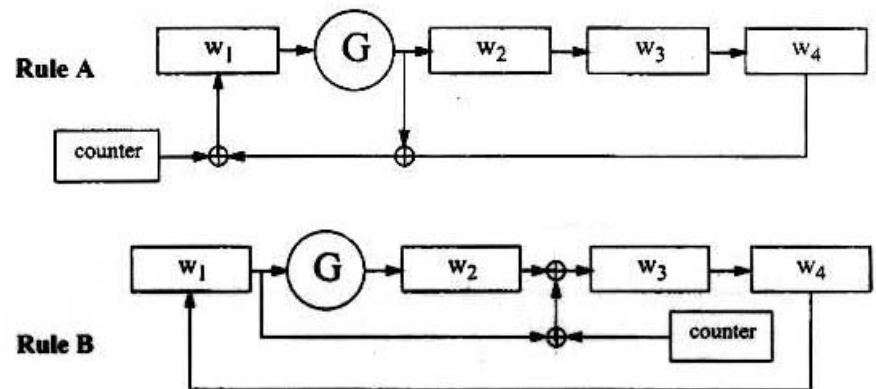


SKIPJACK and Key Exchange Algorithm (KEA)

Carlton J. O'Riley

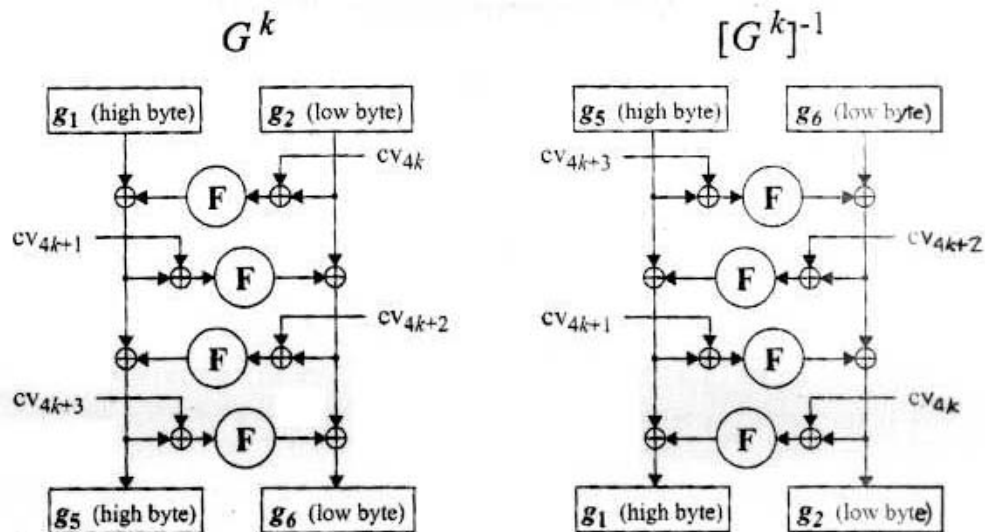
SKIPJACK Algorithm

- 64-bit Input/Output
- 80-bit Cryptovvariable (Key)
- 32 Rounds
 - 8 Rounds of Rule A
 - 8 Rounds of Rule B
 - 8 Rounds of Rule A
 - 8 Rounds of Rule B



SKIPJACK Algorithm (Cont'd)

- G Permutation Function
 - 4 Round Feistel Structure
- F Byte Substitution Table



SKIPJACK Optimizations

- Precomputation of F-Table Key Values

$F(\text{Key}[0][0..255])$

...

$F(\text{Key}[9][0...255])$

- Requires 2560 bytes of a lookup table, but saves 4 XORs per round

SKIPJACK Optimizations (Cont'd)

- Not Shifting Words, But Operands
- Inline G Functions (not subroutines)
- Pointers To Buffer Data
- Compiler Optimizations

SKIPJACK Optimizations (Cont'd)

	Encryption	Decryption
SKIPJACK	34 ms	29 ms
DES	16 ms	16 ms

Key Exchange Algorithm

KEA Parameters

- Common Network Parameters
 - p : 1024-bit prime number
 - q : 160-bit prime divisor of $p-1$
 - g : 1024-bit base for exponentiation
- x : 160-bit Private Keys
- Y : 1024-bit Public Key (based on x)
- r : 160-bit Private Random Number
- R : 1024-bit Public Random Number (based on r)

Store and Forward KEA (Cont'd)

Device A

p, q, g

x_A

Y_A

Y_B

r_A

R_A

common to both devices

private key of each device

$$Y = g^x \text{ mod } p$$

send Y_A in message

A obtains B's public
from directory or local cache

A generates a random number

$$R = g^r \text{ mod } p$$

Device B

p, q, g

x_B

Y_B

v_1, v_2

Key

$$t_{AB} = (Y_B)^{r_A} \text{ mod } p$$

$$u_{AB} = (Y_B)^{x_A} \text{ mod } p$$

$$w = (t_{AB} + u_{AB}) \text{ mod } p$$

check all values received

$$\text{compute } t = g^{r_A x_B} \text{ mod } p$$

$$\text{compute } u = g^{x_A x_B} \text{ mod } p$$

compute w
and check $w \neq 0$

extract v_1 and v_2 from w

form Key from $v_1, v_2, \text{ pad}$

$$t_{BA} = (R_A)^{x_B} \text{ mod } p$$

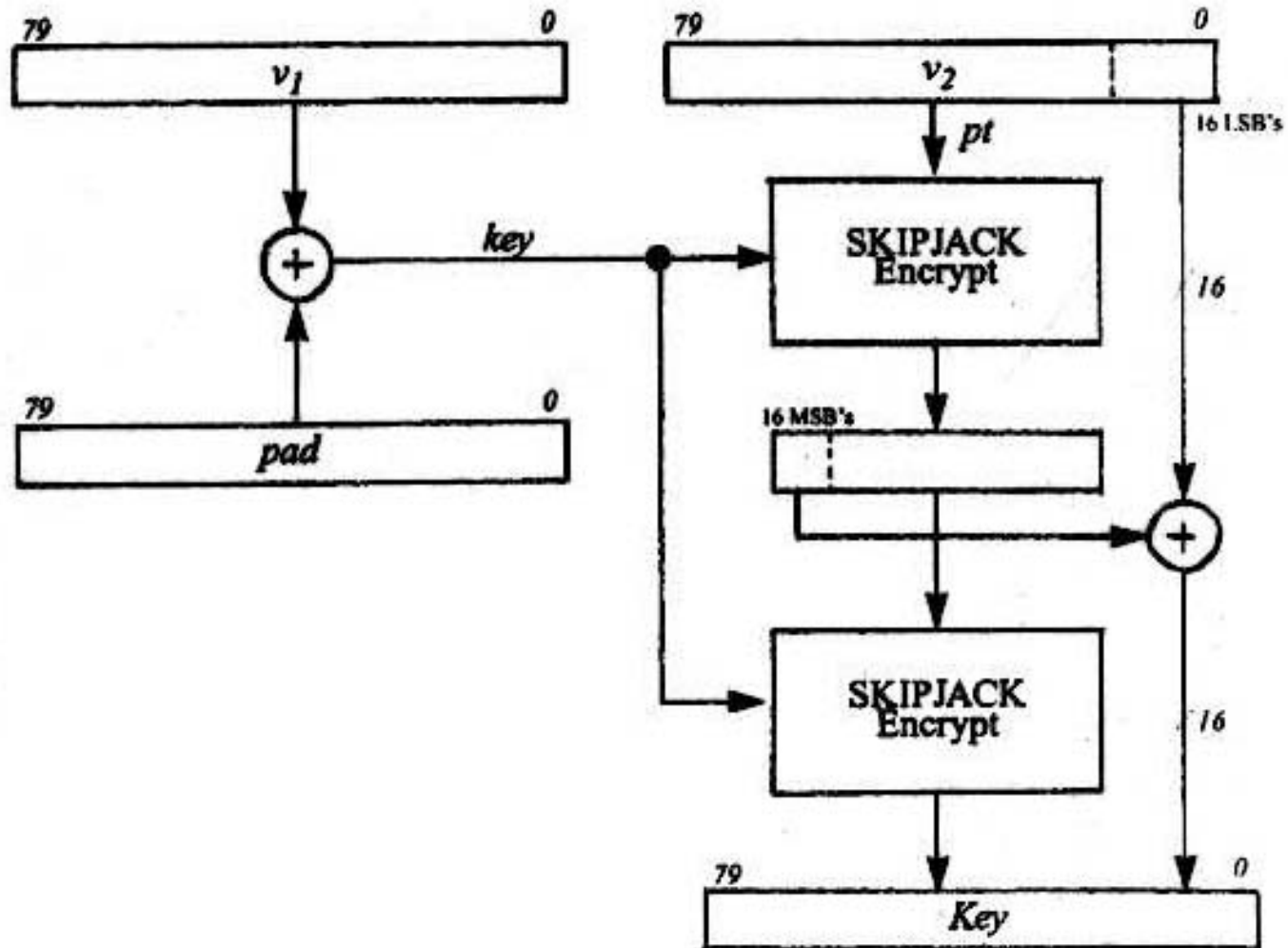
$$u_{BA} = (Y_A)^{x_B} \text{ mod } p$$

$$w = (t_{BA} + u_{BA}) \text{ mod } p$$

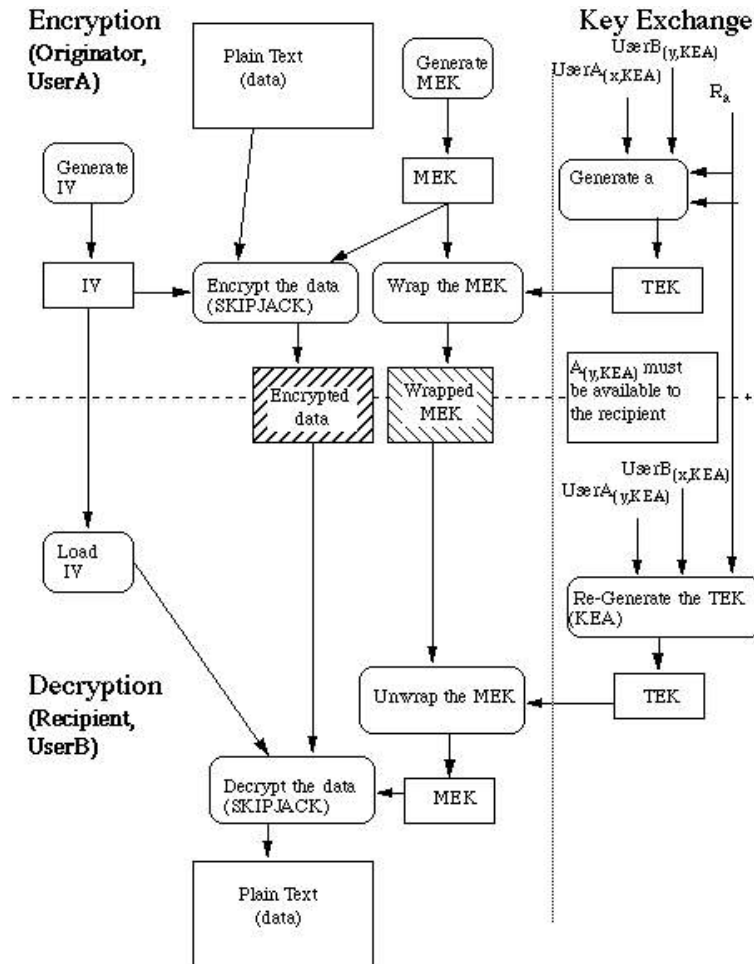
v_1, v_2

Key

KEA Key Generation



FORTEZZA KEA



Wrapping the MEK

- Pad 80-bit MEK With Six Bytes of 06
- Encrypt Each 64-bits Using SKIPJACK With TEK As Key