BLOCK CIPHERS

Permutations and Inverses

A function $f\colon \{0,1\}^\ell \to \{0,1\}^\ell$ is a permutation if there is an inverse function $f^{-1}\colon \{0,1\}^\ell \to \{0,1\}^\ell$ satisfying

$$\forall x \in \{0,1\}^\ell : f^{-1}(f(x)) = x$$

This means f must be one-to-one and onto, meaning for every $y \in \{0,1\}^\ell$ there is a unique $x \in \{0,1\}^\ell$ such that $f(x) = y$.

Permutations and Inverses

x	00	01	10	11
$f(x)$	01	11	00	10

A permutation

x	00	01	10	11
$f(x)$	01	11	11	10

Not a permutation

Permutations and Inverses

x	00	01	10	11
$f(x)$	01	11	00	10

A permutation

x	00	01	10	11
$f^{-1}(x)$	10	00	11	01

Its inverse

Block Ciphers

Let

$$E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$$

be a function taking a key K and input x to return output $E(K,x)$. For each key K we let $E_K\colon \{0,1\}^\ell \to \{0,1\}^\ell$ be the function defined by

$$E_K(x) = E(K,x) \ .$$

We say that E is a block cipher if

- $E_K\colon \{0,1\}^\ell \to \{0,1\}^\ell$ is a permutation for every K, meaning has an inverse E_K^{-1},
- E, E^{-1} are efficiently computable,

where $E^{-1}(K,x) = E_K^{-1}(x)$.

Example

The table entry corresponding to the key in row K and input in column x is $E_K(x)$.

	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

In this case, the inverse cipher E^{-1} is given by the same table: the table entry corresponding to the key in row K and output in column y is $E_K^{-1}(y)$.

Block Ciphers: Example

Let $\ell = k$ and define $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ by

$$E_K(x) = E(K,x) = K \oplus x$$

Then E_K has inverse E_K^{-1} where

$$E_K^{-1}(y) = K \oplus y$$

Why? Because

$$E_K^{-1}(E_K(x)) = E_K^{-1}(K \oplus x) = K \oplus K \oplus x = x$$

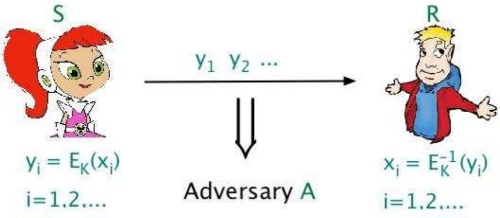
The inverse of block cipher E is the block cipher E^{-1} defined by

$$E^{-1}(K,y) = E_K^{-1}(y) = K \oplus y$$

Block cipher usage

- $K \xleftarrow{\$} \{0,1\}^k$
- K (magically) given to parties S, R, but not to A.
- S,R use E_K

Algorithm E is public! Think of E_K as encryption under key K.



Leads to security requirements like:

- Hard to get K from $y_1, y_2, \ldots$
- Hard to get x_i from y_i

DES History

1972 – NBS (now NIST) asked for a block cipher for standardization

1974 – IBM designs Lucifer

Lucifer eventually evolved into DES.

Widely adopted as a standard including by ANSI and American Bankers association

Used in ATM machines

Replaced (by AES) only a few years ago

DES parameters

Key Length $k = 56$

Block length $\ell = 64$

So,

$$\text{DES}\colon \{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$$

$$\text{DES}^{-1}\colon \{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$$

DES Construction

function $\text{DES}_K(M)$ // $|K| = 56$ and $|M| = 64$

$(K_1, \ldots, K_{16}) \leftarrow \mathit{KeySchedule}(K)$ // $|K_i| = 48$ for $1 \le i \le 16$

$M \leftarrow \mathit{IP}(M)$

Parse M as $L_0 \,\|\, R_0$ // $|L_0| = |R_0| = 32$

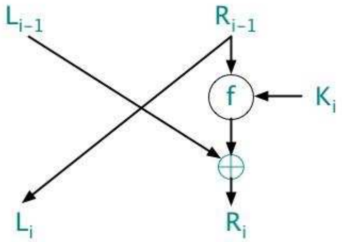
for $i = 1$ to 16 do

$L_i \leftarrow R_{i-1}$; $R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}$

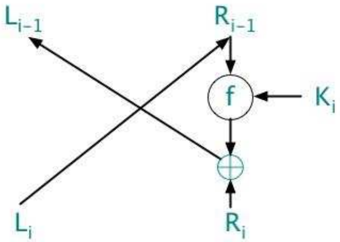
$C \leftarrow \mathit{IP}^{-1}(L_{16} \,\|\, R_{16})$

return C

Round i:



Invertible given K_i:



DES Construction

function $\mathsf{DES}_K(M)$ // $|K| = 56$ and $|M| = 64$

- $(K_1, \ldots, K_{16}) \leftarrow \mathit{KeySchedule}(K)$ // $|K_i| = 48$ for $1 \leq i \leq 16$
- $M \leftarrow IP(M)$
- Parse M as $L_0 \parallel R_0$ // $|L_0| = |R_0| = 32$
- for $i = 1$ to 16 do
  - $L_i \leftarrow R_{i-1}$; $R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}$
- $C \leftarrow IP^{-1}(L_{16} \parallel R_{16})$
- return C

function $\mathsf{DES}_K^{-1}(C)$ // $|K| = 56$ and $|M| = 64$

- $(K_1, \ldots, K_{16}) \leftarrow \mathit{KeySchedule}(K)$ // $|K_i| = 48$ for $1 \leq i \leq 16$
- $C \leftarrow IP(C)$
- Parse C as $L_{16} \parallel R_{16}$
- for $i = 16$ downto 1 do
  - $R_{i-1} \leftarrow L_i$; $L_{i-1} \leftarrow f(K_i, R_{i-1}) \oplus R_i$
- $M \leftarrow IP^{-1}(L_0 \parallel R_0)$
- return M

DES Construction

function $\mathsf{DES}_K(M)$ // $|K| = 56$ and $|M| = 64$

$(K_1, \ldots, K_{16}) \leftarrow KeySchedule(K)$ // $|K_i| = 48$ for $1 \leq i \leq 16$

$M \leftarrow IP(M)$

Parse M as $L_0 \parallel R_0$ // $|L_0| = |R_0| = 32$

for $i = 1$ to 16 do

$L_i \leftarrow R_{i-1}$; $R_i \leftarrow f(K_i, R_{i-1}) \oplus L_{i-1}$

$C \leftarrow IP^{-1}(L_{16} \parallel R_{16})$

return C

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES Construction

function $f(J, R)$ // $|J| = 48$ and $|R| = 32$
  $R \leftarrow E(R)$; $R \leftarrow R \oplus J$
  Parse R as $R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // $|R_i| = 6$ for $1 \leq i$
  for $i = 1, \ldots, 8$ do
    $R_i \leftarrow \mathbf{S}_i(R_i)$ // Each S-box returns 4 bits
  $R \leftarrow R_1 \| R_2 \| R_3 \| R_4 \| R_5 \| R_6 \| R_7 \| R_8$ // $|R| = 32$ bits
  $R \leftarrow P(R)$
  return R

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

S-boxes

$\mathbf{S}_1$:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0
0	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3
1	0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5
1	1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6

$\mathbf{S}_2$:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5
0	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11
1	0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2
1	1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14

$\mathbf{S}_3$:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2
0	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15
1	0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14
1	1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2

Figure: The DES S-boxes.

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
$T \xleftarrow{\$} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

Cryptanalysis: Key Recovery Attacks on Block Ciphers

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
$T \xleftarrow{\$} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

Certainly A should be given $C_1, \ldots, C_q$. But why does A know $M_1, \ldots, M_q$?

- A posteriori revelation of data
- A priori knowledge of context

Good to be conservative!

A posteriori revelation of data

- S, R share key K
- On January 10, S encrypts

$$M = \text{Let's meet tomorrow at 5 pm}$$

  and sends ciphertext C to R.
- Adversary captures C
- On January 11, adversary observes S, R meeting at 5 pm and deduces that M is as above
- Adversary knows C and its decryption M

A priori knowledge of context

- S, R share key K
- E-mails always begin with the keyword "From"
- S encrypts an email
- Adversary gets ciphertext C
- Since it knows part of the plaintext ("From") it may have an input-output example of the block cipher under K

Cryptanalysis: Key Recovery Attacks on Block Ciphers

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \rightarrow \{0,1\}^\ell$
$T \stackrel{\$}{\leftarrow} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

Cryptanalysis: Key Recovery Attacks on Block Ciphers

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
$T \stackrel{\$}{\leftarrow} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

Types of attacks

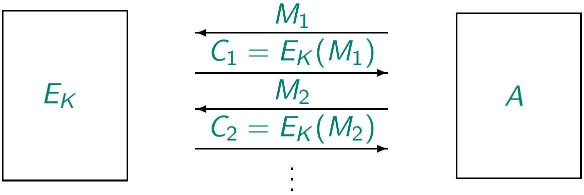
Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Known Message Attack: $M_1, \ldots, M_q$ arbitrary, not chosen by A.

Types of attacks

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Chosen Message Attack: A can pick $M_1, \ldots, M_q$, even adaptively, meaning pick M_i as a function of $(M_1, C_1), \ldots, (M_{i-1}, C_{i-1})$ for $i = 1, \ldots, q$.



Examples:

- A sends S e-mails which S encrypts and forwards to R
- S is a router encrypting any packet it receives

Cryptanalysis: Key Recovery Attacks on Block Ciphers

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
$T \xleftarrow{\$} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
    for $i = 1, \ldots, 2^k$ do
        if $E(T_i, M_1) = C_1$ then return T_i

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $\mathit{EKS}_E(M_1, C_1)$
  for $i = 1, \ldots, 2^k$ do
    if $E(T_i, M_1) = C_1$ then return T_i

Does this find the target key T?

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
for $i = 1, \ldots, 2^k$ do
if $E(T_i, M_1) = C_1$ then return T_i

Does this find the target key T?

Definition: A key K is consistent with (M_1, C_1) if $C_1 = E(K, M_1)$

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
for $i = 1, \ldots, 2^k$ do
if $E(T_i, M_1) = C_1$ then return T_i

Does this find the target key T?

Definition: A key K is consistent with (M_1, C_1) if $C_1 = E(K, M_1)$

Let S be the set of all keys consistent with (M_1, C_1). Then EKS_E finds some key in S.

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \stackrel{\$}{\leftarrow} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
  for $i = 1, \ldots, 2^k$ do
    if $E(T_i, M_1) = C_1$ then return T_i

Does this find the target key T?

Definition: A key K is consistent with (M_1, C_1) if $C_1 = E(K, M_1)$

Let S be the set of all keys consistent with (M_1, C_1). Then EKS_E finds some key in S.

Fact: If $\ell \geq k$ then T is "usually" the only key in S.

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \stackrel{\$}{\leftarrow} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $\mathit{EKS}_E(M_1, C_1)$
    for $i = 1, \ldots, 2^k$ do
        if $E(T_i, M_1) = C_1$ then return T_i

Does this find the target key T? Yes, usually.

Increasing likelihood of getting target key

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let $(M_1, C_1), \ldots, (M_q, C_q)$ satisfy $E_T(M_i) = C_i$ for all $1 \le i \le q$.

algorithm $EKS_E((M_1, C_1), \ldots, (M_q, C_q))$
    for $i = 1, \ldots, 2^k$ do
        if ($E(T_i, M_1) = C_1$ and $\cdots$ and $E(T_i, M_q) = C_q$) then
            return T_i

Exhaustive Key Search

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
    for $i = 1, \ldots, 2^k$ do
        if $E(T_i, M_1) = C_1$ then return T_i

How long does exhaustive key search take?

DES can be computed at 1.6 Gbits/sec in hardware.

DES plaintext = 64 bits

Chip can perform $(1.6 \times 10^9)/64 = 2.5 \times 10^7$ DES computations per second

Expect EKS to succeed in 2^{55} DES computations, so it takes time

$$\frac{2^{55}}{2.5 \times 10^7} \approx 1.4 \times 10^9 \text{ seconds} \approx 45 \text{ years!}$$

Key Complementation $\Rightarrow$ 22.5 years

But this is prohibitive.

Does this mean DES is secure?

Differential and linear cryptanalysis

Exhaustive key search is a generic attack: Did not attempt to "look inside" DES and find/exploit weaknesses.

Method	when	q	Type of attack
Differential cryptanalysis	1992	2^{47}	Chosen-message
Linear cryptanalysis	1993	2^{44}	Known-message

Differential and linear cryptanalysis

Exhaustive key search is a generic attack: Did not attempt to "look inside" DES and find/exploit weaknesses.

Method	when	q	Type of attack
Differential cryptanalysis	1992	2^{47}	Chosen-message
Linear cryptanalysis	1993	2^{44}	Known-message

But merely storing 2^{44} input-output pairs requires 281 Tera-bytes.

In practice these attacks are prohibitively expensive.

EKS revisited

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \xleftarrow{\$} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
    for $i = 1, \ldots, 2^k$ do
        if $E(T_i, M_1) = C_1$ then return T_i

EKS revisited

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \stackrel{\$}{\leftarrow} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
    for $i = 1, \ldots, 2^k$ do
        if $E(T_i, M_1) = C_1$ then return T_i

Observation: The E computations can be performed in parallel.

EKS revisited

Let $T_1, \ldots, T_{2^k}$ be a list of all k bit keys. Let $T \stackrel{\$}{\leftarrow} \{0,1\}^k$ be the target key and let (M_1, C_1) satisfy $E_T(M_1) = C_1$.

algorithm $EKS_E(M_1, C_1)$
  for $i = 1, \ldots, 2^k$ do
    if $E(T_i, M_1) = C_1$ then return T_i

Observation: The E computations can be performed in parallel.

- Wiener 1993:
  - \$1 million
  - 57 chips
  - Finds key in 3.5 hours
- EFF
  - \$250,000
  - Finds key in 56 hours

DES security summary

DES is considered broken because its short key size permits rapid key-search.

But DES is a very strong design as evidenced by the fact that there are no practical attacks that exploit its structure.

2DES

Block cipher $2DES : \{0,1\}^{112} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$ is defined by

$$2DES_{K_1 K_2}(M) = DES_{K_2}(DES_{K_1}(M))$$

- Exhaustive key search takes 2^{112} DES computations, which is too much even for machines
- Resistant to differential and linear cryptanalysis.

Meet-in-the-middle attack on 2DES

Suppose K_1K_2 is a target 2DES key and adversary has M, C such that

$$C = 2DES_{K_1K_2}(M) = DES_{K_2}(DES_{K_1}(M))$$

Then

$$DES^{-1}_{K_2}(C) = DES_{K_1}(M)$$

Meet-in-the-middle attack on 2DES

Suppose $DES_{K_2}^{-1}(C) = DES_{K_1}(M)$ and $T_1, \ldots, T_N$ are all possible DES keys, where $N = 2^{56}$.

T_1	$DES(T_1, M)$
T_i	$DES(T_i, M)$
T_N	$DES(T_N, M)$

Table L

$DES^{-1}(T_1, C)$	T_1
$DES^{-1}(T_j, C)$	T_j
$DES^{-1}(T_N, C)$	T_N

Table R

Attack idea:

- Build L,R tables

Meet-in-the-middle attack on 2DES

Suppose $DES^{-1}_{K_2}(C) = DES_{K_1}(M)$ and $T_1, \ldots, T_N$ are all possible DES keys, where $N = 2^{56}$.

$K_1 \longrightarrow$

T_1	$DES(T_1, M)$
T_i	$DES(T_i, M)$
T_N	$DES(T_N, M)$

Table L

$\overset{\text{equal}}{\longleftrightarrow}$

$DES^{-1}(T_1, C)$	T_1
$DES^{-1}(T_j, C)$	T_j
$DES^{-1}(T_N, C)$	T_N

Table R

$\longleftarrow K_2$

Attack idea:

- Build L,R tables
- Find i, j s.t. $L[i] = R[j]$
- Guess that $K_1 K_2 = T_i T_j$

Meet-in-the-middle attack on 2DES

Let $T_1, \ldots, T_{2^{56}}$ denote an enumeration of DES keys.

$MinM_{2\text{DES}}(M_1, C_1)$
    for $i = 1, \ldots, 2^{56}$ do $L[i] \leftarrow \text{DES}(T_i, M_1)$
    for $j = 1, \ldots, 2^{56}$ do $R[j] \leftarrow \text{DES}^{-1}(T_j, C_1)$
    $S \leftarrow \{ (i,j) \; : \; L[i] = R[j] \}$
    Pick some $(l, r) \in S$ and return $T_l \parallel T_r$

Attack takes about 2^{57} DES/DES^{-1} computations.

Interesting, but not practical.

3DES

Block ciphers

$$3DES3 : \{0,1\}^{168} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

$$3DES2 : \{0,1\}^{112} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

are defined by

$$3DES3_{K_1 \,\|\, K_2 \,\|\, K_3}(M) = DES_{K_3}(DES^{-1}_{K_2}(DES_{K_1}(M))$$

$$3DES2_{K_1 \,\|\, K_2}(M) = DES_{K_2}(DES^{-1}_{K_1}(DES_{K_2}(M))$$

Meet-in-the-middle attack on 3DES3 reduces its "effective" key length to 112.

DESX

$$DESX_{KK_1K_2}(M) = K_2 \oplus DES_K(K_1 \oplus M)$$

- Key length $= 56 + 64 + 64 = 184$
- "effective" key length $= 120$ due to a 2^{120} time meet-in-middle attack
- No more resistant than DES to linear or differential cryptanalysis

Good practical replacement for DES that has lower computational cost than 2DES or 3DES.

Block size limitation

Later we will see "birthday" attacks that "break" a block cipher $E : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ in time $2^{\ell/2}$

For DES this is $2^{64/2} = 2^{32}$ which is small, and this is unchanged for 2DES and 3DES.

Would like a larger block size.

AES

1998: NIST announces competition for a new block cipher

- key length 128
- block length 128
- faster than DES in software

Submissions from all over the world: MARS, Rijndael, Two-Fish, RC6, Serpent, Loki97, Cast-256, Frog, DFC, Magenta, E2, Crypton, HPC, Safer+, Deal

AES

1998: NIST announces competition for a new block cipher

- key length 128
- block length 128
- faster than DES in software

Submissions from all over the world: MARS, Rijndael, Two-Fish, RC6, Serpent, Loki97, Cast-256, Frog, DFC, Magenta, E2, Crypton, HPC, Safer+, Deal

2001: NIST selects Rijndael to be AES.

AES

function $\mathsf{AES}_K(M)$
$(K_0, \ldots, K_{10}) \leftarrow expand(K)$
$s \leftarrow M \oplus K_0$
for $r = 1$ to 10 do
$s \leftarrow S(s)$
$s \leftarrow shift\text{-}rows(s)$
if $r \leq 9$ then $s \leftarrow mix\text{-}cols(s)$ fi
$s \leftarrow s \oplus K_r$
end for
return s

- Fewer tables than DES
- Finite field operations

Security of AES

No key-recovery attack better than EKS is known, and latter is prohibitive for 128 bit keys.

KR - security

Adversary A knows $E\colon \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
$T \xleftarrow{\$} \{0,1\}^k$ is the target key.

Given: $(M_1, C_1), \ldots, (M_q, C_q)$ where $C_i = E(T, M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are distinct.

Find: T

So far, a block cipher has been viewed as secure if it resists key recovery, namely if there is no efficient way to solve the above problem.

Limitations of security against key recovery

Is security against key recovery enough?

Limitations of security against key recovery

Is security against key recovery enough?

Aliens from planet Crypton have a (new) cipher

$$A : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$$

that is guaranteed to resist key recovery. Would you use it encrypt?

Limitations of security against key recovery

Is security against key recovery enough?

Aliens from planet Crypton have a (new) cipher

$$\mathsf{A} : \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

that is guaranteed to resist key recovery. Would you use it encrypt?

The cipher is:

$$\mathsf{A}_{\mathsf{K}}(M) = M$$

- Impossible to find key from input-output examples, but
- Encryption is insecure because given ciphertext I know plaintext.

So what?

Possible reaction: But DES, AES are not designed like A, so why does this matter?

So what?

Possible reaction: But DES, AES are not designed like A, so why does this matter?

Answer: It tells us that security against key recovery is not, as a block-cipher property, sufficient for security of uses of the block cipher.

So what?

Possible reaction: But DES, AES are not designed like A, so why does this matter?

Answer: It tells us that security against key recovery is not, as a block-cipher property, sufficient for security of uses of the block cipher.

As designers and users we want to know what properties of a block cipher give us security when the block cipher is used.

So what is a "good" block cipher?

Possible Properties	Necessary?	Sufficient?
security against key recovery	YES	

So what is a "good" block cipher?

Possible Properties	Necessary?	Sufficient?
security against key recovery	YES	NO!

So what is a “good” block cipher?

Possible Properties	Necessary?	Sufficient?
security against key recovery	YES	NO!
hard to find M given $C = E_K(M)$	YES	

So what is a "good" block cipher?

Possible Properties	Necessary?	Sufficient?
security against key recovery	YES	NO!
hard to find M given $C = E_K(M)$	YES	NO!
$\vdots$		

We can't define or understand security well via some such (indeterminable) list.

We want a single "master" property of a block cipher that is sufficient to ensure security of common usages of the block cipher.