairs $(m_i, MAC_K(m_i))$, where the m_i values might also be chosen by the adversary (chosen plaintext).
  - Cannot compute $(m, MAC_K(m))$ for any new m $(\forall i\ m \neq m_i)$.
  - The adversary must not be able to compute $MAC_K(m)$ *even* for a message m which is “meaningless” (since we don’t know the context of the attack).

- Efficiency: MAC output must be of fixed length, and as short as possible.
  - $\Rightarrow$ The MAC function is not 1-to-1.
  - $\Rightarrow$ An n bit MAC can be broken with prob. of at least 2^{-n}.

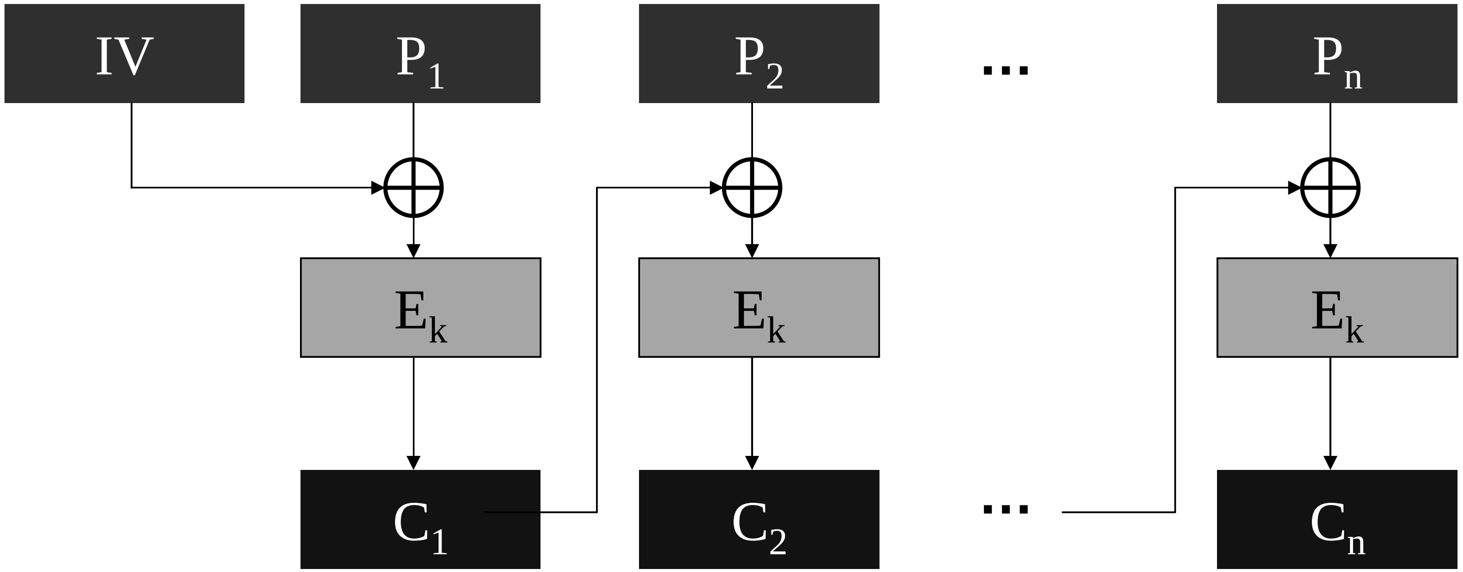
Constructing MACs

- Based on block ciphers (CBC-MAC)

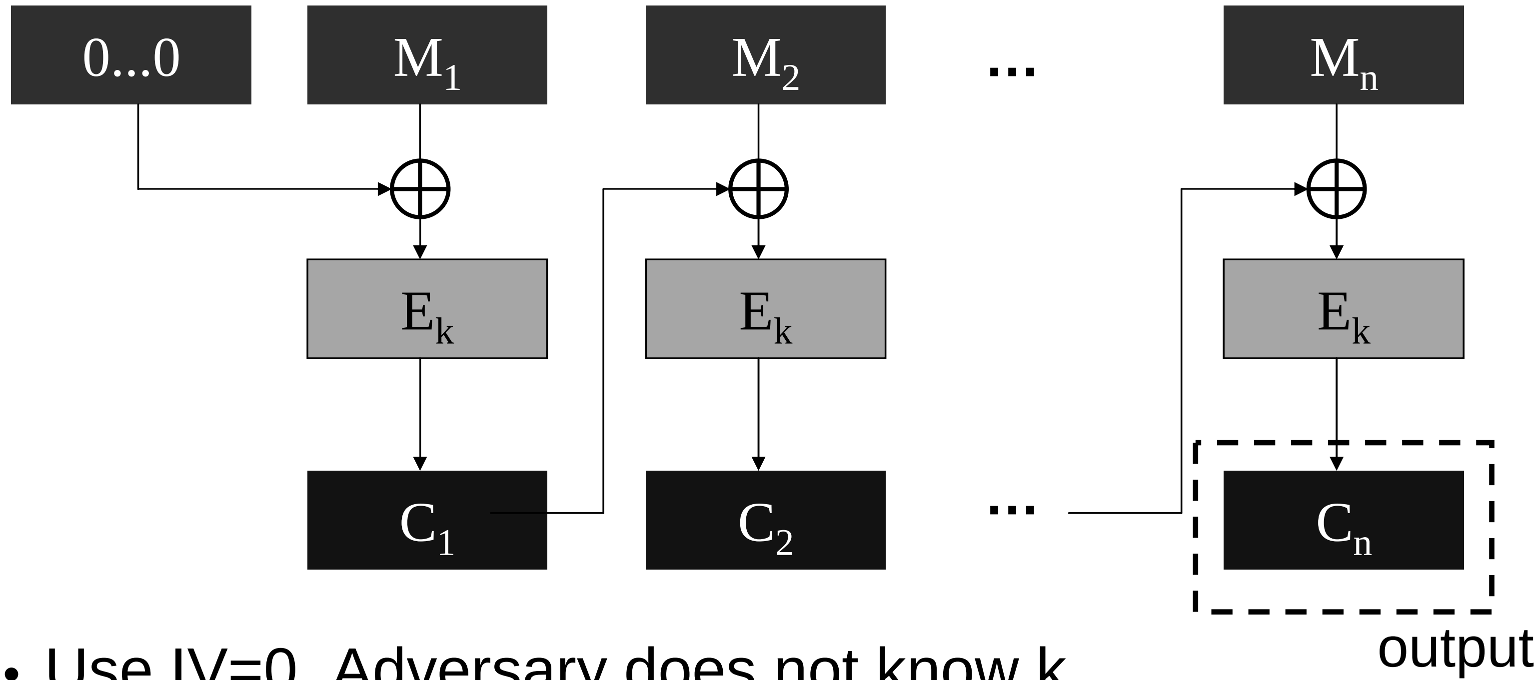
  or,
- Based on hash functions
  - More efficient
  - At the time, encryption technology was controlled (export restricted) and it was preferable to use other means when possible.

CBC

- Reminder: CBC encryption
- Plaintext block is xored with previous ciphertext block



CBC MAC



- Use IV=0. Adversary does not know k.
- Encrypt M in CBC mode, using the MAC key. Discard $C_1,\ldots,C_{n-1}$ and define $MAC_K(M_1,\ldots,M_n)=C_n$.

Security of CBC-MAC

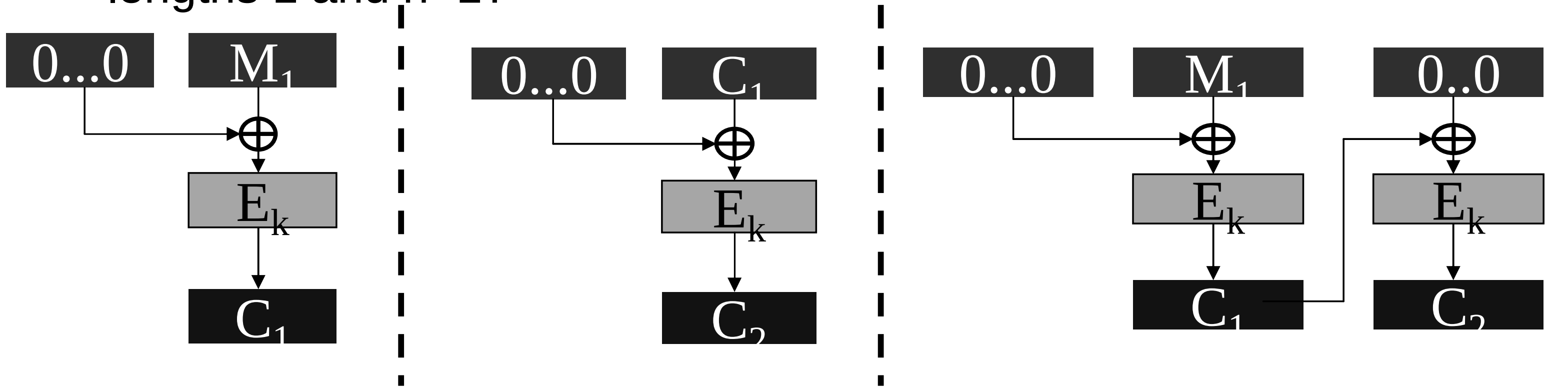
- Claim: if E_K is pseudo-random then CBC-MAC, applied to *fixed length messages*, is a pseudo-random function, and is therefore resilient to forgery.

- But, insecure if variable lengths messages are allowed

Security of CBC-MAC

- Insecurity of CBC-MAC when applied to messages of variable length:
  - Get C_1 = CBC-MAC$_K(M_1) = E_K(0 \oplus M_1)$
  - Ask for MAC of C_1, i.e., C_2 = CBC-MAC$_K(C_1) = E_K(0 \oplus C_1)$
  - But, $E_K(C_1 \oplus 0) = E_K(E_K(0 \oplus M_1) \oplus 0)$ = CBC-MAC$_K(M_1 \mid 0)$

  - It's known that CBC-MAC is secure if message space is prefix-free.
  - Can you show, for every n, a collision between two messages of lengths 1 and n+1?



CBC-MAC for variable length messages

- Solution 1: The first block of the message is set to be its length. I.e., to authenticate $M_1,\ldots,M_n$, apply CBC-MAC to $(n,M_1,\ldots,M_n)$.
  - Works since now message space is prefix-free.
  - Drawback: The message length (n) must be known in advance.
- "Solution 2": apply CBC-MAC to $(M_1,\ldots,M_n,n)$
  - Message length does not have to be known is advance
  - But, this scheme is broken (see, M. Bellare, J. Kilian, P. Rogaway, The Security of Cipher Block Chaining, 1984)
- Solution 3: (preferable)
  - Use a second key K'.
  - Compute $MAC_{K,K'}(M_1,\ldots,M_n) = E_{K'}(MAC_K(M_1,\ldots,M_n))$
  - Essentially the same overhead as CBC-MAC

Hash functions

- MACs can be constructed based on hash functions.
- A hash function $h:X \rightarrow Y$ maps long inputs to fixed size outputs. $(|X|>|Y|)$
- No secret key. The hash function algorithm is public.
- If $|X|>|Y|$ there are collisions ($x \neq x'$ for which $h(x)=h(x')$).

Security definitions for hash functions

1. Weak collision resistance: for any $x \in X$, it is hard to find $x' \neq x$ such that $h(x)=h(x')$. (Also known as "universal one-way hash", or "*second* preimage resistance").
   - In other words, there is no efficient algorithm which is given x and can find an x' such that $h(x)=h(x')$.

2. Strong collision resistance: it is hard to find any x,x' for which h(x)=h(x').
   - In other words, there is no no efficient algorithm which can find a pair x,x' such that $h(x)=h(x')$.

Security definitions for hash functions

- It's easier to find collisions. (Namely, under reasonable assumptions it holds that if it is possible to achieve security according to definition (2) then it is also possible to achieve security according to definition(1).)
- Therefore strong collision resistance is a stronger assumption.

- Real world hash functions: MD5, SHA-1, SHA-256.

Hmm..

The Birthday Phenomenon (Paradox)

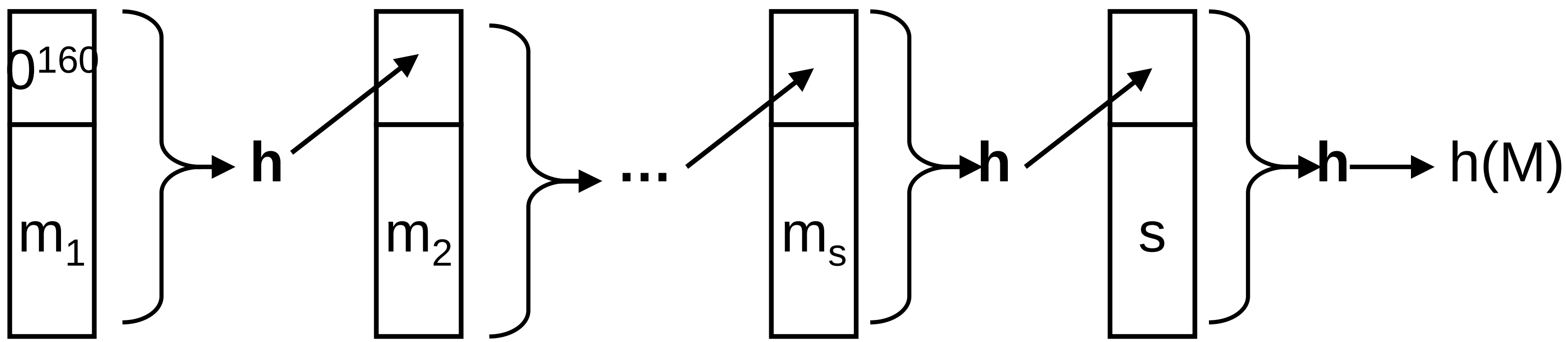
- For 23 people chosen at random, the probability that two of them have the same birthday is ½.
- Compare to: the prob. that one or more of them has the same birthday as Alan Turing is 23/365 (actually, $1-(1-1/365)^{23}$.)

- More generally, for a random $h:X \rightarrow Z$, if we choose about $|Z|^{½}$ elements of Z at random ($1.17\ |Z|^{½}$), the probability that two of them are mapped to the same image is > ½.

- Implication: it's harder to achieve strong collision resistance
  - A random function with a n bit output
    - Find x,x' with h(x)=h(x') after about $2^{n/2}$ tries.
    - Find x≠0 s.t. h(x)=h(0) after about 2^n attempts.

From collision-resistance for fixed length inputs, to collision-resistance for arbitrary input lengths

- Hash function:
  - Input block length is usually 512 bits (|X|=512)
  - Output length is at least 160 bits (birthday attacks)
- Extending the domain to arbitrary inputs (Damgard-Merkle)
  - Suppose $h:\{0,1\}^{512} \rightarrow \{0,1\}^{160}$
  - Input: $M=m_1 \ldots m_s$, $|m_i|=512-160=352$. (what if $|M| \neq 352 \cdot i$ bits?)
  - Define: $y_0=0^{160}$. $y_i=h(y_{i-1},m_i)$. $y_{s+1}=h(y_s,s)$. $h(M)=y_{s+1}$.
  - Why is it secure? What about different length inputs?



Proof

- Show that if we can find $M \neq M'$ for which $H(M)=H(M')$, we can find blocks $m \neq m'$ for which $h(m)=h(m')$.

- Case 1: suppose $|M|=s$, $|M'|=s'$, and $s \neq s'$
  - Then, collision: $H(M)=h(y_s,s) = h(y_{s'},s')=H(M')$
- Case 2: $|M|=|M'|=s$
  - We know that $H(M)=h(y_s,s)=h(y'_s,s)=H(M')$
  - If $y_s \neq y'_s$ then we found a collision in h.
  - Otherwise, go from i=s-1 to i=1:
    - $y_{i+1} = y'_{i+1}$ implies $h(y_i,m_{i+1}) = h(y'_I,m'_{i+1})$.
    - If $y_i \neq y'_i$ or $m_{i+1} \neq m'_{i+1}$, then we found a collision.
    - $M \neq M'$ and therefore there is an i for which $m_{i+1} \neq m'_{i+1}$

The implication of collisions

- Given a hash function with 2^n possible outputs. Collisions can be found
  - after a search of $2^{n/2}$ values
  - even faster if the function is weak (MD5, SHA-1)
- We find x, x' such that h(x)=h(x'), but we cannot control the value of x, x'.
- Can we find "meaningful" colliding values x, x' ?
  - The case of pdf files…

Basing MACs on Hash Functions

- Hash functions are not keyed. MAC_K uses a key.
- Best attack should not succeed with prob > $\max(2^{-|k|}, 2^{-|MAC()|})$.
- Idea: MAC combines message and a secret key, and hashes them with a collision resistant hash function.
  - E.g. $MAC_K(m) = h(k,m)$. (insecure.., given $MAC_K(m)$ can compute $MAC_K(m,|m|,m')$, if using the MD construction)
  - $MAC_K(m) = h(m,k)$. (insecure.., regardless of key length, use a birthday attack to find m,m' such that h(m)=h(m').)

- How should security be proved?:
  - Show that if MAC is insecure then so is hash function h.
  - Insecurity of MAC: adversary can generate $MAC_K(m)$ without knowing k.
  - Insecurity of h: adversary finds collisions ($x \neq x'$, $h(x)=h(x')$.)

HMAC

- Input: message *m,* a key *K*, and a hash function *h*.
- $\text{HMAC}_K(m) = h(\,K \oplus \text{opad}, h(K \oplus \text{ipad}, m))$
  – where ipad, opad are 64 byte long fixed strings
  – K is 64 byte long (if shorter, append 0s to get 64 bytes).
- Overhead: the same as that of applying h to m, plus an additional invocation to a short string.

- It was proven [BCK] that if HMAC is broken then either
  – h is not collision resistant (even when the initial block is random and secret), or
  – The output of h is not “unpredcitable” (when the initial block is random and secret)
- HMAC is used everywhere (SSL, IPSec).

What we learned today

- Differential cryptanalysis of DES
- Message authentication
  - CBC MAC
  - Hash functions
  - The birthday paradox
  - HMAC