AES & Key Management

Deepak Puthal

Email: Deepak.Puthal@uts.edu.au

41900 – Fundamentals of Security

Overview

- **DES**
  - DES Keys
  - 2DES
  - 3DES
  - DESX
- **AES**
  - AES Overview
  - AES in more detail
- **Key Distribution**
  - Key Distribution Centre
  - Merkle's Puzzles
  - Dife Hellman

DES Keys

Given one plaintext/ciphertext pair (m, c), there is a high probability that only one key will satisfy:

$$c = DES(m, k)$$

Consider DES as a collection of permutations: $\pi(1) \ldots \pi(2^{56})$

If π_i are independent permutations then $\forall(m, k)$:

$$Pr[k_1 \neq k : DES(m, k_1) = DES(m, k)]$$
$$= 256 \times 2^{-64}$$
$$= 2^{-56}$$
$$= 1.39 \times 10^{-17}$$
$$= 0.0000000000000139\%$$

Thus, given one (m, c) pair, the key is (almost definitely) uniquely determined.

The problem is to find k

Attacks on DES

Exhaustive Key Search

- Strong **n**-bit block cipher, **j**-bit key, the key can be recovered on average in 2^{j-1} operations, given a small number (< (j + 4)/n) of plaintext/ciphertext pairs
- For **DES**, **j = 56**, **n = 64** so exhaustive key search is expected to yield the key in 2^{55} operations.

Attacks on DES

Ciphertext-Only DES key search

- Example: DES is used to encrypt 8×8 ASCII characters (= 64 bits) per block - one bit is a *parity* bit.
- Let's say we try decrypting - this will yeild all 8 correct parity bits with probability 2^{-8} (≈ 0.4%)
- Thus with **t** blocks, we can safely say that it would have probability of 2^{-8t}
- So, using this - 2^{56} keys - probability of a correct key with all valid parity bits = $(1 - 2^{8t})$
- Therefore, **t** ~5-10 blocks are enough for > 99.99999% sure.

2DES

Double Encryption with DES (*2DES*)

2DES **IS BAD!**

$$2DES_{k_1,k_2}(m) = E_{k_1}(E_{k_2}(m))$$

Vulnerable to the **meet-in-the-middle** attack with known plaintext.

Example:
for a fixed message, **m**, create a table of all possible ciphertext with each 56-bit encryption keys:

$E_k(m)$ for all $k \in \{0, 1\}^{56}$

Then, for c = $E_{k_1 k_2}(m)$, try to decrypt:

$D_k(c)$ for all $k \in \{0, 1\}^{56}$

Until $D_k(c)$ appears in the table, since $D_{k_1}(c) = E_{k_2}(m)$

2DES

What does this mean?

2DES can be broken in 2^{56} operations on average, using 2^{56} memory slots. (A time-space trade-off!).
This is not good when there should be 112-bits (56 + 56) of key.

3DES

Two-key Triple DES (3DES) - DES 3 times, 2 keys. (*112 bits*)

$$3DES_{k1,k2}(m) = E_{k1}(D_{k2}(E_{k1}(m)))$$

The strength of DES/3DES is that it does not form a group!

$$DES_{k1}(DES_{k2}(m)) \neq DES_{k3}(m)$$

3DES

Let's consider that *time-space trade-off* in 2DES

For time $\frac{2^{(56+64)}}{s}$ and space **s**, we can recover k_1 and k_2 in 2DES.

If s > 28 - we can do better than exhaustive search.

If you have three distinct keys - then it has 168 key bits.

(The effective key length = 112 bits because of "*meet-in-the-middle*")

If you use two keys ($k_1 = k_3$, k_2) then it has 112 key bits.

(The effective key length = 80 bits due to chosen/known plaintext attacks)

DESX

A modification of DES to avoid exhaustive key search is **DESX**.

k1 = 56bits (DES Key)

k2 = 64bits (Whitening Key)

k3 = h(k_2, k_3)

= 64bits

$$DESX_{k_1,k_2,k_3}(m) = k_3 \oplus E_{k_1}(m \oplus k_2)$$

The **whitening key** gives greater resilience to brute force attacks.

DESX

Given **j** plaintext / ciphertext pairs, the effective key size is greater or equal to:

$$|k| + n - 1 - \log(j) = 56 + 64 - 1 - \log(j)$$
$$= 119 - \log(j)$$
$$\geqslant 100\text{bits}$$

Replacing DES

US Government wanted DES used as a "**standard**"

RSA Security wanted to demonstrate that DES *sucked*.. it was weak because of the key length.

Timeline:

1997	First DES challenge solved in 96 days using distributed computing (idle CPU) Second DES challenge solved in 41 days using distributed.net (idle CPU)
1998	EFF created Deep Crack for $250K - decrypted in 56 hours
1999	Deep Crack + Distributed.net decrypted DES in 22h 15min

EFFs DES Cracker

Whitfeld Dife and **Martin Hellman** estimated that a machine fast enough to test that many keys in a day would cost about $20 million in 1976. (Minimal cost for NSA or governments...)

Composed of 1856 custom **ASIC DES**

chips, 90 billion (≈ 236) keys per second!

Entire key space in **9 days**!

(On average, key found in half that time!)

2006	COPACOBANA ($10k) recover DES key in ≈ 6.4 days
2008	Reduced to less than one day using 128 off the shelf FPGAs

AES

Advanced Encryption Standard (AES)

In 1997 NIST announced that a competition would be held to choose a new cipher to replace the outdated DES cipher, this to be was named the Advanced Encryption Standard – AES.

Of the contenders, they chose Rijndael as the new AES.

- Block cipher
- 128 bit blocks
- 128/192/256 bit keys
- Criteria:
  - Strength $\geqslant$ 3DES, but much better effiency
  - Flexible - can be implemented in software, hardware or smartcards
  - Simple and Elegant
- Royalty-free worldwide
- Security for over 30 years
- May protect sensitive data for over 100 years
- Public confidence in the cipher

AES Candidates

15 submissions from the international feld.
A number of strong finalists:

Name	Type	Rounds	Rel. Speed (cycles)	Gates
Twofsh	Feistel	16	1254	23k
Serpent	SP-network	32	1800	70k
Mars	Type-3 Feistel	32	1600	70k
Rijndael	SP-network	10, 12, 14	1276	-
RC6	Feistel	20	1436	-

AES

Rijndael (pronounced [rɛinda:l] “rain-dahl”) announced October 2000

- Operates on 128 bit blocks
- Key length is variable: 128, 192 or 256 bits
- It is an SP-network (substitution-permutation network)
- Uses a single S-box which acts on a byte input to give a byte output (a 256 byte lookup table):

$$S(x) = M(x^{-1}) + b \text{ over } GF(2^8)$$

Where M is a predefned matrix, b is a constant and GF is chosen Galois Feld (nonlinearity comes from $x \rightarrow x^{-1}$).

- Construction gives tight differential and linear bounds

AES Overview - Rounds

The number of rounds are variable:

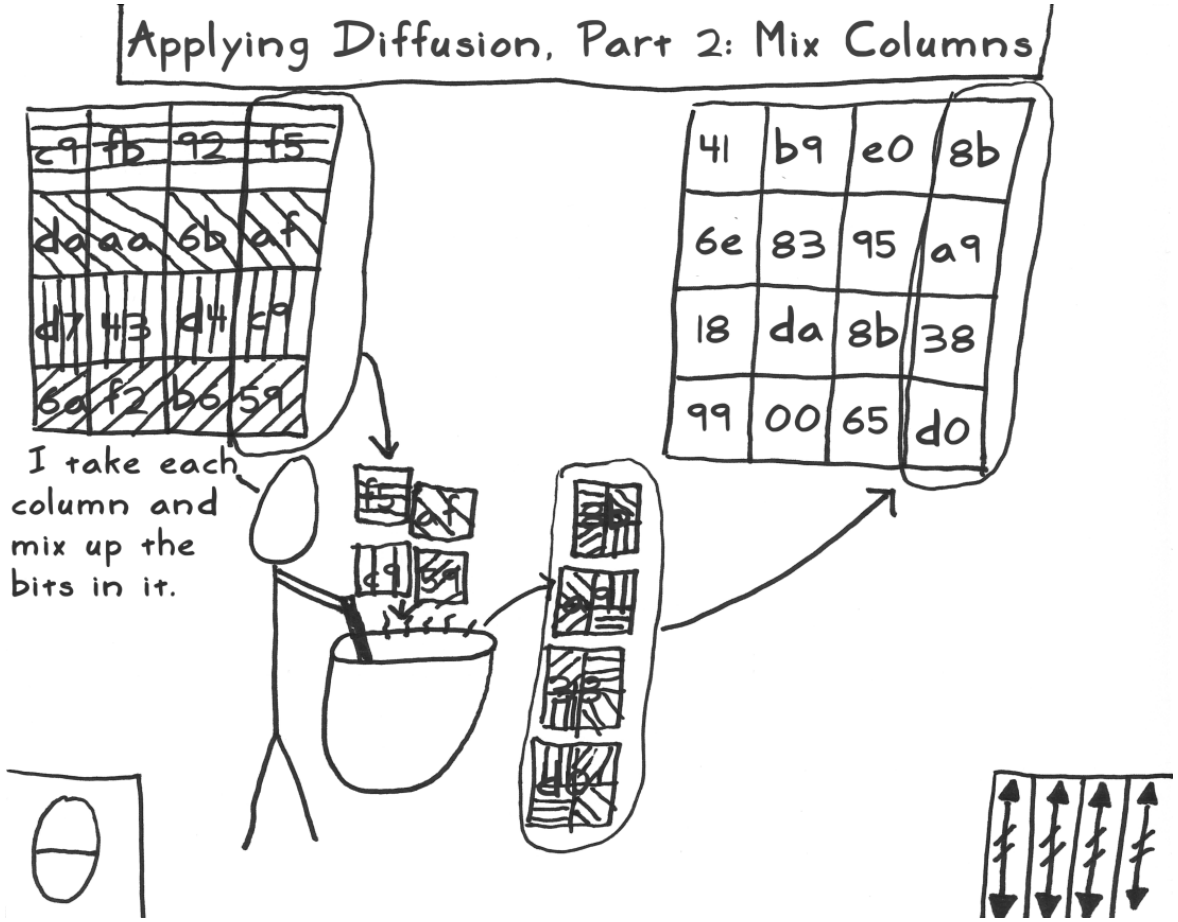
- 10 rounds – 128 bit keys
- 12 rounds – 192 bit keys
- 14 rounds – 256 bit keys

Rounds have a 50% margin of safety based on current known attacks. Potential attacks (which require an *enormous* number of plaintext/ciphertext pairs) are possible on:

- Only 6 rounds for 128 bit keys
- Only 7 rounds for 192 bit keys
- Only 9 rounds for 256 bit keys

Safety against possible attacks believed to currently be ≈ 100%

A Stick Figure Guide to AES



This is mandatory reading for the course.

http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Key Distribution

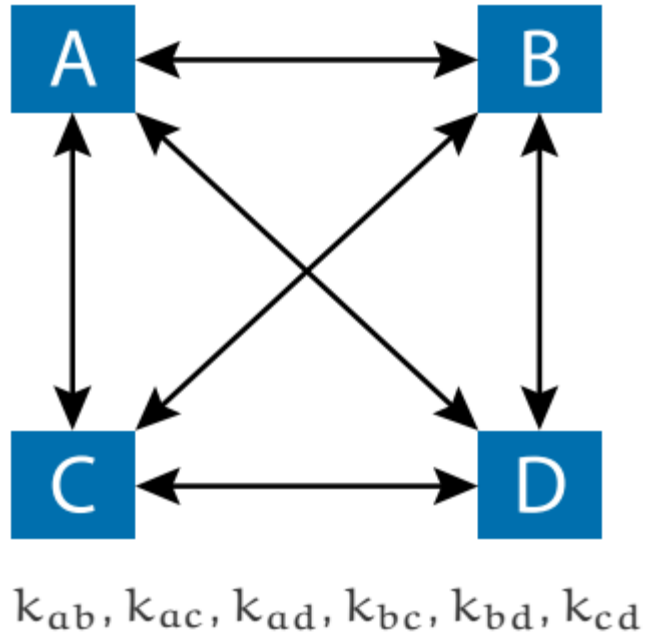
Key Management

Suppose we have a symmetric key network where Alice, Bob, Carol and Dave want to talk to each other.

For secure communication with n parties, we require:

$$\binom{n}{n-1} = \frac{n(n-1)}{k}\ keys$$

Key distribution and management becomes a major issue!



Definitions

Key Establishment is the process whereby a shared key becomes available to two or more parties for subsequent cryptographic use.

Key Management is the set of processes and mechanisms which support key establishment and the maintenance of on going key relationships between parties, including replacing older keys with newer ones. Includes:

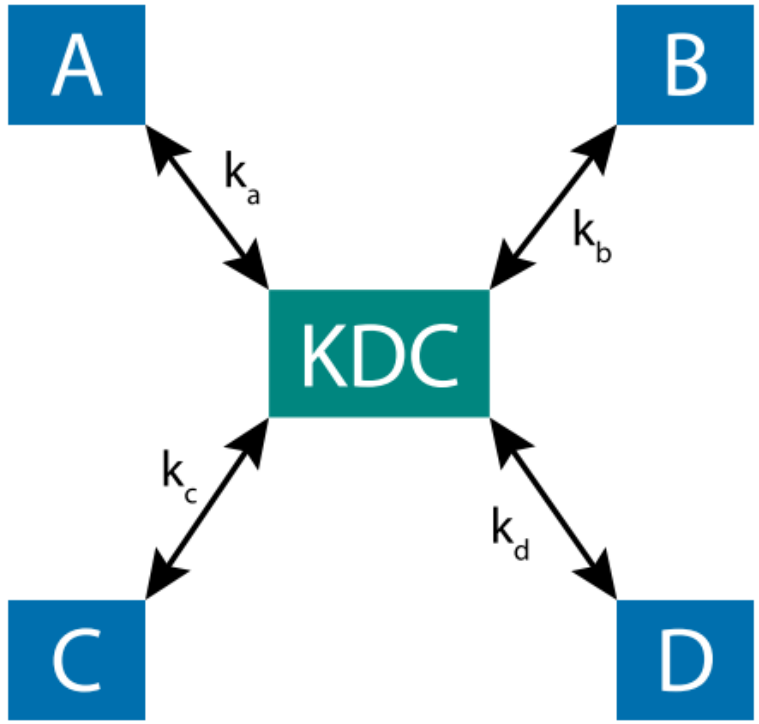
- Key agreement
- Key transport

Key Distribution Centre: Naïve

Protocol:

1. Alice →KDC
   I want to talk to Bob
2. KDC →Alice
   - KDC chooses random k_{ab}
   - Returns:

   $E_{ka}(k_{ab})$, $E_{kb}(k_{ab}$, “for talking to Alice”)
3. Alice decrypts $E_{ka}(k_{ab})$ to get k_{ab}
4. Alice →Bob
   $E_{kb}(k_{ab}$, “for talking to Alice")
5. Bob decrypts using k_b to get k_{ab}
6. Alice & Bob now share k_{ab}

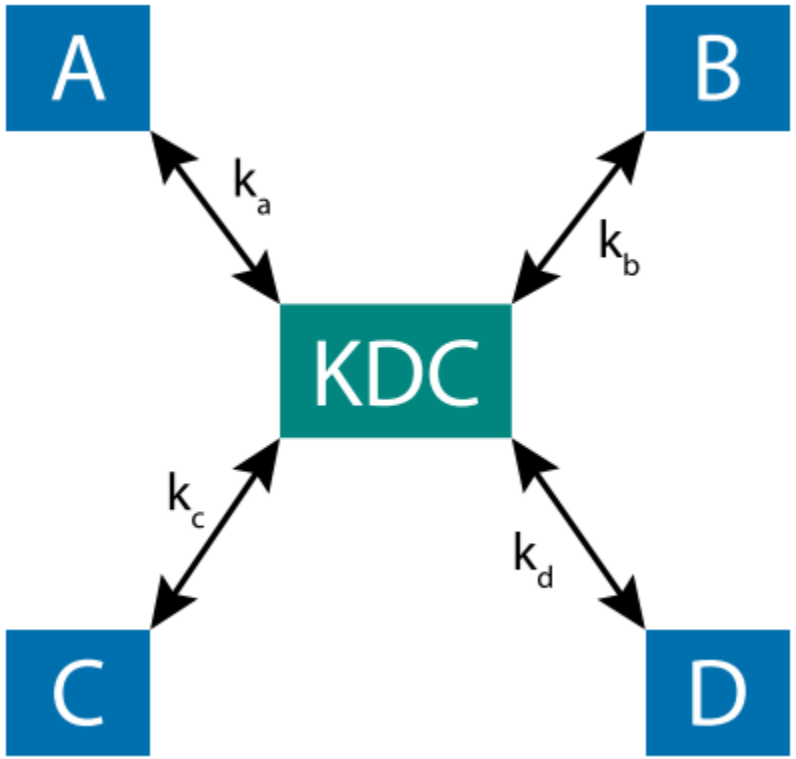


A Key Distribution Centre

Key Distribution Centre: Naïve

Problems:

- The Key Distribution Centre is a single point of failure – *likely to be attacked*
- No authentication
- Poor scalability
- Slow



A Key Distribution Centre

Merkle's Puzzles

Merkle's Puzzles are a way of doing key exchange between Alice and Bob without the need for a third party.

- Alice creates N puzzles $P_1, P_2, \ldots, P_N$, of the form

$$P_i = E_{pi}(\text{“This is puzzle \#}X_i\text{”}, k_i)$$

  - $N \approx 200$
  - $| P_i | \approx 20$ bits (weak)
  - $| K_i | \approx 128$ bits (strong)
  - X_i, p_i, and k_i are chosen randomly and *different* for each *i*.
- Alice sends all puzzles to Bob: $P_1, P_2, \ldots, P_N$.
- Bob chooses a random puzzle P_j for some $j \in \{1, 2, \ldots, N\}$.
  - Finds p_j by brute force (key space search)
  - Recovers k_j and X_j
  - Bob sends X_j to Alice unencrypted
- Alice looks up the index of X_j to fnd they key k_j chosen by Bob.
- Alice & Bob both share key k_j

Attacking Merkle's Puzzles

On average, Eve must break half of the puzzles to find which puzzle contains X_j (and hence obtain k_j).

So for 2^{20} puzzles, Eve must try 2^{19} puzzles on average.

Each puzzle is encrypted with the 20 bit key pi. Eve must search, on average, half of the key space: 2^{19}. $2^{19} \times 2^{19} = 2^{38}$

If Alice and Bob can try 10,000 keys per second:

- It will take about 1 minute for each to perform their steps

  Alice to generate, and Bob to break $p_j = 2^{19}$ keys

- Plus another minute to communicate all the puzzles over ADSL

With comparable resources, it will take Eve about a year to break the system.

Note: Merkle's puzzles uses a lot of bandwidth – impractical!

Diffie-Hellman Key Exchange

Diffie-Hellman key exchange (Stanford, 1976) is a protocol for establishing a cryptographic key using mathematical tricks. It is a worldwide standard for use in SSL, smartcards, etc.

The *rough idea* is this: (details later)

- Alice and Bob agree on some number g.
- Alice generates a random number a, and sends g^a to Bob.
- Bob generates a random number b, and sends g^b to Alice.
- Alice and Bob can each compute g^{ab}, their shared secret.

An eavesdropper only has g^a, g^b, and g. *Assuming that taking logarithms is hard*, they cannot recover a or b.

Next lecture: the maths behind making logarithms hard.

Diffie-Hellman Key Exchange