Foundations of Applied Cryptography

Adam O'Neill

Based on http://cseweb.ucsd.edu/~mihir/cse207/

Notation

$\{0,1\}^n$ is the set of n-bit strings and $\{0,1\}^*$ is the set of all strings of finite length. By ε we denote the empty string.

If S is a set then $|S|$ denotes its size. Example: $|\{0,1\}^2| = 4$.

If x is a string then $|x|$ denotes its length. Example: $|0100| = 4$.

If $m \geq 1$ is an integer then let $\mathbf{Z}_m = \{0, 1, \ldots, m-1\}$. $\mathbb{Z}_m$

By $x \xleftarrow{\$} S$ we denote picking an element at random from set S and assigning it to x. Thus $\Pr[x = s] = 1/|S|$ for every $s \in S$.

$x \xleftarrow{\$} \mathcal{D}$ non-uniform

Functions

Let $n \geq 1$ be an integer. Let $X_1, \ldots, X_n$ and Y be (non-empty) sets.

By $f\colon X_1 \times \cdots \times X_n \to Y$ we denote that f is a function that

- Takes inputs $x_1, \ldots, x_n$, where $x_i \in X_i$ for $1 \leq i \leq n$
- and returns an output $y = f(x_1, \ldots, x_n) \in Y$.

We call n the number of inputs (or arguments) of f. We call $X_1 \times \cdots \times X_n$ the domain of f and Y the range of f.

Example: Define $f\colon \mathbf{Z}_2 \times \mathbf{Z}_3 \to \mathbf{Z}_3$ by $f(x_1, x_2) = (x_1 + x_2) \bmod 3$. This is a function with $n = 2$ inputs, domain $\mathbf{Z}_2 \times \mathbf{Z}_3$ and range $\mathbf{Z}_3$.

Permutations

Suppose $f\colon X \to Y$ is a function with one argument. We say that it is a *permutation* if

- $X = Y$, meaning its domain and range are the same set.
- There is an *inverse* function $f^{-1}\colon Y \to X$ such that $f^{-1}(f(x)) = x$ for all $x \in X$.

This means f must be one-to-one and onto: for every $y \in Y$ there is a unique $x \in X$ such that $f(x) = y$.

$\{F_K\}_{K \in Keys}$ $\mathcal{E}_K(m)$

Function families

A family of functions (also called a function family) is a two-input function $F : \text{Keys} \times \text{D} \to \text{R}$. For $K \in \text{Keys}$ we let $F_K : \text{D} \to \text{R}$ be defined by $F_K(x) = F(K, x)$ for all $x \in \text{D}$.

- The set Keys is called the key space. If $\text{Keys} = \{0,1\}^k$ we call k the key length.
- The set D is called the input space. If $\text{D} = \{0,1\}^\ell$ we call ℓ the input length.
- The set R is called the output space or range. If $\text{R} = \{0,1\}^L$ we call L the output length.

Example: Define $F : \mathbf{Z}_2 \times \mathbf{Z}_3 \to \mathbf{Z}_3$ by $F(K, x) = (K \cdot x) \bmod 3$.

- This is a family of functions with domain $\mathbf{Z}_2 \times \mathbf{Z}_3$ and range $\mathbf{Z}_3$.
- If $K = 1$ then $F_K : \mathbf{Z}_3 \to \mathbf{Z}_3$ is given by $F_K(x) = x \bmod 3$.

What is a blockcipher?

Let $E\colon \mathrm{Keys} \times \mathrm{D} \to \mathrm{R}$ be a family of functions. We say that E is a block cipher if

- $\mathrm{R} = \mathrm{D}$, meaning the input and output spaces are the same set.
- $E_K : \mathrm{D} \to \mathrm{D}$ is a permutation for every key $K \in \mathrm{Keys}$, meaning has an inverse $E_K^{-1}\colon \mathrm{D} \to \mathrm{D}$ such that $E_K^{-1}(E_K(x)) = x$ for all $x \in \mathrm{D}$.

We let $E^{-1}\colon \mathrm{Keys} \times \mathrm{D} \to \mathrm{D}$, defined by $E^{-1}(K, y) = E_K^{-1}(y)$, be the inverse block cipher to E.

In practice we want that E, E^{-1} are efficiently computable.

If $\mathrm{Keys} = \{0,1\}^k$ then k is the key length as before. If $\mathrm{D} = \{0,1\}^\ell$ we call ℓ the block length.

Examples

$Keys = \{0,1\}^k$

$D = \{0,1\}^k$

$R = \{0,1\}^k$

$F_k(x) = K \oplus X$

$K \oplus X_1 \oplus K \oplus X_2 = X_1 \oplus X_2$ $\times$

$\underline{0^k \oplus K = K}$ c

Exercise

Above we had given the following example of a family of functions:
$F\colon \mathbf{Z}_2 \times \mathbf{Z}_3 \to \mathbf{Z}_3$ defined by $F(K, x) = (K \cdot x) \bmod 3$.

Question: Is F a block cipher? Why or why not?

Exercise

Let E: Keys $\times$ D $\rightarrow$ D be a block cipher. Is E a permutation?

- YES
- NO
- QUESTION DOESN'T MAKE SENSE
- WHO CARES?

Baby encryption scheme

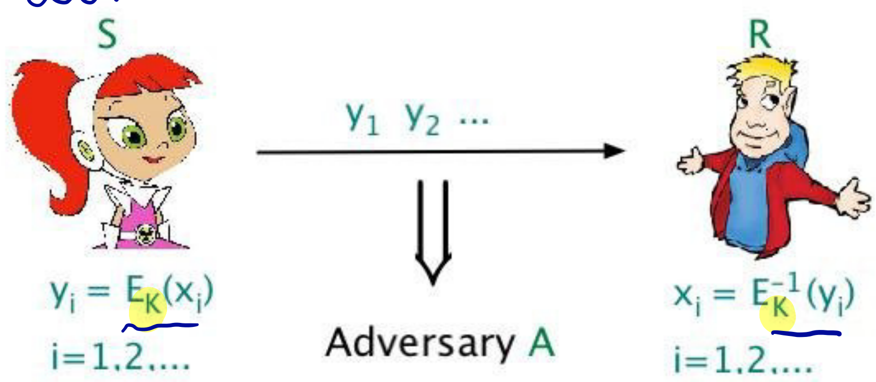
Kerkoff's principle

Blockcipher Usage

Let $E: \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ be a block cipher. It is considered public. In typical usage

- $K \xleftarrow{\$} \{0,1\}^k$ is known to parties S, R, but not given to adversary A.
- S, R use E_K for encryption

Alice Bob



Leads to security requirements like: Hard to get K from $y_1, y_2, \ldots$; Hard to get x_i from y_i; ...

Shannon’s Design Criterion (Informal)

Shannon’s Design Criterion (Informal)

- Confusion: Each bit of the output should depend on many bits of the input

Shannon's Design Criterion (Informal)

- Confusion: Each bit of the output should depend on many bits of the input
- Diffusion: Changing one bit of the input should "re-randomize" the entire output (avalanche effect)

Shannon's Design Criterion (Informal)

- Confusion: Each bit of the output should depend on many bits of the input
- Diffusion: Changing one bit of the input should "re-randomize" the entire output (avalanche effect)
- Not really solved (for many input-outputs) until much later: Data Encryption Standard (DES)

History of DES

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) in 2001.

DES Parameters

Key Length $k = 56$

Block length $\ell = 64$

So,

$$\mathsf{DES}\colon \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

$$\mathsf{DES}^{-1}\colon \{0,1\}^{56} \times \{0,1\}^{64} \to \{0,1\}^{64}$$

DES Construction

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

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

$\quad M \leftarrow IP(M)$

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

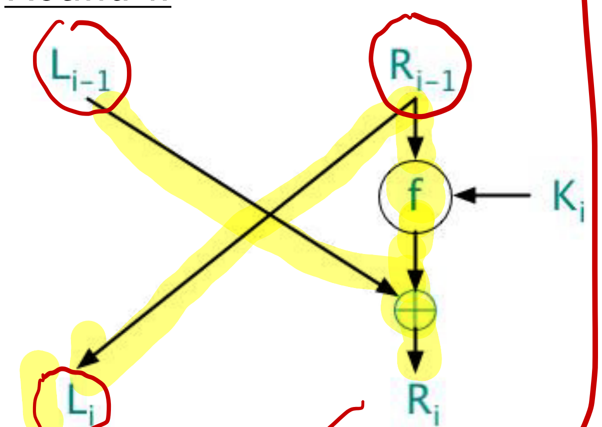
$\quad$ for $i = 1$ to 16 do

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

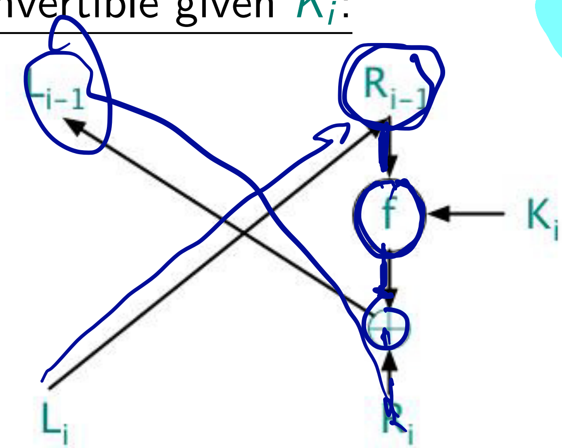
$\quad C \leftarrow IP^{-1}(L_{16} \| R_{16})$

$\quad$ return C

Round i:



Invertible given K_i:



Key-Recovery Attacks

Let E: Keys $\times$ D $\rightarrow$ R be a block cipher known to the adversary A.

- Sender Alice and receiver Bob share a *target key* $K \in$ Keys.
- Alice encrypts M_i to get $C_i = E_K(M_i)$ for $1 \leq i \leq q$, and transmits $C_1, \ldots, C_q$ to Bob
- The adversary gets $C_1, \ldots, C_q$ and also knows $M_1, \ldots, M_q$
- Now the adversary wants to figure out K so that it can decrypt any future ciphertext C to recover $M = E_K^{-1}(C)$.

Question: Why do we assume A knows $M_1, \ldots, M_q$?

Answer: Reasons include a posteriori revelation of data, a priori knowledge of context, and just being conservative!

Security Metrics

We consider two measures (metrics) for how well the adversary does at this key recovery task:

- Target key recovery (TKR)
- Consistent key recovery (KR)

In each case the definition involves a game and an advantage.

The definitions will allow E to be any family of functions, not just a block cipher.

The definitions allow A to pick, not just know, $M_1, \ldots, M_q$. This is called a chosen-plaintext attack.

$Keys = \{1,2\}$ $D = \{1,2\}$ $R = \{1,2\}$ $F_K(x) = x$

Consistent Keys

$(1,1)$ $(2,2)$

Def: Let E: Keys $\times$ D $\rightarrow$ R be a family of functions. We say that key $K' \in$ Keys is *consistent* with $(M_1, C_1), \ldots, (M_q, C_q)$ if $E(K', M_i) = C_i$ for all $1 \leq i \leq q$.

Example: For E: $\{0,1\}^2 \times \{0,1\}^2 \rightarrow \{0,1\}^2$ defined by

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

The entry in row K, column M is $E(K, M)$.

- Key 00 is consistent with $(11, 01)$
- Key 10 is consistent with $(11, 01)$
- Key 00 is consistent with $(01, 00), (11, 01)$
- Key 11 is consistent with $(01, 00), (11, 01)$

Consistent Key Recovery

Let E: Keys $\times$ D $\rightarrow$ R be a family of functions, and A an adversary.

Game KR_E

procedure Initialize
$K \xleftarrow{\$}$ Keys; $i \leftarrow 0$

procedure Fn(M)
$i \leftarrow i + 1$; $\boxed{M_i \leftarrow M}$
$C_i \leftarrow E(K, M_i)$
Return C_i

procedure Finalize(K')
$\mathrm{win} \leftarrow$ true
For $j = 1, \ldots, i$ do
  If $E(K', M_j) \neq C_j$ then $\mathrm{win} \leftarrow$ false
  If $M_j \in \{M_1, \ldots, M_{j-1}\}$ then $\mathrm{win} \leftarrow$ false
Return win

$K' \xleftarrow{\$} A$

Definition: $\mathbf{Adv}_E^{\mathrm{kr}}(A) = \Pr[\mathrm{KR}_E^A \Rightarrow \text{true}]$.

The game returns true if (1) The key K' returned by the adversary is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$, and (2) $M_1, \ldots, M_q$ are distinct.

A is a q-query adversary if it makes q distinct queries to its **Fn** oracle.

Target Key Recovery Game

Game TKR_E	**procedure Fn**(M)
procedure Initialize	Return $E(K, M)$
$K \xleftarrow{\$}$ Keys	**procedure Finalize**(K')
	Return $(K = K')$

Definition: $\mathbf{Adv}_E^{\mathrm{tkr}}(A) = \Pr[\mathrm{TKR}_E^A \Rightarrow \text{true}]$.

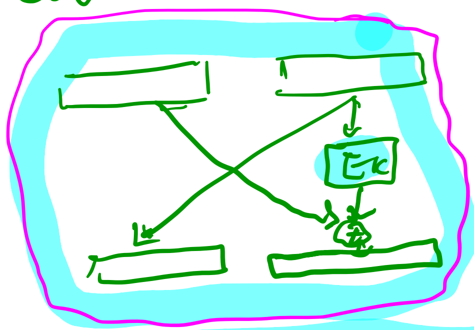
- First **Initialize** executes, selecting *target key* $K \xleftarrow{\$}$ Keys, but not giving it to A.
- Now A can call (query) **Fn** on any input $M \in$ D of its choice to get back $C = E_K(M)$. It can make as many queries as it wants.
- Eventually A will halt with an output K' which is automatically viewed as the input to **Finalize**
- The game returns whatever **Finalize** returns
- The tkr advantage of A is the probability that the game returns true

Exercise: KR of Feistel

Reductions

blockcipher

Suppose WTS if E is TKR-secure then Feistel[E] is TKR-secure



Proof. Assume there is an efficient TKR adversary A with high TKR-advantage against Feistel[E]. Then ∃ efficient TKR-adversary B with high advantage against E.

Algorithm $B^{F_n(\cdot)}$

Run A

When A makes F_n query x do: {

// want to give $E_k(x)$

$y_1 \| y_2 \leftarrow F_n(0^\ell \| x)$

ret y_2 to A

Until A outputs '?'

ret k?

TKR Reductions

Feistel

A relation

Fact: Suppose that, in game KR_E, adversary A makes queries $M_1, \ldots,$ M_q to **Fn**, thereby defining $C_1, \ldots, C_q$. Then the target key K is consistent with $(M_1, C_1), \ldots, (M_q, C_q)$.

Proposition: Let E be a family of functions. Let A be *any* adversary all of whose **Fn** queries are distinct. Then

$$\mathbf{Adv}_E^{\mathrm{kr}}(A) \geq \mathbf{Adv}_E^{\mathrm{tkr}}(A) \; .$$

Why? If the K' that A returns equals the target key K, then, by the Fact, the input-output examples $(M_1, C_1), \ldots, (M_q, C_q)$ will of course be consistent with K'.

generic

Exhaustive Key Search

Let E: Keys $\times$ D $\rightarrow$ R be a function family with Keys $= \{T_1, \ldots, T_N\}$ and D $= \{x_1, \ldots, x_d\}$. Let $1 \leq q \leq d$ be a parameter.

q-query

EKS adversary

adversary A^{q}_{eks}

For $j = 1, \ldots, q$ do $M_j \leftarrow x_j$; $C_j \leftarrow \mathbf{Fn}(M_j)$

For $i = 1, \ldots, N$ do

  if $(\forall j \in \{1, \ldots, q\} : E(T_i, M_j) = C_j)$ then return T_i

Question: What is $\mathbf{Adv}^{\mathrm{kr}}_{E}(A^{q}_{\text{eks}})$? $\simeq 1$.

$E_{\boxed{k}}(x) = x$

Exhaustive Key Search

Let E: Keys $\times$ D $\to$ R be a function family with Keys $= \{T_1, \ldots, T_N\}$ and D $= \{x_1, \ldots, x_d\}$. Let $1 \leq q \leq d$ be a parameter.

$\underline{E_k(\langle i \rangle) = \langle i \rangle}$ $\quad \forall i \in \{1, \ldots, q\}$

adversary A_{eks}

For $j = 1, \ldots, q$ do $M_j \leftarrow x_j$; $C_j \leftarrow \textbf{Fn}(M_j)$
For $i = 1, \ldots, N$ do
    if $(\forall j \in \{1, \ldots, q\} \ : \ E(T_i, M_j) = C_j)$ then return T_i

Question: What is $\underline{\textbf{Adv}_E^{\text{tkr}}(A_{\text{eks}})}$?

Exhaustive Key Search

Let E: Keys $\times$ D $\rightarrow$ R be a function family with Keys $= \{T_1, \ldots, T_N\}$ and D $= \{x_1, \ldots, x_d\}$. Let $1 \leq q \leq d$ be a parameter.

adversary A_{eks}

For $j = 1, \ldots, q$ do $M_j \leftarrow x_j$; $C_j \leftarrow \mathbf{Fn}(M_j)$
For $i = 1, \ldots, N$ do
    if $(\forall j \in \{1, \ldots, q\} \,:\, E(T_i, M_j) = C_j)$ then return T_i

Question: What is $\mathbf{Adv}_E^{\mathrm{tkr}}(A_{\mathrm{eks}})$?

Answer: Hard to say! Say $K = T_m$ but there is a $i < m$ such that $E(T_i, M_j) = C_j$ for $1 \leq j \leq q$. Then T_i, rather than K, is returned.

In practice if E: $\{0,1\}^k \times \{0,1\}^\ell \rightarrow \{0,1\}^\ell$ is a "real" block cipher and $\boxed{q > k/\ell}$, we expect that $\mathbf{Adv}_E^{\mathrm{tkr}}(A_{\mathrm{eks}})$ is close to 1 because K is likely the only key consistent with the input-output examples.

q=1 56/64

generic.

Exhaustive Key-Search on DES

$\boxed{2^{56}}$

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 A_{eks} $(q = 1)$ 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 & Linear cryptanalysis

non-generic

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

The following non-generic key-recovery attacks on DES have advantage close to one and running time smaller than 2^{56} DES computations:

Attack	when	q, running time
Differential cryptanalysis	1992	2^{47}
Linear cryptanalysis	1993	2^{44}

An observation

Observation: The E computations can be performed in parallel!

In 1993, Wiener designed a dedicated DES-cracking machine:

- $1 million
- 57 chips, each with many, many DES processors
- Finds key in 3.5 hours

Increasing Key-Length

Can one use DES to design a new blockcipher with longer effective key-length?

$a \| b$

E:

$k_1 \| k_2$

Define:

$DES'_{k_1 \| k_2}(x) = DES_{k_1}(x) \| DES_{k_2}(x)$

$k = 112$

(x_1, y_1)

$x_1 \to y_1$

$2 \cdot 2^{56} = 2^{57}$

vs

2^{112} EKS.

→ Adversary $A^{Fn(\cdot)}$

Let $x_1 = x_{11} \| x_{12}$ be arbitrary

$y_1 \leftarrow Fn(x_1)$; parse as $y_{11} \| y_{12}$

Let $K_1, \ldots, K_{2^{56}}$

be an enumeration of

DES keys

For $i = 1$ to 2^{56} do: {

if $y_{11} = DES_{K_i}(x_{11})$ {

$K_1^* \leftarrow K_i$; break }

}

For $i = 1$ to 2^{56} do: {

If $y_{12} = DES_{K_i}(x_{12})$

$K_2^* \leftarrow K_i$; break

}

ret $K_1^* \| K_2^*$

2DES

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

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

2DES

Block cipher $2DES : \{0,1\}^{112} \times \{0,1\}^{64} \to \{0,1\}^{64}$ 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

Suppose $K_1 K_2$ is a target 2DES key and adversary has M, C such that

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

Then

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

Meet-in-the-Middle Attack

io example

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

$K_1 \rightarrow$

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

Table L

$\xrightarrow{\text{equal}}$

$DES^{-1}(T_1, C)$	T_1
$DES^{-1}(T_j, C)$	T_j
	T_2^*
$DES^{-1}(T_N, C)$	T_N

Table R

$\leftarrow K_2$

Attack idea:

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

$$\boxed{T_1^* \,\|\, T_2^*}$$

Translating to Pseudocode

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

adversary A_{MinM}

$M_1 \leftarrow 0^{64}$; $C_1 \leftarrow \mathbf{Fn}(M_1)$
for $i = 1, \ldots, 2^{56}$ do $L[i] \leftarrow \mathsf{DES}(T_i, M_1)$
for $j = 1, \ldots, 2^{56}$ do $R[j] \leftarrow \mathsf{DES}^{-1}(T_j, C_1)$
$S \leftarrow \{\, (i, j) \;:\; L[i] = R[j] \,\}$
Pick some $(l, r) \in S$ and return $T_l \,\|\, T_r$

Attack takes about 2^{57} DES/DES^{-1} computations and has $\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{2DES}}(A_{\mathrm{MinM}}) = 1$.

This uses $q = 1$ and is unlikely to return the target key. For that one should extend the attack to a larger value of q.

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

$$3DES3_{K_1 \parallel K_2 \parallel K_3}(M) \;=\; \mathrm{DES}_{K_3}(\mathrm{DES}^{-1}_{K_2}(\mathrm{DES}_{K_1}(M)))$$

$$3DES2_{K_1 \parallel K_2}(M) \;=\; \mathrm{DES}_{K_2}(\mathrm{DES}^{-1}_{K_1}(\mathrm{DES}_{K_2}(M)))$$

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

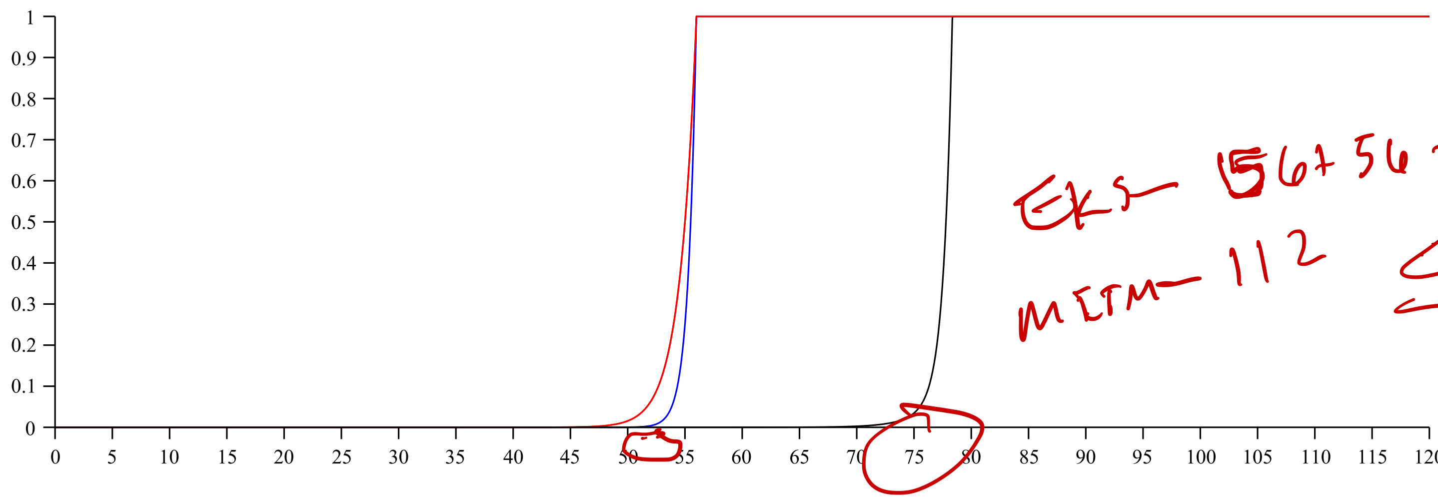
still OK

Can one do better?

underlying blockcipher is a random permutation on every key.

3DES Security

"ideal cipher model"



EKS - 56+56+56

MITM - 112

$\leq$

Figure 1: Upper bound on adversarial advantage (proven security) verses $\log_2 q$ (where q=number of queries) for the cascade construction, assuming key length $k = 56$ and block length $n = 64$. Single encryption is the leftmost curve, double encryption is the middle curve [3], and triple encryption in the rightmost curve, as given by Theorem 4.

bellare and rogaway

Code-Based Game-Playing Proofs and the Security of Triple Encryption

Mihir Bellare [*]   Phillip Rogaway [†]

November 27, 2008

(Draft 3.0)

Abstract

The game-playing technique is a powerful tool for analyzing cryptographic constructions. We illustrate this by using games as the central tool for proving security of three-key triple-encryption, a long-standing open problem. Our result, which is in the ideal-cipher model, demonstrates that for DES parameters (56-bit keys and 64-bit plaintexts) an adversary's maximal advantage is small until it asks about 2^{78} queries. Beyond this application, we develop the foundations for game playing, formalizing a general framework for game-playing proofs and discussing techniques used within such proofs. To further exercise the game-playing framework we show how to use games to get simple proofs for the PRP/PRF Switching Lemma, the security of the basic CBC MAC, and the chosen-plaintext-attack security of OAEP.

Keywords: Cryptographic analysis techniques, games, provable security, triple encryption.

game-playing

THE FUNDAMENTAL LEMMA. The fundamental lemma says that the advantage that an adversary can obtain in distinguishing a pair of identical-until-*bad* games is at most the probability that its execution sets *bad* in one of the games (either game will do).

Lemma 2 [Fundamental lemma of game-playing] *Let G and H be identical-until-bad games and let A be an adversary. Then*

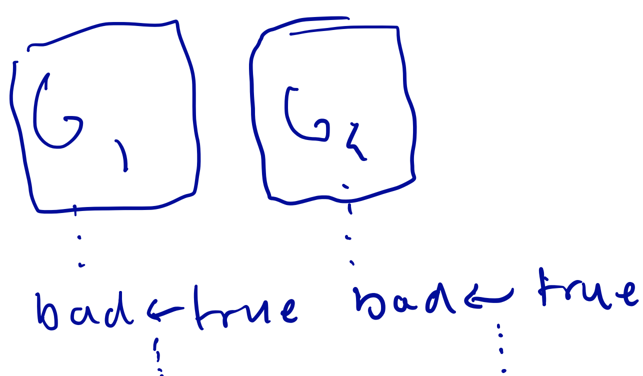
$$\mathbf{Adv}(A^G, A^H) \;\leq\; \Pr[A^G \text{ sets } bad\,] \quad \textit{and} \tag{6}$$

$$\mathbf{Adv}(G^A, H^A) \;\leq\; \Pr[G^A \text{ sets } bad\,]\,. \tag{7}$$

More generally, let G, H, I be identical-until-bad games. Then

$$\left|\mathbf{Adv}(A^G, A^H)\right| \;\leq\; \Pr[A^I \text{ sets } bad\,] \quad \textit{and} \tag{8}$$

$$\left|\mathbf{Adv}(G^A, H^A)\right| \;\leq\; \Pr[I^A \text{ sets } bad\,]\,. \tag{9}$$



G

$\boxed{A^{\pi}}$ or $\boxed{A^{\rho}}$

random func: from Func(n)

random perm

2 The PRP/PRF Switching Lemma

The lemma. The natural and conventional assumption to make about a blockcipher is that it behaves as a pseudorandom permutation (PRP). However, it usually turns out to be easier to analyze the security of a blockcipher-based construction assuming the blockcipher is secure as a pseudorandom function (PRF). The gap is then bridged (meaning, a result about the security of the construct assuming the blockcipher is a PRP is obtained) using the following lemma. In what follows, we denote by $A^P \Rightarrow 1$ the event that adversary A, equipped with an oracle P, outputs the bit 1. Let Perm(n) be the set of all permutations on $\{0,1\}^n$ and let Func(n) be the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. We assume below that π is randomly sampled from Perm(n) and ρ is randomly sampled from Func(n).

Lemma 1 [PRP/PRF Switching Lemma] *Let $n \geq 1$ be an integer. Let A be an adversary that asks at most q oracle queries. Then*

$$\left|\Pr\left[\, A^{\pi} \Rightarrow 1 \,\right] - \Pr\left[\, A^{\rho} \Rightarrow 1 \,\right]\right| \leq \frac{q(q-1)}{2^{n+1}} . \quad \blacksquare$$

q: # of queries

Proof of Lemma

Consider the following two games:

game G_0 / game $\boxed{G_1}$

assume adv doesn't make same query twice:

When A makes query x do:

$T[x] \xleftarrow{\$} \{0,1\}^n$

If $T[x] = T[x']$ for some x'

BAD $\leftarrow$ true

$\boxed{T[x] \xleftarrow{\$} \overline{T}}$

return $T[x]$

all elements of $\{0,1\}^n$ not in the table

Correcting line.

$$\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1] \le \Pr[G_1 \text{ sets BAD}]$$

Let $COLL_i$ be event s.t. there is a collision on i-th query.

By union bound

$$\Pr[\text{BAD is set}] \leq \sum_{i=1}^{q} \Pr[COLL_i].$$

$$\Pr[COLL_i] \leq \frac{i-1}{2^n}$$

$$\Rightarrow \Pr[\text{BAD set}] \leq \sum_{i=q}^{q} \frac{i-1}{2^n} = \frac{q(q-1)}{2^{n+1}} \quad \square$$

sum of first n integer formula

Increasing Block-Length?

We will later see that we would also like a blockcipher with longer block-length.

Increasing Block-Length?

We will later see that we would also like a blockcipher with longer block-length.

Increasing Block-Length?

We will later see that we would also like a blockcipher with longer block-length.

This seems much harder to do using DES.

Increasing Block-Length?

We will later see that we would also like a blockcipher with longer block-length.

This seems much harder to do using DES.

Increasing Block-Length?

We will later see that we would also like a blockcipher with longer block-length.

This seems much harder to do using DES.

Motivated the search for a new blockcipher.

advanced encryption standard

AES History

1998: NIST announces competition for a new block cipher

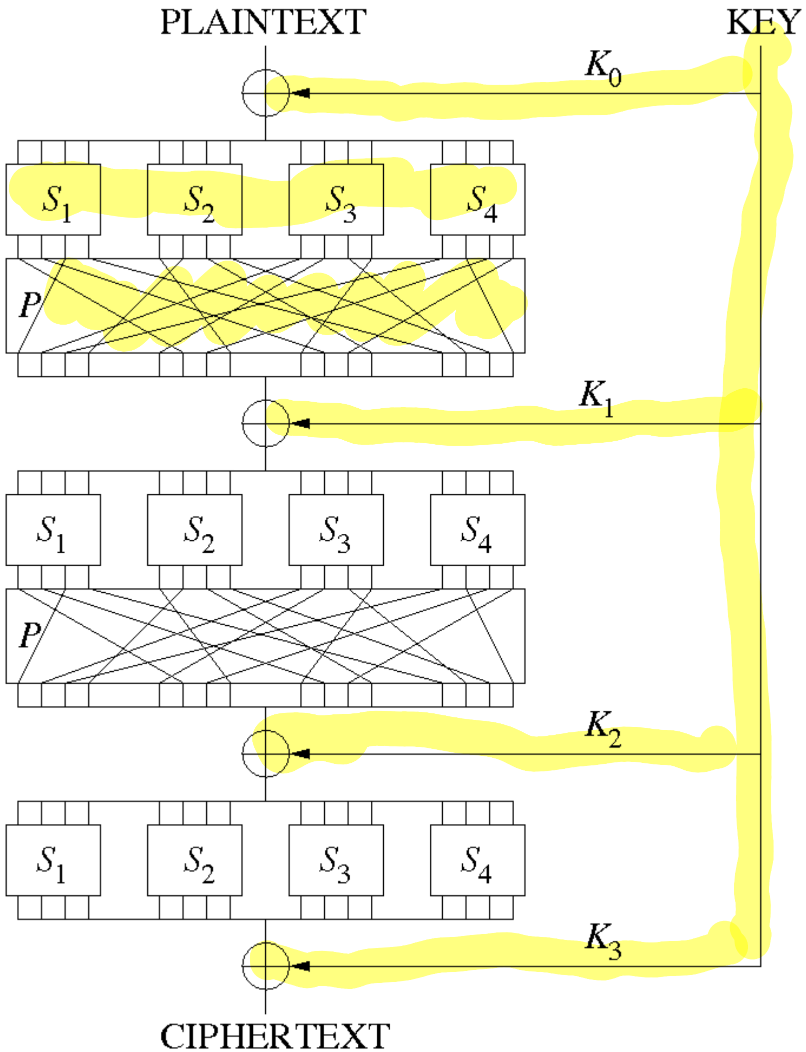
- key length 128
- block length 128
- faster than DES in software

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 Construction

don't need to know how DES works



substitution-permutation network

generic vs non-generic

AES Security

Best known key-recovery attack [BoKhRe11] takes $2^{126.1}$ time, which is only marginally better than the 2^{128} time of EKS.

There are attacks on reduced-round versions of AES as well as on its sibling algorithms AES192, AES256. Many of these are "related-key" attacks. There are also effective side-channel attacks on AES such as "cache-timing" attacks [Be05,OsShTr05].

Limitations of Key Recovery

- malleability
- Stephen's attack

Ex. ~~One-time~~ pad (can recover key)

Ex. Identity block-cipher

$$E_k(x) = x$$

(x_1, x_1)

$\vdots$

(x_n, x_n)

So What?

Possible reaction: But DES, AES are not designed like E above, 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.

Killer Application: Pseudo One-time Pad

- pseudo random generator (PRG)

$G : \{0,1\}^n \rightarrow \{0,1\}^*$

want $G(s) \boxed{\approx_c} R_{|G(s)|}$

where $R_{|x|}$ is random on $\{0,1\}^{|x|}$

∀ efficient D

$\Pr[D(G(s)) \Rightarrow 1] - \Pr[D(R_{|G(s)|}) \Rightarrow 1]$ is small

* can compress key for OTP using PRG.

$$PRG(K) := E_K(\langle 1 \rangle) \| \ldots \| E_K(\langle n \rangle)$$

want to justify this usage.