Introduction to Cryptography

Lecture 4

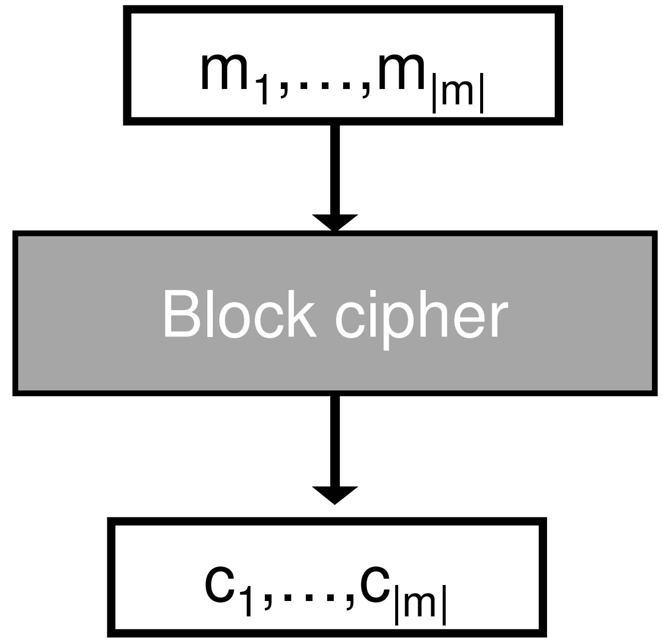
Benny Pinkas

Block Ciphers

- Plaintexts, ciphertexts of **fixed** length, |m|. Usually, |m|=64 or |m|=128 bits.
- The encryption algorithm E_k is a *permutation* over $\{0,1\}^{|m|}$, and the decryption D_k is its inverse. (They *are not* permutations of the bit order, but rather of the entire string.)

- Ideally, use a *random* permutation.
  – Can only be implemented using a table with $2^{|m|}$ entries ☹
- Instead, use a *pseudo-random* permutation, keyed by a key k.
  – Implemented by a computer program whose input is m,k.

- We learned last week how to use a block cipher for encrypting messages longer than the block size.



Block ciphers or stream ciphers?

Performance: Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	Cipher	Block/key size	Speed (MB/sec)
stream	RC4		126
stream	Salsa20/12		643
stream	Sosemanuk		727
block	3DES	64/168	13
block	AES-128	128/128	109

Slide taken from Dan Boneh

Pseudo-random functions (PRFs)

- $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
  - The first input is the key, and once chosen it is kept fixed.
  - For simplicity, assume $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$
  - $F(k,x)$ is written as $F_k(x)$

- F is pseudo-random if $F_k()$ (where k is chosen uniformly at random) is indistinguishable (to a polynomial distinguisher D) from a function *f* chosen at random from all functions mapping $\{0,1\}^n$ to $\{0,1\}^n$
  - There are 2^n choices of F_k, whereas there are $(2^n)^{2^n}$ choices for *f*.
  - The distinguisher D's task:
    - We choose a function G. With probability ½ G is F_k (where $k \in_R \{0,1\}^n$), and with probability ½ it is a random function *f*.
    - D can compute $G(x_1),G(x_2),\dots$ for any $x_1,x_2,\dots$ it chooses.
    - D must say if $G=F_k$ or $G=f$.
    - F_k is pseudo-random if D succeeds with prob ½+negligible..

Pseudo-random permutations (PRPs)

- $F_k(x)$ is a keyed permutation if for every choice of k, $F_k()$ is one-to-one.
  - Note that in this case $F_k(x)$ has an inverse, namely for every y there is exactly one x for which $F_k(x)=y$.

- $F_k(x)$ is a pseudo-random permutation if
  - It is a keyed permutation
  - It is indistinguishable (to a polynomial distinguisher D) from a permutation *f* chosen at random from all permutations mapping $\{0,1\}^n$ to $\{0,1\}^n$.
    - 2^n possible values for F_k
    - $(2^n)!$ possible values for a random permutation
  - It is known how to construct PRPs from PRFs

Block ciphers

- A block cipher is a function $F_k(x)$ with a key k and an |m| bit input x, which has an |m| bit output.
  - $F_k(x)$ is a keyed permutation
  - When analyzing security we assume it to be a PRP (Pseudo-Random Permutation)
- How can we encrypt plaintexts longer than |m|?
- Different modes of operation were designed for this task.
  - Discussed last week.

Practical design of Block Ciphers

- Recall that as with prgs, the design of a block cipher that is provably secure without any assumptions implies P!=NP.

- The design of block ciphers is therefore more an engineering challenge. Based on experience and public scrutiny.

  – It is often based on combining together simple building blocks, which support the following principles:
  – *“Diffusion” (bit shuffling):* each intermediate/output bit is affected by many input bits
  – *“Confusion”*: avoid structural relationships (and in particular, linear relationships) between bits

- Cascaded (round) design: the encryption algorithm is composed of iterative applications of a simple round

Confusion-Diffusion and Substitution-Permutation Networks

- Construct a PRP for a large block using PRPs for small blocks
- Divide the input to small parts, and apply rounds:
  - Feed the parts through PRPs *("confusion")*
  - Mix the parts *("diffusion")*
  - *Repeat*
- Why both confusion and diffusion are necessary?
- Design musts: Avalanche effect. Using reversible s-boxes.

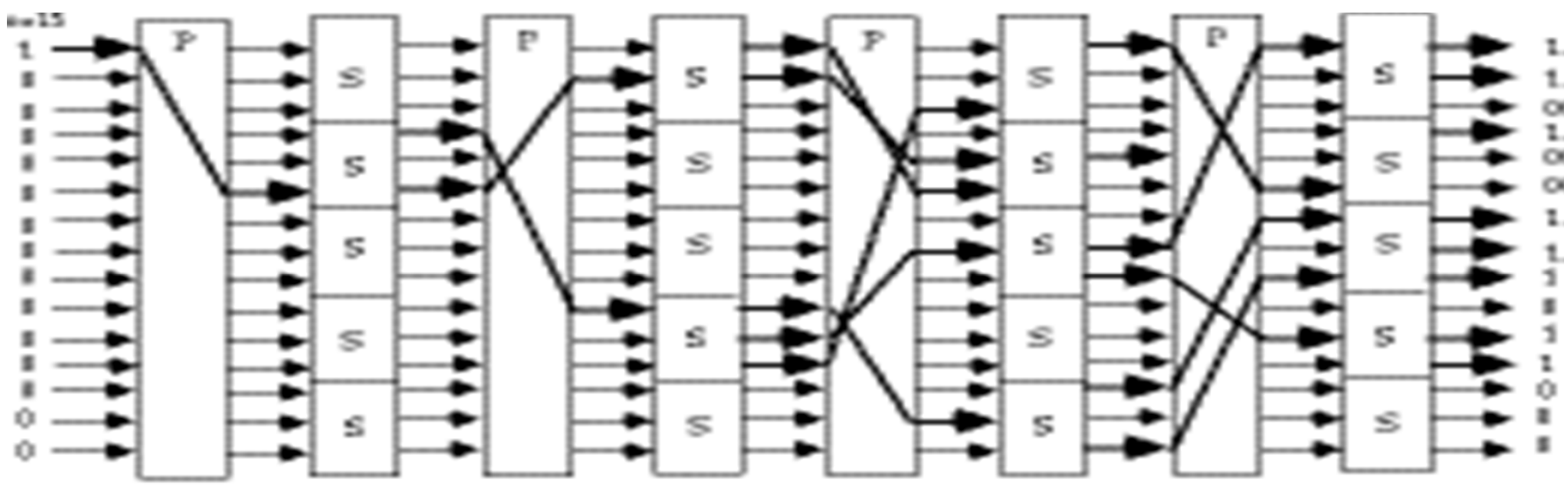


Fig 2.3 - Substitution-Permutation Network with the Avalanche Characteristic

AES (Advanced Encryption Standard)

- Design initiated in 1997 by NIST
  - Goals: improve security and software efficiency of DES
  - 15 submissions, several rounds of public analysis
  - The winning algorithm: Rijndael

- Input block length: 128 bits
- Key length: 128, 192 or 256 bits
- Multiple rounds (10, 12 or 14), but does not use a Feistel network

Rijndael animation

AES

- The S-boxes (SubBytes) are the only non-linear component of AES
  - ShiftRows mixes data in byte level
  - MixColumns mixes blocks of four bytes

- Software implementation
  - A straightforward implementation is well suite for 8bit processors, but does not fully utilize 32b/64b architectures
  - A 32 bit implementation can combine SubBytes, ShiftRows and MixColumns into 16 lookups in tables of 256 32-bit entries
- Hardware implementation: AES is implemented using machine instruction in new Intel processors.

AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES
- **aeskeygenassist**: performs AES key expansion
- Implement AES by doing **aeskeygenassist + 9 x aesenc + aesenclast**
- Claim 14 x speed-up over OpenSSL on same hardware
- Similar instructions on AMD Bulldozer

Slide taken from Dan Boneh

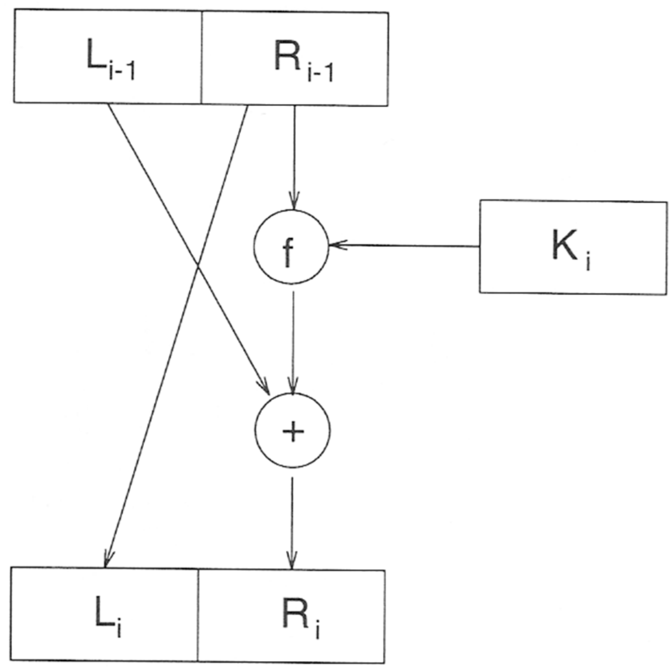
Reversible s-boxes

- Substitution-Permutation networks must use reversible s-boxes
  - Allow for easy decryption
- However, we want the block cipher to be “as random as possible”
  - s-boxes need to have some structure to be reversible
  - Better use non-invertible s-boxes
- Enter Feistel networks
  - A round-based block-cipher which uses s-boxes which are not necessarily reversible
  - Namely, building an invertible function (permutation) from a non-invertible function.

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!).

Security of DES

- Criticized for unpublished design *decisions* (designers did not want to disclose differential cryptanalysis).

- Very secure – the best attack in practice is brute force
  - 2006: $1 million search machine: 30 seconds
    - cost per key: less than $1
  - •2006: 1000 PCs at night: 1 month
    - Cost per key: essentially 0 (+ some patience)

- Some theoretical attacks were discovered in the 90s:
  - Differential cryptanalysis
  - Linear cryptanalysis: requires about 2^{40} known plaintexts
- The use of DES is not recommend since 2004 , but 3-DES is still recommended for use.

Iterated ciphers

- Suppose that E_k is a good cipher, with a key of length *k* bits and plaintext/ciphertext of length *n*.
  - The best attack on E_k is a brute force attack with has O(1) plaintext/ciphertext pairs, and goes over all 2^k possible keys searching for the one which results in these pairs.

- New technological advances make it possible to run this brute force exhaustive search attack. What shall we do?
  - Design a new cipher with a longer key.
  - Encrypt messages using *two* keys k_1, k_2, and the encryption function $E_{k2}(E_{k1}())$. Hoping that the best brute force attack would take $(2^k)^2=2^{2k}$ time.

Iterated ciphers – what can go wrong?

- If encryption is closed under composition, namely for all k_1,k_2 there is a k_3 such that $E_{k2}(E_{k1}())=E_{k3}()$, then we gain nothing.
  - Could just exhaustively search for k_3, instead of separately searching for k_1 and k_2.
  - Substitution ciphers definitely have this property (in fact, they are a permutation group and therefore closed under composition).
  - It was suspected that DES is a group under composition. This assumption was refuted only in 1992.

Iterated Ciphers - Double DES

- DES is out of date due to brute force attacks on its short key (56 bits)

- Why not apply DES twice with two keys?
  - Double DES: $DES_{k1,k2} = E_{k2}(E_{k1}(m))$
  - Key length: 112 bits

- But, double DES is susceptible to a meet-in-the-middle attack, requiring $\approx 2^{56}$ operations and storage.
  - Compared to brute a force attack, requiring 2^{112} operations and O(1) storage.

Meet-in-the-middle attack

- Meet-in-the-middle attack
  - $c = E_{k2}(E_{k1}(m))$
  - $D_{k2}(c) = E_{k1}(m)$
- The attack:
  - Input: *(m,c)* for which $c = E_{k2}(E_{k1}(m))$
  - For every possible value of k_1, generate and store $E_{k1}(m)$.
  - For every possible value of k_2, generate and store $D_{k2}(c)$.
  - Match k_1 and k_2 for which $E_{k1}(m) = D_{k2}(c)$.
  - Might obtain several options for (k_1,k_2). Check them or repeat the process again with a new *(m,c)* pair (see next slide)
- The attack is applicable to any iterated cipher. Running time and memory are $O(2^{|k|})$, where $|k|$ is the key size.

Meet-in-the-middle attack: how many pairs to check?

- The plaintext and the ciphertext are 64 bits long
- The key is 56 bits long

- Suppose that we are given one plaintext-ciphertext pair (m,c)
  - The attack looks for k1,k2, such that $D_{k2}(c) = E_{k1}(m)$
  - The correct values of k1,k2 satisfy this equality
  - There are 2^{112} (actually $2^{112}-1$) other values for k_1,k_2.
  - Each one of these satisfies the equalities with probability 2^{-64}
  - We therefore expect to have $2^{112-64}=2^{48}$ candidates for k_1,k_2.

- Suppose that we are given two pairs (m,c), (m',c')
  - The correct values of k1,k2 satisfy both equalities
  - There are 2^{112} (actually $2^{112}-1$) other values for k_1,k_2.
  - Each one of these satisfies the equalities with probability 2^{-128}
  - We therefore expect to have $2^{112-128}<1$ false candidates for k_1,k_2.

Triple DES

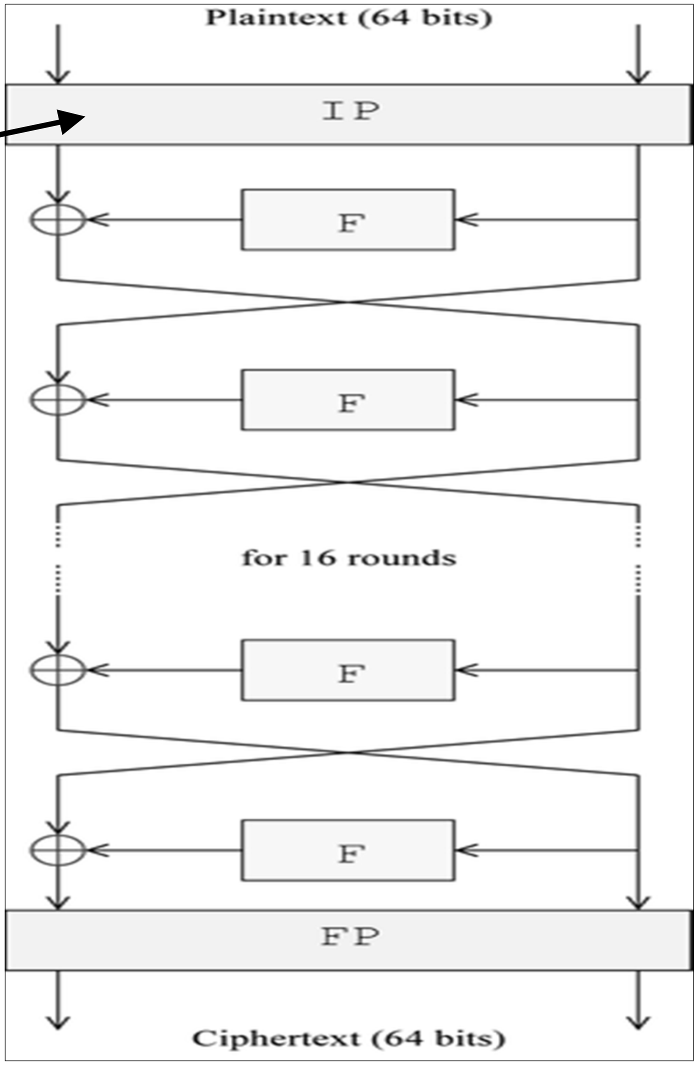
- $3DES_{k1,k2,k3} = E_{k3}(D_{k2}(E_{k1}(m))$
- $\text{Two-key-3DES}_{k1,k2} = E_{k1}(D_{k2}(E_{k1}(m))$

- Why use Enc(Dec(Enc( ))) ?
  - Backward compatibility: setting $k_1=k_2$ is compatible with single key DES

- Two-key-3DES (key length is only 112 bits)
  - There is an attack which requires 2^{56} work and memory, but needs also 2^{56} encryptions of *chosen* plaintexts. Therefore not practical.
  - Without chosen plaintext, best attack needs 2^{112} work and memory.
  - Why isn't it better to use 3DES with three keys? There is a meet-in-the-middle attack against three keys with 2^{112} operations

- 3DES is widely used. Less efficient than DES.

Internals of DES

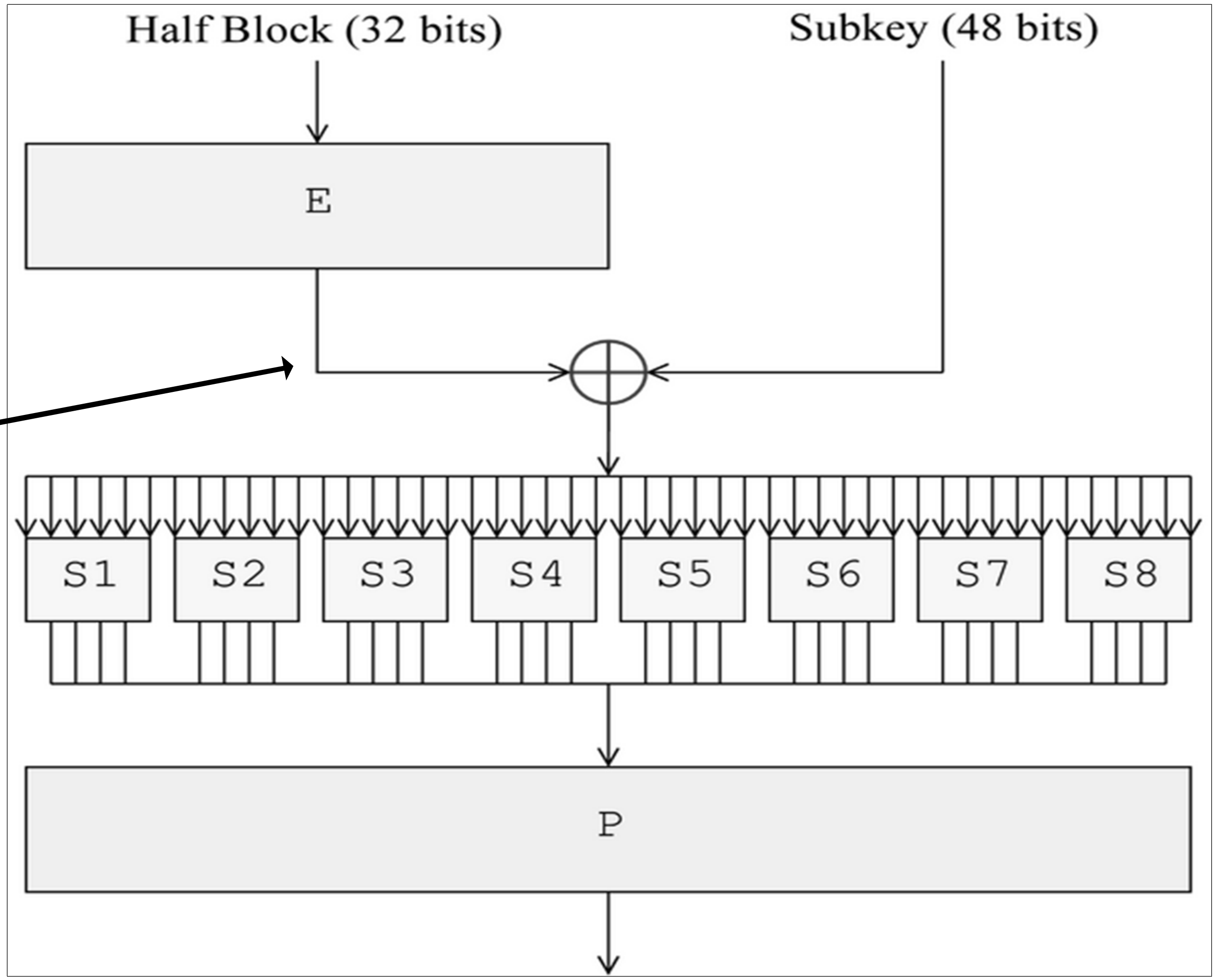


Initial permutation of bit locations:

- not secret
- makes implementations in software less efficient

DES F functions

Expansion to 48 bits



The S-boxes

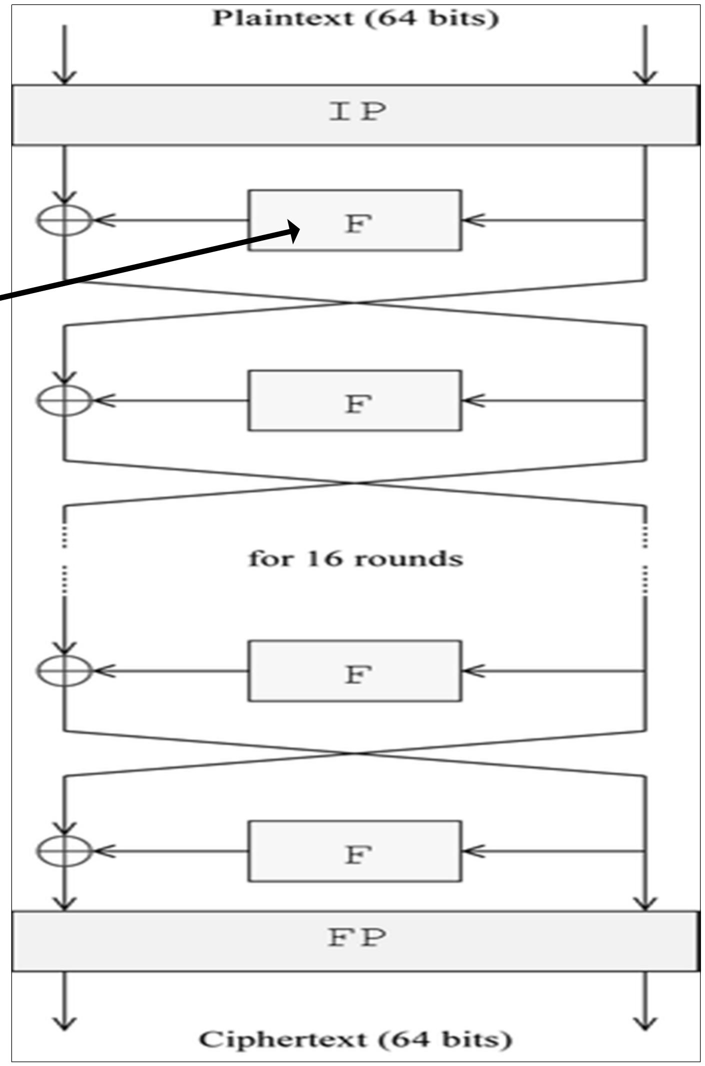
- Very careful design (it is now known that random choices for the S-boxes result in weak encryption).

- 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
  - Curcial property: Changing one input bit changes at least two output bits ⇒ avalanche effect.

Differential Cryptanalysis of DES

DES diagram:

S-boxes



Differential Cryptanalysis [Biham-Shamir 1990]

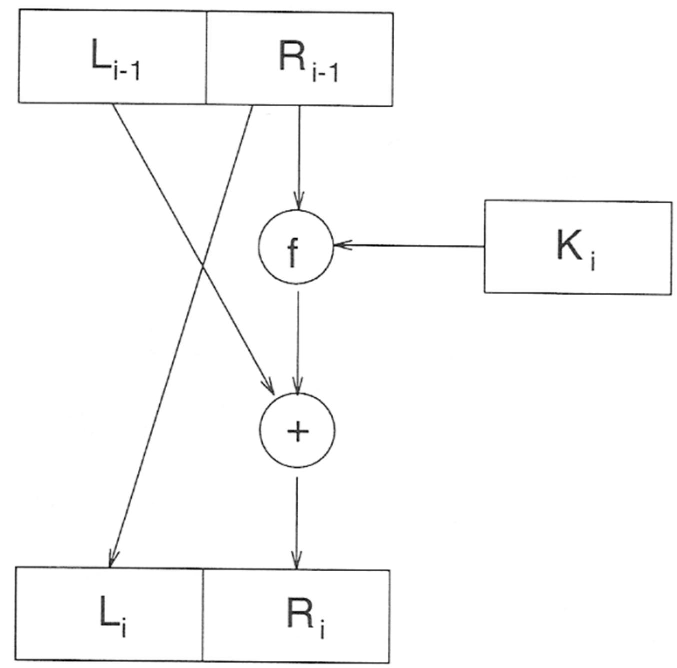
- The first attack to reduce the overhead of breaking DES to below exhaustive search
- Very powerful when applied to other encryption algorithms

- 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 (a known) permuted order
- Linear relations can be exposed by solving a system of linear equations

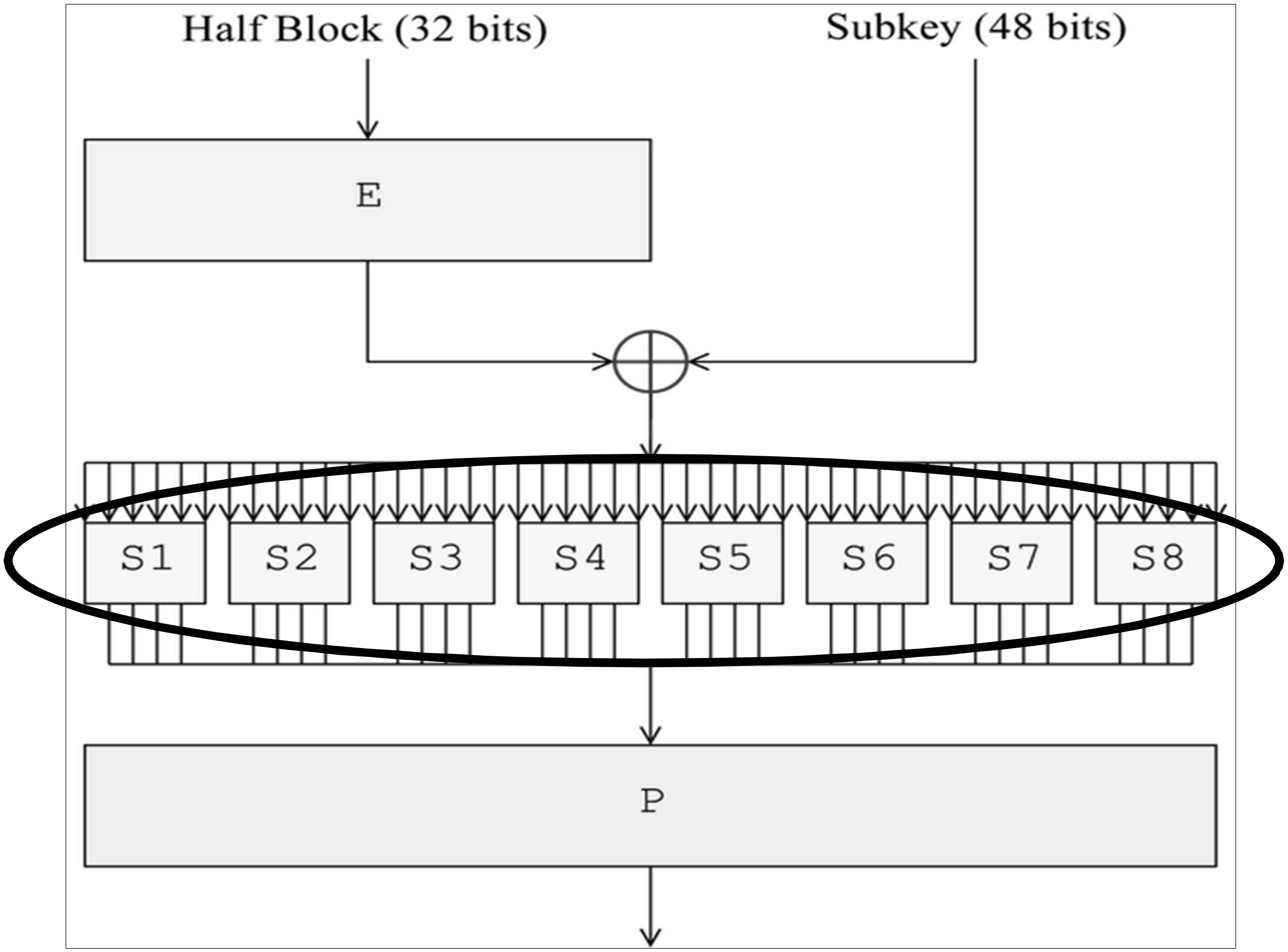
Is a Linear F in a Feistel Network secure?

- Suppose $F(R_{i-1},K_i) = R_{i-1} \oplus K_i$
  - Namely, 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.



DES F functions



Source of
non-linearity

Differential Cryptanalysis

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

- Notation:
  - Denote two different plaintexts as P and P*
  - Their difference 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=0, 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

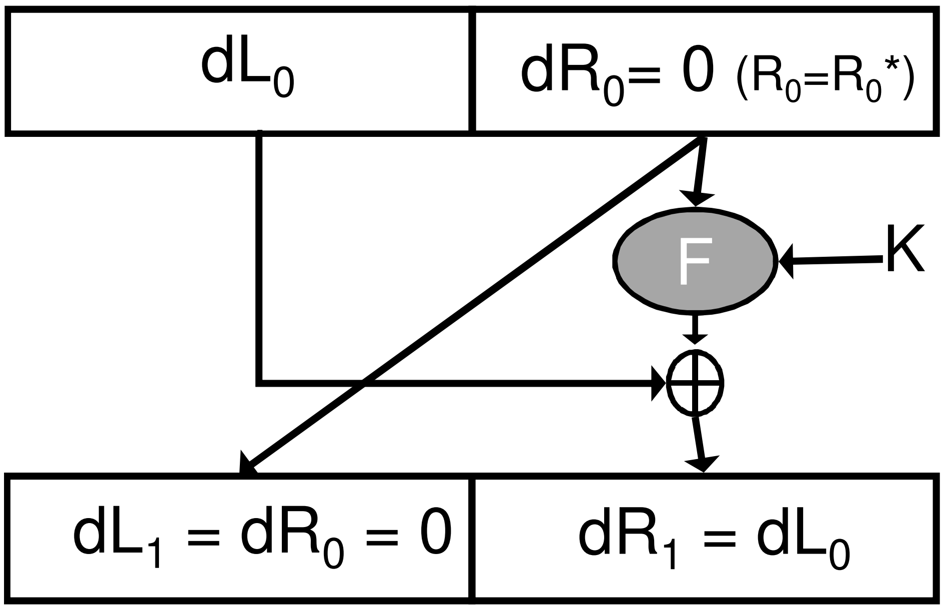
Differential Probabilities

- 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.

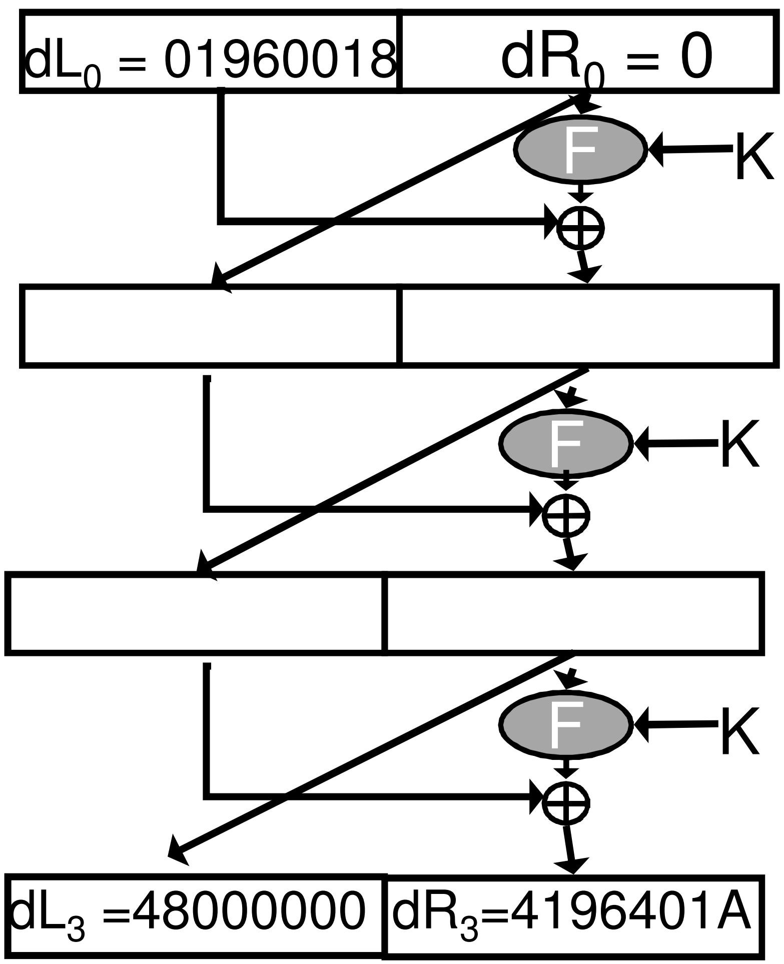
- Differential cryptanalysis uses entries with large values
  - dX=0 $\Rightarrow$ dY=0
  - Entries with value 16/64
  - (Recall that the outputs of the S-box are uniformly distributed, so the attacker gains a lot by looking at differentials rather than the original values.)

Warmup

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

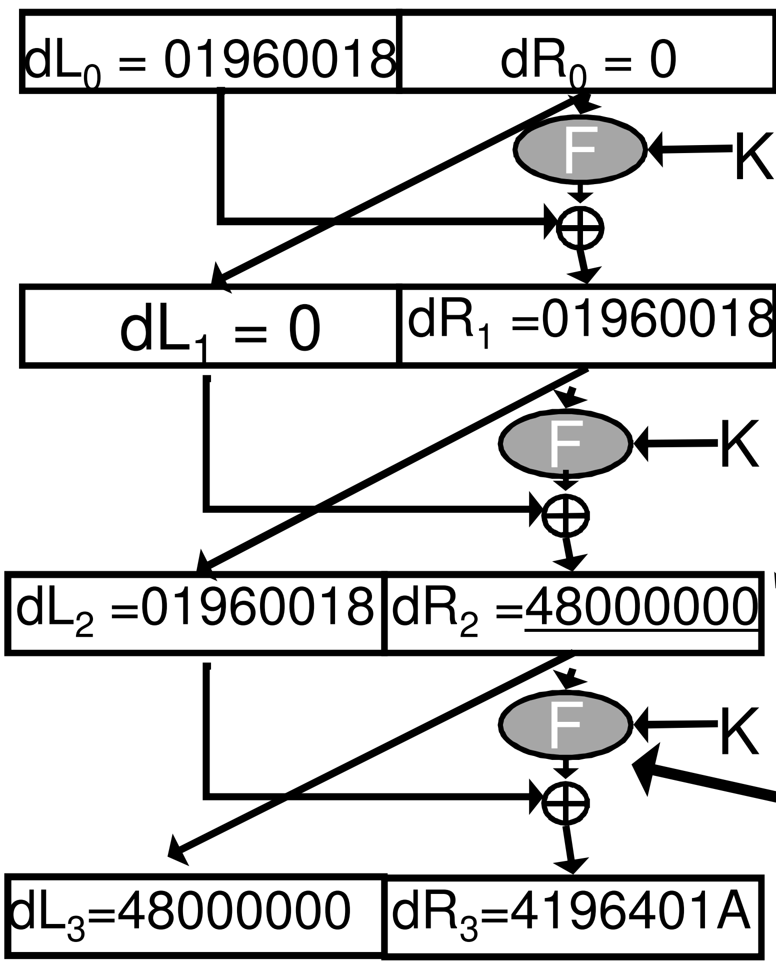


3 Round DES



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

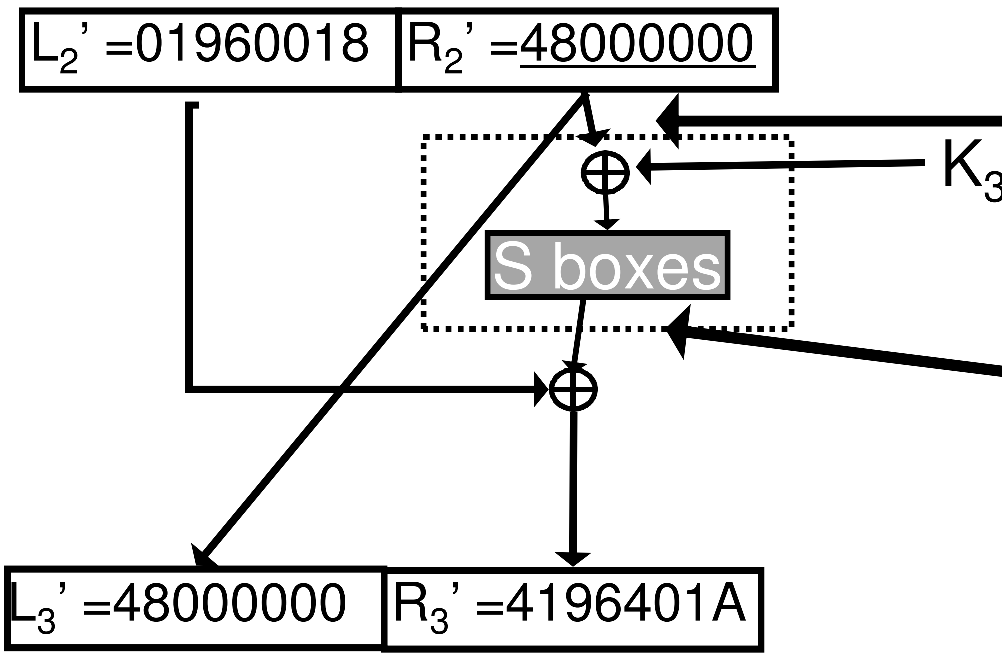
Intermediate differences equal to plaintext/ciphertext differences



Note that here the adversary also knows the actual two values

dF = 4196401A
⊕ 01960018
= 40004002

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...

Linear cryptanalysis of DES [BS'89,M'93]

Given *many* inp/out pairs, can recover key in time less than 2^{56} .

<u>Linear cryptanalysis</u> (overview) : let c = DES(k, m)

Suppose for random k,m :

$$\Pr\Big[\ m[i_1]\oplus\cdots\oplus m[i_r]\ \oplus\ c[j_j]\oplus\cdots\oplus c[j_v]\ =\ k[l_1]\oplus\cdots\oplus k[l_u]\ \Big] = \tfrac{1}{2} + \varepsilon$$

For some ε.

For DES, this exists with $\varepsilon = 1/2^{21} \approx 0.0000000477$

Slide taken from Dan Boneh

Linear attacks

$$\Pr\Big[\ m[i_1]\oplus\cdots\oplus m[i_r]\ \oplus\ c[j_j]\oplus\cdots\oplus c[j_v]\ =\ k[l_1]\oplus\cdots\oplus k[l_u]\ \Big] = ½ + \varepsilon$$

Thm: given $1/\varepsilon^2$ random $\big(m, c=DES(k, m)\big)$ pairs then

$$k[l_1,\ldots,l_u]\ =\ MAJ\Big[\ \ m[i_1,\ldots,i_r]\ \oplus\ c[j_j,\ldots,j_v]\ \ \Big]$$

with prob. $\geq 97.7\%$

$\Rightarrow$ with $1/\varepsilon^2$ inp/out pairs can find $k[l_1,\ldots,l_u]$ in time $\approx 1/\varepsilon^2$

.

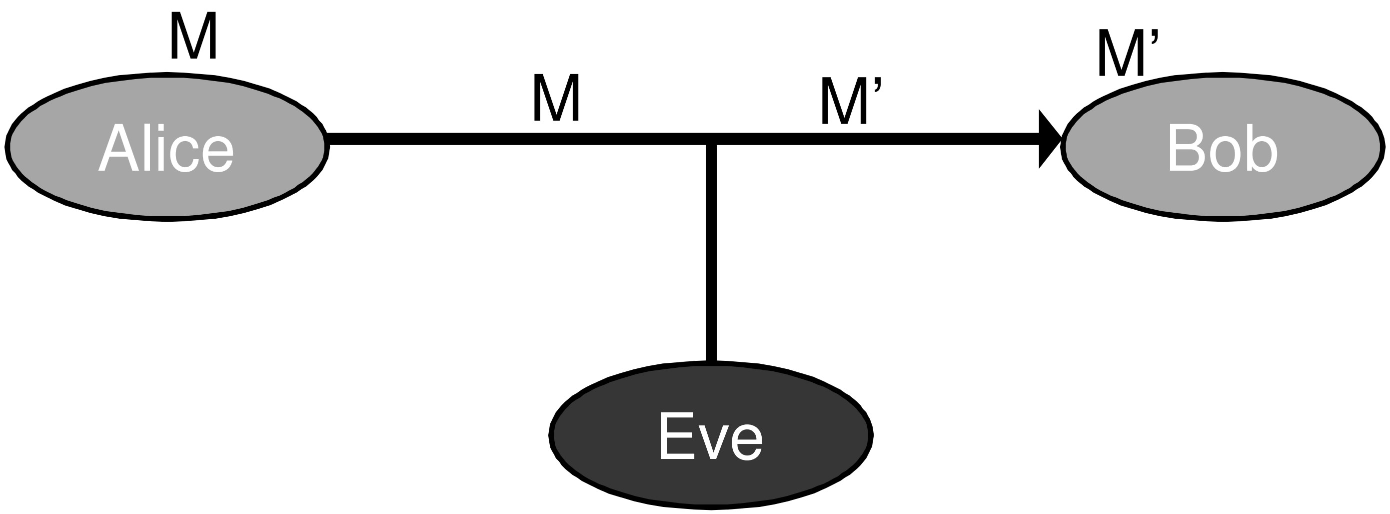
Linear attacks

- For DES, $\varepsilon = 1/2^{21} \Rightarrow$
  - with 2^{42} inp/out pairs can find $k[l_1,\ldots,l_u]$ in time 2^{42}
  - Roughly speaking: can find 14 key "bits" this way in time 2^{42}
  - Apply a brute force attack against remaining $56-14=42$ bits in time 2^{42}
- Total attack time $\approx 2^{43}$ ($<< 2^{56}$)
  - but only if you have 2^{42} random inp/out pairs ☹

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 $k_1k_2\ldots k_n$
  - 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 $1\oplus x_2$

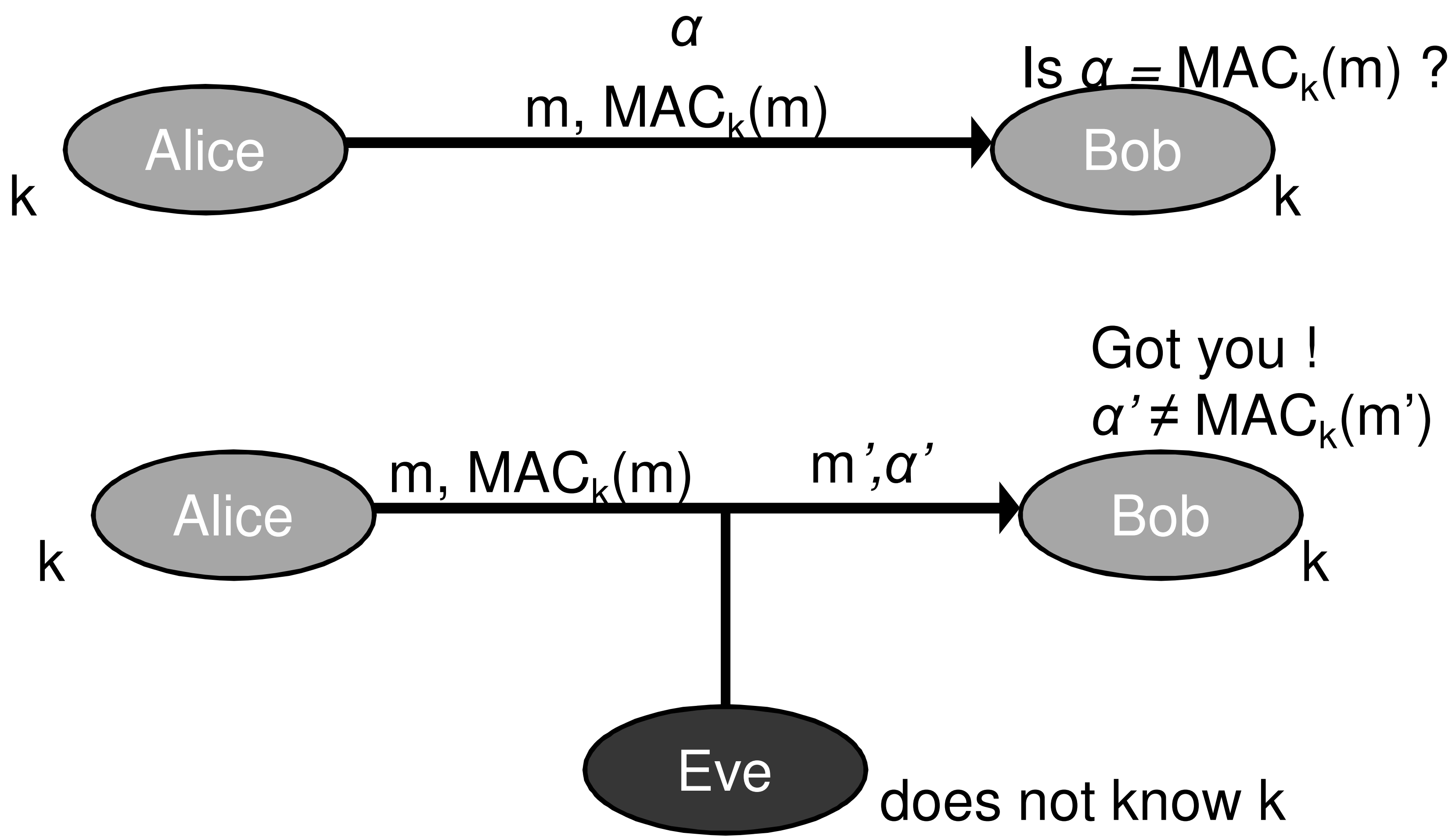
- Error-detection codes are insufficient. (For example, linear codes can be changed by the adversary, even if encrypted.)
  - They were not designed to withstand adversarial behavior.

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

- Length of MAC output must be at least n bits, if we do not want the cheating probability to be greater than 2^{-n}

- Constructions of 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.