ANALYSIS OF AES

This section is a brief review of the three characteristics of AES.

Topics discussed in this section:

7.6.1 Security
7.6.2 Implementation
7.6.3 Simplicity and Cost

7.6.1 Security

AES was designed after DES. Most of the known attacks on DES were already tested on AES.

Brute-Force Attack

AES is definitely more secure than DES due to the larger-size key.

Statistical Attacks

Numerous tests have failed to do statistical analysis of the ciphertext.

Differential and Linear Attacks

There are no differential and linear attacks on AES as yet.

Statistical Attacks

Numerous tests have failed to do statistical analysis of the ciphertext.

Differential and Linear Attacks

There are no differential and linear attacks on AES as yet.

7.6.2 Implementation

AES can be implemented in software, hardware, and firmware. The implementation can use table lookup process or routines that use a well-defined algebraic structure.

7.6.3 Simplicity and Cost

The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.

Concerning to the implementation aspects:

a) Rijndael can be implemented on a Smart Card in a small account of code, using a small account of RAM and taking a small number of cycles; and

b) The round transformation is parallel by design, which is an important advantage in future processors and dedicated hardware.

Comparison of AES with DES

	AES	**DES**
Block size (in bits)	128	64
Key size (in bits)	128, 192, 256	56
Speed	High	Low
Encryption primitives	Substitution, shift, bit mixing	Substitution, permutation
Cryptographic primitives	Confusion, Diffusion	Confusion, Diffusion

Comparison with Triple-DES

	AES	**Triple DES**
Type of algorithm	Symmetric, block cipher	Symmetric, feistel cipher
Key size (in bits)	128, 192, 256	112 or 168
Speed	High	Low
Time to crack	149 trillion years	4.6 billion years
Resource consumption	Low	Medium

IDEA,
RC-4, RC-5

International Data Encryption Algorithm (IDEA)

Overview

- DES algorithm has been a popular secret key encryption algorithm and is used in many commercial and financial applications. However, its key size is too small by current standards and its entire 56 bit key space can be searched in approximately 22 hours
- IDEA is a block cipher designed by Xuejia Lai and James L. Massey in 1991
- It is a minor revision of an earlier cipher, PES (Proposed Encryption Standard)
- IDEA was originally called IPES (Improved PES) and was developed to replace DES

Overview (cont')

- It entirely avoids the use of any lookup tables or S-boxes
- IDEA was used as the symmetric cipher in early versions of the Pretty Good Privacy cryptosystem

Detailed description of IDEA

- IDEA operates with 64-bit plaintext and cipher text blocks and is controlled by a 128-bit key
- Completely avoid substitution boxes and table lookups used in the block ciphers
- The algorithm structure has been chosen such that when different key sub-blocks are used, the encryption process is identical to the decryption process

Key generation

- The 64-bit plaintext block is partitioned into four 16-bit sub-blocks
- six 16-bit key are generated from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of 52 (= 8 x 6 + 4) different 16-bit sub-blocks have to be generated from the 128-bit key.

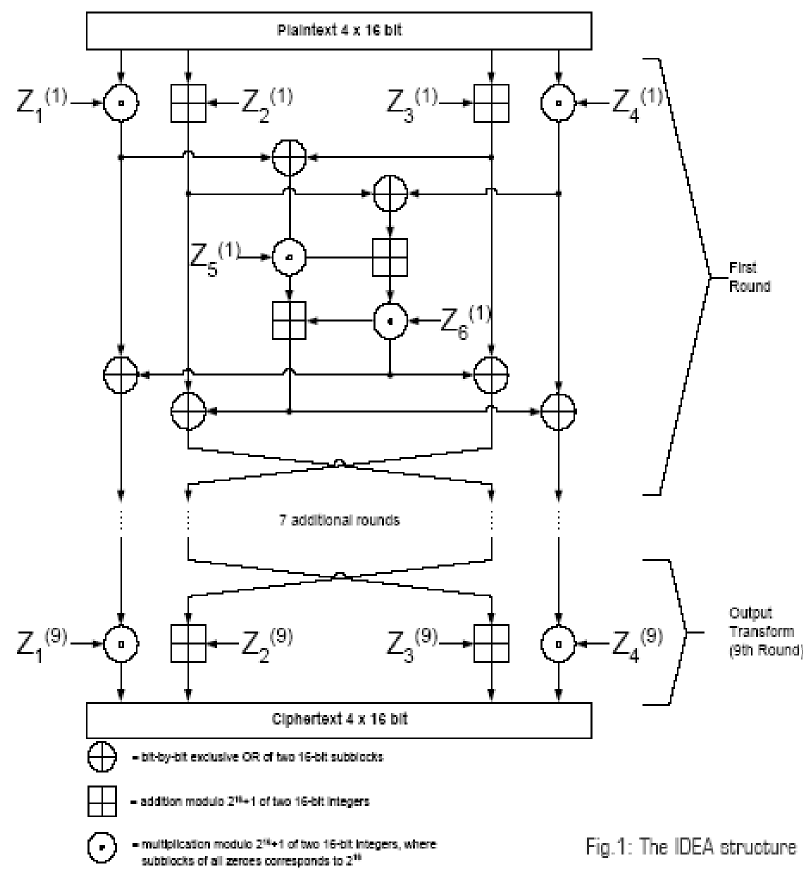


Fig.1: The IDEA structure

Key generation process

- First, the 128-bit key is partitioned into eight 16-bit sub-blocks which are then directly used as the first eight key sub-blocks
- The 128-bit key is then cyclically shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight 16-bit sub-blocks to be directly used as the next eight key sub-blocks
- The cyclic shift procedure described above is repeated until all of the required 52 16-bit key sub-blocks have been generated

Encryption of the key sub-blocks

- The key sub-blocks used for the encryption and the decryption in the individual rounds are shown in Table 1

Encryption of the key sub-blocks

Round 1	$Z_1^{(1)} Z_2^{(1)} Z_3^{(1)} Z_4^{(1)} Z_5^{(1)} Z_6^{(1)}$
Round 2	$Z_1^{(2)} Z_2^{(2)} Z_3^{(2)} Z_4^{(2)} Z_5^{(2)} Z_6^{(2)}$
Round 3	$Z_1^{(3)} Z_2^{(3)} Z_3^{(3)} Z_4^{(3)} Z_5^{(3)} Z_6^{(3)}$
Round 4	$Z_1^{(4)} Z_2^{(4)} Z_3^{(4)} Z_4^{(4)} Z_5^{(4)} Z_6^{(4)}$
Round 5	$Z_1^{(5)} Z_2^{(5)} Z_3^{(5)} Z_4^{(5)} Z_5^{(5)} Z_6^{(5)}$
Round 6	$Z_1^{(6)} Z_2^{(6)} Z_3^{(6)} Z_4^{(6)} Z_5^{(6)} Z_6^{(6)}$
Round 7	$Z_1^{(7)} Z_2^{(7)} Z_3^{(7)} Z_4^{(7)} Z_5^{(7)} Z_6^{(7)}$
Round 8	$Z_1^{(8)} Z_2^{(8)} Z_3^{(8)} Z_4^{(8)} Z_5^{(8)} Z_6^{(8)}$
Output Transform	$Z_1^{(9)} Z_2^{(9)} Z_3^{(9)} Z_4^{(9)}$

Table 1

Encryption

- the first four 16-bit key sub-blocks are combined with two of the 16-bit plaintext blocks using addition modulo 2^{16}, and with the other two plaintext blocks using multiplication modulo $2^{16} + 1$
- At the end of the first encryption round four 16-bit values are produced which are used as input to the second encryption round
- The process is repeated in each of the subsequent 7 encryption rounds
- The four 16-bit values produced at the end of the 8th encryption round are combined with the last four of the 52 key sub-blocks using addition modulo 2^{16} and multiplication modulo $2^{16} + 1$ to form the resulting four 16-bit ciphertext blocks

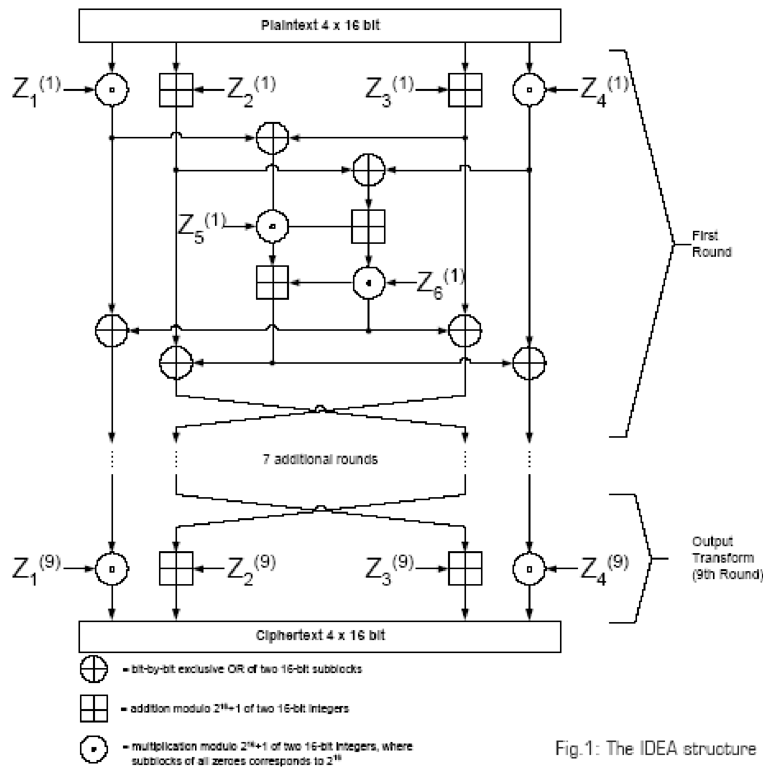


Fig.1: The IDEA structure

Decryption

- The computational process used for decryption of the ciphertext is essentially the same as that used for encryption
- The only difference is that each of the 52 16-bit key sub-blocks used for decryption is the inverse of the key sub-block used during encryption
- In addition, the key sub-blocks must be used in the reverse order during decryption in order to reverse the encryption process

Modes of operation

- IDEA supports all modes of operation such as:
  - Electronic Code Book (ECB) mode
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB) modes
- For plaintext exceeding this fixed size, the simplest approach is to partition the plaintext into blocks of equal length and encrypt each separately. This method is named Electronic Code Book (ECB) mode. However, Electronic Code Book is not a good system to use with small block sizes (for example, smaller than 40 bits)

Applications of IDEA

- Today, there are hundreds of IDEA-based security solutions available in many market areas, ranging from Financial Services, and Broadcasting to Government
- The IDEA algorithm can easily be embedded in any encryption software. Data encryption can be used to protect data transmission and storage. Typical fields are:
  - Audio and video data for cable TV, pay TV, video conferencing, distance learning
  - Sensitive financial and commercial data
  - Email via public networks
  - Smart cards

Conclusion

- As electronic communications grow in importance, there is also an increasing need for data protection
- When PGP was designed, the developers were looking for maximum security. IDEA was their first choice for data encryption
- The fundamental criteria for the development of IDEA were military strength for all security requirements and easy hardware and software implementation

Stream Ciphers

- process the message bit by bit (as a stream)
- typically have a (pseudo) random **stream key**
- combined (XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys any statistically properties in the message
  - $C_i = M_i \text{ XOR } StreamKey_i$
- what could be simpler!!!!
- but must never reuse stream key
  - otherwise can remove effect and recover messages

Stream Cipher Properties

- some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
  - correlation immunity
  - confusion
  - diffusion
  - use of highly non-linear boolean functions

RC4

- a proprietary cipher owned by RSA DSI
- another Ron Rivest design, simple but effective
- variable key size, byte-oriented stream cipher
- widely used (web SSL/TLS, wireless WEP)
- key forms random permutation of all 8-bit values
- uses that permutation to scramble input info processed a byte at a time

RC4 Key Schedule

- **Initialization of S**
- starts with an array S of numbers: 0..255
- use key to well and truly shuffle
- S forms **internal state** of the cipher
- given a key k of length l bytes

```
for i = 0 to 255 do
    S[i] = i
j = 0
for i = 0 to 255 do
    j = (j + S[i] + k[i mod l]) (mod 256)
    swap (S[i], S[j])
```

RC4 Encryption

- **Stream Generation**
- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value
- XOR with next byte of message to en/decrypt

```
i = j = 0
for each message byte M_i
    i = (i + 1) (mod 256)
    j = (j + S[i]) (mod 256)
    swap(S[i], S[j])
    t = (S[i] + S[j]) (mod 256)
    C_i = M_i XOR S[t]
```

RC4 Security

- claimed secure against known attacks
  - have some analyses, none practical
- result is very non-linear
- since RC4 is a stream cipher, must **never reuse a key**
- have a concern with WEP, but due to key handling rather than RC4 itself

Block Cipher Characteristics

- features seen in modern block ciphers are:
  - variable key length / block size / no rounds
  - mixed operators, data/key dependent rotation
  - key dependent S-boxes
  - more complex key scheduling
  - operation of full data in each round
  - varying non-linear functions

RC5

- a proprietary cipher owned by RSADSI
- designed by Ron Rivest (of RSA fame)
- used in various RSADSI products
- can vary key size / data size / no rounds
- very clean and simple design
- easy implementation on various CPUs
- close to Blowfish speeds
- yet still regarded as secure

RC5 Ciphers

- RC5 is a family of ciphers RC5-w/r/b
  - w = word size in bits (16/32/64) nb data=2w
  - r = number of rounds (0...255)
  - b = number of bytes in key (0...255)
- nominal version is RC5-32/12/16
  - ie 32-bit words so encrypts 64-bit data blocks
  - using 12 rounds
  - with 16 bytes (128-bit) secret key

RC5 Key Expansion

- RC5 uses t=2r+2 subkey words (w-bits)
- subkeys are stored in array `S[i]`, i=0..t-1
- then the key schedule consists of
  – initializing S to a fixed pseudorandom value, based on constants e and Φ
  – the key is copied into a c-word array L
  – a mixing operation then combines L and S to form the final S array

RC5 Encryption

- split input into two halves A & B

$L_0 = A + S[0];$

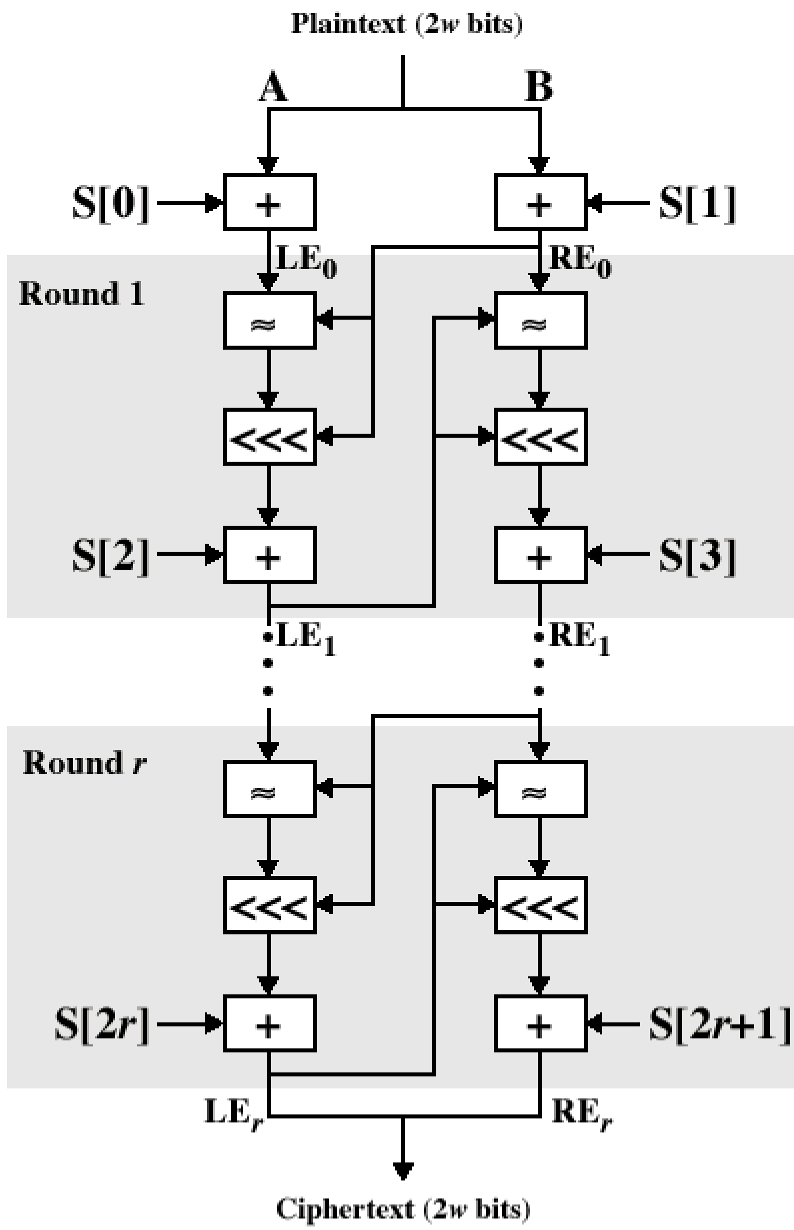
$R_0 = B + S[1];$

for $i = 1$ to r do

$L_i = ((L_{i-1} \text{ XOR } R_{i-1}) \lll R_{i-1}) + S[2i];$

$R_i = ((R_{i-1} \text{ XOR } L_i) \lll L_i) + S[2i + 1];$

- each round is like DES rounds
- note rotation is main source of non-linearity
- need reasonable number of rounds (eg 12-16)


Plaintext (2w bits)
A
B
S[0]
S[1]
LE0
RE0
Round 1
S[2]
S[3]
LE1
RE1
Round r
S[2r]
S[2r+1]
LEr
REr
Ciphertext (2w bits)

RC5 Modes

- RFC2040 defines 4 modes used by RC5
  - RC5 Block Cipher, in ECB mode
  - RC5-CBC, is CBC mode
  - RC5-CBC-PAD, is CBC with padding by bytes with value being the number of padding bytes
  - RC5-CTS, a variant of CBC which is the same size as the original message, uses ciphertext stealing to keep size same as original