Information Technology Laboratory

U.S Department
of Commerce

National Institute
of Standards and
Technology

NIST Special Publication 800-20

Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS):

Requirements and Procedures

Sharon S. Keller

Computer Science

and Technology

NIST Special Publication 800-20

Modes of Operation Validation System for the Triple Data Encryption Algorithm

(TMOVS):

Requirements and Procedures

Sharon Keller

Security Technology Group

Computer Security Division

National Institute of Standards and Technology

Gaithersburg, MD 20899-8930

Original Date: October 1999

Revision Date: April 2000

Revision Date: March 2012

U.S. DEPARTMENT OF COMMERCE

William M. Daley, Secretary

Technology Administration

Gary R. Bachula, Acting Under Secretary for Technology

National Institute of Standards and Technology

Raymond G. Kammer, Director

Revision History

March 2012

Section	Revision
5.7.2, Table 62	Within the "Perform Triple DES" pseudocode for OFB-I mode of operation, change the ELSE command to $Ij=0_{j-3}$. The subscript was j-2 which was incorrect.
5.7.2, Item 2.b.2	Within 2.b.2. change (j-2) to (j-3). This change is made to correct an incorrect subscript. (Same change as above.)
Revision History April 2000	Changed page numbers to section numbers

Revision History

April 2000

Section	Revision
OVERALL	Rename the Modes Tests to the Monte Carlo Tests to coincide with all other documents.
OVERALL	Represent “through” as “..” not “-“
OVERALL	Draw a box around the Triple DES operations in the pseudocode to indicate what code is from the Triple DES standard and what code is part of the Validation test.
OVERALL	Replace subscript numbers with subscript variable names. For example, C_{9999} is replaced with C_j.
2	Make reference to the three different keying options specified in FIPS PUB 46-3.
4.3.2	Input Type 2 – remove “…represented as a 16 character ASCII…”, replace with “…represented as an ASCII…”
4.3.5	Input Type 5 – same as above
4.3.8	Input Type 8 – same as above
4.3.13	Input Type 13 – same as above
4.3.15	Input Type 15 – same as above
4.3.18	Input Type 18 – same as above
4.3.21	Input Type 21 – same as above
4.3.22	Input Type 22 – for TEXT1, TEXT2, and TEXT3 remove “…1 to 64 binary…” replace with “…64 binary…”
4.3.24	Input Type 24 – same as above
4.4.2	Output Type 2 – for DATA and RESULT remove “…is a 16 character hexidecimal…”, replace with “…is 1 – 64 binary bits represented as an ASCII hexidecimal…”

4.4.3	Output Type 3 – same as above
4.4.6	Output Type 6 – same as above
4.4.7	Output Type 7 – same as above
4.4.8	Output Type 8 – same as above
5.1.1.1 Table 1	In pseudocode, Send statement – the subscript on C should be lowercase i.
5.1.1.6 Table 6	Replace $P_0 = C_{9999}$ with $P_0 = C_j$
5.2.1.1 Table 13	In pseudocode, the subscript should be lowercase i
5.2.1.6, 2.f	Switch 2) and 3) to make the text coincide with the pseudocode
5.3.1.1 Table 25	In pseudocode, Send statement – $I1_i$, $I2_i$, and $I3_i$ should be sent instead of $P1_i$, $P2_i$, and $P3_i$
5.3.1.1 Table 25	Clock Cycle T4, 2) – The subscript on TEMP3 should be 1.
5.3.1.1	b. – Replace $P1_i$, $P2_i$, $P3_i$ with $I1_i$, $I2_i$, $I3_i$.
5.3.1.6 Table 30	Replace Ck_{9998} with Ck_{j-1} and replace Ck_{9999} with Ck_j
5.3.1.6	Clock Cycle T4, a) – replace DEA_2 with DEA_3 b) – replace DEA_3 with DEA_2 replace $KEY3_i$ with $KEY2_i$ Swap a) and b) to make the text coincide with the pseudocode.
5.3.1.6	In f), g) and h) – Replace subscript 9999 with j Replace subscript 9998 with j-1 Replace subscript 9997 with j-2
5.3.2.6 Table 36	In the pseudocode, add “FOR k = 1 to 3”

5.3.2.6 Table 36	Replace subscript 9999 with j
5.3.2.6	In Clock Cycle T1– b), Clock Cycle T2: b), and Clock Cycle T3: b): The j in I1j, I2j, and I3j, respectively, should be a subscript: $I1_j$, $I2_j$, $I3_j$.
5.3.2.6	In 2b) – Add comma after $P3_j$.
5.3.2.6	In f. – Add comment "Note j=9999."
5.3.2.6	In f), g) and h) – Replace subscript 9999 with j Replace subscript 9998 with j-1 Replace subscript 9997 with j-2
5.4.2.1 Table 42	In pseudocode, replace the subscript 9999 with j
5.4.2.1	In 2) and 3) – Replace the subscript 9999 with j
5.4.2.1	In 3) – Add subscripts to I and k-bit C so they read I_j and C_j
5.4.2.2 Table 43	In pseudocode, replace the subscript 9999 with j.
5.4.2.2	In 2) and 3) – Replace the subscript 9999 with j.
5.4.2.2	In 2) – Add subscripts to I and k-bit C so they read I_j and C_j
5.5.2.1 Table 49	In pseudocode, the code pertaining to the Triple DES algorithm does not coincide with the Triple DES standard. The subscript on RESULT should be j-3, i.e., $Ij = RM^{(64\text{-}K)} (I(j\text{-}1)) \parallel \text{K-bit } RESULT_{j-2}$ Should be $Ij = RM^{(64\text{-}K)} (I(j\text{-}1)) \parallel \text{K-bit } RESULT_{j-3}$
5.5.2.2	Add a separate Monte Carlo Test for the Encryption and Decryption processes of the CFB-P Mode of operation.
5.5.2.2 Table 50	In pseudocode, replace I9999 with Ij.

5.5.2.2	In b2), replace subscript j-2 with j-3.
5.5.2.2	In 3) and 4) – Replace the subscript 9999 with j. Add statement "…where j = 9999."
5.6.2 Table 56	At the end of the external loop, where new values are generated for the keys, the text and the input block, it was unclear in the generation of the new text which value of text was being referred to in this statement: $TEXT_0 = TEXT_0 \oplus I_j$ $TEXT_0$ is referring to the initial text of the INTERNAL loop. Therefore, add the following code to make this clear: In pseudocode, add statement "INITTEXT = $TEXT_0$" in the external loop before the internal loop. This will capture the initial text used for each internal loop. Also, modify the statement at the end of the external loop: $TEXT_0 = TEXT_0 \oplus I_j$ To read $TEXT_0 = INITTEXT \oplus I_j$
5.6.2 Table 56	In the pseudocode, replace the subscript 9999 with j.
5.6.2	In the text, add after b: "c. Assign the value of $TEXT_0$ to INITTEXT. This will contain the initial text from every j = 0 loop."
5.6.2	Reletter text.
5.6.2	In g 1), 2), and 3) – Replace subscript 9999 with j. Replace subscript 9998 with j-1. Replace subscript 9997 with j-2.
5.6.2	In g 2) – This should read: Assign a new value to the $TEXT_0$. The $TEXT_0$ should be assigned the value of **INITTEXT** exclusive-ORed with I_j.
5.6.2	In g NOTE – P should be replaced with TEXT.

5.7.2 Table 62	In the pseudocode, Replace the subscript 9999 with j.
5.7.2	Replace subscript 9999 with j.
Table A.5	Replace column headings PLAINTEXT1, 2, and 3 with INPUTBLOCK1, 2, and 3.
Table A.7	Replace column headings PLAINTEXT1, 2, and 3 with one column labeled CIPHERTEXTS. Replace column headings CIPHERTEXT1, 2, and 3 with PLAINTEXT 1, 2, and 3.
Table A.8	Start ROUND numbers with 0.
Table A.8	Replace colum headings PLAINTEXT1, 2, and 3 with one column labeled PLAINTEXTS.
Table A.9	Replace column headings PLAINTEXT1, 2, and 3 with PLAINTEXT1 $\oplus$ IV1, PLAINTEXT2 $\oplus$ IV2, and PLAINTEXT3 $\oplus$ IV3, respectively.
Table A.10	Replace column headings PLAINTEXT1, 2, and 3 with IV1, 2, and 3.

TABLE OF CONTENTS

FIGURES

TABLES

ABSTRACT

The National Institute of Standards and Technology (NIST) Triple Data Encryption Algorithm (TDEA) Modes of Operation Validation System (TMOVS) specifies the procedures involved in validating implementations of the Triple DES algorithm in FIPS PUB 46-3 *Data Encryption Standard (DES)* (and ANSI X9.52 – 1998). The TMOVS is designed to perform automated testing on Implementations Under Test (IUTs). This publication provides brief overviews of the Triple DES algorithm and introduces the basic design and configuration of the TMOVS. Included in this overview are the specifications for the two categories of tests that make up the TMOVS, i.e., the Known Answer tests and the Monte Carlo tests. The requirements and administrative procedures to be followed by those seeking formal NIST validation of an implementation of the Triple DES algorithm are presented. The requirements described include the specific protocols for communication between the IUT and the TMOVS, the types of tests which the IUT must pass for formal NIST validation, and general instructions for accessing and interfacing with the TMOVS. An appendix with tables of values and results for the Triple DES Known Answer tests is also provided.

Key words: automated testing, computer security, cryptographic algorithms, cryptography, Triple Data Encryption Algorithm (TDEA), Triple Data Encryption Standard (TDES), Federal Information Processing Standard (FIPS), NVLAP, secret key cryptography, validation.

1. Introduction

1.1 Background

The publication specifies the tests required to validate Implementations Under Test (IUTs) for conformance to the Triple DES algorithm (TDEA) as specified in ANSI X9.52, *Triple Data Encryption Algorithm Modes of Operation*. When applied to IUTs that implement the TDEA, the TDEA Modes of Operation Validation System (TMOVS) provides testing to determine the correctness of the algorithm implementation. This involves both testing the specific components of the algorithm, as well as, exercising the entire algorithm implementation. In addition to determining conformance, the TMOVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the TDEA implementation.

The TMOVS is composed of two types of validation tests, the Known Answer tests and the Monte Carlo tests. The validation tests are based on the standard DES test set and the Monte Carlo test described in Special Publication 800-17, *Modes of Operation Validation System (MOVS): Requirements and Procedures*. By applying the same framework specified in Special Publication 800-17 to TDES, the TMOVS specifies how to validate implementations of the TDEA in software, firmware, hardware, or any combination thereof.

The Known Answer tests are designed to verify the components of the DES algorithm in the IUT (e.g., S boxes, permutation tables,…). The tests exercise each bit of every component of the algorithm implementation. This is accomplished by processing all possible basis vectors through the IUT. To perform the Known Answer tests, the TMOVS supplies known values to the IUT and the

IUT then processes the input through the implemented algorithm. The results produced by the IUT are compared to the expected values.

The Monte Carlo Test is designed to exercise the entire implementation of the TDEA, as opposed to testing only the individual components. The purpose of the Monte Carlo Test is to detect the presence of flaws in the IUT that were not detected with the controlled input of the Known Answer test. The Monte Carlo Test does not guarantee ultimate reliability of the IUT that implements the TDEA (i.e., hardware failure, software corruption, etc.). To perform the Monte Carlo Test, the TMOVS supplies the IUT with pseudorandom values for the initial plaintext, key(s), and, if applicable, initialization vector(s). Using these values, the IUT is exercised through four million DES encryption/decryption iterations. The results are then compared to the expected values.

The successful completion of the tests contained within the TMOVS is required to claim conformance of Triple DES implementations as defined in FIPS PUB 46-3, *Data Encryption Standard (DES)*. Testing for single DES implementations is defined in Special Publication 800-17, *Modes of Operation Validation System (MOVS): Requirements and Procedures*. Testing for the cryptographic module in which Triple DES is implemented is defined in FIPS PUB 140-1, *Security Requirements for Cryptographic Modules.*

1.2 Organization

Section 2 gives a brief overview of the Triple DES algorithm and the five modes of operation allowed by this algorithm as well as the interleaved and pipelined versions of several of these modes. Section 3 provides an overview of the tests that make up the Triple DES Modes of Operation Validation System (TMOVS). Section 4 describes the basic protocol used by the TMOVS. Section 5 provides a detailed explanation of each test required by the TMOVS to validate an IUT of the TDEA. Section 6 outlines the design of the TMOVS. Appendix A provides tables of values for the Known Answer tests for TDEA. These tables include:

— For modes of operation including TECB, TCBC, TCFB, and TOFB:

  Table A.1 - Resulting Ciphertext from the Variable Plaintext Known Answer Test

  Table A.2 - Resulting Ciphertext from the Variable Key Known Answer Test

  Table A.3 - Values to be Used for the Permutation Operation Known Answer Test

  Table A.4 - Values to be Used for the Substitution Tables Known Answer Test

— For the TCBC-I mode of operation:

  Table A.5 - Resulting Ciphertext from the Variable Plaintext Known Answer Test for TCBC-I

  Table A.6 - Resulting Ciphertext from the Inverse Permutation Known Answer Test for TCBC-I

Table A.7 – Resulting Ciphertext from the Initial Permutation Known Answer Test for TCBC-I

Table A.8 - Values to be Used for the Substitution Tables Known Answer Test for TCBC-I

— For the TCFB-P and TOFB-I modes of operation:

Table A.9 - Resulting Ciphertext from the Variable Text Known Answer Test for TCFB-P and TOFB-I

Table A.10 – Values to be Used for the Substitution Tables Known Answer Test for TCFB-P and TOFB-I

— For the TCBC-I, TCFB-P, and TOFB-I modes of operation:

Table A.11 - Resulting Ciphertext from the Variable Key Known Answer Test for TCBC-I, TCFB-P and TOFB-I

Table A.12 – Values to be Used for the Permutation Operation Known Answer Test for TCBC-I, TCFB-P and TOFB-I

1.3 Definition(s)

1.3.1 Basis vector

A vector consisting of a “1” in the i^{th} position and “0” in all of the other positions.

1.3.2 Block

A binary vector. In this document, the input and output of encryption and decryption operation are 64-bit block. The bits are numbered from left to right. The plaintext and ciphertext are segmented to k-bit blocks, k = 1, 8, 64.

1.3.3 Ciphertext

Encrypted (enciphered) data.

1.3.4 Cryptographic boundary

An explicitly defined contiguous perimeter that establishes the physical bounds around the set of hardware, software and firmware which is used to implement the TDEA and the associated cryptographic processes.

1.3.5 Cryptographic key

A parameter that determines the transformation from plaintext to ciphertext and vice versa. (A DEA key is a 64-bit parameter consisting of 56 independent bits and 8 parity bits). Multiple (1, 2 or 3) keys may be used in the Triple Data Encryption Algorithm.

1.3.6 Data Encryption Algorithm

The algorithm specified in FIPS PUB 46-3, *Data Encryption Algorithm (DEA)*.

1.3.7 Decryption

The process of transforming ciphertext into plaintext.

1.3.8 Encryption
The process of transforming plaintext into ciphertext.

1.3.9 Exclusive-OR
The bit-by-bit modulo 2 addition of binary vectors of equal length.

1.3.10 Initialization Vector
A binary vector used as the input to initialize the algorithm for the encryption of a plaintext block sequence to increase security by introducing additional cryptographic variance and to synchronize cryptographic equipment. The initialization vector need not be secret. Some of the Triple Data Encryption Algorithm Modes of Operation require 3 initialization vectors.

1.3.11 Key
See cryptographic key.

1.3.12 Plaintext
Intelligible data that has meaning and can be read or acted upon without the application of decryption. Also known as cleartext.

1.3.13 Self-dual Key

A key with the property that when you encrypt twice with this key, the result is the initial input.

1.3.14 Triple Data Encryption Algorithm
The algorithm specified in FIPS PUB 46-3 –1999, *Data Encryption Algorithm*.

1.4 Symbols (and Acronyms)

1.4.1	C	Ciphertext
1.4.2	C*n*	Block of data representing the Ciphertext *n*
1.4.3	$C^1,\ldots,C^{64}$	Bits of the Ciphertext Block
1.4.4	$D_{KEYx}(Y)$	Decrypt Y with the key KEY_x
1.4.5	DEA	The Data Encryption Algorithm specified in FIPS 46-3
1.4.6	DES	Data Encryption Standard specified in FIPS 46-3
1.4.7	$E_{KEYx}(Y)$	Encrypt Y with the key KEY_x
1.4.8	FIPS PUB	Federal Information Processing Standard Publication
1.4.9	I	Input Block
1.4.10	I*n*	Block of data representing the Input Block *n*
1.4.11	$I^1, \ldots, I^{64}$	Bits of the Input Block

1.4.12 IUT Implementation Under Test

1.4.13 IV Initialization Vector

1.4.14 IV*n* Block of data representing IV *n*

1.4.15 KEY*n* Block of data representing KEY *n*

1.4.16 NIST National Institute of Standards and Technology

1.4.17 O Output Block

1.4.18 $O^1,\ldots,O^{64}$ Bits of the Output Block

1.4.19 O*n* Block of data representing Output Block *n*

1.4.20 P Plaintext

1.4.21 $P^1,\ldots,P^{64}$ Bits of the Plaintext Block

1.4.22 P*n* Block of data representing Plaintext *n*

1.4.23 RESULT*n* Block of data representing Plaintext *n*, if encryption state, or Ciphertext *n*, if decryption state

1.4.24 TCBC TDEA Cipher Block Chaining Mode of Operation

1.4.25 TCBC-I TDEA Cipher Block Chaining Mode of Operation - Interleaved

1.4.26 TCFB TDEA Cipher Feedback Mode of Operation

1.4.27 TCFB-P TEA Cipher Feedback Mode of Operation - Pipelined

1.4.28 TDEA Triple Data Encryption Algorithm specified in FIPS 46-3

1.4.29 TDES Triple Data Encryption Standard specified in FIPS 46-3

1.4.30 TECB TDEA Electronic Codebook Mode of Operation

1.4.31 TEXT*n* Block of data representing Plaintext *n*, if encryption state, or Ciphertext *n*, if decryption state

1.4.32 TMOVS TDEA Modes of Operation Validation System

1.4.33 TOFB TDEA Output Feedback Mode of Operation

1.4.34 TOFB-I TDEA Output Feedback Mode of Operation – Interleaved

1.4.35 $VARIABLE_n$ Block of data representing the value of *VARIABLE* for the n^{th} iteration

1.4.36 $X \oplus Y$ Bit-wise inclusive-or of two bit-strings X and Y of the same bit length.

2. Triple Data Encryption Algorithm (TDEA)

FIPS PUB 46-3 - 1999, *Data Encryption Standard (DES)*, (and ANSI X9.52 – 1998) specifies the Triple Data Encryption Algorithm (TDEA) modes of operation for the enhanced cryptographic protection of digital data. The modes of operation for TDEA provide a means of extending the effective key space of the Data Encryption Algorithm (DEA). Certain modes also provide increased protection against more sophisticated cryptanalytic attacks. FIPS PUB 46-3 - 1999 enhances the basic level of cryptographic protection of digital data provided by DEA, thus extending the useful lifetime of this technology.

The TDEA consists of three components – the DES algorithm (DEA), multiple keys, and initialization vector(s). The DEA is called three times in the TDEA. The TDEA utilizes one to three keys and, depending on the mode of operation being implemented, zero, one or three initialization vectors (IVs).

The basic processing involved in the TDEA is as follows: An input block is read into the first DEA (DEA1) and encrypted using the first key (KEY1). The output produced from this stage is read directly into the second DEA (DEA2) and decrypted using the second key (KEY2). The output produced by the second stage is directly read into the third DEA (DEA3) and encrypted using the third key (KEY3). The resultant output block is used, according to the mode implemented, in the calculation of the ciphertext. Note that the output for the intermediate DEA stages is never revealed outside the cryptographic boundary.

Three different keying options are allowed by the TDEA. The first option specifies that all the keys are independent, i.e., KEY1, KEY2, and KEY3 are independent. This is referred to as Keying Option 1 in FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998). It will be referred to as 3-key TDES in this document. The second option specifies that KEY1 and KEY2 are independent and KEY3 is equal to KEY1, i.e., KEY1 and KEY2 are independent, KEY3 = KEY1. This is referred to as Keying Option 2 in FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) and will be referred to as 2-key TDES in this document. And the third option specifies that KEY1, KEY2 and KEY3 are equal, i.e., KEY1=KEY2=KEY3. This is referred to as Keying Option 3 in FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998 and will be referred to as 1-key TDES in this document. 1-key TDES is equivalent to single DES.

The initialization vector (IV) must meet the following attributes as specified by the TDEA:

- For TECB, no IV is used.
- For all modes using an IV, the IV may be public information.
- For TOFB and TOFB-I, the IV should never be a constant.
- If the mode of operation implemented requires one IV, it may be generated in one of two ways:
  - Randomly or Pseudo-randomly

- As a counter

- If the mode of operation implemented requires three IVs, they should be generated as follows:
    - IV1 should be generated in the same manner as one IV (described above).
    - IV2 = IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555.
    - IV3= IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA.

A thorough explanation of the processing involved in the four modes of operation supplied by TDEA, as well as the new message-interleaved and pipelined versions of these modes can be found in FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998). A brief explanation of each mode is found below.

2.1 TDEA Electronic Codebook (TECB) Mode

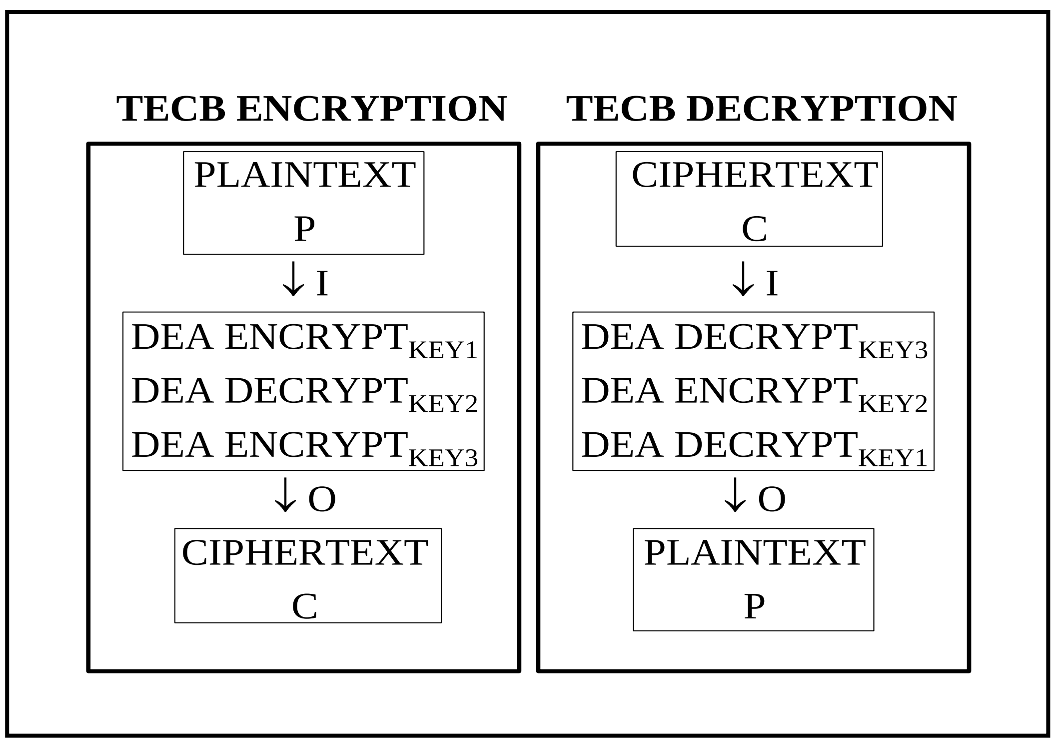

Figure 1 TDEA Electronic Codebook (TECB) Mode

The TDEA Electronic Codebook (TECB) mode is shown in Figure 1. In TECB encryption, a 64-bit plaintext data block (P) is used directly as the input block (I). The input block is processed through the first DEA (DEA1) in the encrypt state using KEY1. The output of this process is fed directly to the input of the second DEA (DEA2) where DES is performed in the decrypt state using KEY2. The output of this process is fed directly to the input of the third DEA (DEA3) where DES is performed in the encrypt state using KEY3. The resultant 64-bit output block (O) is used directly as ciphertext (C).

In TECB decryption, a 64-bit ciphertext block (C) is used directly as the input block (I). The keying sequence is reversed from the encrypt process. The input block is processed through DEA3 in the decrypt state using KEY3. The output of this process is fed directly to the input of DEA2, where DES is performed in the encrypt state using KEY2, and the result is directly fed to the input of DEA1, where DES is performed in the decrypt state using KEY1. The resultant 64-bit output block (O) produces the plaintext (P).

2.2 TDEA Cipher Block Chaining (TCBC) Mode

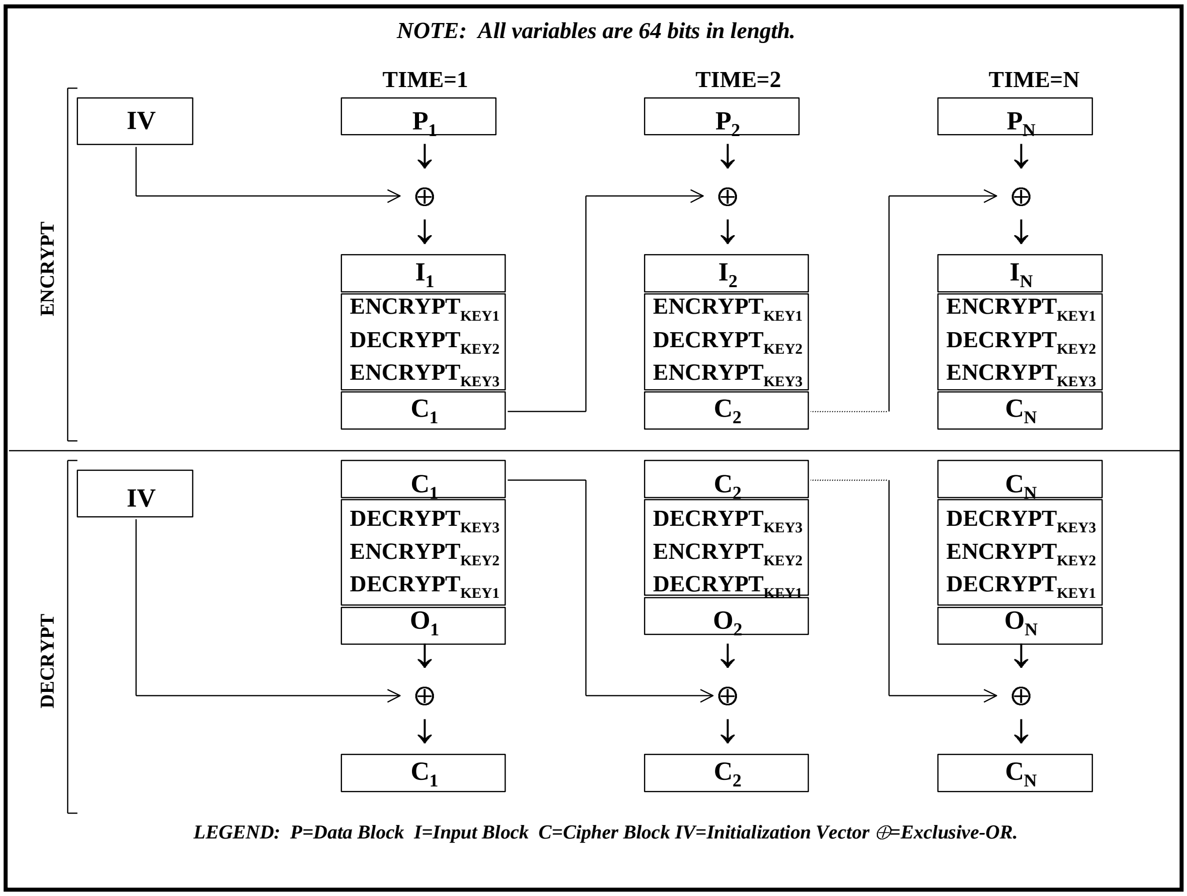

Figure 2 TDEA Cipher Block Chaining (TCBC) Mode

As shown in the upper half of Figure 2, the TDEA Cipher Block Chaining (TCBC) mode begins processing by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I_1) is formed by exclusive-ORing the first plaintext data block (P_1) with a 64-bit initialization vector IV, i.e., ($I_1 = IV \oplus P_1$). The input block is processed through DEA1 in the encrypt state using KEY1. The output of this process is fed directly to the input of DEA2, which performs DES in the decrypt state using KEY2. The output of this process is fed directly to the input of DEA3, which performs DES in the encrypt state using KEY3. The resultant 64-bit output block (O_1) is used directly as ciphertext (C_1), i.e., ($C_1 = O_1$). This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, i.e., (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block. This encryption process continues to "chain" successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted. If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

In TCBC decryption (see the lower half of Figure 2), the first ciphertext block (C_1) is used directly as the input block (I_1). The keying sequence is reversed from the encrypt process. The input block is processed through DEA3 in the decrypt state using KEY3. The output of this process is fed directly to the input of DEA2, where DES is processed in the encrypt state using KEY2. This resulting value is directly fed to the input of DEA1, where DES is processed in the decrypt state using KEY1. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block, i.e., ($P_1 = O_1 \oplus IV$). The second ciphertext block is then used as the next input block and is processed through the TDEA as shown above. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block, i.e., ($P_2 = O_2 \oplus C_1$). (NOTE – P_2 and O_2 refer to the second block.) The TCBC decryption process continues in this manner until the last complete ciphertext block has been decrypted. Ciphertext representing a partial data block must be decrypted in a manner as specified for the application.

2.3 TDEA Cipher Block Chaining - Interleaved (TCBC-I) Mode

Both the encryption and decryption processes of the TDEA Cipher Block Chaining – Interleaved (TCBC-I) mode of operation require 3 IVs, an *n* block message interleaved into three sub-texts, and 3 keys. The IVs, denoted IV1, IV2, and IV3, are generated based on the specifications mentioned in Section 2. For both the encryption and decryption processes of the TCBC-I mode of operation, these values are assigned to the initial values of C1, C2, and C3.

Prior to commencing both the TCBC-I encryption and decryption processes, the TEXT (which refers to plaintext, P, if encrypting and ciphertext, C, if decrypting) is interleaved into three sub-texts. This is accomplished by taking an *n*-block TEXT and subdividing it into groups consisting of three blocks each accordingly:

$$\text{TEXT}=(\text{TEXT}_1, \text{TEXT}_2,..., \text{TEXT}_n) =(\text{TEXT}_{1,1}, \text{TEXT}_{2,1}, \text{TEXT}_{3,1}, \text{TEXT}_{1,2}, \text{TEXT}_{2,2}, \text{TEXT}_{3,2}, \text{TEXT}_{1,3}, \text{TEXT}_{2,3}, \text{TEXT}_{3,3},...,\text{TEXT}_{1,i}, \text{TEXT}_{2,i}, \text{TEXT}_{3,i}), \text{ where } i = n/3.$$

Then the TEXT is decimated into three sub-texts:

$$\text{TEXT}^1 = \text{TEXT}_{1,1}, \text{TEXT}_{1,2}, \text{TEXT}_{1,3},...,\text{TEXT}_{1,n1};$$

$$\text{TEXT}^2 = \text{TEXT}_{2,1}, \text{TEXT}_{2,2}, \text{TEXT}_{2,3},...,\text{TEXT}_{2,n2};$$

$$\text{TEXT}^3 = \text{TEXT}_{3,1}, \text{TEXT}_{3,2}, \text{TEXT}_{3,3},...,\text{TEXT}_{3,n3};$$

where

- if n mod 3 = 0, then $n1 = n2 = n3 = n/3$; The last block in TEXT is $\text{TEXT}_{3,n3}$.

- if n mod 3 = 1, then $n1 = (n+2)/3, n2 = n3 = (n-1)/3$; The last block in TEXT is $\text{TEXT}_{1,n1}$.

- if n mod 3 = 2, then $n1 = n2 = (n+1)/3$, $n3 = (n-2)/3$; The last block in TEXT is $TEXT_{2,n2}$.

The TCBC-I mode of operation is intended for systems equipped with multiple DEA processors. Each of the DEA processors used in both the encryption and decryption processes utilize the same processing as that used in the TCBC mode of operation for all three sub-texts. The DEA processors operate simultaneously.

During the encryption process of the TCBC-I mode of operation, for j=1 to 3 and i=1 to n, the $P_{j,i}$ is exclusive-ORed with the $C_{j,i-1}$. This value is processed through DEA1 in the encrypt state using KEY1. The output of this process is fed directly to the input of DEA2, which performs DES in the decrypt state using KEY2. The output of this process is fed directly to the input of DEA3, which performs DES in the encrypt state using KEY3. The resultant 64-bit output block is used directly as the $C_{j,i}$.

With three DEA functional blocks, DEA1, DEA2, and DEA3, which are simultaneously clocked, the encryption of three sub-plaintexts can be interleaved.

In pseudocode terms,

For j = 1 to 3 {
$C_{j,0} = IV_j$
For i = 1 to n_j {
$C_{j,i} = E_{KEY3}(D_{KEY2}(E_{KEY1}(P_{j,i} \oplus C_{j,i-1})))$
Output $C_{j,i}$
}
}

During the decryption process of the TCBC-I mode of operation, for j=1 to 3 and i=1 to n, the $C_{j,i}$ is processed through DEA3 in the decrypt state using KEY3. The output of this process is fed directly to the input of DEA2, which performs DES in the encrypt state using KEY2. The output of this process is fed directly to the input of DEA1, which performs DES in the decrypt state using KEY1. The resultant 64-bit output block is exclusive-ORed with the $C_{j,i-1}$. This value is used directly as the $P_{j,i}$.

Because there are three DEA functional blocks, DEA_1, DEA_2, and DEA_3, which are simultaneously clocked, the decryption of three sub-ciphertexts can be interleaved.

In terms of pseudocode,

For j = 1 to 3 {
$C_{j,0} = IV_j$
For i = 1 to n_j {
$P_{j,i} = D_{KEY1}(E_{KEY2}(D_{KEY3}(C_{j,i}))) \oplus C_{j,i-1}$
Output $P_{j,i}$
}
}

2.4 TDEA Cipher Feedback (TCFB) Mode

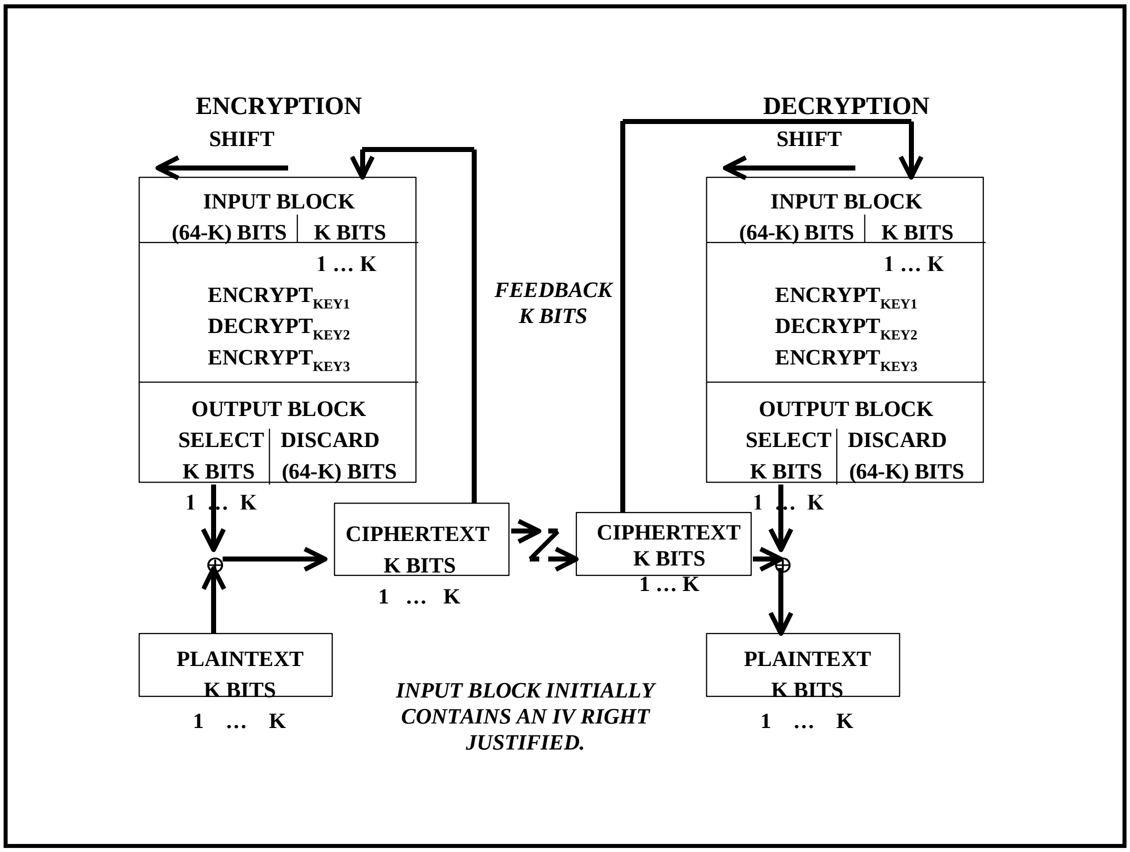

Figure 3 TDEA Cipher Feedback (TCFB) Mode

The TDEA Cipher Feedback (TCFB) mode is shown in Figure 3. A message to be encrypted is divided into K-bit data units, where K may equal 1 through 64 inclusively (K = 1,2,...,64). In both the TCFB encrypt and decrypt operations, an initialization vector (IV) of length 64 is used. The input block is assigned the value of the IV, i.e., (I = IV). The input block is processed through DEA1 in the encrypt state using KEY1. The output of this process is fed directly to the input of DEA2, where DES is performed in the decrypt state using KEY2. The output of this process is fed directly to the input of DEA3, where DES is performed in the encrypt state using KEY3. During encryption, ciphertext is produced by exclusive-ORing a K-bit plaintext data unit with the most significant K bits of the output block, i.e.,$(C^1,C^2,...,C^K) = (P^1 \oplus O^1, P^2 \oplus O^2,..., P^K \oplus O^K)$, where each C^i, P^i, and O^i represents a single bit of the ciphertext block C, plaintext block P, and output block O, respectively. Similarly, during decryption, plaintext is produced by exclusive-ORing a K-bit unit of ciphertext with the most significant K bits of the output block, i.e., $(P^1,P^2,...,P^K) = (C^1 \oplus O^1, C^2 \oplus O^2,..., C^K \oplus O^K)$. In both cases, the unused bits of the output block are discarded. For both the encryption and decryption processes, the next input block is created by discarding the most significant K bits of the previous input block, shifting the remaining bits K positions to the left and then inserting the K bits of ciphertext just produced in the encryption operation or just used in the decryption operation into the least significant bit positions, i.e., $(I^1,I^2,...,I^{64}) = (I^{[K+1]}, I^{[K+2]},..., I^{64},C^1,C^2,...,C^K)$. NOTE -- I, P, and C now refer to the second block. The second block is processed through the TDEA to produce the second ciphertext block (or plaintext block, if decrypting). The input block is then processed through DEA1 in the encrypt state. This process continues until the entire plaintext message has been encrypted or until the entire ciphertext message has been decrypted. For each operation of the TDEA, one K-bit unit of plaintext produces one K-bit unit of ciphertext, and one K-bit unit of ciphertext produces one K-bit unit of plaintext.

2.5 TDEA Cipher Feedback Mode of Operation - Pipelined (TCFB-P)

Both the encryption and decryption processes of the TDEA Cipher Feedback – Pipelined (TCFB-P) mode of operation require 3 IVs, a K-bit TEXT and 3 keys. The IVs, denoted IV1, IV2, and IV3 are generated based on the specifications mentioned in the introduction of Section 2. For both the encryption and decryption processes of the TCFB-P mode of operation, the IV values are assigned to the input block of DEA1 in succession.

The TCFB-P mode of operation is intended for systems equipped with multiple DEA processors. Each of the DEA processors used in both the encryption and decryption processes utilize the same processing as that used in the TCFB mode of operation. With three DEA functional blocks, which are simultaneously clocked, and with three IVs, the TCFB encryption and decryption processes can be pipelined.

Prior to commencing both the TCFB-P encryption and decryption processes, a 3-step initialization process must be conducted as follows:

Step 1: IV1 is input to DEA1 and encrypted using KEY1.

Step 2: The output of DEA1 is input to DEA2 and decrypted using KEY2. Simultaneously, IV2 is input to DEA1 and encrypted using KEY1.

Step 3: The output of DEA2 is input to DEA3 and encrypted using KEY3. This produces the first output block. Simultaneous with encryption by DEA3, the output of DEA1 (from step 2) is input to DEA2 and decrypted using KEY2, and IV3 is input to DEA1 and encrypted using KEY1.

During encryption, a K-bit ciphertext block is produced by exclusive-ORing the most significant K-bits of the output block from DEA3 with the K-bit plaintext block.

Successive input blocks for DEA1 are formed by discarding the most significant K bits of the previous DEA1 input block, shifting the remaining bits K positions to the left and then inserting the K bits of the newest K-bit ciphertext block into the least significant bit positions. DEA1, DEA2 and DEA3 are run simultaneously to produce successive output blocks that are exclusive-ORed to successive K-bit plaintext blocks to produce successive K-bit ciphertext blocks.

Decryption is performed in the same manner as encryption, except that the role of the plaintext and the ciphertext are reversed.

2.6 TDEA Output Feedback (TOFB) Mode

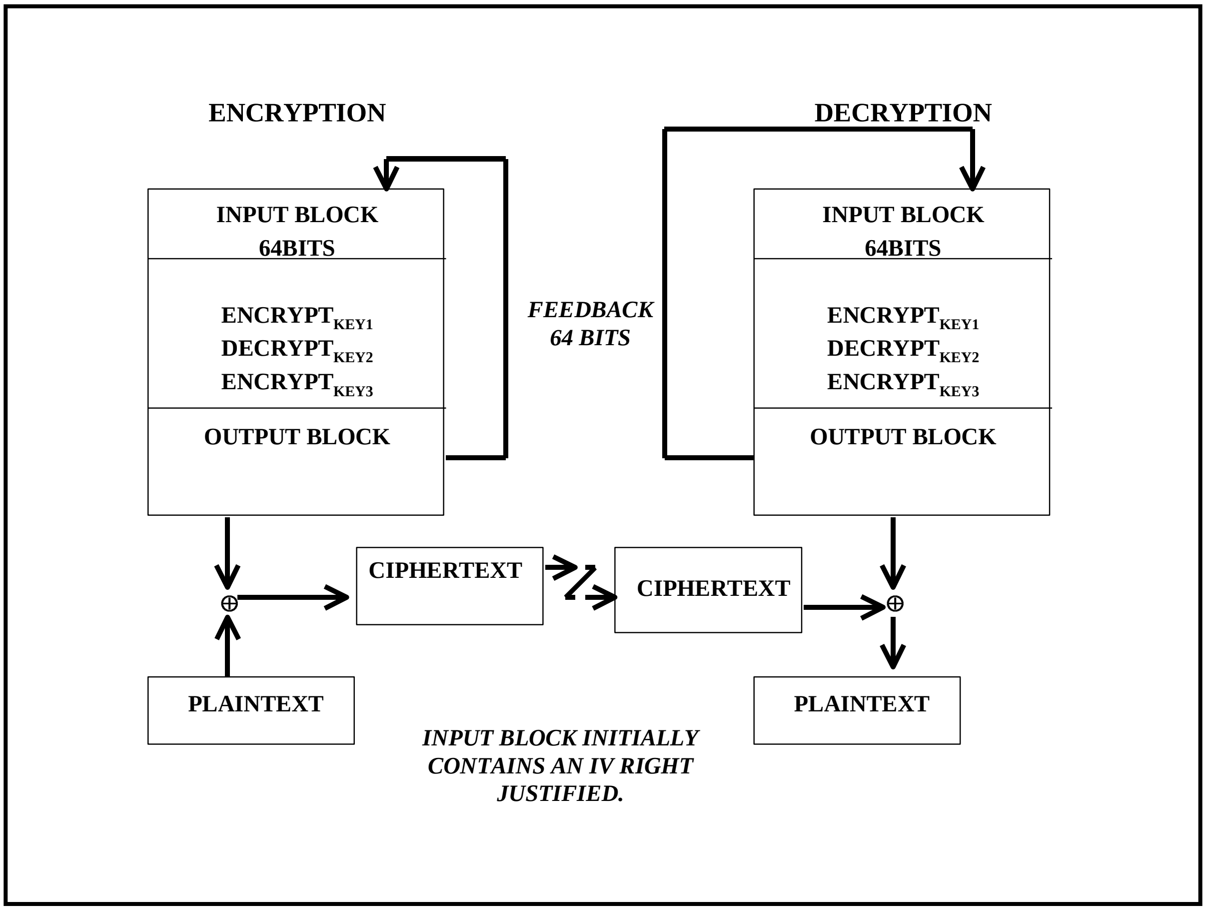

Figure 4 TDEA Output Feedback (TOFB) Mode

The TDEA Output Feedback (TOFB) mode is shown in Figure 4. A message to be encrypted is divided into 64-bit data units. In both the TOFB encrypt and decrypt operations, a 64-bit initialization vector (IV) is used. The IV is used as input in the first round, i.e., (I = IV). This input block is processed through DEA1 where DES is processed in the encrypt state using KEY1. The output of this process is fed directly to the input of DEA2 where DES is processed in the decrypt state using KEY2. The output of this process is fed directly to the input of DEA3 where DES is processed in the encrypt state using KEY3. During encryption, ciphertext is produced by exclusive-ORing a plaintext data unit with an output block, i.e., ($C = P \oplus O$). Similarly, during decryption, plaintext is produced by exclusive-ORing a ciphertext with an output block, i.e., ($P = C \oplus O$). In both cases the next input block is assigned the value of the output block, i.e., (I = O). This input block is then processed through the TDEA as described above. This process continues until the entire plaintext message has been encrypted or until the entire ciphertext message has been decrypted.

2.7 TDEA Output Feedback Mode of Operation - Interleaved (TOFB-I)

Both the encryption and decryption processes of the TDEA Output Feedback – Interleaved (TOFB-I) mode of operation require 3 IVs, a TEXT and 3 keys. The IVs, denoted IV1, IV2, and IV3, are generated based on the specifications mentioned in the introduction of Section 2. For both the encryption and decryption processes of the TOFB-I mode of operation, the IV values are assigned to the input block of DEA1 in succession.

The TOFB-I mode of operation is intended for systems equipped with multiple DEA processors. Each of the DEA processors used in both the encryption and decryption processes utilize the same processing as that used in the TOFB mode of operation. With three DEA functional blocks, which are simultaneously clocked, and with three IVs, the TOFB encryption and decryption processes can be interleaved.

Prior to commencing both the TOFB-I encryption and decryption processes, a 3-step initialization process must be conducted as follows:

Step 1: IV1 is input to DEA1 and encrypted using KEY1.

Step 2: The output of DEA1 is input to DEA2 and decrypted using KEY2. Simultaneously, IV2 is input to DEA1 and encrypted using KEY1.

Step 3: The output of DEA2 is input to DEA3 and encrypted using KEY3. This produces the first output block. Simultaneous with encryption by DEA3, the output of DEA1 (from step 2) is input to DEA and decrypted using KEY2, and IV3 is input to DEA1 and encrypted using KEY1.

During encryption, a ciphertext block is produced by exclusive-ORing the output block from DEA3 with the plaintext block.

Successive input blocks for DEA1 are formed by assigning them the value of the newest ciphertext block. DEA1, DEA2 and DEA3 are run simultaneously to produce successive output blocks that are exclusive-ORed to successive plaintext blocks to produce successive ciphertext blocks.

Decryption is performed in the same manner as encryption, except that the role of the plaintext and the ciphertext are reversed.

3. MODES OF OPERATION VALIDATION SYSTEM FOR THE TRIPLE DES (TDES) ALGORITHM

The TMOVS for the Triple DES algorithm (TDEA) consists of two types of tests, the Known Answer tests and the Monte Carlo tests. The TMOVS provides conformance testing for the components of the algorithm, as well as testing for apparent implementation errors.

The IUTs may be written in software, firmware, hardware, or any combination thereof.

An IUT must allow the TMOVS to have control over the required input parameters for validation to be feasible. The ability to initialize or load known values to the variables required by a specific test may exist at the device level or the chip level in an IUT. If an IUT does not allow the TMOVS to have control over the input parameter values, the TMOVS tests cannot be performed.

An IUT may implement encryption only, decryption only, or both encryption and decryption. This will determine which TMOVS tests will be performed by an IUT.

The following subsections provide an overview of the Known Answer tests and the Monte Carlo tests. This overview discusses the functionality of each test and the components of the TDEA tested by the individual tests.

3.1 The Known Answer Tests

The Known Answer tests are based on the standard DES test set discussed in Special Publication 500-20. They are designed to verify the components of the DES algorithm in the IUT. These components include the initial permutation IP, the inverse permutation IP^{-1}, the expansion matrix E, the data permutation P, the key permutations PC1 and PC2, and the substitution tables S_1, S_2,...,S_8. The tests exercise each bit of every component of the algorithm by processing all possible basis vectors through the IUT.

A generic overview of the sets of Known Answer tests required for the validation of IUTs implementing the encryption and/or decryption processes of all modes of operation for the TDEA is discussed below.

3.1.1 The Encryption Process

An IUT which allows encryption requires the successful completion of five Known Answer tests: the Variable Plaintext Known Answer Test, the Inverse Permutation Known Answer Test, the Variable Key Known Answer Test for the Encryption Process, the Permutation Operation Known Answer Test for the Encryption Process, and the Substitution Table Known Answer Test for the Encryption Process.

These Known Answer tests are also used in the testing of IUTs implementing the decryption process of the TCFB, TCFB-P, TOFB and TOFB-I modes of operation. This is due to the fact that these modes call the three DEA stages in the same order for both the encryption and decryption processes, i.e., encrypt KEY1, decrypt KEY2 and encrypt KEY3.

3.1.1.1 The Variable Plaintext Known Answer Test

To perform the Variable Plaintext Known Answer Test, the TMOVS supplies the IUT with initial values for the three keys, the plaintext(s) and, if applicable, the initialization vector(s). For IUTs supporting the interleaved and pipelined configurations of TDES, initial values for three initialization vectors are supplied by the TMOVS. For IUTs supporting the TCBC-I mode of operation, an initial value is supplied to three plaintext variables. These three plaintext variables are initialized to the same value. The other modes of operation only require one plaintext variable. The TMOVS initializes all keys to zero (with odd parity set). Each block of data input into the TDEA is represented as a 64-bit basis vector.

For the four basic modes of operation (TECB, TCBC, TCFB, and TOFB), the input block is processed through the DEA three times -- first in the encrypt state with KEY1, next in the decrypt state with KEY2, and lastly, in the encrypt state with KEY3. The resulting output block is used in the calculation of the ciphertext.

For modes of operation supporting interleaving and pipelining (TCBC-I, TCFB-P, TOFB-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors resulting in three output blocks which are then used in the calculation of the three ciphertext values. The formation of the input block is dependent upon the mode of operation supported. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated 64 times, using the 64 input basis vectors, allowing for every possible basis vector to be tested. At the completion of the 64^{th} cycle, all results are verified for correctness.

If correct results are obtained from an IUT, the Variable Plaintext Known Answer Test has verified the initial permutation IP and the expansion matrix E via the encrypt operation by presenting a full set of basis vectors to IP and to E. The test also verifies the inverse permutation IP^{-1} via the decrypt operation. It does this by presenting the recovered basis vectors to IP^{-1}.

3.1.1.2 The Inverse Permutation Known Answer Test

To perform the Inverse Permutation Known Answer Test, the TMOVS supplies the IUT with initial values for the three keys, the plaintext(s) and, if applicable, the initialization vector(s). For IUTs supporting the interleaved and pipelined configurations of TDES, three plaintext values and three initialization vector values are supplied by the TMOVS. The values supplied are dependent upon the modes of operation being implemented.

This test performs the same processing as the Variable Plaintext Known Answer Test. The difference is that the plaintext value(s) for this test are set to the ciphertext result(s) obtained from the Variable Plaintext Known Answer Test for the corresponding modes of operation.

The key is initialized to zero (with odd parity set). This key is a self-dual key. A self-dual key is a key with the property that when you encrypt twice with this key, the result is the initial input. Therefore, the result is the same as encrypting and decrypting with the same key. Using a self-dual key allows basis vectors to be presented to components of the DEA to validate the IUT's performance. This is discussed further in the last paragraph of this section.

For the four basic modes of operation (TECB, TCBC, TCFB, and TOFB), the input block is processed through the DEA three times -- first in the encrypt state with KEY1, next in the decrypt state with KEY2, and lastly, in the encrypt state with KEY3. The resulting output block is used in the calculation of the ciphertext, which is then recorded.

For modes of operation supporting interleaving and pipelining (TCBC-I, TCFB-P, TOFB-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three ciphertext values. The formation of the input block is dependent upon the mode of operation supported. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

Using the plaintext and, if applicable, the IV's supplied by the TMOVS, the IUT runs the TDES for 64 cycles. At the completion of the 64^{th} cycle, all results are verified for correctness.

This test, when applied to an IUT, verifies the inverse permutation (IP^{-1}) via the encrypt operation, because as the basis vectors are recovered, each basis vector is presented to the inverse permutation IP^{-1}. By performing the decrypt operation, the initial permutation IP and the expansion matrix E are verified by presenting the full set of basis vectors to them as well.

3.1.1.3 The Variable Key Known Answer Test for the Encryption Process

To implement the Variable Key Known Answer Test for the Encryption Process, the TMOVS supplies the IUT with initial values for the three keys, the plaintext(s), and, if applicable, the initialization vector(s). For IUTs supporting the interleaved and pipelined configurations of TDES, three initialization vector values are supplied by the TMOVS. For IUTs supporting the TCBC-I mode of operation, an initial value is supplied to three plaintext variables. These three plaintext variables are initialized to the same value. The other modes of operation only require one plaintext variable.

During the initialization process, the plaintext value(s) and the initialization vector value(s) are set to zero. All three keys for each round are initialized to a 56-bit key basis vector which contains a "1" in the i^{th} significant position and "0"s in all remaining significant positions of the keys, i.e., KEY1 = KEY2 = KEY3. (NOTE -- the parity bits are not considered as significant bits. These parity bits may be "1"s or "0"s to maintain odd parity.)

For the four basic modes of operation (TECB, TCBC, TCFB, and TOFB), the input block is processed through the DEA three times -- first in the encrypt state with KEY1, next in the decrypt state with KEY2, and lastly, in the encrypt state with KEY3. The resulting output block is used in the calculation of the ciphertext, which is then recorded.

For modes of operation supporting interleaving and pipelining (TCBC-I, TCFB-P, TOFB-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three ciphertext values. The formation of the input block is dependent upon the mode of operation supported. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated 56 times, using the 56 key basis vectors to allow for every possible vector to be tested. At the completion of the 56^{th} cycle, all results are verified for correctness.

When this test is performed for an IUT, the 56 possible key basis vectors which yield unique keys are presented to PC1, verifying the key permutation PC1 via the encrypt operation. Also, during the encrypt operation, a complete set of key basis vectors is presented to PC2 as well, so PC2 is verified.

This test also verifies the right shifts in the key schedule via the DES decrypt operation as the basis vectors are recovered.

3.1.1.4 The Permutation Operation Known Answer Test for the Encryption Process

To implement the Permutation Operation Known Answer Test for the Encryption Process, the TMOVS supplies the IUT with 32 key values. The TMOVS also supplies initial values for the plaintext(s) and, if applicable, the initialization vector(s). For IUTs supporting the interleaved and pipelined configurations of TDES, initial values for three initialization vectors are supplied by the TMOVS. For IUTs supporting the TCBC-I mode of operation, an initial value to be assigned to all three plaintext values is supplied. The other modes of operation only require one plaintext value. During the initialization of a test, the plaintext value(s) and the first (or only) initialization vector value are set to 0, while the key values are assigned to one of the 32 key values supplied by the TMOVS. Note that KEY1=KEY2=KEY3. If more than one initialization vector is used by a TDES mode of operation, the other IVs are computed according to specifications in Section 2.

For the four basic modes of operation (TECB, TCBC, TCFB, and TOFB), the input block is processed through the DEA three times -- first in the encrypt state with KEY1, next in the decrypt state with KEY2, and lastly, in the encrypt state with KEY3. The resulting output block is used in the calculation of the ciphertext, which is then recorded.

For modes of operation supporting interleaving and pipelining (TCBC-I, TCFB-P, TOFB-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three ciphertext values. The formation of the input block is dependent upon the mode of operation supported. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

Each of the 32 key values supplied by the TMOVS is tested. At the completion of the 32nd cycle, all results are verified for correctness.

The 32 key values used in this test present a complete set of basis vectors to the permutation operator P. By doing so, P is verified. This occurs when both the encrypt and decrypt operations are performed.

3.1.1.5 The Substitution Table Known Answer Test for the Encryption Process

To implement the Substitution Table Known Answer Test for the Encryption Process, the TMOVS supplies the IUT with 19 key-data sets. Depending on the mode of operation implemented, the data value will be assigned to the plaintext or to the initialization vector variables. For IUTs supporting the interleaved and pipelined configurations of TDES, initial values for three initialization vectors are also supplied by the TMOVS. For the TCBC-I mode of operation, initial values for three plaintext variables are supplied as well. The other modes of operation only require one plaintext variable. During initialization, the plaintext values (or the initialization vector values, depending on the mode of operation supported), and the key values are initialized to one of the 19 key-data sets supplied by the TMOVS.

For the four basic modes of operation (TECB, TCBC, TCFB, and TOFB), the input block is processed through the DEA three times -- first in the encrypt state with KEY1, next in the decrypt state with KEY2, and lastly, in the encrypt state with KEY3. The resulting output block is used in the calculation of the ciphertext, which is then recorded.

For modes of operation supporting interleaving and pipelining (TCBC-I, TCFB-P, TOFB-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three ciphertext values. The formation of the input block is dependent upon the mode of operation supported. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated for each of the 19 key-data sets, allowing every value in the set of 19 key-data sets to be tested. At the completion of the 19th set, all results are verified for correctness.

The set of 19 key-data sets used in this test result in every entry of all eight S-box substitution tables being used at least once during both the encrypt and decrypt operations. Thus, this test verifies the 64 entries in each of the eight substitution tables.

3.1.2 The Decryption Process

The five Known Answer tests required for validation of IUTs implementing the decryption process of the TDEA consist of the Variable Ciphertext Known Answer Test, the Initial Permutation Known Answer Test, the Variable Key Known Answer Test for the Decryption Process, the Permutation Operation Known Answer Test for the Decryption Process and the Substitution Table Known Answer Test for the Decryption Process. These tests are only performed by IUTs that support the TECB, TCBC, and TCBC-I modes of operation, since only these modes of operation utilize the three DES stages in reverse order during the decryption process. The TCFB, TCFB-P, TOFB, and TOFB-I modes of operation utilize the DES calls in the same order used in the encryption process, i.e., encrypt with KEY1, decrypt with KEY2 and encrypt with KEY3. Therefore, these modes of operation should be tested using the same Known Answer tests used for IUTs that support the encryption process.

3.1.2.1 The Variable Ciphertext Known Answer Test

To perform the Variable Ciphertext Known Answer Test, the TMOVS supplies the IUT with 64 ciphertext values. These values are obtained from the results of the Variable Plaintext Known Answer Test if the IUT performs both encryption and decryption. Otherwise, the TMOVS will supply the IUT with the ciphertext values. If applicable, the TMOVS also supplies initial values for the initialization vector(s). For IUTs supporting the interleaved configuration of the TCBC mode of operation (TCBC-I), 64 sets of ciphertext values consisting of three ciphertext values each and three initialization vectors are supplied. These supplied values are dependent upon the mode of operation being implemented. The keys and initialization vectors are initialized to zero for each test.

For the TECB and TCBC modes of operation, the value of the ciphertext is used directly as the input block of data. The input block is processed through the DEA three times -- first in the decrypt state with KEY3, next in the encrypt state with KEY2, and lastly, in the decrypt state with KEY1. The resulting output block is used in the calculation of the plaintext, which is then recorded.

For the TCBC-I mode of operation, it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three plaintext values. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated once for each of the 64 ciphertext values. If the 64 resulting plaintext values form the set of basis vectors, it can be assumed that all of the operations were performed successfully.

As the basis vectors are recovered via the decrypt operation, they are presented to the inverse permutation IP^{-1}, thus verifying it. This test also verifies the initial permutation IP and the expansion matrix E via the encrypt operation by presenting a full set of basis vectors to these components.

3.1.2.2 The Initial Permutation Known Answer Test

To perform the Initial Permutation Known Answer Test, the TMOVS supplies the IUT with initial values for the ciphertext, the keys, and, if applicable, the initialization vector(s). For IUTs supporting the TCBC-I mode of operation, three ciphertext values and three initialization vector values are supplied. The values supplied are dependent upon the mode of operation being implemented. The ciphertext value(s) are set to the plaintext result(s) obtained from the Variable Ciphertext Known Answer Test.

The key is initialized to zero (with odd parity set). This key is a self-dual key. A self-dual key is a key with the property that when you decrypt (or encrypt) twice with this key, the result is the initial input. Therefore, the result is the same as encrypting and decrypting with the same key. Using a self-dual key allows basis vectors to be presented to components of the DEA to validate the IUT's performance. This is discussed further in the last paragraph of this section.

For the TECB and TCBC modes of operation, the values of the ciphertext are used directly as the input block of data. The input block is processed through the DEA three times -- first in the decrypt state with KEY3, next in the encrypt state with KEY2, and lastly, in the decrypt state with KEY1. The resulting output block is used in the calculation of the plaintext, which is then recorded.

For the TCBC-I mode of operation, it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three plaintext values. The three input blocks are directly assigned the values of the three ciphertext values for each iteration. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is run for each of the 64 ciphertext values. At the completion of the 64th cycle, all results are verified for correctness.

This test, when applied to an IUT, verifies the initial permutation IP and the expansion matrix E via the decrypt operation, by presenting the full set of basis vectors to these components. Via the encrypt operation, this test also verifies the inverse permutation (IP^{-1}) as the basis vectors are recovered by presenting each basis vector to the inverse permutation IP^{-1}.

3.1.2.3 The Variable Key Known Answer Test for the Decryption Process

To implement the Variable Key Known Answer Test for the Decryption Process, the TMOVS supplies the IUT with 56 keys or, for the TCBC-I mode of operation, 56 key sets consisting of three keys each. The TMOVS also supplies initial values for the initialization vector values, if applicable.

During the initialization process, the ciphertext value(s) are initialized in one of two ways. If the IUT supports both encryption and decryption, the values resulting from the encryption performed in the Variable Key Known Answer Test for the Encryption Process will be used to initialize the ciphertext values. Otherwise, the TMOVS will supply the ciphertext values along with the information discussed in the previous paragraph. The initialization vector value(s) are set to zero for each test. All three keys for each round are initialized to a 56-bit key basis vector which contains a "1" in the i^{th} significant position and "0"s in all remaining significant positions of the keys, i.e., KEY1=KEY2=KEY3. (NOTE -- the parity bits are not considered as significant bits. These parity bits may be "1"s or "0"s to maintain odd parity.)

For the TECB and TCBC modes of operation, the values of the ciphertext are used directly as the input blocks of data. The input blocks are processed through the DEA three times -- first in the decrypt state with KEY3, next in the encrypt state with KEY2, and lastly, in the decrypt state with KEY1. The resulting output blocks are used in the calculation of the plaintext values, which are then recorded.

For the interleaved configuration of the TCBC mode of operation (TCBC-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three plaintext values. The three input blocks are directly assigned the values of the three corresponding ciphertext values for each iteration. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated for each of the 56 key basis vectors, allowing for every possible key basis vector to be tested. At the completion of the 56^{th} cycle, all results are verified for correctness.

This test verifies the right shifts in the key schedule via the DES decrypt operation as the basis vectors are recovered.

During the encrypt operation, a complete set of basis vectors is presented to the key permutation, PC1, thus verifying PC1. Since the key schedule consists of left shifts, a complete set of basis vectors is also presented to PC2 verifying PC2 as well.

3.1.2.4 The Permutation Operation Known Answer Test for the Decryption Process

To implement the Permutation Operation Known Answer Test for the Decryption Process, the TMOVS supplies the IUT with 32 key-data sets, consisting of an initial value for the three keys and values for the ciphertext. The TMOVS also supplies initial values for the initialization vector(s), if applicable. For IUTs supporting the TCBC-I mode of operation, three ciphertext values are included in the key-data sets, and three initialization vector values are supplied for each set. The values for the key and ciphertext are supplied in one of two ways. If the IUT performs both encryption and decryption, values for the key and ciphertext resulting from the encryption performed in the Permutation Operation Known Answer Test for the Encryption Process will be used. Otherwise, the key and ciphertext values will be supplied by the TMOVS. If applicable, the initialization vector will be set to zero for each test.

For the TECB and TCBC modes of operation, the values of the ciphertext are used directly as the input blocks of data. The input blocks are processed through the DEA three times -- first in the decrypt state with KEY3, next in the encrypt state with KEY2, and lastly, in the decrypt state with KEY1. The resulting output blocks are used in the calculation of the plaintext values, which are then recorded.

For the TCBC mode of operation supporting interleaving (TCBC-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three plaintext values. The three input blocks are directly assigned the values of the three corresponding ciphertext values for each iteration. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated for each of the 32 key-data sets. At the completion of the 32nd set, the results of each of the 32 tests are verified to be zero.

The 32 key sets used in this test present a complete set of basis vectors to the permutation operator P. By doing so, P is verified. This occurs when both the encrypt and decrypt operations are performed.

3.1.2.5 The Substitution Table Known Answer Test for the Decryption Process

To implement the Substitution Table Known Answer Test for the Decryption Process, the TMOVS supplies the IUT with 19 key-data sets consisting of an initial value for the three keys and values for the ciphertext. The TMOVS also supplies initial values for the initialization vector, if applicable. For IUTs supporting the TCBC-I mode of operation, three ciphertext values are included in the key-data sets and three initialization vector values are supplied for each set. The values for the keys and the ciphertext value(s) are supplied in one of two ways. If the IUT performs both encryption and decryption, the values for the key and ciphertext resulting from the encryption performed in the Substitution Table Known Answer Test for the Encryption Process will be used. Otherwise, the key and ciphertext values will be supplied by the TMOVS. If applicable, the initialization vector will be set to zero for each test.

For the TECB and TCBC modes of operation, the values of the ciphertext are used directly as the input blocks of data. The input blocks are processed through the DEA three times -- first in the decrypt state with KEY3, next in the encrypt state with KEY2, and lastly, in the decrypt state with KEY1. The resulting output blocks are used in the calculation of the plaintext blocks, which are then recorded.

For the TCBC mode of operation supporting interleaving (TCBC-I), it is assumed that multiprocessing is possible, i.e., each block of input data is processed by three DES processors. Therefore, for interleaved modes of operation, three input blocks are processed simultaneously through the three DES processors, resulting in three output blocks which are then used in the calculation of the three plaintext values. The three input blocks are directly assigned the values of the three corresponding ciphertext values for each iteration. Note that the design of the TMOVS assumes that, for security reasons, an IUT is designed so that intermediate values resulting from the first two DES calls are never revealed.

This test is repeated for each of the 19 key-data sets allowing for the set of 19 key-data sets to be processed. At the completion of the 19^{th} set, all results are verified for correctness.

The set of 19 key-data sets used in this test result in every entry of all eight S-box substitution tables being used at least once during both the encrypt and decrypt operations. Thus, this test verifies the 64 entries in each of the eight substitution tables.

The Monte Carlo Test

The Monte Carlo Test is the second type of validation test required to validate IUTs. The Monte Carlo Test is based on the Monte-Carlo test discussed in Special Publication 500-20. It is designed to exercise the entire implementation of the TDEA, as opposed to testing only the individual components. The purpose of the Monte Carlo Test is to detect the presence of flaws in the IUT that were not detected with the controlled input of the Known Answer tests. Such flaws may include pointer problems, errors in the allocation of space, improper error handling, and incorrect behavior of the TDEA implementation when random values are introduced. The Monte Carlo Test does not guarantee ultimate reliability of the IUT that implements the TDEA (i.e., hardware failure, software corruption, etc.).

The TMOVS supplies the IUT with initial input values for the keys, the plaintext(s) (or ciphertext(s)), and, if applicable, initialization vector(s). The Monte Carlo Test is then performed (as described in the following paragraph), and the resulting ciphertext (or plaintext) values are recorded and compared to expected values. If an error is detected, the erroneous result is recorded, and the test terminates abnormally. Otherwise, the test continues. If the IUT's results are correct, the Monte Carlo Test for the IUT ends successfully.

Each Monte Carlo Test consists of four million cycles through the TDEA implemented in the IUT. These cycles are divided into four hundred groups of 10,000 iterations each. Each iteration consists of processing an input block through three operations of the DEA resulting in an output block. For IUTs of the encryption process, the three DES operations are encrypted with KEY1, decrypted with KEY2, and encrypted with KEY3. For IUTs of the decryption process, the three DES operations are decrypted with KEY3, encrypted with KEY2, and decrypted with KEY1. At the $10{,}000^{th}$ cycle in an iteration, new values are assigned to the variables needed for the next iteration. The results of each $10{,}000^{th}$ encryption or decryption cycle are recorded and evaluated as specified in the preceding paragraph.

4. BASIC PROTOCOL

4.1 Overview

Input and output messages used to convey information between the TMOVS and the IUT consist of specific fields. The format of these input and output messages is beyond the scope of this document, and the testing laboratories have the option to determine the specific formats of those messages. However, the results sent to NIST must include certain minimum information, which is specified in Section 4.4 Output Types.

A separate message should be created for each mode of operation supported by an IUT. The information should indicate the algorithm used (Triple DES), the mode of operation (TECB, TCBC, TCBC-I, TCFB-including feedback amounts, TCFB-P-including feedback amounts, TOFB, TOFB-I), the cryptographic process supported (encryption and/or decryption), the test being performed (one of the various Known Answer tests, or the Monte Carlo Tests), and the required data fields. The required data may consist of counts, keys, initialization vectors, and data representing plaintext or ciphertext. Every field in an output message should be clearly labeled to indicate its contents - this is especially important for NIST to be able to ensure that test results are complete.

4.1.1 Conventions

The following conventions should be used in the data portion of messages between the TMOVS and the IUT: (See Section 4.1.2 for these notations.)

1. Integers: integers should be unsigned and should be represented in decimal notation.

2. Hexadecimal strings: should consist of ASCII hexadecimal characters. The ASCII hexadecimal characters to be used should consist of the ASCII characters 0-9 and A-F (or a-f), which represent 4-bit binary values.

3. Characters: the characters to be represented are A-Z (or a-z), 0-9, and underscore (_).

4.1.2 Message Data Types

The following data types should be used in messages between the TMOVS and the IUT:

1. Decimal integers: a decimal integer should have the form

   ddd ... dd

   where each “d” represents a decimal character (0-9); one or more characters should be present. The characters must be contiguous.

2. Hexadecimal strings: a hexadecimal string should have the form

   hhh ... hh

where each “h” should represent an ASCII character 0-9 or A-F (or a-f). Each “h” represents a 4-bit binary value.

3. Characters: an ASCII character should have the form

c

where “c” represents an ASCII character A-Z (or a-z), 0-9, or underscore (_).

4.2 Message Contents

The information included in a message consists of the following:

Algorithm - Triple DES,

Mode - selections consist of TECB, TCBC, TCBC-I, TCFB-including feedback amounts, TCFB-P-including feedback amounts, TOFB, TOFB-I

Process - selections consist of ENCRYPT or DECRYPT,

Test - selections consist of:

VTEXT for Variable Plaintext/Ciphertext Known Answer Test

VKEY for Variable KEY Known Answer Test

INVPERM for Inverse Permutation Known Answer Test

INITPERM for Initial Permutation Known Answer Test

PERM for Permutation Operation Known Answer Test

SUB for Substitution Table Known Answer Test

MODES for Monte Carlo Test

Input/Output Data

The contents of the input/output data included in a message depend on the algorithm, mode, process, and test being performed. These different combinations of data have been organized into input types and output types. The input types are used by the TMOVS to supply data to the IUT for testing. The output types are used by the IUT to supply results from the tests to the TMOVS, and eventually to NIST.

4.3 Input Types

Twenty-five different combinations of input data are used by the TMOVS to support the various Known Answer tests and Monte Carlo tests.

4.3.1 Input Type 1

Input Type 1 consists of:

KEY and DATA

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

DATA is a 16 character ASCII hexadecimal string representing plaintext if the encryption process is being tested, or ciphertext if the decryption process is being tested.

4.3.2 Input Type 2

Input Type 2 consists of:

KEY, IV, and DATA

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector; and

DATA is 1 to 64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested.

4.3.3 Input Type 3

Input Type 3 consists of:

KEY, n, $DATA_1$, $DATA_2$,...,$DATA_n$

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

n is an integer which indicates the number of ciphertext (C) values to follow; and

each $DATA_n$ is 1 to 64 binary bits represented as a 16 character ASCII hexadecimal string. This field should provide plaintext if the encryption process is being tested, or ciphertext if the decryption process is being tested.

4.3.4 Input Type 4

Input Type 4 consists of:

KEY

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3.

4.3.5 Input Type 5

Input Type 5 consists of:

KEY, IV, n, $TEXT_1$, $TEXT_2$,...,$TEXT_n$

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

n is an integer which indicates the number of TEXT values to follow; and

each $TEXT_n$ is 1 to 64 binary bits represented as an ASCII hexadecimal string. TEXT represents P, C, or RESULT.

4.3.6 Input Type 6

Input Type 6 consists of:

KEY and IV

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

IV is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector.

4.3.7 Input Type 7

Input Type 7 consists of

P, KEY_1, KEY_2,...,KEY_{32}

where P is 1 to 64 binary bits represented as a 16 character ASCII hexadecimal string; and each KEY_i, where i=1 to 32, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3.

4.3.8 Input Type 8

Input Type 8 consists of:

$TEXT,IV,KEY_1,KEY_2,...,KEY_{32}$

where TEXT is 1 to 64 binary bits represented as an ASCII hexadecimal string. (NOTE -- TEXT may be referred to as plaintext or text.);

IV is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector; and

each KEY_i, where i=1 to 32, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3.

4.3.9 Input Type 9

Input Type 9 supplies *n* key/input block pairs. It consists of:

$n,PAIR_1,PAIR_2,...,PAIR_n$

In this input type, the integer *n* indicates the number of KEY values to follow. Each $PAIR_i$ consists of:

KEY_i and $TEXT_i$

where each KEY_i, where i=1 to *n*, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

each $TEXT_i$, for $i = 1$ to *n*, is a 16 character ASCII hexadecimal string representing either plaintext or ciphertext.

4.3.10 Input Type 10

Input Type 10 consists of:

$n, KEY_1, KEY_2,...,KEY_n$

where *n* is an integer which indicates the number of KEY values to follow; and each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

4.3.11 Input Type 11

Input Type 11 consists of:

INITVAL, n, $PAIR_1$, $PAIR_2$,...,$PAIR_n$

where INITVAL is a 16 character ASCII hexadecimal string representing either the 64-bit IV or the TEXT, depending on the mode of operation implemented by the IUT. (NOTE -- The TEXT may be referred to as plaintext, ciphertext, or text.);

n is an integer, which indicates the number of KEY/INPUT PAIRs to follow.

Each $PAIR_i$ consists of:

KEY_i and I_i

where each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

each I_i is a 16 character ASCII hexadecimal string representing either the 64-bit IV, P or C, depending on the mode of operation implemented.

4.3.12 Input Type 12

Input Type 12 consists of:

INITVAL, n, KEY_1, KEY_2,..., KEY_n

where INITVAL is a 16 character ASCII hexadecimal string representing either the 64-bit IV or the 64-bit TEXT depending on the mode of operation implemented by the IUT. (NOTE -- The TEXT may be referred to as ciphertext.);

n is an integer which indicates the number of KEYS to follow; and

each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3.

4.3.13 Input Type 13

Input Type 13 consists of:

KEY, IV1, IV2, IV3, DATA

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

and

DATA is 1 to 64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, ciphertext if the decrypt process is being tested, or TEXT for TCFB-P mode. DATA may represent the value of DATA1, DATA2 and DATA3 for Interleaved modes of operation.

4.3.14 Input Type 14

Input Type 14 consists of:

KEY, IV1, IV2, IV3

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

4.3.15 Input Type 15

Input Type 15 consists of:

KEY, IV1, IV2, IV3, n, TEXT1_1,...,TEXT1_n, TEXT2_1,...,TEXT2_n, TEXT3_1,...,TEXT3_n

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

n is an integer, which indicates the number of TEXT1s, TEXT2s, and TEXT3s to follow;

each TEXT1_n is 1 to 64 binary bits represented as an ASCII hexadecimal string. TEXT1 represents P, C, or RESULT;

each TEXT2_n is 1 to 64 binary bits represented as an ASCII hexadecimal string. TEXT2 represents P, C, or RESULT; and

each TEXT3_n is 1 to 64 binary bits represented as an ASCII hexadecimal string. TEXT3 represents P, C, or RESULT.

4.3.16 Input Type 16

Input Type 16 consists of:

IV1, IV2, IV3, n,KEY_1, KEY_2,...,KEY_n

where IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

n is an integer which indicates the number of KEY values to follow; and

each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

4.3.17 Input Type 17

Input Type 17 consists of:

IV1, IV2, IV3, n, GROUP_1, GROUP_2,..., GROUP_n

where IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

n is an integer, which indicates the number of KEY/INPUT GROUPs to follow.

Each GROUP_i consists of:

KEY_i, $TEXT1_i$, $TEXT2_i$, and $TEXT3_i$

where each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

each $TEXT1_i$, $TEXT2_i$, and $TEXT3_i$ is a 16 character ASCII hexadecimal string representing the 64-bit C1, C2, and C3 respectively.

4.3.18 Input Type 18

Input Type 18 consists of

TEXT, IV1, IV2, IV3, KEY_1, KEY_2,...,KEY_{32}

Where TEXT is 1 to 64 binary bits represented as an ASCII hexadecimal string. TEXT may represent P or TEXT depending on the mode of operation being implemented;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA; and

each KEY_i, where i=1 to 32, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3.

4.3.19 Input Type 19

Input Type 19 consists of:

IV1, IV2, IV3, n, $PAIR_1$, $PAIR_2$,..., $PAIR_n$

where IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

n is an integer which indicates the number of KEY/INPUT PAIRs to follow.

Each $PAIR_i$ consists of:

KEY_i and $TEXT_i$

where each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3; and

each $TEXT_i$ is a 16 character ASCII hexadecimal string. TEXT may represent the 64-bit TEXT1, TEXT2, and TEXT3 values, or the IV1 value depending on the mode of operation implemented.

4.3.20 Input Type 20

Input Type 20 consists of:

KEY1, KEY2, KEY3, DATA

where KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity; and

DATA is a 16 character ASCII hexadecimal string representing plaintext if the encryption process is being tested, or ciphertext if the decryption process is being tested.

4.3.21 Input Type 21

Input Type 21 consists of:

KEY1, KEY2, KEY3, IV, and DATA

where KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity.;

IV is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector; and

DATA is 1 to 64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested.

4.3.22 Input Type 22

Input Type 22 consists of:

KEY1, KEY2, KEY3, IV1, IV2, IV3, TEXT1, TEXT2, and TEXT3

where KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity.;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

TEXT1 is 64 binary bits represented as a 16 character ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested;

TEXT2 is 64 binary bits represented as a 16 character ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested; and

TEXT3 is 1 to 64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested.

4.3.23 Input Type 23

Input Type 23 consists of:

KEY, IV1, IV2, IV3, n, $TEXT_1$,...,$TEXT_n$

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

n is an integer which indicates the number of TEXT values to follow; and

each $TEXT_n$ is 1 to 64 binary bits represented as a 16 character ASCII hexadecimal string. TEXT1 represents P, C, or RESULT;

4.3.24 Input Type 24

Input Type 24 consists of:

KEY1, KEY2, KEY3, IV1, IV2, IV3, and TEXT

where KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity.;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA; and

TEXT is 1 to 64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested.

4.3.25 Input Type 25

Input Type 25 consists of:

TEXT, n, $GROUP_1$, $GROUP_2$,..., $GROUP_n$

where TEXT is 1 to 64 binary bits represented as a 16 character ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested;

n is an integer which indicates the number of KEY/IV1/IV2/IV3 groups to follow.

Each $GROUP_i$ consists of:

KEY_i, $IV1_i$, $IV2_i$, and $IV3_i$

where each KEY_i, where i=1 to n, is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

$IV1_i$ is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

$IV2_i$ is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555; and

$IV3_i$ is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA.

4.4 Output Types

Eight different combinations of output data are used by the TMOVS to support the various Known Answer tests and Monte Carlo tests.

4.4.1 Output Type 1

Output Type 1 consists of:

COUNT, KEY, DATA, and RESULT

where COUNT is an integer between 1 and 64, i.e., 0 < COUNT <= 64, representing the output line;

KEY should be represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

DATA is a 16 character hexadecimal string representing plaintext if the encrypt process is being tested or ciphertext if the decrypt process is being tested; and

RESULT is a 16 character hexadecimal string indicating the resulting value. Depending on the process of the IUT being tested, the resulting value represents ciphertext (if encrypting) or plaintext (if decrypting).

4.4.2 Output Type 2

Output Type 2 consists of:

COUNT, KEY, CV, DATA, and RESULT

where COUNT is an integer between 1 and 64, i.e., $0 < COUNT <= 64$, representing the output line;

where KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES keys should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

CV is a 16 character ASCII hexadecimal string;

DATA is 1-64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested.; and

RESULT is 1-64 binary bits represented as an ASCII hexadecimal string indicating the resulting value. Depending on the process of the IUT being tested, the resulting value may be ciphertext (if encrypting) or plaintext (if decrypting).

4.4.3 Output Type 3

Output Type 3 consists of:

COUNT, KEY, IV1, IV2, IV3, DATA1, DATA2, DATA3, RESULT1, RESULT2, RESULT3

where COUNT is an integer between 1 and 64, i.e., $0 < COUNT <= 64$, representing the output line;

KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

DATA1, DATA2, and DATA3 are 1-64 binary bits represented as an ASCII hexadecimal strings representing values of plaintext P1, P2, and P3 or input blocks I1, I2, and I3 respectively, if the encrypt process is being tested, or values of ciphertext for C1, C2, and C3 if the decrypt process is being tested;

RESULT1, RESULT2, and RESULT3 are 1-64 binary bits represented as an ASCII hexadecimal strings indicating the resulting values corresponding to either C1, C2, and C3 or P1, P2, and P3. Depending on the process of the IUT being tested, the resulting value may be ciphertext (if encrypting) or plaintext (if decrypting).

4.4.4 Output Type 4

Output Type 4 consists of:

> COUNT, KEY1, KEY2, KEY3, CV1, CV2, CV3, DATA1, DATA2, DATA3, RESULT1, RESULT2, RESULT3

where COUNT is an integer between 1 and 64, i.e., 0 < COUNT <= 64, representing the output line;

where KEY1, KEY2, and KEY3 is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES keys should be presented in odd parity.;

CV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

CV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

CV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

DATA1, DATA2, and DATA3 are 16 character hexadecimal strings representing values of plaintext for P1, P2, and P3 respectively, if the encrypt process is being tested, or values of ciphertext for C1, C2, and C3 if the decrypt process is being tested;

RESULT1, RESULT2, and RESULT3 are 16 character hexadecimal strings indicating the resulting values corresponding to either C1, C2, and C3 or P1, P2, and P3. Depending on the process of the IUT being tested, the resulting value may be ciphertext (if encrypting) or plaintext (if decrypting).

4.4.5 Output Type 5

Output Type 5 consists of:

> COUNT, KEY1, KEY2, KEY3, DATA, and RESULT

where COUNT is an integer between 1 and 400, i.e., 0 < COUNT <= 400, representing the output line;

KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES keys should be presented in odd parity.;

DATA is a 16 character hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested; and

RESULT is a 16 character hexadecimal string indicating the resulting value. Depending on the process of the IUT being tested, the resulting value represents ciphertext (if encrypting) or plaintext (if decrypting).

4.4.6 Output Type 6

Output Type 6 consists of:

COUNT, KEY1, KEY2, KEY3, CV, DATA, and RESULT

where COUNT is an integer between 1 and 400, i.e., 0 < COUNT <= 400, representing the output line;

KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES keys should be presented in odd parity.;

CV is a 16 character ASCII hexadecimal string representing IV or I depending on the mode implemented;

DATA is 1-64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested, or ciphertext if the decrypt process is being tested; and

RESULT is 1-64 binary bits represented as an ASCII hexadecimal string indicating the resulting value. Depending on the process of the IUT being tested, the resulting value should represent ciphertext (if encrypting) or plaintext (if decrypting).

4.4.7 Output Type 7

Output Type 7 consists of:

COUNT, KEY, IV1, IV2, IV3, DATA, RESULT1, RESULT2, RESULT3

where COUNT is an integer between 1 and 64, i.e., 0 < COUNT <= 64, representing the output line;

KEY is represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES key should be presented in odd parity. KEY represents the value of KEY1, KEY2, and KEY3;

IV1 is a 16 character ASCII hexadecimal string representing the 64-bit initialization vector;

IV2 is assigned the value of IV1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

IV3 is assigned the value of IV1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

DATA is 1-64 binary bits represented as an ASCII hexadecimal string representing the value of the plaintext if the encrypt process is being tested, or the value of the ciphertext if the decrypt process is being tested; and

RESULT1, RESULT2, and RESULT3 is 1-64 binary bits represented as an ASCII hexadecimal string indicating the resulting values, which may be ciphertext (if encrypting), or plaintext (if decrypting).

4.4.8 Output Type 8

Output Type 8 consists of:

COUNT, KEY1, KEY2, KEY3, I1, I2, I3, DATA, and RESULT

where COUNT is an integer between 1 and 400, i.e., 0 < COUNT <= 400, representing the output line;

KEY1, KEY2, and KEY3 are represented as 64 bits in hexadecimal notation (i.e., 4 bits per hexadecimal character). The 8 parity bits should be present but ignored, yielding 56 significant bits. For consistency purposes, the DES keys should be presented in odd parity;

I1 is a 16 character ASCII hexadecimal string representing IV or I;

I2 is assigned the value of I1 + R_1 mod 2^{64}, where R_1 = 5555555555555555;

I3 is assigned the value of I1 + R_2 mod 2^{64}, where R_2 = AAAAAAAAAAAAAAAA;

DATA is 1-64 binary bits represented as an ASCII hexadecimal string representing plaintext if the encrypt process is being tested or ciphertext if the decrypt process is being tested; and

RESULT is 1-64 binary bits represented as an ASCII hexadecimal string indicating the resulting value. Depending on the process of the IUT being tested, the resulting value should represent ciphertext (if encrypting) or plaintext (if decrypting).

5. TESTS REQUIRED TO VALIDATE AN IMPLEMENTATION OF THE TRIPLE DES ALGORITHM

The validation of IUTs of the Triple DES algorithm (TDEA) should require the successful completion of an applicable set of Known Answer tests and the appropriate Monte Carlo tests. The tests required for validation of an IUT should be determined by several factors. These include the mode(s) of operation supported (TECB, TCBC, TCBC-I, TCFB, TCFB-P, TOFB, TOFB-I), the keying option used, and the allowed cryptographic processes (encryption, decryption, both).

A separate set of Known Answer tests has been designed for use with each of the seven modes of TDES. Within these sets of tests are separate subsets of tests corresponding to the encryption and decryption processes. If an IUT implements multiple modes of operation, the set of Known Answer tests corresponding to each supported mode of operation should be tested.

The Monte Carlo tests have been designed for use with each of the seven modes of TDES. For the TECB, TCBC, TCBC-I, TCFB, and TCFB-P modes of operation, there are two tests associated with each: one to be used for IUTs allowing the encryption process, and the other to be used for IUTs allowing the decryption process. If both the encryption and decryption processes are allowed by an IUT, both tests are required. The TOFB and TOFB-I modes of operation only require one Monte Carlo test, which is designed for use with both the encryption and decryption processes of an IUT. For example, if an IUT implements the TCBC mode of operation in both the encryption and decryption processes, the Monte Carlo Test for the encryption process and the Monte Carlo Test for the decryption process of the TCBC mode of operation should be successfully completed to validate the IUT. If an IUT implements both the encryption and decryption processes of the TCFB-P mode of operation, the Monte Carlo Test for the encryption process and the Monte Carlo Test for the decryption process of the TCFB-P mode of operation should be successfully completed to validate the IUT. If an IUT implements both the encryption and decryption processes of the TOFB mode of operation, the Monte Carlo Test for the TOFB mode of operation should be successfully completed to validate the IUT.

If an IUT supports more than one mode of operation, the Monte Carlo Test corresponding to each supported mode should be performed successfully. For example, if an IUT implements the TECB and TCBC modes of operation in the encryption process, the Monte Carlo Test for the encryption process of both the TECB and the TCBC modes of operation should be successfully completed to validate the IUT.

If an IUT supports the 3-key keying option, where KEY1, KEY2 and KEY3 are independent, the Monte Carlo Test should be successfully completed three times – once where the three keys are independent, once where KEY1 and KEY2 are independent and KEY3 = KEY1, and once where KEY1 = KEY2 = KEY3 – to validate the IUT. If an IUT supports the 2-key keying option, where KEY1 and KEY2 are independent and KEY3 = KEY1, the Monte Carlo Test should be successfully completed two times – once where KEY1 and KEY2 are independent and KEY3 = KEY1, and once where KEY1 = KEY2 = KEY3 – to validate the IUT. If an IUT only supports the 1-key keying option, where KEY1=KEY2=KEY3, the Monte Carlo Test should be successfully completed once with all the keys being equal to validate the IUT.

The tests required to successfully validate IUTs are detailed in the following sections. These sections are categorized by mode of operation. Within each mode of operation, the tests are divided into tests to use with the encryption process and tests to use with the decryption process.

5.1 TDEA Electronic Codebook (TECB) Mode

The IUTs which implement the TDES Electronic Codebook (TECB) mode should be validated by the successful completion of a series of Known Answer tests and Monte Carlo tests corresponding to the cryptographic processes allowed by the IUT.

5.1.1 Encryption Process

The process of validating an IUT which implements the TECB mode of operation for the encryption process should involve the successful completion of the following six tests:

1. The Variable Plaintext Known Answer Test - TECB mode

2. The Inverse Permutation Known Answer Test - TECB mode

3. The Variable Key Known Answer Test for the Encryption Process - TECB mode

4. The Permutation Operation Known Answer Test for the Encryption Process - TECB mode

5. The Substitution Table Known Answer Test for the Encryption Process - TECB mode

6. The Monte Carlo Test for the Encryption Process - TECB mode

An explanation of the tests follows.

5.1.1.1 The Variable Plaintext Known Answer Test - TECB Mode

Table 1 The Variable Plaintext Known Answer Test - TECB Mode

TMOVS:	Initialize	KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
.		P_1 = 8000000000000000
.	Send	KEY (representing KEY1, KEY2, KEY3), P_1
IUT:	FOR i = 1 to 64	
	{	
		$I_i = P_i$
Perform Triple DES:		I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1
		TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2
		TEMP2 is encrypted by DEA_3 using KEY3, resulting in C_i
		Send i, KEY (representing KEY1, KEY2, and KEY3), P_i, C_i
		P_{i+1} = basis vector where single "1" bit is in position i+1
	}	
TMOVS:	Compare results from each loop with known answers.	
	See Table A.1.	

Table 1 illustrates the Variable Plaintext Known Answer Test for the TECB mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit plaintext P_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $P_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

   c. Forwards this information to the IUT using Input Type 1.

2. The IUT should perform the following for i=1 through 64:

   a. Set the input block I_i equal to the value of P_i.

   b. Process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in ciphertext C_i.

   c. Forward the current values of the loop number i, KEY (where KEY represents the value of KEY1, KEY2, and KEY3), P_i, and the resulting C_i to the TMOVS as specified in Output Type 1.

   d. Retain C_i for use with the Inverse Permutation Known Answer Test for the TECB Mode (Section 5.1.1.2), and, if the IUT supports the decryption process, for use with the Variable Ciphertext Known Answer Test for the TECB Mode (Section 5.1.2.1).

   e. Assign a new value to P_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64.

NOTE -- This continues until every possible basis vector has been represented by the P, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values found in Table A.1.

5.1.1.2 The Inverse Permutation Known Answer Test - TECB Mode

Table 2 The Inverse Permutation Known Answer Test - TECB Mode

TMOVS:	Initialize:	KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) P_i (where i=1..64) = 64 C values from the Variable Plaintext Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), $P_1,...,P_{64}$
IUT:	FOR i = 1 to 64	
	{	
Perform Triple DES:		$I_i = P_i$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in C_i
		Send i, KEY (representing KEY1, KEY2, and KEY3), P_i, C_i P_{i+1} = corresponding P_{i+1} from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. Should be the set of basis vectors.	

Table 2 illustrates the Inverse Permutation Known Answer Test for the TECB mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}=KEY2_{hex}=KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit plaintext P_i (where i=1..64) to the C_i results obtained from the Variable Plaintext Known Answer Test.

   c. Forwards this information to the IUT using Input Type 3.

2. The IUT should perform the following for i=1 through 64:

a. Set the input block I_i equal to the value of P_i.

b. Process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in ciphertext C_i.

c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), P_i, and the resulting C_i to the TMOVS as specified in Output Type 1.

d. Assign a new value to P_{i+1} by setting it equal to the corresponding output from the TMOVS.

NOTE -- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The C values should be the set of basis vectors.

5.1.1.3 The Variable Key Known Answer Test for the Encryption Process - TECB Mode

Table 3 The Variable Key Known Answer Test for the Encryption Process - TECB Mode

TMOVS:	Initialize	KEYs: $KEY1_1$ = $KEY2_1$ = $KEY3_1$ = 8001010101010101 (odd parity set)
		P=0000000000000000
	Send	KEY_1(representing $KEY1_1$, $KEY2_1$, $KEY3_1$), P
IUT:	FOR i = 1 to 64	
	{	
	IF (i % 8 ≠ 0) {process every bit except parity bits}	
	{	
Perform Triple DES:		I = P I is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
		Send i, KEY_i (representing $KEY1_i$,$KEY2_i$, and $KEY3_i$), P, C_i
		$KEY1_{i+1}$ =$KEY2_{i+1}$ =$KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. Each parity bit may have the value "1" or "0" to make the KEY odd parity.
	}	
	}	
TMOVS:	Compare results of the 56 encryptions with known answers.	
	Use Table A.2.	

As summarized in Table 3, the Variable Key Known Answer Test for the TECB Encryption Process is performed as follows:

1. The TMOVS:

a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

NOTE -- the parity bits are set to "0" or "1" to get odd parity.

b. Initializes the 64-bit plaintext P to the value of 0, i.e., P_{hex} = 00 00 00 00 00 00 00 00.

c. Forwards this information to the IUT using Input Type 1.

2. The IUT should perform the following for i = 1 to 56:

NOTE -- 56 is the number of significant bits in a TDES key.

a. Set the input block I equal to the value of P.

b. Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ parameters, process I through the three DEA stages resulting in ciphertext C_i. This involves processing I through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in ciphertext C_i.

c. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, $KEY3_i$), P, and the resulting C_i to the TMOVS as specified in Output Type 1.

d. If the IUT supports the decryption process, retain C_i for use with the Variable Key Known Answer Test for the Decryption Process for the TECB Mode (Section 5.1.2.3).

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE -- The above processing continues until every significant basis vector has been represented by the KEY parameter. The output from the IUT for this test should consist of 56 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values found in Table A.2.

5.1.1.4 Permutation Operation Known Answer Test for the Encryption Process - TECB Mode

Table 4 The Permutation Operation Known Answer Test for the Encryption Process - TECB Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i = 1-32)=32 KEY values in Table A.3 P = 0000000000000000
	Send	P, $KEY_1, KEY_2,...,KEY_{32}$ (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 32	
	{	
		I = P
	Perform Triple DES:	I is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, $KEY3_i$), P, C_i $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. Use Table A.3.	

Table 4 illustrates the Permutation Operation Known Answer Test for the TECB Encryption Process.

1. The TMOVS:

   a. Initializes the KEY1, KEY2, and KEY3 variables with the 32 constant KEY values from Table A.3.

   b. Initializes the plaintext P to the value of 0, i.e., P_{hex}=00 00 00 00 00 00 00 00.

   c. Forwards this information to the IUT using Input Type 7.

2. The IUT should perform the following for i = 1 to 32:

   a. Set the input block I equal to the value of P.

   b. Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ values, process I through the three DEA stages resulting in ciphertext C_i. This involves processing I through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting ciphertext C_i.

   c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, KEY3), P, and the resulting C_i to the TMOVS as specified in Output Type 1.

   d. If the IUT supports the decryption process, retain C_i for use with the Permutation Operation Known Answer Test for the Decryption Process for the TECB mode (Section 5.1.2.4).

   e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the next key supplied by the TMOVS.

NOTE-- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values found in Table A.3.

5.1.1.5 Substitution Table Known Answer Test for the Encryption Process - TECB Mode

Table 5 The Substitution Table Known Answer Test for the Encryption Process - TECB Mode

TMOVS:	Initialize		$KEY1_i$, $KEY2_i$, $KEY3_i$ (where i=1..19) = 19 KEY values in Table A.4 P_i (where i=1..19) = 19 corresponding P values in Table A.4
	Send		KEY_1, P_1, KEY_2, P_2,..., KEY_{19}, P_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 19		
		{	
			$I_i = P_i$
		Perform Triple DES:	I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
			Send i, KEY_i (representing $KEY1_i$,$KEY2_i$,$KEY3_i$), P_i, C_i $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS $P_{i+1} = P_{i+1}$ from TMOVS
		}	
TMOVS:	Compare results from each loop with known answers. Use Table A.4.		

As summarized in Table 5, the Substitution Table Known Answer Test for the TECB Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY-plaintext (KEY-P) pairs with the 19 constant KEY-P values from Table A.4. The KEY value indicates the value of KEY1, KEY2, and KEY3, i.e., KEY1=KEY2=KEY3.

b. Forwards this information to the IUT using Input Type 9.

2. The IUT should perform the following for i = 1 to 19:

   a. Set the input block I_i equal to the value of P_i.

   b. Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in ciphertext C_i.

   c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, KEY3), P_i, and the resulting C_i to the TMOVS as specified in Output Type 1.

   d. If the IUT supports the decryption process, retain C_i for use with the Substitution Table Known Answer Test for the Decryption Process for the TECB mode (Section 5.1.2.5).

   e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the KEY_{i+1} supplied by the TMOVS.

   f. Set P_{i+1} equal to the corresponding P_{i+1} value supplied by the TMOVS.

NOTE-- The above processing should continue until all 19 KEY-P pairs are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values found in Table A.4.

5.1.1.6 Monte Carlo Test for the Encryption Process - TECB Mode

Table 6 The Monte Carlo Test for the Encryption Process - TECB Mode

TMOVS: Initialize $KEY1_0$, $KEY2_0$, $KEY3_0$, P_0

Send $KEY1_0$, $KEY2_0$, $KEY3_0$, P_0

IUT: FOR i = 0 TO 399

{

Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, P_0

FOR j = 0 TO 9,999

{

Perform Triple DES:

$I_j = P_j$

I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1

TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_j

$P_{j+1} = C_j$

}

Record C_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, P_0, C_j

$KEY1_{i+1} = KEY1_i \oplus C_j$

If ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus C_{j-1}$

else

$KEY2_{i+1} = KEY2_i \oplus C_j$

If ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus C_j$

else

$KEY3_{i+1} = KEY3_i \oplus C_{j-2}$

$P_0 = C_j$

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 6, the Monte Carlo Test for the TECB Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, and the plaintext P variables. The P and KEYs consist of 64 bits.

   b. Forwards this information to the IUT using Input Type 20.

2. The IUT should perform the following for i = 0 through 399:

   a. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, and P_0.

   b. Perform the following for j = 0 through 9999:

      1) Set the input block I_j equal to the value of P_j.

      2) Using the corresponding $KEY1_i$, $KEY2_i$ and $KEY3_i$ values, process I_j through the three DEA stages resulting in ciphertext C_j. This involves processing I_j through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in ciphertext C_j.

      3) Prepare for loop j+1 by assigning P_{j+1} with the current value of C_j.

   c. Record C_j.

d. Forward all recorded information for this loop, as specified in Output Type 5, to the TMOVS.

e. Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop. Note j=9999.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the C_j.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the C_{j-1}. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the C_j.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the C_{j-2}. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the C_j.

f. Assign a new value to P in preparation for the next output loop. P_0 should be assigned the value of the current C_j. Note j = 9999.

NOTE -- the new P should be denoted as P_0 to be used for the first pass through the inner loop when j=0.

NOTE-- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 5.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.1.2 Decryption Process

The process of validating an IUT which implements the TECB mode of operation in the decryption process should involve the successful completion of the following six tests:

1. The Variable Ciphertext Known Answer Test - TECB mode

2. The Initial Permutation Known Answer Test - TECB mode

3. The Variable Key Known Answer Test for the Decryption Process- TECB mode

4. The Permutation Operation Known Answer Test for the Decryption Process- TECB mode

5. The Substitution Table Known Answer Test for the Decryption Process - TECB mode

6. The Monte Carlo Test for the Decryption Process- TECB mode

An explanation of the tests follows.

5.1.2.1 The Variable Ciphertext Known Answer Test - TECB Mode

Table 7 The Variable Ciphertext Known Answer Tests - TECB Mode

TMOVS:	Initialize KEYs: KEY1=KEY2=KEY3=0101010101010101 (odd parity set) If encryption is supported by IUT: Send KEY (representing KEY1, KEY2, and KEY3) If encryption is not supported by IUT: Initialize C_i (where i=1..64) = 64 C values in Table A.1 Send KEY (representing KEY1, KEY2, and KEY3), C_1, C_2,...,C_{64}	
IUT:	If encryption is supported by IUT: Initialize C_1 = first value from output of Variable Plaintext Known Answer Test. Otherwise, use the first value received from the TMOVS. FOR i = 1 to 64 {	
	Perform Triple DES:	$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in TEMP1 TEMP1 is encrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is decrypted by DEA_1 using KEY1, resulting in P_i
		Send i, KEY (representing KEY1, KEY2, and KEY3), C_i, P_i If encryption is supported: C_{i+1}= corresponding C_{i+1} from output of Variable Plaintext Known Answer Test else C_{i+1}= the corresponding C_{i+1} value from TMOVS
	}	

TMOVS:	Compare results from each loop with known answers. Should be the set of basis vectors.

Table 7 illustrates the Variable Ciphertext Known Answer test for the TECB mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. If the IUT does not support encryption, the 64 constant ciphertext values from Table A.1 are initialized.

   c. If encryption is supported by the IUT, the KEYs are forwarded to the IUT using Input Type 4. If encryption is not supported by the IUT, forward the KEYs and 64 C values to the IUT using Input Type 3.

2. The IUT should:

   a. If encryption is supported, initialize the C value with the first C value retained from the Variable Plaintext Known Answer Test for the TECB Mode (Section 5.1.1.1). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i=1 through 64:

      1) Set the input block I_i equal to the value of C_i.

      2) Process I_i through the three DEA stages resulting in plaintext P_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using KEY1, resulting plaintext P_i.

      3) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), C_i, and the resulting P_i to the TMOVS as specified in Output Type 1.

      4) Retain P_i for use with the Initial Permutation Known Answer Test for the TECB Mode (Section 5.1.2.2).

      5) If encryption is supported, set C_{i+1} equal to the corresponding output from the Variable Plaintext Known Answer Test for the TECB mode. If encryption is not supported, assign a new value to C_{i+1} by setting it equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE-- The output from the IUT for this test should consist of 64 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The P results should be the set of basis vectors.

5.1.2.2 The Initial Permutation Known Answer Test - TECB Mode

Table 8 Initial Permutation Known Answer Test - TECB Mode

TMOVS:	Initialize KEYs:	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) C_i (where i=1..64)=64 P values from Variable Ciphertext Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), C_1 .. C_{64}
IUT:	FOR i = 1 to 64	
	{	
Perform Triple DES:		$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in TEMP1 TEMP1 is encrypted by DEA_2 using KEY2 , resulting in TEMP2 TEMP2 is decrypted by DEA_1 using KEY1, resulting in P_i
		Send i, KEY (representing KEY1, KEY2, and KEY3), C_i, P_i C_{i+1}= corresponding C_{i+1} from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. See Table A.1.	

Table 8 illustrates the Initial Permutation Known Answer Test for the TECB mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}=KEY2_{hex}=KEY3_{hex}$=01 01 01 01 01 01 01 01.

   NOTE -- the significant bits are set to "0" and the parity bits are set to "1" to make odd parity.

   b. Initializes the 64-bit ciphertext C_i (where i=1,...,64) to the P_i results obtained from the Variable Ciphertext Known Answer Test.

c. Forwards this information to the IUT using Input Type 3.

2. The IUT should perform the following for i=1 through 64:

   a. Set the input block I_i equal to the value of C_i.

   b. Process I_i through the three DEA stages resulting in plaintext P_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using KEY1, resulting in plaintext P_i.

   c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), C_i, and the resulting P_i to the TMOVS as specified in Output Type 1.

   d. Set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE-- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.1.2.3 The Variable Key Known Answer Test for the Decryption Process - TECB Mode

Table 9 The Variable Key Known Answer Tests for the Decryption Process - TECB Mode

TMOVS:	Initialize KEY_1:	$KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set)
	If encryption is supported by the IUT:	
	Send KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$)	
	If encryption is not supported by the IUT:	
	Initialize C_i(where i=1..56): 56 C values in Table A.2	
	Send KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), C_1, C_2,...,C_{56}	
IUT:	If encryption is supported by the IUT:	
	Initialize C_i = first value from output of Variable Key Known Answer Test for the Encryption Process	
	Otherwise, use the first value received from the TMOVS.	
	FOR i = 1 to 64	
	{	
	IF (i mod 8 $\neq$ 0) {process every bit except parity bits}	
	{	
	Perform Triple DES:	$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$ resulting in TEMP1 TEMP1 is encrypted by DEA_2 using $KEY2_i$ resulting in TEMP2 TEMP2 is decrypted by DEA_1 using $KEY1_i$ resulting in P_i
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), C_i, P_i
		$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in

position i+1. NOTE -- odd parity is set.

If encryption is supported by the IUT:

C_{i+1}= corresponding C_{i+1} from output of Variable Key Known Answer Test for the Encryption Process

else

C_{i+1}= corresponding C_{i+1} from TMOVS

}

}

TMOVS: Compare results of the 56 decryptions with known answers.

Should be P=0000000000000000 for all 56 rounds.

Table 9 illustrates the Variable Key Known Answer Test for the TECB Decryption Process.

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

   NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. If the IUT does not support encryption, the C_i values are initialized with the 56 constant C values from Table A.2.

   c. If encryption is not supported by the IUT, the KEY (representing KEY1, KEY2, KEY3), and the 56 C values are forwarded to the IUT using Input Type 3. Otherwise, the KEY (representing KEY1, KEY2, and KEY3) is forwarded to the IUT using Input Type 4.

2. The IUT should:

   a. If encryption is supported, initialize the C value with the first C value retained from the Variable KEY Known Answer Test for the Encryption Process for the TECB Mode (Section 5.1.1.3). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i = 1 to 56:

NOTE – 56 is the number of significant bits in a TDES key.

1) Set the input block I_i equal to the value of C_i.

2) Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ parameters, process I_i through the three DEA stages resulting in plaintext P_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in plaintext P_i.

3) Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), C_i, and the resulting P_i to the TMOVS as specified in Output Type 1.

4) Set $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1,}$, equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE -- $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$.

5) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Variable Key Known Answer Test for the Encryption Process for TECB mode. If encryption is not supported by the IUT, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The output from the IUT for this test should consist of 56 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The P results should be all zeros.

5.1.2.4 Permutation Operation Known Answer Test for Decryption Process - TECB Mode

Table 10 The Permutation Operation Known Answer Test for the Decryption Process - TECB Mode

TMOVS: Initialize $KEY1_i$ =$KEY2_i$ =$KEY3_i$ =(where i=1..32) = 32 KEY values in Table A.3

If encryption is supported by the IUT:

> Send KEY_1, KEY_2,..., KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)

If encryption is not supported by the IUT:

> Initialize C_i (where i=1..32) = corresponding C values in Table 3
>
> Send KEY_1, C_1, KEY_2, C_2,...,KEY_{32}, C_{32} (The key values represent the values of KEY1, KEY2, and KEY3.)

IUT: If encryption is supported by the IUT:

> Initialize C_i = first value retained from Permutation Operation Known Answer Test for the Encryption Process

Otherwise, use the first value received from the TMOVS.

FOR i = 1 to 32

{

Perform Triple DES:

> $I_i = C_i$
>
> I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1
>
> TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2
>
> TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in P_i

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), C_i, P_i

$KEY1_{i+1}$ = $KEY2_{i+1}$ = $KEY3_{i+1}$ = corresponding KEY_{i+1} supplied by TMOVS

If encryption is supported:

C_{i+1} = corresponding C_{i+1} from output of Permutation Operation Known Answer Test for the Encryption Process

else

C_{i+1} = corresponding C_{i+1} from TMOVS

}

TMOVS: Compare results from each loop with known answers.

Should be P=0000000000000000 for all 32 rounds.

Table 10 illustrates the Permutation Operation Known Answer Test for the TECB Decryption Process.

1. The TMOVS:

   a. If the IUT supports encryption, the KEY1, KEY2, and KEY3 variables are initialized with the 32 constant KEY values from Table A.3. If the IUT does not support encryption, the KEY-ciphertext (KEY-C) pairs are initialized with the 32 constant KEY-C pairs from Table A.3.

      NOTE -- KEY1=KEY2=KEY3.

   b. If encryption is supported by the IUT, the 32 KEY values for KEY1, KEY2, and KEY3 are forwarded using Input Type 10. If encryption is not supported by the IUT, the 32 KEY-C pairs are forwarded to the IUT using Input Type 9.

2. The IUT should:

   a. If encryption is supported, initialize the C value with the first C value retained from the Permutation Operation Known Answer Test for the Encryption Process for the TECB Mode (Section 5.1.1.4). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i = 1 to 32:

      1) Set the input block I_i equal to the value of C_i.

      2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in plaintext P_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value

TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in plaintext P_i.

3) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), C_i, and the resulting P_i to the TMOVS as specified in Output Type 1.

4) Assign a new value to $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ by setting them equal to the corresponding KEY_{i+1} value supplied by the TMOVS.

NOTE -- KEY1=KEY2=KEY3.

5) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Permutation Operation Known Answer Test for the Encryption Process for the TECB mode. If encryption is not supported, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE-- The above processing should continue until all 32 KEY-C values are passed as specified in Input Type 9, or all 32 KEY values are passed as specified in Input Type 10. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The P results should be all zeros.

5.1.2.5 Substitution Table Known Answer Test for the Decryption Process - TECB Mode

Table 11 The Substitution Table Known Answer Test for the Decryption Process - TECB Mode

TMOVS: Initialize $KEY1_i = KEY2_i = KEY3_i$ (where i=1..19) = 19 KEY values in Table A.4

If encryption is supported by the IUT:

Send KEY_1, KEY_2,...,KEY_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)

If encryption is not supported by the IUT:

Initialize C_i (where i=1..19) = corresponding C values in Table A.4

Send KEY_1, C_1, KEY_2, C_2,...,KEY_{19}, C_{19} (The key values represent the values of KEY1, KEY2 and KEY3.)

IUT: If encryption is supported by the IUT:

Initialize C_1 = first value retained from the Substitution Table Known Answer Test for the Encryption Process

Otherwise, use the first value received from the TMOVS.

FOR i = 1 to 19

{

Perform Triple DES:

$I_i = C_i$

I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1

TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in P_i

Send i, KEY_i, C_i, P_i (where KEY_i represents the value of KEY1, KEY2 and KEY3)

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} supplied by TMOVS

If encryption is supported:

C_{i+1} = corresponding C_{i+1} from output of Substitution Table Known

Answer Test for the Encryption Process for the TECB mode

else

C_{i+1} = corresponding C_{i+1} from TMOVS

TMOVS: Compare results from each loop with known answers. See Table A.4.

As summarized in Table 11, the Substitution Table Known Answer Test for the TECB Decryption Process is performed as follows:

1. The TMOVS:

   a. If the IUT supports encryption, the KEY1, KEY2, and KEY3 variables are initialized with the 19 constant KEY values from Table A.4. If the IUT does not support encryption, the KEY-ciphertext (KEY-C) pairs are initialized with the 19 constant KEY-C pairs from Table A.4.

      NOTE -- KEY1=KEY2=KEY3.

   b. If encryption is supported by the IUT, the 19 KEY values for KEY1, KEY2, and KEY3 are forwarded to the IUT using Input Type 10. The 19 KEY-C pairs are forwarded to the IUT using Input Type 9 if encryption is not supported by the IUT.

2. The IUT should:

   a. If encryption is supported, initialize the C_i value with the first C value retained from the Substitution Table Known Answer Test for the Encryption Process for the TECB Mode (Section 5.1.1.5). Otherwise, use the first C value received from the TMOVS.

   b. Perform the following for i = 1 to 19:

      1) Set the input block I_i equal to the value of C_i.

      2) Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in plaintext P_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in plaintext P_i.

      3) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), C_i, and the resulting P_i to the TMOVS as specified in Output Type 1.

4) Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the next key supplied by the TMOVS.

5) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Substitution Table Known Answer Test for the Encryption Process for the TECB mode. If encryption is not supported, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE --The above processing should continue until all 19 KEY-C pairs, as specified in Input Type 9, or all 19 KEY values, as specified in Input Type 10, are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 1.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.1.2.6 Monte Carlo Test for the Decryption Process - TECB Mode

Table 12 The Monte Carlo Test for the Decryption Process - TECB Mode

TMOVS: Initialize $KEY1_0$, $KEY2_0$, $KEY3_0$, C_0

Send $KEY1_0$, $KEY2_0$, $KEY3_0$, C_0

IUT: FOR i = 0 TO 399

{

Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, C_0

FOR j = 0 TO 9,999

{

Process Triple DES:

$I_j = C_j$

I_j is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1

TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in P_j

$C_{j+1} = P_j$

}

Record P_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, C_0, P_j

$KEY1_{i+1} = KEY1_i \oplus P_j$

If ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus P_{j-1}$

else

$KEY2_{i+1} = KEY2_i \oplus P_j$

If ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and

$KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus P_j$

else

$KEY3_{i+1} = KEY3_i \oplus P_{j-2}$

$C_0 = P_j$

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 12, the Monte Carlo Test for the TECB Decryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, and the ciphertext C variables. The C and KEYs consist of 64 bits.

   b. Forwards this information to the IUT using Input Type 20.

2. The IUT should perform the following for i = 0 through 399:

   a. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, and C_0.

   b. Perform the following for j = 0 through 9999:

      1) Set the input block I_j equal to the value of C_j.

      2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in plaintext P_j. This involves processing I_j through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in plaintext P_j.

      3) Prepare for loop j+1 by assigning C_{j+1} with the current value of P_j.

   c. Record P_j.

   d. Forward all recorded information for this loop, as specified in Output Type 5, to the TMOVS.

e. Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop. Note j = 9999.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the P_j.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. . If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the P_{j-1}. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the P_j.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the P_{j-2}. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the P_j.

f. Assign a new value to C in preparation for the next output loop. C_0 should be assigned the value of the current P_j. Note j = 9999.

NOTE -- the new C should be denoted as C_0 to be used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 5.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.2 Cipher Block Chaining (TCBC) Mode

The IUTs which implement the Cipher Block Chaining (TCBC) mode are validated by successfully completing a series of Known Answer tests and Monte Carlo tests corresponding to the cryptographic processes allowed by the IUT.

5.2.1 Encryption Process

The process of validating an IUT which implements the TCBC mode of operation in the encryption process should involve the successful completion of the following six tests:

1. The Variable Plaintext Known Answer Test - TCBC mode

2. The Inverse Permutation Known Answer Test - TCBC mode

3. The Variable Key Known Answer Test for the Encryption Process - TCBC mode

4. The Permutation Operation Known Answer Test for the Encryption Process - TCBC mode

5. The Substitution Table Known Answer Test for the Encryption Process - TCBC mode

6. The Monte Carlo Test for the Encryption Process - TCBC mode

An explanation of the tests follows.

5.2.1.1 The Variable Plaintext Known Answer Test - TCBC Mode

Table 13 The Variable Plaintext Known Answer Test - TCBC Mode

TMOVS:	Initialize	KEYs:	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
			IV = 0000000000000000
			P_1 = 8000000000000000
	Send		KEY (representing KEY1, KEY2, and KEY3), IV, P_1
IUT:	FOR i = 1 to 64		
	{		
	Perform Triple DES:		$I_i = P_i \oplus IV$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in C_i
			Send i, KEY (representing KEY1, KEY2, and KEY3), IV, P_i, C_i
			P_{i+1} = basis vector where single "1" bit is in position i+1
	}		
TMOVS:	Compare results from each loop with known answers. See Table A.1.		

Table 13 illustrates the Variable Plaintext Known Answer Test for the TCBC mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. Initializes the 64-bit plaintext P_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $P_{1\ bin}$ = 10000000 00000000

00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i=1 through 64:

a. Calculate the input block I_i by exclusive-ORing P_i with IV.

b. Process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in ciphertext C_i.

c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, P_i, and the resulting C_i to the TMOVS as specified in Output Type 2.

d. Retain C_i for use with the Inverse Permutation Known Answer Test for the TCBC Mode (Section 5.2.1.2), and, if the IUT supports the decryption process, for use with the Variable Ciphertext Known Answer Test for the TCBC Mode (Section 5.2.2.1).

e. Assign a new value to P_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i +1=2,...,64.

NOTE -- This continues until every possible basis vector has been represented by the P, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.1.

5.2.1.2 The Inverse Permutation Known Answer - TCBC Mode

Table 14 The Inverse Permutation Known Answer Test - TCBC Mode

TMOVS:	Initialize	KEYs:	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
			IV = 0000000000000000
			P_i (where i=1..64) = 64 C values from the Variable Plaintext Known Answer Test
	Send		KEY (representing KEY1, KEY2, and KEY3), IV, P_1,...,P_{64}
IUT:	FOR i = 1 to 64		
	{		
	Perform Triple DES:	$I_i = P_i \oplus IV$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in C_i	
		Send i, KEY (representing KEY1, KEY2, KEY3), IV, P_i, C_i	
		P_{i+1} = corresponding C_{i+1} from TMOVS	
	}		
TMOVS:	Compare results from each loop with known answers.		
	Should be the set of basis vectors.		

Table 14 illustrates the Inverse Permutation Known Answer Test for the TCBC mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

NOTE -- the significant bits are set to "0" and the parity bits are set to "1" to make odd parity.

b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c. Initializes the 64-bit plaintext P_i (where i=1..64) to the C_i results obtained from the Variable Plaintext Known Answer Test.

d. Forwards this information to the IUT using Input Type 5.

2. The IUT should perform the following for i=1 through 64:

   a. Calculate the input block I_i by exclusive-ORing P_i with IV.

   b. Process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in ciphertext C_i.

   c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, P_i, and the resulting C_i to the TMOVS as specified in Output Type 2.

   d. Assign a new value to P_{i+1} by setting it equal to the corresponding output from the TMOVS.

NOTE -- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The C values should be the set of basis vectors.

5.2.1.3 The Variable Key Known Answer Test for the Encryption Process - TCBC Mode

Table 15 The Variable Key Known Answer Test for the Encryption Process - TCBC Mode

TMOVS:	Initialize KEY_1:	$KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set) IV = 0000000000000000 P = 0000000000000000
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV, P

IUT:	FOR i = 1 to 64
	
	{
	
	IF (i mod 8 $\neq$ 0) {process every bit except parity bits}
	
	{
	
	Perform Triple DES: $I_i = P \oplus IV$
	
	I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1
	
	TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2
	
	TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
	
	Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, P, C_I
	
	$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1.
	
	NOTE -- that parity bits are "0" or "1" to make the KEYs odd parity.
	
	}
	
	}	

TMOVS:	Compare results of the 56 encryptions with known answers.	

Use Table A.2.

As summarized in Table 15, the Variable Key Known Answer Test for the TCBC Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

   NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. Initializes the 64-bit plaintext P to the value of 0, i.e., P_{hex} = 00 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i = 1 to 56:

   NOTE -- 56 is the number of significant bits in a TDES key.

   a. Calculate the input block I_i by exclusive-ORing P with IV.

   b. Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ parameters, process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in ciphertext C_i.

   c. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$,), IV, P, and the resulting C_i to the TMOVS as specified in Output Type 2.

   d. If the IUT supports the decryption process, retain C_i for use with the Variable KEY Known Answer Test for the Decryption Process for the TCBC Mode (Section 5.2.2.3).

e. Set $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1,}$, equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE -- The above processing continues until every significant basis vector has been represented by the KEY parameter. The output from the IUT for this test should consist of 56 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.2.

5.2.1.4 Permutation Operation Known Answer Test for the Encryption Process - TCBC Mode

Table 16 The Permutation Operation Known Answer Test for the Encryption Process - TCBC Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..32) = 32 KEY values in Table A.3
		IV = 0000000000000000
		P = 0000000000000000
	Send	P, IV, KEY_1, KEY_2,...,KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2 and KEY3.)
IUT:	FOR i = 1 to 32	
	{	
	Perform Triple DES:	$I_i = P \oplus IV$ I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, P, C_i
		$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS
	}	
TMOVS:	Compare results with known answers. Use Table A.3.	

Table 16 illustrates the Permutation Operation Known Answer Test for the TCBC Encryption Process.

1. The TMOVS:

   a. Initializes the KEY1, KEY2, and KEY3 variables with the 32 constant KEY values from Table A.3.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c. Initializes the plaintext P to the value of 0, i.e., P_{hex}=00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 8.

2. The IUT should perform the following for i = 1 to 32:

a. Calculate the input block I_i by exclusive-ORing P with IV.

b. Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in ciphertext C_i.

c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, P, and the resulting C_i to the TMOVS as specified in Output Type 2.

d. If the IUT supports the decryption process, retain C_i for use with the Permutation Operation Known Answer Test for the Decryption Process for the TCBC mode (Section 5.2.2.4).

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the next key supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.3.

5.2.1.5 Substitution Table Known Answer Test for the Encryption Process - TCBC Mode

Table 17 The Substitution Table Known Answer Test for the Encryption Process - TCBC Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$(where i=1..19) = 19 KEY values in Table A.4
		P_i (where i=1..19) = 19 corresponding P values in Table A.4
		IV = 0000000000000000
	Send	IV, 19, KEY_1, P_1, KEY_2, P_2,...,KEY_{19}, P_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2 and KEY3.)
IUT:	FOR i = 1 to 19	
	{	
	Perform Triple DES:	$I_i = P_i \oplus IV$ I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_i
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, P_i, C_i
		$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS
		P_{i+1} = corresponding P_{i+1} from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. Use Table A.4.	

As summarized in Table 17, the Substitution Table Known Answer Test for the TCBC Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY-plaintext (KEY-P) pairs with the 19 constant KEY-P values from Table A.4. The KEY value indicates the value of KEY1, KEY2, and KEY3, i.e., KEY1=KEY2=KEY3.

b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c. Forwards this information to the IUT using Input Type 11.

2. The IUT should perform the following for i = 1 to 19:

a. Calculate the input block I_i by exclusive-ORing P_i with IV.

b. Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in ciphertext C_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in ciphertext C_i.

c. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, P_i, and the resulting C_i to the TMOVS as specified in Output Type 2.

d. If the IUT supports the decryption process, retain C_i for use with the Substitution Table Known Answer Test for the Decryption Process for the TCBC mode (Section 5.2.2.5).

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the KEY_{i+1} supplied by the TMOVS.

f. Set P_{i+1} equal to the corresponding P_{i+1} value supplied by the TMOVS.

NOTE -- The above processing should continue until all 19 KEY-P pairs are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.4.

5.2.1.6 Monte Carlo Test for the Encryption Process - TCBC Mode

Table 18 The Monte Carlo Test for the Encryption Process - TCBC Mode

TMOVS: Initialize $KEY1_0$, $KEY2_0$, $KEY3_0$, IV, P_0

Send $KEY1_0$, $KEY2_0$, $KEY3_0$, IV, P_0

IUT: FOR i = 0 TO 399

{

If (i==0) CV_0 = IV

Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0, P_0

FOR j = 0 TO 9,999

{

Process Triple DES:

$I_j = P_j \oplus CV_j$

I_j is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1

TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in C_j

IF j=0

$P_{j+1} = CV_0$

ELSE

$P_{j+1} = C_{j-1}$

$CV_{j+1} = C_j$

}

Record C_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0, P_0, C_j

$KEY1_{i+1} = KEY1_i \oplus C_j$

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i$ = $KEY1_i$) or ($KEY1_i$,

$KEY2_i$, and $KEY3_i$ are independent)

$KEY2_{i+1} = KEY2_i \oplus C_{j-1}$

ELSE

$KEY2_{i+1} = KEY2_i \oplus C_j$

IF ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_I$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$)

$KEY3_{i+1} = KEY3_i \oplus C_j$

ELSE

$KEY3_{i+1} = KEY3_i \oplus C_{j-2}$

$P_0 = C_{j-1}$

$CV_0 = C_j$

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 18, the Monte Carlo Test for the TCBC Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vector IV, and the plaintext P variables. The P, IV, and KEYs consist of 64 bits each.

   b. Forwards this information to the IUT using Input Type 21.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), set the chaining value CV_0 equal to the IV.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0, and P_0.

   c. Perform the following for j = 0 through 9999:

      1) Calculate the input block I_j by exclusive-ORing P_j with CV_j.

2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in ciphertext C_j. This involves processing I_j through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in ciphertext C_j.

3) Prepare for loop j+1 by doing the following:

   a) If the inner loop being processed in the first loop, i.e., j=0, assign P_{j+1} with the current value of CV_0. Otherwise, assign P_{j+1} with the C from the previous inner cycle, C_{j-1}.

   b) Assign CV_{j+1} with the current value of C_j.

d. Record the C_j.

e. Forward all recorded information from this loop, as specified in Output Type 6, to the TMOVS.

f. In preparation for the next outer loop (Note j = 9999):

1) Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the C_j.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the C_{j-1}. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the C_j.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the C_{j-2}. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the C_j.

2) Assign a new value to P_0 in preparation for the next output loop. P_0 should be assigned the value of C_{j-1}.

NOTE -- the new P should be denoted as P_0 to be used for the first pass through the inner loop when j=0.

3) Assign a new value to CV_0 in preparation for the next outer loop. CV_0 should be assigned the value of C_j.

NOTE -- the new CV should be denoted as CV_0 because this value is used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 6.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.2.2 Decryption Process

The process of validating an IUT for the TCBC mode which implements the decryption process involves the successful completion of the following six tests:

1. The Variable Ciphertext Known Answer Test - TCBC mode

2. The Initial Permutation Known Answer Test - TCBC mode

3. The Variable Key Known Answer Test for the Decryption Process - TCBC mode

4. The Permutation Operation Known Answer Test for the Decryption Process - TCBC mode

5. The Substitution Table Known Answer Test for the Decryption Process - TCBC mode

6. The Monte Carlo Test for the Decryption Process - TCBC mode

An explanation of the tests follows.

5.2.2.1 The Variable Ciphertext Known Answer Test - TCBC Mode

Table 19 The Variable Ciphertext Known Answer Test - TCBC Mode

TMOVS: Initialize KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)

IV = 0000000000000000

If encryption is supported by the IUT:

Send KEY (representing KEY1, KEY2, and KEY3), IV

If encryption is not supported by the IUT:

Initialize C_i (where i=1..64) = C values in Table A.1

Send KEY (representing KEY1, KEY2, and KEY3), IV, C_1, C_2,...,C_{64}

IUT: If encryption is supported

Initialize C_1= first value from output of Variable Plaintext Known Answer Test.

Otherwise, use the first value received from the TMOVS.

FOR i = 1 to 64

{

Perform Triple DES:

$I_i = C_i$

I_i is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in TEMP1

TEMP1 is encrypted by DEA_2 using KEY2, resulting in TEMP2

TEMP2 is decrypted by DEA_1 using KEY1, resulting in O_i

$P_i = O_i \oplus IV$

Send i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, P_i

If encryption is supported:

C_{i+1} = corresponding C_{i+1} from output of Variable Plaintext Known Answer Test

else

C_{i+1} = corresponding C_{i+1} value from TMOVS

}

TMOVS: Compare results from each loop with known answers. Should be the set of basis vectors.

Table 19 illustrates the Variable Ciphertext Known Answer Test for the TCBC mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. If the IUT does not support encryption, the 64 constant ciphertext values are initialized with the 64 constant C values from Table A.1.

   d. If encryption is supported by the IUT, the KEYs and the IV are forwarded to the IUT, as specified in Input Type 6. If encryption is not supported by the IUT, the KEYs, the IV, and 64 C values are forwarded to the IUT using Input Type 5.

2. The IUT should:

   a. If encryption is supported, initialize the C value with the first C value retained from the Variable Plaintext Known Answer Test for the TCBC Mode (Section 5.2.1.1). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i=1 through 64:

      1) Set the input block I_i equal to the value of C_i.

      2) Process I_i through the three DEA stages resulting in the output block O_i. This involves processing I_i through the DEA stage DEA_3, in the decrypt state using KEY3, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2, in the encrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1, in the decrypt state using KEY1, resulting in output block O_i.

      3) Calculate the plaintext P_i by exclusive-ORing O_i with IV.

      4) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, and the resulting P_i to the TMOVS as specified in Output Type 2.

5) Retain P_i for use with the Initial Permutation Known Answer Test for the TCBC Mode (Section 5.2.2.2).

6) If encryption is supported, set C_{i+1} equal to the corresponding output from the Variable Plaintext Known Answer Test for the TCBC mode. If encryption is not supported, assign a new value to C_{i+1} by setting it equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The output from the IUT for this test should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The P results should be the set of basis vectors.

5.2.2.2 The Initial Permutation Known Answer Test - TCBC Mode

Table 20 The Initial Permutation Known Answer Test - TCBC Mode

TMOVS:	Initialize	KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) IV = 0000000000000000 C_i (where i=1..64) = 64 P values from Variable Ciphertext Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), IV, $C_1..C_{64}$
IUT:	FOR i = 1 to 64 {	
	Perform Triple DES:	$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in TEMP1 TEMP1 is encrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is decrypted by DEA_1 using KEY1, resulting in O_i $P_i = O_i \oplus IV$
		Send i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, P_i C_{i+1} = corresponding C_{i+1} value from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers.	

Table 20 illustrates the Initial Permutation Known Answer Test for the TCBC mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c Initializes the 64 C values with the 64 P values obtained from the Variable Ciphertext Known Answer Test.

d. Forwards the KEY (representing KEY1, KEY2, and KEY3), IV, and the 64 C values to the IUT using Input Type 5.

2. The IUT should perform the following for i=1 through 64:

   a. Set the input block I_i equal to the value of C_i.

   b. Process I_i through the three DEA stages resulting in the output block O_i. This involves processing I_i through the DEA stageDEA_3 in the decrypt state using KEY3, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using KEY1, resulting in O_i.

   c. Calculate the plaintext P_i by exclusive-ORing O_i with IV.

   d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, and the resulting P_i to the TMOVS as specified in Output Type 2.

   e. Set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.2.2.3 The Variable Key Known Answer Test for the Decryption Process - TCBC Mode

Table 21 The Variable Key Known Answer Test for the Decryption Process - TCBC Mode

TMOVS:	Initialize	KEYs: $KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set) IV=0000000000000000
	If encryption is supported by the IUT:	
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV
	If encryption is not supported by the IUT:	
	Initialize	C_i values (where i=1..56): C values in Table A.2
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV, C_1, C_2,..., C_{56}
IUT:	If encryption is supported by the IUT:	
	Initialize	C_1 = first value from output of Variable Key Known Answer Test for the Encryption Process
	Otherwise, use the first value received from the TMOVS.	

	FOR i = 1 to 64
	{
	IF (i mod 8 $\neq$ 0) {process every bit except parity bits}
	{	

	Perform Triple DES:	$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1 TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in O_i $P_i = O_i \oplus IV$
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, C_i, P_i

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} =$ vector consisting of "0" in every significant bit position except for a single "1" bit in the $i+1^{th}$ position.
NOTE -- odd parity is set.

If encryption is supported by the IUT:

C_{i+1} = corresponding C_{i+1} from output of Variable Key Known Answer Test for the Encryption Process

else

C_{i+1} = corresponding C_{i+1} value from TMOVS

}

}

TMOVS: Compare results of the 56 decryptions with known answers.

Should be P=0000000000000000 for all 56 rounds.

Table 21 illustrates the Variable Key Known Answer Test for the TCBC Decryption Process.

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

   NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. If the IUT does not support encryption, the C_i values are initialized with the 56 constant C values from Table A.2.

   d. If encryption is not supported by the IUT, the KEY (representing KEY1, KEY2, and KEY3), IV, and the 56 C values are forwarded to the IUT, as specified in Input Type 5. Otherwise, the KEY1, KEY2, KEY3, and IV are forwarded to the IUT, as specified in Input Type 6.

2. The IUT should:

a. If encryption is supported, initialize the C_1 value with the first C value retained from the Variable KEY Known Answer Test for the Encryption Process for the TCBC Mode (Section 5.2.1.3). Otherwise, use the first value received from the TMOVS.

b. Perform the following for i = 1 to 56:

NOTE -- 56 is the number of significant bits in a TDES key.

1) Set the input block I_i equal to the value of C_i.

2) Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ parameters, process I_i through the three DEA stages resulting in the output block O_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in O_i.

3) Calculate the plaintext P_i by exclusive-ORing O_i with IV.

4) Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, C_i, and the resulting P_i to the TMOVS as specified in Output Type 2.

5) Set $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1,}$, equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE -- $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$.

6) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Variable Key Known Answer Test for the Encryption Process for TCBC mode. If encryption is not supported by the IUT, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The output from the IUT for this test should consist of 56 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The P results should be all zeros.

5.2.2.4 Permutation Operation Known Answer Test for Decryption Process - TCBC Mode

Table 22 The Permutation Operation Known Answer Test for the Decryption Process - TCBC Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..32) = 32 KEY values in Table A.3

IV = 0000000000000000

If encryption is supported by the IUT:

Send IV, KEY_1, KEY_2,...,KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)

If encryption not supported by the IUT:

Initialize C_i (where i=1..32) = corresponding C values in Table A.3

Send IV, KEY_1, C_1, KEY_2, C_2,...,KEY_{32}, C_{32} (The key values represent the values of KEY1, KEY2, and KEY3.)

IUT: If encryption is supported by the IUT:

Initialize C_1 = first value retained from Permutation Operation Known Answer Test for the Encryption Process

Otherwise, use the first value received from the TMOVS.

FOR i = 1 to 32

{

Perform Triple DES:

$I_i = C_i$

I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1

TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in O_i

$P_i = O_i \oplus IV$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, C_i, P_i

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} supplied by TMOVS

If encryption is supported:

C_{i+1} = corresponding C_{i+1} from output of Permutation Operation Known Answer Test for the Encryption Process

else

C_{i+1} = corresponding C_{i+1} from TMOVS

}

TMOVS: Compare results from each loop with known answers.

Should be P=0000000000000000 for all 32 rounds.

Table 22 illustrates the Permutation Operation Known Answer Test for the TCBC Decryption Process.

1. The TMOVS:

   a. If the IUT supports encryption, the KEY1, KEY2, and KEY3 variables are initialized with the 32 constant KEY values from Table A.3. If the IUT does not support encryption, the KEY-ciphertext (KEY-C) pairs are initialized with the 32 constant KEY-C pairs from Table A.3.

      NOTE -- KEY1=KEY2=KEY3.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. If encryption is supported by the IUT, the 32 KEY values for KEY1, KEY2, and KEY3, and the IV value are forwarded to the IUT using Input Type 12. If encryption is not supported by the IUT, the 32 KEY and C pairs and the IV value are forwarded to the IUT using Input Type 11.

2. The IUT should:

   a. If encryption is supported, initialize the C_i value with the first C value retained from the Permutation Operation Known Answer Test for the Encryption Process for the TCBC Mode (Section 5.2.1.4). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i = 1 to 32:

      1) Set the input block I_i equal to the value of C_i.

      2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in output block O_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in O_i.

3) Calculate the plaintext P_i by exclusive-ORing O_i with IV.

4) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, and the resulting P_i to the TMOVS as specified in Output Type 2.

5) Assign a new value to $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ by setting them equal to the corresponding KEY_{i+1} value supplied by the TMOVS.

NOTE -- KEY1=KEY2=KEY3.

6) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Permutation Operation Known Answer Test for the Encryption Process for the TCBC mode. If encryption is not supported, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY-C values are passed as specified in Input Type 11, or all 32 KEY values are passed as specified in Input Type 12. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The P results should be all zeros.

5.2.2.5 Substitution Table Known Answer Test for the Decryption Process - TCBC Mode

Table 23 The Substitution Table Known Answer Test for the Decryption Process - TCBC Mode

TMOVS:	Initialize	$KEY1_i$, $KEY2_i$, $KEY3_i$ (where i=1..19)= 19 KEY values in Table A.4
		IV = 0000000000000000
	If encryption is supported by the IUT:	
	Send	IV, KEY_1, KEY_2,...,KEY_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
	If encryption not supported:	
	Initialize	C_i (where i=1..19)= 19 C values in Table A.4
	Send	IV, KEY_1, C_1, KEY_2, C_2,...,KEY_{19}, C_{19} (These key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	If encryption is supported:	
	Initialize	C_1 = first C value retained from the Substitution Table Known Answer Test for the Encryption Process.
	Otherwise, use the first value received from the TMOVS	
	FOR i = 1 to 19	
	{	
	Perform Triple DES:	$I_i = C_i$ I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1 TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in O_i $P_i=O_i \oplus IV$
		Send i, KEY_i, IV, C_i, P_i (where KEY_i represents the value of KEY1, KEY2 and KEY3)
		$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} supplied by TMOVS

If encryption is supported:

C_{i+1} = corresponding C from output of Substitution Table Known Answer Test for the Encryption Process for the TCBC mode

else

C_{i+1} = corresponding C_{i+1} from TMOVS

}

TMOVS: Compare results from each loop with known answers. See Table A.4.

As summarized in Table 23, the Substitution Table Known Answer Test for the TCBC Decryption Process is performed as follows:

1. The TMOVS:

   a. If the IUT supports encryption, the KEY1, KEY2, and KEY3 variables are initialized with the 19 constant KEY values from Table A.4. If the IUT does not support encryption, the KEY-ciphertext (KEY-C) pairs are initialized with the 19 constant KEY-C pairs from Table A.4.

      NOTE -- KEY1=KEY2=KEY3.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. If encryption is supported by the IUT, the IV and the 19 KEY values for KEY1, KEY2, and KEY3 are forwarded to the IUT using Input Type 12. Otherwise, if encryption is not supported by the IUT, the IV and the 19 KEY-C pairs are forwarded to the IUT using Input Type 11.

2. The IUT should:

   a. If encryption is supported, initialize the C_1 value with the first C value retained from the Substitution Table Known Answer Test for the Encryption Process for the TCBC Mode (Section 5.2.1.5). Otherwise, use the first C value received from the TMOVS.

   b. Perform the following for i = 1 to 19:

      1) Set the input block I_i equal to the value of C_i.

      2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_i through the three DEA stages resulting in the output block O_i. This involves processing I_i through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA

stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in O_i.

3) Calculate the plaintext P_i by exclusive-ORing O_i with IV.

4) Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, C_i, and the resulting P_i to the TMOVS as specified in Output Type 2.

5) Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the next key supplied by the TMOVS.

6) If encryption is supported, set C_{i+1} equal to the corresponding C_{i+1} value retained from the Substitution Table Known Answer Test for the Encryption Process for the TCBC mode. If encryption is not supported, set C_{i+1} equal to the corresponding C_{i+1} value supplied by the TMOVS.

NOTE -- The above processing should continue until all 19 KEY-C pairs, as specified in Input Type 11, or all 19 KEY values, as specified in Input Type 12, are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.2.2.6 Monte Carlo Test for the Decryption Process - TCBC Mode

Table 24 The Monte Carlo Test for the Decryption Process - TCBC Mode

TMOVS: Initialize $KEY1_0$, $KEY2_0$, $KEY3_0$, IV, C_0

Send $KEY1_0$, $KEY2_0$, $KEY3_0$, IV, C_0

IUT: FOR i = 0 TO 399

{

If (i==0) CV_0 = IV

Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0, C_0

FOR j = 0 TO 9,999

{

Perform Triple DES:

$I_j = C_j$

I_i is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in TEMP1

TEMP1 is encrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is decrypted by DEA_1 using $KEY1_i$, resulting in O_j

$P_j = O_j \oplus CV_j$

$CV_{j+1} = C_j$

$C_{j+1} = P_j$

}

Record P_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0, C_0, P_j

$KEY1_{i+1} = KEY1_i \oplus P_j$

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus P_{j-1}$

ELSE

$KEY2_{i+1} = KEY2_i \oplus P_j$

IF ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus P_j$

ELSE

$KEY3_{i+1} = KEY3_i \oplus P_{j-2}$

$CV_0 = C_j$

$C_0 = P_j$

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 24, the Monte Carlo Test for the TCBC Decryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vector IV, and the ciphertext C variables. All variables consist of 64 bits.

   b. Forwards this information to the IUT using Input Type 21.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), set the chaining value CV_0 equal to IV.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, CV_0 and C_0.

   c. Perform the following for j = 0 through 9999:

      1) Set the input block I_j equal to the value of C_j.

      2) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in the output block O_j. This involves processing I_j through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the DEA

stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in O_j.

3) Calculate the plaintext P_j by exclusive-ORing O_j with CV_j.

4) Prepare for loop j+1 by:

   a) Assigning CV_{j+1} with the current value of C_j.

   b) Assigning C_{j+1} with the current value of P_j.

d. Record $P_{j.}$

e. Forward all recorded information for this loop, as specified in Output Type 6 to the TMOVS.

f. Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop. Note j = 9999.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the P_j.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the P_{j-1}. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the P_j.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the P_{j-2}. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the P_j.

g. Assign a new value to CV_0 in preparation for the next outer loop. CV_0 should be assigned the value of the current C_j. Note j = 9999.

NOTE -- the new CV should be denoted as CV_0 to be used for the first pass through the inner loop when j=0.

h. Assign a new value to C_0 in preparation for the next outer loop. C_0 should be assigned the value of the current P_j. Note j = 9999.

NOTE -- the new C should be denoted as C_0 to be used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 6.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.3 Cipher Block Chaining Mode - Interleaved (TCBC-I)

The IUTs in the Cipher Block Chaining mode - Interleaved (TCBC-I) are validated by successfully completing a series of Known Answer tests and Monte Carlo tests corresponding to the cryptographic processes allowed by the IUT.

The interleaved configuration is intended for systems equipped with multiple DEA processors. By interleaving the data, throughput is improved and propagation delay is minimized by initializing the three individual DEA stages and then simultaneously clocking them. Thus, with each clock cycle, data is processed by each DEA_i stage (where i = 1, 2, 3) and passed onward to the output buffer or the next stage so that idle DEA_i stages are minimized.

The processing for each Known Answer test and Monte Carlo Test is broken down into clock cycles T1, T2, T3.... Within each clock cycle, the processing occurring on each active DEA is discussed. For convenience, let KEY1 represent the key used on processor DEA_1, KEY2 represent the key used on processor DEA_2, and KEY3 represent the key used on processor DEA_3.

5.3.1 Encryption Process

The process of validating an IUT which implements the TCBC-I mode of operation in the encryption process involves the successful completion of the following six tests:

1. The Variable Plaintext Known Answer Test - TCBC-I mode

2. The Inverse Permutation Known Answer Test - TCBC-I mode

3. The Variable Key Known Answer Test for the Encryption Process - TCBC-I mode

4. The Permutation Operation Known Answer Test for the Encryption Process - TCBC-I mode

5. The Substitution Table Known Answer Test for the Encryption Process - TCBC-I mode

6. The Monte Carlo Test for the Encryption Process - TCBC-I mode

An explanation of the tests follows.

5.3.1.1 The Variable Plaintext Known Answer Test - TCBC-I Mode

Table 25 The Variable Plaintext Known Answer Test - TCBC-I Mode

TMOVS:	Initialize		KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) IV1=0000000000000000 IV2 = 5555555555555555 (based on specifications in ANSI X9.52 - 1998) IV3 = AAAAAAAAAAAAAAAA (based on specifications in ANSI X9.52 - 1998) $P1_1 = P2_1 = P3_1$ = 8000000000000000
	Send		KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $P1_1$, $P2_1$, $P3_1$
IUT:	FOR i = 1 to 64 {		
	Perform Triple DES:	T1:	$I1_i = P1_i \oplus IV1$ $I1_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$
		T2:	$I2_i = P2_i \oplus IV2$ $I2_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$
		T3:	$I3_i = P3_i \oplus IV3$ $I3_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$ $TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $C1_i$
		T4:	$TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP3_2$

	$TEMP2_2$ is encrypted by DEA_3 using KEY3, resulting in $C2_i$
T5:	$TEMP3_2$ is encrypted by DEA_3 using KEY3, resulting in $C3_i$

Send i, KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $I1_i$, $I2_i$, and $I3_i$, $C1_i$,$C2_i$,$C3_i$

$P1_{i+1} = P2_{i+1} = P3_{i+1}$ = basis vector where single "1" bit is in position i+1

}

TMOVS: Compare results from each loop with known answers. See Table A.5.

Table 25 illustrates the Variable Plaintext Known Answer Test for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. Initializes the 64-bit plaintext values $P1_1$, $P2_1$, $P3_1$ to the basis vector containing a "1" in the first bit position and "0" in the following 63 positions, i.e., $P1_{1\ bin}$ = $P2_{1\ bin}$ = $P3_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 64:

NOTE -- the processing for each clock cycle T*i* is displayed.

   a. At clock cycle T1:

      1) Calculate the input block $I1_i$ by exclusive-ORing $P1_i$ with IV1.

      2) Process $I1_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

1) Calculate the input block $I2_i$ by exclusive-ORing $P2_i$ with IV2.

2) Process $I2_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Calculate the input block $I3_i$ by exclusive-ORing $P3_i$ with IV3.

2) Process $I3_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in the ciphertext value $C1_i$.

At clock cycle T4:

1) Process $TEMP2_2$ through the third DEA stage, denoted DEA_2, in the encrypt state using KEY3, resulting in the ciphertext value $C2_i$.

2) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in the ciphertext value $C3_i$.

b. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $I1_i$, $I2_i$, $I3_i$, and the resulting $C1_i$, $C2_i$, and $C3_i$, to the TMOVS as specified in Output Type 3.

c. Retain $C1_i$, $C2_i$, and $C3_i$, for use with the Inverse Permutation Known Answer Test for the TCBC-I Mode (Section 5.3.1.2), and, if the IUT supports the decryption process, for use with the Variable Ciphertext Known Answer Test for the TCBC-I Mode (Section 5.3.2.1).

d. Assign a new value to $P1_{i+1}$, $P2_{i+1}$, and $P3_{i+1}$, by setting them equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64.

NOTE -- This continues until every possible basis vector has been represented by the P variables, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received C1 results to the known values found in Table A.5.

5.3.1.2 The Inverse Permutation Known Answer Test - TCBC-I Mode

Table 26 The Inverse Permutation Known Answer Test - TCBC-I Mode

TMOVS:	Initialize		KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
			IV1=0000000000000000
			IV2 = 5555555555555555 (based on specifications in ANSI X9.52 - 1998)
			IV3 = AAAAAAAAAAAAAAAA (based on specifications in ANSI X9.52 - 1998)
			Pk_i (where k=1..3 and i=1..64) = 64 corresponding Ck_i values from Variable Plaintext Known Answer Test
	Send		KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $P1_1,...,P1_{64}$, $P2_1,...,P2_{64}$, $P3_1,...,P3_{64}$
IUT:	FOR i = 1 to 64		
	{		
	Perform Triple DES:	T1:	$I1_i = P1_i \oplus IV1$
			$I1_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$
		T2:	$I2_i = P2_i \oplus IV2$
			$I2_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$
			$TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$
		T3:	$I3_i = P3_i \oplus IV3$
			$I3_i$ is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$
			$TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$
			$TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $C1_i$
		T4:	$TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in

TEMP3$_2$

TEMP2$_2$ is encrypted by DEA$_3$ using KEY3, resulting in C2$_i$

T5: TEMP3$_2$ is encrypted by DEA$_3$ using KEY3, resulting in C3$_i$

Send i, Key (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, P1$_i$, P2$_i$, P3$_i$, C1$_i$,C2$_i$,C3$_i$

Pk$_{i+1}$ (where k=1..3) = corresponding Ck$_{i+1}$ from TMOVS

}

TMOVS: Compare C1, C2, and C3 results from each loop with known answers. See Table A.6.

C1 should be the set of basis vectors.

Table 26 illustrates the Inverse Permutation Known Answer Test for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., KEY1$_{hex}$=KEY2$_{hex}$=KEY3$_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., IV1$_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R$_1$ mod 2^{64} where R$_1$=5555555555555555, i.e., IV2$_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R$_2$ mod 2^{64} where R$_2$=AAAAAAAAAAAAAAAA, i.e., IV3$_{hex}$ = AA AA AA AA AA AA AA AA.

   c. Initializes the 64-bit plaintext values P1, P2, P3 to the 64-bit ciphertext values C1, C2, and C3 respectively, obtained from the Variable Plaintext Known Answer Test.

   d. Forwards this information to the IUT using Input Type 15.

2. The IUT should perform the following for i=1 through 64:

NOTE -- the processing for each clock cycle T*i* is displayed.

   a. At clock cycle T1:

      1) Calculate the input block I1$_i$ by exclusive-ORing P1$_i$ with IV1.

      2) Process I1$_i$ through the first DEA stage, denoted DEA$_1$, in the encrypt state using KEY1, resulting in intermediate value TEMP1$_1$.

At clock cycle T2:

1) Calculate the input block $I2_i$ by exclusive-ORing $P2_i$ with IV2.

2) Process $I2_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Calculate the input block $I3_i$ by exclusive-ORing $P3_i$ with IV3.

2) Process $I3_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in the ciphertext value $C1_i$.

At clock cycle T4:

1) Process $TEMP2_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in the ciphertext value $C2_i$.

2) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in the ciphertext value $C3_i$.

b. Forward the current values of the loop number i, KEY (which represents KEY1, KEY2, and KEY3), IV1, IV2, IV3, $P1_i$, $P2_i$, and $P3_i$, and the resulting $C1_i$, $C2_i$, and $C3_i$, to the TMOVS as specified in Output Type 3.

c. Assign a new value to the plaintext values, $P1_{i+1}$, $P2_{i+1}$, and $P3_{i+1}$, by setting them equal to the corresponding output from the TMOVS.

NOTE -- This processing continues until all ciphertext values from the Variable Plaintext Known Answer Test have been used as input. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received C1, C2, and C3 results to known values. The C1 values should be the set of basis vectors. See Table A.6.

5.3.1.3 The Variable Key Known Answer Test for the Encryption Process - TCBC-I Mode

Table 27 The Variable Key Known Answer Test for the Encryption Process - TCBC-I Mode

TMOVS:	Initialize		$KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set) IV1=0000000000000000 IV2 = 5555555555555555 IV3 = AAAAAAAAAAAAAAAA P1 = P2 = P3 = 0000000000000000
	Send		KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV1, IV2, IV3, P1, P2, P3

IUT:	FOR i = 1 to 64
	
	{
	
	IF (i mod 8 ≠ 0) {process every bit except parity bits}
	
	{		

	Perform Triple DES:	T1:	$I1_i$= P1 ⊕ IV1 $I1_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$
		T2:	$I2_i$= P2 ⊕ IV2 $I2_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$
		T3:	$I3_i$= P3 ⊕ IV3 $I3_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$

		$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C1_i$
	T4:	$TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$
		$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C2_i$
	T5:	$TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C3_i$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, P1, P2, P3, $C1_i$, $C2_i$, $C3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1.

NOTE -- the parity bits are "0" or "1" to set odd parity.

}

}

TMOVS: Compare results of the 3 triple DES encryptions per 56 different keys with known answers. See Table A.11.

Table 27 illustrates the Variable Key Known Answer Test for the Encryption Process for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin} = KEY2_{1\ bin} = KEY3_{1\ bin}$ = 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

NOTE -- the parity bits are set to "0" or"1" to get odd parity.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

c. Initializes the P parameters P1, P2, and P3 to the constant hexadecimal value 0, i.e., $P1_{hex}=P2_{hex}=P3_{hex}$=00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 56:

NOTE -- 56 is the number of significant bits in a TDES key.

NOTE -- the processing for each clock cycle T*i* is displayed.

a. At clock cycle T1:

1) Calculate the input block $I1_i$ by exclusive-ORing P1 with IV1.

2) Process $I1_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

1) Calculate the input block $I2_i$ by exclusive-ORing P2 with IV2.

2) Process $I2_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Calculate the input block $I3_i$ by exclusive-ORing P3 with IV3.

2) Process $I3_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C1_i$.

At clock cycle T4:

1) Process $TEMP2_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C2_i$.

2) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C3_i$.

b. Forward the current values of the loop number i, KEY_i (which represents $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, P1, P2, and P3, and the resulting $C1_i$, $C2_i$, and $C3_i$, to the TMOVS as specified in Output Type 3.

c. If the IUT supports the decryption process, retain C1, C2, and C3 for use with the Variable KEY Known Answer Test for the Decryption Process for the TCBC-I Mode (Section 5.3.2.3).

d. Assign a new value to $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$, by setting them equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE -- The above processing continues until every significant basis vector has been represented by the KEY parameters, i.e., 56 times. The output from the IUT should consist of 56 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values found in Table A.11.

5.3.1.4 Permutation Operation Known Answer Test for the Encryption Process - TCBC-I Mode

Table 28 The Permutation Operation Known Answer Test for the Encryption Process - TCBC-I Mode

TMOVS:	Initialize	$KEY1_i$ =$KEY2_i$ = $KEY3_i$ (where i=1..32) = 32 KEY values from Table A.12
		IV1=0000000000000000
		IV2 = 5555555555555555
		IV3 = AAAAAAAAAAAAAAAA
		P1 = P2 = P3 = 0000000000000000
	Send	P (where P represents the values of P1, P2, and P3),
		IV1, IV2, and IV3,
		KEY_1, KEY_2,..., KEY_{32} (where KEY represents the values of KEY1, KEY2, and KEY3)
IUT:	FOR i = 1 to 32	
	{	
	Process Triple DES:	T1: $I1_i$= P1 ⊕ IV1
		$I1_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$
		T2: $I2_i$= P2 ⊕ IV2
		$I2_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$
		$TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$
		T3: $I3_i$= P3 ⊕ IV3
		$I3_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$
		$TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$

	$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C1_i$ T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$ $TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C2_i$ T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C3_i$ Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, P1, P2, P3, $C1_i$,$C2_I$,$C3_i$ $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS
}	
TMOVS: Compare results with known answers. See Table A.12.	

Table 28 illustrates the Permutation Operation Known Answer Test for the Encryption Process for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, $KEY3_1$ to the 32 constant KEY values from Table A.12.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. Initializes the P parameters P1, P2, and P3 to the constant hexadecimal value 0, i.e., $P1_{hex}$=$P2_{hex}$=$P3_{hex}$=00 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 18.

2. The IUT should perform the following for i=1 through 32:

NOTE -- that the processing for each clock cycle T*i* is displayed.

   a. At clock cycle T1:

      1) Calculate the input block $I1_i$ by exclusive-ORing P1 with IV1.

2) Process $I1_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

1) Calculate the input block $I2_i$ by exclusive-ORing P2 with IV2.

2) Process $I2_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Calculate the input block $I3_i$ by exclusive-ORing P3 with IV3.

2) Process $I3_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C1_i$.

5) Set ciphertext1 $C1_i$ equal to the value of $O1_i$.

At clock cycle T4:

1) Process $TEMP2_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C2_i$.

2) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C3_i$.

b. Forward the current values of the loop number i, KEY (which represents KEY1, KEY2, and KEY3), IV1, IV2, IV3, $P1_i$, $P2_i$, and $P3_i$, and the resulting $C1_i$, $C2_i$, and $C3_i$, to the TMOVS as specified in Output Type 3.

c. If the IUT supports the decryption process, retain $C1_i$, $C2_i$, and $C3_i$ for use with the Permutation Operation Known Answer Test for the Decryption Process for the TCBC-I Mode (Section 5.3.2.4).

d. Assign a new value to $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$, by setting them equal to the next key supplied by the TMOVS.

NOTE -- The above processing continues until all 32 KEY values are processed. The output from the IUT should consist of 32 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.12.

5.3.1.5 Substitution Table Known Answer Test for the Encryption Process - TCBC-I Mode

Table 29 The Substitution Table Known Answer Test for the Encryption Process - TCBC-I Mode

TMOVS:	Initialize		$KEY1_i = KEY2_i = KEY3_i$ (where i=1..19) = 19 KEY values from Table A.8 $P1_i = P2_i = P3_i$ (where i=1..19) = 19 corresponding P values from Table A.8 IV1 = 0000000000000000 IV2 = 5555555555555555 IV3 = AAAAAAAAAAAAAAAA
	Send		IV1, IV2, and IV3, $P_1, P_2, ..., P_{19}$(where P represents the values of P1, P2, and P3), $KEY_1, KEY_2, ..., KEY_{19}$ (where KEY represents the values of KEY1, KEY2, and KEY3)
IUT:	FOR i = 1 to 19 {		
	Perform Triple DES:	T1:	$I1_i = P1_i \oplus IV1$ $I1_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$
		T2:	$I2_i = P2_i \oplus IV2$ $I2_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$
		T3:	$I3_i = P3_i \oplus IV3$ $I3_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$
			$TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in

$TEMP2_2$

$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C1_i$

T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C2_i$

T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C3_i$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $P1_i$, $P2_i$, $P3_i$, $C1_i$,$C2_i$,$C3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS

$P1_{i+1} = P2_{i+1} = P3_{i+1} =$ corresponding P_{i+1} from TMOVS

}

TMOVS: Compare results from each loop with known answers. See Table A.8.

Table 29 illustrates the Substitution Table Known Answer Test for the Encryption Process of the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, KEY3 to the 19 constant KEY values from Table A.8.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: $IV1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation $IV1 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. Initializes the P parameters P1, P2, and P3 to the 19 constant P values from Table A.8.

   d. Forwards this information to the IUT using Input Type 19.

2. The IUT should perform the following for i=1 through 19:

NOTE -- the processing for each clock cycle T*i* is displayed.

   a. At clock cycle T1:

1) Calculate the input block $I1_i$ by exclusive-ORing $P1_i$ with IV1.

2) Process $I1_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

1) Calculate the input block $I2_i$ by exclusive-ORing $P2_i$ with IV2.

2) Process $I2_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Calculate the input block $I3_i$ by exclusive-ORing $P3_i$ with IV3.

2) Process $I3_i$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C1_i$..

At clock cycle T4:

1) Process $TEMP2_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C2_i$.

2) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C3_i$.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $P1_i$, $P2_i$, $P3_i$, and the resulting $C1_i$, $C2_i$, and $C3_i$, to the TMOVS as specified in Output Type 3.

c. If the IUT supports the decryption process, retain C1, C2, and C3 for use with the Substitution Table Known Answer Test for the Decryption Process for the TCBC-I Mode (Section 5.3.2.5).

d. Assign a new value to $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ by setting them equal to the next key supplied by the TMOVS.

e. Assign a new value to $P1_{i+1}$, $P2_{i+1}$, and $P3_{i+1}$ by setting them equal to the corresponding P supplied by the TMOVS.

NOTE -- The above processing continues until all 19 KEY-P values are processed. The output from the IUT should consist of 19 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.8.

5.3.1.6 Monte Carlo Test for the Encryption Process - TCBC-I Mode

Table 30 The Monte Carlo Test for the Encryption Process - TCBC-I Mode

TMOVS:	Initialize	$KEY1_0$, $KEY2_0$, $KEY3_0$, IV1, IV2, IV3, $P1_0$,$P2_0$, $P3_0$
	Send	$KEY1_0$, $KEY2_0$, $KEY3_0$, IV1, IV2, IV3, $P1_0$,$P2_0$, $P3_0$
IUT:	FOR i = 0 TO 399	
	{	
	If (i==0)	
	FOR k = 1 TO 3	
	CVk_0 = IVk	
	Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $CV1_0$, $CV2_0$, $CV3_0$, $P1_0$, $P2_0$, $P3_0$	
	FOR j = 0 TO 9,999	
	{	
	Perform Triple DES:	T1: $I1_j = P1_j \oplus CV1_j$ $I1_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$ T2: $I2_j = P2_j \oplus CV2_j$ $I2_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$ T3: $I3_j = P3_j \oplus CV3_j$ $I3_i$ is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$ $TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $C1_j$
		T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

```
        TEMP2_2 is encrypted by DEA_3 using KEY3_i, resulting in C2_j
T5:     TEMP3_2 is encrypted by DEA_3 using KEY3_i, resulting in C3_j
IF (j==0)
        FOR k = 1 TO 3
                Pk_{j+1} = CVk_0
ELSE
        FOR k = 1 TO 3
                Pk_{j+1} = Ck_{j-1}
FOR k = 1 TO 3
        CVk_{j+1} = Ck_j
}
Record C1_j,C2_j,C3_j
Send i, KEY1_i, KEY2_i, KEY3_i, CV1_0, CV2_0, CV3_0, P1_0, P2_0, P3_0, C1_i, C2_i,
C3_j
KEY1_{i+1} = KEY1_i ⊕ C1_j
IF (KEY1_i and KEY2_i are independent and KEY3_i = KEY1_i) or (KEY1_i,
KEY2_i, and KEY3_i are independent)
    KEY2_{i+1}= KEY2_i ⊕ C2_{j-1}
ELSE
    KEY2_{i+1}= KEY2_i ⊕ C1_j
IF (KEY1_i= KEY2_i= KEY3_i) or (KEY1_i and KEY2_i are independent and
KEY3_i = KEY1_i)
    KEY3_{i+1}= KEY3_i ⊕ C1_j
ELSE
    KEY3_{i+1}= KEY3_i ⊕ C3_{j-2}
FOR k = 1 TO 3
```

{

$Pk_0 = Ck_{j-1}$

$CVk_0 = Ck_j$

}

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 30, the Monte Carlo Test for the TCBC-I Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vectors IV1, IV2, and IV3, and the plaintext variables P1, P2, and P3. The P, IV, and KEY parameters consist of 64 bits each. IV2 is assigned the value of $IV1+R_1 \bmod 2^{64}$ where R_1 = 5555555555555555. IV3 is assigned the value of $IV1+R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA.

   b. Forwards this information to the IUT using Input Type 22.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), set the chaining value $CV1_0$ equal to the IV1, $CV2_0$ equal to the IV2, and $CV3_0$ equal to the IV3.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $CV1_0$, $CV2_0$, $CV3_0$, and $P1_0$, $P2_0$, $P3_0$.

   c. Perform the following for j = 0 through 9999:

NOTE -- the processing for each clock cycle T*i* is displayed.

1) At clock cycle T1:

   a) Calculate the input block $I1_j$ by exclusive-ORing $P1_j$ with $CV1_j$.

   b) Process $I1_j$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

   At clock cycle T2:

   a) Calculate the input block $I2_j$ by exclusive-ORing $P2_j$ with $CV2_j$.

b) Process $I2_j$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

a) Calculate the input block $I3_j$ by exclusive-ORing $P3_j$ with $CV3_j$.

b) Process $I3_j$ through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

d) Process $TEMP1_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C1_j$.

At clock cycle T4:

a) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

b) Process $TEMP2_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C2_j$.

At clock cycle T5:

a) Process $TEMP3_2$ through the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in the ciphertext value $C3_j$.

2) Prepare for loop j+1 by doing the following:

a) If the inner loop being processed is the first loop, i.e., j=0, assign $P1_{j+1}$, $P2_{j+1}$, and $P3_{j+1}$, with the current value of $CV1_0$, $CV2_0$, and $CV3_0$, respectively. Otherwise, assign $P1_{j+1}$ with $C1_{j-1}$, $P2_{j+1}$ with $C2_{j-1}$, and $P3_{j+1}$ with $C3_{j-1}$.

b) Assign $CV1_{j+1}$, $CV2_{j+1}$, $CV3_{j+1}$, with the current value of $C1_j$, $C2_j$, $C3_j$, respectively.

d. Record the $C1_j$, $C2_j$, $C3_j$.

e. Forward all recorded information from this loop, as specified in Output Type 4, to the TMOVS.

f. Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop. Note j = 9999.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the $C1_j$.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $C2_{j-1}$. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $C1_j$.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $C3_{j-2}$. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $C1_j$.

g. Assign new values to $CV1_0$, $CV2_0$, and $CV3_0$, in preparation for the next outer loop. $CV1_0$, $CV2_0$, and $CV3_0$ should be assigned the value of the current $C1_j$, $C2_j$, and $C3_j$.

NOTE -- the new CV should be denoted as CV_0 because this value is used for the first pass through the inner loop when j=0.

h. Assign a new value to $P1_0$, $P2_0$, and $P3_0$ in preparation for the next output loop. $P1_0$ should be assigned the value of the $C1_{j-1}$. $P2_0$ should be assigned the value of the $C2_{j-1}$, and $P3_0$ should be assigned the value of the $C3_{j-1}$.

NOTE -- the new P variables, P1, P2, and P3 should be denoted as $P1_0$, $P2_0$, and $P3_0$, respectively, to be used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 4.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.3.2 Decryption Process

The process of validating an IUT for the TCBC-I mode which implements the decryption process involves the successful completion of the following six tests:

1. The Variable Ciphertext Known Answer Test - TCBC-I mode

2. The Initial Permutation Known Answer Test - TCBC-I mode

3. The Variable Key Known Answer Test for the Decryption Process - TCBC-I mode

4. The Permutation Operation Known Answer Test for the Decryption Process - TCBC-I mode

5. The Substitution Table Known Answer Test for the Decryption Process - TCBC-I mode

6. The Monte Carlo Test for the Decryption Process - TCBC-I mode

An explanation of the tests follows.

5.3.2.1 The Variable Ciphertext Known Answer Test - TCBC-I Mode

Table 31 The Variable Ciphertext Known Answer Test - TCBC-I Mode

TMOVS:	Initialize	KEY1 =KEY2 = KEY3 = 0101010101010101 (odd parity set)
		IV1=0000000000000000
		IV2 = 5555555555555555 (based on specifications in ANSI X9.52 – 1998)
		IV3 = AAAAAAAAAAAAAAAA (based on specifications in ANSI X9.52 – 1998)
	Send	KEY (representing KEY1, KEY2, and KEY3), IV1,IV2,IV3
	If encryption is not supported by the IUT:	
	Initialize	Ck_i (where k=1..3 and i=1..64) = Ck values in Table A.5
	Send	$C1_1, C1_2,...,C1_{64}$, $C2_1, C2_2,...,C2_{64}$ $C3_1, C3_2,...,C3_{64}$
IUT:	If encryption is supported:	
	Initialize	$C1_1$, $C2_1$, $C3_1$ = corresponding values from output of Variable Plaintext Known Answer Test.
	Otherwise, use the corresponding values received from the TMOVS.	
	FOR i = 1 to 64	
	{	
	Process Triple DES:	T1: $I1_i = C1_i$ $I1_i$ is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in $TEMP1_1$ T2: $I2_i = C2_i$ $I2_i$ is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in $TEMP2_1$ $TEMP1_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP1_2$ T3: $I3_i = C3_i$ $I3_i$ is read into TDEA and is decrypted by DEA_3 using KEY3,

resulting in $TEMP3_1$

$TEMP2_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP2_2$

$TEMP1_2$ is decrypted by DEA_1 using KEY1, resulting in $O1_i$

$P1_i = O1_i \oplus IV1$

T4: $TEMP3_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP3_2$

$TEMP2_2$ is decrypted by DEA_1 using KEY1, resulting in $O2_i$

$P2_i = O2_i \oplus IV2$

T5: $TEMP3_2$ is decrypted by DEA_1 using KEY1, resulting in $O3_i$

$P3_i = O3_i \oplus IV3$

Send i, KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, $P1_i$, $P2_i$, $P3_i$

If encryption is supported:

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} from output of Variable Plaintext Known Answer Test

}

TMOVS: Compare results from each loop with known answers. Should be the set of basis vectors.

Table 31 illustrates the Variable Ciphertext Known Answer Test for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

c. If the IUT does not support encryption, the 64 constant ciphertext values C1, C2, and C3 are initialized with the corresponding 64 constant C1, C2, and C3 values from Table A.5.

d. If encryption is supported by the IUT, the KEYs and the IVs are forwarded to the IUT, as specified in Input Type 14. If encryption is not supported by the IUT, the KEYs, the IVs, and the 64 $C1_i$, $C2_i$, and $C3_i$ values are forwarded to the IUT using Input Type 15.

2. The IUT should:

a. If encryption is supported, initialize the C values $C1_1$, $C2_1$, and $C3_1$, with the corresponding $C1_1$, $C2_1$, and $C3_1$ values retained from the Variable Plaintext Known Answer Test for the TCBC-I Mode (Section 5.3.1.1). Otherwise, use the first values received from the TMOVS.

b. Perform the following for i = 1 through 64:

NOTE -- the processing for each clock cycle T*i* is displayed.

At clock cycle T1:

1) Set the input block $I1_i$ equal to the value of $C1_i$.

2) Process $I1_i$ through the DEA stage DEA_3, in the decrypt state using KEY3, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

1) Set the input block $I2_i$ equal to the value of $C2_i$.

2) Process $I2_i$ through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Set the input block $I3_i$ equal to the value of $C3_i$.

2) Process $I3_i$ through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the DEA stage DEA_1 in the decrypt state using KEY1, resulting in the output block $O1_i$.

5) Calculate the plaintext $P1_i$ by exclusive-ORing $O1_i$ with IV1.

At clock cycle T4:

1) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using KEY1, resulting in the output block $O2_i$.

2) Calculate the plaintext $P2_i$ by exclusive-ORing $O2_i$ with IV2.

3) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using KEY1, resulting in the output block $O3_i$.

2) Calculate the plaintext $P3_i$ by exclusive-ORing $O3_i$ with IV3.

c. Forward the current values of the loop number i, KEY (which represents KEY1, KEY2, and KEY3), IV1, IV2, IV3, $C1_i$, $C2_i$, and $C3_i$, and the resulting $P1_1$, $P2_1$ and $P3_1$ as specified in Output Type 3.

d. Retain $P1_i$, $P2_i$, and $P3_i$, for use with the Initial Permutation Known Answer Test for the TCBC-I Mode (Section 5.3.2.2).

e. If encryption is supported, set $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding output from the Variable Plaintext Known Answer Test for the TCBC-I mode. If encryption is not supported, assign new values to $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$, by setting them equal to the corresponding C values supplied by the TMOVS.

NOTE -- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The values of the P1, P2, and P3 variables should be the set of basis vectors.

5.3.2.2 The Initial Permutation Known Answer - TCBC-I Mode

Table 32 The Initial Permutation Known Answer Test - TCBC-I Mode

TMOVS:	Initialize	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) IV1=0000000000000000 IV2 = 5555555555555555 IV3 = AAAAAAAAAAAAAAAA

		Ck_i (where k=1..3 and i=1..64) = 64 corresponding Pk_i values from Variable Ciphertext Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $C1_1,...,C1_{64}$, $C2_1,...,C2_{64}$, $C3_1,...,C3_{64}$
IUT:	FOR i = 1 to 64	
	{	

Perform Triple DES:

T1:	$I1_i = C1_i$
	$I1_i$ is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in $TEMP1_1$
T2:	$I2_i = C2_i$
	$I2_i$ is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in $TEMP2_1$
	$TEMP1_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP1_2$
T3:	$I3_i = C3_i$
	$I3_i$ is read into TDEA and is decrypted by DEA_3 using KEY3, resulting in $TEMP3_1$
	$TEMP2_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP2_2$
	$TEMP1_2$ is decrypted by DEA_1 using KEY1, resulting in $O1_i$
	$P1_i = O1_i \oplus IV1$
T4:	$TEMP3_1$ is encrypted by DEA_2 using KEY2, resulting in $TEMP3_2$
	$TEMP2_2$ is decrypted by DEA_1 using KEY1, resulting in $O2_i$
	$P2_i = O2_i \oplus IV2$
T5:	$TEMP3_2$ is decrypted by DEA_1 using KEY1, resulting in $O3_i$
	$P3_i = O3_i \oplus IV3$

Send i, KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, $P1_i$,$P2_i$,$P3_i$

Ck_{i+1} (where k=1..3) = corresponding Pk_{i+1} from TMOVS

}

TMOVS: Compare C1, C2, and C3 results from each loop with known answers.

See Table A.7.

Table 32 illustrates the Initial Permutation Known Answer Test for the TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. Initializes the 64-bit ciphertext values $C1_i$, $C2_i$, and $C3_i$ to the corresponding plaintext values $P1_i$, $P2_i$, and $P3_i$, respectively, obtained from the Variable Ciphertext Known Answer Test.

   d. Forwards this information to the IUT using Input Type 15.

2. The IUT should perform the following for i=1 through 64:

NOTE -- the processing for each clock cycle T*i* is displayed.

   a. At clock cycle T1:

      1) Set the input block $I1_i$ equal to the value of $C1_i$.

      2) Process $I1_i$ through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value $TEMP1_1$.

      At clock cycle T2:

      1) Set the input block $I2_i$ equal to the value of $C2_i$.

2) Process $I2_i$ through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value $TEMP2_1$.

3) Process $TEMP1_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

1) Set the input block $I3_i$ equal to the value of $C3_i$.

2) Process $I3_i$ through the DEA stage DEA_3 in the decrypt state using KEY3, resulting in intermediate value $TEMP3_1$.

3) Process $TEMP2_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

4) Process $TEMP1_2$ through the DEA stage DEA_1, in the decrypt state using KEY1, resulting in the output block $O1_i$.

5) Calculate the plaintext $P1_i$ by exclusive-ORing $O1_i$ with IV1.

At clock cycle T4:

1) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using KEY1, resulting in the output block $O2_i$.

2) Calculate the plaintext $P2_i$ by exclusive-ORing $O2_i$ with IV2.

3) Process $TEMP3_1$ through the second DEA stage, denoted DEA_2, in the encrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

1) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using KEY1, resulting in the output block $O3_i$.

2) Calculate the plaintext $P3_i$ by exclusive-ORing $O3_i$ with IV3.

b. Forward the current values of the loop number i, KEY (which represents KEY1, KEY2, and KEY3), IV1, IV2, IV3, $C1_i$, $C2_i$, and $C3_i$, and the resulting $P1_i$, $P2_i$ and $P3_i$ as specified in Output Type 3.

c. Set $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding output supplied by the TMOVS.

NOTE -- The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received C1, C2, and C3 results to the known values. See Table A.7.

5.3.2.3 The Variable Key Known Answer Test for the Decryption Process - TCBC-I Mode

Table 33 The Variable Key Known Answer Test for the Decryption Process - TCBC-I Mode

TMOVS: Initialize $KEY1_1$= $KEY2_1$ = $KEY3_1$ = 8001010101010101 (odd parity set)

IV1=0000000000000000

IV2 = 5555555555555555

IV3 = AAAAAAAAAAAAAAAA

If encryption supported by IUT:

Send KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV1, IV2, IV3

If encryption is not supported by the IUT:

Initialize Ck_i (where k=1..3 and i=1..56) = Ck values in Table A.11

Send KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV1,IV2,IV3, $C1_1$,...,$C1_{56}$, $C2_1$,...,$C2_{56}$, $C3_1$,...,$C3_{56}$

IUT: If encryption is supported by the IUT:

Initialize Ck_1 (where k=1..3) = corresponding values from output of Variable Key Known Answer Test for the Encryption Process.

Otherwise, use the corresponding value received from the TMOVS.

FOR i = 1 to 64

{

If (i mod 8 ≠ 0) {process every bit except parity bits}

{

Perform Triple DES:

T1: $I1_i = C1_i$

$I1_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP1_1$

T2: $I2_i = C2_i$

$I2_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP2_1$

$TEMP1_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$

T3: $I3_i = C3_i$

$I3_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP3_1$

$TEMP2_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$

$TEMP1_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O1_i$

$P1_i = O1_i \oplus IV1$

T4: $TEMP3_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O2_i$

$P2_i= O2_i \oplus IV2$

T5: $TEMP3_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O3_i$

$P3_i= O3_i \oplus IV3$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, $P1_i$, $P2_i$, $P3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in the i+1th position. NOTE -- odd parity is set.

If encryption is supported:

Assign $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ to corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ values from Variable Key Known Answer Test for the Encryption Process

else

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} value from TMOVS

}

}

TMOVS: Compare results from the 56 decryptions with known answers. Should be P1 = P2 =

P3 = 0 for all 56 rounds.

Table 33 illustrates the Variable Key Known Answer Test for the Decryption Process - TCBC-I mode of operation.

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

      NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. If the IUT does not support encryption, $C1_i$, $C2_i$, and $C3_i$ values are initialized with the constant $C1_i$, $C2_i$, and $C3_i$ values from Table A.11 where i=1..56.

   d. If encryption is not supported by the IUT, KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, and IV3, and the 56 C1, C2, and C3 values are forwarded to the IUT, as specified in Input Type 15. Otherwise, the KEY (representing KEY1, KEY2, and KEY3), IV1, IV2, and IV3 are forwarded to the IUT, as specified in Input Type 14.

2. The IUT should:

   a. If encryption is supported, initialize the $C1_1$, $C2_1$, and $C3_1$ values with the first corresponding C1, C2, and C3 values retained from the Variable KEY Known Answer Test for the Encryption Process for the TCBC-I Mode (Section 5.3.1.3). Otherwise, use the first values received from the TMOVS.

   b. Perform the following for i = 1 to 56:

      NOTE -- 56 is the number of significant bits in a TDES key.

NOTE -- the processing for each clock cycle T*i* is displayed.

      1) At clock cycle T1:

         a) Set the input block $I1_i$ equal to the value of $C1_i$.

b) Process $I1_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

a) Set the input block $I2_i$ equal to the value of $C2_i$.

b) Process $I2_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP2_1$.

c) Process $TEMP1_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

a) Set the input block $I3_i$ equal to the value of $C3_i$.

b) Process $I3_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP3_1$.

c) Process $TEMP2_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

d) Process $TEMP1_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O1_i$.

e) Calculate the plaintext $P1_i$ by exclusive-ORing $O1_i$ with IV1.

At clock cycle T4:

a) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O2_i$.

b) Calculate the plaintext $P2_i$ by exclusive-ORing $O2_i$ with IV2.

c) Process $TEMP3_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

a) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O3_i$.

b) Calculate the plaintext $P3_i$ by exclusive-ORing $O3_i$ with IV3.

2) Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, and the resulting $P1_i$, $P2_i$, and $P3_i$ to the TMOVS as specified in Output Type 3.

3) Set $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1,}$, equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits may contain "1" or "0" to make odd parity.

NOTE --$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$.

4) If encryption is supported, set the C values $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$, equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ values retained from the Variable KEY Known Answer Test for the Encryption Process for TCBC-I mode. If encryption is not supported by the IUT, set $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$, values supplied by the TMOVS.

NOTE -- The output from the IUT for this test should consist of 56 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The P1, P2, and P3 results should be all zeros.

5.3.2.4 Permutation Operation Known Answer Test for the Decryption Process - TCBC-I Mode

Table 34 The Permutation Operation Known Answer Test for the Decryption Process - TCBC-I Mode

TMOVS:	Initialize	$KEY1_i$ =$KEY2_i$ = $KEY3_i$ (where i=1..32) = 32 KEY values from Table A.12
		IV1 = 0000000000000000
		IV2 = 5555555555555555
		IV3 = AAAAAAAAAAAAAAAA
	If encryption is supported by the IUT:	
	Send	IV1, IV2, and IV3
		KEY_1, KEY_2,..., KEY_{32} (where KEY represents the values of KEY1, KEY2, and KEY3)
	If encryption is not supported by the IUT:	
	Initialize	Ck_i (where k=1..3 and i=1..32) = corresponding Ck_i values from Table A.12
	Send	IV1, IV2, and IV3,
		KEY_1, KEY_2,..., KEY_{32} (where KEY represents the values of KEY1, KEY2, and KEY3)
		$C1_1$,...,$C1_{32}$, $C2_1$,...,$C2_{32}$, $C3_1$,...,$C3_{32}$
IUT:	If encryption is supported by the IUT:	
	Initialize	$C1_1$, $C2_1$, $C3_1$ values with corresponding values retained from Permutation Operation Known Answer Test for Encryption Process.
	Otherwise, use the first values received from the TMOVS.	
	FOR i = 1 to 32	
	{	
	Perform	T1: $I1_i = C1_i$

Triple DES:

$I1_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP1_1$

T2: $I2_i = C2_i$

$I2_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP2_1$

$TEMP1_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$

T3: $I3_i = C3_i$

$I3_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP3_1$

$TEMP2_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$

$TEMP1_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O1_i$

$P1_i = O1_i \oplus IV1$

T4: $TEMP3_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O2_i$

$P2_i = O2_i \oplus IV2$

T5: $TEMP3_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O3_i$

$P3_i = O3_i \oplus IV3$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, $P1_i$, $P2_i$, $P3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} supplied from TMOVS

If encryption is supported:

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} from Permutation Operation Known Answer Test for the Encryption Process

else

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} from TMOVS

}

TMOVS: Compare results with known answers. Results should be P1=P2=P3=0.

Table 34 illustrates the Permutation Operation Known Answer Test for the TCBC-I Decryption Process.

1. The TMOVS:

   a. If the IUT supports encryption, the $KEY1_i$, $KEY2_i$, and $KEY3_i$ variables are initialized with the 32 constant KEY_i values from Table A.12. If the IUT does not support encryption, the KEY variables, $KEY1_i$, $KEY2_i$, and $KEY3_i$, and the C variables, C1, C2, and C3 are initialized with the 32 constant KEY and C1, C2, and C3 values from Table A.12.

      NOTE -- KEY1=KEY2=KEY3.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. If encryption is supported by the IUT, the 32 KEY values for KEY1, KEY2, and KEY3, and the IV1, IV2, and IV3 values are forwarded to the IUT using Input Type 16. If encryption is not supported by the IUT, the 32 KEY, C1, C2, and C3 groups and the IV1, IV2, and IV3 values are forwarded to the IUT using Input Type 17.

2. The IUT should:

   a. If encryption is supported by the IUT, initialize the $C1_1$, $C2_1$, and $C3_1$ values with the first $C1_1$, $C2_1$, and $C3_1$ values retained from the Permutation Operation Known Answer Test for the Encryption Process for the TCBC-I Mode (Section 5.3.1.4). Otherwise, use the first values received from the TMOVS.

   b. Perform the following for i = 1 to 32:

NOTE -- the processing for each clock cycle T*i* is displayed.

      1) At clock cycle T1:

         a) Set the input block $I1_i$ equal to the value of $C1_i$.

         b) Process $I1_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

a) Set the input block $I2_i$ equal to the value of $C2_i$.

b) Process $I2_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP2_1$.

c) Process $TEMP1_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

a) Set the input block $I3_i$ equal to the value of $C3_i$.

b) Process $I3_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP3_1$.

c) Process $TEMP2_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

d) Process $TEMP1_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O1_i$.

e) Calculate the plaintext $P1_i$ by exclusive-ORing $O1_i$ with IV1.

At clock cycle T4:

a) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O2_i$.

b) Calculate the plaintext $P2_i$ by exclusive-ORing $O2_i$ with IV2.

c) Process $TEMP3_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

a) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O3_i$.

b) Calculate the plaintext $P3_i$ by exclusive-ORing $O3_i$ with IV3.

2) Forward the current values of the loop number i, KEY_I (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, and the resulting $P1_i$, $P2_i$, and $P3_i$ to the TMOVS as specified in Output Type 3.

3) Assign new values to $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1,}$ by setting them equal to the corresponding KEY values supplied by the TMOVS.

NOTE -- KEY1=KEY2=KEY3.

4) If encryption is supported, set the C values $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ values retained from the Permutation Operation Known Answer Test for the Encryption Process for TCBC-I mode. If encryption is not supported by the IUT, set $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$, values supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The resulting P1, P2, and P3 results should be all zeros.

5.3.2.5 Substitution Table Known Answer Test for the Decryption Process - TCBC-I Mode

Table 35 The Substitution Table Known Answer Test for the Decryption Process - TCBC-I Mode

TMOVS:	Initialize	$KEY1_i$ =$KEY2_i$ = $KEY3_i$ (where i = 1..19) = 19 KEY values from Table A.8
		IV1=0000000000000000
		IV2 = 5555555555555555
		IV3 = AAAAAAAAAAAAAAAA
	If encryption is supported by the IUT:	
	Send	IV1, IV2, and IV3,
		KEY_1, KEY_2,..., KEY_{19} (where KEY represents the values of KEY1, KEY2, and KEY3)
	If encryption is not supported by the IUT:	
	Initialize	Ck_i (where k=1..3 and where i=1..19) = corresponding Ck_i values from Table A.8
	Send	IV1, IV2, and IV3,
		KEY_1, KEY_2,..., KEY_{19} (where KEY represents the values of KEY1, KEY2, and KEY3)
		$C1_1$,...,$C1_{19}$, $C2_1$,...,$C2_{19}$, $C3_1$,...,$C3_{19}$
IUT:	If encryption is supported by the IUT:	
	Initialize	$C1_1$, $C2_1$, $C3_1$ values retained from Substitution Table Known Answer Test for Encryption Process.
	Otherwise, use the first values received from the TMOVS.	
	FOR i = 1 to 19	
	{	
Perform Triple DES:	T1:	$I1_i$ = $C1_i$
		$I1_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP1_1$

T2: $I2_i = C2_i$

$I2_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP2_1$

$TEMP1_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$

T3: $I3_i = C3_i$

$I3_i$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP3_1$

$TEMP2_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$

$TEMP1_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O1_i$

$P1_i = O1_i \oplus IV1$

T4: $TEMP3_1$ is encrypted by DEA2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O2_i$

$P2_i = O2_i \oplus IV2$

T5: $TEMP3_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O3_i$

$P3_i = O3_i \oplus IV3$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, $P1_i$,$P2_i$,$P3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} supplied from TMOVS

If encryption is supported:

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} from output of Substitution Table Known Answer Test for the Encryption Process

else

Ck_{i+1} (where k=1..3) = corresponding Ck_{i+1} values from TMOVS

}

TMOVS:	Compare results with known answers.

Table 35 illustrates the Substitution Table Known Answer Test for the TCBC-I Decryption Process.

1. The TMOVS:

   a. If the IUT supports encryption, the $KEY1_i$, $KEY2_i$, and $KEY3_i$ variables are initialized with the 19 constant KEY values from Table A.8. If the IUT does not support encryption, the KEY variables, KEY1, KEY2, and KEY3, and the C variables, C1, C2, and C3 are initialized with the 19 constant KEY and C1, C2, and C3 values from Table A.8.

      NOTE -- KEY1=KEY2=KEY3.

   b. Initializes the 64-bit IV parameters, IV1, IV2, and IV3. IV1 is initialized to the constant hexadecimal value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555, i.e., $IV2_{hex}$ = 55 55 55 55 55 55 55 55. IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA, i.e., $IV3_{hex}$ = AA AA AA AA AA AA AA AA.

   c. If encryption is supported by the IUT, the 19 KEY values for KEY1, KEY2, and KEY3, and the IV1, IV2, and IV3 values are forwarded to the IUT using Input Type 16. If encryption is not supported by the IUT, the 19 KEY, C1, C2, and C3 groups and the IV1, IV2, and IV3 values are forwarded to the IUT using Input Type 17.

2. The IUT should:

   a. If encryption is supported by the IUT, initialize the C1, C2, and C3 values with the first C1, C2, and C3 values retained from the Substitution Table Known Answer Test for the Encryption Process for the TCBC-I Mode (Section 5.3.1.5). Otherwise, use the first value received from the TMOVS.

   b. Perform the following for i = 1 to 19:

NOTE -- the processing for each clock cycle T*i* is displayed.

      1) At clock cycle T1:

         a) Set $I1_i$ equal to the value of $C1_i$.

         b) Process $I1_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP1_1$.

         At clock cycle T2:

         a) Set $I2_i$ equal to the value of $C2_i$.

b) Process $I2_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP2_1$.

c) Process $TEMP1_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

a) Set $I3_i$ equal to the value of $C3_i$.

b) Process $I3_i$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP3_1$.

c) Process $TEMP2_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

d) Process $TEMP1_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O1_i$.

e) Calculate the plaintext $P1_i$ by exclusive-ORing $O1_i$ with IV1.

At clock cycle T4:

a) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O2_i$.

b) Calculate the plaintext $P2_i$ by exclusive-ORing $O2_i$ with IV2.

c) Process $TEMP3_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

a) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O3_i$.

b) Calculate the plaintext $P3_i$ by exclusive-ORing $O3_i$ with IV3.

2) Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, $C1_i$, $C2_i$, $C3_i$, and the resulting $P1_i$, $P2_i$, and $P3_i$ to the TMOVS as specified in Output Type 3.

3) Assign new values to $KEY1_{i+1,}$, $KEY2_{i+1,}$, and $KEY3_{i+1}$, by setting them equal to the corresponding KEY_{i+1} values supplied by the TMOVS.

NOTE -- $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$.

4) If encryption is supported, set the C values $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ values retained from the Substitution Table Known Answer Test for the Encryption Process for

TCBC-I mode. If encryption is not supported by the IUT, set $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$ equal to the corresponding $C1_{i+1}$, $C2_{i+1}$, and $C3_{i+1}$, values supplied by the TMOVS.

NOTE -- The above processing should continue until all 19 KEY, C1, C2, and C3 groups, as specified in Input Type 17, or all 19 KEY values, as specified in Input Type 16, are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.3.2.6 Monte Carlo Test for the Decryption Process - TCBC-I Mode

Table 36 The Monte Carlo Test for the Decryption Process - TCBC-I Mode

TMOVS:	Initialize		$KEY1_0$, $KEY2_0$, $KEY3_0$, IV1, IV2, IV3, $C1_0$,$C2_0$, $C3_0$
	Send		$KEY1_0$, $KEY2_0$, $KEY3_0$, IV1, IV2, IV3, $C1_0$,$C2_0$, $C3_0$
IUT:	FOR i = 0 TO 399		
	{		
	If (i==0)		
	FOR k=1 to 3		
	CVk_0 = IVk		
	Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $CV1_0$, $CV2_0$, $CV3_0$, $C1_0$, $C2_0$, $C3_0$		
	FOR j = 0 TO 9,999		
	{		
	Perform Triple DES:	T1:	$I1_j = C1_j$
			$I1_j$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP1_1$
		T2:	$I2_j = C2_j$
			$I2_j$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP2_1$
			$TEMP1_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$
		T3:	$I3_j = C3_j$
			$I3_j$ is read into TDEA and is decrypted by DEA_3 using $KEY3_i$, resulting in $TEMP3_1$
			$TEMP2_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$
			$TEMP1_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O1_j$

$P1_j = O1_j \oplus CV1$

T4: $TEMP3_1$ is encrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O2_j$

$P2_j = O2_j \oplus CV2$

T5: $TEMP3_2$ is decrypted by DEA_1 using $KEY1_i$, resulting in $O3_j$

$P3_j = O3_j \oplus CV3$

FOR k= 1 to 3

{

$CVk_{j+1} = Ck_j$

$Ck_{j+1} = Pk_j$

}

}

Record $P1_j$, $P2_j$, $P3_j$

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $CV1_0$, $CV2_0$, $CV3_0$, $C1_0$, $C2_0$, $C3_0$, $P1_i$, $P2_i$, $P3_j$

$KEY1_{i+1} = KEY1_i \oplus P1_j$

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent)

$KEY2_{i+1} = KEY2_i \oplus P2_{j-1}$

ELSE

$KEY2_{i+1} = KEY2_i \oplus P1_j$

IF ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$)

$KEY3_{i+1} = KEY3_i \oplus P1_j$

ELSE

$KEY3_{i+1} = KEY3_i \oplus P3_{j-2}$

FOR k= 1 to 3

{

$CVk_0 = Ck_j$

$Ck_0 = Pk_j$

}

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 36, the Monte Carlo Test for the TCBC-I Decryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vectors IV1, IV2, and IV3, and the ciphertext variables C1, C2, and C3. The C variables, the IV variables , and the KEY variables consist of 64 bits each. IV2 is assigned the value of $IV1+R_1 \bmod 2^{64}$ where R_1 = 5555555555555555. IV3 is assigned the value of $IV1+R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA.

   b. Forwards this information to the IUT using Input Type 22.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), set the chaining value $CV1_0$ equal to the IV1, $CV2_0$ equal to the IV2, and $CV3_0$ equal to the IV3.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $CV1_0$, $CV2_0$, $CV3_0$, and $C1_0$, $C2_0$, $C3_0$.

   c. Perform the following for j = 0 through 9999:

NOTE -- the processing for each clock cycle T*i* is displayed.

      1) At clock cycle T1:

         a) Set the input block $I1_j$ equal to the value of $C1_j$.

b) Process $I1_j$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP1_1$.

At clock cycle T2:

a) Set the input block $I2_j$ equal to the value of $C2_j$.

b) Process $I2_j$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP2_1$.

c) Process $TEMP1_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At clock cycle T3:

a) Set the input block $I3_j$ equal to the value of $C3_j$.

b) Process $I3_j$ through the DEA stage DEA_3 in the decrypt state using $KEY3_i$, resulting in intermediate value $TEMP3_1$.

c) Process $TEMP2_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

d) Process $TEMP1_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O1_j$.

e) Calculate the plaintext $P1_j$ by exclusive-ORing $O1_j$ with IV1.

At clock cycle T4:

a) Process $TEMP2_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O2_j$.

b) Calculate the plaintext $P2_j$ by exclusive-ORing $O2_j$ with IV2.

c) Process $TEMP3_1$ through the DEA stage DEA_2 in the encrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At clock cycle T5:

a) Process $TEMP3_2$ through the DEA stage DEA_1 in the decrypt state using $KEY1_i$, resulting in the output block $O3_j$.

b) Calculate the plaintext $P3_j$ by exclusive-ORing $O3_j$ with IV3.

2) Prepare for loop j+1 by doing the following:

a) Assign $CV1_{j+1}$, $CV2_{j+1}$, $CV3_{j+1}$ the current value of $C1_j$, $C2_j$, $C3_j$, respectively.

b) Assign $C1_{j+1}$, $C2_{j+1}$, $C3_{j+1}$ the current value of $P1_j$, $P2_j$, $P3_j$, respectively.

d. Record $P1_j$, $P2_j$, $P3_j$.

e. Forward all recorded information for this loop, as specified in Output Type 4, to the TMOVS.

f. Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop. Note j = 9999.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the $P1_j$.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $P2_{j-1}$. If $KEY1_i=KEY2_i=KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $P1_j$.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $P3_{j-2}$. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i=KEY2_i=KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $P1_j$.

g. Assign new values to $CV1_0$, $CV2_0$, and $CV3_0$, in preparation for the next outer loop. $CV1_0$, $CV2_0$, and $CV3_0$ should be assigned the value of the current $C1_j$, $C2_j$, and $C3_j$, respectively.

NOTE -- the new CV should be denoted as CV_0 because this value is used for the first pass through the inner loop when j=0.

h. Assign a new value to $C1_0$, $C2_0$, and $C3_0$ in preparation for the next output loop. $C1_0$ should be assigned the value of the $P1_j$. Likewise, $C2_0$ should be assigned the value of the $P2_j$, and $C3_0$ should be assigned the value of the $P3_j$.

NOTE -- the new C variables, C1, C2, and C3 should be denoted as $C1_0$, $C2_0$, and $C3_0$, respectively, to be used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 4.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

5.4 The Cipher Feedback (TCFB) Mode

The IUTs in the TDES Cipher Block Feedback (TCFB) mode of operation are validated by successfully completing (1) a set of Known Answer tests applicable to both IUTs supporting encryption and/or decryption and (2) a Monte Carlo test for each cryptographic process supported by the IUT.

The process of validating an IUT which supports the K-bit TCFB mode in either the encryption and/or decryption process involves the successful completion of the following six tests:

1. The Variable Text Known Answer Test - K-bit TCFB mode

2. The Inverse Permutation Known Answer Test - K-bit TCFB mode

3. The Variable Key Known Answer Test - K-bit TCFB mode

4. The Permutation Operation Known Answer Test - K-bit TCFB mode

5. The Substitution Table Known Answer Test - K-bit TCFB mode

6. The Monte Carlo Test for the Encryption Process - K-bit TCFB mode (if encryption is supported)

   OR

   The Monte Carlo Test for the Decryption Process - K-bit TCFB mode (if decryption is supported)

NOTE -- For IUTs, K can range from 1 to 64 bits.

An explanation of the tests follows.

5.4.1 The Known Answer Tests - TCFB Mode

The K-bit TCFB mode has one set of Known Answer tests which is used regardless of supported process, i.e., the same set of Known Answer tests is for IUTs supporting the encryption and/or decryption processes.

Throughout this section, TEXT and RESULT will refer to different variables depending on whether the encryption or decryption process is being tested. If the IUT performs TCFB encryption, TEXT refers to plaintext, and RESULT refers to ciphertext. If the IUT performs TCFB decryption, TEXT refers to ciphertext, and RESULT refers to plaintext.

The notation $LM^K(A)$ refers to the leftmost K-bits of A.

5.4.1.1 The Variable TEXT Known Answer Test - TCFB Mode

Table 37 The Variable TEXT Known Answer Test - TCFB Mode

TMOVS:	Initialize	KEYS:	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		IV_1 = 8000000000000000	
		K-bit TEXT = 0	
	Send	KEY (representing KEY1, KEY2, and KEY3), IV_1, K-bit TEXT	
IUT:	FOR i = 1 to 64		
	{		
	Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in O_i K-bit $RESULT_i = LM^K(O_i) \oplus$ K-bit TEXT	
		Send i, KEY (representing KEY1, KEY2, and KEY3), IV_i, K-bit TEXT, K-bit $RESULT_i$	
		IV_{i+1} = basis vector where single "1" bit is in position i+1	
	}		
TMOVS:	Compare RESULT from each loop with known answers.		
	Use K bits of output in Table A.1.		

As summarized in Table 37, the Variable TEXT Known Answer Test for the TCFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

b. Initializes the 64-bit initialization vector IV_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

c. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i=1 through 64:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages, resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in output block O_i.

c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i with the K-bit TEXT, i.e., $(RESULT^1{}_i, RESULT^2{}_i, ..., RESULT^K{}_i) = (O^1{}_i \oplus TEXT^1, O^2{}_i \oplus TEXT^2, ..., O^K{}_i \oplus TEXT^K)$.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV_i, K-bit TEXT, and the resulting K-bit $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Retain the K-bit RESULT values for use with the Inverse Permutation Known Answer Test for the TCFB Mode (Section 5.4.1.2).

f. Assign a new value to IV_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64.

NOTE -- This continues until every possible basis vector has been represented by the IV, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.1. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in Table A.1 are used.

5.4.1.2 The Inverse Permutation Known Answer Test - TCFB Mode

Table 38 The Inverse Permutation Known Answer Test - TCFB Mode

TMOVS:	Initialize	KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) K-bit $TEXT_i$ (where i=1..64)=64 RESULT values from the Variable TEXT Known Answer Test IV_1= 8000000000000000
	Send	KEY (representing KEY1, KEY2, and KEY3), IV_1, K-bit $TEXT_1$,...,K-bit $TEXT_{64}$
IUT:	FOR i = 1 to 64 {	
	Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in O_i K-bit $RESULT_i$= $LM^K(O_i) \oplus$ K-bit $TEXT_i$
		Send i, KEY (representing KEY1, KEY2, and KEY3), IV_i, K-bit $TEXT_i$, K-bit $RESULT_i$ IV_{i+1} = basis vector where single "1" bit is in position i+1 K-bit $TEXT_{i+1}$ = corresponding K-bit RESULT value from the TMOVS
	}	
TMOVS:	Compare RESULT from each loop with known answers. The RESULTs should be all zeros.	

As summarized in Table 38, the Inverse Permutation Known Answer Test for the TCFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

b. Initializes the 64-bit initialization vector IV_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

c. Initializes the K-bit $TEXT_i$ (where i=1..64) to the $RESULT_i$ values obtained from the Variable TEXT Known Answer Test.

d. Forwards this information to the IUT using Input Type 5.

2. The IUT should perform the following for i=1 through 64:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages, resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in output block O_i.

c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i with the K-bit $TEXT_i$, i.e., $(RESULT^1_i, RESULT^2_i, ..., RESULT^K_i) = (O^1_i \oplus TEXT^1_i, O^2_i \oplus TEXT^2_i, ..., O^K_i \oplus TEXT^K_i)$.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, K-bit $TEXT_i$, and the resulting K-bit $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Assign a new value to IV_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64.

f. Assign a new value to $TEXT_{i+1}$ by setting it equal to the corresponding output from the TMOVS.

NOTE -- This processing continues until all RESULT values from the Variable TEXT Known Answer Test have been used as input. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The RESULT values should be all zeros.

5.4.1.3 The Variable KEY Known Answer Test - TCFB Mode

Table 39 The Variable Key Known Answer Test - TCFB Mode

TMOVS:	Initialize	KEYs: $KEY1_1$ = $KEY2_1$ = $KEY3_1$ = 8001010101010101 (odd parity set) IV = 0000000000000000 K-bit TEXT = 0
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV, K-bit TEXT

IUT:	FOR i = 1 to 64
	{
	IF(i mod 8 ≠ 0) {process all bits except parity bits}
	{	

	Perform Triple DES:	I_i = IV I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted in DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i K-bit $RESULT_i$= $LM^K(O_i) \oplus$ K-bit TEXT
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, K-bit TEXT, K-bit $RESULT_i$ $KEY1_{i+1}$ =$KEY2_{i+1}$ =$KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. NOTE: Each parity bit may have the value "1" or "0" to make the KEY odd parity.
	} }	
TMOVS:	Compare results of the 56 encryptions with known answers. Use K bits of the results in Table A.2.	

As summarized in Table 39, the Variable Key Known Answer Test for the TCFB mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

      NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. Initializes the K-bit TEXT to the constant hexadecimal value 0. It is represented as K binary bits, where K=1,...,64, i.e., $TEXT_{bin} = 0^1\ 0^2,...,0^K$. This is then translated into hexadecimal.

   d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i = 1 to 56:

NOTE -- 56 is the number of significant bits in a TDES key.

   a. Assign the value of the initialization vector IV to the input block I_i.

   b. Process I_i through the three DEA stages, resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

   c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i with the K-bit TEXT, i.e., $(RESULT^1{}_i, RESULT^2{}_i,..., RESULT^K{}_i) = (O^1{}_i \oplus TEXT^1, O^2{}_i \oplus TEXT^2,..., O^K{}_i \oplus TEXT^K)$.

   d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, K-bit TEXT, and the resulting K-bit $RESULT_i$ to the TMOVS as specified in Output Type 2.

   e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the vector consisting of “0" in every significant bit position except for a single “1" bit in position i+1. The parity bits contain “1" or “0" to make odd parity.

NOTE -- This processing should continue until every significant basis vector has been represented by the KEY parameters. The output from the IUT should consist of 56 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.2. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT in Table A.2 are used.

5.4.1.4 The Permutation Operation Known Answer Test - TCFB Mode

Table 40 The Permutation Operation Known Answer Test - TCFB Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i =1..32) = 32 KEY values in Table A.3 IV = 0000000000000000 K-bit TEXT = 0
	Send	K-bit TEXT, IV, KEY_1, KEY_2,...,KEY_{32} (Since all three keys are the same, these KEY values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 32 {	
	Perform Triple DES:	I_i = IV I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i K-bit $RESULT_i = LM^K(O_i) \oplus$ K-bit TEXT
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, K-bit TEXT, K-bit $RESULT_i$ $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = corresponding KEY_{i+1} from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. Use Table A.3.	

As summarized in Table 40, the Permutation Operation Known Answer Test for the TCFB mode of operation is performed as follows:

1. The TMOVS:

    a. Initializes the KEY parameters KEY1, KEY2, and KEY3 with the 32 constant KEY values from Table A.3.

b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 8.

2. The IUT should perform the following for i=1 through 32:

a. Assign the value of the initialization vector IV to the input block I_i.

b. Process I_i through the three DEA stages resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i with the K-bit TEXT, i.e., $(RESULT^1{}_i, RESULT^2{}_i,..., RESULT^K{}_i) = (O^1{}_i \oplus TEXT^1, O^2{}_i \oplus TEXT^2,..., O^K{}_i \oplus TEXT^K)$.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, K-bit $TEXT_i$, and the resulting K-bit $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.3. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in Table A.3 are used.

5.4.1.5 The Substitution Table Known Answer Test - TCFB Mode

Table 41 The Substitution Table Known Answer Test - TCFB Mode

TMOVS:	Initialize	$KEY1_i$,$KEY2_i$,$KEY3_i$ (where i=1..19) = 19 KEY values in Table A.4 IV_i (where i=1..19) = 19 corresponding TEXT values in Table A.4 K-bit TEXT = 0
	Send	K-bit TEXT, 19, KEY_1, IV_1, KEY_2, IV_2,..., KEY_{19}, IV_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2 and KEY3.)
IUT:	FOR i = 1 to 19 {	
	Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i K-bit $RESULT_i$= $LM^K(O_i) \oplus$ K-bit TEXT
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV_i, K-bit TEXT, K-bit $RESULT_i$ $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS IV_{i+1} = corresponding $DATA_{i+1}$ from TMOVS
	}	
TMOVS:	Compare results from each loop with known answers. Use Table A.4.	

As summarized in Table 41, the Substitution Table Known Answer Test for the TCFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY-IV pairs with the 19 constant KEY-DATA values from Table A.4. The DATA values are assigned to the values of the initialization vectors IV. The KEY value indicates the values of KEY1, KEY2, and KEY3, i.e., KEY1=KEY2=KEY3.

b. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, where K=1,...,64, i.e., $TEXT_{bin} = 0^1, 0^2,..., 0^K$.

c. Forwards this information to the IUT using Input Type 11.

2. The IUT should perform the following for i=1 through 19:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i, $LM^K(O_i)$, with the K-bit TEXT, i.e., $(RESULT^1{}_i, RESULT^2{}_i,..., RESULT^K{}_i) = (O^1{}_i \oplus TEXT^1, O^2{}_i \oplus TEXT^2,..., O^K{}_i \oplus TEXT^K)$.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV, K-bit $TEXT_i$, and the resulting K-bit $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

f. Set IV_{i+1} equal to the corresponding $DATA_{i+1}$ supplied by the TMOVS.

NOTE -- The above processing should continue until all 19 KEY-DATA are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.4. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in the Table A.4 are used.

5.4.2 The Monte Carlo Tests - TCFB Mode

The Monte Carlo Tests required to validate an IUT for the TCFB mode of operation are determined by the process or processes allowed by an IUT. The K-bit TCFB Monte Carlo Test for the Encryption Process is successfully completed if an IUT supports the encryption process of the TCFB mode of operation. The K-bit TCFB Monte Carlo Test for the Decryption Process is successfully completed if an IUT supports the decryption process.

5.4.2.1 The Monte Carlo Test for the Encryption Process - TCFB Mode

Table 42 The Monte Carlo Test for the Encryption Process - TCFB Mode

TMOVS:	Initialize	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV, K-bit P_0
	Send	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV, K-bit P_0
IUT:	FOR i = 0 TO 399	
	{	
(Part of Triple DES processing):	If (i==0) I_0 = IV	
	Record i, $KEY1_i$,$KEY2_i$,$KEY3_i$, P_0	
	FOR j = 0 TO 9,999	
	{	
Perform Triple DES:	I_j is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j. Select the leftmost K bits of the O_j, $LM^K(O_j)$, discarding the rest. K-bit $C_j = LM^K(O_j) \oplus$ K-bit P_j	
	K-bit $P_{j+1} = LM^K(I_j)$	
(Part of Triple DES processing):	$I_{j+1} = RM^{(64-K)}(I_j) \parallel$ K-bit C_j	

}

Record K-bit C_j, I_0

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, I_0, K-bit P_0, K-bit C_j

Concatenate enough Cs together to get (length(KEY)*3) bits (192 bits)

$KEY1_{i+1} = KEY1_i \oplus$ bits 129-192 of C

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus$ bits 65-128 of C

ELSE

$KEY2_{i+1} = KEY2_i \oplus$ bits 129-192 of C

IF ($KEY1_i$=$KEY2_i$=$KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus$ bits 129-192 of C

ELSE

$KEY3_{i+1} = KEY3_i \oplus$ bits 1-64 of C

K-bit $P_0 = LM^K(I_j)$

$I_0 = RM^{(64-K)}(I_j) \parallel$ K-bit C_j

}

TMOVS: Check the IUT's output for correctness.

As summarized in Table 42, the Monte Carlo Test for the TCFB Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vector IV, and the plaintext P variables. The IV, and KEYs consist of 64 bits each. The P is represented as K-bits, where K=1,...,64.

   b. Forwards this information to the IUT using Input Type 21.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), assign the value of the initialization vector IV to the input block I_i.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, and the K-bit P_0.

   c. Perform the following for j = 0 through 9999:

      1) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in output block O_j. This involves processing I_j through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_j.

      2) Calculate the K-bit C_j by exclusive-ORing the leftmost K-bits of O_j, $LM^K(O_j)$, with the K-bit P_j, i.e., $(C^1_j, C^2_j,..., C^K_j) = (O^1_j \oplus P^1_j, O^2_j \oplus P^2_j,..., O^K_j \oplus P^K_j)$.

      3) Prepare for loop j+1 by doing the following:

         a) Assign the K-bit P_{j+1} with the value of the leftmost K-bits of the I_j, i.e., $(P^1_{j+1}, P^2_{j+1},..., P^K_{j+1}) = (I^1_j, I^2_j,..., I^K_j)$.

         b) Assign I_{j+1} with the value of the concatenation of the rightmost (64-K) bits of I_j with the K-bit C_j, i.e.,$(I^1_{j+1}, I^2_{j+1},...,I^{64}_{j+1}) = (I^{[K+1]}_j, I^{[K+2]}_j,..., I^{64}_j, C^1_j, C^2_j,...,C^K_j)$.

   d. Record the K-bit C_j and I_0.

   e. Forward all recorded values for this loop, as specified in Output Type 6, to the TMOVS.

   f. In preparation for the next output loop:

      1) Assign new values to the KEY parameters KEY1, KEY2, and KEY3. This is accomplished by exclusive-ORing C with the KEY value to obtain the new KEY. If the length of the C is less than 64 (the length of a DES key), then C should be expanded in length to 64*3 (to correspond to the combined lengths of KEY1+KEY2+KEY3) before forming the new KEY values. This expansion should be accomplished by concatenating X of the most current Cs together to obtain 192 bits of C. For example, if the length of the C is 50 bits (K=50), the expanded C = $(C^9_{9996},...,C^{50}_{9996}, C^1_{9997},..., C^{50}_{9997}, C^1_{9998},...,C^{50}_{9998}, C^1_{9999},...,C^{50}_{9999})$.

Bits 129-192 of the expanded C will be exclusive-ORed with KEY1 to form the new KEY1.

The calculation of the new KEY2 and KEY3 are based on the values of the keys. . If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or KEY1, KEY2 and KEY3 are independent, the new KEY2 should be calculated by exclusive-ORing the current KEY2 with bits 65-128 of the expanded C. If KEY1=KEY2=KEY3, the current KEY2 will be exclusive-ORed with bits 129-192 of the expanded C to calculate the new KEY2.

If KEY1, KEY2, and KEY3 are independent, the new KEY3 should be calculated by exclusive-ORing the current KEY3 with bits 1-64 of the expanded C. Otherwise, the current KEY3 will be exclusive-ORed with bits 129-192 of the expanded C to calculate the new KEY3.

2) Assign a new value to the K-bit P_0. The K-bit P_0 should be assigned the value of the leftmost K-bits of the current I_j, i.e., $(P^1{}_0, P^2{}_0,...,P^K{}_0) = (I^1{}_j, I^2{}_j,..., I^K{}_j)$. Note j = 9999.

3) Assign a new value to I_0. I_0 should be assigned the value of the rightmost (64-K) bits of the current I_j concatenated with the current K-bit C_j, i.e., $(I^1{}_0, I^2{}_0,..., I^{64}{}_0) = (I^{[K+1]}{}_j, I^{[K+2]}{}_j,..., I^{64}{}_j, C^1{}_j, C^2{}_j,..., C^K{}_j)$. Note j = 9999.

NOTE -- the new P and I should be denoted as P_0 and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 6.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.4.2.2 The Monte Carlo Test for the Decryption Process - TCFB Mode

Table 43 The Monte Carlo Test for the Decryption Process - TCFB Mode

TMOVS: Initialize $KEY1_0$,$KEY2_0$,$KEY3_0$, IV, K-bit C_0

Send $KEY1_0$,$KEY2_0$,$KEY3_0$, IV, K-bit C_0

IUT: FOR i = 0 TO 399

{

(Part of Triple DES processing):

If (i==0) I_0 = IV

Record i, $KEY1_i$,$KEY2_i$,$KEY3_i$, K-bit C_0

FOR j = 0 TO 9,999

{

Perform Triple DES:

I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1

TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j

Select the leftmost K bits of the O_i, $LM^K(O_i)$,discarding the rest.

K-bit $P_j = LM^K(O_j) \oplus$ K-bit C_j

$I_{j+1} = RM^{(64-K)}(I_j) \parallel$ K-bit C_j

K-bit $C_{j+1} = LM^K(O_j)$

}

Record I_0, K-bit P_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, I_0, K-bit P_j, K-bit C_j

Concatenate enough Ps together to get (length(KEY)*3) bits (192 bits)

$KEY1_{i+1} = KEY1_i \oplus$ bits 129-192 of P

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus$ bits 65-128 of P

ELSE

$KEY2_{i+1} = KEY2_i \oplus$ bits 129-192 of P

IF ($KEY1_i$=$KEY2_i$=$KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus$ bits 129-192 of P

ELSE

$KEY3_{i+1} = KEY3_i \oplus$ bits 1-64 of P

$I_0 = RM^{(64\text{-}K)}(I_j) \| \text{K-bit } C_j$

K-bit $C_0 = LM^K(O_j)$

}

TMOVS: Check the IUT's output for correctness.

As summarized in Table 43, the Monte Carlo Test for the TCFB Decryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3, the initialization vector IV, and the ciphertext C variables. The IV and KEYs consist of 64 bits each. The C is represented as K-bits, where K=1,...,64.

   b. Forwards this information to the IUT using Input Type 21.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), assign the value of the initialization vector IV to the input block I_i.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, and the K-bit C_0.

   c. Perform the following for j = 0 through 9999:

1) Using the corresponding $KEY1_i$, $KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in output block O_j. This involves processing I_j through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_j.

2) Calculate the K-bit P_j by exclusive-ORing the leftmost K-bits of O_j, $LM^K(O_j)$, with the K-bit C_j, i.e., $(P^1{}_j, P^2{}_j,..., P^K{}_j) = (O^1{}_j \oplus C^1{}_j, O^2{}_j \oplus C^2{}_j,..., O^K{}_j \oplus C^K{}_j)$.

3) Prepare for loop j+1 by doing the following:

a) Assign I_{j+1} with the value of the concatenation of the rightmost (64-K) bits of I_j with the K-bit C_j, i.e.,$(I^1{}_{j+1}, I^2{}_{j+1},...,I^{64}{}_{j+1}) = (I^{[K+1]}{}_j, I^{[K+2]}{}_j,..., I^{64}{}_j, C^1{}_j, C^2{}_j,...,C^K{}_j)$.

b) Assign the K-bit C_{j+1} with the value of the leftmost K-bits of the O_j, i.e., $(C^1{}_{j+1}, C^2{}_{j+1},..., C^K{}_{j+1}) = (O^1{}_j, O^2{}_j,..., O^K{}_j)$.

d. Record the K-bit P_j and I_0.

e. Forward all recorded values for this loop, as specified in Output Type 6, to the TMOVS.

f. In preparation for the next output loop:

1) Assign new values to the KEY parameters KEY1, KEY2, and KEY3. This is accomplished by exclusive-ORing P with the KEY value to obtain the new KEY. If the length of the P is less than 64 (the length of a DES key), then P should be expanded in length to 64*3 (to correspond to the combined lengths of KEY1+KEY2+KEY3) before forming the new KEY values. This expansion should be accomplished by concatenating X of the most current Ps together to obtain 192 bits of P. For example, if the length of the P is 50 bits (K=50), the expanded P = $(P^9{}_{9996},...,P^{50}{}_{9996}, P^1{}_{9997},..., P^{50}{}_{9997}, P^1{}_{9998},...,P^{50}{}_{9998}, P^1{}_{9999},...,P^{50}{}_{9999})$.

Bits 129-192 of the expanded P will be exclusive-ORed with KEY1 to form the new KEY1.

The calculation of the new KEY2 and KEY3 are based on the values of the keys. . If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or KEY1, KEY2 and KEY3 are independent, the new KEY2 should be calculated by exclusive-ORing the current KEY2 with bits 65-128 of the expanded P. If KEY1=KEY2=KEY3, the current KEY2 will be exclusive-ORed with bits 129-192 of the expanded P to calculate the new KEY2.

If KEY1, KEY2, and KEY3 are independent, the new KEY3 should be calculated by exclusive-ORing the current KEY3 with bits 1-64 of the expanded P. Otherwise, the current KEY3 will be exclusive-ORed with bits 129-192 of the expanded P to calculate the new KEY3.

2) Assign a new value to I_0. I_0 should be assigned the value of the rightmost (64-K) bits of the current I_j concatenated with the current K-bit C_j, i.e., $(I^1{}_0, I^2{}_0,...,I^{64}{}_0) = (I^{[K+1]}{}_j, I^{[K+2]}{}_j,..., I^{64}{}_j, C^1{}_j, C^2{}_j,..., C^K{}_j)$. Note j = 9999.

3) Assign a new value to the K-bit C_0. The K-bit C_0 should be assigned the value of the leftmost K-bits of the current O_j, i.e., $(C^1{}_0, C^2{}_0,..., C^K{}_0) = (O^1{}_j, O^2{}_j,..., O^K{}_j)$. Note j = 9999.

NOTE -- the new C and I should be denoted as C_0 and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 6.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.5 The Cipher Feedback (CFB-P) Mode

The IUTs that implement the Cipher Block Feedback - Pipelined (CFB-P) mode of operation are validated by successfully completing (1) a set of Known Answer tests applicable to both IUTs supporting encryption and/or decryption and (2) a Monte Carlo test designed for use with both the encryption process and the decryption process.

The pipelined configuration is intended for systems equipped with multiple DEA processors. By pipelining the data, throughput is improved and propagation delay is minimized by initializing the three individual DEA stages and then simultaneously clocking them. Thus, with each clock cycle, data is processed by each DEA_i stage and passed onward to the output buffer or the next stage so that idle DEA_i stages are minimized.

The processing for each Known Answer test and Monte Carlo test is broken down into clock cycles T1, T2, T3,.... Within each clock cycle, the processing occurring on each active DEA is discussed. For convenience, let KEY1 represent the key used on processor DEA_1, KEY2 represent the key used on processor DEA_2, and KEY3 represent the key used on processor DEA_3.

The process of validating an IUT which only supports the K-bit TCFB-P mode in either the encryption and/or decryption processes involves the successful completion of the following six tests:

1. The Variable Text Known Answer Test - K-bit TCFB-P mode

2. The Inverse Permutation Known Answer Test - K-bit TCFB-P mode

3. The Variable Key Known Answer Test - K-bit TCFB-P mode

4. The Permutation Operation Known Answer Test - K-bit TCFB-P mode

5. The Substitution Table Known Answer Test - K-bit TCFB-P mode

6. The Monte Carlo Test - K-bit TCFB-P mode

NOTE -- for IUTs, K can range from 1 to 64 bits.

The notation $LM^K(A)$ refers to the leftmost K bits of A.

An explanation of the tests follows.

5.5.1 The Known Answer Tests - TCFB-P Mode

The K-bit TCFB-P mode has only one set of Known Answer tests which are used regardless of process, i.e., the same set of Known Answer tests are used for IUTs supporting the encryption and/or decryption processes.

Throughout this section, TEXT and RESULT will refer to different variables depending on whether the encryption or decryption process is being tested. If the IUT performs TCFB-P encryption, TEXT refers to plaintext, and RESULT refers to ciphertext. If the IUT performs TCFB-P decryption, TEXT refers to ciphertext, and RESULT refers to plaintext.

5.5.1.1 The Variable TEXT Known Answer Test - TCFB-P Mode

Table 44 The Variable TEXT Known Answer Test - TCFB-P Mode

TMOVS:	Initialize	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set) $IV1_1$ = 8000000000000000 $IV2_1$ = D555555555555555 (based on specifications in ANSI X9.52-1998) $IV3_1$ = 2AAAAAAAAAAAAAAA (based on specifications in ANSI X9.52-1998) K-bit TEXT = 0
	Send	KEY (representing KEY1, KEY2, and KEY3), $IV1_1$, $IV2_1$, $IV3_1$, K-bit TEXT
IUT:	FOR i = 1 to 64 {	
	Perform Triple DES:	With the feedback path disconnected: T1: $I1 = IV1_i$ I1 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$ T2: $I2 = IV2_i$ I2is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$ T3: $I3 = IV3_i$ I3 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$ Connect the feedback path: $TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $O1_i$

K-bit $RESULT1_i = LM^K(O1_i) \oplus$ K-bit TEXT

T4: $TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using KEY3, resulting in $O2_i$

K-bit $RESULT2_i = LM^K(O2_i) \oplus$ K-bit TEXT

T5: $TEMP3_2$ is encrypted by DEA_3 using KEY3, resulting in $O3_i$

K-bit $RESULT3_i = LM^K(O3_i) \oplus$ K-bit TEXT

Send i, KEY (representing KEY1, KEY2, and KEY3), I1, I2, I3, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, K-bit $RESULT3_i$

$IV1_{i+1}$ = basis vector where single "1" bit is in position i+1

$IV2_{i+1} = IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$IV3_{i+1} = IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

}

TMOVS: Compare RESULT1, RESULT2, and RESULT3 from each loop with known answers.

Use K bits of output in Table A.9.

As summarized in Table 44, the Variable TEXT Known Answer Test for the TCFB-P mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}=KEY2_{hex}=KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 3 initialization vectors accordingly: $IV1_1$ is set to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, $IV2_1$ is computed by the following equation: $IV1_1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555 . In hexadecimal, this equates to D5 55 55 55 55 55 55 55. And $IV3_1$ is computed by the equation $IV1_1 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA . In hexadecimal, this equates to 2A AA AA AA AA AA AA AA.

c. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O1_i$.

e) Calculate the K-bit $RESULT1_i$ by exclusive-ORing the leftmost K-bits of $O1_i$ with the K-bit TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O2_i$.

b) Calculate the K-bit $RESULT2_i$ by exclusive-ORing the leftmost K-bits of $O2_i$ with the K-bit TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O3_i$.

b) Calculate the K-bit $RESULT3_i$ by exclusive-ORing the leftmost K-bits of $O3_i$ with the K-bit TEXT.

b. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), $IV1_i$, $IV2_i$, $IV3_i$, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, and K-bit $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Retain the K-bit RESULT1, RESULT2, and RESULT3 values for use with the Inverse Permutation Known Answer Test for the TCFB-P Mode (Section 5.5.1.2).

d. Assign a new value to the IV variables. $IV1_{i+1}$ is set to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64. $IV2_{i+1}$ is set to the value of $IV1_{i+1}$ + R_1 mod 2^{64} where R_1=5555555555555555. And $IV3_{i+1}$ is set to $IV1_{i+1}$ + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA.

NOTE -- This continues until every possible basis vector has been represented by the IV1, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.9. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in the Table A.9 are used.

5.5.1.2 The Inverse Permutation Known Answer Test - TCFB-P Mode

Table 45 The Inverse Permutation Known Answer Test - TCFB-P Mode

TMOVS:	Initialize	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		$IV1_1$ = 8000000000000000
		$IV2_1$ = D555555555555555
		$IV3_1$ = 2AAAAAAAAAAAAAAA
		K-bit $TEXTr_i$ (where r=1..3 and i=1..64) = 64 corresponding $RESULTr_i$ values from Variable TEXT Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), $IV1_1$, $IV2_1$, $IV3_1$, K-bit $TEXT1_1$,..., K-bit $TEXT1_{64}$, K-bit $TEXT2_1$,..., K-bit $TEXT2_{64}$, K-bit $TEXT3_1$,..., K-bit $TEXT3_{64}$
IUT:	FOR i = 1 to 64	
	{	
		With the feedback path disconnected:
		T1: I1 = $IV1_i$
	Perform Triple DES:	I1 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$
		T2: I2 = $IV2_i$
		I2 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$
		$TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$
		T3: I3 = $IV3_i$
		I3 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$
		$TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$
		Connect the feedback path:
		$TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $O1_i$

K-bit $RESULT1_i = LM^K(O1_i) \oplus$ K-bit $TEXT1_i$

T4: $TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using KEY3, resulting in $O2_i$

K-bit $RESULT2_i = LM^K(O2_i) \oplus$ K-bit $TEXT2_i$

T5: $TEMP3_2$ is encrypted by DEA_3 using KEY3, resulting in $O3_i$

K-bit $RESULT3_i = LM^K(O3_i) \oplus$ K-bit $TEXT3_i$

Send i, KEY (representing KEY1, KEY2, and KEY3), I1, I2, I3, K-bit $TEXT1_i$, , K-bit $TEXT2_i$, K-bit $TEXT3_i$, K-bit $RESULT1_i$, K-bit $RESULT2_i$, K-bit $RESULT3_i$

$IV1_{i+1}$ = basis vector where single "1" bit is in position i+1

$IV2_{i+1} = IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$IV3_{i+1} = IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

K-bit $TEXTr_{i+1}$ (where r = 1..3)= corresponding K-bit $RESULTr_{i+1}$ value from the TMOVS

}

TMOVS: Compare RESULT1, RESULT2, and RESULT3 from each loop with known answers,

They should be all zeros.

As summarized in Table 45 the Inverse Permutation Known Answer Test for the TCFB-P mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 3 initialization vectors accordingly: $IV1_1$ is set to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., IV_{1bin} = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, $IV2_1$ is computed by the following equation: $IV1_1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555 . In hexadecimal, this equates to D5 55 55 55 55 55 55 55. And $IV3_1$ is computed by the equation $IV1_1 + R_2$

mod 2^{64} where R_2=AAAAAAAAAAAAAAAA . In hexadecimal, this equates to 2A AA AA AA AA AA AA AA.

c. Initializes the K-bit $TEXTr_i$ (where r=1,…,3 and i=1,…,64) to the $RESULTr_i$ obtained from the Variable TEXT Known Answer Test.

d. Forwards this information to the IUT using Input Type 15.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O1_i$.

e) Calculate the K-bit $RESULT1_i$ by exclusive-ORing the leftmost K-bits of $O1_i$ with the K-bit $TEXT1_i$.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O2_i$.

b) Calculate the K-bit $RESULT2_i$ by exclusive-ORing the leftmost K-bits of $O2_i$ with the K-bit $TEXT2_i$.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O3_i$.

b) Calculate the K-bit $RESULT3_i$ by exclusive-ORing the leftmost K-bits of $O3_i$ with the K-bit $TEXT3_i$.

b. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), $IV1_i$, $IV2_i$, $IV3_i$, K-bit $TEXT1_i$, K-bit $TEXT2_i$, K-bit $TEXT3_i$, K-bit $RESULT1_i$, K-bit $RESULT2_i$, and K-bit $RESULT3_i$, to the TMOVS as specified in Output Type 3.

c. Assign a new value to the IV variables. $IV1_{i+1}$ is set to the value of a basis vector with a "1" bit in position i+1, where i+1=2,…,64. $IV2_{i+1}$ is set to the value of $IV1_{i+1}$ + R_1 mod 2^{64} where R_1=5555555555555555. And $IV3_{i+1}$ is set to $IV1_{i+1}$ + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA.

d. Assign a new value to the K-bit $TEXT1_{i+1}$, K-bit $TEXT2_{i+1}$, and K-bit $TEXT3_{i+1}$ by setting it equal to the corresponding output from the TMOVS.

NOTE -- This continues until every RESULT1, RESULT2, and RESULT3 value from the Variable TEXT Known Answer Test has been used as input. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The RESULT1, RESULT2, and RESULT3 values should be all zeros.

5.5.1.3 The Variable KEY Known Answer Test - TCFB-P Mode

Table 46 The Variable KEY Known Answer Test - TCFB-P Mode

TMOVS:	Initialize	$KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set) IV1 = 0000000000000000 IV2 = 5555555555555555 IV3 = AAAAAAAAAAAAAAAA K-bit TEXT = 0
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV1, IV2, IV3, K-bit TEXT
IUT:	FOR i = 1 to 64	
	IF (i mod 8≠0){process all bits except parity bits}	
	{	
	Perform Triple DES:	With the feedback path disconnected: T1: I1 = IV1 I1 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$ T2: I2 = IV2 I2 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$ T3: I3 = IV3 I3 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$ Connect the feedback path: $TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O1_i$

K-bit $RESULT1_i = LM^K(O1_i) \oplus$ K-bit TEXT

T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O2_i$

K-bit $RESULT2_i = LM^K(O2_i) \oplus$ K-bit TEXT

T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O3_i$

K-bit $RESULT3_i = LM^K(O3_i) \oplus$ K-bit TEXT

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), I1, I2, I3, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, K-bit $RESULT3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$= vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. Each parity bit may have the value "1" or "0" to make the KEY odd parity.

}

TMOVS: Compare results of the 56 encryptions with known answers.

Use K bits of the results in Table A.11.

As summarized in Table 46, the Variable KEY Known Answer Test for the TCFB-P Mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}= KEY2_{1\ bin}= KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

      NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 3 initialization vectors accordingly: IV1 is assigned the value 0, i.e., $IV1_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R_1 mod 2^{64} where R_1=5555555555555555. In hexadecimal, this equates to 55 55 55 55 55 55 55 55. And IV3 is computed by the equation IV1 + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA. In hexadecimal, this equates to AA AA AA AA AA AA AA AA.

c. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector IV1 to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector IV2 to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector IV3 to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the K-bit $RESULT1_i$ by exclusive-ORing the leftmost K-bits of $O1_i$ with the K-bit TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate the K-bit $RESULT2_i$ by exclusive-ORing the leftmost K-bits of $O2_i$ with the K-bit TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the K-bit $RESULT3_i$ by exclusive-ORing the leftmost K-bits of $O3_i$ with the K-bit TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, and K-bit $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits contain "1" or "0" to make odd parity.

NOTE -- This processing should continue until every significant basis vector has been represented by the KEY parameters. The output from the IUT should consist of 56 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.11. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in the Table A.11 are used.

5.5.1.4 The Permutation Operation Known Answer Test - TCFB-P Mode

Table 47 The Permutation Operation Known Answer Test - TCFB-P Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..32) = 32 KEY values in Table A.12 IV1= 0000000000000000 IV2 =5555555555555555 IV3 =AAAAAAAAAAAAAAAA K-bit TEXT = 0
	Send	IV1, IV2, IV3, K-bit TEXT, KEY_1, KEY_2,...,KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 32 {	
Perform Triple DES:	With the feedback path disconnected: T1:	I1 = IV1 I1 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP1_1$
	T2:	I2 = IV2 I2 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA using $KEY2_i$, resulting in $TEMP1_2$
	T3:	I3 = IV3 I3 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA using $KEY2_i$, resulting in $TEMP2_2$
	Connect the feedback path:	
		$TEMP1_2$ is encrypted by DEA using $KEY3_i$, resulting in $O1_i$

K-bit $RESULT1_i = LM^K(O1_i) \oplus$ K-bit TEXT

T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O2_i$

K-bit $RESULT2_i = LM^K(O2_i) \oplus$ K-bit TEXT

T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O3_i$

K-bit $RESULT3_i = LM^K(O3_i) \oplus$ K-bit TEXT

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), I1, I2, I3, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, K-bit $RESULT3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$= corresponding KEY_{i+1} from TMOVS.

TMOVS: Compare results from each loop with known answers.

Use K bits of output in Table A.12.

As summarized in Table 47, the Permutation Operation Known Answer Test for the TCFB-P mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 with the 32 constant KEY values from Table A.12.

   b. Initializes the 3 initialization vectors accordingly: IV1 is assigned the value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: $IV1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555. In hexadecimal, this equates to 55 55 55 55 55 55 55 55. And IV3 is computed by the equation $IV1 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA. In hexadecimal, this equates to AA AA AA AA AA AA AA AA.

   c. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 18.

2. The IUT should perform the following for i=1 through 32:

   a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector IV1 to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector IV2 to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector IV3 to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the K-bit $RESULT1_i$ by exclusive-ORing the leftmost K-bits of $O1_i$ with the K-bit TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate the K-bit $RESULT2_i$ by exclusive-ORing the leftmost K-bits of $O2_i$ with the K-bit TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the K-bit $RESULT3_i$ by exclusive-ORing the leftmost K-bits of $O3_i$ with the K-bit TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, and K-bit $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found Table A.12. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in Table A.12 are used.

5.5.1.5 The Substitution Table Known Answer Test - TCFB-P Mode

Table 48 The Substitution Table Known Answer Test - TCFB-P Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..19) = 19 KEY values in Table A.10
		$IV1_i$ (where i=1..19) = 19 corresponding TEXT values in Table A.10
		$IV2_i = IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555
		$IV3_i = IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA
		K-bit TEXT = 0
	Send	$IV1_i$, $IV2_i$, $IV3_i$, K-bit TEXT, KEY_1, KEY_2,...,KEY_{19} (These key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 19	
	{	
	Perform Triple DES:	With the feedback path disconnected:
		T1: $I1 = IV1_i$
		I1 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$
		T2: $I2 = IV2_i$
		I2 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$
		$TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$
		T3: $I3 = IV3_i$
		I3 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$
		$TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$
		Connect the feedback path:
		$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O1_i$

K-bit $RESULT1_i = LM^K(O1_i) \oplus$ K-bit TEXT

T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O2_i$

K-bit $RESULT2_i = LM^K(O2_i) \oplus$ K-bit TEXT

T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O3_i$

K-bit $RESULT3_i = LM^K(O3_i) \oplus$ K-bit TEXT

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), $IV1_i$, $IV2_i$, $IV3_i$, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, K-bit $RESULT3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$= Corresponding KEY_{i+1} from TMOVS

$IV1_{i+1}$= corresponding $DATA_{i+1}$ from TMOVS

$IV2_{i+1} = IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$IV3_{i+1} = IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

}

TMOVS: Compare results from each loop with known answers.

Use K bits of output in Table A.10.

As summarized in Table 48, the Substitution Table Known Answer Test for the TCFB-P Mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY-IV1 pairs with the 19 constant KEY-DATA values from Table A.10. The DATA values are assigned to the values of the initialization vectors $IV1_i$. The KEY value indicates the values of $KEY1_i$, $KEY2_i$, and $KEY3_i$, i.e., $KEY1_i$=$KEY2_i$=$KEY3_i$. Based on specifications in ANSI X9.52-1998, $IV2_i$ is set to $IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555 and $IV3_i$ is set to $IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA.

   b. Initializes the K-bit TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

   c. Forwards this information to the IUT using Input Type 25.

2. The IUT should perform the following for i=1 through 19:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the K-bit $RESULT1_i$ by exclusive-ORing the leftmost K-bits of $O1_i$ with the K-bit TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate the K-bit $RESULT2_i$ by exclusive-ORing the leftmost K-bits of $O2_i$ with the K-bit TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the K-bit $RESULT3_i$ by exclusive-ORing the leftmost K-bits of $O3_i$ with the K-bit TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), $IV1_i$, $IV2_i$, $IV3_i$, K-bit TEXT, K-bit $RESULT1_i$, K-bit $RESULT2_i$, and K-bit $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

d. Set $IV1_{i+1}$ equal to the corresponding $DATA_{i+1}$ supplied by the TMOVS. Based on specifications in ANSI X9.52-1998, $IV2_{i+1}$ is set to $IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555 and $IV3_{i+1}$ is set to $IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA.

NOTE -- The above processing should continue until all 19 KEY-DATA are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.10. For IUTs where K is less than 64, the leftmost K bits of output for each RESULT value in the Table A.10 are used.

5.5.2 The Monte Carlo Tests - TCFB-P Mode

The Monte Carlo tests required to validate an IUT for the K-bit TCFB-P mode of operation are determined by the process or processes allowed by an IUT. The K-bit TCFB-P Monte Carlo Test for the Encryption Process is successfully completed if an IUT supports the encryption process of the TCFB-P mode of operation. The K-bit TCFB-P Monte Carlo Test for the Decryption Process is successfully completed if an IUT supports the decryption process.

5.5.2.1 The Monte Carlo Test for the Encryption Process – K-bit TCFB-P Mode

Table 49 The Monte Carlo Test for the Encryption Process - K-bit TCFB-P Mode

TMOVS:	Initialize	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, K-bit P_0
	Send	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, K-bit P_0
IUT:	FOR i = 0 TO 399	
	{	
	FOR j = 0 TO 9,999	
	{	

	Perform Triple DES:	IF (j == 0, 1, or 2)
		Ij = IV (j+1)
		ELSE
		Ij = $RM^{(64-K)}$ (I(j-1)) ‖ K-bit C_{j-3}
		Ij is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1
		TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2
		TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j
		K-bit $C_j = LM^K (O_j) \oplus$ K-bit P_j

		K-bit $P_{j+1} = LM^K$ (Ij)
		}
	Record I0, C_j	
	Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, I0, I1, I2, K-bit P_0 , K-bit C_j	

Concatenate enough C together to get (length(KEY)*3) bits (192 bits)

$KEY1_{i+1} = KEY1_i \oplus$ bits 129-192 of C

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus$ bits 65-128 of C

ELSE

$KEY2_{i+1} = KEY2_i \oplus$ bits 129-192 of C

IF ($KEY1_i$=$KEY2_i$=$KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus$ bits 129-192 of C

ELSE

$KEY3_{i+1} = KEY3_i \oplus$ bits 1-64 of C

K-bit $P_0 = LM^K(Ij)$

$I0 = RM^{(64-K)}(Ij) \| \text{K-bit } C_j$

$I1 = I0 + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$I2 = I0 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

}

TMOVS: Check the IUT's output for correctness.

As summarized in Table 49, the Monte Carlo Test for the TCFB-P Encryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_0$, $KEY2_0$, and $KEY3_0$, the initialization vectors IV1, IV2, and IV3, and the K-bit plaintext P_0. The IVs, and KEYs consist of 64 bits each. The P is represented as K-bits, where K=1,...,64. IV2 is assigned the value of IV1+R_1 mod 2^{64} where R_1 = 5555555555555555. IV3 is assigned the value of IV1+R_2 mod 2^{64} where R_2 = AAAAAAAAAAAAAAAA.

   b. Forwards this information to the IUT using Input Type 24.

2. The IUT should perform the following for i = 0 through 399:

a. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, IV1, IV2, IV3, and P_0.

b. Perform the following for j = 0 through 9,999:

1) If j = 0, 1, or 2, assign the value of the initialization vector IV_{j+1} to the input block Ij.

2) If j > 2, assign Ij with the value of the concatenation of the rightmost (64-K) bits of I(j-1) with the K-bit C_{j-3}.

3) Process I_j through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1.

4) TEMP1 is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2.

5) TEMP2 is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block O_j.

6) Calculate the K-bit C_j by exclusive-ORing the leftmost K-bits of O_j with the K-bit P_j.

7) Prepare for loop j+1 by assigning the K-bit P_{j+1} with the value of the leftmost K-bits of the Ij.

c. Record the current values of the input block I0 and C_j.

d. Forward all recorded values for this loop, as specified in Output Type 8, to the TMOVS.

e. In preparation for the next output loop:

1) Concatenate enough C values together to obtain (length (KEY)*3) bits of data (192 bits).

2) Assign new values to the KEY parameters KEY1, KEY2, and KEY3. This is accomplished by exclusive-ORing C with the KEY value to obtain the new KEY. If the length of the C is less than 64 (the length of a DES key), the C should be expanded in length to 64*3 (to correspond to the combined lengths of KEY1+KEY2+KEY3) before forming the new KEY values. This expansion should be accomplished by concatenating X of the most current Cs together to obtain 192 bits of C. For example, if the length of the C is 50 bits (K=50), the expanded C = (C^{9}_{9996},..., C^{50}_{9996}, C^{1}_{9997} ,..., C^{50}_{9997}, C^{1}_{9998},..., C^{50}_{9998}, C^{1}_{9999},..., C^{50}_{9999}).

Bits 129-192 of the expanded C will be exclusive-ORed with KEY1 to form the new KEY1.

The calculation of the new KEY2 and KEY3 are based on the values of the keys. . If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or KEY1, KEY2 and KEY3 are independent, the new KEY2 should be calculated by exclusive-ORing the current KEY2 with bits 65-128 of the expanded C. If KEY1=KEY2=KEY3, the current KEY2 will be exclusive-ORed with bits 129-192 of the expanded C to calculate the new KEY2.

If KEY1, KEY2, and KEY3 are independent, the new KEY3 should be calculated by exclusive-ORing the current KEY3 with bits 1-64 of the expanded C. Otherwise, the current KEY3 will be exclusive-ORed with bits 129-192 of the expanded C to calculate the new KEY3.

3) Assign a new value to the K-bit P_0. The K-bit P_0 should be assigned the value of the leftmost K-bits of the current I_j, where j = 9999, i.e., $(P^1{}_0, P^2{}_0,\ldots,P^K{}_0) = (I^1{}_j, I^2{}_j,..., I^K{}_j)$.

4) Assign a new value to the I parameters. I_0 should be assigned the value of the concatenation of the rightmost (64-K) bits of I_j with the K-bit C_j, i.e. $(I^1{}_0, I^2{}_0,\ldots, I^{64}{}_0) = (I^{[K+1]}{}_j, I^{[K+2]}{}_j,\ldots,I^{64}{}_j, C^1{}_j, C^2{}_j,\ldots,C^K{}_j)$. I_1 should be assigned the value of $I_0+R_1 \bmod 2^{64}$ where R_1 = 5555555555555555. I_2 should be assigned the value of $I_0+R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA. Note j = 9999.

NOTE -- the new P and I should be denoted as P_0 and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 8.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.5.2.2 The Monte Carlo Test for the Decryption Process – K-bit TCFB-P Mode

Table 50 The Monte Carlo Test for the Decryption Process - K-bit TCFB-P Mode

TMOVS:	Initialize	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, K-bit C_0
	Send	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, K-bit C_0
IUT:	FOR i = 0 TO 399	
	{	
	FOR j = 0 TO 9,999	

{

Perform Triple DES:

IF (j == 0, 1, or 2)

$Ij = IV\ (j+1)$

ELSE

$Ij = RM^{(64\text{-}K)}\ (I(j\text{-}1))\ \|\ K\text{-bit } C_{j\text{-}3}$

Ij is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1

TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2

TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j

K-bit $P_j = LM^K\ (O_j) \oplus$ K-bit C_j

K-bit $C_{j+1} = LM^K\ (O_j)$

}

Record I0, K-bit P_j

Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, I0, I1, I2, K-bit C_0 , K-bit P_j

Concatenate enough Ps together to get (length(KEY)*3) bits (192 bits)

$KEY1_{i+1} = KEY1_i \oplus$ bits 129-192 of P

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, $KEY3_i$ are independent),

$KEY2_{i+1} = KEY2_i \oplus$ bits 65-128 of P

ELSE

$KEY2_{i+1} = KEY2_i \oplus$ bits 129-192 of P

IF ($KEY1_i$=$KEY2_i$=$KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$),

$KEY3_{i+1} = KEY3_i \oplus$ bits 129-192 of P

ELSE

$KEY3_{i+1} = KEY3_i \oplus$ bits 1-64 of P

$I0 = RM^{(64-K)}(Ij) \| \text{K-bit } C_j$

$I1 = I0 + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$I2 = I0 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

K-bit $C_0 = LM^K(Oj)$

}

TMOVS: Check the IUT's output for correctness.

As summarized in Table 50, the Monte Carlo Test for the TCFB-P Decryption Process is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_0$, $KEY2_0$, and $KEY3_0$, the initialization vectors IV1, IV2, and IV3, and the K-bit C_0. The IVs, and KEYs consist of 64 bits each. The C is represented as K-bits, where K=1,...,64. IV2 is assigned the value of $IV1+R_1 \bmod 2^{64}$ where R_1 = 5555555555555555. IV3 is assigned the value of $IV1+R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA.

   b. Forwards this information to the IUT using Input Type 24.

2. The IUT should perform the following for i = 0 through 399:

   a. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, IV1, IV2, IV3, and C_0.

   b. Perform the following for j = 0 through 9,999:

      1) If j = 0, 1, or 2, assign the value of the initialization vector IV_{j+1} to the input block Ij.

      2) If j > 2, assign Ij with the value of the concatenation of the rightmost (64-K) bits of I(j-1) with the K-bit C_{j-3}.

      3) Process I_j through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1.

      4) TEMP1 is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2.

      5) TEMP2 is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block O_j.

6) Calculate the K-bit P_j by exclusive-ORing the leftmost K-bits of O_j with the K-bit C_j.

7) Prepare for loop j+1 by assigning the K-bit C_{j+1} with the value of the leftmost K-bits of the O_j.

c. Record the current values of the input block I0 and K-bit P_j.

d. Forward all recorded values for this loop, as specified in Output Type 8, to the TMOVS.

e. In preparation for the next output loop:

1) Concatenate enough P values together to obtain (length (KEY)*3) bits of data (192 bits).

3) Assign new values to the KEY parameters KEY1, KEY2, and KEY3. This is accomplished by exclusive-ORing P with the KEY value to obtain the new KEY. If the length of the P is less than 64 (the length of a DES key), the P should be expanded in length to 64*3 (to correspond to the combined lengths of KEY1+KEY2+KEY3) before forming the new KEY values. This expansion should be accomplished by concatenating X of the most current Ps together to obtain 192 bits of P. For example, if the length of the P is 50 bits (*K*=50), the expanded P = (P^9_{9996},..., P^{50}_{9996}, P^1_{9997} ,..., P^{50}_{9997}, P^1_{9998},..., P^{50}_{9998}, P^1_{9999},..., P^{50}_{9999}).

Bits 129-192 of the expanded P will be exclusive-ORed with KEY1 to form the new KEY1.

The calculation of the new KEY2 and KEY3 are based on the values of the keys. . If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or KEY1, KEY2 and KEY3 are independent, the new KEY2 should be calculated by exclusive-ORing the current KEY2 with bits 65-128 of the expanded P. If KEY1=KEY2=KEY3, the current KEY2 will be exclusive-ORed with bits 129-192 of the expanded P to calculate the new KEY2.

If KEY1, KEY2, and KEY3 are independent, the new KEY3 should be calculated by exclusive-ORing the current KEY3 with bits 1-64 of the expanded P. Otherwise, the current KEY3 will be exclusive-ORed with bits 129-192 of the expanded P to calculate the new KEY3.

3) Assign a new value to the I parameters. I_0 should be assigned the value of the concatenation of the rightmost (64-K) bits of I_j with the K-bit C_j, i.e. (I^1_0, $I^2_0,\ldots, I^{64}_0$) = ($I^{[K+1]}_j$, $I^{[K+2]}_j,\ldots,I^{64}_j$, C^1_j, $C^2_j,\ldots,C^K_j$). I_1 should be assigned the value of $I_0+R_1 \bmod 2^{64}$ where R_1 = 5555555555555555. I_2 should be assigned the value of $I_0+R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA. Note j = 9999.

4) Assign a new value to the K-bit C_0. The K-bit C_0 should be assigned the value of the leftmost K-bits of the current O_j, where j = 9999, i.e., $(C^1{}_0, C^2{}_0,\ldots,C^K{}_0) = (O^1{}_j, O^2{}_j,\ldots, O^K{}_j)$.

NOTE -- the new C and I should be denoted as C_0 and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 8.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.6 The Output Feedback Mode - TOFB Mode

The IUTs which implement the Output Feedback (TOFB) mode of operation are validated by successfully completing a set of Known Answer tests and a Monte Carlo test applicable to both IUTs supporting encryption and/or decryption. Encryption and decryption using the TOFB mode of operation involve processing an input block through the encryption process of the specified algorithm. Therefore, the same set of Known Answer tests and Monte Carlo test can be applied to IUTs supporting both encryption and decryption.

The process of validating an IUT which supports the encryption and/or decryption processes of the TOFB mode of operation involves the successful completion of the following six tests:

1. The Variable Text Known Answer Test - TOFB mode

2 The Inverse Permutation Known Answer Test - TOFB mode

3. The Variable Key Known Answer Test - TOFB mode

4. The Permutation Operation Known Answer Test - TOFB mode

5. The Substitution Table Known Answer Test - TOFB mode

6. The Monte Carlo Test - TOFB mode

An explanation of the tests for the TOFB mode follows.

5.6.1 The Known Answer Tests - TOFB Mode

In the following description of the Known Answer tests, TEXT refers to plaintext, and RESULT refers to ciphertext if the IUT performs TOFB encryption. If the IUT supports TOFB decryption, TEXT refers to ciphertext, and RESULT refers to plaintext.

5.6.1.1 The Variable TEXT Known Answer Test - TOFB Mode

Table 51 The Variable TEXT Known Answer Test - TOFB Mode

TMOVS:	Initialize	KEYs: KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		IV_1= 8000000000000000
		TEXT = 0000000000000000
	Send	KEY (representing KEY1, KEY2, and KEY3), IV_1, TEXT
IUT:	FOR i = 1 to 64	
	{	
	Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in O_i $RESULT_i = O_i \oplus TEXT$
		Send i, KEY (representing KEY1, KEY2, and KEY3), IV_i, TEXT, $RESULT_i$
		IV_{i+1} = basis vector where single "1" bit is in position i+1
	}	
TMOVS:	Compare results from each loop with known answers. Use Table A.1.	

As summarized in Table 51, the Variable TEXT Known Answer Test for the TOFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

b. Initializes the 64-bit initialization vector IV_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

c. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i=1 through 64:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages, resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in output block O_i.

c. Calculate the K-bit $RESULT_i$ by exclusive-ORing the leftmost K-bits of O_i with the K-bit TEXT.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV_i, TEXT, and the $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Retain the RESULT values for use with the Inverse Permutation Known Answer Test for the TOFB Mode (Section 5.6.1.2).

f. Assign a new value to IV_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,…,64.

NOTE -- This continues until every possible basis vector has been represented by the IV, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.1.

5.6.1.2 The Inverse Permutation Known Answer Test - TOFB Mode

Table 52 The Inverse Permutation Known Answer Test - TOFB Mode

TMOVS:	Initialize	KEYs:KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		$TEXT_i$ (where i=1..64)=64 RESULT values from the Variable TEXT Known Answer Test
		IV_1= 8000000000000000
	Send	KEY (representing KEY1, KEY2, and KEY3), IV_1, $TEXT_1$..$TEXT_{64}$
IUT:	FOR i = 1 to 64	
	{	
	Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using KEY2, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using KEY3, resulting in O_i $RESULT_i = O_i \oplus TEXT_I$
		Send i, KEY (representing KEY1, KEY2, and KEY3), IV_i, $TEXT_i$, $RESULT_i$
		IV_{i+1} = basis vector where single "1" bit is in position i+1
		$TEXT_{i+1}$ = corresponding $RESULT_{i+1}$ value from the TMOVS
	}	
TMOVS:	Compare RESULT from each loop with known answers.	
	The TEXT should be all zeros.	

As summarized in Table 52 the Inverse Permutation Known Answer Test for the TOFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

b. Initializes the 64-bit initialization vector IV_1 to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00.

c. Initializes the $TEXT_i$ (where i=1,...,64) to the $RESULT_i$ values obtained from the Variable TEXT Known Answer Test.

d. Forwards this information to the IUT using Input Type 5.

2. The IUT should perform the following for i=1 through 64:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using KEY1, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using KEY2, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using KEY3, resulting in output block O_i.

c. Calculate the $RESULT_i$ by exclusive-ORing the O_i with the $TEXT_i$.

d. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), IV_i, $TEXT_i$, and the $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Assign a new value to IV_{i+1} by setting it equal to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64.

f. Assign a new value to $TEXT_{i+1}$ by setting it equal to the corresponding output from the TMOVS.

NOTE -- This processing continues until all RESULT values from the Variable TEXT Known Answer Test have been used as input. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values. The RESULT values should be all zeros.

5.6.1.3 The Variable KEY Known Answer Test - TOFB Mode

Table 53 The Variable Key Known Answer Test - TOFB Mode

TMOVS:	Initialize	KEYs: $KEY1_1 = KEY2_1 = KEY3_1$ = 8001010101010101 (odd parity set) IV = 0000000000000000 TEXT = 0000000000000000
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV, TEXT
IUT:	FOR i = 1 to 64	
	{	
	IF (i mod 8 $\neq$ 0) {process all bits except parity bits}	
	{	
	Perform Triple DES:	I_i = IV I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i $RESULT_i = O_i \oplus TEXT$
		Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, TEXT, $RESULT_i$ $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$ = vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. Each parity bit may have the value "1" or "0" to make the KEY odd parity.
	}	
	}	
TMOVS:	Compare results of the 56 encryptions with known answers. Use Table A.2.	

As summarized in Table 53, the Variable Key Known Answer Test for the TOFB mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_1$, $KEY2_1$, and $KEY3_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit $KEY1_{1\ bin}$= $KEY2_{1\ bin}$= $KEY3_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

   NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

   c. Initializes the TEXT to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 2.

2. The IUT should perform the following for i = 1 to 56:

NOTE -- 56 is the number of significant bits in a TDES key.

   a. Assign the value of the initialization vector IV to the input block I_i.

   b. Process I_i through the three DEA stages resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

   c. Calculate the $RESULT_i$ by exclusive-ORing the O_i with the TEXT.

   d. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, TEXT, and the resulting $RESULT_i$ to the TMOVS as specified in Output Type 2.

   e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, $KEY3_{i+1}$ equal to the vector consisting of “0" in every significant bit position except for a single “1" bit in position i+1. The parity bits contain “1" or “0" to make odd parity.

NOTE -- This processing should continue until every significant basis vector has been represented by the KEY parameters. The output from the IUT should consist of 56 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.2.

5.6.1.4 The Permutation Operation Known Answer Test - TOFB Mode

Table 54 The Permutation Operation Known Answer Test - TOFB Mode

TMOVS:	Initialize	$KEY1_i$=$KEY2_i$=$KEY3_i$ (where i=1..32) = 32 KEY values in Table A.3 IV = 0000000000000000 TEXT = 0000000000000000
	Send	TEXT, IV, KEY_1, KEY_2,..., KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 32 {	
	Perform Triple DES:	I_i = IV I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i $RESULT_i$= $O_i \oplus$ TEXT
		Send i, KEY_i (representing $KEY1_i$,$KEY2_i$, and $KEY3_i$), IV, TEXT, $RESULT_i$ $KEY1_{i+1}$ = $KEY2_{i+1}$ = $KEY3_{i+1}$ = Corresponding KEY_{i+1} in Table A.3
	}	
TMOVS:	Compare results with known answers. Use Table A.3.	

As summarized in Table 54, the Permutation Operation Known Answer Test for the TOFB mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the $KEY1_i$, $KEY2_i$, and $KEY3_i$ variables with the 32 constant KEY values from Table A.3.

b. Initializes the 64-bit IV parameter to the constant hexadecimal value 0, i.e., IV_{hex} = 00 00 00 00 00 00 00 00.

c. Initializes the TEXT to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 8.

2. The IUT should perform the following for i=1 through 32:

a. Assign the value of the initialization vector IV to the input block I_i.

b. Process I_i through the three DEA stages resulting in a 64-bit output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

c. Calculate the $RESULT_i$ by exclusive-ORing the O_i with the TEXT.

d. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV, TEXT, and the $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

NOTE --The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.3.

5.6.1.5 The Substitution Table Known Answer Test - TOFB Mode

Table 55 The Substitution Table Known Answer Test - TOFB Mode

TMOVS:	Initialize	$KEY1_i$, $KEY2_i$, $KEY3_i$ (where i=1..19) = 19 KEY values in Table A.4 IV_i (where i=1..19) = 19 corresponding DATA values in Table A.4 TEXT = 0000000000000000
	Send	TEXT, KEY_1, IV_1, KEY_2, IV_2,...,KEY_{19}, IV_{19} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 19	
	{	
Perform Triple DES:	$I_i = IV_i$ I_i is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_i $RESULT_i = O_i \oplus TEXT$	
	Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV_i, TEXT, $RESULT_i$ $KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1} = KEY_{i+1}$ from TMOVS IV_{i+1} = corresponding $DATA_{i+1}$ from TMOVS }	
TMOVS:	Compare results from each loop with known answers. Use Table A.4.	

As summarized in Table 55, the Substitution Table Known Answer Test for the TOFB mode of operation is performed as follows:

1. The TMOVS:

a. Initializes the KEY-IV pairs with the 19 constant KEY-DATA values from Table A.4. The DATA values are assigned to the values of the initialization vectors IVs. The KEY value indicates the value of KEY1, KEY2 and KEY3, i.e., KEY1=KEY2=KEY3.

b. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

c. Forwards this information to the IUT using Input Type 11.

2. The IUT should perform the following for i=1 through 19:

a. Assign the value of the initialization vector IV_i to the input block I_i.

b. Process I_i through the three DEA stages, resulting in an output block O_i. This involves processing I_i through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_i.

c. Calculate the $RESULT_i$ by exclusive-ORing the O_i with the TEXT.

d. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV_i, TEXT, and the $RESULT_i$ to the TMOVS as specified in Output Type 2.

e. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} value supplied by the TMOVS.

f. Set IV_{i+1} equal to the corresponding $DATA_{i+1}$ value supplied by the TMOVS.

NOTE -- The above processing should continue until all 19 KEY-DATA pairs are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 2.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.4.

5.6.2 The Monte Carlo Test - TOFB Mode

The TOFB mode has one Monte Carlo test that is used regardless of process, i.e., the same Monte Carlo test is used for IUTs supporting the encryption and/or decryption processes.

Table 56 The Monte Carlo Test - TOFB Mode

TMOVS:	Initialize	$KEY1_0$,$KEY2_0$, $KEY3_0$, IV, $TEXT_0$
	Send	$KEY1_0$,$KEY2_0$, $KEY3_0$, IV, $TEXT_0$
IUT:	FOR i = 0 TO 399	
	{	
(Part of Triple DES processing):		If (i==0) I_0 = IV
		Record i, $KEY1_i$, $KEY2_i$, $KEY3_i$, $TEXT_0$
		INITTEXT = $TEXT_0$
		FOR j = 0 TO 9,999
		{
	Perform Triple DES:	I_j is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1 TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2 TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j $RESULT_j = O_j \oplus TEXT_j$
		$TEXT_{j+1} = I_j$
	(Part of Triple DES processing):	$I_{j+1} = O_j$
		}
		Record I_0, $RESULT_j$
		Send i,$KEY1_i$,$KEY2_i$, $KEY3_i$, I_0, $TEXT_0$, $RESULT_j$

$KEY1_{i+1} = KEY1_i \oplus RESULT_j$

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent)

$KEY2_{i+1}= KEY2_i \oplus RESULT_{j-1}$

ELSE

$KEY2_{i+1}= KEY2_i \oplus RESULT_j$

IF ($KEY1_i= KEY2_i= KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$)

$KEY3_{i+1}= KEY3_i \oplus RESULT_j$

ELSE

$KEY3_{i+1}= KEY3_i \oplus RESULT_{j-2}$

$TEXT_0 = INITTEXT \oplus I_j$

$I_0 = O_j$

}

TMOVS: Check IUT's output for correctness.

As summarized in Table 56, the Monte Carlo Test for the TOFB mode of operation is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_0$, $KEY2_0$, and $KEY3_0$, the initialization vector IV, and the $TEXT_0$ variables. All variables consist of 64 bits each.

   b. Forwards this information to the IUT using Input Type 21.

2. The IUT should perform the following for i = 0 through 399:

   a. If i=0 (if this is the first time through this loop), assign the value of the initialization vector IV to the input block I_0.

   b. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, and the $TEXT_0$.

c. Assign the value of $TEXT_0$ to INITTEXT. This will contain the initial value of the text from every j = 0 loop.

d. Perform the following for j = 0 through 9999:

1) Using the corresponding $KEY1_i$,$KEY2_i$, and $KEY3_i$ values, process I_j through the three DEA stages resulting in output block O_j. This involves processing I_j through the first DEA stage, denoted DEA_1, in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1. TEMP1 is fed directly into the second DEA stage, denoted DEA_2, in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2. TEMP2 is fed directly into the third DEA stage, denoted DEA_3, in the encrypt state using $KEY3_i$, resulting in output block O_j.

2) Calculate the $RESULT_j$ by exclusive-ORing the O_j with the $TEXT_j$.

3) Prepare for loop j+1 by doing the following:

a) Assign the $TEXT_{j+1}$ with the value of the I_j.

b) Assign I_{j+1} with the value of the O_j.

e. Record the $RESULT_j$ and the $I_{0.}$

f. Forward all recorded information for this loop, as specified in Output Type 6, to the TMOVS.

g. In preparation for the next output loop (note j = 9999):

1) Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the $RESULT_j$.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $RESULT_{j-1}$. If $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $RESULT_j$.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $RESULT_{j-2}$. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i$=$KEY2_i$=$KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $RESULT_j$.

2) Assign a new value to the $TEXT_0$. The $TEXT_0$ should be assigned the value of INITTEXT exclusive-ORed with I_j.

3) Assign a new value to I_0. I_0 should be assigned the value of the O_j.

NOTE -- the new TEXT and I should be denoted as $TEXT_0$ and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 6.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values.

5.7 The Output Feedback Interleaved (OFB-I) Mode

The IUTs which implement the Output Feedback Interleaved (OFB-I) mode are validated by successfully completing a set of Known Answer tests and a Monte Carlo test applicable to both IUTs supporting encryption and/or decryption. Encryption and decryption using the TOFB-I mode of operation involve initializing the three individual DEA stages and then simultaneously clocking them. This improves the throughput and minimizes the propagation delay. Each clock cycle involves the data being processed by each DEA_i stage and passing it onward to the output buffer or the next stage so that idle DEA_i stages are minimized. The pipelined configuration is intended for systems equipped with multiple DEA processors. The same set of Known Answer tests and Monte Carlo test can be applied to IUTs supporting both encryption and decryption because the same sequence of encrypt with KEY1, decrypt with KEY2 and encrypt with KEY3 is used for both encryption and decryption.

The processing for each Known Answer test and Monte Carlo test is broken down into clock cycles T1, T2, T3,.... Within each clock cycle, the processing occurring on each active DEA is discussed. For convenience, let KEY1 represent the key used on processor DEA_1, KEY2 represent the key used on processor DEA_2, and KEY3 represent the key used on processor DEA_3.

The process of validating an IUT which supports the TOFB-I mode of operation in the encryption and/or the decryption processes involves the successful completion of the following six tests:

1. The Variable Text Known Answer Test - TOFB-I mode

2. The Inverse Permutation Known Answer Test - TOFB-I mode

3. The Variable Key Known Answer Test - TOFB-I mode

4. The Permutation Operation Known Answer Test - TOFB-I mode

5. The Substitution Table Known Answer Test - TOFB-I mode

6. The Monte Carlo Test - TOFB-I mode

An explanation of the tests for the TOFB-I mode follows.

5.7.1 The Known Answer Tests - TOFB-I Mode

In the following description of the Known Answer tests, TEXT refers to plaintext, and RESULT refers to ciphertext if the IUT performs TOFB-I encryption. If the IUT supports TOFB-I decryption, TEXT refers to ciphertext, and RESULT refers to plaintext.

5.7.1.1 The Variable TEXT Known Answer Test - TOFB-I Mode

Table 57 The Variable TEXT Known Answer Test - TOFB-I Mode

TMOVS:	Initialize	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		$IV1_1$ = 8000000000000000
		$IV2_1$ = D555555555555555
		$IV3_1$ = 2AAAAAAAAAAAAAAA
		TEXT = 0
	Send	KEY (representing KEY1, KEY2, and KEY3), $IV1_1$, $IV2_1$, $IV3_1$, TEXT
IUT:	FOR i = 1 to 64	
	{	
		With feedback path disconnected:
Perform Triple DES:	T1:	$I1 = IV1_i$
		I1 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$
	T2:	$I2 = IV2_i$
		I2 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$
		$TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$
	T3:	$I3 = IV3_i$
		I3 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$
		$TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$
		Connect the feedback path:
		$TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $O1_i$
		$RESULT1_i = O1_i \oplus TEXT$

	T4:	$TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP3_2$
		$TEMP2_2$ is encrypted by DEA_3 using KEY3, resulting in $O2_i$
		$RESULT2_i = O2_i \oplus TEXT$
	T5:	$TEMP3_2$ is encrypted by DEA_3 using KEY3, resulting in $O3_i$
		$RESULT3_i = O3_i \oplus TEXT$
	Send i, KEY (representing KEY1, KEY2, and KEY3), I1, I2, I3, TEXT, $RESULT1_i$, $RESULT2_i$, $RESULT3_i$	
	$IV1_{i+1}$ = basis vector where single "1" bit is in position i+1	
	$IV2_{i+1} = IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555	
	$IV3_{i+1} = IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA	
	}	
TMOVS:	Compare RESULT1, RESULT2, and RESULT3 from each loop with known answers. See Table A.9.	

As summarized in Table 57, the Variable TEXT Known Answer Test for the TOFB-I mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}$=$KEY2_{hex}$=$KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 3 initialization vectors accordingly: $IV1_1$ is set to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, $IV2_1$ is computed by the following equation: $IV1_1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555. In hexadecimal, this equates to D5 55 55 55 55 55 55 55. And $IV3_1$ is computed by the equation $IV1_1 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA. In hexadecimal, this equates to 2A AA AA AA AA AA AA AA.

   c. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP1_1$.

2) At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

3) At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O1_i$.

e) Calculate the $RESULT1_i$ by exclusive-ORing $O1_i$ with $TEXT1_i$.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O2_i$.

b) Calculate $RESULT2_i$ by exclusive-ORing $O2_i$ with $TEXT2_i$.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O3_i$.

b) Calculate the $RESULT3_i$ by exclusive-ORing $O3_i$ with $TEXT3_i$.

b. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), $IV1_i$, $IV2_i$, $IV3_i$, TEXT, $RESULT1_i$, $RESULT2_i$, and $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Retain the RESULT1, RESULT2, and RESULT3 values for use with the Inverse Permutation Known Answer Test for the TOFB-I Mode (Section 5.7.1.2).

d. Assign a new value to the IV variables. $IV1_{i+1}$ is set to the value of a basis vector with a "1" bit in position i+1, where i+1=2,...,64. $IV2_{i+1}$ is set to the value of $IV1_{i+1}$ + R_1 mod 2^{64} where R_1=5555555555555555. And $IV3_{i+1}$ is set to $IV1_{i+1}$ + R_2 mod 2^{64} where R_2=AAAAAAAAAAAAAAAA.

NOTE -- This continues until every possible basis vector has been represented by the IV1, i.e., 64 times. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.9.

5.7.1.2 The Inverse Permutation Known Answer Test - TOFB-I Mode

Table 58 The Inverse Permutation Known Answer Test - TOFB-I Mode

TMOVS:	Initialize	KEY1 = KEY2 = KEY3 = 0101010101010101 (odd parity set)
		$IV1_1$ = 8000000000000000
		$IV2_1$ = D555555555555555
		$IV3_1$ = 2AAAAAAAAAAAAAAA
		$TEXTr_i$ (where r=1..3 and i=1..64) = 64 corresponding $RESULTr_i$ values from Variable TEXT Known Answer Test
	Send	KEY (representing KEY1, KEY2, and KEY3), $IV1_1$, $IV2_1$, $IV3_1$ $TEXT1_1$,...,$TEXT1_{64}$, $TEXT2_1$,...,$TEXT2_{64}$, $TEXT3_1$,...,$TEXT3_{64}$
IUT:	FOR i = 1 to 64	
	{	
	Perform Triple DES:	With the feedback path disconnected: T1: I1 = $IV1_i$ I1 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP1_1$ T2: I2 = $IV2_i$ I2 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP1_2$ T3: I3 = $IV3_i$ I3 is read into TDEA and is encrypted by DEA_1 using KEY1, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP2_2$ Connect the feedback path: $TEMP1_2$ is encrypted by DEA_3 using KEY3, resulting in $O1_i$

$RESULT1_i = O1_i \oplus TEXT1_i$

T4: $TEMP3_1$ is decrypted by DEA_2 using KEY2, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using KEY3, resulting in $O2_i$

$RESULT2_i = O2_i \oplus TEXT2_i$

T5: $TEMP3_2$ is encrypted by DEA_3 using KEY3, resulting in $O3_i$

$RESULT3_i = O3_i \oplus TEXT3_i$

Send i, KEY (representing KEY1, KEY2, and KEY3), I1, I2, I3, $TEXT1_i$, $TEXT2_i$, $TEXT3_i$, $RESULT1_i$, $RESULT2_i$, $RESULT3_i$

$IV1_{i+1}$ = basis vector where single "1" bit is in position i+1

$IV2_{i+1} = IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$IV3_{i+1} = IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

$TEXTr_{i+1}$ (where r = 1..3)= corresponding $RESULTr_{i+1}$ value from the TMOVS

}

TMOVS: Compare RESULT1, RESULT2, and RESULT3 from each loop with known answers.

They should be all zeros.

As summarized in Table 58 the Inverse Permutation Known Answer Test for the TOFB-I mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 to the constant hexadecimal value 0 with odd parity set, i.e., $KEY1_{hex}=KEY2_{hex}=KEY3_{hex}$=01 01 01 01 01 01 01 01.

   b. Initializes the 3 initialization vectors accordingly: $IV1_1$ is set to the basis vector containing a"1" in the first bit position and "0" in the following 63 positions, i.e., $IV_{1\ bin}$ = 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000. The equivalent of this value in hexadecimal notation is 80 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, $IV2_1$ is computed by the following equation: $IV1_1 + R_1 \bmod 2^{64}$ where R_1=5555555555555555. In hexadecimal, this equates to D5 55 55 55 55 55 55 55. And $IV3_1$ is computed by the equation $IV1_1 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA. In hexadecimal, this equates to 2A AA AA AA AA AA AA AA.

c. Initializes the $TEXTr_i$ (where r=1,…,3 and i=1,…,64) to the $RESULTr_i$ obtained from the Variable TEXT Known Answer Test.

d. Forwards this information to the IUT using Input Type 15.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP1_1$.

2) At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP1_2$.

3) At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using KEY1, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O1_i$.

e) Calculate the $RESULT1_i$ by exclusive-ORing $O1_i$ with $TEXT1_i$.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O2_i$.

b) Calculate the $RESULT2_i$ by exclusive-ORing $O2_i$ with $TEXT2_i$.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using KEY2, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using KEY3, resulting in output block $O3_i$.

b) Calculate the $RESULT3_i$ by exclusive-ORing $O3_i$ with $TEXT3_i$.

b. Forward the current values of the loop number i, KEY (representing KEY1, KEY2, and KEY3), $IV1_i$, $IV2_i$, $IV3_i$, $TEXT1_i$, $TEXT2_i$, $TEXT3_i$, $RESULT1_i$, $RESULT2_i$, and $RESULT3_i$, to the TMOVS as specified in Output Type 3.

c. Assign a new value to the IV variables. $IV1_{i+1}$ is set to the value of a basis vector with a "1" bit in position i+1, where i+1=2,…,64. $IV2_{i+1}$ is set to the value of $IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555. And $IV3_{i+1}$ is set to $IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA.

d. Assign a new value to the $TEXT1_{i+1}$, $TEXT2_{i+1}$, and $TEXT3_{i+1}$ by setting it equal to the corresponding output from the TMOVS.

NOTE -- This continues until every RESULT1, RESULT2, and RESULT3 value from the Variable TEXT Known Answer Test has been used as input. The output from the IUT should consist of 64 output strings. Each output string should consist of information included in Output Type 3.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values. The RESULT1, RESULT2, and RESULT3 values should be all zeros.

5.7.1.3 The Variable KEY Known Answer Test - TOFB-I Mode

Table 59 The Variable KEY Known Answer Test - TOFB-I Mode

TMOVS:	Initialize	$KEY1_1$ = $KEY2_1$ = $KEY3_1$ = 8001010101010101 (odd parity set) IV1 = 0000000000000000 IV2 = 5555555555555555 IV3 = AAAAAAAAAAAAAAAA TEXT = 0
	Send	KEY_1 (representing $KEY1_1$, $KEY2_1$, and $KEY3_1$), IV1, IV2, IV3, TEXT

IUT:	FOR i = 1 to 64
	
	IF (i mod 8 ≠ 0){process all bits except parity bits}
	
	{	

Perform Triple DES:	With the feedback path disconnected: T1: I1 = IV1 I1 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$ T2: I2 = IV2 I2 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$ T3: I3 = IV3 I3 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$ Connect the feedback path:	
	$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O1_i$	

RESULT1$_i$ = O1$_i$ ⊕ TEXT

T4: TEMP3$_1$ is decrypted by DEA$_2$ using KEY2$_i$, resulting in TEMP3$_2$

TEMP2$_2$ is encrypted by DEA$_3$ using KEY3$_i$, resulting in O2$_i$

RESULT2$_i$ = O2$_i$ ⊕ TEXT

T5: TEMP3$_2$ is encrypted by DEA$_3$ using KEY3$_i$, resulting in O3$_i$

RESULT3$_i$ = O3$_i$ ⊕ TEXT

Send i, KEY$_i$ (representing KEY1$_i$, KEY2$_i$, and KEY3$_i$), I1, I2, I3, TEXT, RESULT1$_i$, RESULT2$_i$, RESULT3$_i$

KEY1$_{i+1}$ = KEY2 $_{i+1}$ = KEY3 $_{i+1}$= vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. Each parity bit may have the value "1" or "0" to make the KEY odd parity.

}

TMOVS: Compare results of the 56 encryptions with known answers.

See Table A.11.

As summarized in Table 59, the Variable KEY Known Answer Test for the TOFB-I Mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1$_1$, KEY2$_1$, and KEY3$_1$ to contain "0" in every significant bit except for a "1" in the first position, i.e., the 64-bit KEY1$_{1\ bin}$= KEY2$_{1\ bin}$= KEY3$_{1\ bin}$= 10000000 00000001 00000001 00000001 00000001 00000001 00000001 00000001. The equivalent of this value in hexadecimal notation is 80 01 01 01 01 01 01 01.

   NOTE -- the parity bits are set to "0" or "1" to get odd parity.

   b. Initializes the 3 initialization vectors accordingly: IV1 is set to zero, i.e., IV$_{1\ hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed as IV1 + R$_1$ mod 2^{64} where R$_1$ = 5555555555555555 and IV3 is computed as IV1 + R$_2$ mod 2^{64} where R$_2$ = AAAAAAAAAAAAAAAA. Since IV1$_{hex}$ = 0000000000000000, this equates to IV2$_{hex}$ = 5555555555555555 and IV3$_{hex}$ = AAAAAAAAAAAAAAAA.

c. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

d. Forwards this information to the IUT using Input Type 13.

2. The IUT should perform the following for i=1 through 64:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector IV1 to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

2) At time T2:

a) Assign the value of the initialization vector IV2 to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

3) At time T3:

a) Assign the value of the initialization vector IV3 to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the $RESULT1_i$ by exclusive-ORing $O1_i$ with TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate the $RESULT2_i$ by exclusive-ORing $O2_i$ with TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the $RESULT3_i$ by exclusive-ORing $O3_i$ with TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, TEXT, $RESULT1_i$, $RESULT2_i$, and $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the vector consisting of "0" in every significant bit position except for a single "1" bit in position i+1. The parity bits contain "1" or "0" to make odd parity.

NOTE -- This processing should continue until every significant basis vector has been represented by the KEY parameters. The output from the IUT should consist of 56 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.11.

5.7.1.4 The Permutation Operation Known Answer Test - TOFB-I Mode

Table 60 The Permutation Operation Known Answer Test - TOFB-I Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..32) = 32 KEY values in Table A.12 IV1= 0000000000000000 IV2 =5555555555555555 IV3 =AAAAAAAAAAAAAAAA TEXT = 0
	Send	IV1, IV2, IV3, TEXT, KEY_1, KEY_2,...,KEY_{32} (Since all three keys are the same, these key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 32 {	
	Perform Triple DES:	With the feedback path disconnected: T1: $I1 = IV1$ I1 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP1_1$ T2: $I2 = IV2$ I2 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA using $KEY2_i$, resulting in $TEMP1_2$ T3: $I3 = IV3$ I3 is read into TDEA and is encrypted by DEA using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA using $KEY2_i$, resulting in $TEMP2_2$ Connect the feedback path:

		TEMP1$_2$ is encrypted by DEA using KEY3$_i$, resulting in O1$_i$ RESULT1$_i$ = O1$_i$ $\oplus$ TEXT
	T4:	TEMP3$_1$ is decrypted by DEA$_2$ using KEY2$_i$, resulting in TEMP3$_2$ TEMP2$_2$ is encrypted by DEA$_3$ using KEY3$_i$, resulting in O2$_i$ RESULT2$_i$ = O2$_i$ $\oplus$ TEXT
	T5:	TEMP3$_2$ is encrypted by DEA$_3$ using KEY3$_i$, resulting in O3$_i$ RESULT3$_i$ = O3$_i$ $\oplus$ TEXT
	Send i, KEY$_i$ (representing KEY1$_i$, KEY2$_i$, and KEY3$_i$), I1, I2, I3, TEXT, RESULT1$_i$, RESULT2$_i$, RESULT3$_i$ KEY1$_{i+1}$ = KEY2$_{i+1}$ = KEY3$_{i+1}$= corresponding KEY$_{i+1}$ from TMOVS.	
TMOVS:	Compare results from each loop with known answers. See Table A.12.	

As summarized in Table 60, the Permutation Operation Known Answer Test for the TOFB-I Mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters KEY1, KEY2, and KEY3 with the 32 constant KEY values from Table A.12.

   b. Initializes the 3 initialization vectors accordingly: IV1 is assigned the value 0, i.e., IV$_{hex}$ = 00 00 00 00 00 00 00 00. Based on specifications in ANSI X9.52-1998, IV2 is computed by the following equation: IV1 + R$_1$ mod 2^{64} where R$_1$=5555555555555555. In hexadecimal, this equates to 55 55 55 55 55 55 55 55. And IV3 is computed by the equation IV1 + R$_2$ mod 2^{64} where R$_2$=AAAAAAAAAAAAAAAA. In hexadecimal, this equates to AA AA AA AA AA AA AA AA.

   c. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., TEXT$_{hex}$ = 00 00 00 00 00 00 00 00.

   d. Forwards this information to the IUT using Input Type 18.

2. The IUT should perform the following for i=1 through 32:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector IV1 to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector IV2 to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector IV3 to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the $RESULT1_i$ by exclusive-ORing $O1_i$ with TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate the $RESULT2_i$ by exclusive-ORing $O2_i$ with TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the $RESULT3_i$ by exclusive-ORing $O3_i$ with TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), IV1, IV2, IV3, TEXT, $RESULT1_i$, $RESULT2_i$, and $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

NOTE -- The above processing should continue until all 32 KEY values are processed. The output from the IUT for this test should consist of 32 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found Table A.12.

5.7.1.5 The Substitution Table Known Answer Test - TOFB-I Mode

Table 61 The Substitution Table Known Answer Test - TOFB-I Mode

TMOVS:	Initialize	$KEY1_i = KEY2_i = KEY3_i$ (where i=1..19) = 19 KEY values in Table A.10 $IV1_i$ (where i=1..19) = 19 corresponding TEXT values in Table A.10 $IV2_i = IV1_i + R_1 \bmod 2^{64}$ where R_1=5555555555555555 $IV3_i = IV1_i + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA TEXT = 0
	Send	$IV1_i$, $IV2_i$, $IV3_i$, TEXT, KEY_1, KEY_2,...,KEY_{19} (These key values represent the values of KEY1, KEY2, and KEY3.)
IUT:	FOR i = 1 to 19 {	
	Perform Triple DES:	With the feedback path disconnected: T1: $I1 = IV1_i$ I1 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP1_1$ T2: $I2 = IV2_i$ I2 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP2_1$ $TEMP1_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP1_2$ T3: $I3 = IV3_i$ I3 is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in $TEMP3_1$ $TEMP2_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP2_2$ Connect the feedback path:
		$TEMP1_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O1_i$

$RESULT1_i = O1_i \oplus TEXT$

T4: $TEMP3_1$ is decrypted by DEA_2 using $KEY2_i$, resulting in $TEMP3_2$

$TEMP2_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O2_i$

$RESULT2_i = O2_i \oplus TEXT$

T5: $TEMP3_2$ is encrypted by DEA_3 using $KEY3_i$, resulting in $O3_i$

$RESULT3_i = O3_i \oplus TEXT$

Send i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), $IV1_i$, $IV2_i$, $IV3_i$, TEXT, $RESULT1_i$, $RESULT2_i$, $RESULT3_i$

$KEY1_{i+1} = KEY2_{i+1} = KEY3_{i+1}$= Corresponding KEY_{i+1} from TMOVS

$IV1_{i+1}$= corresponding $DATA_{i+1}$ from TMOVS

$IV2_{i+1} = IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$IV3_{i+1} = IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

}

TMOVS: Compare results from each loop with known answers.

See Table A.10.

As summarized in Table 61, the Substitution Table Known Answer Test for the TCFB-P Mode is performed as follows:

1. The TMOVS:

   a. Initializes the KEY-IV1 pairs with the 19 constant KEY-DATA values from Table A.10. The DATA values are assigned to the values of the initialization vectors $IV1_i$. The KEY value indicates the values of $KEY1_i$, $KEY2_i$, and $KEY3_i$, i.e., $KEY1_i$=$KEY2_i$=$KEY3_i$. Based on specifications in ANSI X9.52-1998, $IV2_i$ is computed as $IV1_i + R_1 \bmod 2^{64}$ where R_1 = 5555555555555555 and $IV3_i$ is computed as $IV1_i + R_2 \bmod 2^{64}$ where R_2 = AAAAAAAAAAAAAAAA.

   b. Initializes the TEXT parameter to the constant hexadecimal value 0, i.e., $TEXT_{hex}$ = 00 00 00 00 00 00 00 00.

   c. Forwards this information to the IUT using Input Type 25.

2. The IUT should perform the following for i=1 through 19:

a. With the feedback path disconnected:

1) At time T1:

a) Assign the value of the initialization vector $IV1_i$ to the input block I1.

b) Process I1 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP1_1$.

At time T2:

a) Assign the value of the initialization vector $IV2_i$ to the input block I2.

b) Process I2 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP2_1$.

c) $TEMP1_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP1_2$.

At time T3:

a) Assign the value of the initialization vector $IV3_i$ to the input block I3.

b) Process I3 through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value $TEMP3_1$.

c) $TEMP2_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP2_2$.

Connect the feedback path:

d) $TEMP1_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O1_i$.

e) Calculate the $RESULT1_i$ by exclusive-ORing $O1_i$ with TEXT.

At time T4:

a) $TEMP2_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O2_i$.

b) Calculate $RESULT2_i$ by exclusive-ORing $O2_i$ with TEXT.

c) $TEMP3_1$ is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value $TEMP3_2$.

At time T5:

a) $TEMP3_2$ is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block $O3_i$.

b) Calculate the $RESULT3_i$ by exclusive-ORing $O3_i$ with TEXT.

b. Forward the current values of the loop number i, KEY_i (representing $KEY1_i$, $KEY2_i$, and $KEY3_i$), $IV1_i$, $IV2_i$, $IV3_i$, TEXT, $RESULT1_i$, $RESULT2_i$, and $RESULT3_i$, to the TMOVS as specified in Output Type 7.

c. Set $KEY1_{i+1}$, $KEY2_{i+1}$, and $KEY3_{i+1}$ equal to the corresponding KEY_{i+1} supplied by the TMOVS.

d. Set $IV1_{i+1}$ equal to the corresponding $DATA_{i+1}$ supplied by the TMOVS. Based on specifications in ANSI X9.52-1998, $IV2_{i+1}$ is set to $IV1_{i+1} + R_1 \bmod 2^{64}$ where R_1=5555555555555555 and $IV3_{i+1}$ is set to $IV1_{i+1} + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA.

NOTE -- The above processing should continue until all 19 KEY-DATA are processed. The output from the IUT for this test should consist of 19 output strings. Each output string should consist of information included in Output Type 7.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to the known values found in Table A.10.

5.7.2 The Monte Carlo Tests - TOFB-I Mode

The TOFB-I mode of operation has one Monte Carlo test that is used regardless of the process, i.e., the same Monte Carlo test is used for IUTs supporting the encryption and/or decryption process. In the following description of the Monte Carlo test, TEXT refers to plaintext, and RESULT refers to ciphertext if the IUT performs TOFB-I encryption. If the IUT supports TOFB-I decryption, TEXT refers to ciphertext, and RESULT refers to plaintext**.**

Table 62 The Monte Carlo Test - TOFB-I Mode

TMOVS:	Initialize	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, $TEXT_0$
	Send	$KEY1_0$,$KEY2_0$,$KEY3_0$, IV1, IV2, IV3, $TEXT_0$
IUT:	FOR i = 0 TO 399	
	{	
	FOR j = 0 TO 9,999	
	{	

Perform Triple DES:	IF (j == 0, 1, or 2)
	Ij = IV(j+1)
	ELSE
	Ij=O_{j-3}
	Ij is read into TDEA and is encrypted by DEA_1 using $KEY1_i$, resulting in TEMP1
	TEMP1 is decrypted by DEA_2 using $KEY2_i$, resulting in TEMP2
	TEMP2 is encrypted by DEA_3 using $KEY3_i$, resulting in O_j
	$RESULT_j = O_j \oplus TEXT_j$	

	$TEXT_{j+1}$ = Ij	
	}	
	Record I0, $RESULT_j$	
	Send i, $KEY1_i$, $KEY2_i$, $KEY3_i$, I0, I1, I2, $TEXT_0$, $RESULT_j$	
	$KEY1_{i+1} = KEY1_i \oplus RESULT_j$	

IF ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$) or ($KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent)

$KEY2_{i+1} = KEY2_i \oplus RESULT_{j-1}$

ELSE

$KEY2_{i+1} = KEY2_i \oplus RESULT_j$

IF ($KEY1_i = KEY2_i = KEY3_i$) or ($KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$)

$KEY3_{i+1} = KEY3_i \oplus RESULT_j$

ELSE

$KEY3_{i+1} = KEY3_i \oplus RESULT_{j-2}$

$TEXT_0 = TEXT_0 \oplus Ij$

$I0 = O_j$

$I1 = I0 + R_1 \bmod 2^{64}$ where R_1=5555555555555555

$I2 = I0 + R_2 \bmod 2^{64}$ where R_2=AAAAAAAAAAAAAAAA

}

TMOVS: Check the IUT's output for correctness.

As summarized in Table 62, the Monte Carlo Test for the TOFB-I is performed as follows:

1. The TMOVS:

   a. Initializes the KEY parameters $KEY1_0$, $KEY2_0$, and $KEY3_0$, the initialization vectors IV1, IV2, and IV3, and the $TEXT_0$. The TEXT, IVs, and KEYs consist of 64 bits each. IV2 is assigned the value of IV1+R_1 mod 2^{64} where R_1 = 5555555555555555. IV3 is assigned the value of IV1+R_2 mod 2^{64} where R_2 = AAAAAAAAAAAAAAAA.

   b. Forwards this information to the IUT using Input Type 24.

2. The IUT should perform the following for i = 0 through 399:

   a. Record the current values of the output loop number i, $KEY1_i$, $KEY2_i$, $KEY3_i$, IV1, IV2, IV3, and $TEXT_0$.

b. Perform the following for j = 0 through 9,999:

1) If j = 0, 1, or 2, assign the value of the initialization vector IV(j+1) to the input block Ij.

2) If j > 2, assign Ij with the value of the output block O(j-3).

3) Process Ij through the DEA stage DEA_1 in the encrypt state using $KEY1_i$, resulting in intermediate value TEMP1.

4) TEMP1 is fed into the DEA stage DEA_2 in the decrypt state using $KEY2_i$, resulting in intermediate value TEMP2.

5) TEMP2 is fed into the DEA stage DEA_3 in the encrypt state using $KEY3_i$, resulting in output block O_j.

6) Calculate the $RESULT_j$ by exclusive-ORing the O_j with the $TEXT_j$.

7) Prepare for loop j+1 by assigning the $TEXT_{j+1}$ with the value of the Ij.

c. Record the current values of the input block I0 and $RESULT_j$.

d. Forward all recorded values for this loop, as specified in Output Type 8, to the MOVS.

e. In preparation for the next output loop:

1) Assign new values to the KEY parameters, KEY1, KEY2, and KEY3 in preparation for the next outer loop.

The new $KEY1_{i+1}$ should be calculated by exclusive-ORing the current $KEY1_i$ with the $RESULT_j$.

The new $KEY2_{i+1}$ calculation is based on the values of the keys. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $RESULT_{j-1}$. If $KEY1_i=KEY2_i=KEY3_i$, the new $KEY2_{i+1}$ should be calculated by exclusive-ORing the current $KEY2_i$ with the $RESULT_j$.

The new $KEY3_{i+1}$ calculation is also based on the values of the keys. If $KEY1_i$, $KEY2_i$, and $KEY3_i$ are independent, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $RESULT_{j-2}$. If $KEY1_i$ and $KEY2_i$ are independent and $KEY3_i = KEY1_i$, or if $KEY1_i=KEY2_i=KEY3_i$, the new $KEY3_{i+1}$ should be calculated by exclusive-ORing the current $KEY3_i$ with the $RESULT_j$.

2) Assign a new value to the $TEXT_0$. The $TEXT_0$ should be assigned the value of the current Ij exclusive-ORed with $TEXT_0$.

3) Assign a new value to the I parameters. I0 should be assigned the value of O_j. I1 should be assigned the value of I0+R_1 mod 2^{64} where R_1 = 5555555555555555. I2 should be assigned the value of I0+R_2 mod 2^{64} where R_2 = AAAAAAAAAAAAAAAA.

NOTE -- the new TEXT and I should be denoted as $TEXT_0$ and I_0 because these values are used for the first pass through the inner loop when j=0.

NOTE -- The output from the IUT for this test should consist of 400 output strings. Each output string should consist of information included in Output Type 8.

3. The TMOVS checks the IUT's output for correctness by comparing the received results to known values.

6. DESIGN OF THE TRIPLE DES MODES OF OPERATION VALIDATION SYSTEM (TMOVS)

6.1 Design Philosophy

NIST validation programs are conformance tests rather than measures of product security. NIST validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation by NIST should not be interpreted as an evaluation or endorsement of overall product security.

An IUT is considered validated for a test option when it passes the appropriate set of TMOVS tests. TMOVS testing is via statistical sampling, so validation of an option does not guarantee 100 % conformance with the option in the standards.

The intent of the validation process is to provide a rigorous conformance process that can be performed at modest cost. NIST does not try to prevent a dishonest vendor from purchasing a validated implementation and using this implementation as the vendor's IUT. Customers who wish to protect themselves against a dishonest vendor could require that the vendor revalidate the IUT in the customer's presence.

6.2 Operation of the TMOVS

TMOVS testing is done through the NIST Cryptographic Module Validation (CMV) Program. The CMV Program uses laboratories accredited by the NIST National Voluntary Laboratory Accreditation Program (NVLAP) to perform conformance tests to cryptographic-related FIPS. A vendor contracts with a Cryptographic Module Testing (CMT) Laboratory accredited by NVLAP. The CMT laboratory either conducts the TMOVS tests on the IUT or supplies initial values to the vendor to conduct the tests. If the vendor conducts the tests, the vendor sends the results to the CMT where they are verified. In both situations, the CMT laboratory submits the results to NIST for validation. If the IUT has successfully completed the tests, NIST issues a validation certificate for the IUT to the vendor. A list of CMT laboratories is available at http://csrc.nist.gov/cryptval.

.

Appendix A
Tables of Values for the Known Answer Tests

Tables A.1 through A.4 were used with single DES. They will work with the triple DES Validation Modes that are backwards compatible with their single counterparts. These include TECB, TCBC, TCFB, and TOFB.

The other tables include the values obtained for ciphertexts C2 and C3. These values are a result of having three initialization vectors. These tables can be used with the interleaved and pipelined configurations of the Triple DES modes of operation.

Table A.1 Resulting Ciphertext from the Variable Plaintext Known Answer Test

for the TECB, TCBC, TCFB, and TOFB Modes of Operation

(NOTE -- KEY1=KEY2=KEY3 = 01 01 01 01 01 01 01 01 (odd parity set))

ROUND	PLAINTEXT or IV (depending on mode)	CIPHERTEXT
0	80 00 00 00 00 00 00 00	95 F8 A5 E5 DD 31 D9 00
1	40 00 00 00 00 00 00 00	DD 7F 12 1C A5 01 56 19
2	20 00 00 00 00 00 00 00	2E 86 53 10 4F 38 34 EA
3	10 00 00 00 00 00 00 00	4B D3 88 FF 6C D8 1D 4F
4	08 00 00 00 00 00 00 00	20 B9 E7 67 B2 FB 14 56
5	04 00 00 00 00 00 00 00	55 57 93 80 D7 71 38 EF
6	02 00 00 00 00 00 00 00	6C C5 DE FA AF 04 51 2F
7	01 00 00 00 00 00 00 00	0D 9F 27 9B A5 D8 72 60

ROUND	PLAINTEXT or IV (depending on mode)	CIPHERTEXT
8	00 80 00 00 00 00 00 00	D9 03 1B 02 71 BD 5A 0A
9	00 40 00 00 00 00 00 00	42 42 50 B3 7C 3D D9 51
10	00 20 00 00 00 00 00 00	B8 06 1B 7E CD 9A 21 E5
11	00 10 00 00 00 00 00 00	F1 5D 0F 28 6B 65 BD 28
12	00 08 00 00 00 00 00 00	AD D0 CC 8D 6E 5D EB A1
13	00 04 00 00 00 00 00 00	E6 D5 F8 27 52 AD 63 D1
14	00 02 00 00 00 00 00 00	EC BF E3 BD 3F 59 1A 5E
15	00 01 00 00 00 00 00 00	F3 56 83 43 79 D1 65 CD
16	00 00 80 00 00 00 00 00	2B 9F 98 2F 20 03 7F A9
17	00 00 40 00 00 00 00 00	88 9D E0 68 A1 6F 0B E6
18	00 00 20 00 00 00 00 00	E1 9E 27 5D 84 6A 12 98
19	00 00 10 00 00 00 00 00	32 9A 8E D5 23 D7 1A EC
20	00 00 08 00 00 00 00 00	E7 FC E2 25 57 D2 3C 97
21	00 00 04 00 00 00 00 00	12 A9 F5 81 7F F2 D6 5D
22	00 00 02 00 00 00 00 00	A4 84 C3 AD 38 DC 9C 19
23	00 00 01 00 00 00 00 00	FB E0 0A 8A 1E F8 AD 72

ROUND	PLAINTEXT or IV (depending on mode)	CIPHERTEXT
24	00 00 00 80 00 00 00 00	75 0D 07 94 07 52 13 63
25	00 00 00 40 00 00 00 00	64 FE ED 9C 72 4C 2F AF
26	00 00 00 20 00 00 00 00	F0 2B 26 3B 32 8E 2B 60
27	00 00 00 10 00 00 00 00	9D 64 55 5A 9A 10 B8 52
28	00 00 00 08 00 00 00 00	D1 06 FF 0B ED 52 55 D7
29	00 00 00 04 00 00 00 00	E1 65 2C 6B 13 8C 64 A5
30	00 00 00 02 00 00 00 00	E4 28 58 11 86 EC 8F 46
31	00 00 00 01 00 00 00 00	AE B5 F5 ED E2 2D 1A 36
32	00 00 00 00 80 00 00 00	E9 43 D7 56 8A EC 0C 5C
33	00 00 00 00 40 00 00 00	DF 98 C8 27 6F 54 B0 4B
34	00 00 00 00 20 00 00 00	B1 60 E4 68 0F 6C 69 6F
35	00 00 00 00 10 00 00 00	FA 07 52 B0 7D 9C 4A B8
36	00 00 00 00 08 00 00 00	CA 3A 2B 03 6D BC 85 02
37	00 00 00 00 04 00 00 00	5E 09 05 51 7B B5 9B CF
38	00 00 00 00 02 00 00 00	81 4E EB 3B 91 D9 07 26
39	00 00 00 00 01 00 00 00	4D 49 DB 15 32 91 9C 9F

ROUND	PLAINTEXT or IV (depending on mode)	CIPHERTEXT
40	00 00 00 00 00 80 00 00	25 EB 5F C3 F8 CF 06 21
41	00 00 00 00 00 40 00 00	AB 6A 20 C0 62 0D 1C 6F
42	00 00 00 00 00 20 00 00	79 E9 0D BC 98 F9 2C CA
43	00 00 00 00 00 10 00 00	86 6E CE DD 80 72 BB 0E
44	00 00 00 00 00 08 00 00	8B 54 53 6F 2F 3E 64 A8
45	00 00 00 00 00 04 00 00	EA 51 D3 97 55 95 B8 6B
46	00 00 00 00 00 02 00 00	CA FF C6 AC 45 42 DE 31
47	00 00 00 00 00 01 00 00	8D D4 5A 2D DF 90 79 6C
48	00 00 00 00 00 00 80 00	10 29 D5 5E 88 0E C2 D0
49	00 00 00 00 00 00 40 00	5D 86 CB 23 63 9D BE A9
50	00 00 00 00 00 00 20 00	1D 1C A8 53 AE 7C 0C 5F
51	00 00 00 00 00 00 10 00	CE 33 23 29 24 8F 32 28
52	00 00 00 00 00 00 08 00	84 05 D1 AB E2 4F B9 42
53	00 00 00 00 00 00 04 00	E6 43 D7 80 90 CA 42 07
54	00 00 00 00 00 00 02 00	48 22 1B 99 37 74 8A 23
55	00 00 00 00 00 00 01 00	DD 7C 0B BD 61 FA FD 54

ROUND	PLAINTEXT or IV (depending on mode)	CIPHERTEXT
56	00 00 00 00 00 00 00 80	2F BC 29 1A 57 0D B5 C4
57	00 00 00 00 00 00 00 40	E0 7C 30 D7 E4 E2 6E 12
58	00 00 00 00 00 00 00 20	09 53 E2 25 8E 8E 90 A1
59	00 00 00 00 00 00 00 10	5B 71 1B C4 CE EB F2 EE
60	00 00 00 00 00 00 00 08	CC 08 3F 1E 6D 9E 85 F6
61	00 00 00 00 00 00 00 04	D2 FD 88 67 D5 0D 2D FE
62	00 00 00 00 00 00 00 02	06 E7 EA 22 CE 92 70 8F
63	00 00 00 00 00 00 00 01	16 6B 40 B4 4A BA 4B D6

Table A.2 Resulting Ciphertext from the Variable Key Known Answer Test

for the TECB, TCBC, TCFB, and TOFB Modes of Operation

(NOTE -- Plaintext/text = 00 00 00 00 00 00 00 00 and, where applicable, IV = 00 00 00 00 00 00 00 00)

ROUND	KEY	CIPHERTEXT
0	80 01 01 01 01 01 01 01	95 A8 D7 28 13 DA A9 4D
1	40 01 01 01 01 01 01 01	0E EC 14 87 DD 8C 26 D5
2	20 01 01 01 01 01 01 01	7A D1 6F FB 79 C4 59 26
3	10 01 01 01 01 01 01 01	D3 74 62 94 CA 6A 6C F3
4	08 01 01 01 01 01 01 01	80 9F 5F 87 3C 1F D7 61
5	04 01 01 01 01 01 01 01	C0 2F AF FE C9 89 D1 FC
6	02 01 01 01 01 01 01 01	46 15 AA 1D 33 E7 2F 10
7	01 80 01 01 01 01 01 01	20 55 12 33 50 C0 08 58
8	01 40 01 01 01 01 01 01	DF 3B 99 D6 57 73 97 C8
9	01 20 01 01 01 01 01 01	31 FE 17 36 9B 52 88 C9
10	01 10 01 01 01 01 01 01	DF DD 3C C6 4D AE 16 42
11	01 08 01 01 01 01 01 01	17 8C 83 CE 2B 39 9D 94
12	01 04 01 01 01 01 01 01	50 F6 36 32 4A 9B 7F 80

ROUND	KEY	CIPHERTEXT
13	01 02 01 01 01 01 01 01	A8 46 8E E3 BC 18 F0 6D
14	01 01 80 01 01 01 01 01	A2 DC 9E 92 FD 3C DE 92
15	01 01 40 01 01 01 01 01	CA C0 9F 79 7D 03 12 87
16	01 01 20 01 01 01 01 01	90 BA 68 0B 22 AE B5 25
17	01 01 10 01 01 01 01 01	CE 7A 24 F3 50 E2 80 B6
18	01 01 08 01 01 01 01 01	88 2B FF 0A A0 1A 0B 87
19	01 01 04 01 01 01 01 01	25 61 02 88 92 45 11 C2
20	01 01 02 01 01 01 01 01	C7 15 16 C2 9C 75 D1 70
21	01 01 01 80 01 01 01 01	51 99 C2 9A 52 C9 F0 59
22	01 01 01 40 01 01 01 01	C2 2F 0A 29 4A 71 F2 9F
23	01 01 01 20 01 01 01 01	EE 37 14 83 71 4C 02 EA
24	01 01 01 10 01 01 01 01	A8 1F BD 44 8F 9E 52 2F
25	01 01 01 08 01 01 01 01	4F 64 4C 92 E1 92 DF ED
26	01 01 01 04 01 01 01 01	1A FA 9A 66 A6 DF 92 AE
27	01 01 01 02 01 01 01 01	B3 C1 CC 71 5C B8 79 D8
28	01 01 01 01 80 01 01 01	19 D0 32 E6 4A B0 BD 8B

ROUND	KEY	CIPHERTEXT
29	01 01 01 01 40 01 01 01	3C FA A7 A7 DC 87 20 DC
30	01 01 01 01 20 01 01 01	B7 26 5F 7F 44 7A C6 F3
31	01 01 01 01 10 01 01 01	9D B7 3B 3C 0D 16 3F 54
32	01 01 01 01 08 01 01 01	81 81 B6 5B AB F4 A9 75
33	01 01 01 01 04 01 01 01	93 C9 B6 40 42 EA A2 40
34	01 01 01 01 02 01 01 01	55 70 53 08 29 70 55 92
35	01 01 01 01 01 80 01 01	86 38 80 9E 87 87 87 A0
36	01 01 01 01 01 40 01 01	41 B9 A7 9A F7 9A C2 08
37	01 01 01 01 01 20 01 01	7A 9B E4 2F 20 09 A8 92
38	01 01 01 01 01 10 01 01	29 03 8D 56 BA 6D 27 45
39	01 01 01 01 01 08 01 01	54 95 C6 AB F1 E5 DF 51
40	01 01 01 01 01 04 01 01	AE 13 DB D5 61 48 89 33
41	01 01 01 01 01 02 01 01	02 4D 1F FA 89 04 E3 89
42	01 01 01 01 01 01 80 01	D1 39 97 12 F9 9B F0 2E
43	01 01 01 01 01 01 40 01	14 C1 D7 C1 CF FE C7 9E
44	01 01 01 01 01 01 20 01	1D E5 27 9D AE 3B ED 6F

ROUND	KEY	CIPHERTEXT
45	01 01 01 01 01 01 10 01	E9 41 A3 3F 85 50 13 03
46	01 01 01 01 01 01 08 01	DA 99 DB BC 9A 03 F3 79
47	01 01 01 01 01 01 04 01	B7 FC 92 F9 1D 8E 92 E9
48	01 01 01 01 01 01 02 01	AE 8E 5C AA 3C A0 4E 85
49	01 01 01 01 01 01 01 80	9C C6 2D F4 3B 6E ED 74
50	01 01 01 01 01 01 01 40	D8 63 DB B5 C5 9A 91 A0
51	01 01 01 01 01 01 01 20	A1 AB 21 90 54 5B 91 D7
52	01 01 01 01 01 01 01 10	08 75 04 1E 64 C5 70 F7
53	01 01 01 01 01 01 01 08	5A 59 45 28 BE BE F1 CC
54	01 01 01 01 01 01 01 04	FC DB 32 91 DE 21 F0 C0
55	01 01 01 01 01 01 01 02	86 9E FD 7F 9F 26 5A 09

Table A.3 Values To Be Used for the Permutation Operation Known Answer Test

for the TECB, TCBC, TCFB, and TOFB Modes of Operation

(NOTE -- P/TEXT = 00 00 00 00 00 00 00 00 for each round

where applicable, IV = 00 00 00 00 00 00 00 00.

ROUND	KEY	C/RESULT
0	10 46 91 34 89 98 01 31	88 D5 5E 54 F5 4C 97 B4
1	10 07 10 34 89 98 80 20	0C 0C C0 0C 83 EA 48 FD
2	10 07 10 34 C8 98 01 20	83 BC 8E F3 A6 57 01 83
3	10 46 10 34 89 98 80 20	DF 72 5D CA D9 4E A2 E9
4	10 86 91 15 19 19 01 01	E6 52 B5 3B 55 0B E8 B0
5	10 86 91 15 19 58 01 01	AF 52 71 20 C4 85 CB B0
6	51 07 B0 15 19 58 01 01	0F 04 CE 39 3D B9 26 D5
7	10 07 B0 15 19 19 01 01	C9 F0 0F FC 74 07 90 67
8	31 07 91 54 98 08 01 01	7C FD 82 A5 93 25 2B 4E
9	31 07 91 94 98 08 01 01	CB 49 A2 F9 E9 13 63 E3
10	10 07 91 15 B9 08 01 40	00 B5 88 BE 70 D2 3F 56
11	31 07 91 15 98 08 01 40	40 6A 9A 6A B4 33 99 AE

ROUND	KEY	C/RESULT
12	10 07 D0 15 89 98 01 01	6C B7 73 61 1D CA 9A DA
13	91 07 91 15 89 98 01 01	67 FD 21 C1 7D BB 5D 70
14	91 07 D0 15 89 19 01 01	95 92 CB 41 10 43 07 87
15	10 07 D0 15 98 98 01 20	A6 B7 FF 68 A3 18 DD D3
16	10 07 94 04 98 19 01 01	4D 10 21 96 C9 14 CA 16
17	01 07 91 04 91 19 04 01	2D FA 9F 45 73 59 49 65
18	01 07 91 04 91 19 01 01	B4 66 04 81 6C 0E 07 74
19	01 07 94 04 91 19 04 01	6E 7E 62 21 A4 F3 4E 87
20	19 07 92 10 98 1A 01 01	AA 85 E7 46 43 23 31 99
21	10 07 91 19 98 19 08 01	2E 5A 19 DB 4D 19 62 D6
22	10 07 91 19 98 1A 08 01	23 A8 66 A8 09 D3 08 94
23	10 07 92 10 98 19 01 01	D8 12 D9 61 F0 17 D3 20
24	10 07 91 15 98 19 01 0B	05 56 05 81 6E 58 60 8F
25	10 04 80 15 98 19 01 01	AB D8 8E 8B 1B 77 16 F1
26	10 04 80 15 98 19 01 02	53 7A C9 5B E6 9D A1 E1
27	10 04 80 15 98 19 01 08	AE D0 F6 AE 3C 25 CD D8

ROUND	KEY	C/RESULT
28	10 02 91 15 98 10 01 04	B3 E3 5A 5E E5 3E 7B 8D
29	10 02 91 15 98 19 01 04	61 C7 9C 71 92 1A 2E F8
30	10 02 91 15 98 10 02 01	E2 F5 72 8F 09 95 01 3C
31	10 02 91 16 98 10 01 01	1A EA C3 9A 61 F0 A4 64

Table A.4 Values To Be Used for the Substitution Table Known Answer Test

for the TECB, TCBC, TCFB, and TOFB Modes of Operation

ROUND	KEY	P/TEXT	C/RESULT
0	7C A1 10 45 4A 1A 6E 57	01 A1 D6 D0 39 77 67 42	69 0F 5B 0D 9A 26 93 9B
1	01 31 D9 61 9D C1 37 6E	5C D5 4C A8 3D EF 57 DA	7A 38 9D 10 35 4B D2 71
2	07 A1 13 3E 4A 0B 26 86	02 48 D4 38 06 F6 71 72	86 8E BB 51 CA B4 59 9A
3	38 49 67 4C 26 02 31 9E	51 45 4B 58 2D DF 44 0A	71 78 87 6E 01 F1 9B 2A
4	04 B9 15 BA 43 FE B5 B6	42 FD 44 30 59 57 7F A2	AF 37 FB 42 1F 8C 40 95
5	01 13 B9 70 FD 34 F2 CE	05 9B 5E 08 51 CF 14 3A	86 A5 60 F1 0E C6 D8 5B
6	01 70 F1 75 46 8F B5 E6	07 56 D8 E0 77 47 61 D2	0C D3 DA 02 00 21 DC 09
7	43 29 7F AD 38 E3 73 FE	76 25 14 B8 29 BF 48 6A	EA 67 6B 2C B7 DB 2B 7A
8	07 A7 13 70 45 DA 2A 16	3B DD 11 90 49 37 28 02	DF D6 4A 81 5C AF 1A 0F
9	04 68 91 04 C2 FD 3B 2F	26 95 5F 68 35 AF 60 9A	5C 51 3C 9C 48 86 C0 88
10	37 D0 6B B5 16 CB 75 46	16 4D 5E 40 4F 27 52 32	0A 2A EE AE 3F F4 AB 77
11	1F 08 26 0D 1A C2 46 5E	6B 05 6E 18 75 9F 5C CA	EF 1B F0 3E 5D FA 57 5A
12	58 40 23 64 1A BA 61 76	00 4B D6 EF 09 17 60 62	88 BF 0D B6 D7 0D EE 56
13	02 58 16 16 46 29 B0 07	48 0D 39 00 6E E7 62 F2	A1 F9 91 55 41 02 0B 56

ROUND	KEY	P/TEXT	C/RESULT
14	49 79 3E BC 79 B3 25 8F	43 75 40 C8 69 8F 3C FA	6F BF 1C AF CF FD 05 56
15	4F B0 5E 15 15 AB 73 A7	07 2D 43 A0 77 07 52 92	2F 22 E4 9B AB 7C A1 AC
16	49 E9 5D 6D 4C A2 29 BF	02 FE 55 77 81 17 F1 2A	5A 6B 61 2C C2 6C CE 4A
17	01 83 10 DC 40 9B 26 D6	1D 9D 5C 50 18 F7 28 C2	5F 4C 03 8E D1 2B 2E 41
18	1C 58 7F 1C 13 92 4F EF	30 55 32 28 6D 6F 29 5A	63 FA C0 D0 34 D9 F7 93

Table A.5 Resulting Ciphertext from the Variable Plaintext Known Answer Test for TCBC-I Mode of Operation

(NOTE -- KEY1 = KEY2 = KEY3 = 01 01 01 01 01 01 01 01

IV1 = 00 00 00 00 00 00 00 00

IV2 = 55 55 55 55 55 55 55 55

IV3 = aa aa aa aa aa aa aa aa)

ROUND	INPUTBLOCK 1	CIPHERTEXT1	INPUTBLOCK 2	CIPHERTEXT2	INPUTBLOCK 3	CIPHERTEXT3
0	8000000000000000	95f8a5e5dd31d900	d555555555555555	f7552ab6cb21e2bc	2aaaaaaaaaaaaaaa	5a48d3de869557fd
1	4000000000000000	dd7f121ca5015619	1555555555555555	e0c2af1ebd89a262	eaaaaaaaaaaaaaaa	f15ee2019a5b547c
2	2000000000000000	2e8653104f3834ea	7555555555555555	05b865a1e49ed109	8aaaaaaaaaaaaaaa	3bee595ef860316a
3	1000000000000000	4bd388ff6cd81d4f	4555555555555555	b447313fc704d321	baaaaaaaaaaaaaaa	f6089ca9b722765c
4	0800000000000000	20b9e767b2fb1456	5d55555555555555	c39193d42381b313	a2aaaaaaaaaaaaaa	af15a8e9b2c14de5
5	0400000000000000	55579380d77138ef	5155555555555555	6a2afdae188494b8	aeaaaaaaaaaaaaaa	45089186180bd591
6	0200000000000000	6cc5defaaf04512f	5755555555555555	1359f4d663a3209c	a8aaaaaaaaaaaaaa	280d3ae3a00cfbc9
7	0100000000000000	0d9f279ba5d87260	5455555555555555	4a035e6a81d1314b	abaaaaaaaaaaaaaa	d27eb94e56c3172a
8	0080000000000000	d9031b0271bd5a0a	55d5555555555555	4334b5fe1b7f5320	aa2aaaaaaaaaaaaa	b0555ab990b7e95c
9	0040000000000000	424250b37c3dd951	5515555555555555	f41a29e0d31107b4	aaeaaaaaaaaaaaaa	f54f2bd8e2eb2bc6
10	0020000000000000	b8061b7ecd9a21e5	5575555555555555	c8eb2e340855325b	aa8aaaaaaaaaaaaa	d51175259c607fb4
11	0010000000000000	f15d0f286b65bd28	5545555555555555	b75847a2f3f2458a	aabaaaaaaaaaaaaa	72ea3aadb569af43

ROUND	INPUTBLOCK 1	CIPHERTEXT1	INPUTBLOCK 2	CIPHERTEXT2	INPUTBLOCK 3	CIPHERTEXT3
12	0008000000000000	add0cc8d6e5deba1	555d555555555555	be433af4c5ae0f97	aaa2aaaaaaaaaaaa	9b003151e8602b7d
13	0004000000000000	e6d5f82752ad63d1	5551555555555555	f68101d125e2e284	aaaeaaaaaaaaaaaa	fc1463bb9bba9e11
14	0002000000000000	ecbfe3bd3f591a5e	5557555555555555	fa510732fa871094	aaa8aaaaaaaaaaaa	65f94c59c59b06e1
15	0001000000000000	f356834379d165cd	5554555555555555	458d97a8b6ebd0d7	aaabaaaaaaaaaaaa	fbcfc086f8111572
16	0000800000000000	2b9f982f20037fa9	5555d55555555555	f4169ca3fc6799ed	aaaa2aaaaaaaaaaa	68c9e70b9de8db79
17	0000400000000000	889de068a16f0be6	5555155555555555	f47b9f01a5ee74e9	aaaaeaaaaaaaaaaa	63fc8ec1421399b8
18	0000200000000000	e19e275d846a1298	5555755555555555	ee26a403caca387d	aaaa8aaaaaaaaaaa	3f1d10e9a1a44a92
19	0000100000000000	329a8ed523d71aec	5555455555555555	af7e5ad1d9f4ecf8	aaaabaaaaaaaaaaa	e3f663de44003f9b
20	0000080000000000	e7fce22557d23c97	55555d5555555555	bb04e854f99f6352	aaaaa2aaaaaaaaaa	bc2452fd13e00dcc
21	0000040000000000	12a9f5817ff2d65d	5555515555555555	01f57b1e69290d90	aaaaaeaaaaaaaaaa	4432a11e1c320e7a
22	0000020000000000	a484c3ad38dc9c19	5555575555555555	8ae9dee849b46527	aaaaa8aaaaaaaaaa	a1e9e67f13f932b3
23	0000010000000000	fbe00a8a1ef8ad72	5555545555555555	cb706efba6b5110e	aaaaabaaaaaaaaaa	6fd1d0793c1b7af2
24	0000008000000000	750d079407521363	555555d555555555	b8b27d1286bdbb26	aaaaaa2aaaaaaaaa	3d2c39f9d26b589e
25	0000004000000000	64feed9c724c2faf	5555551555555555	9862c9d770558095	aaaaaaeaaaaaaaaa	e3a7abc88132ad7d
26	0000002000000000	f02b263b328e2b60	5555557555555555	a213c5c56fdca139	aaaaaa8aaaaaaaaa	08cd945738a222c8
27	0000001000000000	9d64555a9a10b852	5555554555555555	a3bebc0e23ab87f2	aaaaaabaaaaaaaaa	568fa34d2fc7225e

ROUND	INPUTBLOCK 1	CIPHERTEXT1	INPUTBLOCK 2	CIPHERTEXT2	INPUTBLOCK 3	CIPHERTEXT3
28	0000000800000000	d106ff0bed5255d7	5555555d55555555	c32c19229d84e2b4	aaaaaaa2aaaaaaaa	3771887d7266b49d
29	0000000400000000	e1652c6b138c64a5	5555555155555555	e628ceae5cb3bb34	aaaaaaaeaaaaaaaa	edd6029a6b80a442
30	0000000200000000	e428581186ec8f46	5555555755555555	5924454953ad5732	aaaaaaa8aaaaaaaa	0313da097aec4a43
31	0000000100000000	aeb5f5ede22d1a36	5555555455555555	7cc987f5fb33b813	aaaaaaabaaaaaaaa	91f5b30f015b4a54
32	0000000080000000	e943d7568aec0c5c	55555555d5555555	88e3dd1448c4e0ff	aaaaaaaa2aaaaaaa	1e60759f038beec1
33	0000000040000000	df98c8276f54b04b	5555555515555555	a49d286e5dfc6143	aaaaaaaaeaaaaaaa	97061699383bbfe0
34	0000000020000000	b160e4680f6c696f	5555555575555555	a5206a311e9c2515	aaaaaaaa8aaaaaaa	311f3c96e071f173
35	0000000010000000	fa0752b07d9c4ab8	5555555545555555	b6e4686a8b957cf2	aaaaaaaabaaaaaaa	1a6849edcb701b07
36	0000000008000000	ca3a2b036dbc8502	555555555d555555	af1200418fd37fdd	aaaaaaaaa2aaaaaa	fa5b2fa26d03558b
37	0000000004000000	5e0905517bb59bcf	5555555551555555	487deccf0fde5b88	aaaaaaaaaeaaaaaa	bcaa0b7b7b3464c5
38	0000000002000000	814eeb3b91d90726	5555555557555555	456a1865905ed57d	aaaaaaaaa8aaaaaa	3d245b501c6abb74
39	0000000001000000	4d49db1532919c9f	5555555554555555	3e2601fa20895e62	aaaaaaaaabaaaaaa	62133d9330e2e86b
40	0000000000800000	25eb5fc3f8cf0621	5555555555d55555	58da89972266a7e3	aaaaaaaaaa2aaaaa	5d7d6bd225890b4d
41	0000000000400000	ab6a20c0620d1c6f	5555555555155555	feaca17e5dd05c87	aaaaaaaaaaeaaaaa	db36baba70c3b9af
42	0000000000200000	79e90dbc98f92cca	5555555555755555	88249b73e99c5ac0	aaaaaaaaaa8aaaaa	a2f5ea90c2179ab4
43	0000000000100000	866ecedd8072bb0e	5555555555455555	5f8add8784cc3174	aaaaaaaaaabaaaaa	70470a07cb34e109

ROUND	INPUTBLOCK 1	CIPHERTEXT1	INPUTBLOCK 2	CIPHERTEXT2	INPUTBLOCK 3	CIPHERTEXT3
44	0000000000080000	8b54536f2f3e64a8	55555555555d5555	cd8dc942ae2bb175	aaaaaaaaaaa2aaaa	659610094ab3824e
45	0000000000040000	ea51d3975595b86b	5555555555515555	cf8442863e68e644	aaaaaaaaaaaeaaaa	26e6223634c857a3
46	0000000000020000	caffc6ac4542de31	5555555555575555	16952dc89c0acd65	aaaaaaaaaaa8aaaa	ddd0a647be96041f
47	0000000000010000	8dd45a2ddf90796c	5555555555545555	8a4fca2b00c49807	aaaaaaaaaaabaaaa	363219d8cec5a9f3
48	0000000000008000	1029d55e880ec2d0	555555555555d555	b40225aea121c8d3	aaaaaaaaaaaa2aaa	bb5710f9dc8dde46
49	0000000000004000	5d86cb23639dbea9	5555555555551555	711c066c13222f1c	aaaaaaaaaaaaeaaa	ae527ed311a25ea2
50	0000000000002000	1d1ca853ae7c0c5f	5555555555557555	4fb69c832db68026	aaaaaaaaaaaa8aaa	af94496800a32656
51	0000000000001000	ce332329248f3228	5555555555554555	f24c7444edf1c394	aaaaaaaaaaaabaaa	c55d7544a1eae274
52	0000000000000800	8405d1abe24fb942	5555555555555d55	6be457abc511e87c	aaaaaaaaaaaaa2aa	9ba49db251748896
53	0000000000000400	e643d78090ca4207	5555555555555155	6136fefebb0c8118	aaaaaaaaaaaaaeaa	3d19267de9c12e7b
54	0000000000000200	48221b9937748a23	5555555555555755	d23a8dfe39c98883	aaaaaaaaaaaaa8aa	5ce84637532650c8
55	0000000000000100	dd7c0bbd61fafd54	5555555555555455	afe2e34f009924e2	aaaaaaaaaaaaabaa	d43941ab72932bb0
56	0000000000000080	2fbc291a570db5c4	55555555555555d5	0adcf552ec1754c6	aaaaaaaaaaaaaa2a	816c454ba7894865
57	0000000000000040	e07c30d7e4e26e12	5555555555555515	c06e80c5238135bb	aaaaaaaaaaaaaaea	74bc744f10f63889
58	0000000000000020	0953e2258e8e90a1	5555555555555575	0912754e7c42f637	aaaaaaaaaaaaaa8a	3d2565d9bf62cdbd
59	0000000000000010	5b711bc4ceebf2ee	5555555555555545	b4f82967c658adb8	aaaaaaaaaaaaaaba	a2e13c5701a60444

ROUND	INPUTBLOCK 1	CIPHERTEXT1	INPUTBLOCK 2	CIPHERTEXT2	INPUTBLOCK 3	CIPHERTEXT3
60	0000000000000008	cc083f1e6d9e85f6	555555555555555d	006fa12a796ac4d3	aaaaaaaaaaaaaaa2	cbe2873fd6f63048
61	0000000000000004	d2fd8867d50d2dfe	5555555555555551	1a4a364616460d44	aaaaaaaaaaaaaaae	cc6adcef1be975ef
62	0000000000000002	06e7ea22ce92708f	5555555555555557	f307b5bcd44f3d8d	aaaaaaaaaaaaaaa8	991d770b2bf051dc
63	0000000000000001	166b40b44aba4bd6	5555555555555554	9cb1c3932c005c49	aaaaaaaaaaaaaaab	17d8e9c374d14494

Table A.6 Resulting Ciphertext from the Inverse Permutation Known Answer Test for TCBC-I Mode of Operation (Encryption Process)

(NOTE -- KEY1 = KEY2 = KEY3 = 01 01 01 01 01 01 01 01
IV1 = 00 00 00 00 00 00 00 00
IV2 = 55 55 55 55 55 55 55 55
IV3 = aa aa aa aa aa aa aa aa)

ROUND	PLAINTEXT1	CIPHERTEXT1	PLAINTEXT2	CIPHERTEXT2	PLAINTEXT3	CIPHERTEXT3
0	95f8a5e5dd31d900	8000000000000000	f7552ab6cb21e2bc	713d058fe58a43f7	5a48d3de869557fd	e4999d5c3cceee44
1	dd7f121ca5015619	4000000000000000	e0c2af1ebd89a262	0ac760c01e5927ef	f15ee2019a5b547c	accd15b5dde0b5c2
2	2e8653104f3834ea	2000000000000000	05b865a1e49ed109	363130ca94da9d8a	3bee595ef860316a	69732f3dbb5652b1
3	4bd388ff6cd81d4f	1000000000000000	b447313fc704d321	1e14d9109bc1f46c	f6089ca9b722765c	ace935a115450a05
4	20b9e767b2fb1456	0800000000000000	c39193d42381b313	6a46ef972da6a833	af15a8e9b2c14de5	c1b2f69f9a21090d
5	55579380d77138ef	0400000000000000	6a2afdae188494b8	330aec7886295181	45089186180bd591	a8f987e6d0d3af25
6	6cc5defaaf04512f	0200000000000000	1359f4d663a3209c	e518b154c8b8c8a6	280d3ae3a00cfbc9	87f0fbcb6b40af68
7	0d9f279ba5d87260	0100000000000000	4a035e6a81d1314b	8dec119b560a53d0	d27eb94e56c3172a	6aa899298c76715b
8	d9031b0271bd5a0a	0080000000000000	4334b5fe1b7f5320	d8807ced29f8f8d1	b0555ab990b7e95c	7f17a4e7532b04f9
9	424250b37c3dd951	0040000000000000	f41a29e0d31107b4	dbe8eba35e2a295b	f54f2bd8e2eb2bc6	5c899d0cf0f8a135
10	b8061b7ecd9a21e5	0020000000000000	c8eb2e340855325b	fa5b70d1b836e88d	d51175259c607fb4	726616043a1c0107
11	f15d0f286b65bd28	0010000000000000	b75847a2f3f2458a	4be2d4ffa6f22133	72ea3aadb569af43	ba0432be3b5bb6f8
12	add0cc8d6e5deba1	0008000000000000	be433af4c5ae0f97	b85a5c395b3a5885	9b003151e8602b7d	e40807ea13dd109e
13	e6d5f82752ad63d1	0004000000000000	f68101d125e2e284	9f65cff48d26c258	fc1463bb9bba9e11	7851707ef934aa75
14	ecbfe3bd3f591a5e	0002000000000000	fa510732fa871094	40e8813c718539ac	65f94c59c59b06e1	d51aab52aa37dc8d
15	f356834379d165cd	0001000000000000	458d97a8b6ebd0d7	289a7729f22d7703	fbcfc086f8111572	266e7b0862cf5fc2
16	2b9f982f20037fa9	0000800000000000	f4169ca3fc6799ed	a11b556e8c1b26c5	68c9e70b9de8db79	aedab274b2ef15c9
17	889de068a16f0be6	0000400000000000	f47b9f01a5ee74e9	3683a86916c7b11d	63fc8ec1421399b8	80fbb2539dd96d8f
18	e19e275d846a1298	0000200000000000	ee26a403caca387d	9f073f4f068f3d0e	3f1d10e9a1a44a92	498437929c6ccf59
19	329a8ed523d71aec	0000100000000000	af7e5ad1d9f4ecf8	07712f196c02eb9b	e3f663de44003f9b	c4ebb01e305e41e2
20	e7fce22557d23c97	0000080000000000	bb04e854f99f6352	93f4126615626c01	bc2452fd13e00dcc	82fb4a9ce4c92818
21	12a9f5817ff2d65d	0000040000000000	01f57b1e69290d90	b6958170aba384c9	4432a11e1c320e7a	91239239e22f0280
22	a484c3ad38dc9c19	0000020000000000	8ae9dee849b46527	3bb724cf5e35707d	a1e9e67f13f932b3	cc30662b51d40c1a
23	fbe00a8a1ef8ad72	0000010000000000	cb706efba6b5110e	9fe1afb876cdb756	6fd1d0793c1b7af2	8e67cf5371a467a2

ROUND	PLAINTEXT1	CIPHERTEXT1	PLAINTEXT2	CIPHERTEXT2	PLAINTEXT3	CIPHERTEXT3
24	750d079407521363	0000008000000000	b8b27d1286bdbb26	1db03e2b95785d8a	3d2c39f9d26b589e	6e79366486097eba
25	64feed9c724c2faf	0000004000000000	9862c9d770558095	ea4e26144ada8e2b	e3a7abc88132ad7d	ce2971055091a1af
26	f02b263b328e2b60	0000002000000000	a213c5c56fdca139	97255bd98b5ed9b3	08cd945738a222c8	252e33166953cd68
27	9d64555a9a10b852	0000001000000000	a3bebc0e23ab87f2	85a52d6656cf13be	568fa34d2fc7225e	39a971317391242b
28	d106ff0bed5255d7	0000000800000000	c32c19229d84e2b4	6965b2633fbe37a8	3771887d7266b49d	d95a7aa0bec4fa7a
29	e1652c6b138c64a5	0000000400000000	e628ceae5cb3bb34	0e8317ae44e3caa0	edd6029a6b80a442	4dfdcc7a4279b2c0
30	e428581186ec8f46	0000000200000000	5924454953ad5732	567efb50dc99f5dc	0313da097aec4a43	96bb89c941631bed
31	aeb5f5ede22d1a36	0000000100000000	7cc987f5fb33b813	46814855930b3a3f	91f5b30f015b4a54	1c3ba8fbadab9a22
32	e943d7568aec0c5c	0000000080000000	88e3dd1448c4e0ff	a77142eabd2bd877	1e60759f038beec1	8fc77798b1692ab2
33	df98c8276f54b04b	0000000040000000	a49d286e5dfc6143	76395f51bdf699db	97061699383bbfe0	ace5681dfba69ceb
34	b160e4680f6c696f	0000000020000000	a5206a311e9c2515	c3e20437ad6c32b7	311f3c96e071f173	782058f728c21174
35	fa0752b07d9c4ab8	0000000010000000	b6e4686a8b957cf2	34cfbfca8df5fb9d	1a6849edcb701b07	fc14dafe9d171db5
36	ca3a2b036dbc8502	0000000008000000	af1200418fd37fdd	b372320762d438f8	fa5b2fa26d03558b	339189931ada4474
37	5e0905517bb59bcf	0000000004000000	487deccf0fde5b88	882402b6dec6675f	bcaa0b7b7b3464c5	c6d1f875363bf7ea
38	814eeb3b91d90726	0000000002000000	456a1865905ed57d	69e1758b520187d4	3d245b501c6abb74	31097d931da2e7bd
39	4d49db1532919c9f	0000000001000000	3e2601fa20895e62	ab8232a31d78e0fc	62133d9330e2e86b	0bff0085bb36e9b0
40	25eb5fc3f8cf0621	0000000000800000	58da89972266a7e3	aeed06b9f51ce37a	5d7d6bd225890b4d	5d09a28ee99cb585
41	ab6a20c0620d1c6f	0000000000400000	feaca17e5dd05c87	96dc5bd6e0b10d83	db36baba70c3b9af	46d9a629a0616379
42	79e90dbc98f92cca	0000000000200000	88249b73e99c5ac0	55a4cdc28ecf0541	a2f5ea90c2179ab4	ab239da3e3fab21b
43	866ecedd8072bb0e	0000000000100000	5f8add8784cc3174	7349bfc7f6461210	70470a07cb34e109	9331573af5067b09
44	8b54536f2f3e64a8	0000000000080000	cd8dc942ae2bb175	90b4544c9e6ad23b	659610094ab3824e	3133eeddd4f2ffec
45	ea51d3975595b86b	0000000000040000	cf8442863e68e644	2d7e77de47d0dad4	26e6223634c857a3	408e7d58ba623208
46	caffc6ac4542de31	0000000000020000	16952dc89c0acd65	b87887b6dddaab6f	ddd0a647be96041f	0e5b54a5a9cfbed1
47	8dd45a2ddf90796c	0000000000010000	8a4fca2b00c49807	8fdec1977d446e54	363219d8cec5a9f3	b875b2ffa6fea146
48	1029d55e880ec2d0	0000000000008000	b40225aea121c8d3	aedc1e02bd099571	bb5710f9dc8dde46	1a190ba501176f51
49	5d86cb23639dbea9	0000000000004000	711c066c13222f1c	1404bcbe41ce6aa1	ae527ed311a25ea2	863541107db40094
50	1d1ca853ae7c0c5f	0000000000002000	4fb69c832db68026	83804ddd1b5cd4fd	af94496800a32656	0d3834749def9e7a
51	ce332329248f3228	0000000000001000	f24c7444edf1c394	5f54383a55d6198a	c55d7544a1eae274	b601d210b21d541b
52	8405d1abe24fb942	0000000000000800	6be457abc511e87c	f1c2172a084f656f	9ba49db251748896	50d294abb12450bb
53	e643d78090ca4207	0000000000000400	6136fefebb0c8118	88b53f4066285776	3d19267de9c12e7b	010a1b96b9017a94
54	48221b9937748a23	0000000000000200	d23a8dfe39c98883	4dc3b1bc755eb684	5ce84637532650c8	15acb37fde2a095a
55	dd7c0bbd61fafd54	0000000000000100	afe2e34f009924e2	45c93fbf9ea29104	d43941ab72932bb0	7bd2597948ce5bc8

ROUND	PLAINTEXT1	CIPHERTEXT1	PLAINTEXT2	CIPHERTEXT2	PLAINTEXT3	CIPHERTEXT3
56	2fbc291a570db5c4	0000000000000080	0adcf552ec1754c6	e5c336ae5360d967	816c454ba7894865	b3f30f939f9bc4db
57	e07c30d7e4e26e12	0000000000000040	c06e80c5238135bb	31c1c1914e9d7278	74bc744f10f63889	d30cbd5808d8e0ef
58	0953e2258e8e90a1	0000000000000020	0912754e7c42f637	ca1dad0fa1978258	3d2565d9bf62cdbd	b30b208b6ccecada
59	5b711bc4ceebf2ee	0000000000000010	b4f82967c658adb8	afd29a3fba18602a	a2e13c5701a60444	027d03f04016c3c2
60	cc083f1e6d9e85f6	0000000000000008	006fa12a796ac4d3	c291dff5ec01e8b3	cbe2873fd6f63048	c0950b7f3c1bfaca
61	d2fd8867d50d2dfe	0000000000000004	1a4a364616460d44	6491ba623149f3d0	cc6adcef1be975ef	2e475e2153d1c64a
62	06e7ea22ce92708f	0000000000000002	f307b5bcd44f3d8d	87c6963b33be0353	991d770b2bf051dc	f8f7ded629f3fc48
63	166b40b44aba4bd6	0000000000000001	9cb1c3932c005c49	4fce2baa2cd647d3	17d8e9c374d14494	776bd1e53ef1d7d6

Table A.7 Resulting Ciphertext from the Initial Permutation Known Answer Test for TCBC-I Mode of Operation (Decryption Process)

(NOTE -- KEY1 = KEY2 = KEY3 01 01 01 01 01 01 01 01
IV1 = 00 00 00 00 00 00 00 00
IV2 = 55 55 55 55 55 55 55 55
IV3 = aa aa aa aa aa aa aa aa)

ROUND	CIPHERTEXTS	PLAINTEXT1	PLAINTEXT2	PLAINTEXT3
0	8000000000000000	95f8a5e5dd31d900	c0adf0b088648c55	3f520f4f779b73aa
1	4000000000000000	dd7f121ca5015619	882a4749f054034c	77d5b8b60fabfcb3
2	2000000000000000	2e8653104f3834ea	7bd306451a6d61bf	842cf9bae5929e40
3	1000000000000000	4bd388ff6cd81d4f	1e86ddaa398d481a	e1792255c672b7e5
4	0800000000000000	20b9e767b2fb1456	75ecb232e7ae4103	8a134dcd1851befc
5	0400000000000000	55579380d77138ef	0002c6d582246dba	fffd392a7ddb9245
6	0200000000000000	6cc5defaaf04512f	39908baffa51047a	c66f745005aefb85
7	0100000000000000	0d9f279ba5d87260	58ca72cef08d2735	a7358d310f72d8ca
8	0080000000000000	d9031b0271bd5a0a	8c564e5724e80f5f	73a9b1a8db17f0a0
9	0040000000000000	424250b37c3dd951	171705e629688c04	e8e8fa19d69773fb
10	0020000000000000	b8061b7ecd9a21e5	ed534e2b98cf74b0	12acb1d467308b4f
11	0010000000000000	f15d0f286b65bd28	a4085a7d3e30e87d	5bf7a582c1cf1782
12	0008000000000000	add0cc8d6e5deba1	f88599d83b08bef4	077a6627c4f7410b
13	0004000000000000	e6d5f82752ad63d1	b380ad7207f83684	4c7f528df807c97b
14	0002000000000000	ecbfe3bd3f591a5e	b9eab6e86a0c4f0b	4615491795f3b0f4
15	0001000000000000	f356834379d165cd	a603d6162c843098	59fc29e9d37bcf67
16	0000800000000000	2b9f982f20037fa9	7ecacd7a75562afc	813532858aa9d503
17	0000400000000000	889de068a16f0be6	ddc8b53df43a5eb3	22374ac20bc5a14c
18	0000200000000000	e19e275d846a1298	b4cb7208d13f47cd	4b348df72ec0b832
19	0000100000000000	329a8ed523d71aec	67cfdb8076824fb9	9830247f897db046
20	0000080000000000	e7fce22557d23c97	b2a9b770028769c2	4d56488ffd78963d
21	0000040000000000	12a9f5817ff2d65d	47fca0d42aa78308	b8035f2bd5587cf7
22	0000020000000000	a484c3ad38dc9c19	f1d196f86d89c94c	0e2e6907927636b3

ROUND	CIPHERTEXTS	PLAINTEXT1	PLAINTEXT2	PLAINTEXT3
23	0000010000000000	fbe00a8a1ef8ad72	aeb55fdf4badf827	514aa020b45207d8
24	0000008000000000	750d079407521363	205852c152074636	dfa7ad3eadf8b9c9
25	0000004000000000	64feed9c724c2faf	31abb8c927197afa	ce544736d8e68505
26	0000002000000000	f02b263b328e2b60	a57e736e67db7e35	5a818c91982481ca
27	0000001000000000	9d64555a9a10b852	c831000fcf45ed07	37cefff030ba12f8
28	0000000800000000	d106ff0bed5255d7	8453aa5eb8070082	7bac55a147f8ff7d
29	0000000400000000	e1652c6b138c64a5	b430793e46d931f0	4bcf86c1b926ce0f
30	0000000200000000	e428581186ec8f46	b17d0d44d3b9da13	4e82f2bb2c4625ec
31	0000000100000000	aeb5f5ede22d1a36	fbe0a0b8b7784f63	041f5f474887b09c
32	0000000080000000	e943d7568aec0c5c	bc168203dfb95909	43e97dfc2046a6f6
33	0000000040000000	df98c8276f54b04b	8acd9d723a01e51e	7532628dc5fe1ae1
34	0000000020000000	b160e4680f6c696f	e435b13d5a393c3a	1bca4ec2a5c6c3c5
35	0000000010000000	fa0752b07d9c4ab8	af5207e528c91fed	50adf81ad736e012
36	0000000008000000	ca3a2b036dbc8502	9f6f7e5638e9d057	609081a9c7162fa8
37	0000000004000000	5e0905517bb59bcf	0b5c50042ee0ce9a	f4a3affbd11f3165
38	0000000002000000	814eeb3b91d90726	d41bbe6ec48c5273	2be441913b73ad8c
39	0000000001000000	4d49db1532919c9f	181c8e4067c4c9ca	e7e371bf983b3635
40	0000000000800000	25eb5fc3f8cf0621	70be0a96ad9a5374	8f41f5695265ac8b
41	0000000000400000	ab6a20c0620d1c6f	fe3f75953758493a	01c08a6ac8a7b6c5
42	0000000000200000	79e90dbc98f92cca	2cbc58e9cdac799f	d343a71632538660
43	0000000000100000	866ecedd8072bb0e	d33b9b88d527ee5b	2cc464772ad811a4
44	0000000000080000	8b54536f2f3e64a8	de01063a7a6b31fd	21fef9c58594ce02
45	0000000000040000	ea51d3975595b86b	bf0486c200c0ed3e	40fb793dff3f12c1
46	0000000000020000	caffc6ac4542de31	9faa93f910178b64	60556c06efe8749b
47	0000000000010000	8dd45a2ddf90796c	d8810f788ac52c39	277ef087753ad3c6
48	0000000000008000	1029d55e880ec2d0	457c800bdd5b9785	ba837ff422a4687a
49	0000000000004000	5d86cb23639dbea9	08d39e7636c8ebfc	f72c6189c9371403
50	0000000000002000	1d1ca853ae7c0c5f	4849fd06fb29590a	b7b602f904d6a6f5
51	0000000000001000	ce332329248f3228	9b66767c71da677d	649989838e259882
52	0000000000000800	8405d1abe24fb942	d15084feb71aec17	2eaf7b0148e513e8
53	0000000000000400	e643d78090ca4207	b31682d5c59f1752	4ce97d2a3a60e8ad
54	0000000000000200	48221b9937748a23	1d774ecc6221df76	e288b1339dde2089

ROUND	CIPHERTEXTS	PLAINTEXT1	PLAINTEXT2	PLAINTEXT3
55	0000000000000100	dd7c0bbd61fafd54	88295ee834afa801	77d6a117cb5057fe
56	0000000000000080	2fbc291a570db5c4	7ae97c4f0258e091	851683b0fda71f6e
57	0000000000000040	e07c30d7e4e26e12	b5296582b1b73b47	4ad69a7d4e48c4b8
58	0000000000000020	0953e2258e8e90a1	5c06b770dbdbc5f4	a3f9488f24243a0b
59	0000000000000010	5b711bc4ceebf2ee	0e244e919bbea7bb	f1dbb16e64415844
60	0000000000000008	cc083f1e6d9e85f6	995d6a4b38cbd0a3	66a295b4c7342f5c
61	0000000000000004	d2fd8867d50d2dfe	87a8dd32805878ab	785722cd7fa78754
62	0000000000000002	06e7ea22ce92708f	53b2bf779bc725da	ac4d40886438da25
63	0000000000000001	166b40b44aba4bd6	433e15e11fef1e83	bcc1ea1ee010e17c

Table A.8 Values To Be Used for the Substitution Table Known Answer Test

for TCBC-I Mode of Operation

(NOTE -- IV1 = 00 00 00 00 00 00 00 00
IV2 = 55 55 55 55 55 55 55 55
IV3 = aa aa aa aa aa aa aa aa)

ROUND	KEY	PLAINTEXTS	CIPHERTEXT1	CIPHERTEXT2	CIPHERTEXT3
0	7ca110454a1a6e57	01a1d6d039776742	690f5b0d9a26939b	89202f224f1f2261	585a1e8d89705d10
1	0131d9619dc1376e	5cd54ca83def57da	7a389d10354bd271	6dda0de99d3c86b9	99985b67b598bd25
2	07a1133e4a0b2686	0248d43806f67172	868ebb51cab4599a	8200616c589bc7aa	d2ff67461377fbb5
3	3849674c2602319e	51454b582ddf440a	7178876e01f19b2a	64757292febccad1	93bd8beeea2310fc
4	04b915ba43feb5b6	42fd443059577fa2	af37fb421f8c4095	204fc6123992d4e9	6bfb4df0569cebce
5	0113b970fd34f2ce	059b5e0851cf143a	86a560f10ec6d85b	1fa86f6f735603a3	0be3558738c6d7c3
6	0170f175468fb5e6	0756d8e0774761d2	0cd3da020021dc09	65e05d62b35aa365	3bfc9a3f034da292
7	43297fad38e373fe	762514b829bf486a	ea676b2cb7db2b7a	95c0f9e595aec2ff	ea9ab3585f166586
8	07a7137045da2a16	3bdd119049372802	dfd64a815caf1a0f	127359c20e10e25a	953a36ff13a08906
9	04689104c2fd3b2f	26955f6835af609a	5c513c9c4886c088	b089d90f84ef0c4c	08bd60f6f80d6fad
10	37d06bb516cb7546	164d5e404f275232	0a2aeeae3ff4ab77	32bbdd67d4e66dd6	83a30606fc78d740
11	1f08260d1ac2465e	6b056e18759f5cca	ef1bf03e5dfa575a	b4873081fdebc81d	6445799c9b701694
12	584023641aba6176	004bd6ef09176062	88bf0db6d70dee56	988fe2e8e1755e78	1e1fdd8660a75bb5
13	025816164629b007	480d39006ee762f2	a1f9915541020b56	ee6c0febb212b218	60bae59c51767394
14	49793ebc79b3258f	437540c8698f3cfa	6fbf1cafcffd0556	c03adc2b6aa85b5b	826ec7e02f486885
15	4fb05e1515ab73a7	072d43a077075292	2f22e49bab7ca1ac	096a4136e0f65f76	9e30377b7a39d5d3
16	49e95d6d4ca229bf	02fe55778117f12a	5a6b612cc26cce4a	bf4da6aa59ed5751	64b77306321a932c
17	018310dc409b26d6	1d9d5c5018f728c2	5f4c038ed12b2e41	aab93390e13d3bb3	3b17daff733fcfb0
18	1c587f1c13924fef	305532286d6f295a	63fac0d034d9f793	db3c4106c5db5648	7f38215d73b0ee62

Table A.9 Resulting Ciphertext from the Variable TEXT Known Answer Test

for TCFB-P and TOFB-I Modes of Operation

(NOTE -- TEXT = 00 00 00 00 00 00 00 00

IV1 = 00 00 00 00 00 00 00 00

IV2 = 55 55 55 55 55 55 55 55

IV3 = aa aa aa aa aa aa aa aa)

RND	PLAINTEXT1 ⊕ IV1	CIPHERTEXT1	PLAINTEXT2 ⊕ IV2	CIPHERTEXT2	PLAINTEXT3 ⊕ IV3	CIPHERTEXT3
0	8000000000000000	95f8a5e5dd31d900	d555555555555555	f7552ab6cb21e2bc	2aaaaaaaaaaaaaaa	5a48d3de869557fd
1	4000000000000000	dd7f121ca5015619	9555555555555555	0c783d97d0dbf51a	eaaaaaaaaaaaaaaa	f15ee2019a5b547c
2	2000000000000000	2e8653104f3834ea	7555555555555555	05b865a1e49ed109	caaaaaaaaaaaaaaa	f925b68465b6078c
3	1000000000000000	4bd388ff6cd81d4f	6555555555555555	9e51152dbce90b02	baaaaaaaaaaaaaaa	f6089ca9b722765c
4	0800000000000000	20b9e767b2fb1456	5d55555555555555	c39193d42381b313	b2aaaaaaaaaaaaaa	4f1b8036d441af95
5	0400000000000000	55579380d77138ef	5955555555555555	e293394891554b68	aeaaaaaaaaaaaaaa	45089186180bd591
6	0200000000000000	6cc5defaaf04512f	5755555555555555	1359f4d663a3209c	acaaaaaaaaaaaaaa	d86dd807085fa8e6
7	0100000000000000	0d9f279ba5d87260	5655555555555555	0d0f03e8f8594a66	abaaaaaaaaaaaaaa	d27eb94e56c3172a
8	0080000000000000	d9031b0271bd5a0a	55d5555555555555	4334b5fe1b7f5320	ab2aaaaaaaaaaaaa	d6ad42065e31bdb1
9	0040000000000000	424250b37c3dd951	5595555555555555	9484c1c29b62c41e	aaeaaaaaaaaaaaaa	f54f2bd8e2eb2bc6
10	0020000000000000	b8061b7ecd9a21e5	5575555555555555	c8eb2e340855325b	aacaaaaaaaaaaaaa	6cf8932328c7e49b

RND	PLAINTEXT1 ⊕ IV1	CIPHERTEXT1	PLAINTEXT2 ⊕ IV2	CIPHERTEXT2	PLAINTEXT3 ⊕ IV3	CIPHERTEXT3
11	0010000000000000	f15d0f286b65bd28	5565555555555555	e88a676ef848e6d1	aabaaaaaaaaaaaaa	72ea3aadb569af43
12	0008000000000000	add0cc8d6e5deba1	555d555555555555	be433af4c5ae0f97	aab2aaaaaaaaaaaa	0d71ecadd7a49fec
13	0004000000000000	e6d5f82752ad63d1	5559555555555555	9e32639bb9d27cc7	aaaeaaaaaaaaaaaa	fc1463bb9bba9e11
14	0002000000000000	ecbfe3bd3f591a5e	5557555555555555	fa510732fa871094	aaacaaaaaaaaaaaa	31568f2e0ac0d693
15	0001000000000000	f356834379d165cd	5556555555555555	9f1b31571ed41078	aaabaaaaaaaaaaaa	fbcfc086f8111572
16	0000800000000000	2b9f982f20037fa9	5555d55555555555	f4169ca3fc6799ed	aaab2aaaaaaaaaaa	d67ca5071769cafe
17	0000400000000000	889de068a16f0be6	5555955555555555	e9a738ac85e2ca4b	aaaaeaaaaaaaaaaa	63fc8ec1421399b8
18	0000200000000000	e19e275d846a1298	5555755555555555	ee26a403caca387d	aaaacaaaaaaaaaaa	5d84b7acabb63bfb
19	0000100000000000	329a8ed523d71aec	5555655555555555	0b3f88ef87d85953	aaaabaaaaaaaaaaa	e3f663de44003f9b
20	0000080000000000	e7fce22557d23c97	55555d5555555555	bb04e854f99f6352	aaaab2aaaaaaaaaa	4e5892f230b6d6d1
21	0000040000000000	12a9f5817ff2d65d	5555595555555555	f0881280455dec63	aaaaaeaaaaaaaaaa	4432a11e1c320e7a
22	0000020000000000	a484c3ad38dc9c19	5555575555555555	8ae9dee849b46527	aaaaacaaaaaaaaaa	02ce21a9c83ba4d6
23	0000010000000000	fbe00a8a1ef8ad72	5555565555555555	74b7d252cae558fb	aaaaabaaaaaaaaaa	6fd1d0793c1b7af2
24	0000008000000000	750d079407521363	555555d555555555	b8b27d1286bdbb26	aaaaab2aaaaaaaaa	fc286fa362d8c93c
25	0000004000000000	64feed9c724c2faf	5555559555555555	4e3dd222e292dd96	aaaaaaeaaaaaaaaa	e3a7abc88132ad7d
26	0000002000000000	f02b263b328e2b60	5555557555555555	a213c5c56fdca139	aaaaaacaaaaaaaaa	8868d3114021a027
27	0000001000000000	9d64555a9a10b852	5555556555555555	05df49a56a345cf9	aaaaaabaaaaaaaaa	568fa34d2fc7225e

RND	PLAINTEXT1 ⊕ IV1	CIPHERTEXT1	PLAINTEXT2 ⊕ IV2	CIPHERTEXT2	PLAINTEXT3 ⊕ IV3	CIPHERTEXT3
28	0000000800000000	d106ff0bed5255d7	5555555d55555555	c32c19229d84e2b4	aaaaaab2aaaaaaaa	1f81cbb9403ecc59
29	0000000400000000	e1652c6b138c64a5	5555555955555555	89c6e06ce6164d84	aaaaaaaeaaaaaaaa	edd6029a6b80a442
30	0000000200000000	e428581186ec8f46	5555555755555555	5924454953ad5732	aaaaaaacaaaaaaaa	ef90911c0f9a66f3
31	0000000100000000	aeb5f5ede22d1a36	5555555655555555	7a3e15c0953b08cc	aaaaaaabaaaaaaaa	91f5b30f015b4a54
32	0000000080000000	e943d7568aec0c5c	55555555d5555555	88e3dd1448c4e0ff	aaaaaaab2aaaaaaa	a5aec2896cff08e5
33	0000000040000000	df98c8276f54b04b	5555555595555555	9f55ebaca42cb845	aaaaaaaaeaaaaaaa	97061699383bbfe0
34	0000000020000000	b160e4680f6c696f	5555555575555555	a5206a311e9c2515	aaaaaaaacaaaaaaa	08e218f2cb1ede18
35	0000000010000000	fa0752b07d9c4ab8	5555555565555555	e944c64af09dfa84	aaaaaaaabaaaaaaa	1a6849edcb701b07
36	0000000008000000	ca3a2b036dbc8502	555555555d555555	af1200418fd37fdd	aaaaaaaab2aaaaaa	85480c507233c006
37	0000000004000000	5e0905517bb59bcf	5555555559555555	574a377b5a150353	aaaaaaaaaeaaaaaa	bcaa0b7b7b3464c5
38	0000000002000000	814eeb3b91d90726	5555555557555555	456a1865905ed57d	aaaaaaaaacaaaaaa	0439f36972dc531f
39	0000000001000000	4d49db1532919c9f	5555555556555555	8427c42d027a34d0	aaaaaaaaabaaaaaa	62133d9330e2e86b
40	0000000000800000	25eb5fc3f8cf0621	5555555555d55555	58da89972266a7e3	aaaaaaaaab2aaaaa	f9c2472742b5f9e8
41	0000000000400000	ab6a20c0620d1c6f	5555555555955555	1ed858bcbc934c17	aaaaaaaaaaeaaaaa	db36baba70c3b9af
42	0000000000200000	79e90dbc98f92cca	5555555555755555	88249b73e99c5ac0	aaaaaaaaaacaaaaa	0758b13e912d53cb
43	0000000000100000	866ecedd8072bb0e	5555555555655555	69314212c7a9d6b1	aaaaaaaaaabaaaaa	70470a07cb34e109
44	0000000000080000	8b54536f2f3e64a8	55555555555d5555	cd8dc942ae2bb175	aaaaaaaaaab2aaaa	9c6ade3a9e772c7c

RND	PLAINTEXT1 ⊕ IV1	CIPHERTEXT1	PLAINTEXT2 ⊕ IV2	CIPHERTEXT2	PLAINTEXT3 ⊕ IV3	CIPHERTEXT3
45	0000000000040000	ea51d3975595b86b	5555555555595555	4c0a052894ed7436	aaaaaaaaaaaeaaaa	26e6223634c857a3
46	0000000000020000	caffc6ac4542de31	5555555555575555	16952dc89c0acd65	aaaaaaaaaaacaaaa	72dfd337fe183a6d
47	0000000000010000	8dd45a2ddf90796c	5555555555565555	92ef4c4350711745	aaaaaaaaaaabaaaa	363219d8cec5a9f3
48	0000000000008000	1029d55e880ec2d0	555555555555d555	b40225aea121c8d3	aaaaaaaaaaab2aaa	4bc89c1804bcae82
49	0000000000004000	5d86cb23639dbea9	5555555555559555	a9eab121edde0ca7	aaaaaaaaaaaaeaaa	ae527ed311a25ea2
50	0000000000002000	1d1ca853ae7c0c5f	5555555555557555	4fb69c832db68026	aaaaaaaaaaaacaaa	a1584c1024f61f3d
51	0000000000001000	ce332329248f3228	5555555555556555	761b3d1ff06c513e	aaaaaaaaaaaabaaa	c55d7544a1eae274
52	0000000000000800	8405d1abe24fb942	5555555555555d55	6be457abc511e87c	aaaaaaaaaaaab2aa	aef861c69fd34489
53	0000000000000400	e643d78090ca4207	5555555555555955	ebb5a1887b1f6e3a	aaaaaaaaaaaaaeaa	3d19267de9c12e7b
54	0000000000000200	48221b9937748a23	5555555555555755	d23a8dfe39c98883	aaaaaaaaaaaaacaa	ade513b3ed994800
55	0000000000000100	dd7c0bbd61fafd54	5555555555555655	9f986bb8f7e6fa46	aaaaaaaaaaaaabaa	d43941ab72932bb0
56	0000000000000080	2fbc291a570db5c4	55555555555555d5	0adcf552ec1754c6	aaaaaaaaaaaaab2a	7f7352dfade13e13
57	0000000000000040	e07c30d7e4e26e12	5555555555555595	6c25b868caf1f7d3	aaaaaaaaaaaaaaea	74bc744f10f63889
58	0000000000000020	0953e2258e8e90a1	5555555555555575	0912754e7c42f637	aaaaaaaaaaaaaaca	a483f2da4099a136
59	0000000000000010	5b711bc4ceebf2ee	5555555555555565	2fa6a76d9b83e3dd	aaaaaaaaaaaaaaba	a2e13c5701a60444
60	0000000000000008	cc083f1e6d9e85f6	555555555555555d	006fa12a796ac4d3	aaaaaaaaaaaaaab2	bc10a45ceedb56b3
61	0000000000000004	d2fd8867d50d2dfe	5555555555555559	6a0bd7954b5aa04d	aaaaaaaaaaaaaaae	cc6adcef1be975ef

RND	PLAINTEXT1 ⊕ IV1	CIPHERTEXT1	PLAINTEXT2 ⊕ IV2	CIPHERTEXT2	PLAINTEXT3 ⊕ IV3	CIPHERTEXT3
62	0000000000000002	06e7ea22ce92708f	5555555555555557	f307b5bcd44f3d8d	aaaaaaaaaaaaaaac	3dc004f9cd4a9c22
63	0000000000000001	166b40b44aba4bd6	5555555555555556	009e8232891c8a36	aaaaaaaaaaaaaaab	17d8e9c374d14494

Table A.10 Values to be Used for the Substitution Table Known Answer Test

for TCFB-P and TOFB-I Modes of Operation

(NOTE -- TEXT = 00 00 00 00 00 00 00 00)

RND	KEY	IV1	CIPHERTEXT2	IV2	CIPHERTEXT2	IV3	CIPHERTEXT3
0	7ca110454a1a6e57	01a1d6d039776742	690f5b0d9a26939b	56f72c258eccbc97	97fc1b9381f05ffa	ac4c817ae42211ec	e90a658ca212b240
1	0131d9619dc1376e	5cd54ca83def57da	7a389d10354bd271	b22aa1fd9344ad2f	1697f74514a33238	077ff752e89a0284	21329d25683b4606
2	07a1133e4a0b2686	0248d43806f67172	868ebb51cab4599a	579e298d5c4bc6c7	3c33dc00289664d0	acf37ee2b1a11c1c	66477e326b77dd91
3	3849674c2602319e	51454b582ddf440a	7178876e01f19b2a	a69aa0ad8334995f	941fcf0e43a965af	fbeff602d889eeb4	8d71d3da699fa6f5
4	04b915ba43feb5b6	42fd443059577fa2	af37fb421f8c4095	98529985aeacd4f7	1e327e778501022a	eda7eedb04022a4c	9e547f92a9ad358c
5	0113b970fd34f2ce	059b5e0851cf143a	86a560f10ec6d85b	5af0b35da724698f	637038eaaa7d167e	b04608b2fc79bee4	6f975aa305eb7548
6	0170f175468fb5e6	0756d8e0774761d2	0cd3da020021dc09	5cac2e35cc9cb727	1c7fe0ddc80d3f6e	b201838b21f20c7c	cad8716fc1176297
7	43297fad38e373fe	762514b829bf486a	ea676b2cb7db2b7a	cb7a6a0d7f149dbf	4b36062823e8190f	20cfbf62d469f314	664e8d98d3986cfe
8	07a7137045da2a16	3bdd119049372802	dfd64a815caf1a0f	913266e59e8c7d57	1ff289bc8e07c5f3	e687bc3af3e1d2ac	948ab876125e7c7f
9	04689104c2fd3b2f	26955f6835af609a	5c513c9c4886c088	7beab4bd8b04b5ef	19f76ad4a415b1c1	d1400a12e05a0b44	75d6085d1b1e472d
10	37d06bb516cb7546	164d5e404f275232	0a2aeeae3ff4ab77	6ba2b395a47ca787	c78b293dc022c9aa	c0f808eaf9d1fcdc	6ac4da432141aa16
11	1f08260d1ac2465e	6b056e18759f5cca	ef1bf03e5dfa575a	c05ac36dcaf4b21f	5469ad2a9c97bf19	15b018c3204a0774	9983b852b915da86
12	584023641aba6176	004bd6ef09176062	88bf0db6d70dee56	55a12c445e6cb5b7	77aeb7e9d51577e5	aaf68199b3c20b0c	fb716445f1a43232
13	025816164629b007	480d39006ee762f2	a1f9915541020b56	9d628e55c43cb847	08cdd6072e276e2e	f2b7e3ab19920d9c	fdb44a9e6f4bd7dc

RND	KEY	IV1	CIPHERTEXT2	IV2	CIPHERTEXT2	IV3	CIPHERTEXT3
14	49793ebc79b3258f	437540c8698f3cfa	6fbf1cafcffd0556	98ca961dbee4924f	0aa3768ad4358b6c	ee1feb731439e7a4	68b40c29c2238233
15	4fb05e1515ab73a7	072d43a077075292	2f22e49bab7ca1ac	5c8298f5cc5ca7e7	7fd1411fd6a31497	b1d7ee4b21b1fd3c	dd6359e601656be3
16	49e95d6d4ca229bf	02fe55778117f12a	5a6b612cc26cce4a	5853aaccd66d467f	116a6ae6e1e47270	ada900222bc29bd4	b16f4467a4f95fd0
17	018310dc409b26d6	1d9d5c5018f728c2	5f4c038ed12b2e41	72f2b1a56e4c7e17	de11d7e1c6d5797c	c84806fac3a1d36c	9cb7c0a87fa2bdbe
18	1c587f1c13924fef	305532286d6f295a	63fac0d034d9f793	85aa877dc2c47eaf	9896336cbadada37	daffdcd31819d404	1c5e61a81d05a5ef

Table A.11 Resulting Ciphertext from the Variable KEY Known Answer Test for TCBC-I, TCFB-P and TOFB-I Modes of Operation

(NOTE -- TEXT1 = TEXT2 = TEXT3 = 00 00 00 00 00 00 00 00

IV1 = 00 00 00 00 00 00 00 00

IV2 = 55 55 55 55 55 55 55 55

IV3 = aa aa aa aa aa aa aa aa)

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
0	8001010101010101	95a8d72813daa94d	b8bc8dbc0b24cfa9	1e08a515c11e0de1
1	4001010101010101	0eec1487dd8c26d5	badb3425df504209	0608b0c77f0ab511
2	2001010101010101	7ad16ffb79c45926	34069d06536cfaf8	3d090b850910022e
3	1001010101010101	d3746294ca6a6cf3	53edd6c7b2d8663c	19d83418eaf8e3ab
4	0801010101010101	809f5f873c1fd761	17d1d4a8731b3acd	91da457d7e16d6a5
5	0401010101010101	c02faffec989d1fc	51454c54f4ea817e	6a4ec92bc50c9503
6	0201010101010101	4615aa1d33e72f10	8f640c66e3ad6c5f	a185e92b67a45257
7	0180010101010101	2055123350c00858	e09a8dbe2b782986	0b7e13fdbadc96aa
8	0140010101010101	df3b99d6577397c8	6b1e20d1be1c25e5	eacef886f5087ce8
9	0120010101010101	31fe17369b5288c9	d7c9ed116a4ca5c3	69c60f1118060221
10	0110010101010101	dfdd3cc64dae1642	bb34b6ec92447bdc	99547b8b947e8c44

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
11	0108010101010101	178c83ce2b399d94	39ad35b103ea754c	aef4932bb880ffe7
12	0104010101010101	50f636324a9b7f80	502c48c0b6f5da1e	cd7942c2f0db9598
13	0102010101010101	a8468ee3bc18f06d	6da06bc26cd27347	b299efe073df56d0
14	0101800101010101	a2dc9e92fd3cde92	048b509f61329322	57fd7a94bd090076
15	0101400101010101	cac09f797d031287	cf18ef06ff4726dd	364898370f13783a
16	0101200101010101	90ba680b22aeb525	5e68a2a3f420ced2	7021fa3c611c5353
17	0101100101010101	ce7a24f350e280b6	f2241608a9c01443	4ad01e2a4f325e1b
18	0101080101010101	882bff0aa01a0b87	4d5268c568b57e87	d06a7e3c1016a256
19	0101040101010101	25610288924511c2	12537c78d5b135f5	af1c2074ea3952f7
20	0101020101010101	c71516c29c75d170	2a447d1d0918e635	643eacd845d0ac81
21	0101018001010101	5199c29a52c9f059	c45e53dbad3642c6	077f60d16feecc6d
22	0101014001010101	c22f0a294a71f29f	86b57a072d1af70c	2add0d3ff6b568ba
23	0101012001010101	ee371483714c02ea	3c6c5d0ad80d7409	0730787152b406bc
24	0101011001010101	a81fbd448f9e522f	3613b5811324cac7	ae3ef9ebdca26f00
25	0101010801010101	4f644c92e192dfed	50ed144cedb736ac	2abd3b256652632b
26	0101010401010101	1afa9a66a6df92ae	bc5bc5a66a53b929	a2e9fa40e6b6cfca

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
27	0101010201010101	b3c1cc715cb879d8	5d1f09ffcd80d21b	bd11881fa1f9c189
28	0101010180010101	19d032e64ab0bd8b	a8b79d2e02415d8e	925d1851ab04bafa
29	0101010140010101	3cfaa7a7dc8720dc	932c31352789dff9	4dafea6ad259c035
30	0101010120010101	b7265f7f447ac6f3	2ec8e9923a8a010c	e0f7a70dbdd597b7
31	0101010110010101	9db73b3c0d163f54	f36e475bb9a8fb57	88dad0c28986f116
32	0101010108010101	8181b65babf4a975	73f174b827a22fbf	205fd48356602a2f
33	0101010104010101	93c9b64042eaa240	c76d844d9918627d	ddaba956a4fd22c5
34	0101010102010101	5570530829705592	beff48907877eedd	775f3bbfea9a0637
35	0101010101800101	8638809e878787a0	7829e156fdd34db6	c26ea76714b38596
36	0101010101400101	41b9a79af79ac208	7b2545576a6992d9	46ca820bcf0a462b
37	0101010101200101	7a9be42f2009a892	0b59503dc812b27f	2a5e46fd70852d73
38	0101010101100101	29038d56ba6d2745	07b67fe9359a3026	145ad75857e4b4b3
39	0101010101080101	5495c6abf1e5df51	a82b120e4080136e	99525cafa664a0f9
40	0101010101040101	ae13dbd561488933	e3533571ee3d99eb	d1c679a7a2c4156c
41	0101010101020101	024d1ffa8904e389	eb57f8c58f18b849	e653401e4d004c74
42	0101010101018001	d1399712f99bf02e	505e3b0af188d731	02b8091c05f5e061

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
43	0101010101014001	14c1d7c1cffec79e	0f38a59e95a70f13	9879d116764dafe3
44	0101010101012001	1de5279dae3bed6f	97108885fe2018ed	154b6e3c9a2871b1
45	0101010101011001	e941a33f85501303	71147052540af3d8	21397c0ec6a47e75
46	0101010101010801	da99dbbc9a03f379	563df95ec668d933	d11d4e56261716a9
47	0101010101010401	b7fc92f91d8e92e9	c8003e219b996cc7	fb258b1abf89b7c4
48	0101010101010201	ae8e5caa3ca04e85	722fb450715fb317	c52f5e37f39d1e6f
49	0101010101010180	9cc62df43b6eed74	7edfaaa980158515	e91439e9838dcc9d
50	0101010101010140	d863dbb5c59a91a0	82fb07d5e1d5b100	78c2810a85028047
51	0101010101010120	a1ab2190545b91d7	04f0cbaff1735340	d466ec944a1fe7f7
52	0101010101010110	0875041e64c570f7	70ee1ae9b095db22	2fcd9094c8d397f2
53	0101010101010108	5a594528bebef1cc	004dd0b91a2e7709	80181b831cdc8d61
54	0101010101010104	fcdb3291de21f0c0	cab8e849e0ab0c32	3367b1fbb4d2ffa7
55	0101010101010102	869efd7f9f265a09	451f0c33f24fb8dc	2b74c1d96cde840b

Table A.12 Values To Be Used for the Permutation Operation Known Answer Test

for TCBC-I, TCFB-P and TOFB-I Modes of Operation

(NOTE -- TEXT1 = TEXT2 = TEXT3 = 00 00 00 00 00 00 00 00

IV1 = 00 00 00 00 00 00 00 00

IV2 = 55 55 55 55 55 55 55 55

IV3 = aa aa aa aa aa aa aa aa)

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
0	1046913489980131	88d55e54f54c97b4	23c25ab3e19b6b94	e5b490db69b0f2ec
1	1007103489988020	0c0cc00c83ea48fd	9e7b9f655eafef5d	2031be52988cd49e
2	10071034c8980120	83bc8ef3a6570183	948e0180ec95ab61	fcb4a56abf4b7b4e
3	1046103489988020	df725dcad94ea2e9	e97bb3b10db9f700	f627685cf879c481
4	1086911519190101	e652b53b550be8b0	df9e3ce144e6a0df	373a495e2a289a9e
5	1086911519580101	af527120c485cbb0	5fc7e5405519f6fb	5d8c63f84dc7b760
6	5107b01519580101	0f04ce393db926d5	4ce6c34fc99a7e47	43599c906eaa26af
7	1007b01519190101	c9f00ffc74079067	d59da3b97fa77d57	3ad69f58d64555fd
8	3107915498080101	7cfd82a593252b4e	2c90e8dcbfd28764	f5fec7cc3602fb9c
9	3107919498080101	cb49a2f9e91363e3	e3ef1da5cdfe2040	cbab42d154f3248c
10	10079115b9080140	00b588be70d23f56	ab256e068344f3d9	2957f7aec090659f

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
11	3107911598080140	406a9a6ab43399ae	142df8fbcdf06f6c	f3e52c8470bd4d49
12	1007d01589980101	6cb773611dca9ada	646449eb196edbc7	2c73895acb28e4d4
13	9107911589980101	67fd21c17dbb5d70	5bc918389c2a4f52	6d09d8d4450d34ef
14	9107d01589190101	9592cb4110430787	325e278ccb35a9b4	c67bed021618f6e8
15	1007d01598980120	a6b7ff68a318ddd3	bb2eaf9937470838	e45e7c5e8ba13dae
16	1007940498190101	4d102196c914ca16	a79acae80a89e1cf	73a5317d256ee9e6
17	0107910491190401	2dfa9f4573594965	70ce079b819d62a4	a6683459b9162215
18	0107910491190101	b46604816c0e0774	d40017b0499f3b3f	ef4c12c38fa94b67
19	0107940491190401	6e7e6221a4f34e87	484e191a8899dbd3	5bc2e500fd653804
20	19079210981a0101	aa85e74643233199	34ca696261a93635	d566849104e9f2f4
21	1007911998190801	2e5a19db4d1962d6	59a314314758d33c	fde57dae97810b56
22	10079119981a0801	23a866a809d30894	7782def75ae242b2	efaaba105ea97d41
23	1007921098190101	d812d961f017d320	e216e1e31589ec45	046bb3c67162342f
24	100791159819010b	055605816e58608f	75ecaecf73060451	e1729017bbdcfbd2
25	1004801598190101	abd88e8b1b7716f1	19dfcaebdf3f8958	ab3b5a50ebd4c354
26	1004801598190102	537ac95be69da1e1	16886a23bbb4cdba	353357f88bec120f
27	1004801598190108	aed0f6ae3c25cdd8	fc9e390a9093a7ac	8868a9829113d4a3

ROUND	KEY	C1/RESULT1	C2/RESULT2	C3/RESULT3
28	1002911598100104	b3e35a5ee53e7b8d	13685e1b83c61eef	0ec122be6dc26c83
29	1002911598190104	61c79c71921a2ef8	1d19adde7fb74e34	9792ca21f5adbce6
30	1002911598100201	e2f5728f0995013c	1423db30c7e118fb	e5f2d4dd2f43d9d1
31	1002911698100101	1aeac39a61f0a464	31eed52fa33c013d	dcf4548cf2374875

REFERENCES

[1] Triple Data Encryption Algorithm Modes of Operation, ANSI X9.52 - 1998.

[2] Modes of Operation Validation System (MOVS): Requirements and Procedures, NIST Special Publication 800-17, 1998.

[3] Data Encryption Standard (DES), FIPS PUB 46-3 - 1999.