

NIST Special Publication 800-67  
Version 1.2

**NIST**

**National Institute of  
Standards and Technology**

Technology Administration  
U.S. Department of Commerce

# Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher

**William C. Barker**

## I N F O R M A T I O N   S E C U R I T Y

Computer Security Division  
Information Technology Laboratory  
National Institute of Standards and Technology  
Gaithersburg, MD 20899-8930

*March 2004*



**U.S. DEPARTMENT OF COMMERCE**

*Donald L. Evans, Secretary*

**TECHNOLOGY ADMINISTRATION**

*Phillip J. Bond, Under Secretary of Commerce for Technology*

**NATIONAL INSTITUTE OF STANDARDS AND  
TECHNOLOGY**

*Arden L. Bement, Jr., Director*

## **REPORTS ON COMPUTER SYSTEMS TECHNOLOGY**

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of non-national security-related information in Federal information systems. This special publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

## Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided A-130, Appendix III.

This recommendation has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should this recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

**National Institute of Standards and Technology, Draft Special Publication 800-67  
Natl. Inst. Stand. Technol. Spec. Publ. 800-67, 40 pages (March 2004)**

THE PUBLIC COMMENT PERIOD FOR THIS DOCUMENT BEGINS ON MARCH 8, 2004  
AND ENDS ON MAY 6, 2004. COMMENTS MAY BE SUBMITTED TO THE COMPUTER  
SECURITY DIVISION, NIST, VIA ELECTRONIC MAIL AT [WBARKER@NIST.GOV](mailto:WBARKER@NIST.GOV)

OR VIA REGULAR MAIL AT

100 BUREAU DRIVE (MAIL STOP 8930)  
GAITHERSBURG, MD 20899-8930

### **Acknowledgements**

The author wishes to thank his colleagues who reviewed drafts of this document and contributed to its development. The author also gratefully acknowledges and appreciates the many comments from the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

## **Abstract**

The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its electronic data systems. This publication specifies the Triple Data Encryption Algorithm (TDEA), including its primary component cryptographic engine, the Data Encryption Algorithm (DEA). When implemented in an SP 800-38 series-compliant mode of operation and in a FIPS 140-2 compliant cryptographic module, TDEA may be used by Federal organizations to protect sensitive unclassified data. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data. This recommendation precisely defines the mathematical steps required to cryptographically protect data using TDEA and to subsequently process such protected data. The Triple Data Encryption Algorithm (TDEA) is made available for use by Federal agencies within the context of a total security program consisting of physical security procedures, good information management practices, and computer system/network access controls.

Key words: block cipher, computer security, cryptography, data encryption algorithm, security, triple data encryption algorithm.

## FOREWORD

The Triple Data Encryption Algorithm (TDEA) is an approved cryptographic algorithm as required by FIPS 140-2, *Security Requirements for Cryptographic Modules*. TDEA specifies both the DEA cryptographic engine employed by TDEA and the TDEA algorithm itself.

This recommendation provides a description of a mathematical algorithm for cryptographically protecting binary coded information (e.g., using encryption and authentication). The algorithm described in this recommendation specifies cryptographic operations that are based on a binary number called a key.

Authorized users of computer data cryptographically protected using TDEA must have the key that was used to protect the data in order to process the protected data. The cryptographic algorithm specified in this recommendation is assumed to be commonly known among its users. The cryptographic security of the data depends on the security provided for the key used to protect the data.

Data that is determined by a responsible authority to be sensitive, data that has a high value, or data that represents a high value should be cryptographically protected if it is vulnerable to unauthorized disclosure or undetected modification during transmission or while in storage. A risk analysis should be performed under the direction of a responsible authority to determine potential threats. The costs of providing cryptographic protection using this recommendation, as well as of alternative methods for providing this protection, should be projected. A responsible authority then should make a decision, based on these analyses, whether or not to use cryptographic protection and this recommendation.

The Data Encryption Standard became effective July 1977. It was reaffirmed in 1983, 1988, 1993, and 1999. The DES has now been withdrawn. The use of DES is permitted only as a component function of TDEA. This recommendation applies to all Federal agencies, contractors of Federal agencies, or other organizations that process information (using a computer or telecommunications system) on behalf of the Federal Government to accomplish a Federal function. Each Federal agency or department may issue internal directives for the use of this recommendation by their operating units based on their data security requirement determinations.

With the withdrawal of the FIPS 46-3 standard:

1. Triple DES (i.e., TDEA), as specified in this Recommendation is recognized as the only FIPS approved use of the DES algorithm.
2. Other implementations of the DES function are no longer authorized for protection of Federal government information.

Note: Through the year 2030<sup>1</sup>, Triple DES (TDEA) and the FIPS 197 Advanced Encryption Standard (AES) will coexist as FIPS approved algorithms – thus, allowing for a gradual transition to AES. (The AES is a new symmetric based encryption standard approved by NIST.)

With regard to the prohibition against use of ANSI X9.52 TDEA Keying Option 3 (key bundle composed of three identical 64-bit keys), exhaustion of the DEA cryptographic engine (i.e., breaking a DEA encrypted ciphertext by trying all possible keys) has become increasingly feasible as available computing power has grown. Following the development of practical hardware based DEA key exhaustion attacks, NIST can no longer support the use of DES implementations based on a single 64-bit key for Federal government applications. Therefore, Government agencies with legacy single DES systems and TDEA with Keying Option 3 are required to transition to FIPS-approved versions of Triple DES (TDEA) or AES. Agencies are advised to implement only FIPS-approved versions of Triple DES.

Cryptographic devices implementing this recommendation may be covered by U.S. and foreign patents, including patents issued to the International Business Machines Corporation. However, IBM has granted nonexclusive, royalty-free licenses under the patents to make, use, and sell apparatus which complies with the recommendation. The terms, conditions and scope of the licenses are set out in notices published in the May 13, 1975 and August 31, 1976 issues of the Official Gazette of the United States Patent and Trademark Office (934 O.G. 452 and 949 O.G. 1717).

---

<sup>1</sup> Recommendations regarding use of Option 2 are contained in SP 800-57, Part 1.

## TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1 Basis .....	1
1.2 Applicability.....	2
1.3 Applications .....	2
1.4 Implementations .....	3
1.5 Alternative Modes of Using the TDEA.....	3
1.6 Organization.....	4
<b>2. DATA ENCRYPTION ALGORITHM CRYPTOGRAPHIC ENGINE .....</b>	<b>5</b>
2.1 DEA Forward Transformation.....	5
2.2 DEA Inverse Transformation.....	8
2.3 The Function $f$ .....	9
<b>3. TRIPLE DATA ENCRYPTION ALGORITHM .....</b>	<b>13</b>
3.1 Basic TDEA Forward and Inverse Cipher Operations.....	13
3.2 TDEA Keying Options.....	13
3.3 TDEA Modes of Operation.....	13
3.4 Keys.....	14
3.4.1 Key Requirements.....	14
3.4.2 Weak Keys .....	14
3.5 Usage Guidance .....	15
<b>APPENDIX A: PRIMITIVE FUNCTIONS FOR THE DATA ENCRYPTION ALGORITHM .....</b>	<b>17</b>
<b>APPENDIX B: EXAMPLE OF TDEA FORWARD AND INVERSE CIPHER OPERATIONS .....</b>	<b>23</b>
B.1 TDEA Block Cipher Encryption - ECB Mode .....	23
B.2 TDEA Block Cipher Decryption - ECB Mode .....	24



**APPENDIX C: GLOSSARY**.....27

**APPENDIX D: REFERENCES** .....29

Initial Public Draft

**[This page intentionally left blank.]**

**National Institute of Standards and Technology  
Special Publication 800-67**

**2004 March 8**

**SPECIFICATIONS FOR THE  
TRIPLE DATA ENCRYPTION ALGORITHM (TDEA) BLOCK CIPHER**

## **1. INTRODUCTION**

This recommendation specifies the Triple Data Encryption Algorithm (TDEA) block cipher. The TDEA block cipher includes a Data Encryption Algorithm (DEA) cryptographic engine implemented as a component of TDEA as specified in Section 3. TDEA functions incorporating the DEA cryptographic engine **shall** be designed in such a way that they may be used in a computer system, storage facility, or network to provide cryptographic protection to binary coded data. The method of implementation will depend on the application and environment. TDEA implementations **shall** be subject to being tested and validated as accurately performing the transformations specified in the TDEA algorithm and in NIST Special Publication 800-38, Recommendation for Block Cipher Modes of Operation - Methods and Techniques.

### **1.1 Basis**

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided A-130, Appendix III.

This recommendation has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this recommendation should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should this recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Conformance testing of implementations of the block cipher that is specified in this recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of the NIST and the Communications Security Establishment of the Government of Canada. An implementation of the TDEA block cipher must adhere to the requirements of this recommendation in order to be validated under the CMVP. Requirements and procedures for validation of TDEA modes can be found in Special Publication 800-20, *Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures*.

## 1.2 Applicability

This recommendation may be used by Federal departments and agencies when the following conditions apply:

1. An authorized official or manager responsible for data security or the security of any computer system decides that cryptographic protection is required; and
2. The data is not classified according to the Federal Information Systems Management Act of 2002, the National Security Act of 1947, as amended, or the Atomic Energy Act of 1954, as amended.

Other FIPS approved cryptographic algorithms may be used in addition to, or in lieu of, TDEA when implemented in accordance with FIPS 140-2.

Federal agencies or departments that use cryptographic devices for protecting data classified according to either of these acts can use those devices for protecting sensitive data in lieu of TDEA or other FIPS approved cryptographic algorithms.

In addition, this recommendation may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations and/or is necessary for interoperability with cryptographically protected systems.

## 1.3 Applications

Cryptography is utilized in various applications and environments. The specific utilization of encryption and the implementation of TDEA<sup>3</sup> will be based on many factors particular to the computer system and its associated components. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft or technical intrusion (e.g., hacker attacks). In the first case, the key must be available at the transmitter and receiver simultaneously during communication. In the second case, the key must be maintained and accessible for the duration of the storage period. NIST Special Publications 800-56, *Key Management Schemes Specification*, and 800-57, *Key*

---

<sup>3</sup> And the cryptographic engine that forms the basis for TDEA.

*Management Guideline*, provide recommendations for managing cryptographic keys, including the keys used by the algorithm specified in this recommendation.

## 1.4 Implementations

Cryptographic modules that implement this recommendation **shall** conform to the requirements of FIPS 140-2 and NIST Special Publication 800-20, *Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures*. The algorithm specified in this recommendation may be implemented in software, firmware, hardware, or any combination thereof. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc. Implementations that may comply with this recommendation include electronic devices (e.g., VLSI chip packages), microprocessors using Read Only Memory (ROM), Programmable Read Only Memory (PROM), or Electronically Erasable Read Only Memory (EEROM), and computers<sup>4</sup> using Random Access Memory (RAM). When an algorithm is implemented in software or firmware, the processor on which the algorithm runs must be specified as part of the validation process. Implementations of TDEA that are tested and validated in conformance to NIST standards and guidelines will be considered as complying with the recommendation. Note that FIPS 140-2 places additional requirements on cryptographic modules for Government use. Information about devices that have been validated and procedures for testing and validating equipment for conformance with this recommendation and FIPS 140-2 are available from the National Institute of Standards and Technology, Information Technology Laboratory, 100 Bureau Dr. Stop 8930, Gaithersburg, MD 20899-8930 or at <http://csrc.nist.gov/cryptval>.

The successful completion of the tests contained within the TMOVS is required to claim conformance of Triple DES implementations to this recommendation and the modes in SP 800-38. The validation system consists of validation tests for the DEA cryptographic engine and the TDEA block cipher's modes of operation. The algorithm validation procedures for TDEA are outlined in NIST Special Publication 800-20. This can be found on the CMVP website (<http://csrc.nist.gov/cryptval>).

## 1.5 Alternative Modes of Using the TDEA

NIST Special Publication 800-38, *Recommendation for Block Cipher Modes Methods and Techniques*, describes modes of operation for the TDEA block cipher described in this recommendation. These modes of operation are approved for the protection of Federal government information.

---

<sup>4</sup> Including mainframes, servers, and personal computers (including desk top, laptop, and other mobile computer implementations).

## **1.6 Organization**

Section 2 of this recommendation describes the DEA cryptographic engine employed by TDEA.

Section 3 of the recommendation describes the basic TDEA algorithm.

Appendices are provided for DEA primitives, examples of encryption and decryption using the TDEA block cipher operation, a glossary of terms, and a list of references.

## 2. DATA ENCRYPTION ALGORITHM CRYPTOGRAPHIC ENGINE

The DEA cryptographic engine is used by TDEA to cryptographically protect blocks of data consisting of 64 bits under the control of a 64-bit key<sup>5</sup>. Subsequent processing of the protected data is accomplished using the same key as was used to protect the data. Each 64-bit key **shall** contain 56 bits that are randomly generated and used directly by the algorithm as key bits. The other eight bits, which are not used by the algorithm, may be used for error detection. The eight error detecting bits are set to make the parity of each 8-bit byte of the key odd. That is, there are an odd number of "1"s in each 8-bit byte<sup>6</sup>.

During each application of the DEA engine, a block is subjected to an initial permutation **IP**, then to a complex key-dependent computation and finally to a permutation that is the inverse of the initial permutation **IP**<sup>-1</sup>. The key-dependent computation can be simply defined in terms of a function **f** and a function **KS**, called the key schedule. The DEA engine can be run in two directions - as a forward transformation and as an inverse transformation. The two directions differ only by the order in which the bits of the key are used.

A description of the forward and inverse transformations are provided below, followed by a definition of the function **f** in terms of primitive functions called by the selection functions **S<sub>i</sub>**, and the permutation function **P**. Values for **S<sub>i</sub>**, **P** and **KS** of the engine are contained in Appendix A.

The following notation is convenient: Given two blocks **L** and **R** of bits, **LR** denotes the block consisting of the bits of **L** followed by the bits of **R**. Since concatenation is associative, **B<sub>1</sub>B<sub>2</sub>...B<sub>8</sub>**, for example, denotes the block consisting of the bits of byte **B<sub>1</sub>** followed by the bits of byte **B<sub>2</sub>**...followed by the bits of byte **B<sub>8</sub>**.

### 2.1 DEA Forward Transformation

A sketch of the forward transformation is given in **Figure 1**.

---

<sup>5</sup> Blocks are composed of bits numbered from left to right, i.e., the left most bit of a block is bit one.

<sup>6</sup> Sometimes keys are generated in an encrypted form. A random 64-bit number is generated and defined to be the cipher formed by the encryption of a key using a key encrypting key. In this case the parity bits of the encrypted key cannot be set until after the key is decrypted.

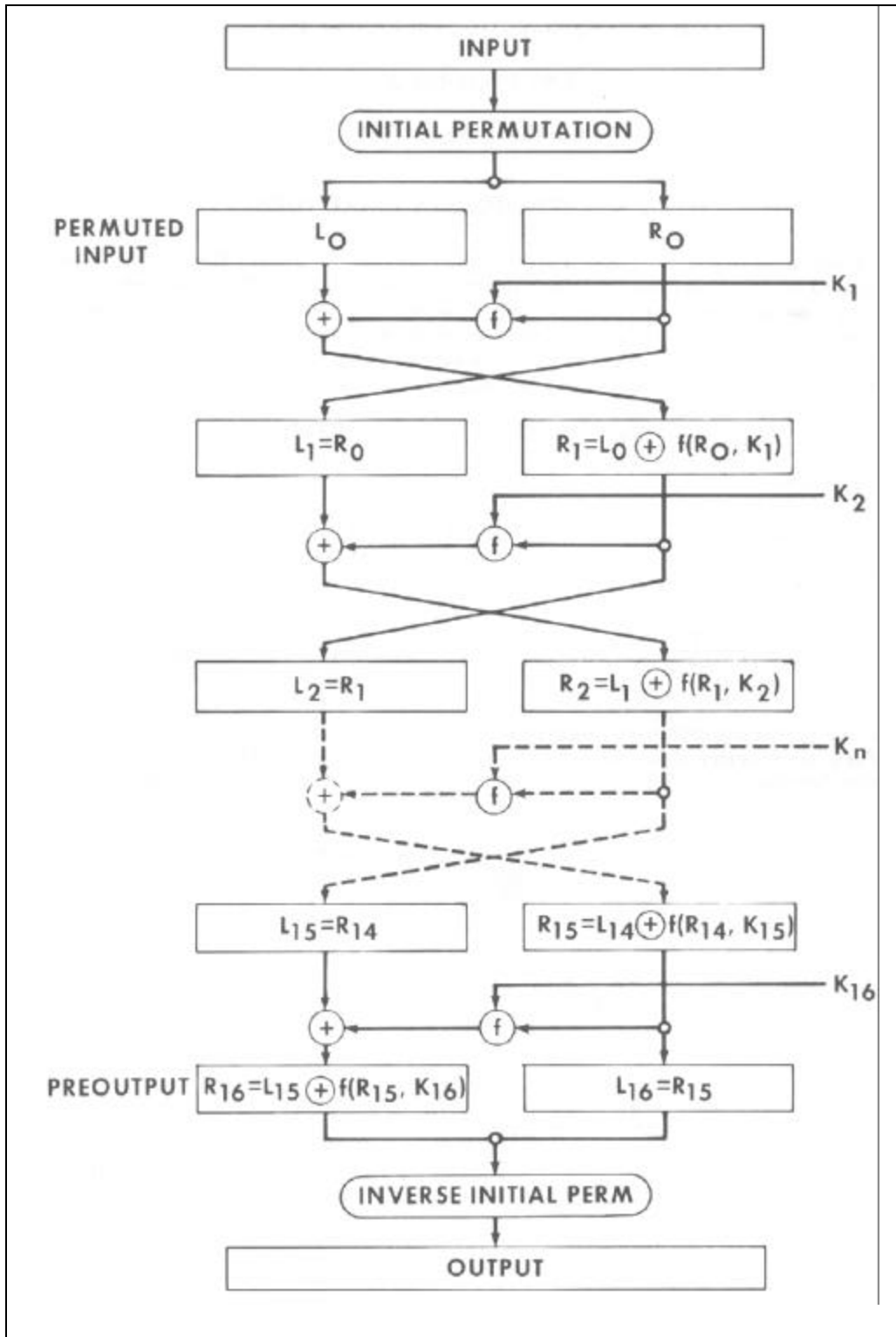


Figure 1. Forward Transformation of the DEA Cryptographic Engine



The 64 bits of the input block for the forward transformation are first subjected to the following permutation, called the initial permutation  $IP$ :

<u><math>IP</math></u>							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

That is, the permuted input has bit 58 of the input as its first bit, bit 50 as its second bit, and so on, with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation that is described below. The output of that computation, called the preoutput, is then subjected to the following permutation that is the inverse of the initial permutation:

<u><math>IP^{-1}</math></u>							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

That is, the output of the algorithm has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

The key-dependent computation that uses the permuted input block as its input to produce the preoutput block consists, except for a final interchange of blocks, of 16 iterations of a calculation that is described below in terms of the function  $f$ . This function operates on two blocks, one of 32 bits and one of 48 bits, to produce a block of 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32 bit block  $L$ , followed by a 32 bit block  $R$ . Using the notation defined above, the input block is then  $LR$ .

Let  $K$  be a block of 48 bits chosen from the 64-bit key. Then the output  $L'R'$  of an iteration with input  $LR$  is defined by:

$$(1) \quad \begin{aligned} L' &= R \\ R' &= L \oplus f(R, K) \end{aligned}$$

where  $\oplus$  denotes bit-by-bit addition modulo 2 (also known as exclusive-or or XOR).

As remarked before, the input of the first iteration of the calculation is the permuted input block. If  $L'R'$  is the output of the 16th iteration, then  $R'L'$  is the preoutput block. At each iteration, a different block  $K$  of key bits is chosen from the 64-bit key designated by  $KEY$ .

With more notation, the iterations of the computation can be described in more detail. Let  $KS$  be a function that takes an integer  $n$  in the range from 1 to 16 and a 64-bit block  $KEY$  as input. The output of  $KS$  is a 48-bit block  $K_n$  that is a permuted selection of bits from  $KEY$ . That is:

$$(2) \quad K_n = KS(n, KEY)$$

with  $K_n$  determined by the bits in 48 distinct bit positions of  $KEY$ .  $KS$  is called the key schedule because the block  $K$  used in the  $n$ 'th iteration of (1) is the block  $K_n$  determined by (2).

As before, let the permuted input block be  $LR$ . Finally, let  $L_{()}$  and  $R_{()}$  be respectively  $L$  and  $R$ , and let  $L_n$  and  $R_n$  be respectively  $L'$  and  $R'$  of (1) when  $L$  and  $R$  are respectively  $L_{n-1}$  and  $R_{n-1}$ , and  $K$  is  $K_n$ ; that is, when  $n$  is in the range from 1 to 16,

$$(3) \quad \begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \oplus f(R_{n-1}, K_n) \end{aligned}$$

The preoutput block is then  $R_{16}L_{16}$ .

The key schedule  $KS$  of the algorithm is described in detail in Appendix A. The key schedule produces the 16  $K_n$  that are required for the algorithm.

## 2.2 DEA Inverse Transformation

The permutation  $IP^{-1}$  applied to the preoutput block is the inverse of the initial permutation  $IP$  applied to the input. Further, from (1) it follows that:

$$(4) \quad \begin{aligned} R &= L' \\ L &= R' \oplus f(L', K) \end{aligned}$$

Consequently, to apply the inverse transformation, it is only necessary to apply the *very same algorithm to a block of the protected data produced by the forward transformation*, taking care that at each iteration of the computation, *the same block of key bits  $K$  is used during the inverse transformation as was used during the forward transformation*. This requires using the key schedule in reverse order. That is, for the forward transformation,  $K_1$  is used as the first 48 bit block, and  $K_{16}$  is used as the last block, whereas for the inverse transformation,  $K_{16}$  is used as the first block, and  $K_1$  is used as the last block.

Using the notation of the previous section, this can be expressed by the equations:

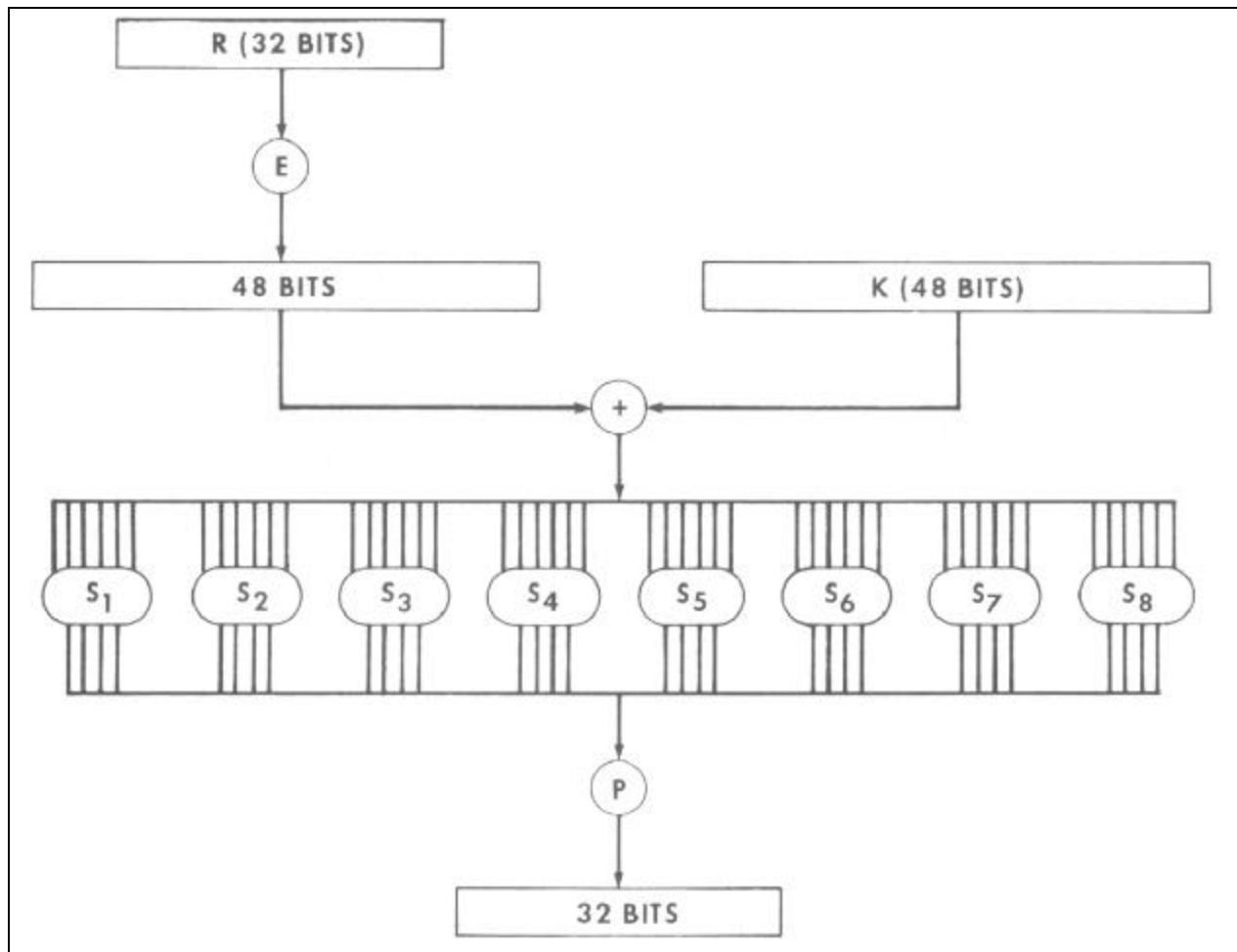
$$(5) \quad \begin{aligned} R_{n-1} &= L_n \\ L_{n-1} &= R_n \oplus f(L_n, K_n) \end{aligned}$$

where  $R_{16}L_{16}$  is the permuted input block for the inverse transformation, and  $L_0R_0$  is the preoutput block. That is, for the inverse transformation with  $R_{16}L_{16}$  as the permuted input,  $K_{16}$  is used in the first iteration,  $K_{15}$  in the second, and so on, with  $K_1$  used in the 16th iteration.

### 2.3 The Function $f$

A sketch of the calculation of  $f(R, K)$  is given in **Figure 2**.

Let  $E$  denote a function, which takes a block of 32 bits as input and yields a block of 48 bits as output. Let  $E$  be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to Table 1:



**Figure 2.** Calculation of  $f(R, K)$

**Table 1:  $E$  BIT-SELECTION TABLE**

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Thus, the first three bits of  $E(\mathbf{R})$  are the bits in positions 32, 1 and 2 of  $\mathbf{R}$ , while the last 2 bits of  $E(\mathbf{R})$  are the bits in positions 32 and 1.

Each of the unique selection functions  $S_1, S_2, \dots, S_8$ , takes a 6-bit block as input and yields a 4-bit block as output and is illustrated by using Table 2. Table 2 contains  $S_1$ :

**Table 2:  $S_1$** 

	Column Number															
Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

If  $S_1$  is the function defined in this table, and  $\mathbf{B}$  is a block of 6 bits, then  $S_1(\mathbf{B})$  is determined as follows: The first and last bits of  $\mathbf{B}$  represent, in base 2, a number in the range 0 to 3. Let that number be  $i$ . The middle 4 bits of  $\mathbf{B}$  represent, in base 2, a number in the range 0 to 15. Let that number be  $j$ . Using the table, look up the number in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column. It is a number in the range 0 to 15 and is uniquely represented by a 4-bit block. That block is the output  $S_1(\mathbf{B})$  of  $S_1$  for the input  $\mathbf{B}$ . For example, for input 011011 the row is 01 (i.e., row 1), and the column is determined by 1101 (i.e., column 13). The number 5 appears in row 1, column 13, so the output is 0101.

Selection functions  $S_1, S_2, \dots, S_8$  of the algorithm appear in Appendix A.

The permutation function  $P$  yields a 32-bit output from a 32-bit input by permuting the bits of the input block. This function is defined by Table 3:

**Table 3:  $P$** 

16	7	0	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

The output  $P(L)$  for the function  $P$  defined by this table is obtained from the input  $L$  by taking the 16th bit of  $L$  as the first bit of  $P(L)$ , the 7th bit as the second bit of  $P(L)$ , and so on until the 25th bit of  $L$  is taken as the 32nd bit of  $P(L)$ . The permutation function  $P$  of the algorithm is repeated in Appendix A.

Now let  $S_1, \dots, S_8$  be eight distinct selection functions, let  $P$  be the permutation function, and let  $E$  be the function defined above.

To define  $f(R, K)$ , let  $B_1, \dots, B_8$  be blocks of 6 bits each for which

$$(6) \quad B_1 B_2 \dots B_8 = K \oplus E(R)$$

The block  $f(R, K)$  is then defined to be

$$(7) \quad P(S_1(B_1)S_2(B_2)\dots S_8(B_8))$$

Thus,  $K \oplus E(R)$  is first divided into the 8 blocks as indicated in (6). Then each  $B_i$  is taken as an input to  $S_i$ , and the 8 blocks  $S_1(B_1), S_2(B_2), \dots, S_8(B_8)$  of 4 bits each are consolidated into a single block of 32 bits, which forms the input to  $P$ . The output (7) is then the output of the function  $f$  for the inputs  $R$  and  $K$ .

[This page intentionally left blank.]

### 3. TRIPLE DATA ENCRYPTION ALGORITHM

#### 3.1 Basic TDEA Forward and Inverse Cipher Operations

In this recommendation, each TDEA forward and inverse cipher operation is a compound operation of DEA forward and inverse transformations as specified in Section 2.

A TDEA key consists of three keys for the cryptographic engine (**Key**<sub>1</sub>, **Key**<sub>2</sub> and **Key**<sub>3</sub>); the three keys are also referred to as a key bundle (**KEY**). Two options for the selection of the keys in a key bundle are allowed. Option 1, the preferred option, employs three mutually independent keys (i.e. **Key**<sub>1</sub>, **Key**<sub>2</sub> and **Key**<sub>3</sub>, where **Key**<sub>1</sub> ≠ **Key**<sub>2</sub> ≠ **Key**<sub>3</sub> ≠ **Key**<sub>1</sub>). Option 2 employs two mutually independent keys and a third key that is the same as the first key (i.e. **Key**<sub>1</sub>, **Key**<sub>2</sub> and **Key**<sub>3</sub>, where **Key**<sub>1</sub> ≠ **Key**<sub>2</sub> and **Key**<sub>3</sub> = **Key**<sub>1</sub>). A key bundle **shall not** consist of three identical keys.

Let  $F_{KeyX}(d)$  and  $I_{KeyY}(d)$ , respectively, represent the DEA forward and inverse transformations on data  $d$  using **Key** <sub>$x$</sub>  and **Key** <sub>$y$</sub> , respectively. The following operations are used:

1. TDEA forward cipher operation: the transformation of a 64-bit block  $d$  into a 64-bit block  $O$  that is defined as follows:

$$O = F_{Key3}(I_{Key2}(F_{Key1}(d))).$$

2. TDEA inverse cipher operation: the transformation of a 64-bit block  $d$  into a 64-bit block  $O$  that is defined as follows:

$$O = I_{Key1}(F_{Key2}(I_{Key3}(d))).$$

#### 3.2 TDEA Keying Options

This recommendation specifies the following keying options for a TDEA key bundle (**Key**<sub>1</sub>, **Key**<sub>2</sub>, **Key**<sub>3</sub>)

1. Keying Option 1: **Key**<sub>1</sub>, **Key**<sub>2</sub> and **Key**<sub>3</sub> are independent keys (i.e., **Key**<sub>1</sub> ≠ **Key**<sub>2</sub> ≠ **Key**<sub>3</sub> ≠ **Key**<sub>1</sub>);
2. Keying Option 2: **K**<sub>1</sub> and **K**<sub>2</sub> are independent keys (i.e., **Key**<sub>1</sub> ≠ **Key**<sub>2</sub>), and **Key**<sub>3</sub> = **Key**<sub>1</sub>.

#### 3.3 TDEA Modes of Operation

TDEA **shall** be implemented using one or more of the modes of operation specified in NIST Special Publication 800-38 (SP 800-38), *Recommendation for Block Cipher Modes of Operation Methods and Techniques*. These modes of operation are approved for the protection of Federal government sensitive, but unclassified information. Each of the modes employs the TDEA forward or inverse ciphers as defined in Section 3.1. Note that the TDEA block cipher **shall** be used to provide cryptographic security only when used in an approved mode of operation.

### 3.4 Keys

The TDEA keys **shall** be managed in accordance with NIST Special Publication (SP) 800-57, *Key Management Guidelines*. SP 800-57 also specifies time frames during which the TDEA keying options may be used. The following specifications for keys **shall** be met in implementing the TDEA modes of operation.

#### 3.4.1 Key Requirements

For all TDEA modes of operation, three cryptographic keys (*Key*<sub>1</sub>, *Key*<sub>2</sub>, *Key*<sub>3</sub>) define a TDEA key bundle. The bundle and the individual keys must:

- a. be secret;
- b. be generated randomly or pseudorandomly (See [ANSI 9.82]);
- c. be independent of other key bundles;
- d. have integrity whereby each key in the bundle has not been altered in an unauthorized manner since the time it was generated, transmitted, or stored by an authorized source;
- e. be used in the appropriate order as specified by the particular mode;
- f. be considered a fixed quantity in which an individual key cannot be manipulated while leaving the other two keys unchanged; and cannot be unbundled except for its designated purpose.

#### 3.4.2 Weak Keys

There are a few keys that are considered weak for the DEA cryptographic engine. The use of weak keys can reduce the effective security afforded by TDEA and should be avoided. Keys that are considered weak are (in hexadecimal format):

- 0000000 0000000
- 0000000 FFFFFFFF
- FFFFFFFF 0000000
- FFFFFFFF FFFFFFFF

Some pairs of keys encrypt plaintext to identical ciphertext. These semi-weak keys are (in hex):

- 011F011F010E010E and 1F011F010E010E01
- 01E001E001F101F1 and E001E001F101F101



- 01FE01FE01FE01FE and FE01FE01FE01FE01
- 1FE01FE00EF10EF1 and E01FE01FF10EF10E
- 1FFE1FFE0EFE0EFE and FE1FFE1FFE0EFE0E
- E0FEE0FEF1FEF1FE and FEE0FEE0FEF1FEF1

There are also 48 keys that produce only four distinct subkeys (instead of 16) - these are called possibly-weak keys. These possibly-weak keys are (in hex):

01011F1F01010E0E	1F1F01010E0E0101	E0E01F1FF1F10E0E
0101E0E00101F1F1	1F1FE0E00E0EF1F1	E0E0FEFEF1F1FEFE
0101FEFE0101FEFE	1F1FFEFE0E0EFEFE	E0FE011FF1FE010E
011F1F01010E0E01	1FE001FE0EF101FE	E0FE1F01F1FE0E01
011FE0FE010EF1FE	1FE0E01F0EF1F10E	E0FEFEE0F1FEFEF1
011FFEE0010EFEF1	1FE0FE010EF1FE01	FE0101FEFE0101FE
01E01FFE01F10EFE	1FFE01E00EFE01F1	FE011FE0FE010EF1
01E01FFE01F1F10E	1FFEE0010EFEF001	FE1F01E0FE0E01F1
01E0E00101F1F101	1FFEFE1F0EFEFE0E	FE1FE001FE0EF101
01E0FE1F01F1FE0E	E00101E0F10101F1	FE1F1FFEFE0E0EFE
01FE1FE001FE0EF1	E0011FFE1F1010EFE	FEE0011FFE1F1010E
01FEE01F01FEF10E	E001FE1FF101FE0E	FEE01F01FEF10E01
01FEFE0101FEFE01	E01F01FEF10E01FE	FEE0E0FEFEF1F1FE
1F01011F0E01010E	E01F1FE0F10E0EF1	FEFE0101FEFE0101
1F01E0FE0E01F1FE	E01FFE01F10EFE01	FEFE1F1FFEFE0E0E
1F01FEE00E01FEF1	E0E00101F1F10101	FEFEE0E0FEFEF1F1

### 3.5 Usage Guidance

The security of TDEA is affected by the number of blocks processed with one key bundle. One key bundle should not be used to process more than  $2^{32}$  64-bit data blocks.

[This page intentionally left blank.]

## APPENDIX A: PRIMITIVE FUNCTIONS FOR THE DATA ENCRYPTION ALGORITHM

The choice of the primitive functions  $KS$ ,  $S_1, \dots, S_8$  and  $P$  is critical to the strength of the transformations resulting from the algorithm. The tables below specify the functions  $S_1, \dots, S_8$  and  $P$ . For the interpretation of the tables describing these functions, see the discussion in Section 2.

The primitive functions  $S_1, \dots, S_8$  are:

### $S_1$

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

### $S_2$

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

### $S_3$

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

### $S_4$

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

### $S_5$

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

$S_6$ 

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

 $S_7$ 

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

 $S_8$ 

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The primitive function  $P$  is:

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Recall that  $K_n$ , for  $1 \leq n \leq 16$ , is the block of 48 bits in (2) of the algorithm. Hence, to describe  $KS$ , it is sufficient to describe the calculation of  $K_n$  from a key ( $Key_i$ ) from the key bundle for  $n = 1, 2, \dots, 16$ . That calculation is illustrated in Figure 4. To complete the definition of  $KS$ , it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts. One bit in each 8-bit byte of  $Key_i$  may be utilized for error detection in key generation, distribution and storage. Bits 8, 16, ..., 64 are for use in assuring that each byte is of odd parity. [Note that these eight parity bits have no effect on the operation of the algorithm.]

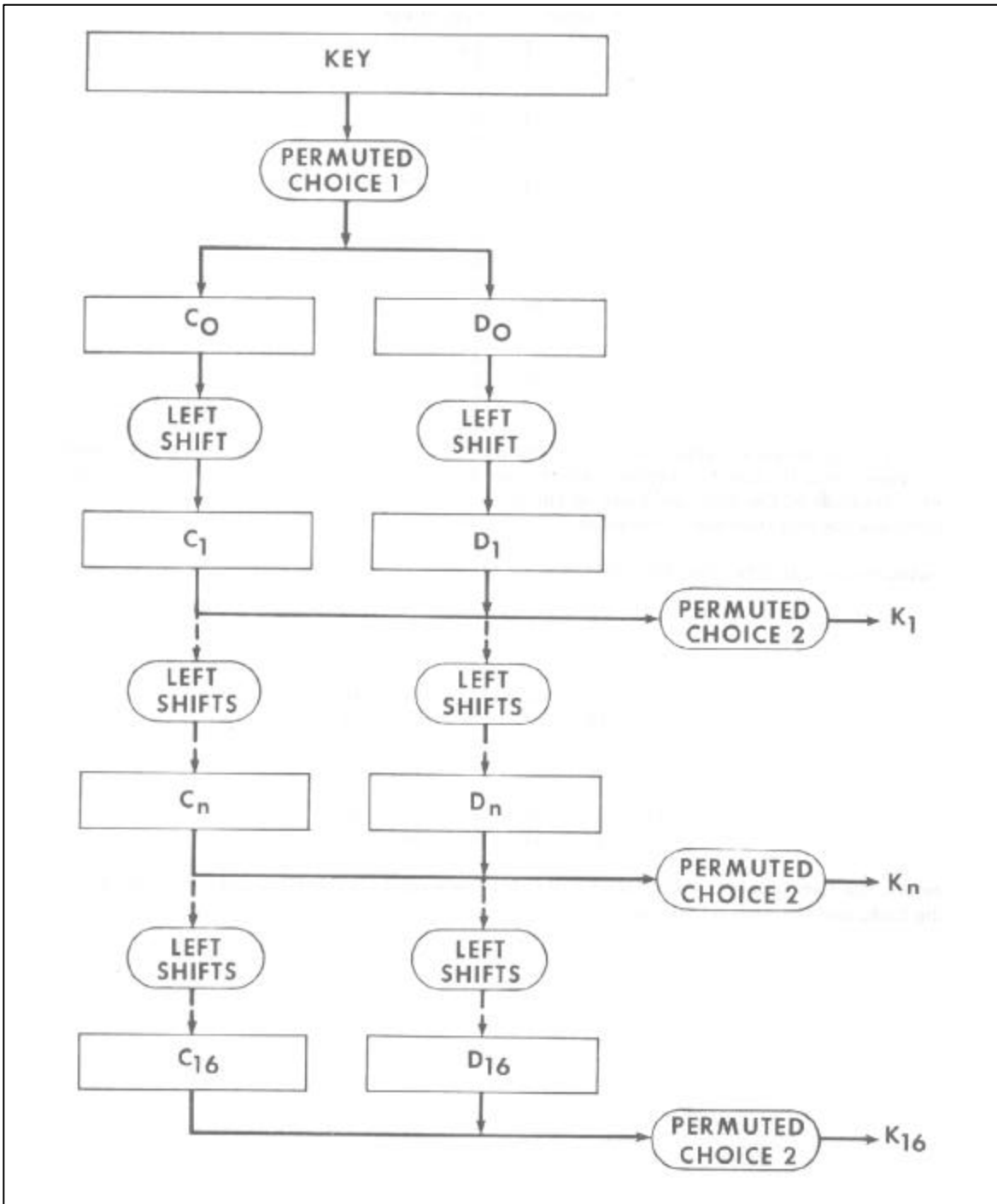


Figure 4: Key Schedule Calculation

Permuted choice 1 is determined by the following table:

<u><b>PC-1</b></u>						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

The table has been divided into two parts, with the first part determining how the bits of  $C_{( )}$  are chosen, and the second part determining how the bits of  $D_{( )}$  are chosen. The bits of  $Key_i$  are numbered 1 through 64. The bits of  $C_{( )}$  are respectively bits 57, 49, 41, ..., 44 and 36 of  $Key_i$ , with the bits of  $D_{( )}$  being bits 63, 55, 47, ..., 12 and 4 of  $Key_i$ .

With  $C_{( )}$  and  $D_{( )}$  defined, the blocks  $C_n$  and  $D_n$  are obtained from the blocks  $C_{n-1}$  and  $D_{n-1}$ , respectively, for  $n = 1, 2, \dots, 16$ , by adhering to the following schedule of left shifts of the individual blocks:

Iteration <i>Number</i>	Number of <i>Left Shifts</i>
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

For example,  $C_3$  and  $D_3$  are obtained from  $C_2$  and  $D_2$ , respectively, by two left shifts, and  $C_{16}$  and  $D_{16}$  are obtained from  $C_{15}$  and  $D_{15}$ , respectively, by one left shift. In all cases, by a single left

shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

Permuted choice 2 is determined by the following table:

**PC-2**

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Therefore, the first bit of  $K_n$  is the 14<sup>th</sup> bit of  $C_nD_n$ , the second bit of  $K_n$  is the 17<sup>th</sup> bit of  $C_nD_n$ , and so on, with the 47<sup>th</sup> bit of  $K_n$  as the 29<sup>th</sup> bit of  $C_nD_n$ , and the 48<sup>th</sup> bit of  $K_n$  as the 32<sup>nd</sup> bit of  $C_nD_n$ .

[This page intentionally left blank.]



## APPENDIX B: EXAMPLE OF TDEA FORWARD AND INVERSE CIPHER OPERATIONS

This Appendix presents an example that may be used when implementing the TDEA forward and inverse cipher operations. Appendices B.1 and B.2 provide an example of TDEA forward and inverse cipher operations in the Electronic Codebook (ECB) mode of encryption as specified in SP 800-38A.

In this example, all keys, plaintext and ciphertext are expressed in hexadecimal. The example uses three independent keys (Keying Option 1), which are:

$Key_1 = 0123456789ABCDEF$   
 $Key_2 = 23456789ABCDEF01$   
 $Key_3 = 456789ABCDEF0123$

The plaintext for the example is selected from the ASCII encoding of the phrase “The quick brown fox jumped over the lazy dog’s back”. The example employs the first 24 characters of the phrase (i.e., The quick brown fox jump).

The ASCII encoding of the above phrase is segmented as follows:

“The quic”	5468652071756663
“k brown ”	6B2062726F776E20
“fox jump”	666F78206A756D70

### B.1 TDEA Block Cipher Encryption - ECB Mode

In the example below, the input and output of the DEA cryptographic engine are given sequentially. At step 1, the input to  $DEA_1$  is  $P_1$ , and the output of  $DEA_1$  is “A28E91724C4BBA31”. At step 2, the input to  $DEA_2$  is the output of  $DEA_1$ , and the output of  $DEA_2$  is “5A2EA7F983A2F53F”. At step 3, the input to  $DEA_3$  is the output of  $DEA_2$ , and the output of  $DEA_3$  is “A826FD8CE53B855F”. The output of  $DEA_3$  is the ciphertext  $C_1$ .

$P_1 = \text{“The quic”} = 5468652071756663$

	Input	Output
$DEA_1 - F_{Key1}$	5468652071756663	A28E91724C4BBA31
$DEA_2 - I_{Key2}$	A28E91724C4BBA31	5A2EA7F983A2F53F
$DEA_3 - F_{Key3}$	5A2EA7F983A2F53F	A826FD8CE53B855F

Initial Public Draft

$C_1 = A826FD8CE53B855F$

During the second TDEA operation, the input is  $P_2$ , and the output after the three steps is ciphertext  $C_2$ .

$P_2 = \text{"k brown"} = 6B2062726F776E20$

	Input	Output
$DEA_1 - F_{Key1}$	6B2062726F776E20	167E47EC24F71D63
$DEA_2 - I_{Key2}$	167E47EC24F71D63	EA141A7DD69701F0
$DEA_3 - F_{Key3}$	EA141A7DD69701F0	CCE21C8112256FE6

$C_2 = CCE21C8112256FE6$

During the third TDEA operation, the input is  $P_3$ , and the output after the three steps is ciphertext  $C_3$ .

$P_3 = \text{"fox jump"} = 666F78206A756D70$

	Input	Output
$DEA_1 - F_{Key1}$	666F78206A756D70	2C1A917234425365
$DEA_2 - I_{Key2}$	2C1A917234425365	8059EE8212E22A79
$DEA_3 - F_{Key3}$	8059EE8212E22A79	68D5C05DD9B6B900

$C_3 = 68D5C05DD9B6B900$

The resulting ciphertext is the concatenation of  $C_1$ ,  $C_2$  and  $C_3$  (i.e.,  $A826FD8CE53B855FCCE21C8112256FE668D5C05DD9B6B900$ ).

## B.2 TDEA Block Cipher Decryption - ECB Mode

During decryption in the ECB mode, the ciphertext  $C_1$ ,  $C_2$  and  $C_3$  from Appendix B.1 are fed into the TDEA to produce the plaintext  $P_1$ ,  $P_2$  and  $P_3$ . The output of  $DEA_3$  becomes the input to  $DEA_2$ , and the output of  $DEA_2$  becomes the input to  $DEA_1$ .

$C_1 = A826FD8CE53B855F$

	Input	Output
$DEA_3 - I_{Key3}$	A826FD8CE53B855F	5A2EA7F983A2F53F
$DEA_2 - F_{Key2}$	5A2EA7F983A2F53F	A28E91724C4BBA31
$DEA_1 - I_{Key1}$	A28E91724C4BBA31	5468652071756663

$P_1 = 5468652071756663 = \text{"The quic"}$ .

Initial Public Draft

$C_2 = \text{CCE21C8112256FE6}$

	<b>Input</b>	<b>Output</b>
$\text{DEA}_3 - I_{\text{Key3}}$	CCE21C8112256FE6	EA141A7DD69701F0
$\text{DEA}_2 - F_{\text{Key2}}$	EA141A7DD69701F0	167E47EC24F71D63
$\text{DEA}_1 - I_{\text{Key1}}$	167E47EC24F71D63	6B2062726F776E20

$P_2 = \text{6B2062726F776E20} = \text{"k brown"}$ .

$C_3 = \text{68D5C05DD9B6B900}$

	<b>Input</b>	<b>Output</b>
$\text{DEA}_3 - D_{\text{Key3}}$	68D5C05DD9B6B900	8059EE8212E22A79
$\text{DEA}_2 - E_{\text{Key2}}$	8059EE8212E22A79	2C1A917234425365
$\text{DEA}_1 - D_{\text{Key1}}$	2C1A917234425365	666F78206A756D70

$P_3 = \text{666F78206A756D70} = \text{"fox jump"}$ .

The plaintext is the ASCII encoding of "The quick brown fox jump".

Initial Public Draft

[This page intentionally left blank.]

## APPENDIX C: GLOSSARY

Approved	FIPS-approved or NIST-recommended: an algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation.
Authentication	Provides assurance of the authenticity and, the refore, the integrity of data.
Bit	A binary digit having a value of zero or one.
Block	In this recommendation, a binary string, for example, a plaintext or a ciphertext, is segmented with a given length. Each segment is called a block. Data is processed block by block, from left to right.
Block Cipher Algorithm	A family of functions and their inverses that is parameterized by cryptographic keys; the function maps bit strings of a fixed length to bit strings of the same length.
Byte	A group of eight bits that is treated either as a single entity or as an array of eight individual bits.
Ciphertext	Encrypted (enciphered) data.
Cryptographic Key	A parameter that determines the transformation using DEA and TDEA forward and inverse operations.
Data Encryption Algorithm	The DEA cryptographic engine that is used by the Triple Data Encryption Algorithm (TDEA).
Decryption	The process of transforming ciphertext into plaintext.
Encryption	The process of transforming plaintext into ciphertext.
Exclusive-OR	The bit-by-bit modulo 2 addition of binary vectors of equal length.
FIPS	Federal Information Processing Standard.
Forward Cipher	One of the two functions of the block cipher algorithm that is determined by the choice of a cryptographic key.
Inverse Cipher	The block cipher algorithm function that is the inverse of the forward cipher function.
Key	See cryptographic key.
Key Bundle	The three cryptographic keys ( $Key_1$ , $Key_2$ , $Key_3$ ) that are used with a TDEA mode.
Plaintext	Intelligible data that has meaning and can be read or acted upon without the application of decryption. Also known as cleartext.

Initial Public Draft

[This page intentionally left blank.]

## APPENDIX D: REFERENCES

- ANSI X9.52            *Triple Data Encryption Algorithm Modes Of Operation, X9.52 – 1998*, Accredited Standards Committee X9, American National Standards Institute, July 27, 1998.
- ANSI X9.82  
[Draft]                *Random Number Generation, X9.82 – Draft*, Accredited Standards Committee X9, American National Standards Institute, [date].
- FIPS 140-2            *Security Requirements for Cryptographic Modules*, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, May 25, 2001.
- FIPS 197              *Advanced Encryption Standard*, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, November 2001.
- SP 800-20             *Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures*, SP 800-20, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, April 2000 Revision.
- SP 800-38A            *Recommendation for Block Cipher Modes of Operation*, SP 800-38A 2001 ED, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, December 2001.
- SP 800-38B  
[Draft]                *Recommendation for Block Cipher Modes of Operation: the RMAC Authentication Mode*, SP 800-38B [Draft], Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, November 2002.
- SP 800-57  
[Draft]                *Key Management Guideline*, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, [date]





