

Modern Ciphers

Cryptography after DES

Designing a new cipher

- Competing requirements
 - Code footprint
 - Key agility
 - Efficient software and hardware implementation
 - Speed
 - Flexibility
 - Security

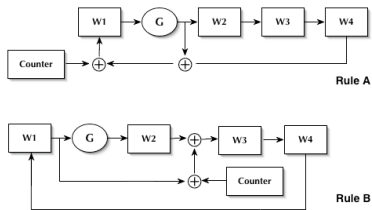
Skipjack

- Developed by NSA
- For use with the Clipper chip
- Caused controversy:
 - Secret specification
 - Push for Government-based key escrow
- Being implemented in some constrained environments.

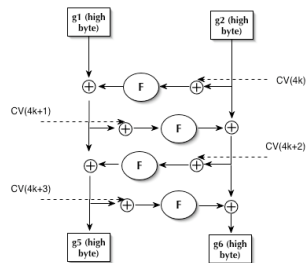
Skipjack

- 64-bit block size
- 80-bit keys
- 32 rounds
 - 8 rounds of “Rule A,” followed by 8 rounds of “Rule B”, and repeat
- The round function is a Feistel network; the encryption and decryption use different algorithms

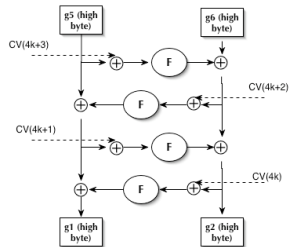
Skipjack Rules



Round function G



Decryption round function G^{-1}



The function F

- The function F is given by table lookup
- As in other ciphers, the design of F takes into consideration several goals:
 - Achieves high diffusion and confusion parameters
 - Achieves avalanche effect
 - Is far from a linear function

Key scheduling algorithm

- Extremely simple
- n -th byte of CV = $(n \bmod 10)$ -th byte of Key
- The notation CV stands for “crypto-variable”, an NSA idiosyncratic terminology for KEY.

Increasing key length

Product Ciphers and Triple-DES

Product cipher

- Consider a cipher $E()$ with key length t bits.
- Take two keys k_1 and k_2 and encrypt a message m by first encrypting it with k_1 and then with k_2 :
 - $C = E_{k_2}(E_{k_1}(m))$
 - Does the key length increase to $2t$ imply a corresponding gain in security?

Man-in-the-Middle

- Consider the following strategy to find the keys of a product cipher:
 - Obtain some pairs (P_i, C_i) of plaintext and ciphertext encrypted with the product cipher.
 - $C_i = Enc(K'', Enc(K', P_i))$ iff $Dec(K'', C_i) = Enc(K', P_i)$
 - Encrypt P_i with all possible keys K' .
 - Decrypt C_i with all possible keys K'' .
 - Select key pairs s.t. $D(K'', C_i) = E(K', C_2)$

Attack

K'	K''
$Enc(K^1, P_1)$	$Dec(K^1, C_1)$
$Enc(K^2, P_1)$	$Dec(K^2, C_1)$
$Enc(K^3, P_1)$	$Dec(K^3, C_1)$
$Enc(K^4, P_1)$	$Dec(K^4, C_1)$
...	...
$Enc(K^{2^{t-1}}, P_1)$	$Dec(K^{2^{t-1}}, C_1)$
$Enc(K^{2^t}, P_1)$	$Dec(K^{2^t}, C_1)$

- In the example, encrypting P_1 with K_1 matches decrypting C_1 with the K_3 :
 - The pair (K_1, K_3) is a candidate for the double encryption
 - The table has 2^{t+1} entries, not 2^{2t}
 - Security equivalent to using a key only one or two bits longer!

Time/memory trade-off

- In order to optimize finding matches:
- Compute $A_i = Enc(K', P_i)$. Interpret A_i as an address location and store the value K' there.
- Compute $B_i = Dec(K'', C_i)$. Interpret B_i as an address location and store the value K'' there.
- If a key K' and a key K'' try to occupy the same address, output (K', K'') as a possible key pair.
- Time/memory trade-off:
 - Use only part of the block value to address.
 - Store only part of the key data.

Triple-DES

- 3-DES is usually implemented as a encryption followed by a decryption (with a different key) followed by another encryption: DES-EDE
- Two- and three-key variants, to accomplish 112 and 168-bit keys:
 - $DES(K_1, DES^{-1}(K_2, DES(K_1, m)))$
 - $DES(K_3, DES^{-1}(K_2, DES(K_1, m)))$

Attacking triple encryption

- Merkle devised the following attack.
- For each key K' , compute $P(K') = \text{Dec}(K'', \mathbf{0})$.
- For each K' , get $P(K')$ encrypted (using 3-encryption).
- Apply previous attack on double encryption, with $P_1 = \mathbf{0}$, to find candidates (K'', K''') for each K' .
- Conclusion: 2^{3t} computations not required
 - Many pairs of plaintext/ciphertext required.
