SKIPJACK and KEA Algorithm Specifications

Version 2.0

29 May 1998
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I. Introduction</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>II. Algorithms</strong></td>
<td>3</td>
</tr>
<tr>
<td>A. SKIPJACK Modes of Operation</td>
<td>3</td>
</tr>
<tr>
<td>B. SKIPJACK Specification</td>
<td>5</td>
</tr>
<tr>
<td>1. Notation and Terminology</td>
<td>5</td>
</tr>
<tr>
<td>2. Basic Structure</td>
<td>5</td>
</tr>
<tr>
<td>3. Stepping Rule Equations</td>
<td>6</td>
</tr>
<tr>
<td>4. Stepping Sequence</td>
<td>6</td>
</tr>
<tr>
<td>5. G-Permutation</td>
<td>7</td>
</tr>
<tr>
<td>6. Cryptovariable Schedule</td>
<td>7</td>
</tr>
<tr>
<td>7. F-Table</td>
<td>8</td>
</tr>
<tr>
<td>C. KEA Specification</td>
<td>9</td>
</tr>
<tr>
<td>D. E-Mail Applications of KEA</td>
<td>13</td>
</tr>
<tr>
<td>1. Sending E-Mail</td>
<td>13</td>
</tr>
<tr>
<td>2. Receiving E-Mail</td>
<td>15</td>
</tr>
<tr>
<td><strong>III. ANNEX - Test Vectors</strong></td>
<td>18</td>
</tr>
<tr>
<td>A. SKIPJACK Codebook Mode</td>
<td>18</td>
</tr>
<tr>
<td>B. Key Exchange Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>C. KEA Exchange for E-Mail</td>
<td>21</td>
</tr>
<tr>
<td><strong>IV. References</strong></td>
<td>23</td>
</tr>
</tbody>
</table>
I. Introduction

This document provides details of the SKIPJACK and KEA algorithms. The algorithms are supported in single chip cryptoprocessors such as CLIPPER (SKIPJACK only), CAPSTONE, KEYSTONE, REGENT, KRYPTON and the FORTEZZA and FORTEZZA Plus PC Card firmware which runs on them, and also in other FORTEZZA family products.

II. Algorithms

This document will discuss the following algorithms:

- SKIPJACK: Codebook Encryptor/Decryptor Algorithm
- KEA: Key Exchange Algorithm

A. SKIPJACK Modes of Operation

SKIPJACK is a 64 bit codebook utilizing an 80-bit crypto-variable. The modes of operation are a subset of the FIPS-81 description of modes of operation for DES [1]. These include:

- Output Feed-Back (OFB) Modes: 64 bit
- Cipher Feed-Back (CFB) Modes: 64 bit/32 bit/16 bit/8 bit
- Codebook: 64 bit
- Cipher-Block Chaining (CBC): 64 bit

![Diagram](image)

*Figure 1. “Output Feed-Back Modes Diagram”*
Figure 2. “Cipher Feed-Back Mode Diagram”

Figure 3. “Codebook Mode Diagram”

Figure 4. “Cipher-Block Chaining Mode Diagram”
B. SKIPJACK Specification

1. Notation and terminology:

- \( V^n \): the set of all \( n \)-bit values.
- word: an element of \( V^{16} \); a 16-bit value.
- byte: an element of \( V^8 \); an 8-bit value.
- permutation on \( V^n \): an invertible (one-to-one and onto) function from \( V^n \) to \( V^n \). That is, the values are permuted within \( V^n \), not the bits within the value.
- \( X \oplus Y \): the bitwise exclusive-or of \( X \) and \( Y \).
- \( X \| Y \): \( X \) concatenated with \( Y \). Let \( X, Y \) be bytes, then \( X \| Y = X \times 2^8 + Y \) is a word. Furthermore, \( X \) is the high-order byte, and \( Y \) is the low-order byte.

2. Basic structure: SKIPJACK encrypts 4-word (i.e., 8-byte) data blocks by alternating between the two stepping rules (A and B) shown below. A step of rule A does the following:

   a. \( G \) permutes \( w_1 \).
   b. the new \( w_1 \) is the xor of the \( G \) output, the counter, and \( w_4 \).
   c. words \( w_2 \) and \( w_3 \) shift one register to the right; i.e., become \( w_3 \), and \( w_4 \) respectively,
   d. the new \( w_2 \) is the \( G \) output,
   e. the counter is incremented by one.

Rule B works similarly.

![Figure 5. “SKIPJACK Stepping Rules”](image-url)
3. **Stepping rule equations.** In the equations below, the superscript is the step number.

**ENCRYPT**

**Rule A**

\[ w_{1}^{k+1} = G^{k}(w_{1}^{k}) \oplus w_{4}^{k} \oplus \text{counter}^{k} \]

\[ w_{2}^{k+1} = G^{k}(w_{2}^{k}) \]

\[ w_{3}^{k+1} = w_{2}^{k} \]

\[ w_{4}^{k+1} = w_{3}^{k} \]

**Rule B**

\[ w_{1}^{k+1} = w_{4}^{k} \]

\[ w_{2}^{k+1} = G^{k}(w_{1}^{k}) \]

\[ w_{3}^{k+1} = w_{1}^{k} \oplus w_{2}^{k} \oplus \text{counter}^{k} \]

\[ w_{4}^{k+1} = w_{3}^{k} \]

**DECRESS**

**Rule A\(^{-1}\)**

\[ w_{1}^{k-1} = [G^{k-1}]^{-1}(w_{2}^{k}) \]

\[ w_{2}^{k-1} = w_{3}^{k} \]

\[ w_{3}^{k-1} = w_{4}^{k} \]

\[ w_{4}^{k-1} = w_{1}^{k} \oplus w_{2}^{k} \oplus \text{counter}^{k-1} \]

**Rule B\(^{-1}\)**

\[ w_{1}^{k-1} = [G^{k-1}]^{-1}(w_{2}^{k}) \]

\[ w_{2}^{k-1} = [G^{k-1}]^{-1}(w_{2}^{k}) \oplus w_{3}^{k} \oplus \text{counter}^{k-1} \]

\[ w_{3}^{k-1} = w_{4}^{k} \]

\[ w_{4}^{k-1} = w_{1}^{k} \]

4. **Stepping sequence:** The algorithm requires a total of 32 steps.

   a. To encrypt: The input is \( w_{i}^{0}, 1 \leq i \leq 4 \), (i.e., \( k = 0 \) for the beginning step). Start the counter at 1. Step according to Rule A for 8 steps, then switch to Rule B and step 8 more times. Return to Rule A for the next 8 steps, then complete the encryption with 8 steps in Rule B. The counter increments by one after each step. The output is \( w_{i}^{32}, 1 \leq i \leq 4 \).

   b. To decrypt: The input is \( w_{i}^{32}, 1 \leq i \leq 4 \), (i.e., \( k = 32 \) for the beginning step). Start the counter at 32. Step according to Rule B\(^{-1}\) for 8 steps, then switch to Rule A\(^{-1}\) and step 8 more times. Return to Rule B\(^{-1}\) for the next 8 steps, then complete the decryption with 8 steps in Rule A\(^{-1}\). The counter decrements by one after every step. The output is \( w_{i}^{0}, 1 \leq i \leq 4 \).
5. **G-permutation**: The cryptovariable-dependent permutation $G$ on $\nu^{16}$ is a four-round Feistel structure. The round function is a fixed byte-substitution table (permutation on $\nu^8$), which will be called the $F$-table. Each round of $G$ also incorporates a byte of cryptovariable. We give two characterizations of the function below:

a. **recursively (mathematically):** 
\[ G^k(w = g_1||g_2) = g_5||g_6 \] 
where 
\[ g_i = F(g_{i-1} \oplus cv_{4k+i-3}) \oplus g_{i-2} \] 
and where $k$ is the step number (the first step is 0), $F$ is the substitution table, and $cv_{4k+i}$ is the $(4k+i)$th byte in the cryptovariable schedule. Thus, 
\[ g_3 = F(g_2 \oplus cv_{4k}) \oplus g_1 \]
\[ g_4 = F(g_3 \oplus cv_{4k+1}) \oplus g_2 \]
\[ g_5 = F(g_4 \oplus cv_{4k+2}) \oplus g_3 \]
\[ g_6 = F(g_5 \oplus cv_{4k+3}) \oplus g_4 \]

Similarly, for the inverse, 
\[ [G^k]^{-1}(w = g_5||g_6) = g_1||g_2 \] 
where 
\[ g_{i-2} = F(g_{i-1} \oplus cv_{4k+i-3}) \oplus g_i \].

b. **schematically:**

![G-permutation diagram](image)

*Figure 6. "G-permutation diagram"

6. **Cryptovariable schedule**: The cryptovariable is 10 bytes long (labelled 0 through 9) and used in its natural order. So the schedule subscripts given in the definition of the G-permutation are to be interpreted mod-10.
7. F Table: The SKIPJACK F-table is given below in hexadecimal notation. The high order 4 bits of the input index the row and the low order 4 bits index the column. For example, F(7a) = d6.

<table>
<thead>
<tr>
<th>x0</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>xA</th>
<th>xB</th>
<th>xC</th>
<th>xD</th>
<th>xE</th>
<th>xF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x</td>
<td>a3</td>
<td>d7</td>
<td>09</td>
<td>83</td>
<td>f8</td>
<td>48</td>
<td>f6</td>
<td>f4</td>
<td>b3</td>
<td>21</td>
<td>15</td>
<td>78</td>
<td>99</td>
<td>b1</td>
<td>af</td>
</tr>
<tr>
<td>1x</td>
<td>c7</td>
<td>2d</td>
<td>4d</td>
<td>8a</td>
<td>ce</td>
<td>4c</td>
<td>ca</td>
<td>2e</td>
<td>52</td>
<td>95</td>
<td>d9</td>
<td>le</td>
<td>4e</td>
<td>38</td>
<td>44</td>
</tr>
<tr>
<td>2x</td>
<td>0a</td>
<td>df</td>
<td>02</td>
<td>a0</td>
<td>17</td>
<td>f1</td>
<td>60</td>
<td>68</td>
<td>12</td>
<td>b7</td>
<td>7a</td>
<td>c3</td>
<td>e9</td>
<td>fa</td>
<td>3d</td>
</tr>
<tr>
<td>3x</td>
<td>96</td>
<td>84</td>
<td>6b</td>
<td>ba</td>
<td>f2</td>
<td>63</td>
<td>9a</td>
<td>19</td>
<td>7c</td>
<td>ae</td>
<td>e5</td>
<td>f5</td>
<td>f7</td>
<td>16</td>
<td>6a</td>
</tr>
<tr>
<td>4x</td>
<td>39</td>
<td>b6</td>
<td>7b</td>
<td>0f</td>
<td>c1</td>
<td>93</td>
<td>81</td>
<td>1b</td>
<td>ee</td>
<td>b4</td>
<td>la</td>
<td>ea</td>
<td>do</td>
<td>91</td>
<td>2f</td>
</tr>
<tr>
<td>5x</td>
<td>55</td>
<td>b9</td>
<td>da</td>
<td>85</td>
<td>3f</td>
<td>41</td>
<td>bf</td>
<td>e0</td>
<td>5a</td>
<td>58</td>
<td>80</td>
<td>5f</td>
<td>66</td>
<td>0b</td>
<td>d8</td>
</tr>
<tr>
<td>6x</td>
<td>35</td>
<td>d5</td>
<td>e0</td>
<td>a7</td>
<td>33</td>
<td>06</td>
<td>65</td>
<td>69</td>
<td>45</td>
<td>00</td>
<td>94</td>
<td>56</td>
<td>6d</td>
<td>98</td>
<td>9b</td>
</tr>
<tr>
<td>7x</td>
<td>97</td>
<td>fc</td>
<td>b2</td>
<td>c2</td>
<td>b0</td>
<td>fe</td>
<td>db</td>
<td>20</td>
<td>e1</td>
<td>eb</td>
<td>d6</td>
<td>e4</td>
<td>dd</td>
<td>47</td>
<td>4a</td>
</tr>
<tr>
<td>8x</td>
<td>42</td>
<td>ed</td>
<td>9e</td>
<td>6e</td>
<td>49</td>
<td>3c</td>
<td>cd</td>
<td>43</td>
<td>27</td>
<td>d2</td>
<td>07</td>
<td>d4</td>
<td>de</td>
<td>c7</td>
<td>67</td>
</tr>
<tr>
<td>9x</td>
<td>89</td>
<td>cb</td>
<td>30</td>
<td>1f</td>
<td>8d</td>
<td>c6</td>
<td>8f</td>
<td>aa</td>
<td>c8</td>
<td>74</td>
<td>dc</td>
<td>c9</td>
<td>5d</td>
<td>5c</td>
<td>31</td>
</tr>
<tr>
<td>Ax</td>
<td>70</td>
<td>88</td>
<td>61</td>
<td>2c</td>
<td>9f</td>
<td>0d</td>
<td>2b</td>
<td>87</td>
<td>50</td>
<td>82</td>
<td>54</td>
<td>64</td>
<td>26</td>
<td>7d</td>
<td>03</td>
</tr>
<tr>
<td>Bx</td>
<td>34</td>
<td>4b</td>
<td>1c</td>
<td>73</td>
<td>d1</td>
<td>c4</td>
<td>fd</td>
<td>3b</td>
<td>cc</td>
<td>fb</td>
<td>7f</td>
<td>ab</td>
<td>e6</td>
<td>3c</td>
<td>5b</td>
</tr>
<tr>
<td>Cx</td>
<td>ad</td>
<td>04</td>
<td>23</td>
<td>9e</td>
<td>14</td>
<td>51</td>
<td>22</td>
<td>f0</td>
<td>29</td>
<td>79</td>
<td>71</td>
<td>7e</td>
<td>ff</td>
<td>8c</td>
<td>0e</td>
</tr>
<tr>
<td>Dx</td>
<td>0c</td>
<td>ef</td>
<td>bc</td>
<td>72</td>
<td>75</td>
<td>6f</td>
<td>37</td>
<td>a1</td>
<td>ec</td>
<td>d3</td>
<td>8e</td>
<td>62</td>
<td>8b</td>
<td>86</td>
<td>10</td>
</tr>
<tr>
<td>Ex</td>
<td>08</td>
<td>77</td>
<td>11</td>
<td>be</td>
<td>92</td>
<td>4f</td>
<td>24</td>
<td>e5</td>
<td>32</td>
<td>36</td>
<td>9d</td>
<td>cf</td>
<td>f3</td>
<td>a6</td>
<td>bb</td>
</tr>
<tr>
<td>Fx</td>
<td>5e</td>
<td>6c</td>
<td>a9</td>
<td>13</td>
<td>57</td>
<td>25</td>
<td>b5</td>
<td>e3</td>
<td>bd</td>
<td>a8</td>
<td>3a</td>
<td>01</td>
<td>05</td>
<td>59</td>
<td>2a</td>
</tr>
</tbody>
</table>
C. KEA Specification

KEA is a key exchange algorithm. All calculations for KEA require a 1024-bit prime modulus. This modulus and related values are to be generated as per the DSS specification [2]. The KEA is based upon a Diffie-Hellman protocol utilizing SKIPJACK to reduce final values to an 80 bit key.

KEA operations require exponents of length 160 bits. One exponent used in KEA is a user specific secret component.

The KEA provides security commensurate with that provided by SKIPJACK. This is on the order of $2^{80}$ operations.

KEA requires that each user be able to validate the public values received from others, but does not specify how that is to be done.

The devices must be provided the following data in order to implement the Key Exchange Algorithm (KEA).

- $p$ 1024-bit prime modulus which defines the field where
  \[ p = p_{1023}p_{1022} \cdots p_0 \]

- $q$ 160-bit prime divisor of $p-1$ for public component checking
  \[ q = q_{159}q_{158} \cdots q_0 \]

- $g$ 1024-bit base for the exponentiation. An element of order $q$ in the multiplicative group mod $p$.
  \[ g = g_{1023}g_{1022} \cdots g_0 \]

- $x$ 160-bit user secret number chosen so that $0 < x < q$
  \[ x = x_{159}x_{158} \cdots x_0 \]

- $Y$ 1024-bit public value corresponding to private value $x$
  \[ Y = g^x \mod p = Y_{1023}Y_{1022} \cdots Y_0 \]

- $pad$ 80 bit padding value
  \[ pad = pad_{79}pad_{78} \cdots pad_0 \]
  \[ = 72f1a87e92824198ab0b \] hex.

- $r$ 160-bit random number
  \[ r = r_{159}r_{158} \cdots r_0 \]

A signaling requirement for the determination of the initiator and the recipient of an exchange is not necessary. A description of the process follows. For two users A and B, the subscripts A and B are used to denote the ‘owner’ of the respective values.

a. A and B exchange or obtain from a directory the certificate(s) of the far terminal. From the certificate(s), the public value $Y$ of the other terminal can be obtained along with associated user identification and other information.
b. Each device validates the public key \( Y \) to determine that it is indeed the public key of a valid user on the network. If the validation fails, the process terminates. If the validation checks, go to step c.

c. Each device exchanges the random component. Device A generates a 160-bit private random number \( r_A \) and sends the public version of this number

\[
R_A = g^{r_A} \mod p
\]

Device B generates a 160-bit \( r_B \) and sends

\[
R_B = g^{r_B} \mod p
\]

Each of these public random components is 1024-bits in length.

d. After receiving the public random component and the far end public key, each device will check to verify both the received values are of order \( q \). Device A will compute and verify:

\[
1 < R_B, Y_B < p
\]

\[
(R_B)^q \equiv 1 \mod p \quad \text{and} \quad (Y_B)^q \equiv 1 \mod p
\]

Device B will compute and verify

\[
1 < R_A, Y_A < p
\]

\[
(R_A)^q \equiv 1 \mod p \quad \text{and} \quad (Y_A)^q \equiv 1 \mod p
\]

If the verification checks, go to step e. Should the verification fail, stop.

e. Device A will take \( Y_B \) and compute the value \( t_{AB} \). Device B will compute the equivalent value \( t_{BA} \) using the received random component

\[
t_{AB} = (Y_B)^{r_A} \mod p = g^{s_Br_A} \mod p
\]

\[
t_{BA} = (R_A)^{s_B} \mod p = g^{r_As_B} \mod p = g^{s_Br_A} \mod p
\]

f. Each device computes \( u \) in a similar manner as they computed \( t \)

\[
u_{BA} = (Y_A)^{r_B} \mod p = s_Ar_B \mod p
\]

\[
u_{AB} = (R_B)^{s_A} \mod p = g^{rBs_A} \mod p = g^{s_Ar_B} \mod p
\]

g. Each device computes \( w \) and checks to make sure that

\[
w = (t + u) \mod p \neq 0
\]

If this check passes, go to step h. Else stop.
h. This result is split into two sections

\[ v_1 = \left( \frac{w}{2^{(1024-80)}} \right) \mod 2^{80} \quad v_2 = \left( \frac{w}{2^{(1024-160)}} \right) \mod 2^{80} \]

i.e., if we number the bits in \( w \) as \( w_{1023} \ldots w_0 \) from MSB to LSB, then

\[ v_1 = w_{1023} \ldots w_{944} \quad \text{and} \quad v_2 = w_{943} \ldots w_{864} \]

i. The Key is

\[ Key = 2^{16} \left[ E_{v_1 \oplus pad} \left( E_{v_1 \oplus pad} \left( \frac{v_2}{2^{16}} \mod 2^{64} \right) \right) \right] \]

\[ \oplus \left[ \frac{E_{v_1 \oplus pad} \left( \frac{v_2}{2^{16}} \mod 2^{64} \right)}{2^{48}} \right] \oplus (v_2 \mod 2^{16}) \]

Note that this function represents the encryption of \( v_2 \) with \( v_1 \) XOR \( pad \).

Pictorially,

![Diagram](image_url)

*Figure 7. "Key Formation Diagram"*
A summary of a full KEA exchange between devices A and B is as follows:

**Device A**

- \( p, q, g \) common to both devices
- \( x_A \), private key of each device
- \( Y_A = g^{x_A} \mod p \)
- \( Y_B \) obtain other devices' public via certificate or sent in msg

**Device B**

- \( p, q, g \)
- \( x_B \)
- \( Y_B \)

**A and B generate random numbers**

- \( r_A \)
- \( R_A = g^{r_A} \mod p \)
- \( R_B \) exchange public random numbers

- \( t_{AB} = (Y_B)^{r_A} \mod p \) check all values received
- \( u_{AB} = (R_B)^{x_A} \mod p \) compute \( t = g^{r_A x_B} \mod p \)
- \( w = (t_{AB} + u_{AB}) \mod p \) compute \( u = g^{x_A r_B} \mod p \) and check \( w \neq 0 \)

- \( v_1, v_2 \) extract \( v_1 \) and \( v_2 \) from \( w \)

**Key**

- form Key from \( v_1, v_2, \) pad
D. E-Mail Applications of KEA

For electronic mail applications where the recipient does not participate in the formation of the key, the recipients contribution to the random exchange is replaced with the public key of the recipient. For the following, let A be the sender and B be the recipient of the E-mail message. We first begin with the formation of the E-mail message.

1. Sending E-Mail

a. Device A obtains from a directory or a local cache the certificate(s) of the far terminal. From the certificate(s), the public value $Y_B$ of terminal B can be obtained along with associated user identification and other information.

b. Device A validates the public key $Y_B$ to determine that it is indeed the public key of a valid user on the network. If the validation fails, the process terminates. If the validation checks, go to step c.

c. Device A will then verify:

$$1 < Y_B < p \quad \text{and} \quad (Y_B)^q \equiv 1 \mod p$$

If the verification checks, go to step d. Should the verification fail, stop.

d. Device A generates the random number $r_A$ and computes $R_A$ which is placed in the message packet to be sent to the far terminal.

$$R_A = g^{r_A} \mod p$$

This random component is 1024 bits in length.

e. Device A will then take $Y_B$ and compute the value $t_{AB}$.

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

f. Device A computes

$$u_{AB} = (Y_B)^{x_A} \mod p = g^{x_A r_A} \mod p = g^{x_A r_B} \mod p$$

g. Device A then computes $w$ and checks to make sure that

$$w = (t_{AB} + u_{AB}) \mod p \neq 0$$

If this check passes, go to step h. Else stop.
h. This result is split into two sections

\[ v_1 = \left( \frac{w}{2^{(1024 - 80)}} \right) \mod 2^{80} \quad v_2 = \left( \frac{w}{2^{(1024 - 160)}} \right) \mod 2^{80} \]

i.e., if we number the bits in \( w \) as \( w_{1023} \ldots w_0 \) from MSB to LSB, then

\[ v_1 = w_{1023} \ldots w_{944} \quad \text{and} \quad v_2 = w_{943} \ldots w_{864} \]

i. The Key is

\[ Key = 2^{16} \left[ E_{v_1 \oplus pad} \left( E_{v_1 \oplus pad} \left[ \frac{v_2}{2^{16}} \mod 2^{64} \right] \right) \right] \]

\[ \oplus \left( \frac{v_2}{2^{16}} \mod 2^{64} \right) \]

\[ \oplus \left( v_2 \mod 2^{16} \right) \]

Note that function represents the encryption of \( v_2 \) with \( v1 \) XOR pad. Pictorially,

![Diagram](image)

*Figure 8. "Key Formation Diagram"*
2. Receiving E-Mail

a. Device B obtains the certificate(s) of the far terminal, A, in the received E-mail message. From the certificate(s), the public value $Y_A$ of terminal A can be obtained along with associated user identification and other information.

b. Device B validates the public key $Y_A$ to determine that it is indeed the public key of a valid user on the network. If the validation fails, the process terminates. If the validation checks, go to step c.

c. Device B receives the random component that A generated.

$$R_A = g^{r_A} \mod p$$

This random component is 1024-bits in length.

d. Device B will compute and verify:

$$1 < R_A \cdot Y_A < p$$

$$(R_A)^q \equiv 1 \mod p \quad \text{and} \quad (Y_A)^q \equiv 1 \mod p$$

If the verification checks, go to step e. Should the verification fail, stop.

e. Device B will take $R_A$ and compute the value $t_{BA}$.

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

f. Device B computes:

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

g. Device B computes $w$ and checks to make sure that

$$w = (t_{BA} + u_{BA}) \mod p \neq 0$$

If this check passes, go to step h. Else stop.

h. This result is split into two sections

$$v_1 = \left(\frac{w}{2^{(1024-80)}}\right) \mod 2^{80} \quad \quad v_2 = \left(\frac{w}{2^{(1024-160)}}\right) \mod 2^{80}$$

i.e., if we number the bits in $w$ as $w_{1023} \ldots w_0$ from MSB to LSB, then $v_1 = w_{1023} \ldots w_{944}$ and $v_2 = w_{943} \ldots w_{864}$
i. The Key is

\[ Key = 2^{16} \left[ E_{v_1 \oplus pad} \left( E_{v_1 \oplus pad} \left( \frac{v_2}{2^{16}} \mod 2^{64} \right) \right) \right] \]

\[ \oplus \left( \frac{E_{v_1 \oplus pad} \left( \frac{v_2}{2^{16}} \mod 2^{64} \right)}{2^{48}} \right) \oplus (v_2 \mod 2^{16}) \]

Note that function represents the encryption of \( v_2 \) with \( v_1 \) XOR pad. Pictorially,

**Figure 9.** “Key Formation Diagram”
A summary of an E-mail KEA exchange between devices A and B is as follows:

**Device A**

- \( p, q, g \) common to both devices
- \( x_A \) private key of each device
- \( Y_A = g^{x_A} \mod p \)
- Send \( Y_A \) in message
- \( r_A \) A generates a random number
- \( R_A = g^{r_A} \mod p \)

**Device B**

- \( p, q, g \)
- \( x_B \)
- \( Y_B = g^{x_B} \mod p \)
- \( Y_B \) A obtains B's public from directory or local cache

**Calculations**

- \( t_{AB} = (Y_B)^{r_A} \mod p \)
- Compute \( t = g^{r_A x_B} \mod p \)
- \( t_{BA} = (R_A)^{x_B} \mod p \)
- \( u_{AB} = (Y_B)^{x_A} \mod p \)
- Compute \( u = g^{x_A x_B} \mod p \)
- \( u_{BA} = (Y_A)^{x_B} \mod p \)
- \( w = (t_{AB} + u_{AB}) \mod p \)
- Compute \( w = (t_{BA} + u_{BA}) \mod p \)

- \( v1, v2 \) extract \( v1 \) and \( v2 \) from \( w \)
- Key form Key from \( v1, v2 \), pad
III. ANNEX - Test Vectors

All values are hexadecimal. This data does not imply or specify any interface convention. All information is presented with the Most Significant Bit/Byte/Word to the left. X represents "don’t-care".

A. SKIPJACK - CODEBOOK MODE

Plaintext input: 33221100ddccbbaa
Cryptovariable: 00998877665544332211
Intermediate steps:

<table>
<thead>
<tr>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>w4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>33221100 ddccbbaa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b0040baf 1100ddcc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>e6883b46 0baf1100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3c762d75 3b460baf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4c4547ee 2d753b46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b949820a 47ee2d75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>f0e3dd90 820a47ee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>f9b9be50 dd90820a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>d79bb599 be50dd90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>dd90e0b 820bbe50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>be504c52 c391820b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>820b7f51 f209c391</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>c391fc9c2 fd56f209</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>f20925ff 3a5efd56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>fd5665da d7f83a5e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3a5e69d9 9883d7f8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>d7f88990 53979883</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>c0000492 89905397</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>9fddcc59 04928990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>3731beb2 cc590492</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>7afbe7ed beb2cc59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>7759bb15 7e7deb5b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>fb6445c0 bb157e7d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6f7f1115 45c0bb15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>65a7dea1 111545c0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>45c0e0f9 bb141115</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>11153913 a523bb14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>bb148ee6 281da523</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>a523bfe2 35ee281d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>281d0d84 1adc35ee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>35ee6f1 25871adc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1adc60ee d3002587</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2587cae2 7a12d300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ciphertext output: 2587cae27a12d300
B. Key Exchange Algorithm (KEA)

\[ p = 9d4c6e6d 42ea91c8 28d67d49 94a9f01b 8e5b5b73 0d0faae7 bd569dd1 914e3ad4 759c8053 31eda145 9fb56be8 a8e4e376 652a82b2 76e82acd 63f5b78d 0b75a03e b34d397d be7b3740 8f72136a cb0879fe 61c718a3 7f5154b5 078a7649 fb3d4f64 c481e010 62c5241f 229fa580 423368dd 51090dbf 25351f0c 5f00de05 b92ba6a9 \]

\[ q = 97ad85fd 2b371ed0 69818ab3 c6ee8773 d9db029d \]

\[ g = 595d3443 ec897c82 51e5fa9d 02ab8b75 c0fc57b0 969f880d a366a100 01912a01 96bcb81c 41ac8485 031ac598 b5f481ea 2726b719 d8d9915a 61059734 72386c0a 6a2c732c d67003d4 1f54bf28 d12d692d e2fa05f5 5e898e2c 20bb8a26 02db1ba0 7de672e3 b96d9ac2 9a188450 63d918c3 2ed71266 b783311a 0a8d08ac 487bea44 \]

\[ ra = 6201dd56 237c228a 3f54bc7e 794bdff32 41c67ea6 \]

\[ xa = 62319a4c 7dc14518 0ab3322c 59e2b600 2738e949 \]

\[ Ya = 2d29ec0d 2e3497a6 7222d8de bc286131 d149f458 1b3e586d 0151024c 02e8b23d a09a430e 2ca5e51a 4b2d7725 62316ed4 2804d226 788284ed 655cf546 10d38f66 fab1a0a2 e2d3c661 4401901d 9758d566 722aff1f 734b2a0d d2b67f13 00ce45f6 00968ca7 91a87678 67363d7d 49ee74a2 8dc349d9 fdfdb96b 01f0fc1f 0690ec96 \]

\[ xb = 63decdaa 4487e7b1 31d4ff45 1cfb0e39 446b9b3d \]

\[ rb = 52bfaf7d 2f1cf0f8 0ff6d9a5 15fb7483 167eb0e7 \]

\[ Yb = 7730d4bb f3a2efdb 218e7041 3e861020 14cec06c 205f5419 293b65c6 9a971e54 55eb79a0 bdb90ab2 14c5240e de6cfd55 8c7c19c5 269d57df \]

\[ f60b61c1 db2f6648 64bee519 87f27003 4bc390ad 7316f209 5e42608c 3d7987f9 649f4b7f 6887633e b574b39c c73df899 51fc1bd6 d3889d4 \]

Computed by A:

\[ Ra = 97c1fd8a 69fc8f34 a74c7ec3 c1ab176a b91fa0ea d0e6b097 06a0e7a1 fbf80da6 67032ea4 798082b8 caea827b 4f604b71 e6c24469 211363ea 4bd2122f 4aa6ab9b 4857ff06 9db03701 2b289057 b4855e70 f8f7ac4f 92f4a1e7 6c2a5c82 781ee611 1c1fbd7f a6eb9dc3 59a8fca0 b632ef3a 2af82e52 c0a7f6a6 d2a9f61c 7c67f418 \]

Computed by B:

\[ Rb = 91f61808 3f03d5b 6be53bfe 6e0bf3cb 9d8a8beef199334 b398708b b0c848da 860f0f27 62cc9e48 e496f8fc 94945538 cf6f1719 57ee4f1 \]

\[ e2eca2ba ddb340da f406e636 bbc6366e 4658f0f0 1a41cbe9 5adb4086 42d03ec 4e85920c 8e7530bd e2b78cb8 7c9ae364 31de337c d2e8af29 d8412932 8550dd8c f33e03c2 1a5056a0 \]
Results for user A:
uab=1585d5ba c06b963d 6ef5a60e 5c40220b 76fe0528 660be31a c496d1cb
0883ba8e 5a0331e9 ce3fe382 f47a353c edc6896d fdb4c0b5 67aafdf2
4ba0ff6f fc0a428 fcb07a3d bf6fb8e6 22c5c4a7 7c9bb9be 882da3f5
4cc57980 c174352f 13434623 ce3df2d4 14a9efb 7a90ef8e 4ab282d5
e76e70a5 55dbb38 27c297f9 08ea2c8

tab=8032eb2c b67534a9 c5faf6be a1eb6e3f 0e0d3f48 c86be2a40 8f807e66
8622b9f3 87e0f50f a5868bf5 29ff008d 3a655e9c 4366bad4 ae4190ce
bc3ae56f 34bf70b6 3ca021d3 563005db bc7e62bb ccc9127a 3603b00
be8f3e9b f46f5b3b 86c4a761 4b43adfe 7282efe4 f9c146b7 1e9f89e6
2bd3c7ed 7d127719 ebf0e0f8 79e0d0d9
w=95b8c6e7 76e0caee 34f0099 cfe2b9f0d 550b4471 2e77c55b 54175031
8ea67481 e1e426f9 73c6678e 1e7935ca 289be80a 411b78b8 15ec8e41
07dbe4de 60cf14df 39509c0d 159fbee69 df442d03 4964cc47 be3163f6
0b55481c b5e02a67 9a07e85 198a0e0 872cd0e0 7451a69f 69520c4c
13423827 d2ed3252 13b37897 82caf9al
vl=95b8c6e7 76e0caee 34f0099 e482b7e7 9ff88xxx
v2=XORpad=e7496e99 e4628b7e7 9ff88xxx

Key for user A= 740839de e833add4 6b41xxx

Results for user B:
tab=8032eb2c b67534a9 c5faf6be a1eb6e3f 0e0d3f48 c86be2a40 8f807e66
8622b9f3 87e0f50f a5868bf5 29ff008d 3a655e9c 4366bad4 ae4190ce
bc3ae56f 34bf70b6 3ca021d3 563005db bc7e62bb ccc9127a 3603b00
be8f3e9b f46f5b3b 86c4a761 4b43adfe 7282efe4 f9c146b7 1e9f89e6
2bd3c7ed 7d127719 ebf0e0f8 79e0d0d9
uab=1585d5ba c06b963d 6ef5a60e 5c40220b 76fe0528 660be31a c496d1cb
0883ba8e 5a0331e9 ce3fe382 f47a353c edc6896d fdb4c0b5 67aafdf2
4ba0ff6f fc0a428 fcb07a3d bf6fb8e6 22c5c4a7 7c9bb9be 882da3f5
4cc57980 c174352f 13434623 ce3df2d4 14a9efb 7a90ef8e 4ab282d5
e76e70a5 55dbb38 27c297f9 08ea2c8
w=95b8c6e7 76e0caee 34f0099 cfe2b9f0d 550b4471 2e77c55b 54175031
8ea67481 e1e426f9 73c6678e 1e7935ca 289be80a 411b78b8 15ec8e41
07dbe4de 60cf14df 39509c0d 159fbee69 df442d03 4964cc47 be3163f6
0b55481c b5e02a67 9a07e85 198a0e0 872cd0e0 7451a69f 69520c4c
13423827 d2ed3252 13b37897 82caf9al
vl=95b8c6e7 76e0caee 34f0099 e482b7e7 9ff88xxx
v2=XORpad=e7496e99 e4628b7e7 9ff88xxx

Key for user B= 740839de e833add4 6b41xxx
C. KEA Exchange for E-Mail

\[ p = 9d4c66e6d 42ea91c8 28d67d49 94a90f1b 8e5b5b73 0d0faae7 bd569dd1914e3ad4 759c8053 31eda145 9fb56be8 a8de4736 652a82b2 76e82acd63f5b78d 0b75a03e b34d397d be7b3740 8f72136a cb0879fe 61c718a37f5154b5 078a7649 fb3d4fb4 c481e010 62c5241f 229fa580 423368dd51090dbf 25351f0c 5800de05 b92ba6a9 q = 97ad85fd 2b371ed0 69818ab3 c6ee8773 d9db029d g = 595d3443 ec897c82 51e5f9a9 02ab8b75 c0ff57b0 969f880d a366a10019120a01 96bcb81c 41ac8485 031ac598 b5481eae 2726b719 d8d9915a61059734 72386c0a 6a2c732c d6700d34 1f54fb28 d12d692d e2fa05f55e898c2e 20bb8a26 02db1ba0 7de672e3 b969ac2c 9a188450 63d918c32ed71266 b783311a 0a8d08ac 4878ea44

\[ ra = 6201dd56 237c228a 3f54bc7e 794bdf32 41c67ea6 xa = 623194ac 7de14518 0ab3d22c 59e2b600 2781e494 Ya = 2d29cc02 0e3497a6 7222d8de bc386131 d149f458 1b3e586d 0151024c 0e8b23d a09a430e 2ca5ed1a 4b2d7725 62316e4d 2804d226 788284ed 655cf546 10d3bf66 fabla0a2 e2d3c661 4401901d 9758d566 722aff1f 734b2adb d2b67f13 00ce455f 00968ca7 91a87678 673637d9 49ee74a2 834c39d9 fdfdb96b 01f0fc1f 0690ec96

\[ xb = 63decda0 4487eb71 31d0ff4f 1cfbae39 446b9b3d Yb = 7730d4bb f3a2efdb 210e7041 3e8e6102 14c806c6 205f5419 293b6c629a971e54 55eb79a0 bdb90ab2 14c5240e de6cfdd5 8c7c19c5 269d57df 60b61c1 db2ff648 64bee519 87f27003 4bc390ad 73168209 5e42608c 3d7987f9 649fbf71 6887633e b574b39c c73df899 51fc1bd6 d3889d48 fe2244b8 29af3d05 06ab9221 ba562c07

Computed by A:
\[ Ra = 9c7c1fda8a 69fc8f34 a74c7ec3 c1ab176a b91fa0ea d0e6b097 06a07a1 fb8f30a6 67032b6e 798082b8 caea2b72 4f60b71 e6c24469 211363ea 4bd2122f 4aa6af9b 4857ff06 9db03701 2b289057 b4855e70 f8f7ac4f 92fa1fe7 6c2a5c82 781e611 1cfbdf7 a6eb9dc3 59a8fca0 b632ef3a 2af82e52 c0a7f6a6 a2c961ea fc67f418

Results for user A:
\[ utab = 8032eb2c b67534a9 c5f6af6be a1eb6ef1 de0d3f48 c68be240 8f807e66 8622b9f3 87e0f50f a5868bf5 29ff008d 3ad55e9c 4366bad4 ae4190ce bc3ae56f 34bf70b6 3ca021dd 563005db bc7e62bb ccc9127a 3603bf00 be8f0e9b f46bf538 86d4a761 4b43adfe 7282eefe4 f9c146b7 1ef89d6 2bd3c7ed 7d1d7217 ebf0e0f8 79e0d0d9 uab = 17087175 9f1d6d8f b0a0c05e 0ee49abd 49586033 93a7df3 3d99bc61 68ad318a 7cfc81fa8 74f4eb04 4433abe0 6423eb2f 1ebbb3cd 33067152 242d7cf8 987f208d cfdf3797 6398ccdd 6a0bdc1b 2bd6734 35dedacc9 06bd6d71 a4516738 b91f2a52 689a2d60 802de96d 150fe661 469a2643
18c8d8f5 9ec040ea c623c51a 91d861d1
w=973b5ca2 558c1469 769bb71c b0d009af 27659f7c 5c166033 cd1a3ac7
eecfеб7е 04d914b8 1a7b76f9 6e32ac6d 9ef949cb 6221f7af e1480220
e0686267 cd3e9144 0c7f5974 b9c8d2b1 268a3ed6 f8c679ae 6be29bc9
c54d3c0d 98bd5c71 3fe3d1b3 b3dddb5e f2b0d952 0ed12d18 6539b019
d49ca0e3 1bd2b804 b214a613 0bb932aa
vl=973b5ca2 558c1469 769bbxxxx
v2=b71cb0d0 09af2765 9f7cxxxx
vl XOR pad = e5caf4dc c70e55f1 dd90xxxx

Key for user A = 97fd1c6b d86bc439 115bxxxx

Results for user B:
tab=8032eb2c b67534a9 c5af6be a1eb6ef1 de0d3f48 c86be240 8f807e66
8622b9f3 87e0f50f a5868bf5 29ff008d 3ad55e9c 4366bad4 ae4190ce
bc3ae56f 34bf70b6 3ca021dd 563005db bc7e62bb ccc9127a 3603bf00
deb8cfe9b f46bf3b8 88c4a6b1 4b43ade7 /282ete4 f9c146b7 1e9t89d6
2bd3c7ed 7d127719 ebf0e0f8 79e0d0d9
uab=17087175 9f16dfbf b0a0c05e 0e49abd 49586033 93aa7df3 3d99bc61
68ad318a 7cf81fa8 74f4eb04 4433abe0 6423eb2f lebb3cdb 33067152
242d7cf8 987f208d cdf5797 6398c6d5 6a0bdc1b 2bf6734 35edacc9
06b6d71a 4516738 b91f2a52 689a2d60 802de96d 150fe661 469a2643
18c8d8f5 9ec040ea c623c51a 91d861d1
w=973b5ca2 558c1469 769bb71c b0d009af 27659f7c 5c166033 cd1a3ac7
eecfеб7е 04d914b8 1a7b76f9 6e32ac6d 9ef949cb 6221f7af e1480220
e0686267 cd3e9144 0c7f5974 b9c8d2b1 268a3ed6 f8c679ae 6be29bc9
c54d3c0d 98bd5c71 3fe3d1b3 b3dddb5e f2b0d952 0ed12d18 6539b019
d49ca0e3 1bd2b804 b214a613 0bb932aa
vl=973b5ca2 558c1469 769bbxxxx
v2=b71cb0d0 09af2765 9f7cxxxx
vl XOR pad = e5caf4dc c70e55f1 dd90xxxx

Key for user B = 97fd1c6b d86bc439 115bxxxx
IV. References:

1. US DEPARTMENT OF COMMERCE Technology Administration/National Institute of Standards and Technology, DES MODES OF OPERATION, FIPS PUB 81, 2 December 1980.