Why not 2-DES?

- 2DES: \( C = \text{DES} ( K_1, \text{DES} ( K_2, P ) ) \)
- Seems to be hard to break by "brute force", approx. \( 2^{111} \) trials
- Assume Eve is trying to break 2DES and has a single (P,C) pair

**Meet-in-the-middle (or Rendevouz) ATTACK:**

I. For each possible \( K'_i \) (where \( 0 < i < 2^{56} \))
   1. Compute \( C'_i = \text{DES} ( K'_i, P ) \)
   2. Store: \([ K'_i, C'_i ]\) in table \( T \) (sorted by \( C'_i \))

II. For each possible \( K''_i \) (where \( 0 < i < 2^{56} \))
   1. Compute \( C''_i = \text{DES}^{-1} ( K''_i, C ) \)
   2. Lookup \( C''_i \) in \( T \) \( \Leftrightarrow \) not expensive!
   3. If lookup succeeds, output: \( K1=K'_i, K2=K''_i \)

**TOTAL COST:** \( O(2^{56}) \) operations + \( O(2^{56}) \) storage
3-DES (triple DES)
- $C = E(K1, D(K2, E(K1, P))) \rightarrow 112$ effective key bits
- $C = E(K3, D(K2, E(K1, P))) \rightarrow 168$ effective key bits

DESx
- $C = K3 \text{ XOR } E(K2, (K1 \text{ XOR } P)) \rightarrow$ seems like 184 key bits
- Effective key bits $\rightarrow$ approx. 118

2-DES:
- $C = E(K2, E(K1, P)) \rightarrow$ rendezvous (meet-in-the-middle attack)

Another simple variation:
- $C = K1 \text{ XOR } E(K1', P) \rightarrow$ weak!

**DES Variants**

**Why does 3-DES (or generally n-DES) work?**

Because, as a function, DES is not a *group*...

A “group” is an algebraic structure. One of its properties is that, taking any 2 elements of the group $(a, b)$ and applying an operator $F()$ yields another element $c$ in the group.

Suppose: $C = DES(K1, DES(K2, P))$

There is no $K$, such that:

for each possible plaintext $P$, $DES(K, P) = C$
DES summary

• Permutation/substitution block cipher
• 64-bit data blocks
• 56-bit keys (8 parity bits)
• 16 rounds (shifts, XORs)
• Key schedule
• S-box selection secret...

• DES “aging”
• 2-DES: rendezvous attack
• 3-DES: 112-bit security
• DESx : 118-bit security

Other Symmetric Ciphers

Skipjack
• Classified algorithm originally designed for Clipper,
• declassified in 1998
• 32 rounds, breakable with 31 rounds
• 80 bit key, inadequate for long-term security

GOST
• GOST 28147, Russian answer to DES
• 32 rounds, 256 bit key
• Incompletely specified
Other Symmetric Ciphers

- **IDEA (X. ILai, J. Massey, ETH)**
  - Developed as PES (proposed encryption standard),
  - adapted to resist differential cryptanalysis
  - Gained popularity via PGP, 128 bit key
  - Patented (Ascom CH)

- **Blowfish (B. Schneier, Counterpane)**
  - Optimized for high-speed execution on 32-bit processors
  - 448 bit key, relatively slow key setup
  - Fast for bulk data on most PCs/laptops
  - Easy to implement, runs in ca. 5K of memory

Other Symmetric Ciphers

RC4 (Ron’s Cipher #4) Stream cipher:
- Optimized for fast software implementation
- Character streaming (not bit)
- 8-bit output
- Former trade secret of RSADSI,
- Reverse-engineered and posted to the net in 1994:
- 2048-bit key
- Used in many products until about 1999-2000
Other Symmetric Ciphers (RC4)

\[ x = y = 0; \]
\[ \text{while( length-- ) } \]
\[ \{ \quad /* \text{state[0-255] contains key bytes } */ \]
\[ \quad sx = \text{state[ ++x & 0xFF ];} \]
\[ \quad y += sx \& 0xFF; \]
\[ \quad sy = \text{state[ y ];} \]
\[ \quad \text{state[ y ] = sx;} \]
\[ \quad \text{state[ x ] = sy;} \]
\[ \quad *\text{data}++ ^{= \text{state[ ( sx+sy ) \& 0xFF ];}} \]
\[ \} \]

Takes about a minute to implement from memory

Other Symmetric Ciphers

- **RC5**
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key (0-256 bytes)
  - Very low memory requirements
  - High security (no effective attacks, yet...)
  - Data-dependent rotations
Other Symmetric Ciphers

- RC5 single round pseudocode:

\[
L \leftarrow L \text{ XOR } R \\
L \leftarrow L << R \\
L \leftarrow L + \text{ subkey}[2i] \\
R \leftarrow R \text{ XOR } L \\
R \leftarrow R << L \\
R \leftarrow R + \text{ subkey}[2i + 1]
\]

AES:
The Rijndael Block Cipher
Introduction and History

• National Institute of Science and Technology (NIST) regulates standardization in the US
• DES is an aging standard that no longer meets today’s needs for strong encryption
• Triple-DES: Endorsed by NIST as a “de facto” standard
• AES: Advanced Encryption Standard
  - Finalized in 2001
  - Goal is to define the Federal Information Processing Standard (FIPS) by selecting a new encryption algorithm suitable for encrypting (non-classified non-military) government documents
  - Candidate algorithms must be:
    • Symmetric-key ciphers supporting 128, 192, and 256 bit keys
    • Royalty-Free
    • Unclassified (i.e. public domain)
    • Available for worldwide export

Introduction and History

• AES Round-3 Finalist Algorithms:
  - MARS
    • Candidate offering from IBM Research
  - RC6
    • By Ron Rivest of MIT & RSA Labs, creator of the widely used RC4/RC5 algorithm and “R” in RSA
  - Twofish
    • From Counterpane Internet Security, Inc. (MN)
  - Serpent
    • by Ross Anderson (UK), Eli Biham (ISR) and Lars Knudsen (NO)
  - Rijndael
    • by Joan Daemen and Vincent Rijmen (B)
The Winner: Rijndael

- Joan Daemen (of Proton World International) and Vincent Rijmen (of Katholieke Universiteit Leuven).
- Pronounced “Rhine-doll”
- Allows only 128, 192, and 256-bit key sizes (unlike other candidates)
- Variable input block length: 128, 192, or 256 bits. All nine combinations of key-block length possible.
  - A block is the smallest data size the algorithm will encrypt
- Vast speed improvement over DES in both hw and sw implementations
  - 8,416 bytes/sec on a 20MHz 8051
  - 8.8 Mbytes/sec on a 200MHz Pentium Pro

Key is expanded to a set of \( n \) round keys
- Input block \( P \) put thru \( n \) rounds, each with a distinct round sub-key.
- Strength of algorithm relies on difficulty of obtaining intermediate results (or state) of round \( i \) from round \( i+1 \) without the round key.
Rijndael

Each round performs the following operations:
- Non-linear Layer: No linear relationship between the input and output of a round
- Linear Mixing Layer: Guarantees high diffusion over multiple rounds
  - Very small correlation between bytes of the round input and the bytes of the output
- Key Addition Layer: Bytes of the input are simply XOR'ed with the expanded round key

Rijndael

- Three layers provide strength against known types of cryptographic attacks: Rijndael provides “full diffusion” after only two rounds
- Immune to:
  - Linear and differential cryptanalysis
  - Related-key attacks
  - Square attack
  - Interpolation attacks
  - Weak keys
- Rijndael has been “shown” secure:
  - No key recovery attacks faster than exhaustive search exist
  - No known symmetry properties in the round mapping
  - No weak keys identified
  - No related-key attacks: No two keys have a high number of expanded round keys in common
Rijndael: ByteSub (192)

Each byte at the input of a round undergoes a non-linear byte substitution according to the following transform:

Substitution ("S")-box

Rijndael: ShiftRow

Depending on the block length, each "row" of the block is cyclically shifted according to the above table.
Rijndael: MixColumn

Each column is multiplied by a fixed polynomial
\[ C(x) = '03' \times X^3 + '01' \times X^2 + '01' \times X + '02' \]

This corresponds to matrix multiplication \( b(x) = c(x) \otimes a(x) \):

\[
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 02
\end{bmatrix}
\]

Rijndael: Key Expansion and Addition

Each word is simply XOR'ed with the expanded round key

Key Expansion algorithm:

\[
\text{KeyExpansion} \{ \text{int} \* \text{Key}[4*Nk], \text{int} \* \text{EKey}[Nb*(Nr+1)] \} \\
\{ \\
\text{for}(i = 0; i < Nk; i++) \\
\text{EKey}[i] = (\text{Key}[4*i], \text{Key}[4*i+1], \text{Key}[4*i+2], \text{Key}[4*i+3]); \\
\text{for}(i = Nk; i < Nb * (Nr + 1); i++) \\
\{ \\
\text{temp} = \text{EKey}[i - 1]; \\
\text{if} (i \% Nk == 0) \\
\text{temp} = \text{SubByte(RotByte(temp))} \xor \text{Rcon}[i / Nk]; \\
\text{EKey}[i] = \text{EKey}[i - Nk] \xor \text{temp}; \\
\} \\
\}
\]
Rijndael: Implementations

- Well-suited for software implementations on 8-bit processors (important for “Smart Cards”)
  - Atomic operations focus on bytes and nibbles, not 32- or 64-bit integers
  - Layers such as ByteSub can be efficiently implemented using small tables in ROM (e.g. < 256 bytes).
  - No special instructions are required to speed up operation, e.g. barrel rotates
- For 32-bit implementations:
  - An entire round can be implemented via a fast table lookup routine on machines with 32-bit or higher word lengths
  - Considerable parallelism exists in the algorithm
    - Each layer of Rijndael operates in a parallel manner on the bytes of the round state, all four component transforms act on individual parts of the block
    - Although the Key expansion is complicated and cannot benefit much from parallelism, it only needs to be performed once until the two parties switch keys.

Rijndael: Implementations

- Hardware Implementations
  - Rijndael performs very well in software, but there are cases when better performance is required (e.g. server and VPN applications).
  - Multiple S-Box engines, round-key XORs, and byte shifts can all be implemented efficiently in hardware when absolute speed is required
  - Small amount of hardware can vastly speed up 8-bit implementations
- Inverse Cipher
  - Except for the non-linear ByteSub step, each part of Rijndael has a straightforward inverse and the operations simply need to be undone in the reverse order.
  - However, Rijndael was specially written so that the same code that encrypts a block can also decrypt the same block simply by changing certain tables and polynomials for each layer. The rest of the operation remains identical.
Conclusions and The Future

- Rijndael is an extremely fast, state-of-the-art, highly secure algorithm
- Amenable to efficient implementation in both hw and sw; requires no special instructions to obtain good performance on any computing platform
- Triple-DES, still highly secure and supported by NIST, is expected to be common for the foreseeable future.

Reminder: World’s best cipher!
One-time pad

For each character:

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\oplus \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{array}
\]

- pad (key)
- msg (plaintext)
- ciphertext (encrypted msg)

One-time pad (cont.)

- Symmetric
- Pad is selected at random
- Pad is as long as plaintext
- Perfectly secure, but...
- One time only:
  - so sending the pad is just as hard as sending the msg
A more realistic version: Pseudo-random OTP

**Pseudo-random bit string (PRBS)** generator:

- **seed** (short) \(01101\) → PRBS → **string** (long) \(1010010110\ldots\)

*Computationally Hard* to guess a bit (after seeing many others)