US009270458B2

(12) United States Patent
Shibutani et al.

(10) Patent No.: US 9,270,458 B2
(45) Date of Patent: Feb. 23, 2016

(54) ENCRYPTION PROCESSING DEVICE, ENCRYPTION PROCESSING METHOD, AND PROGRAM

(75) Inventors: **Kyoji Shibutani**, Tokyo (JP); **Atsushi Mitsuda**, Tokyo (JP); **Toru Akishita**, Tokyo (JP); **Takanori Isobe**, Tokyo (JP); **Taizo Shirai**, Kanagawa (JP); **Harunaga Hiwatari**, Kanagawa (JP)

(73) Assignee: **Sony Corporation**, Tokyo (JP)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 48 days.

(21) Appl. No.: **14/002,462**

(22) PCT Filed: **Feb. 20, 2012**

(86) PCT No.: **PCT/JP2012/053933**
§ 371 (c)(1),
(2), (4) Date: **Aug. 30, 2013**

(87) PCT Pub. No.: **WO2012/132623**
PCT Pub. Date: **Oct. 4, 2012**

(65) **Prior Publication Data**

US 2013/0343546 A1 Dec. 26, 2013

(30) **Foreign Application Priority Data**

Mar. 28, 2011	(JP)	2011-069185
Sep. 22, 2011	(JP)	2011-207705

(51) **Int. Cl.**
H04L 9/08 (2006.01)
G09C 1/00 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 9/0894*** (2013.01); ***G09C 1/00*** (2013.01); ***H04L 9/002*** (2013.01); *H04L 9/0625* (2013.01); *H04L 2209/24* (2013.01)

(58) **Field of Classification Search**
CPC H04L 9/0625; H04L 9/0631; H04L 9/0894
USPC 380/28–29, 37, 277, 286
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,747,011	B2 *	6/2010	Shirai et al.	380/28
7,881,466	B2 *	2/2011	Gorissen et al.	380/28

(Continued)

FOREIGN PATENT DOCUMENTS

JP	2008-051829	A	3/2008
JP	2008-145791	A	6/2008
WO	WO 97/09705	A1	3/1997

OTHER PUBLICATIONS

Haruki Seki et al., Differential Cryptanalysis of CAST-256 Reduced to Nine Quad-Rounds, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E84-A No. 4 (Apr. 2001).*

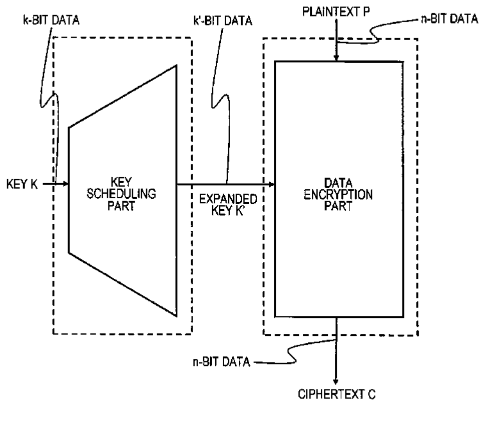
(Continued)

Primary Examiner — Samson Lemma
(74) *Attorney, Agent, or Firm* — Sony Corporation

(57) **ABSTRACT**

An encryption processing device including an encryption processing part configured to divide configuration bits of data to be data processed into plural lines, and to input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and a key scheduling part configured to output round keys to a round calculation executing unit in the encryption processing part. The key scheduling part is a replacement type key scheduling part configured to generate plural round keys or round key configuration data by dividing a secret key stored beforehand into plural parts. The plural round keys are output to a round calculation executing unit sequentially executing in the encryption processing part such that a constant sequence is not repeated. The encryption processing configuration has a high level of security and a high level of resistance to repeated key attacks or other attacks.

16 Claims, 43 Drawing Sheets



(51) **Int. Cl.**

H04L 9/00	(2006.01)
H04L 9/06	(2006.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,073,140	B2 *	12/2011	Shirai et al.	380/29
8,295,478	B2 *	10/2012	Shirai et al.	380/28
8,737,603	B2 *	5/2014	Shirai et al.	380/28
2004/0049687	A1 *	3/2004	Orsini et al.	713/189
2009/0010425	A1	1/2009	Shibutani et al.	
2010/0061548	A1 *	3/2010	Shirai et al.	380/28

OTHER PUBLICATIONS

U.S. Appl. No. 14/002,379, filed Aug. 30, 2013, Shibutani, et al.
U.S. Appl. No. 14/005,663, filed Sep. 17, 2013, Shibutani, et al.
U.S. Appl. No. 14/006,392, filed Sep. 20, 2013, Shibutani, et al.
Office Action issued Oct. 21, 2014 in Japanese Patent Application No. 2011-207705 (with English language translation).
Youngdai Ko, et al., "Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST" Lecture Notes in Computer Science, vol. 3017, 2004, pp. 299-316 and cover page.
International Search Report issued Apr. 3, 2012 in Application No. PCT/JP2012/053933.
Yuliang Zheng, et al., "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses" Advances in Cryptology—CRYPT'89 Proceedings Lecture Notes in Computer Science, vol. 435, 1990, pp. 461-480.
Kaisa Nyberg, "Generalized Feistel Networks", Advances in Cryptology—ASIACRYPT'96, Lecture Notes in Computer Science, vol. 1163, 1996, pp. 91-104.
"The 128-bit Blockcipher CLEFIA: Algorithm Specification", Sony Corporation, Revision 1.0, Jun. 1, 2007, 41 pages.
Kazumaro Aoki, et al., "Specification of Camellia—a 128-bit Block Cipher", Nippon Telegraph and Telephone & Mitsubishi Electric Corporations, Version 1.0: Jul. 12, 2000, Version 2.0: Sep. 26, 2001, 35 pages.
"GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms", V. Dolmatov, Ed., Cryptocom, Ltd., ISSN: 2070-1721, Mar. 2010, 19 pages.
3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification (Release 9), 3GPP TS 35.202 V9.0.0 (Dec. 2009), 2009, 24 pages.
Youngdai Ko, et al., "Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST", Lecture Notes in Computer Science, vol. 3017, Jul. 28, 2004, pp. 299-316.
Kyoji Shibutani, et al., "Piccolo: An Ultra-Lightweight Blockcipher", Lecture Notes in Computer Science, vol. 6917, Sep. 27, 2011, pp. 342-357.
Extended European Search Report issued Feb. 20, 2015 in Patent Application No. 12765375.6.
Lawrie Brown, et al., "Introducing the new LOKI97 Block Cipher" Retrieved from the Internet: URL:http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C2B792B35E02A510DB42AE-53F0EF82B4?doi=10.1.1.26.6958&rep=rep1&type=pdf [retrieved on Feb. 9, 2015], XP055168204, Jun. 12, 1998, 22 Pages.
Ronald L. Rivest, et al., "The RC6™ Block Cipher" First Advanced Encryption Standard (AES) Conference, Retrieved from the Internet: URL:http://people.csail.mit.edu/rivest/Rc6.pdf [retrieved on Aug. 27, 2009] XP002543314, Aug. 20, 1998, 21 Pages.
"Universal Mobile Telecommunications System (UMTS); LTE; Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification (3GPP TS 35.202 version 9.0.0 Release 9)" ETSI TS 135 202 V9.0.0 (Feb. 2010), XP014045969, Feb. 1, 2010, 26 Pages.
Ralph C. Merkle, "Fast Software Encryption Functions" Advances in Cryptology—CRYPTO '90 Proceedings Lecture Notes in Computer Science, vol. 537, XP047288201, 1991, pp. 477-501.
Panasayya Yalla, et al., "Compact FPGA Implementation of Camellia" International Conference on Field Programmable Logic and Applications, XP031534082, Aug. 31, 2009, pp. 658-661.

* cited by examiner

FIG. 1

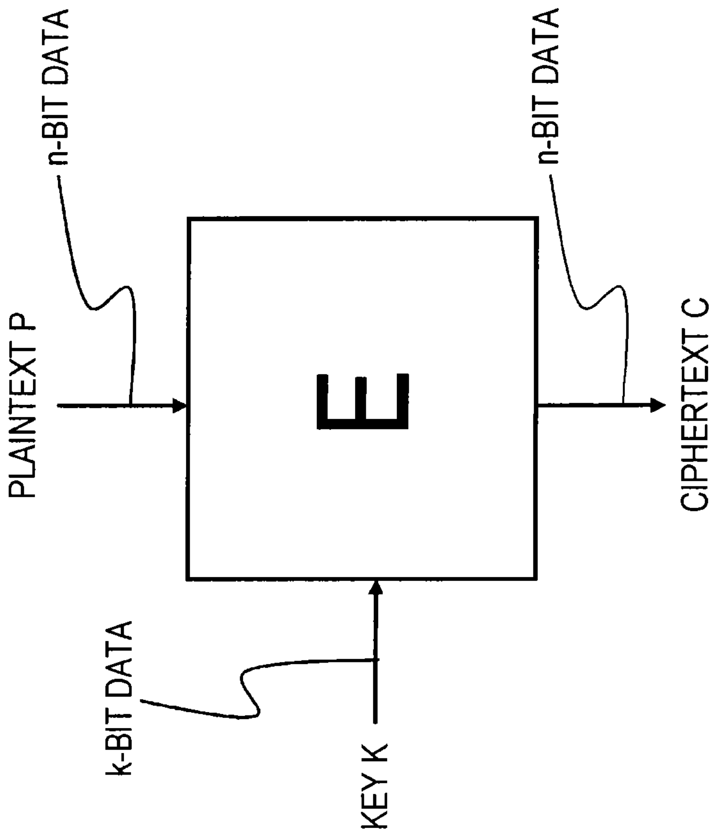

PLAINTEXT P
n-BIT DATA
E
n-BIT DATA
CIPHERTEXT C
k-BIT DATA
KEY K

FIG. 2

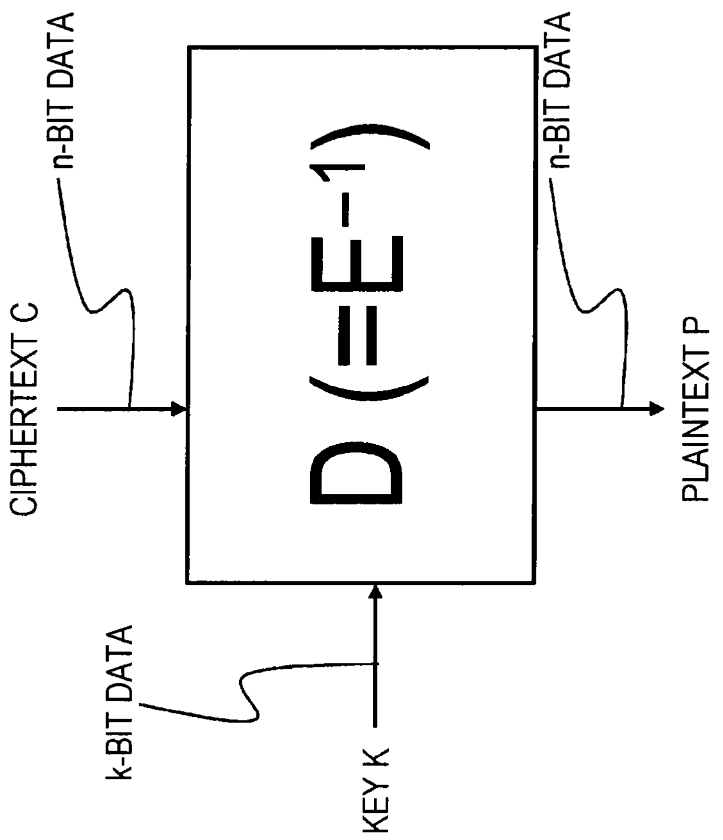
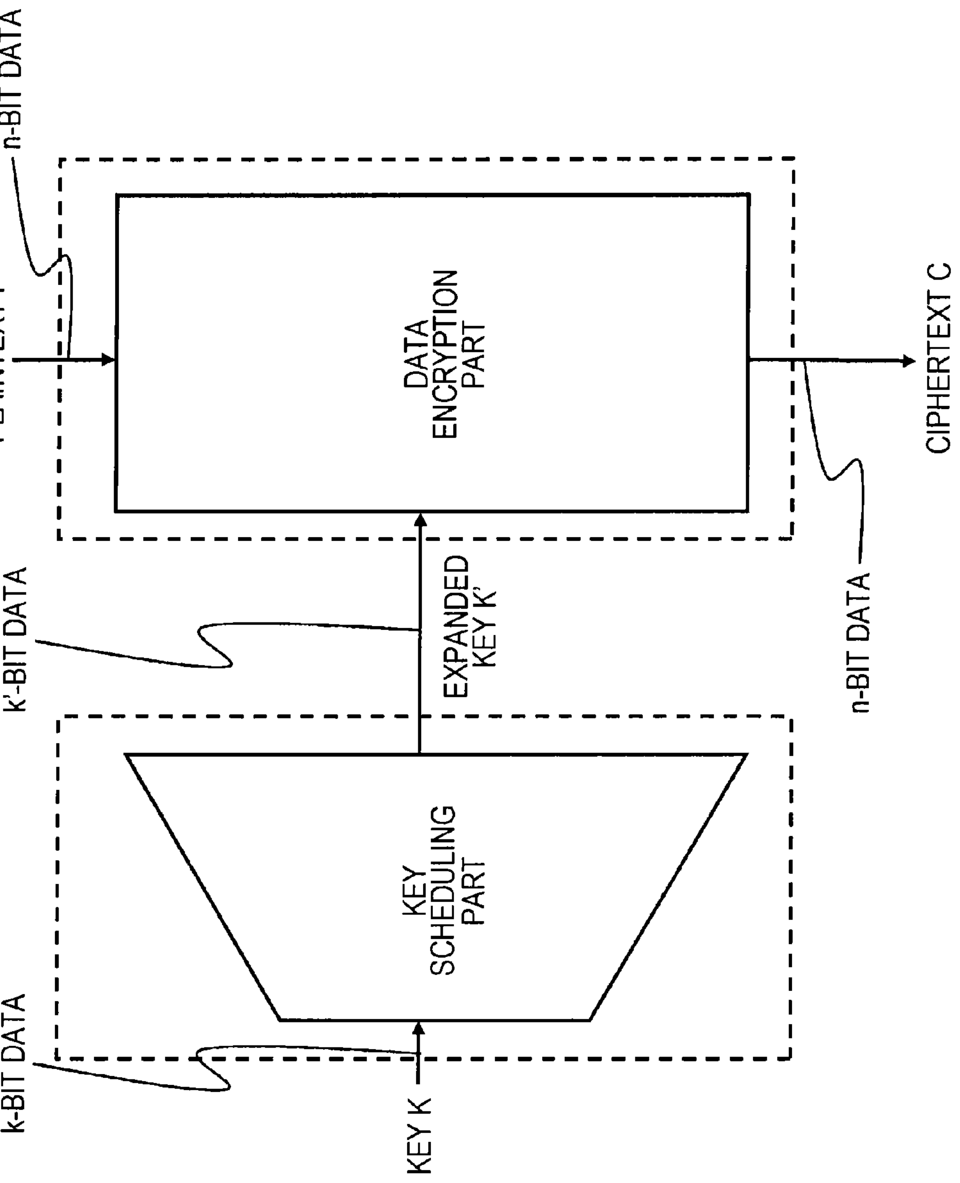

CIPHERTEXT C
n-BIT DATA
D (=E^{-1})
n-BIT DATA
PLAINTEXT P
k-BIT DATA
KEY K

FIG. 3



PLAINTEXT P
n-BIT DATA
DATA ENCRYPTION PART
CIPHERTEXT C
n-BIT DATA
k'-BIT DATA
EXPANDED KEY K'
KEY SCHEDULING PART
k-BIT DATA
KEY K

FIG. 4

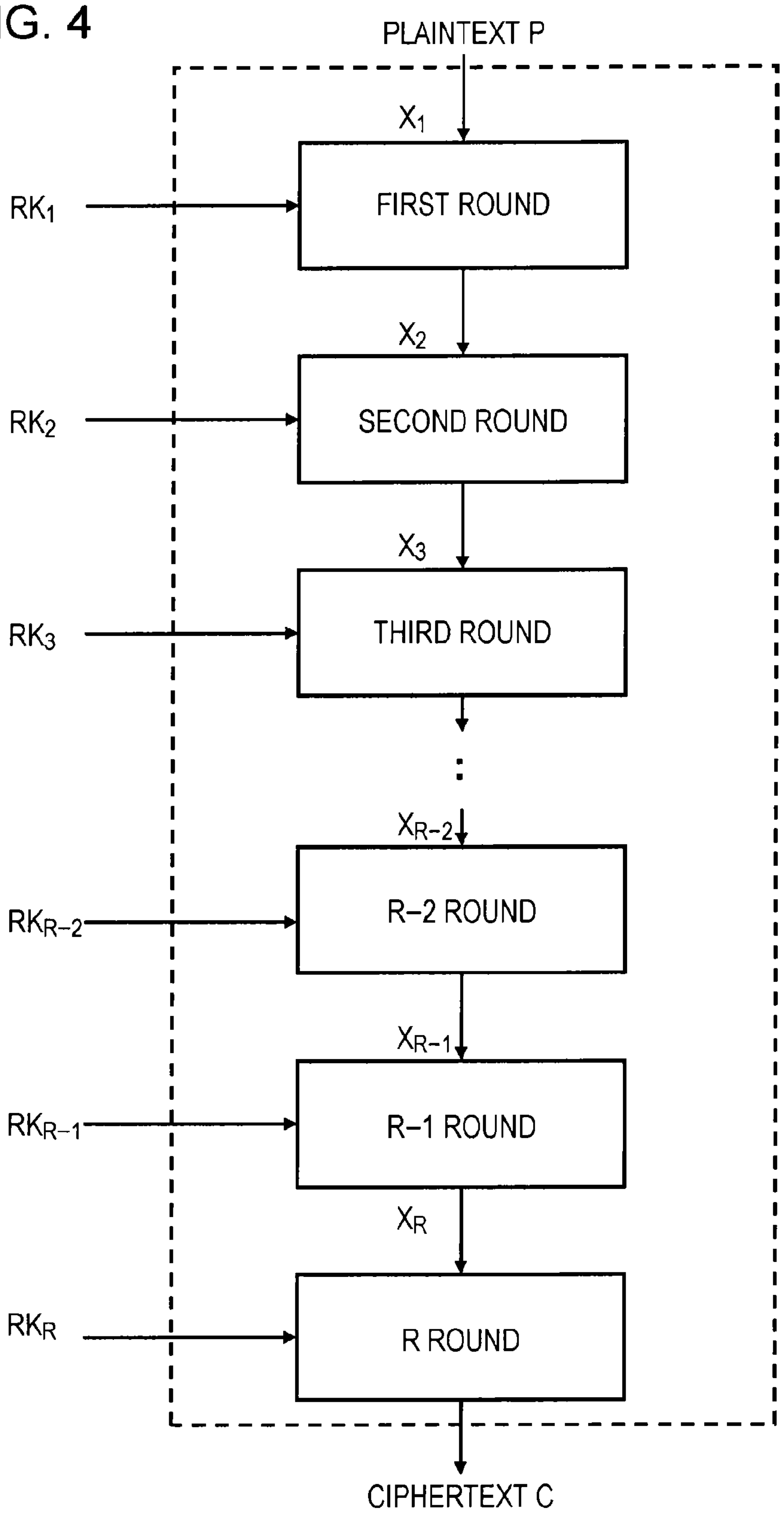

PLAINTEXT P
X_1
RK_1
FIRST ROUND
X_2
RK_2
SECOND ROUND
X_3
RK_3
THIRD ROUND
X_{R-2}
RK_{R-2}
R−2 ROUND
X_{R-1}
RK_{R-1}
R−1 ROUND
X_R
RK_R
R ROUND
CIPHERTEXT C

FIG. 5

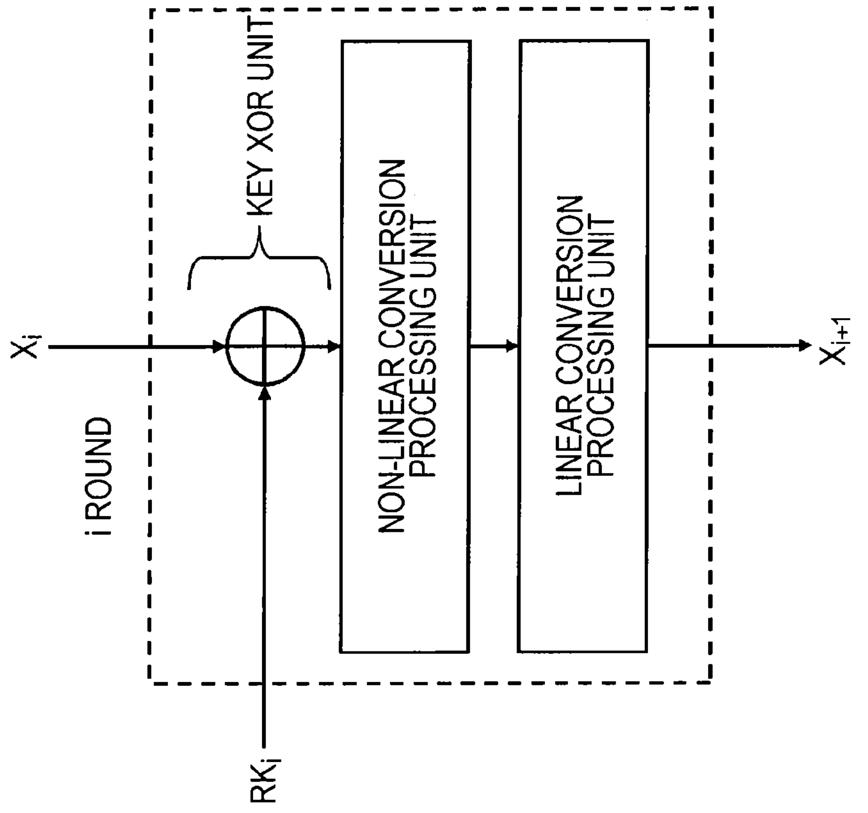
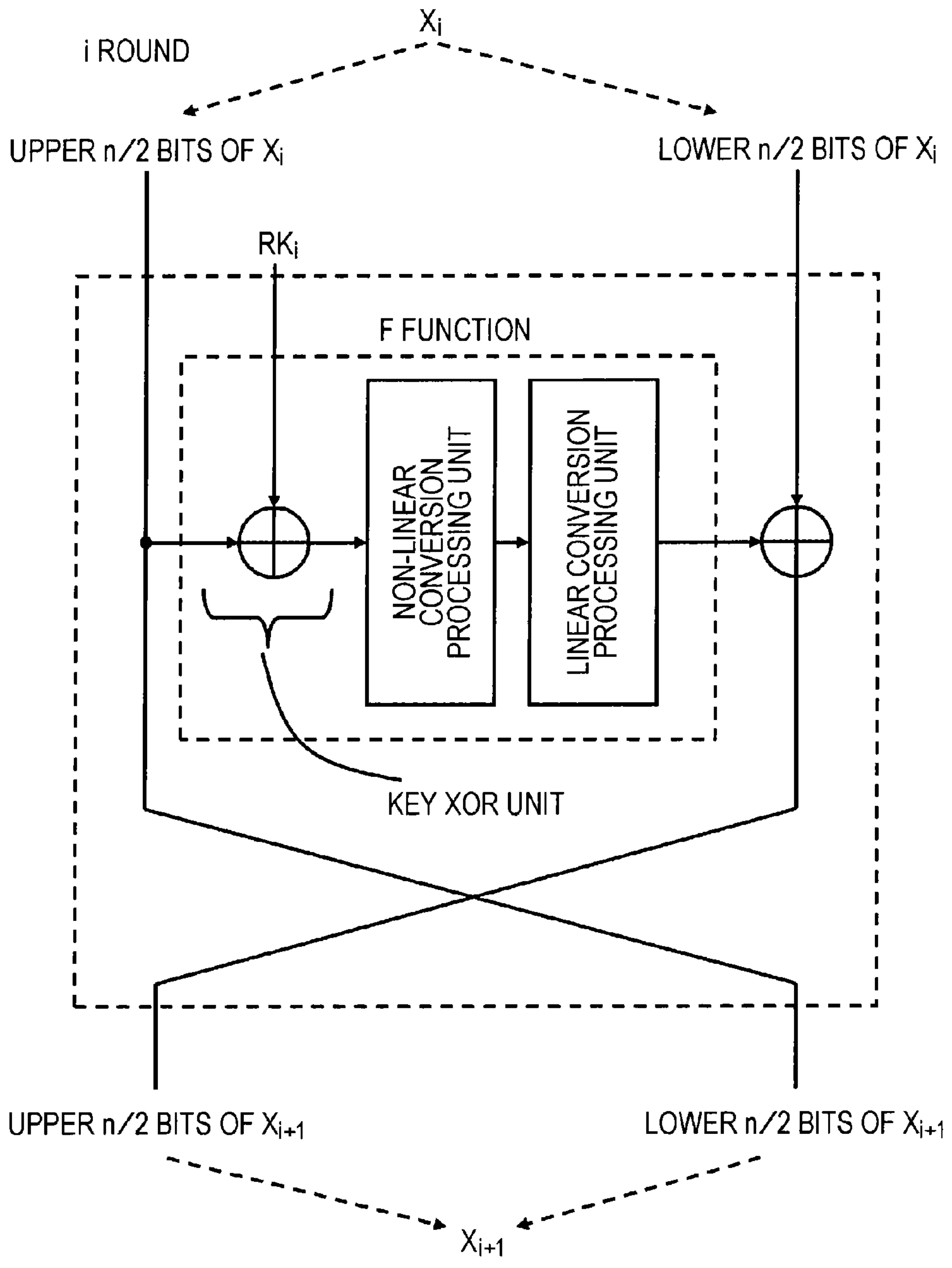

X_i
i ROUND
KEY XOR UNIT
NON-LINEAR CONVERSION PROCESSING UNIT
LINEAR CONVERSION PROCESSING UNIT
X_{i+1}
RK_i

FIG. 6



X_i
i ROUND
UPPER n/2 BITS OF X_i
LOWER n/2 BITS OF X_i
RK_i
F FUNCTION
NON-LINEAR CONVERSION PROCESSING UNIT
LINEAR CONVERSION PROCESSING UNIT
KEY XOR UNIT
UPPER n/2 BITS OF X_{i+1}
LOWER n/2 BITS OF X_{i+1}
X_{i+1}

FIG. 7

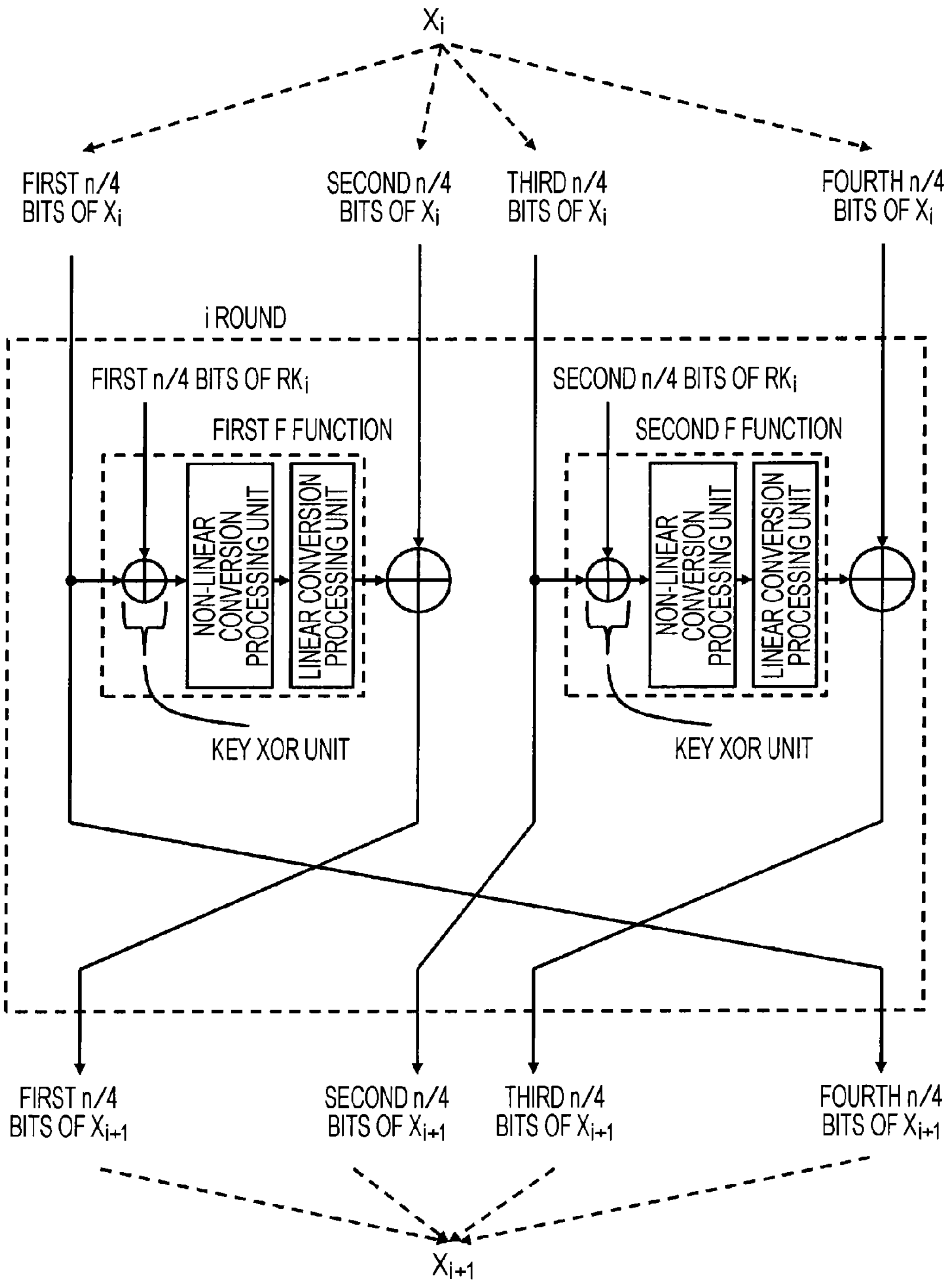

X_i
FIRST n/4 BITS OF X_i
SECOND n/4 BITS OF X_i
THIRD n/4 BITS OF X_i
FOURTH n/4 BITS OF X_i
i ROUND
FIRST n/4 BITS OF RK_i
FIRST F FUNCTION
NON-LINEAR CONVERSION PROCESSING UNIT
LINEAR CONVERSION PROCESSING UNIT
KEY XOR UNIT
SECOND n/4 BITS OF RK_i
SECOND F FUNCTION
NON-LINEAR CONVERSION PROCESSING UNIT
LINEAR CONVERSION PROCESSING UNIT
KEY XOR UNIT
FIRST n/4 BITS OF X_{i+1}
SECOND n/4 BITS OF X_{i+1}
THIRD n/4 BITS OF X_{i+1}
FOURTH n/4 BITS OF X_{i+1}
X_{i+1}

FIG. 8

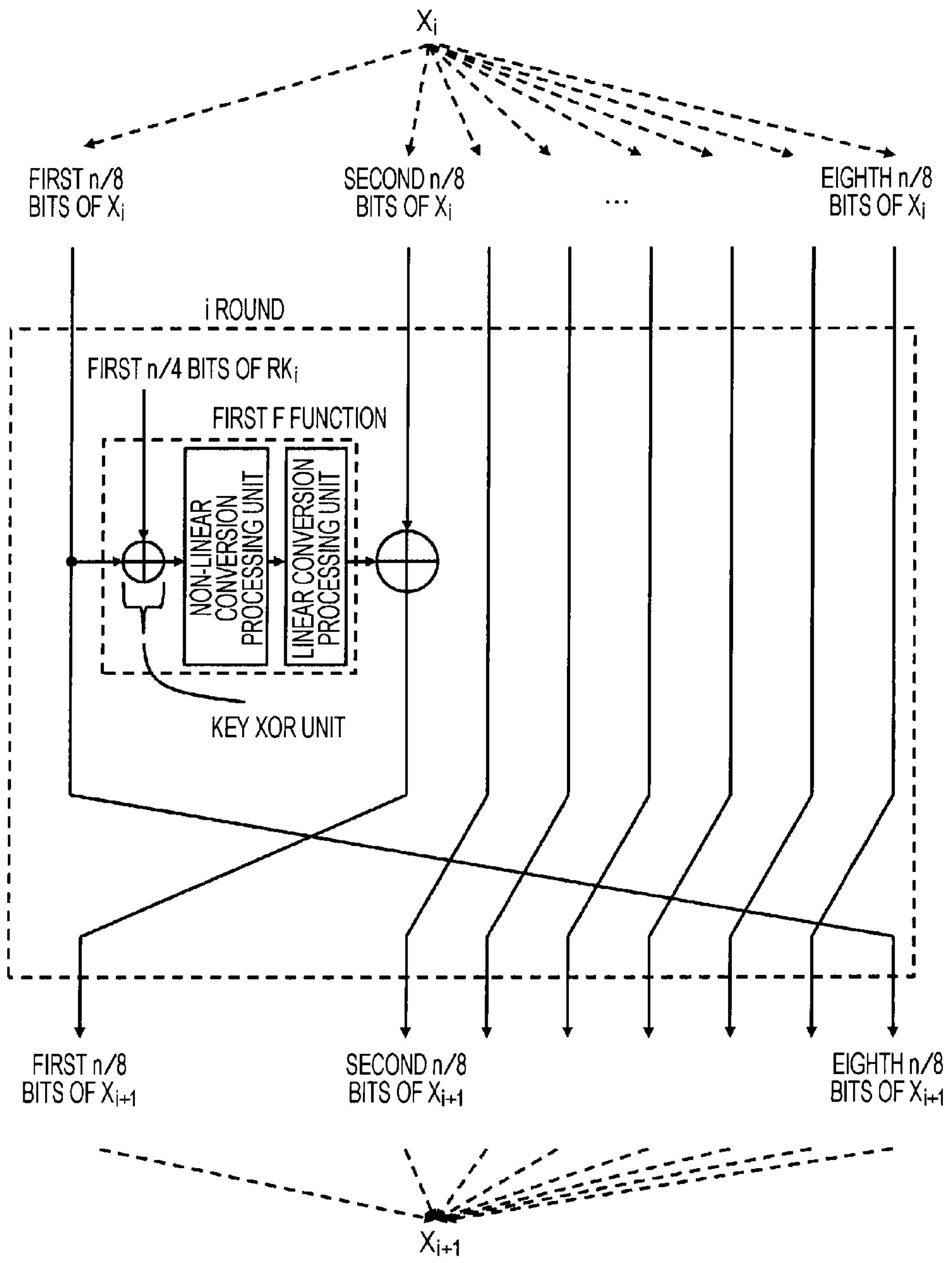

X_i
FIRST n/8 BITS OF X_i
SECOND n/8 BITS OF X_i
...
EIGHTH n/8 BITS OF X_i
i ROUND
FIRST n/4 BITS OF RK_i
FIRST F FUNCTION
NON-LINEAR CONVERSION PROCESSING UNIT
LINEAR CONVERSION PROCESSING UNIT
KEY XOR UNIT
FIRST n/8 BITS OF X_{i+1}
SECOND n/8 BITS OF X_{i+1}
EIGHTH n/8 BITS OF X_{i+1}
X_{i+1}

FIG. 9

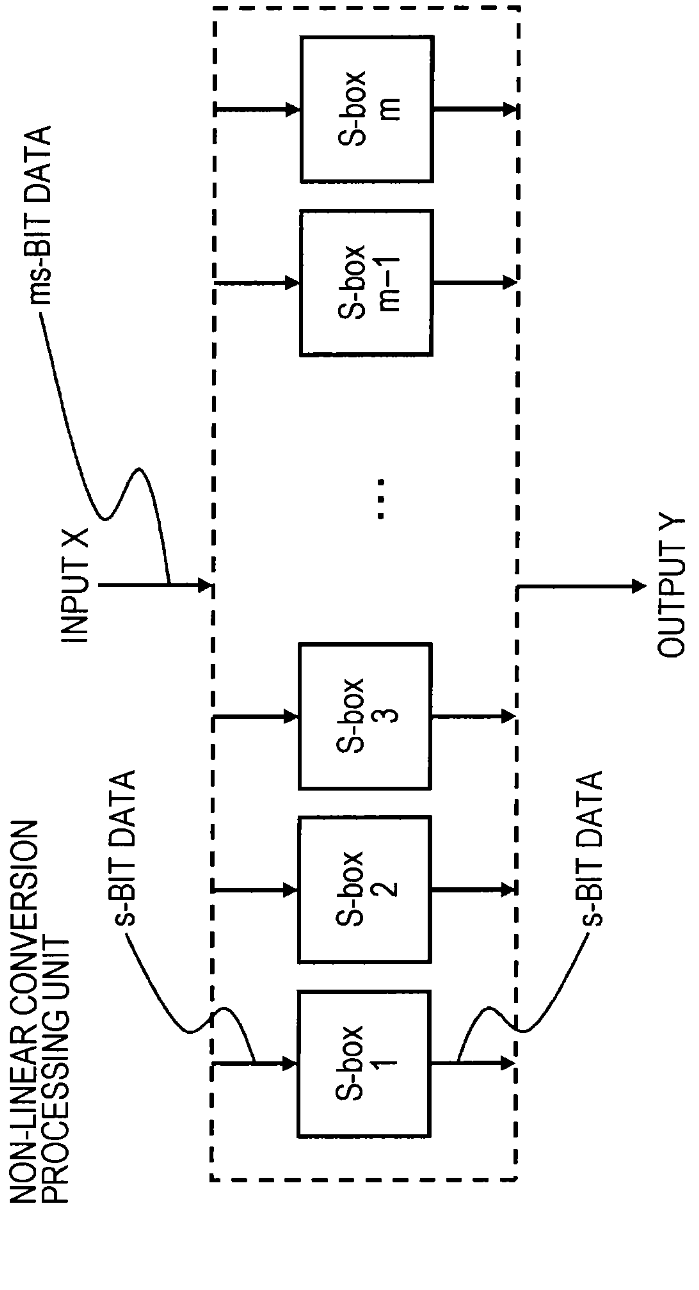
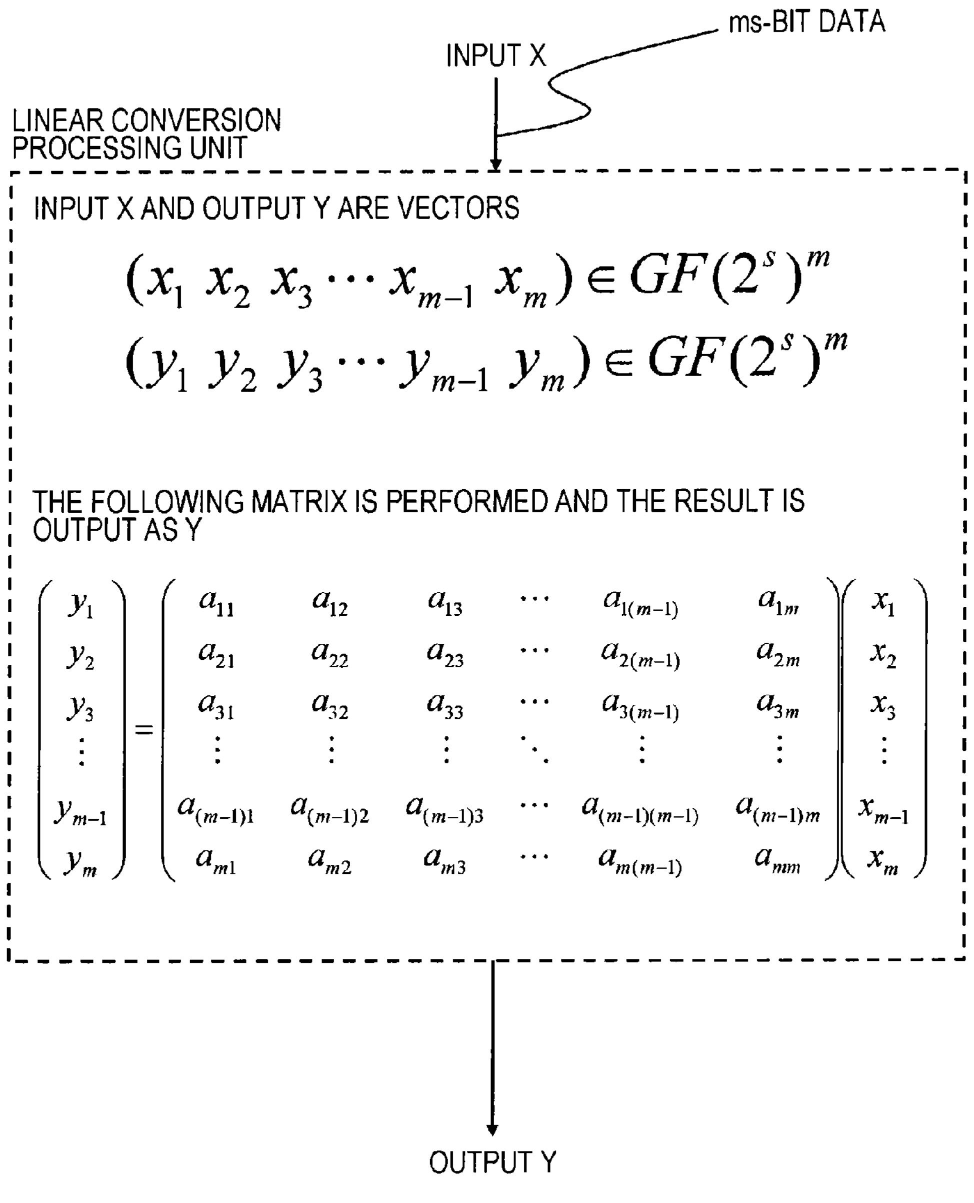

NON-LINEAR CONVERSION PROCESSING UNIT
s-BIT DATA
INPUT X
ms-BIT DATA
S-box 1
S-box 2
S-box 3
...
S-box m−1
S-box m
s-BIT DATA
OUTPUT Y

FIG. 10


ms-BIT DATA
INPUT X
LINEAR CONVERSION PROCESSING UNIT
INPUT X AND OUTPUT Y ARE VECTORS
THE FOLLOWING MATRIX IS PERFORMED AND THE RESULT IS OUTPUT AS Y
OUTPUT Y

$$(x_1\ x_2\ x_3 \cdots x_{m-1}\ x_m) \in GF(2^s)^m$$

$$(y_1\ y_2\ y_3 \cdots y_{m-1}\ y_m) \in GF(2^s)^m$$

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(m-1)} & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2(m-1)} & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3(m-1)} & a_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(m-1)1} & a_{(m-1)2} & a_{(m-1)3} & \cdots & a_{(m-1)(m-1)} & a_{(m-1)m} \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{m(m-1)} & a_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{m-1} \\ x_m \end{pmatrix}$$

FIG. 11

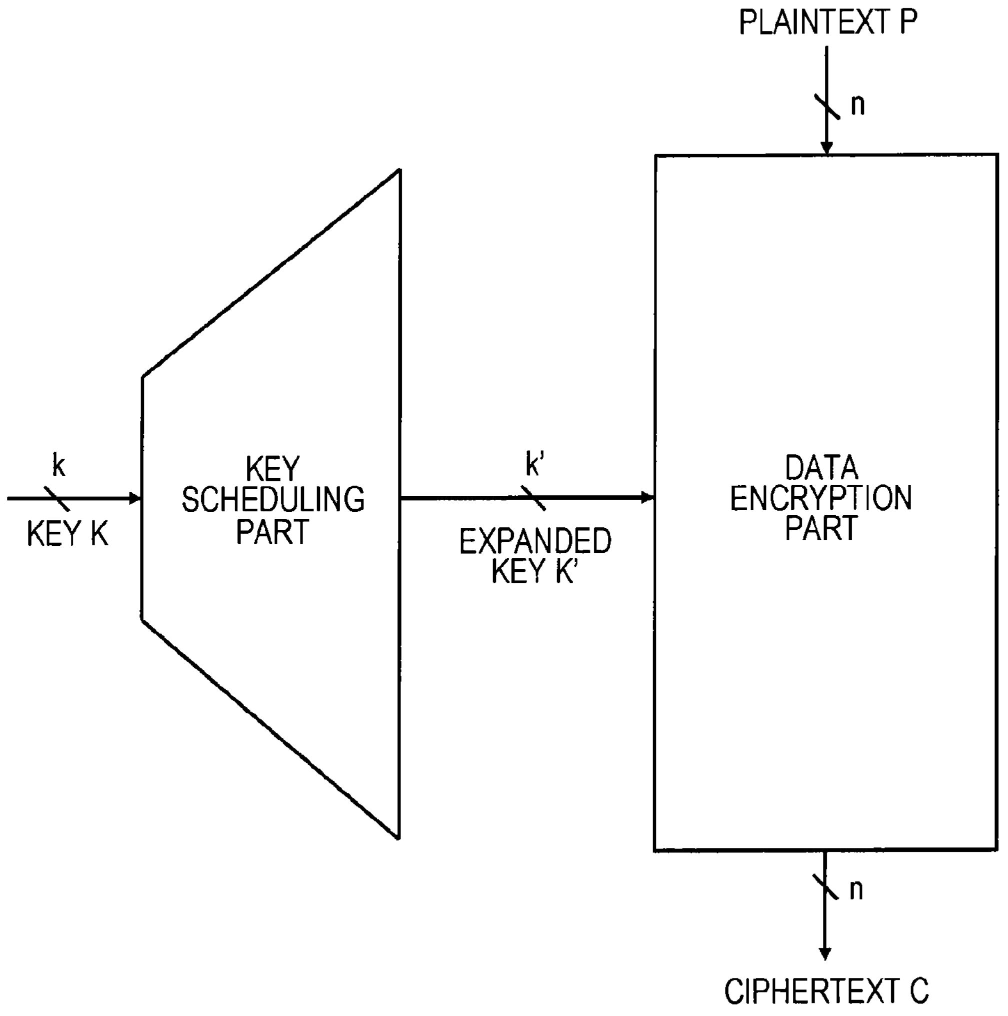

PLAINTEXT P
n
k
KEY K
KEY SCHEDULING PART
k'
EXPANDED KEY K'
DATA ENCRYPTION PART
n
CIPHERTEXT C

FIG. 12

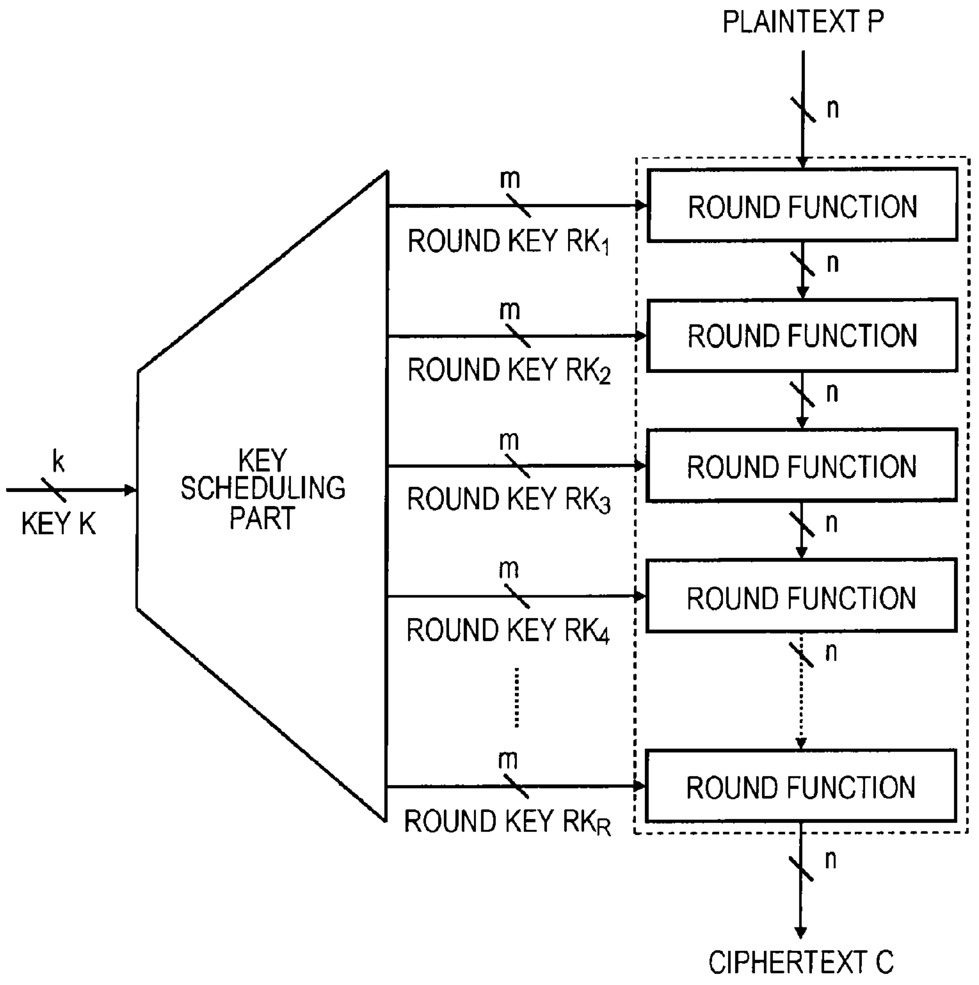

PLAINTEXT P
n
KEY K
k
KEY SCHEDULING PART
m
ROUND KEY RK_1
ROUND KEY RK_2
ROUND KEY RK_3
ROUND KEY RK_4
ROUND KEY RK_R
ROUND FUNCTION
CIPHERTEXT C

FIG. 13

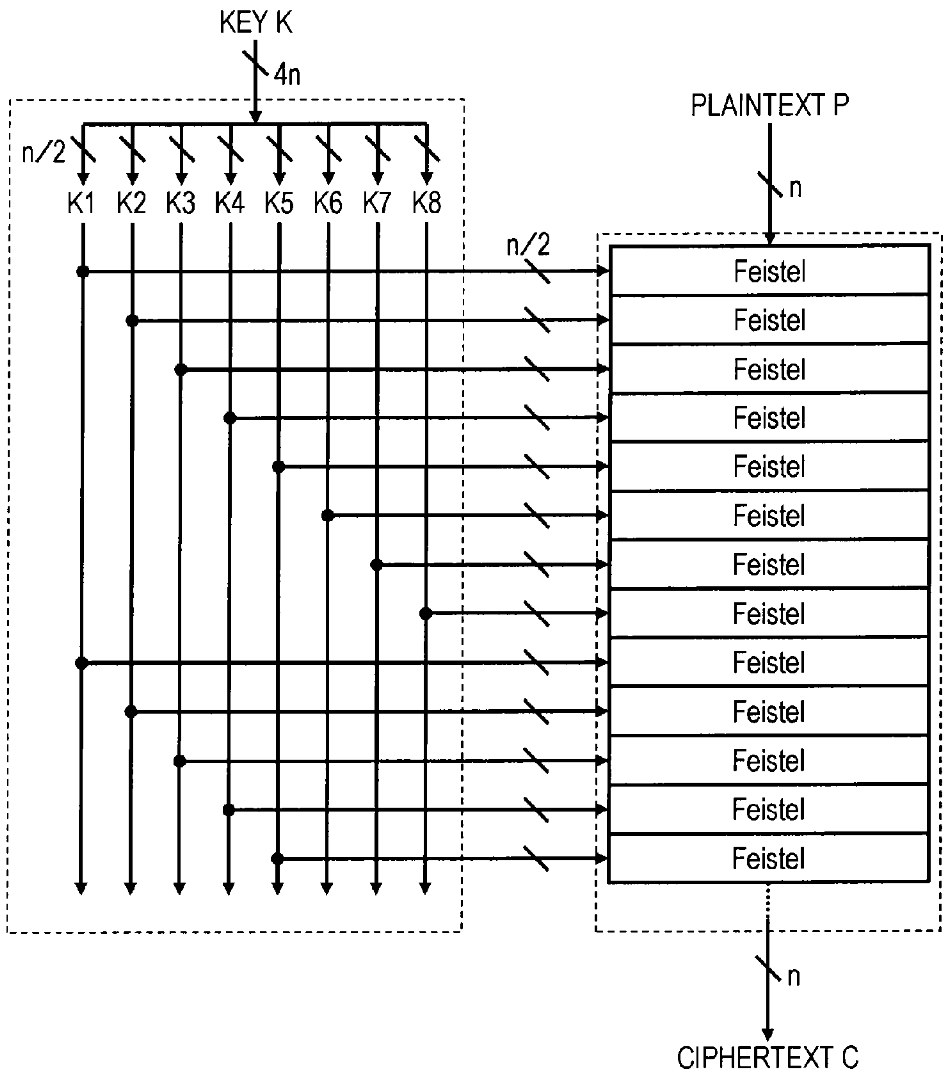

KEY K
4n
n/2
K1
K2
K3
K4
K5
K6
K7
K8
PLAINTEXT P
n
n/2
Feistel
n
CIPHERTEXT C

FIG. 14

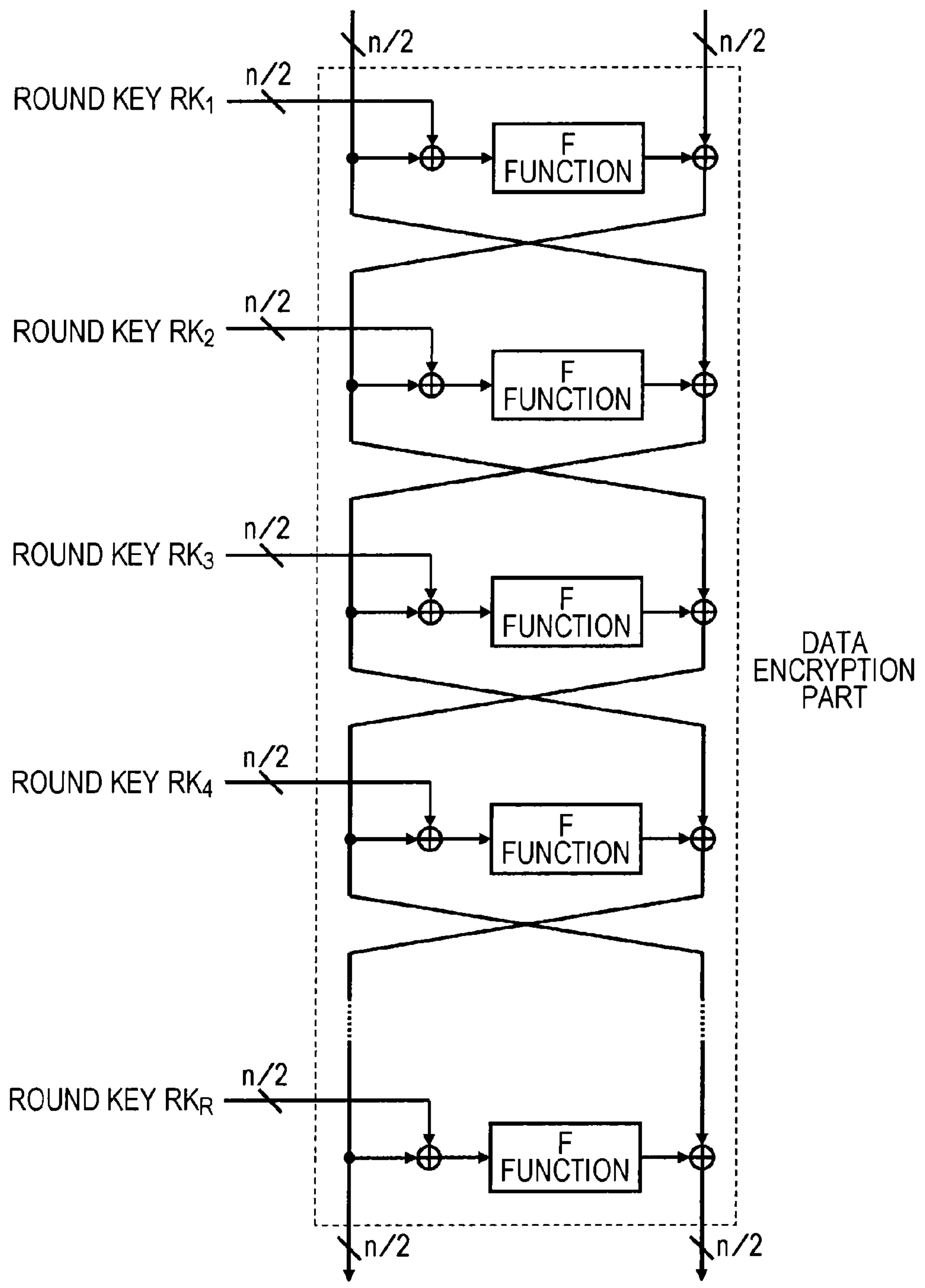

n/2
n/2
ROUND KEY RK_1
n/2
F FUNCTION
ROUND KEY RK_2
n/2
F FUNCTION
ROUND KEY RK_3
n/2
F FUNCTION
DATA ENCRYPTION PART
ROUND KEY RK_4
n/2
F FUNCTION
ROUND KEY RK_R
n/2
F FUNCTION
n/2
n/2

FIG. 15

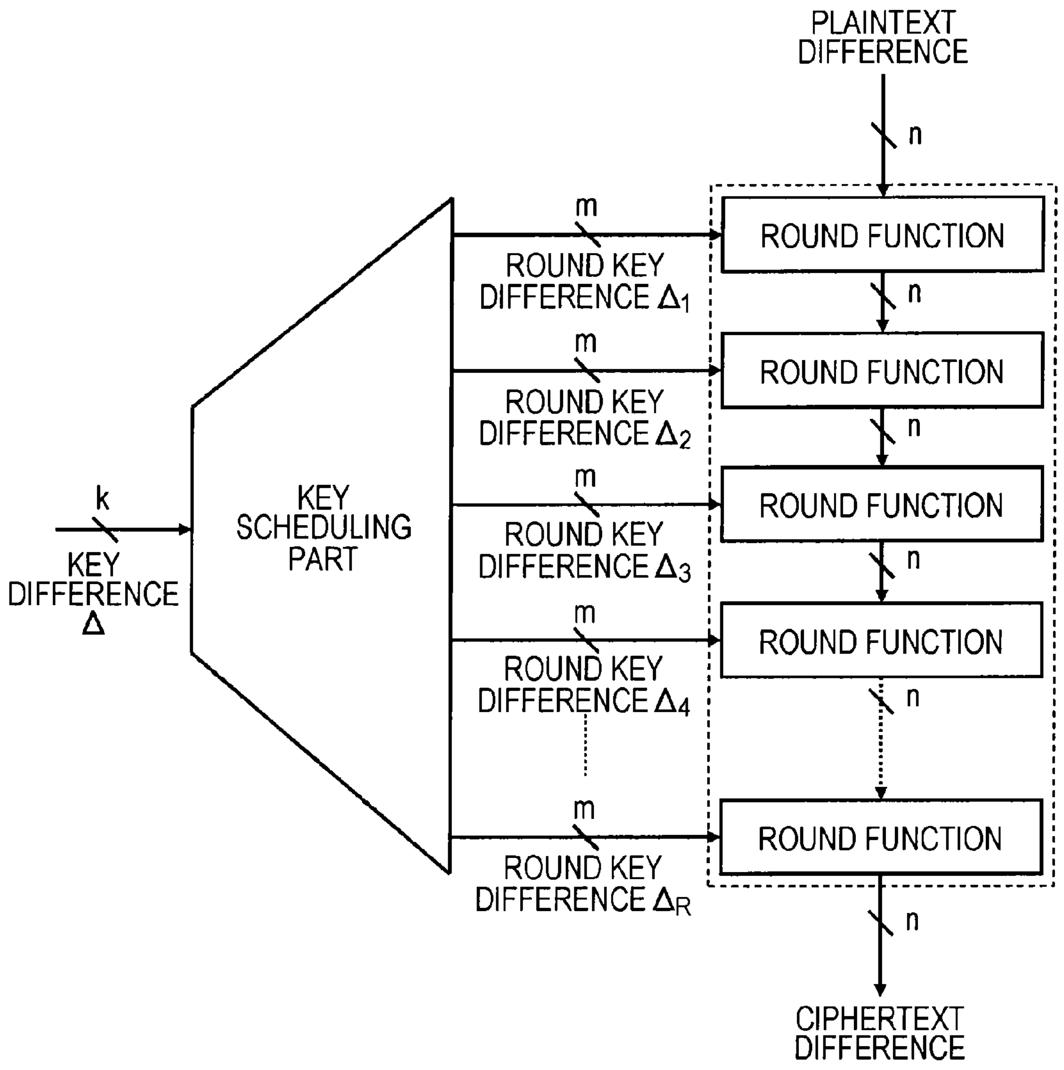

PLAINTEXT DIFFERENCE
n
m
ROUND FUNCTION
ROUND KEY DIFFERENCE Δ1
n
m
ROUND FUNCTION
ROUND KEY DIFFERENCE Δ2
n
k
KEY SCHEDULING PART
m
ROUND FUNCTION
KEY DIFFERENCE Δ
ROUND KEY DIFFERENCE Δ3
n
m
ROUND FUNCTION
ROUND KEY DIFFERENCE Δ4
n
m
ROUND FUNCTION
ROUND KEY DIFFERENCE ΔR
n
CIPHERTEXT DIFFERENCE

FIG. 16

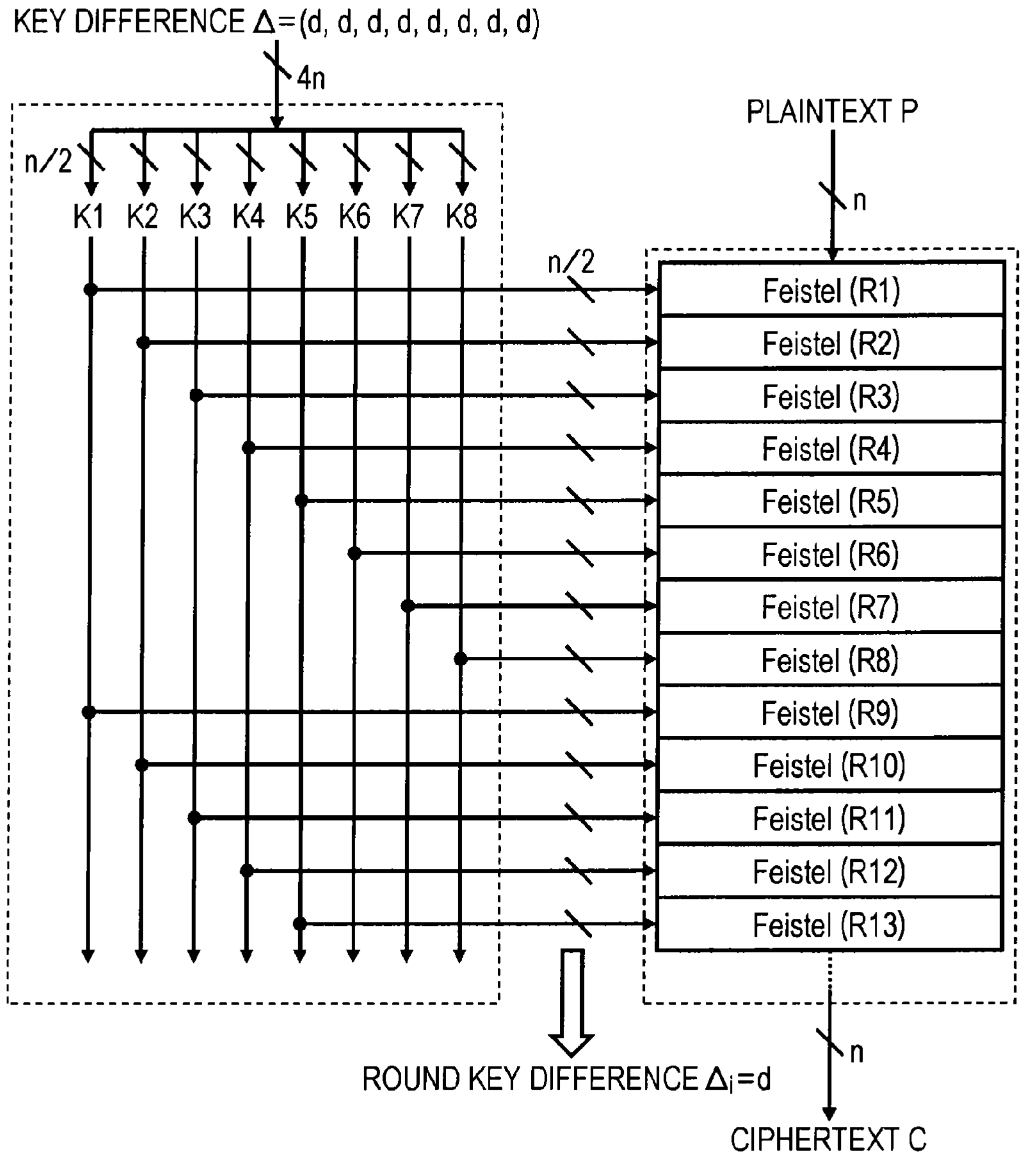

KEY DIFFERENCE Δ=(d, d, d, d, d, d, d, d)
4n
n/2
K1
K2
K3
K4
K5
K6
K7
K8
n/2
PLAINTEXT P
n
Feistel (R1)
Feistel (R2)
Feistel (R3)
Feistel (R4)
Feistel (R5)
Feistel (R6)
Feistel (R7)
Feistel (R8)
Feistel (R9)
Feistel (R10)
Feistel (R11)
Feistel (R12)
Feistel (R13)
ROUND KEY DIFFERENCE Δi=d
n
CIPHERTEXT C

FIG. 17

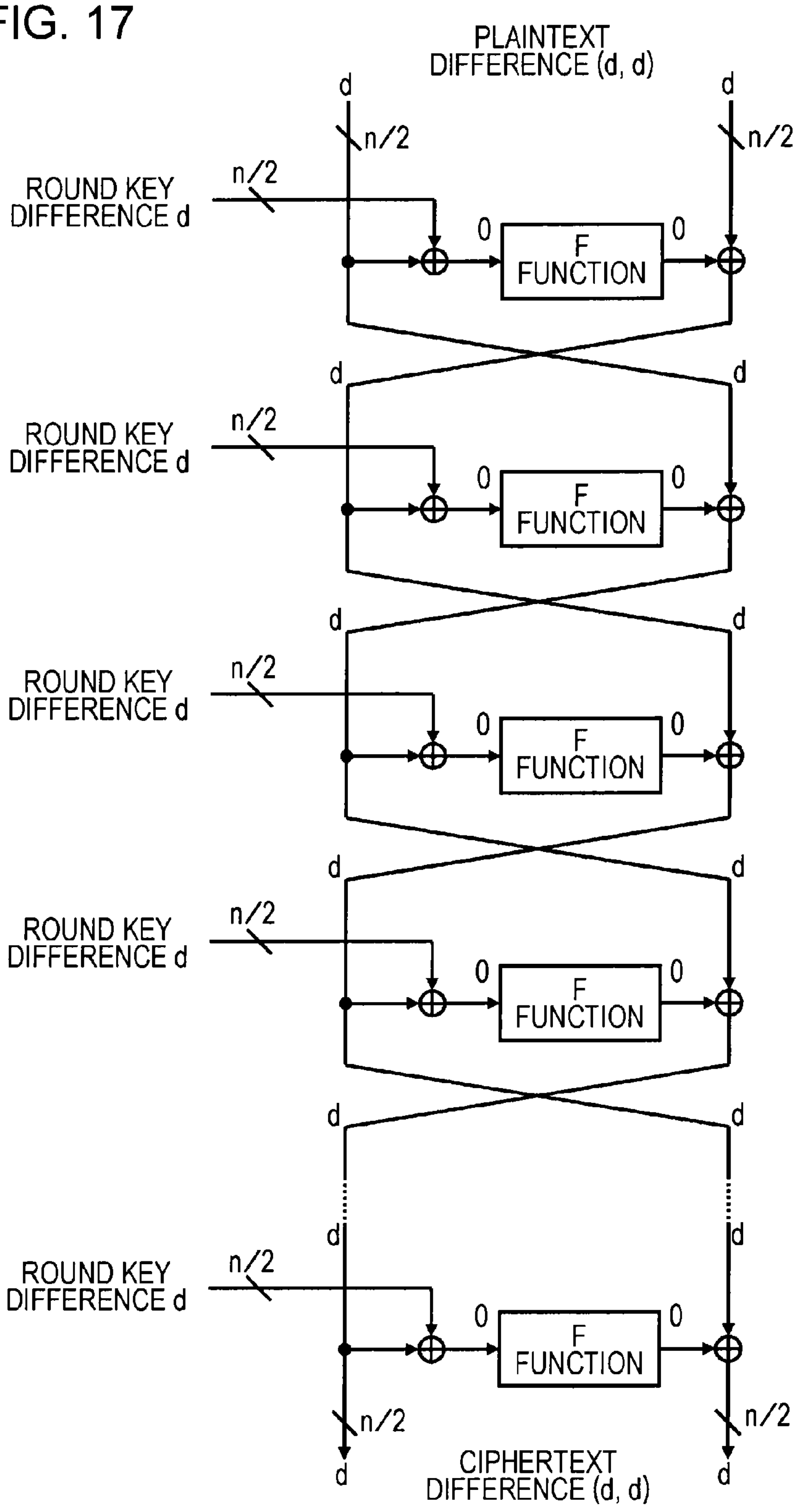

PLAINTEXT
DIFFERENCE (d, d)
d
d
n/2
n/2
ROUND KEY
DIFFERENCE d
n/2
0
0
F
FUNCTION
d
d
ROUND KEY
DIFFERENCE d
n/2
0
0
F
FUNCTION
d
d
ROUND KEY
DIFFERENCE d
n/2
0
0
F
FUNCTION
d
d
ROUND KEY
DIFFERENCE d
n/2
0
0
F
FUNCTION
d
d
d
d
ROUND KEY
DIFFERENCE d
n/2
0
0
F
FUNCTION
n/2
n/2
d
d
CIPHERTEXT
DIFFERENCE (d, d)

FIG. 18

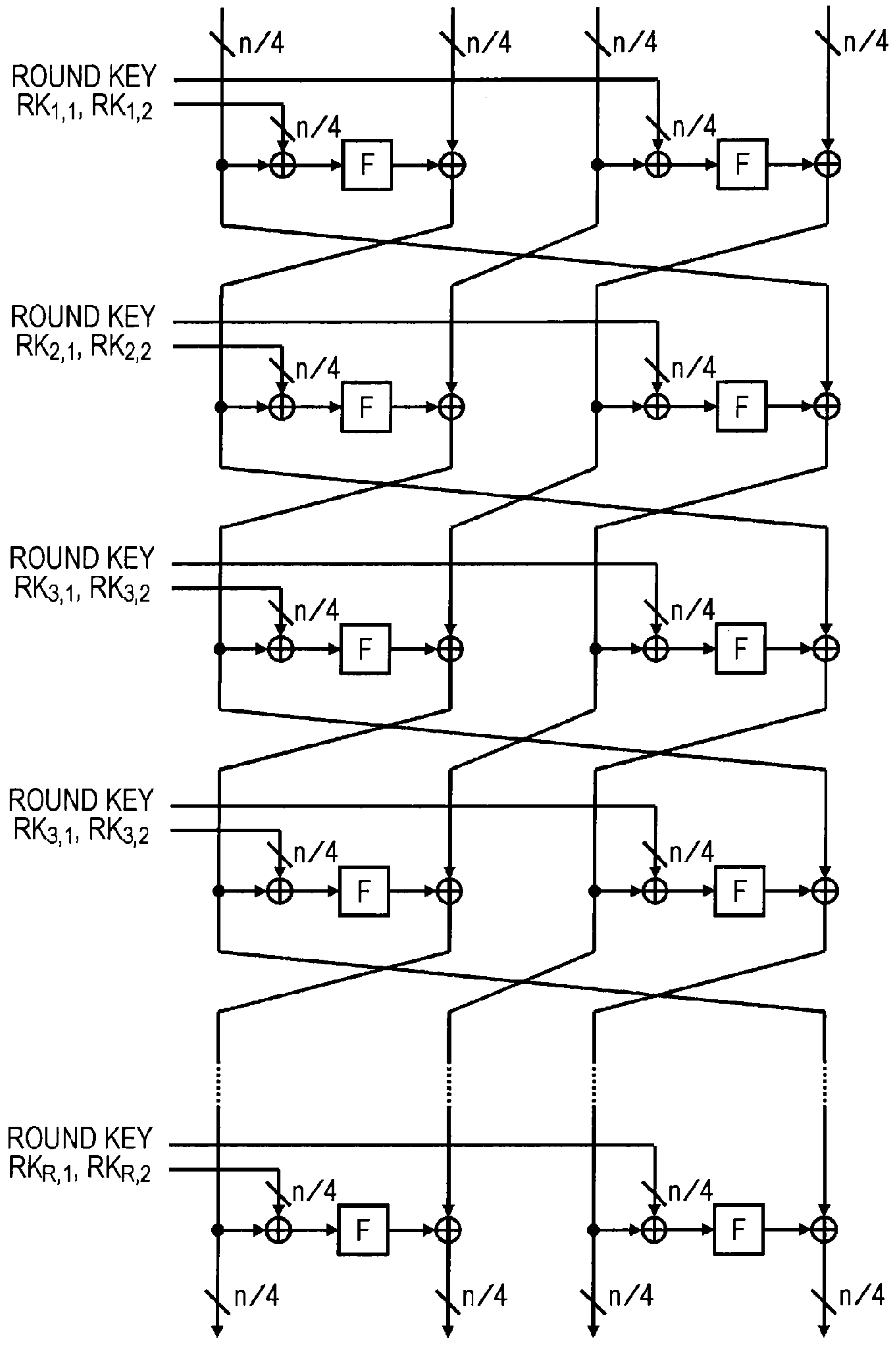

n/4
ROUND KEY
RK1,1, RK1,2
F
ROUND KEY
RK2,1, RK2,2
ROUND KEY
RK3,1, RK3,2
ROUND KEY
RK3,1, RK3,2
ROUND KEY
RKR,1, RKR,2

FIG. 19

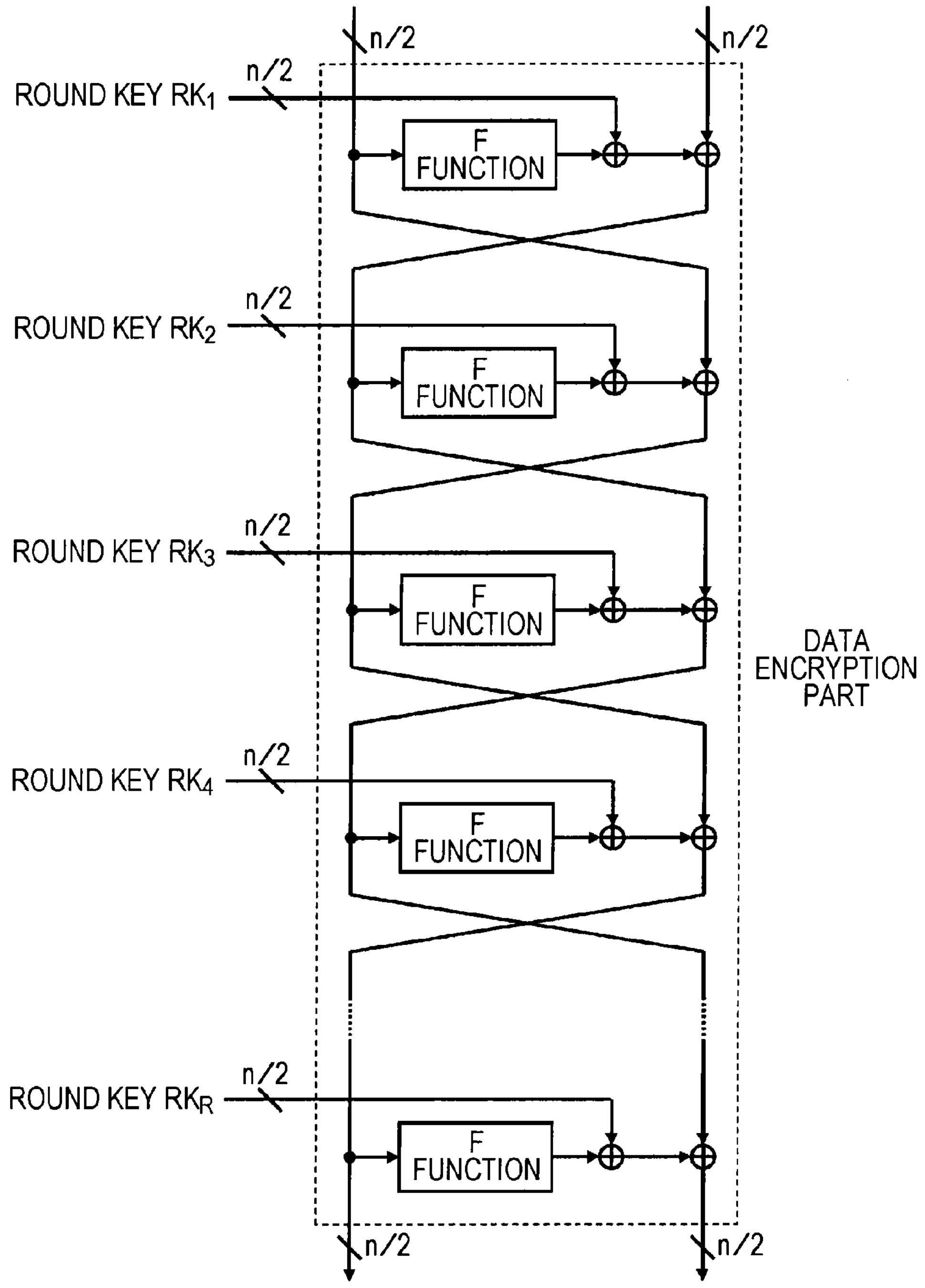

n/2
n/2
ROUND KEY RK_1
n/2
F FUNCTION
ROUND KEY RK_2
n/2
F FUNCTION
ROUND KEY RK_3
n/2
F FUNCTION
DATA ENCRYPTION PART
ROUND KEY RK_4
n/2
F FUNCTION
ROUND KEY RK_R
n/2
F FUNCTION
n/2
n/2

FIG. 20

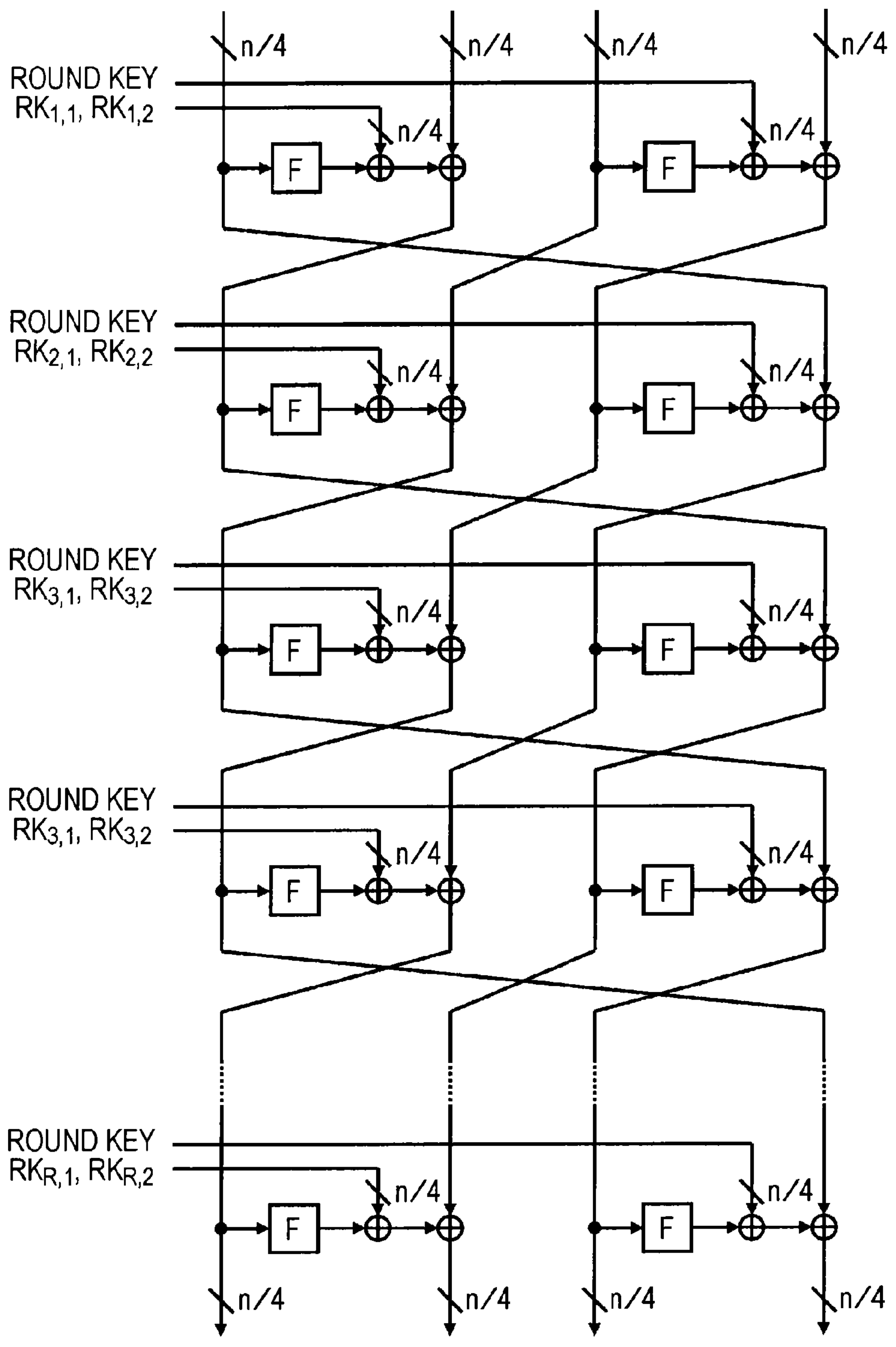

n/4
ROUND KEY $RK_{1,1}$, $RK_{1,2}$
ROUND KEY $RK_{2,1}$, $RK_{2,2}$
ROUND KEY $RK_{3,1}$, $RK_{3,2}$
ROUND KEY $RK_{3,1}$, $RK_{3,2}$
ROUND KEY $RK_{R,1}$, $RK_{R,2}$
F

FIG. 21

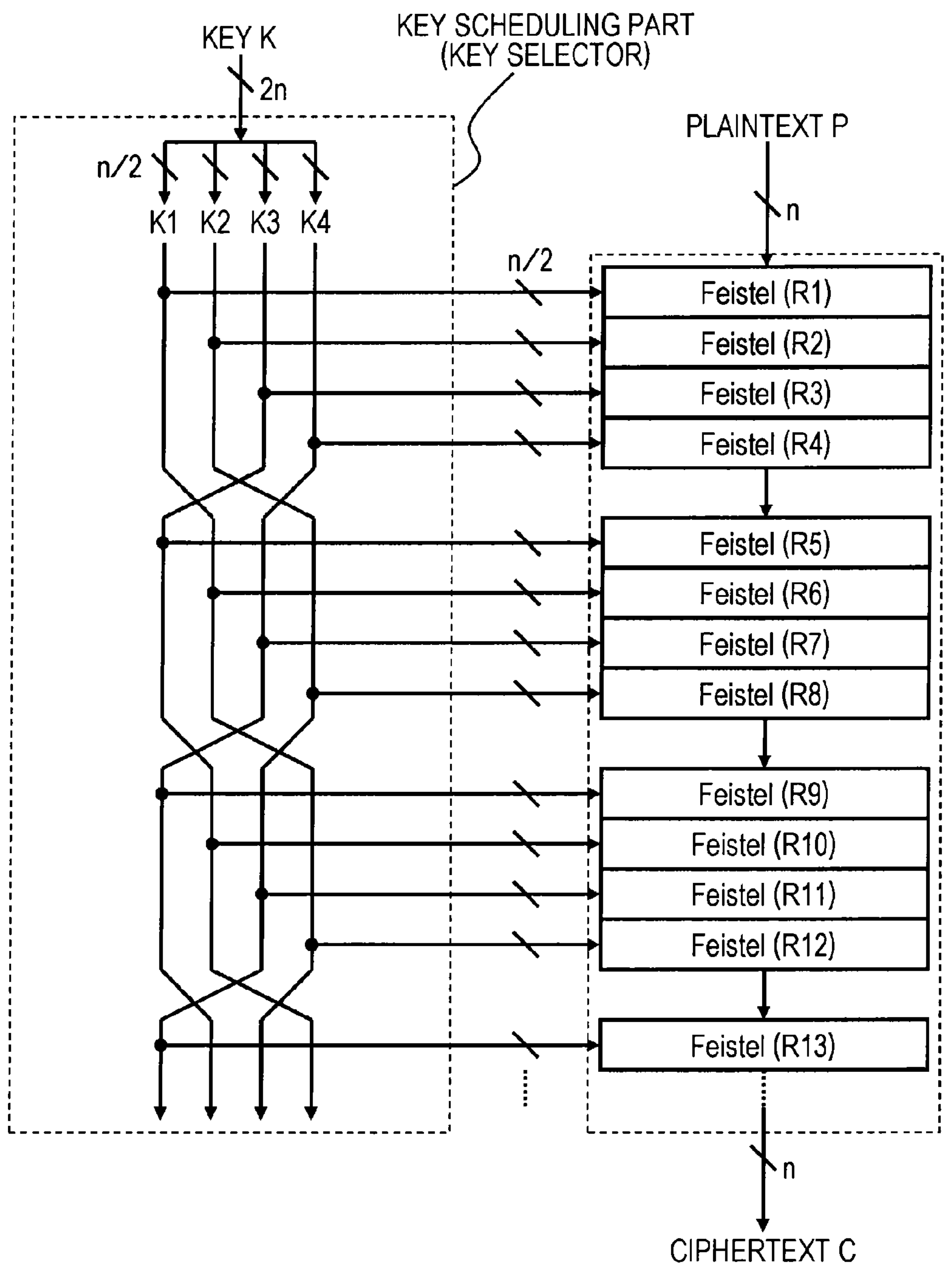

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/2
K1
K2
K3
K4
n/2
Feistel (R1)
Feistel (R2)
Feistel (R3)
Feistel (R4)
Feistel (R5)
Feistel (R6)
Feistel (R7)
Feistel (R8)
Feistel (R9)
Feistel (R10)
Feistel (R11)
Feistel (R12)
Feistel (R13)
n
CIPHERTEXT C

FIG. 22

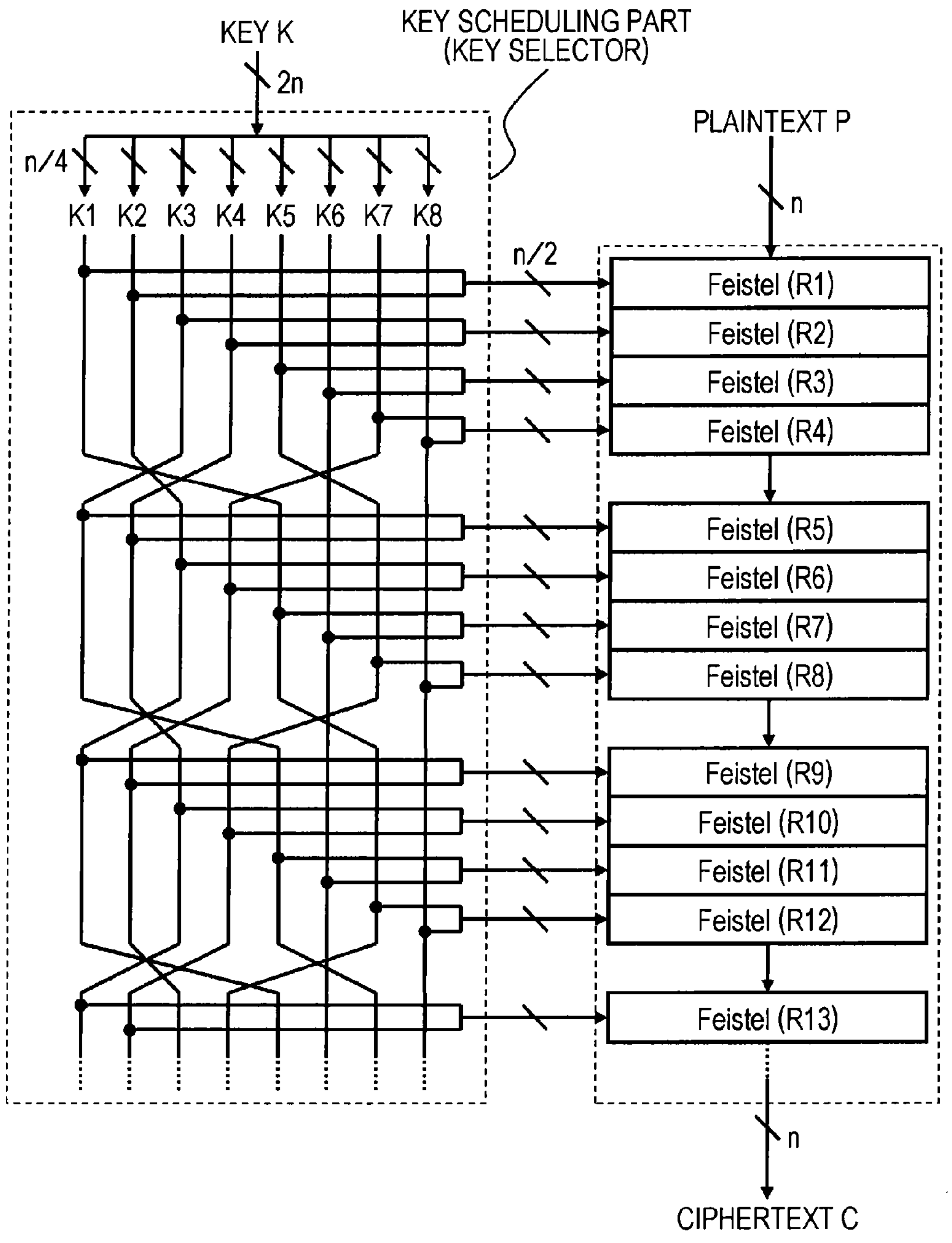

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/2
Feistel (R1)
Feistel (R2)
Feistel (R3)
Feistel (R4)
Feistel (R5)
Feistel (R6)
Feistel (R7)
Feistel (R8)
Feistel (R9)
Feistel (R10)
Feistel (R11)
Feistel (R12)
Feistel (R13)
n
CIPHERTEXT C

FIG. 23

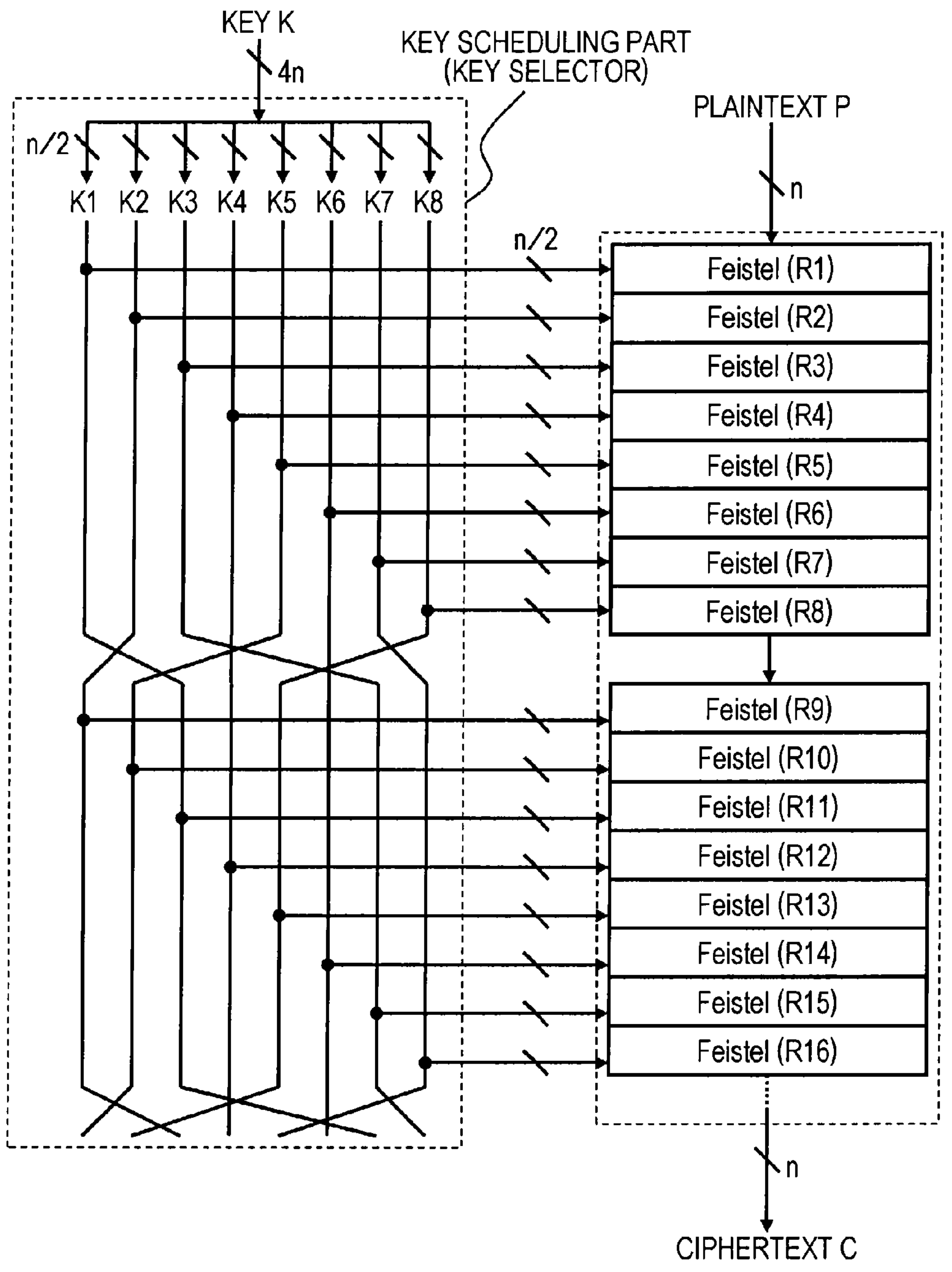

KEY K
4n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n/2
K1 K2 K3 K4 K5 K6 K7 K8
n
n/2
Feistel (R1)
Feistel (R2)
Feistel (R3)
Feistel (R4)
Feistel (R5)
Feistel (R6)
Feistel (R7)
Feistel (R8)
Feistel (R9)
Feistel (R10)
Feistel (R11)
Feistel (R12)
Feistel (R13)
Feistel (R14)
Feistel (R15)
Feistel (R16)
n
CIPHERTEXT C

FIG. 24

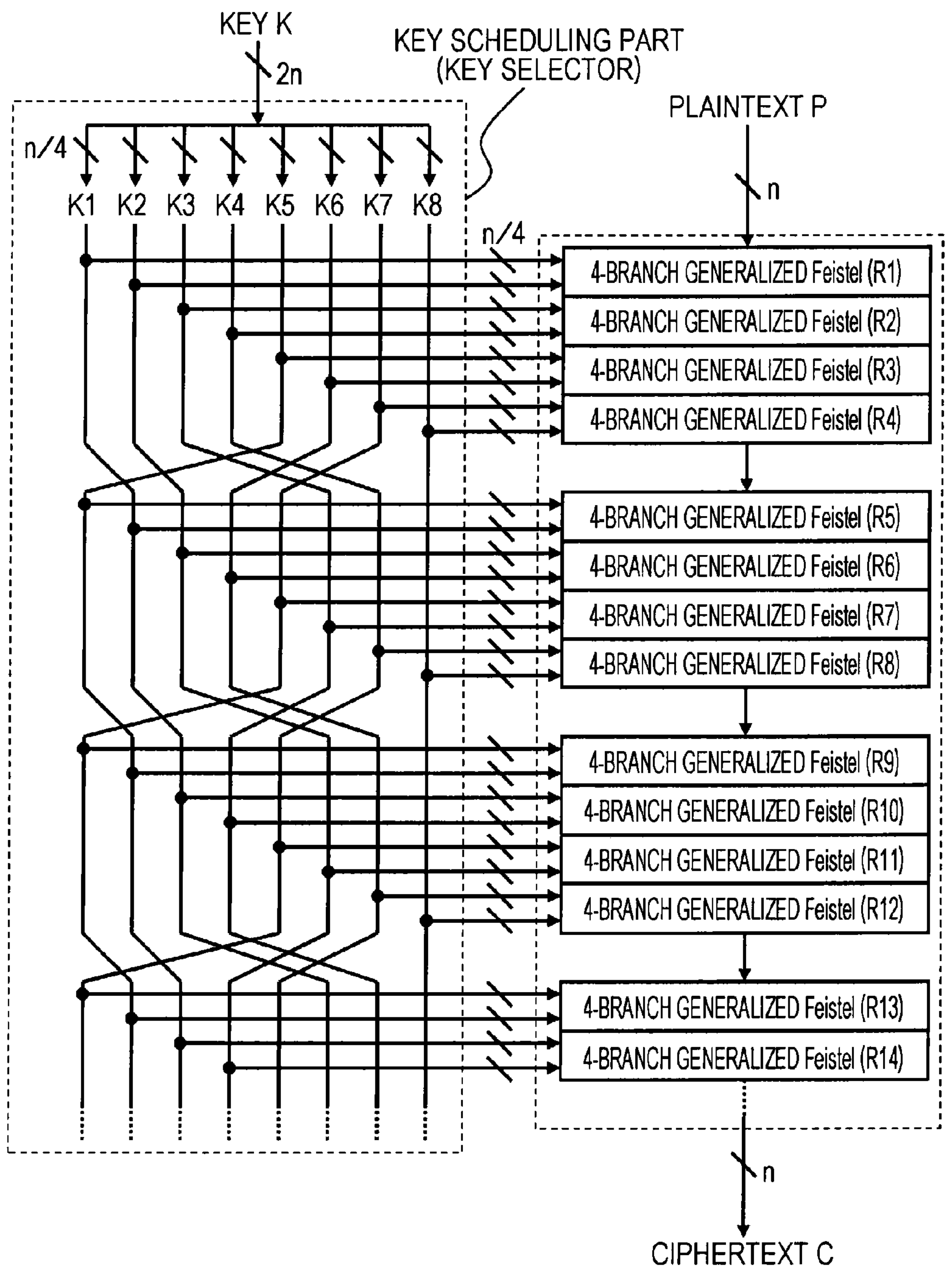

KEY K
2n
KEY SCHEDULING PART (KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 25

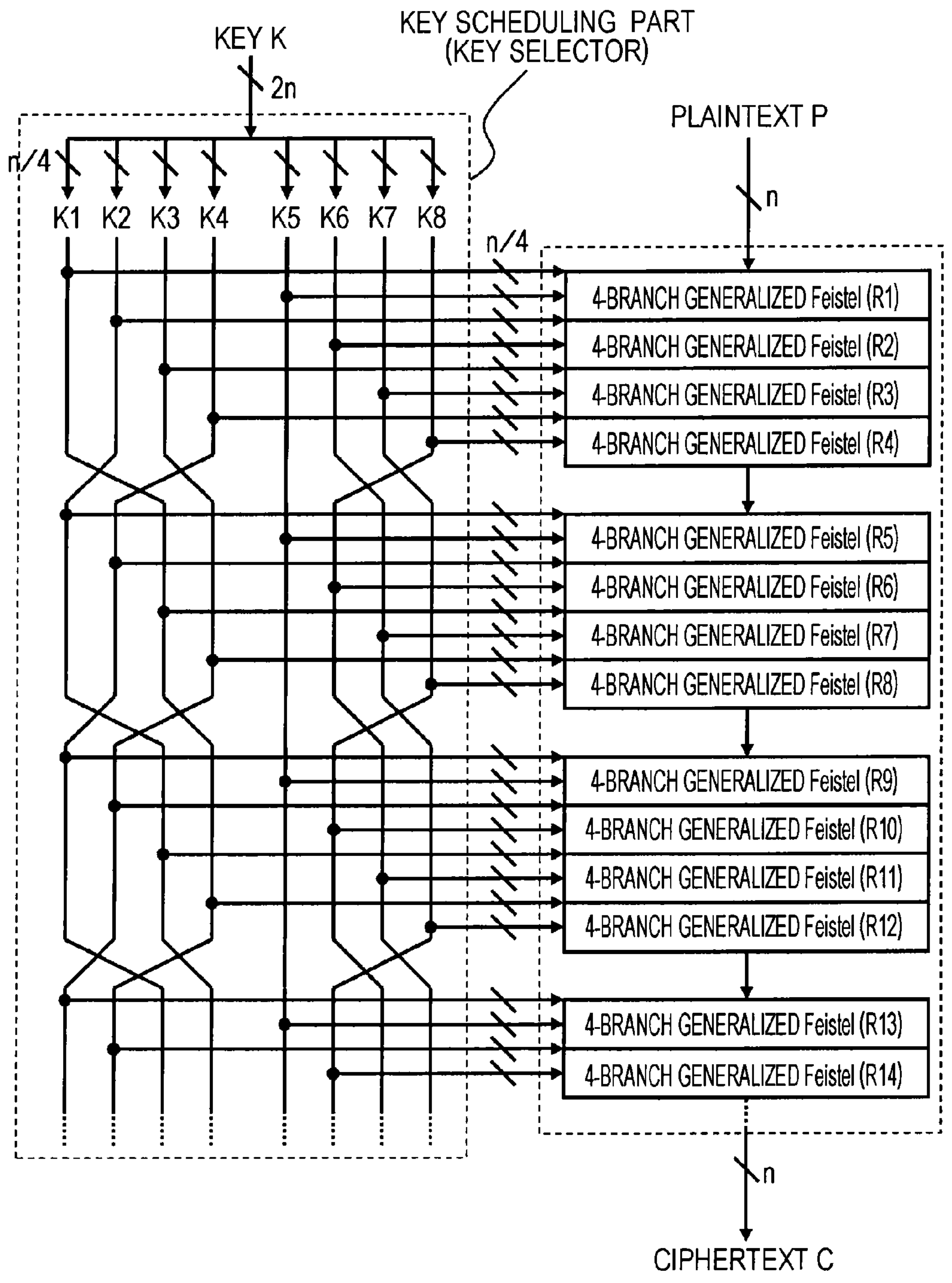

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1
K2
K3
K4
K5
K6
K7
K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 26

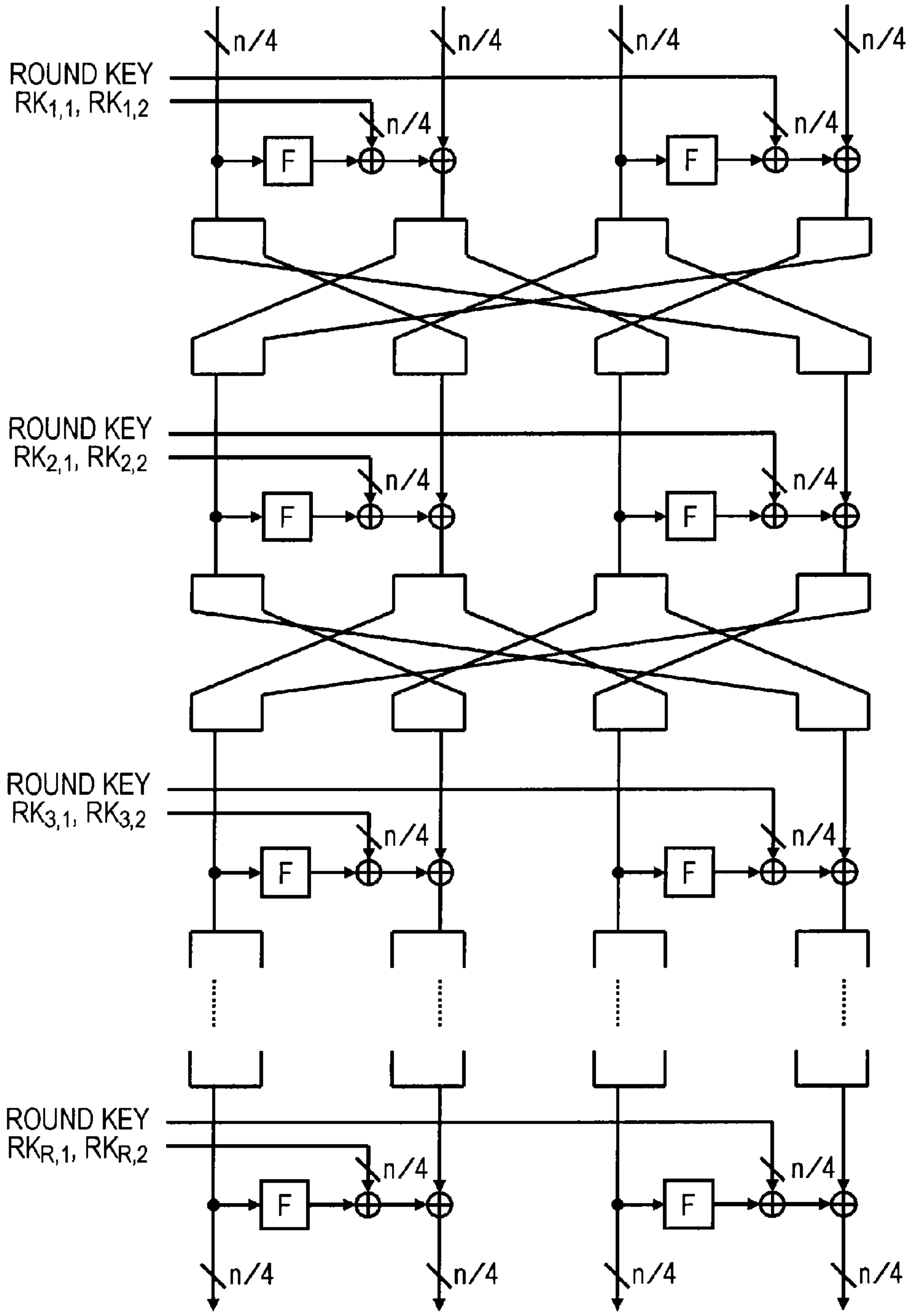

n/4
ROUND KEY
$RK_{1,1}$, $RK_{1,2}$
F
ROUND KEY
$RK_{2,1}$, $RK_{2,2}$
ROUND KEY
$RK_{3,1}$, $RK_{3,2}$
ROUND KEY
$RK_{R,1}$, $RK_{R,2}$

FIG. 27

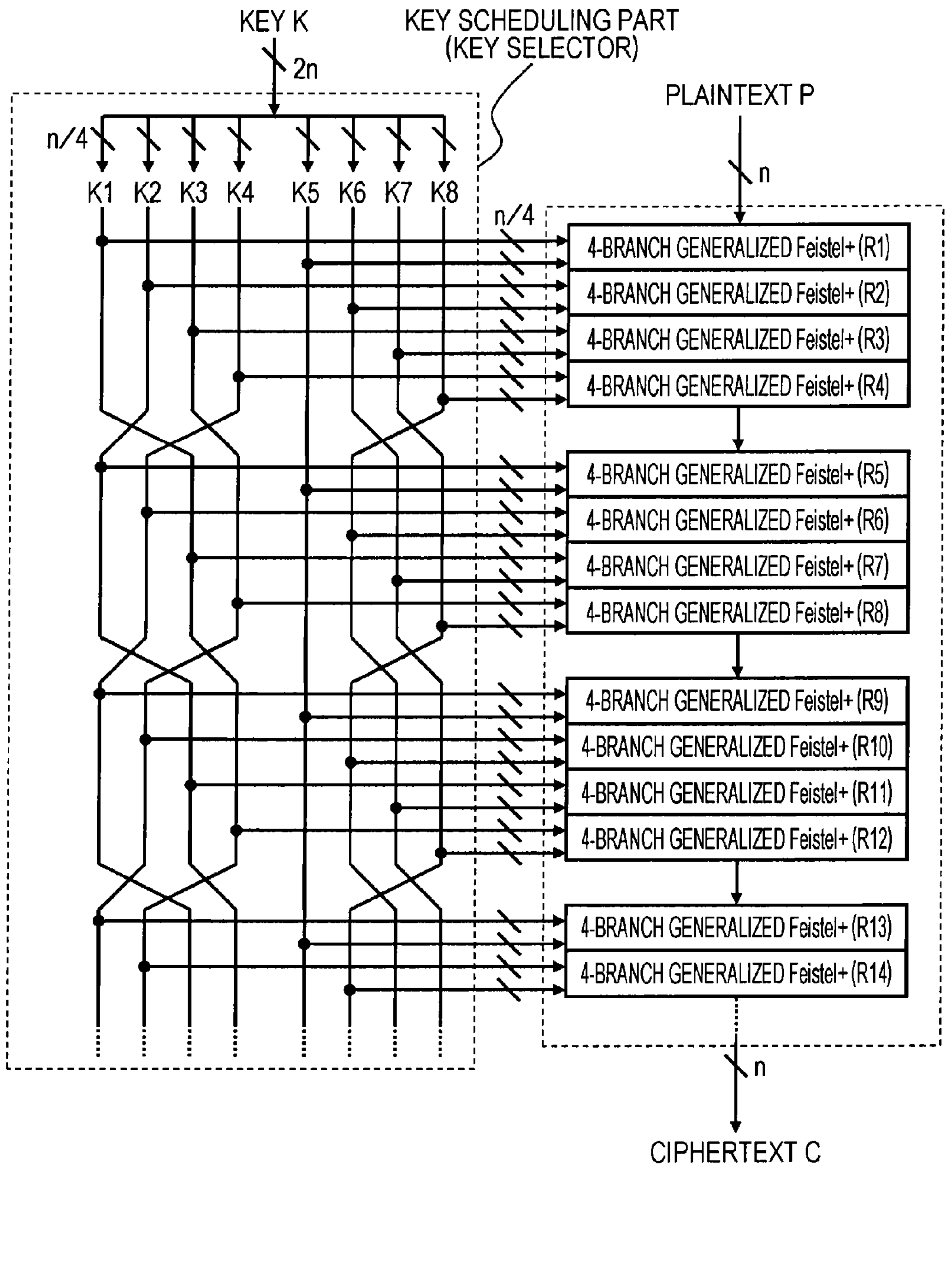

KEY K
2n
KEY SCHEDULING PART (KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel+ (R1)
4-BRANCH GENERALIZED Feistel+ (R2)
4-BRANCH GENERALIZED Feistel+ (R3)
4-BRANCH GENERALIZED Feistel+ (R4)
4-BRANCH GENERALIZED Feistel+ (R5)
4-BRANCH GENERALIZED Feistel+ (R6)
4-BRANCH GENERALIZED Feistel+ (R7)
4-BRANCH GENERALIZED Feistel+ (R8)
4-BRANCH GENERALIZED Feistel+ (R9)
4-BRANCH GENERALIZED Feistel+ (R10)
4-BRANCH GENERALIZED Feistel+ (R11)
4-BRANCH GENERALIZED Feistel+ (R12)
4-BRANCH GENERALIZED Feistel+ (R13)
4-BRANCH GENERALIZED Feistel+ (R14)
n
CIPHERTEXT C

FIG. 28

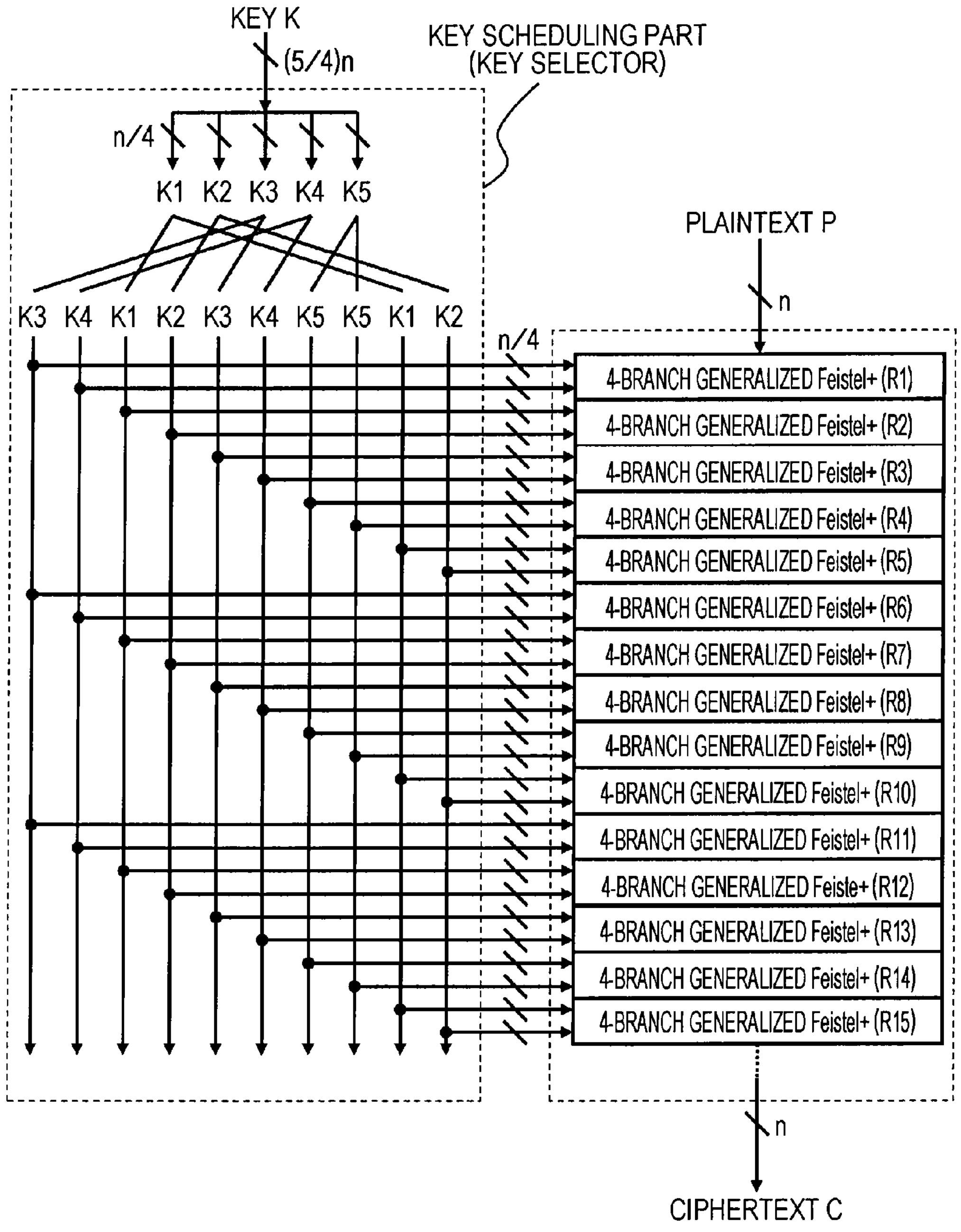

KEY K
(5/4)n
KEY SCHEDULING PART
(KEY SELECTOR)
n/4
K1 K2 K3 K4 K5
PLAINTEXT P
K3 K4 K1 K2 K3 K4 K5 K5 K1 K2
n
n/4
4-BRANCH GENERALIZED Feistel+ (R1)
4-BRANCH GENERALIZED Feistel+ (R2)
4-BRANCH GENERALIZED Feistel+ (R3)
4-BRANCH GENERALIZED Feistel+ (R4)
4-BRANCH GENERALIZED Feistel+ (R5)
4-BRANCH GENERALIZED Feistel+ (R6)
4-BRANCH GENERALIZED Feistel+ (R7)
4-BRANCH GENERALIZED Feistel+ (R8)
4-BRANCH GENERALIZED Feistel+ (R9)
4-BRANCH GENERALIZED Feistel+ (R10)
4-BRANCH GENERALIZED Feistel+ (R11)
4-BRANCH GENERALIZED Feiste+ (R12)
4-BRANCH GENERALIZED Feistel+ (R13)
4-BRANCH GENERALIZED Feistel+ (R14)
4-BRANCH GENERALIZED Feistel+ (R15)
n
CIPHERTEXT C

FIG. 29

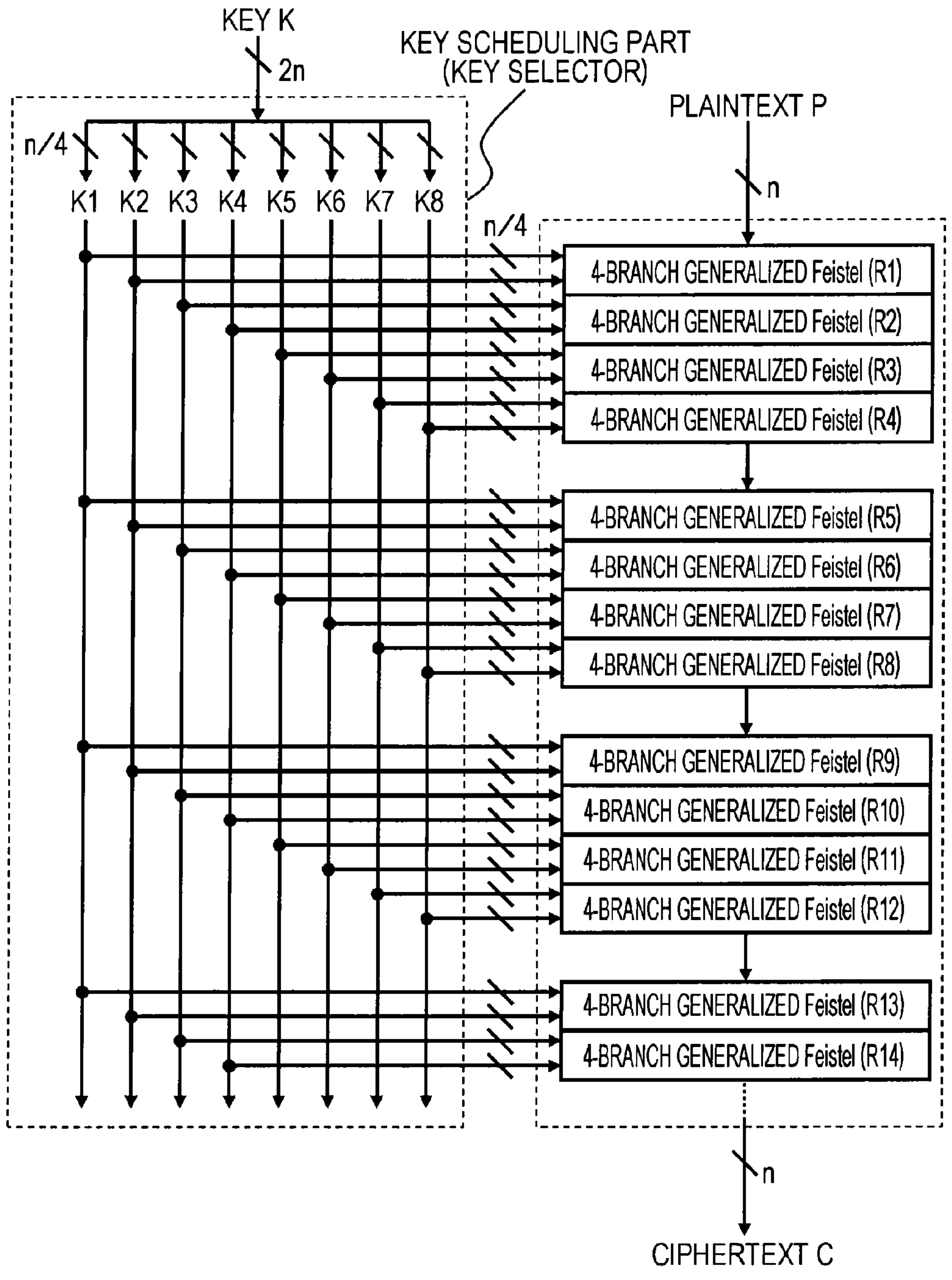

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 30

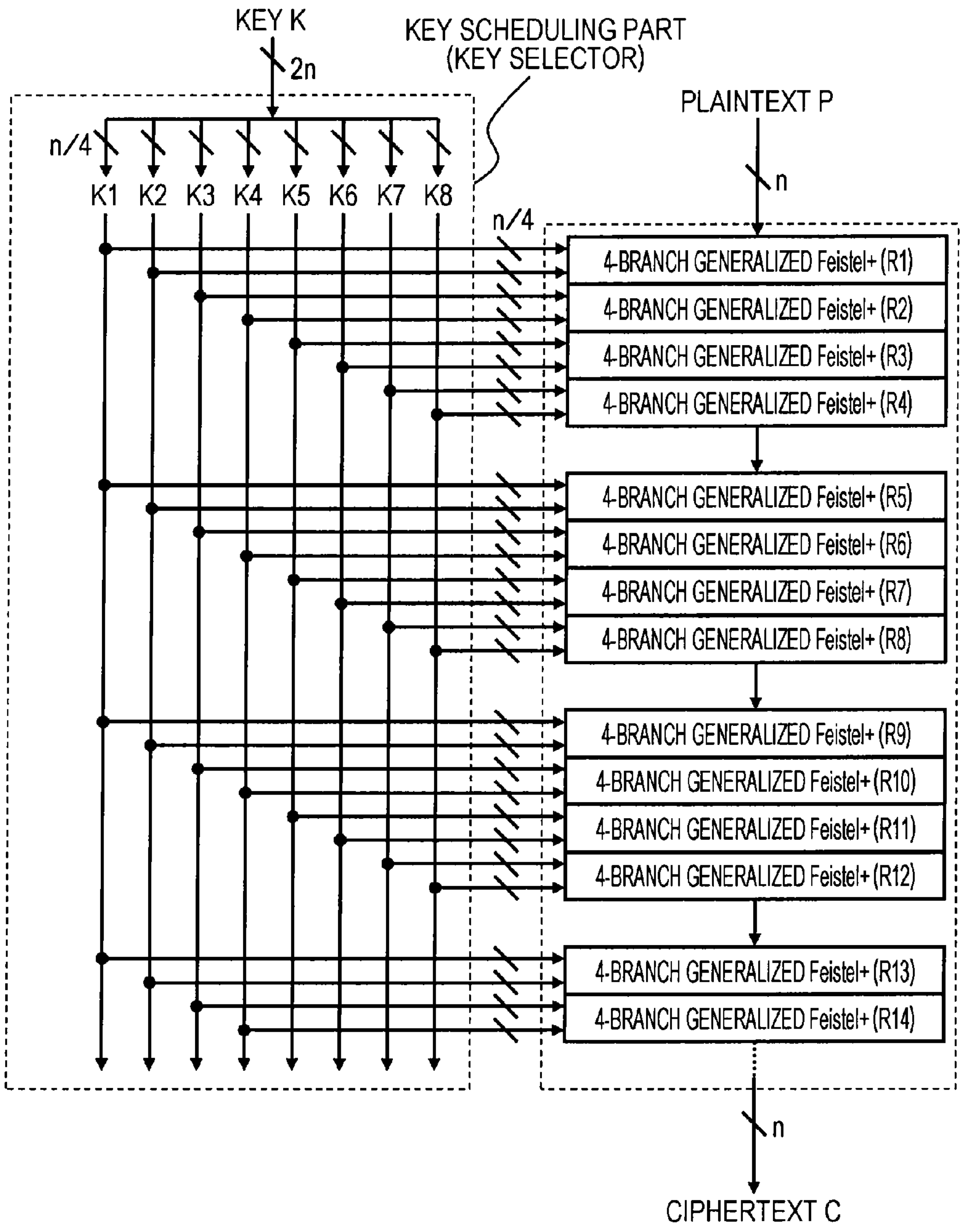

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel+ (R1)
4-BRANCH GENERALIZED Feistel+ (R2)
4-BRANCH GENERALIZED Feistel+ (R3)
4-BRANCH GENERALIZED Feistel+ (R4)
4-BRANCH GENERALIZED Feistel+ (R5)
4-BRANCH GENERALIZED Feistel+ (R6)
4-BRANCH GENERALIZED Feistel+ (R7)
4-BRANCH GENERALIZED Feistel+ (R8)
4-BRANCH GENERALIZED Feistel+ (R9)
4-BRANCH GENERALIZED Feistel+ (R10)
4-BRANCH GENERALIZED Feistel+ (R11)
4-BRANCH GENERALIZED Feistel+ (R12)
4-BRANCH GENERALIZED Feistel+ (R13)
4-BRANCH GENERALIZED Feistel+ (R14)
n
CIPHERTEXT C

FIG. 31

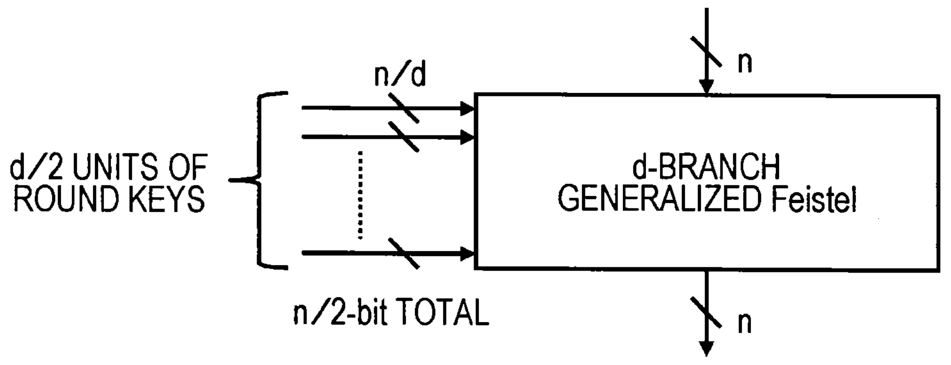

n
d/2 UNITS OF ROUND KEYS
n/d
d-BRANCH GENERALIZED Feistel
n/2-bit TOTAL
n

FIG. 32

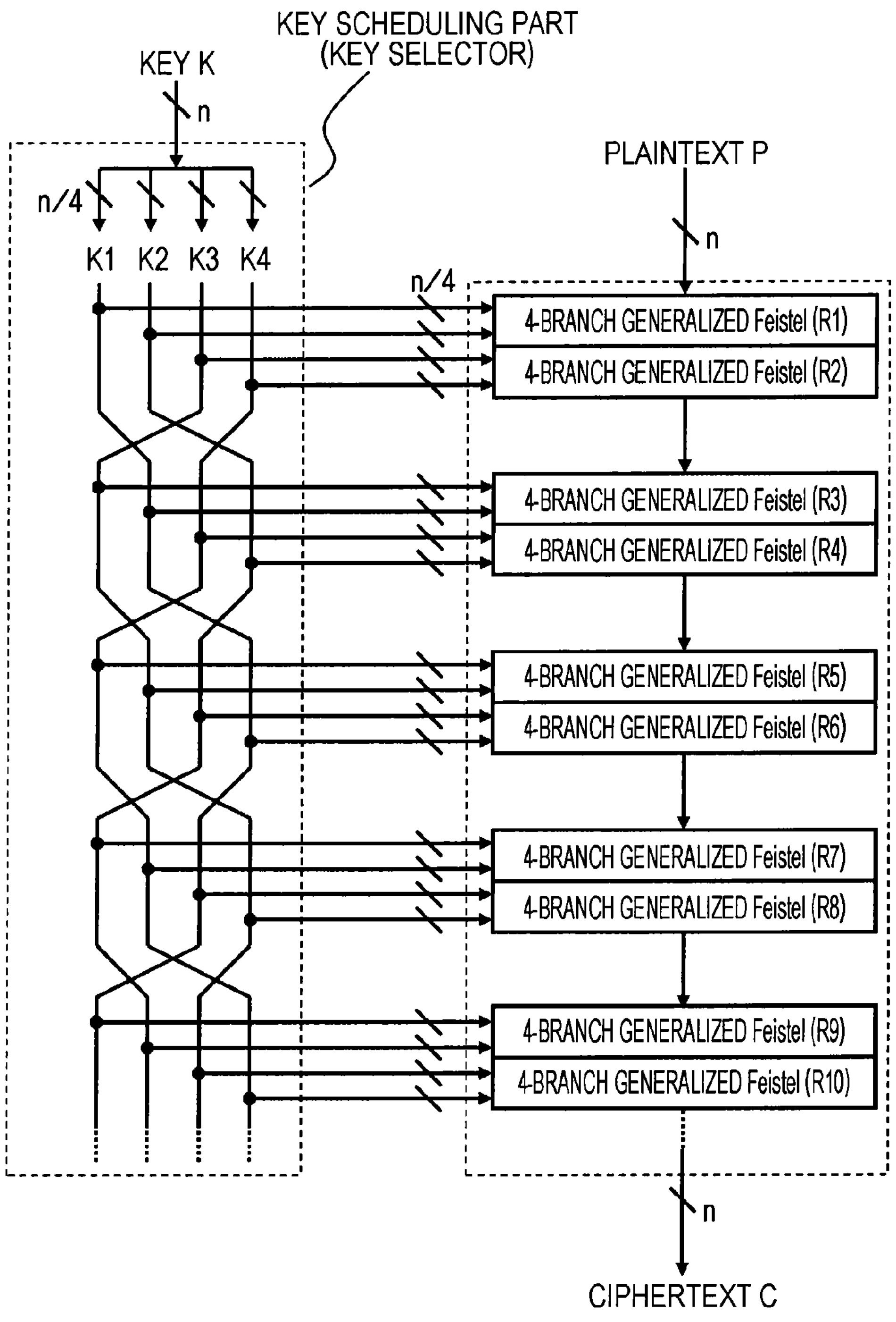

KEY SCHEDULING PART
(KEY SELECTOR)
KEY K
n
n/4
K1
K2
K3
K4
PLAINTEXT P
n
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
n
CIPHERTEXT C

FIG. 33

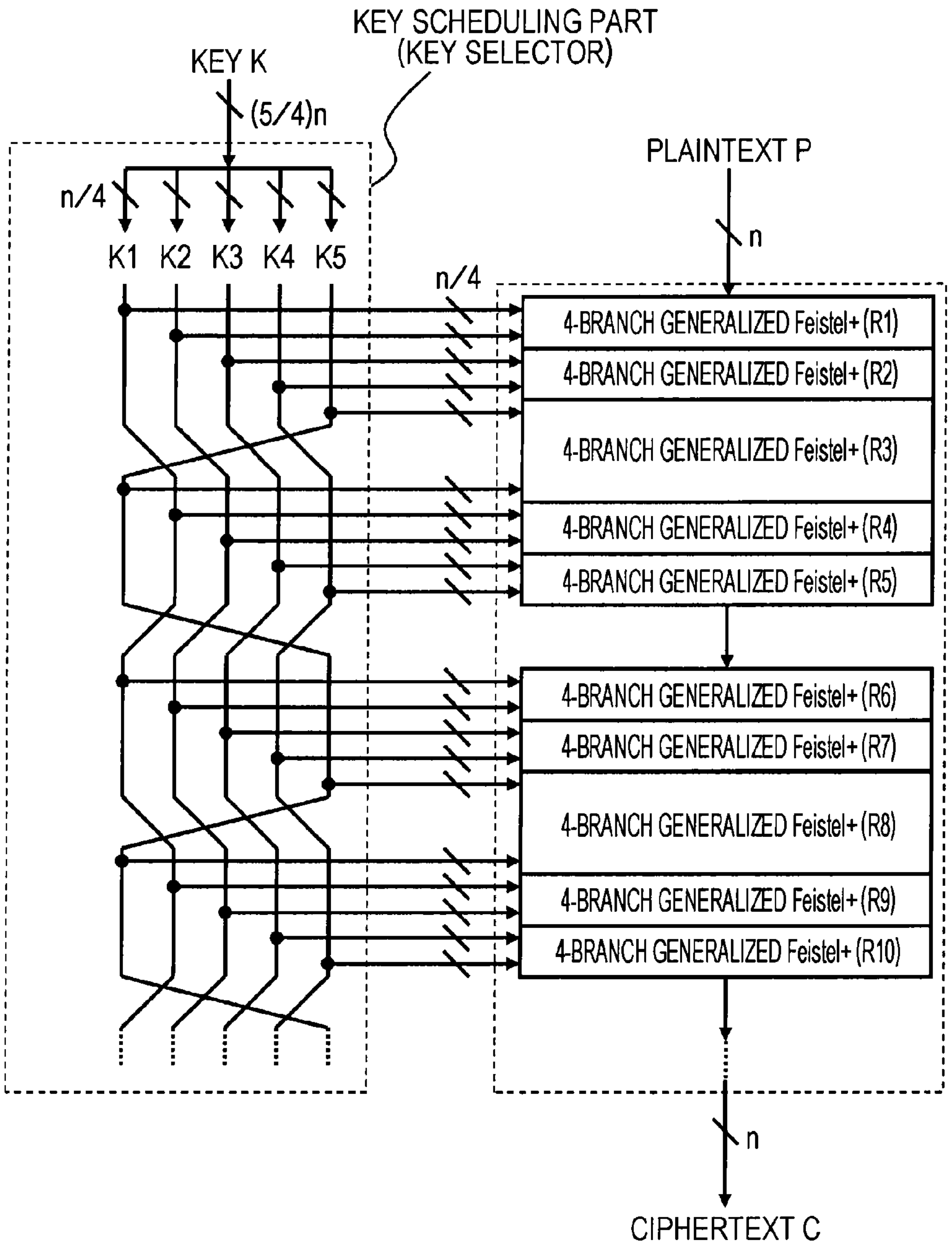

KEY SCHEDULING PART
(KEY SELECTOR)
KEY K
(5/4)n
n/4
K1
K2
K3
K4
K5
n/4
PLAINTEXT P
n
4-BRANCH GENERALIZED Feistel+ (R1)
4-BRANCH GENERALIZED Feistel+ (R2)
4-BRANCH GENERALIZED Feistel+ (R3)
4-BRANCH GENERALIZED Feistel+ (R4)
4-BRANCH GENERALIZED Feistel+ (R5)
4-BRANCH GENERALIZED Feistel+ (R6)
4-BRANCH GENERALIZED Feistel+ (R7)
4-BRANCH GENERALIZED Feistel+ (R8)
4-BRANCH GENERALIZED Feistel+ (R9)
4-BRANCH GENERALIZED Feistel+ (R10)
n
CIPHERTEXT C

FIG. 34

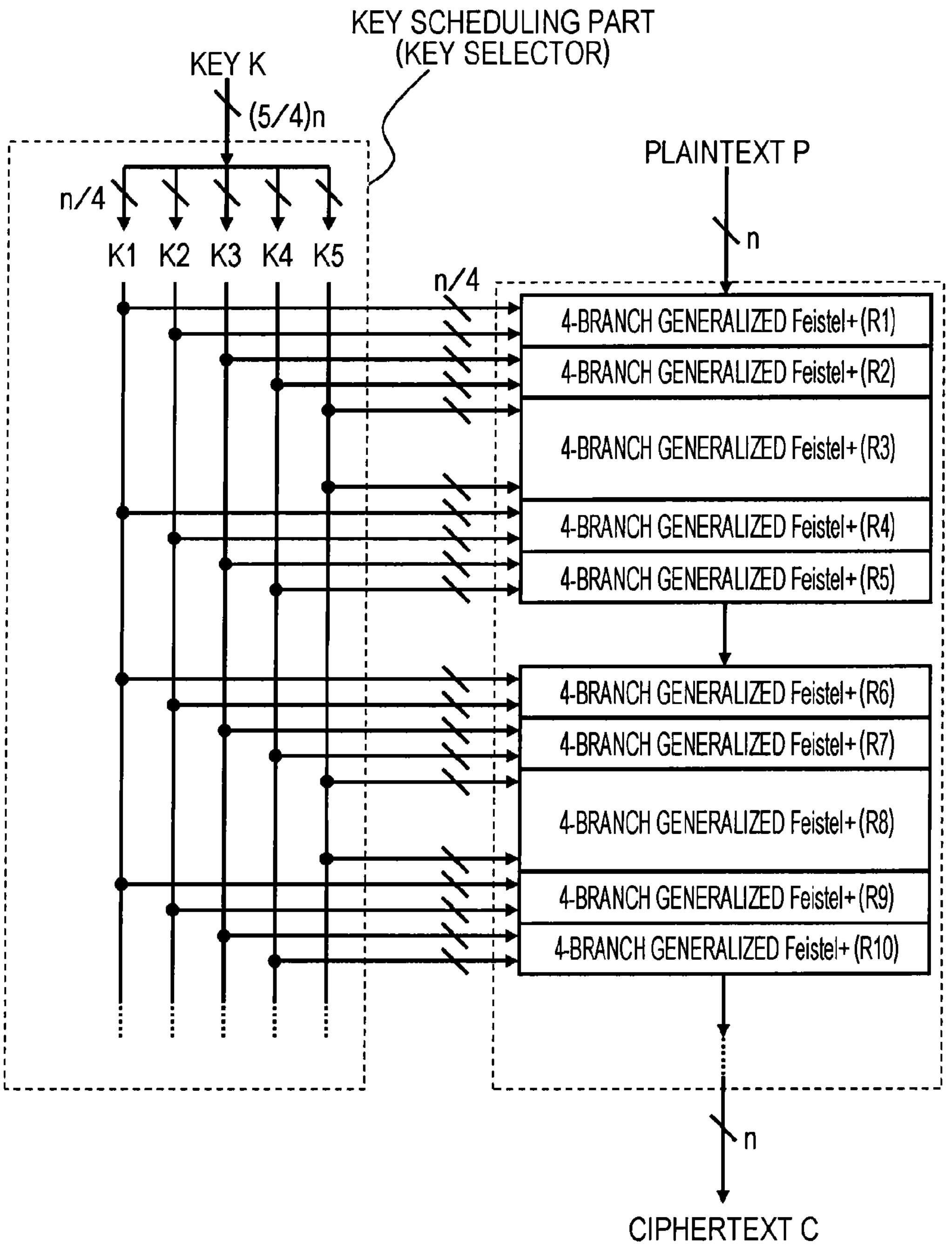

KEY SCHEDULING PART
(KEY SELECTOR)
KEY K
(5/4)n
n/4
K1 K2 K3 K4 K5
PLAINTEXT P
n
n/4
4-BRANCH GENERALIZED Feistel+(R1)
4-BRANCH GENERALIZED Feistel+(R2)
4-BRANCH GENERALIZED Feistel+(R3)
4-BRANCH GENERALIZED Feistel+(R4)
4-BRANCH GENERALIZED Feistel+(R5)
4-BRANCH GENERALIZED Feistel+(R6)
4-BRANCH GENERALIZED Feistel+(R7)
4-BRANCH GENERALIZED Feistel+(R8)
4-BRANCH GENERALIZED Feistel+(R9)
4-BRANCH GENERALIZED Feistel+(R10)
n
CIPHERTEXT C

FIG. 35

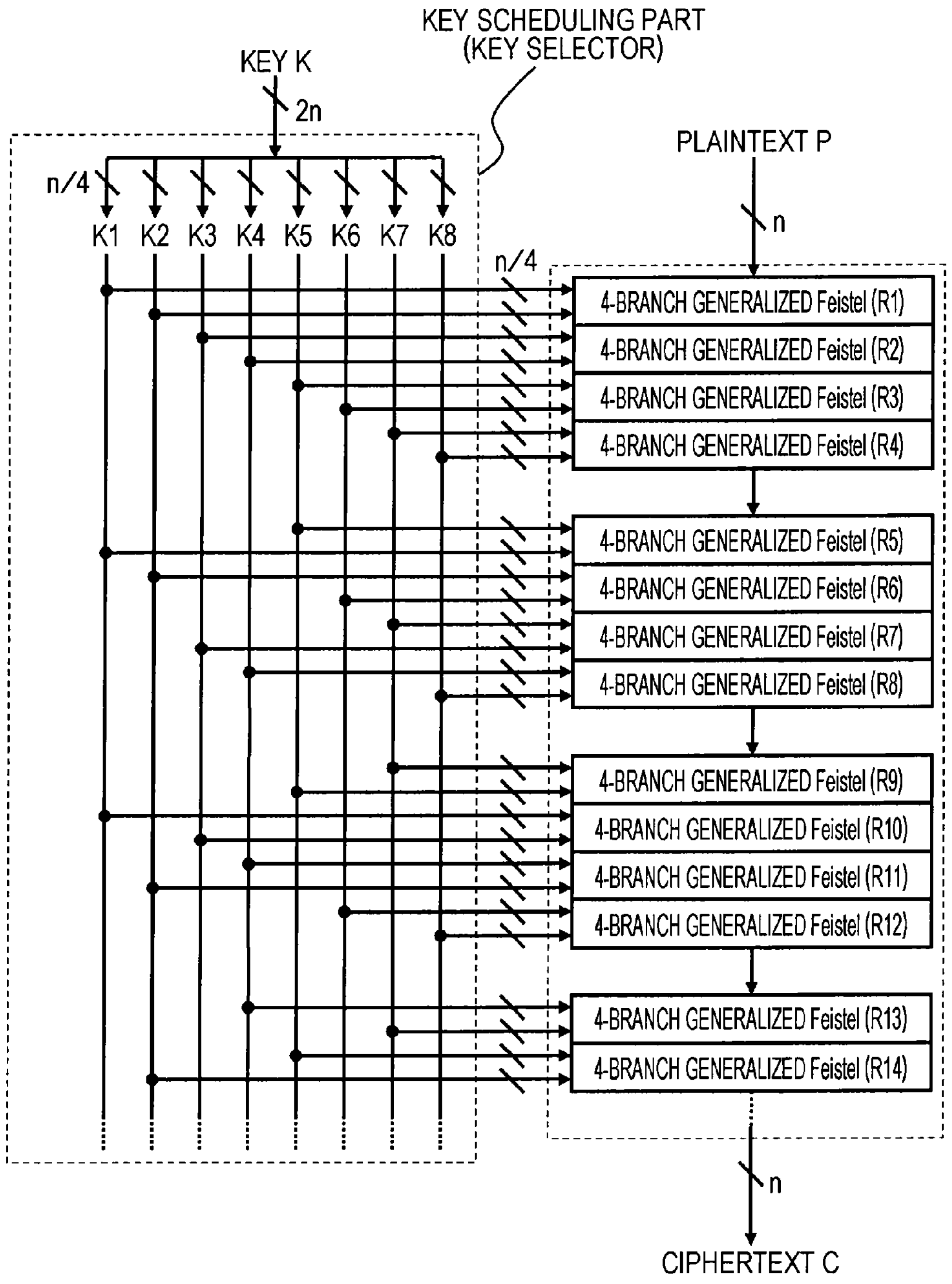

KEY SCHEDULING PART
(KEY SELECTOR)
KEY K
2n
n/4
K1
K2
K3
K4
K5
K6
K7
K8
PLAINTEXT P
n
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 36

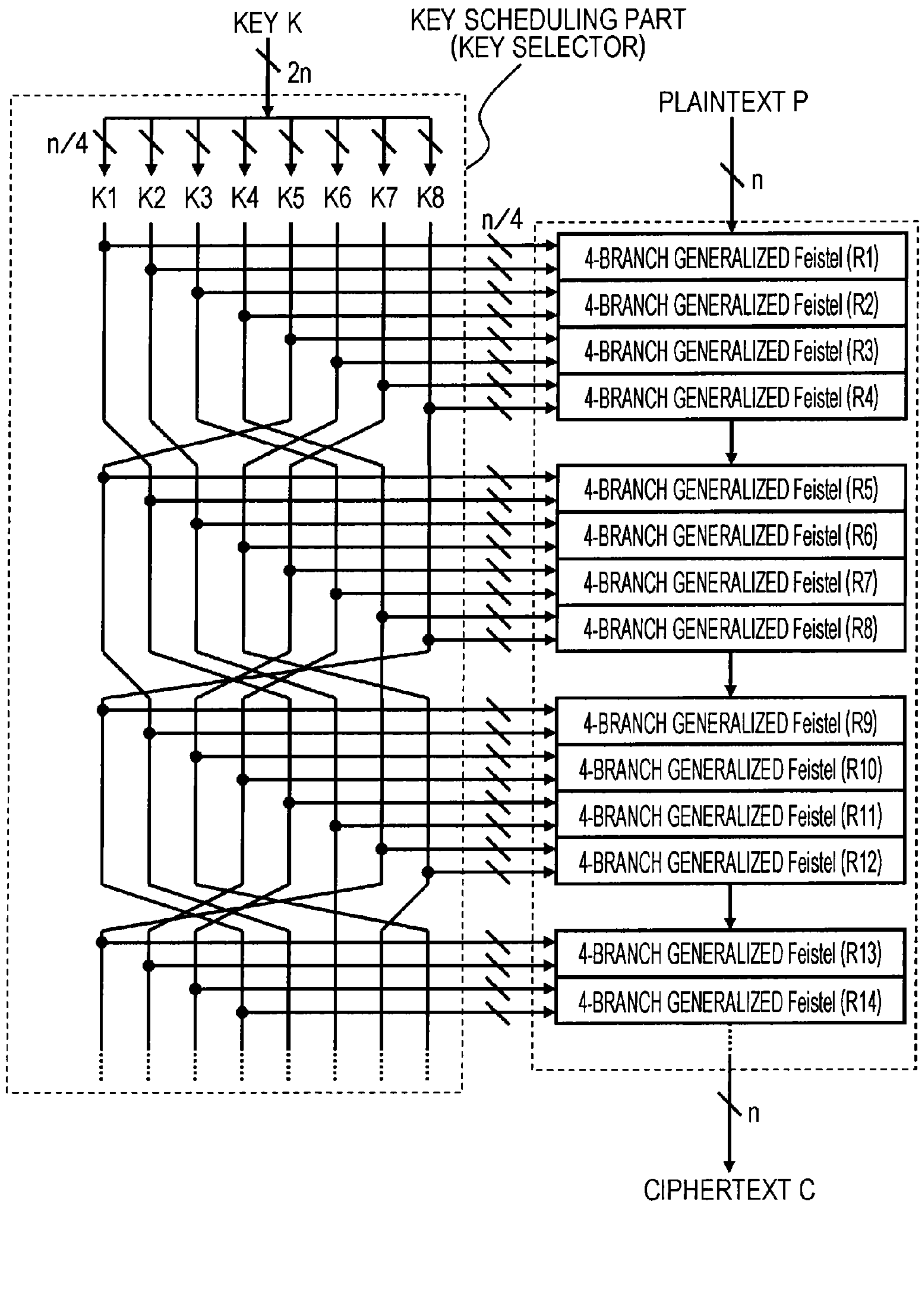

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 37

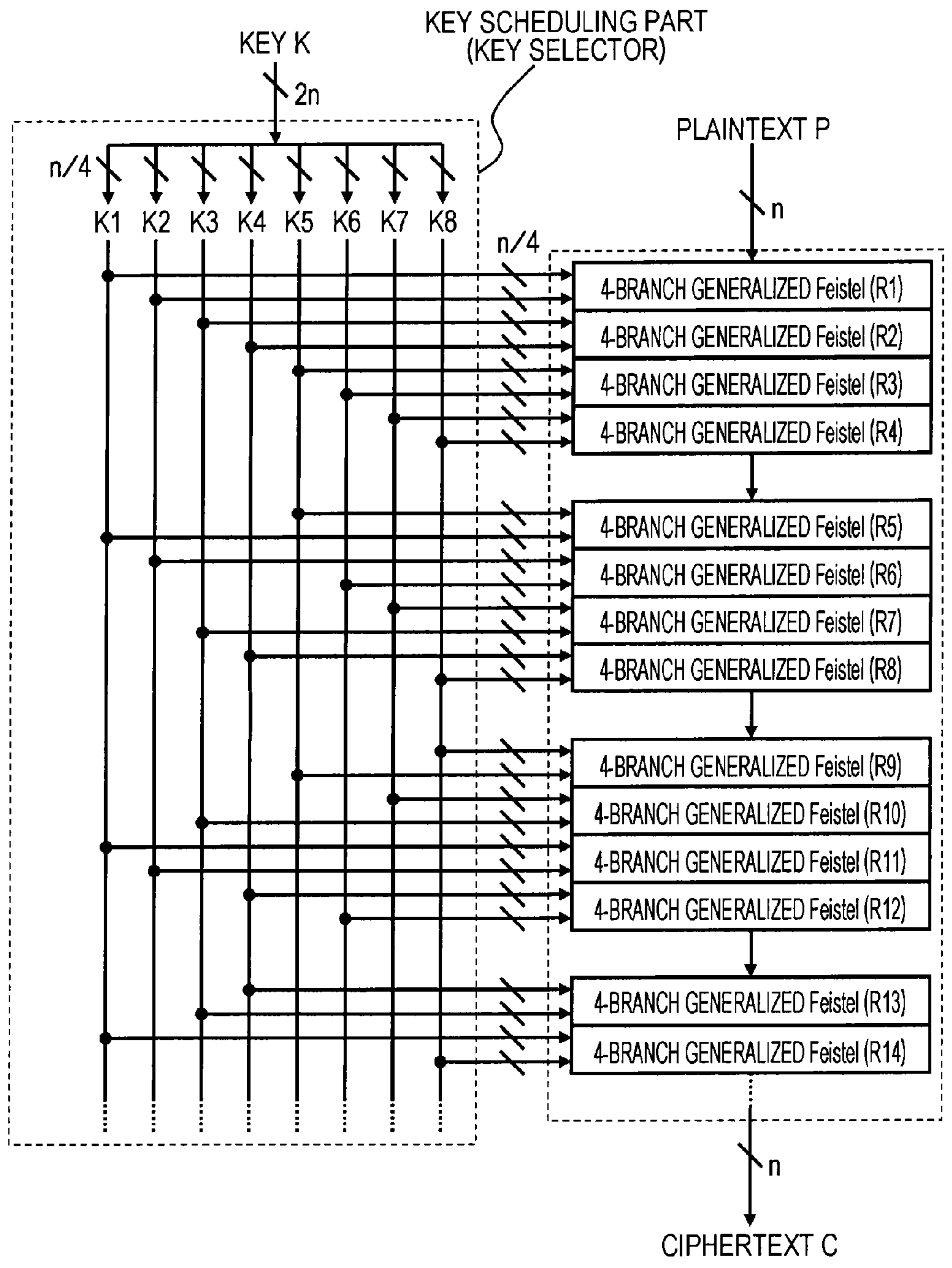

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n
n/4
K1
K2
K3
K4
K5
K6
K7
K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
4-BRANCH GENERALIZED Feistel (R13)
4-BRANCH GENERALIZED Feistel (R14)
n
CIPHERTEXT C

FIG. 38

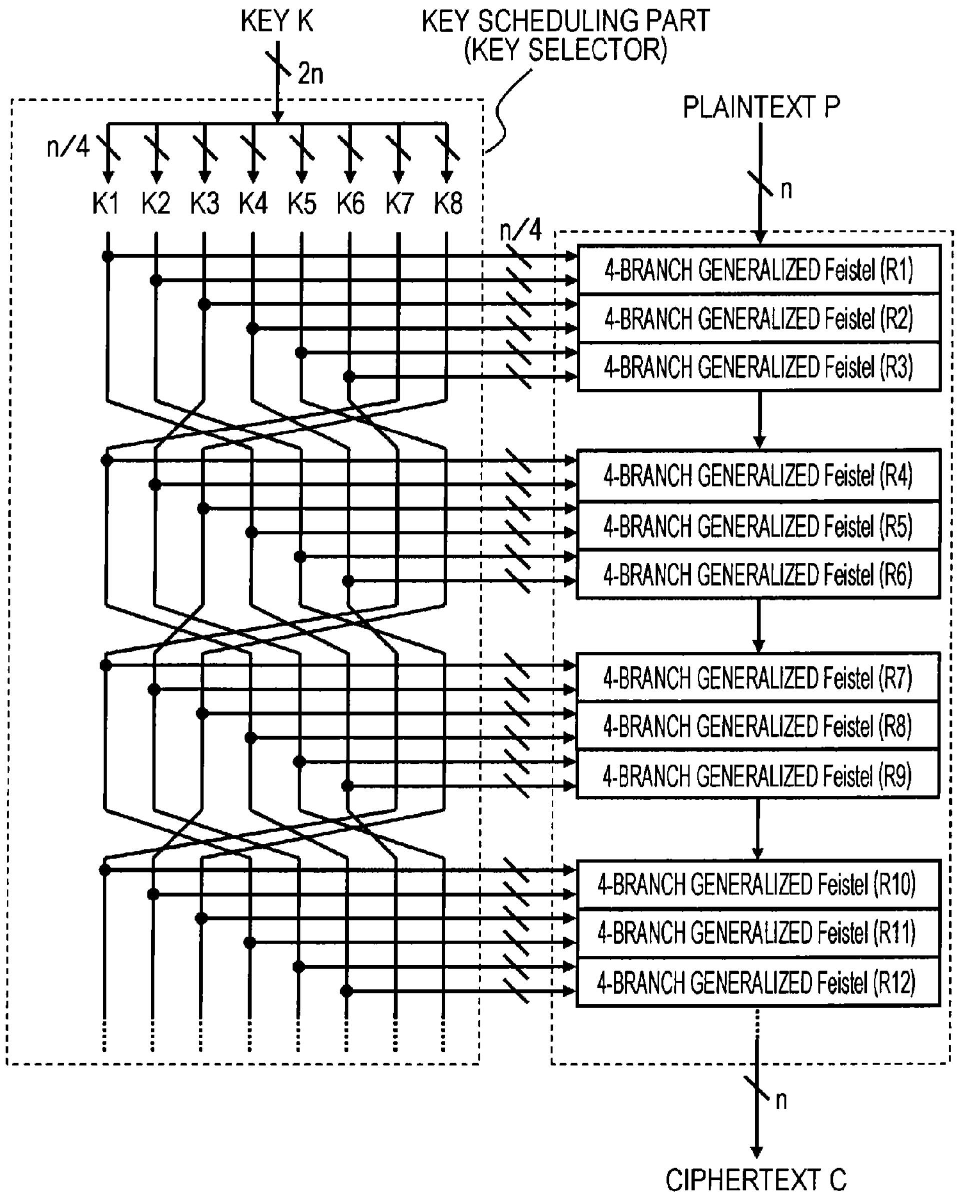

KEY K
2n
KEY SCHEDULING PART (KEY SELECTOR)
PLAINTEXT P
n
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
n
CIPHERTEXT C

FIG. 39

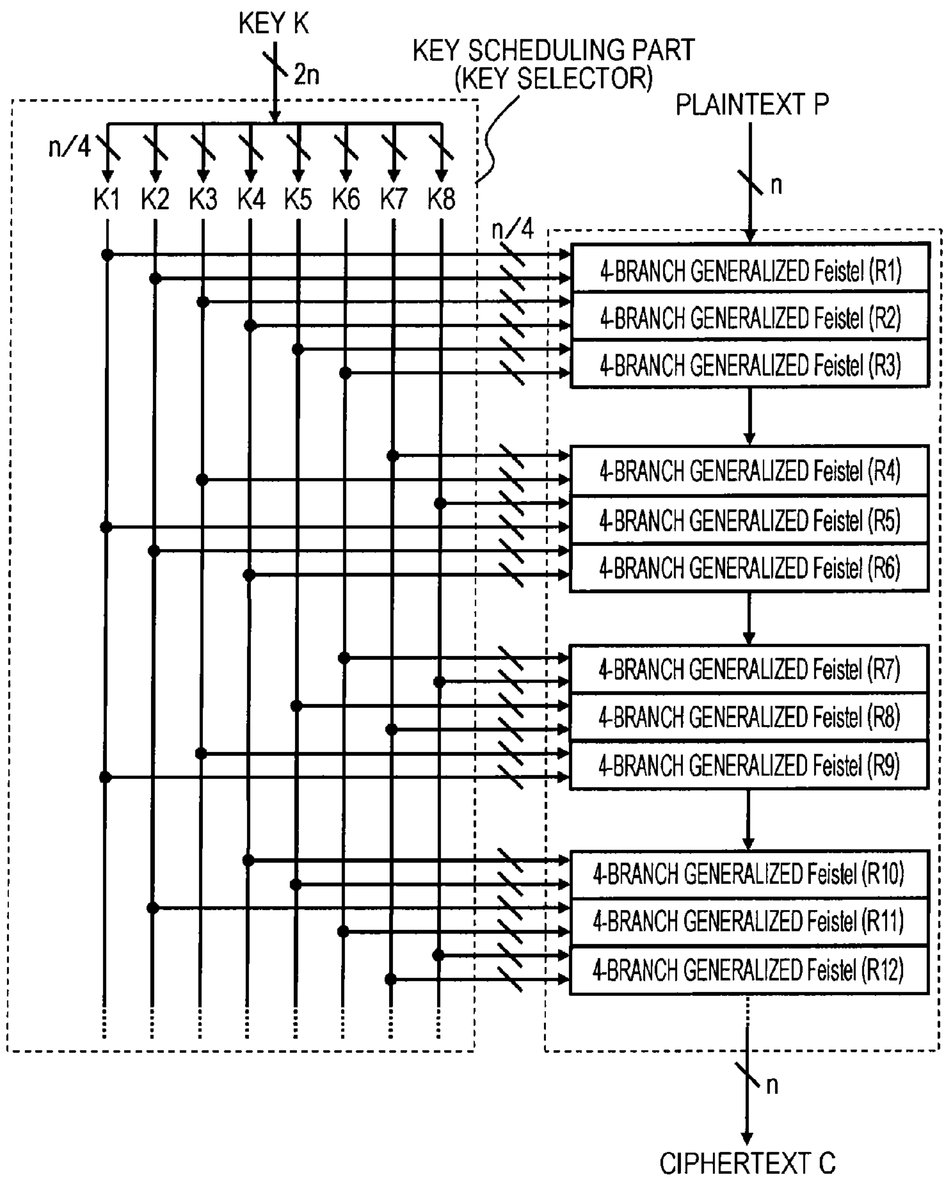

KEY K
2n
KEY SCHEDULING PART
(KEY SELECTOR)
PLAINTEXT P
n/4
K1 K2 K3 K4 K5 K6 K7 K8
n
n/4
4-BRANCH GENERALIZED Feistel (R1)
4-BRANCH GENERALIZED Feistel (R2)
4-BRANCH GENERALIZED Feistel (R3)
4-BRANCH GENERALIZED Feistel (R4)
4-BRANCH GENERALIZED Feistel (R5)
4-BRANCH GENERALIZED Feistel (R6)
4-BRANCH GENERALIZED Feistel (R7)
4-BRANCH GENERALIZED Feistel (R8)
4-BRANCH GENERALIZED Feistel (R9)
4-BRANCH GENERALIZED Feistel (R10)
4-BRANCH GENERALIZED Feistel (R11)
4-BRANCH GENERALIZED Feistel (R12)
n
CIPHERTEXT C

FIG. 40

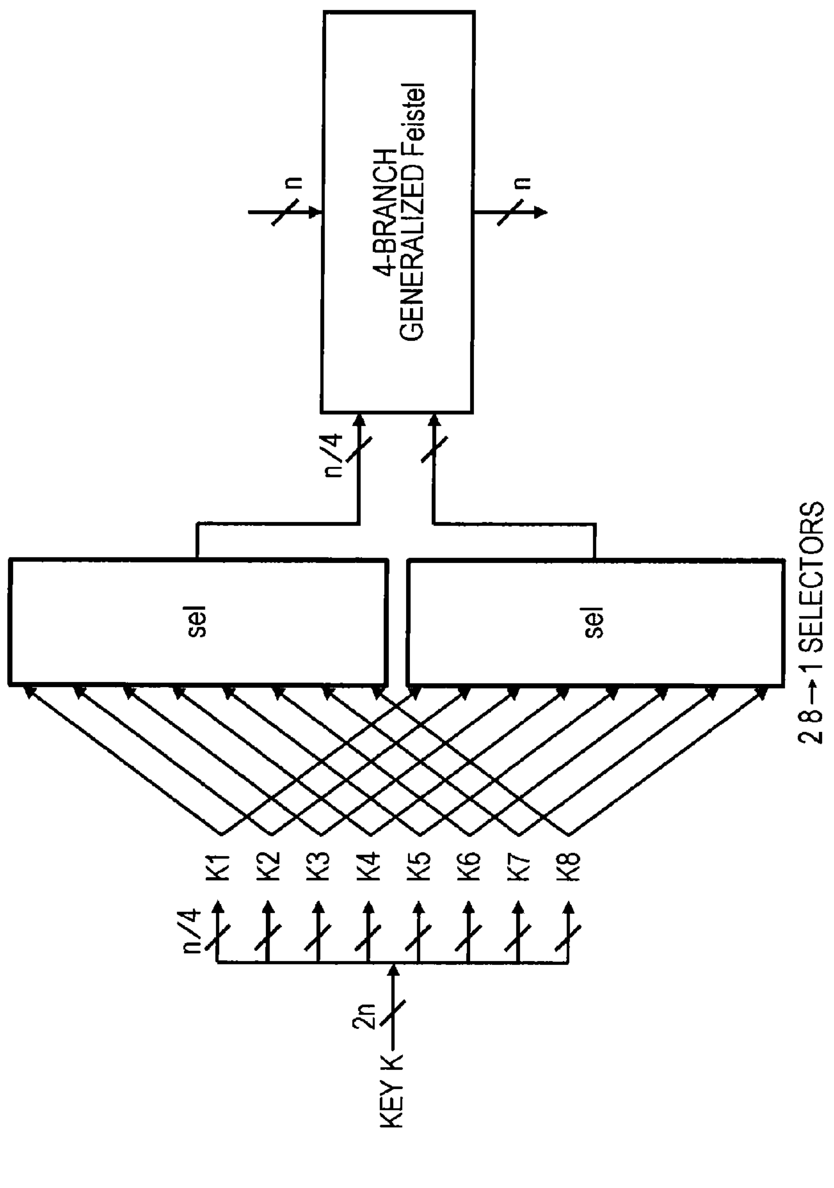

KEY K
2n
n/4
K1
K2
K3
K4
K5
K6
K7
K8
sel
sel
2 8→1 SELECTORS
n/4
4-BRANCH GENERALIZED Feistel
n
n

FIG. 41

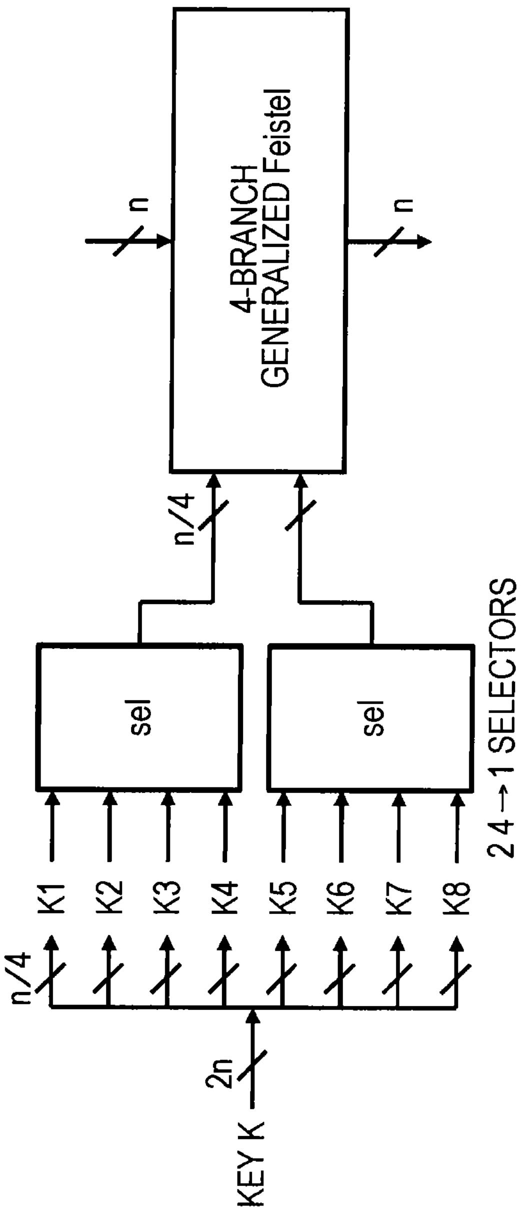

KEY K
2n
n/4
K1
K2
K3
K4
K5
K6
K7
K8
sel
sel
2 4→1 SELECTORS
n/4
4-BRANCH GENERALIZED Feistel
n
n

FIG. 42

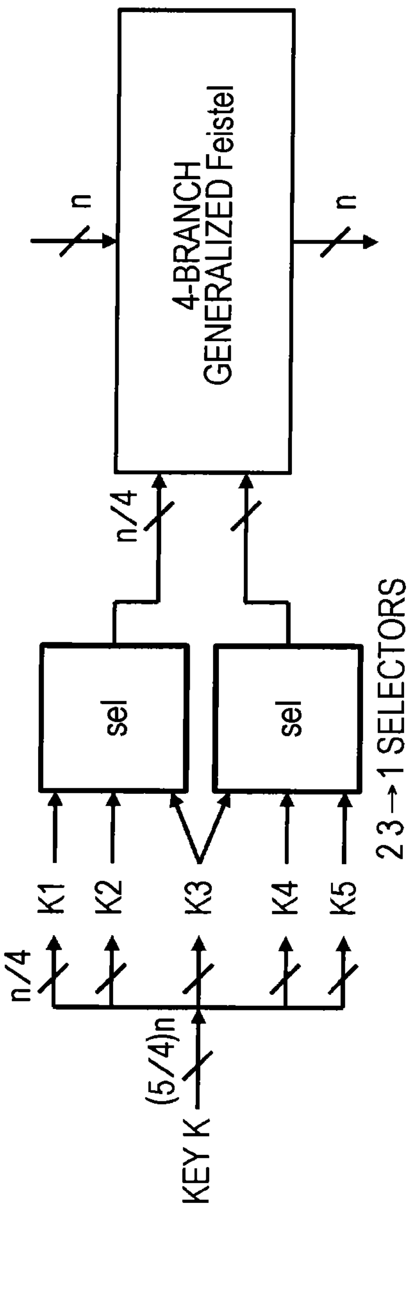
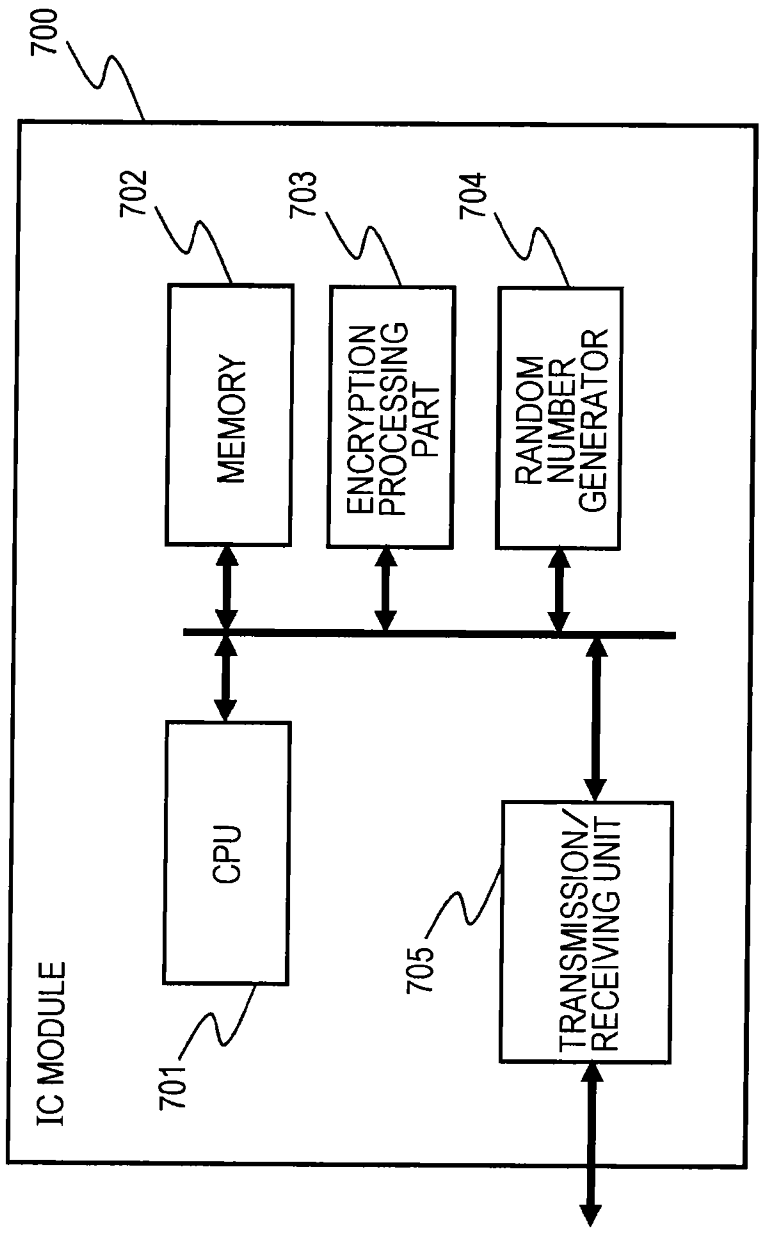

4-BRANCH
GENERALIZED Feistel
n
n
n/4
sel
sel
2 3→1 SELECTORS
K1
K2
K3
K4
K5
n/4
(5/4)n
KEY K

FIG. 43


700
IC MODULE
701
CPU
702
MEMORY
703
ENCRYPTION PROCESSING PART
704
RANDOM NUMBER GENERATOR
705
TRANSMISSION/ RECEIVING UNIT

ENCRYPTION PROCESSING DEVICE, ENCRYPTION PROCESSING METHOD, AND PROGRAM

TECHNICAL FIELD

The present disclosure relates to an encryption processing device, an encryption processing method, and a program. More specifically, this relates to an encryption processing device, an encryption processing method, and a program for executing shared key encryption.

BACKGROUND ART

As the information society continues to develop, the necessity of information security technologies for securely protecting information used increases. One configuration element of information security technologies are encryption technologies, and encryption technologies are currently used by various products and systems.

Though there are various types of encryption processing algorithms, one of the basic technologies is called a shared key block encryption. According to the shared key block encryption, a key for encryption and a key for decryption are shared items. In both the encryption processing and the decryption processing, multiple keys are generated from these shared keys, and a data conversion processing is repeatedly executed in block data units of a certain block unit such as 64 bits, 128 bits, 256 bits, or other.

DES (Data Encryption Standard), which was the previous US standard, and AES (Advanced Encryption Standard), which is the current US standard, are known as representative shared key block encryption algorithms. Other various shared key block encryptions continue to be proposed, and the CLEFIA proposed by Sony Corporation in 2007 is also a shared key block encryption.

These kind of shared key block encryption algorithms are mainly configured with an encryption processing part including a round function executing unit for repeatedly executing conversions of input data, and a key scheduling part for generating round keys to be applied at each round regarding the round function unit. The key scheduling part first generates an expanded key in which the bit count is increased on the basis of a master key (master key), which is a secret key, and generates round keys (secondary keys) to be applied at each round function unit regarding the encryption processing part, based on the generated expanded key.

Configurations for repeatedly executing the round function including linear conversion units and non-linear conversion units are known as specific configurations of these kinds of algorithms. Representative structures include the Feistel structure and the generalized Feistel structure, for example. The Feistel structure and the generalized Feistel structure include structures that convert plaintext into ciphertext by the repetition of a simple round function including an F function as a data conversion function. The linear conversion processing and the non-linear conversion processing are executed by the F function. Further, NPL 1 and NPL 2 are examples of literature which discloses encryption processing applying Feistel structures.

Conversely, various new techniques are emerging such as a technique in which analysis of encryption algorithms or analysis of keys are illegally executed to perform cryptanalysis. One example of this is the related key attack. Though

various countermeasures have been considered to deal with these kinds of attacks, the current state is that these are not sufficient.

CITATION LIST

Non Patent Literature

NPL 1: K. Nyberg, "Generalized Feistel Networks", ASIACRYPT 96, SpringerVerlag, 1996, pp. 91-104.

NPL 2: Yuliang Zheng, Tsutomu Matsumoto, Hideki Imai: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. CRYPTO 1989: 461-480.

NPL 3: Sony Corporation, "The 128-bit Blockcipher CLEFIA Algorithm Specification", Revision 1.0, 2007.

NPL 4: Aoki, Ichikawa, Kanda, Matsui, Moriai, Nakajima, Tokita, "128-bit Block Encryption Camellia Algorithm Specification", Version 2.0, 2001.

NPL 5: GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms, RFC 5830.

NPL 6: 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, V9.0.0, 2009.

SUMMARY OF INVENTION

Technical Problem

The present disclosure is the result of considering the previously described situation, for example, and aims to provide an encryption processing device, an encryption processing method, and a program with a high level of security that makes illegal cryptanalysis such as related key attacks difficult.

Solution to Problem

A first aspect of the present disclosure is an encryption processing device comprising:

an encryption processing part configured to divide configuration bits of data to be data processed into a plurality of lines, and to input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and

a key scheduling part configured to output round keys to a round calculation executing unit in the encryption processing part;

wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys or round key configuration data by dividing a secret key stored beforehand into a plurality of parts;

and wherein the plurality of round keys or plurality of round keys generated from a combination of the round key configuration data are output to a round calculation executing unit sequentially executing in the encryption processing part such that a constant sequence is not repeated.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part changes the input sequence of the multiple round keys corresponding to the round calculation executing unit at units of multiple rounds regarding the round calculation executing unit.

Regarding an embodiment of the encryption processing device according to the present disclosure, the encryption processing part includes an F function executing unit configured to input the data divided into multiple lines and includes a non-linear conversion processing and a linear conversion processing, and a calculating unit configured to execute calculations applying the round keys against the output of the F function executing unit.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part divides a secret key stored beforehand into multiple parts, and generates multiple round keys having the same number of bits as the round key input into the round calculation executing unit.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part divides a secret key stored beforehand into multiple parts, and generates multiple round keys having a smaller number of bits as the round key input into the round calculation executing unit, and performs multiple combinations of the multiple round key configuration data, and generates a round key having the same number of bits as the round key input into the round calculation executing unit.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part outputs juxtaposed multiple round keys that are applied in parallel to the round calculation executing unit regarding the round calculation executing unit sequentially executing in the encryption processing part.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part includes at least one selector configured to perform a selection supply processing of keys corresponding to the round calculation executing unit.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part sets multiple groups by classifying the multiple round keys or the round key configuration data, and performs control processing of the key supply sequence corresponding to the round calculation executing unit at units of the set groups.

Regarding an embodiment of the encryption processing device according to the present disclosure, the key scheduling part includes selectors in units of the groups.

Regarding an embodiment of the encryption processing device according to the present disclosure, the encryption processing part executes encryption processing to convert plaintext as the input data into ciphertext, and executes decryption processing to convert ciphertext as the input data into plaintext.

Further, a second aspect of the present disclosure is an encryption processing method to be executed in an encryption processing device, the encryption processing method comprising:

an encryption processing step in which an encryption processing part is configured to divide configuration bits of data to be data processed into a plurality of lines and input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and

a key scheduling step in which a key scheduling part is configured to output round keys to a round calculation executing unit in the encryption processing part;

wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys or round key configuration data by dividing a secret key stored beforehand into a plurality of parts;

and wherein the plurality of round keys or plurality of round keys generated from a combination of round key configuration data are output to a round key calculation executing unit sequentially executing in the encryption processing part at a setting such that a constant sequence is not repeated.

Further, a third aspect of the present disclosure is a program to execute encryption processing in an encryption processing device, the program comprising:

an encryption processing step in which an encryption processing part is configured to divide configuration bits of data to be data processed into a plurality of lines and input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and

a key scheduling step in which a key scheduling part is configured to output round keys to a round calculation executing unit in the encryption processing part;

wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys or round key configuration data by dividing a secret key stored beforehand into a plurality of parts;

and wherein the plurality of round keys or plurality of round keys generated from a combination of round key configuration data are output to a round calculation executing unit sequentially executing in the encryption processing part at a setting such that a constant sequence is not repeated.

Further, the program according to the present disclosure is a program supplied to a computer system or information processing device capable of executing various program code, for example, by a recording medium, for example. The processing is achieved through the program by executing this kind of program with program executing unit in the information processing device or computer system.

Other objects, features, and advantages of the present disclosure will become clear by the detailed descriptions based on the embodiments of the present invention described later and the attached drawings. Further, the system regarding the present specification is a logical combination configuration of multiple devices, and so each configuration of the devices is not limited to being housed within the same physical unit.

Advantageous Effects of Invention

According to the embodiments of the present disclosure, an encryption processing device with a high level of security is achieved by supply control of round keys.

Specifically, included is an encryption processing part configured to divide configuration bits of data to be data processed into a plurality of lines, and to input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and a key scheduling part configured to output round keys to a round calculation executing unit in the encryption processing part; wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys or round key configuration data by dividing a secret key stored beforehand into a plurality of parts; and wherein the plurality of round keys are output to a round calculation executing unit sequentially executing in the encryption processing part such that a constant sequence is not repeated. According to the present configuration, an encryption processing configuration with a high level of security is achieved that has a high level of resistance to repeated key attacks or other attacks, for example.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** is a diagram describing an n-bit shared key block encryption algorithm corresponding to a key length of k bits.

FIG. **2** is a diagram describing a decryption algorithm corresponding to the n-bit shared key block encryption algorithm corresponding to a key length of k bits, illustrated in FIG. **1**.

FIG. **3** is a diagram describing a relationship between a key scheduling part and a data encryption part.

FIG. **4** is a diagram describing an example configuration of the data encryption part.

FIG. **5** is a diagram describing an example of an SPN structure round function.

FIG. **6** is a diagram describing an example of a Feistel structure round function.

FIG. **7** is a diagram describing an example of an expanded Feistel structure.

FIG. **8** is a diagram describing an example of an expanded Feistel structure.

FIG. **9** is a diagram describing an example configuration of a non-linear conversion unit.

FIG. **10** is a diagram describing an example configuration of a linear conversion processing unit.

FIG. **11** is a diagram describing a key scheduling part (expanded key generating unit).

FIG. **12** is a diagram describing a key scheduling part (expanded key generating unit).

FIG. **13** is a diagram describing an example configuration of a block encryption GOST.

FIG. **14** is a diagram describing a Feistel structure which is a GOST data structure.

FIG. **15** is a diagram describing an attack which can provide an arbitrary secret key difference Δ.

FIG. **16** is a diagram describing a case in which each round key difference Δ_i is divided into the secret key difference Δ at every m bits when there is a replacement type key scheduling part.

FIG. **17** is a diagram describing that the input difference for the F function becomes zero by providing an average difference of (d, d).

FIG. **18** is a diagram describing an example of a type-2 generalized Feistel structure.

FIG. **19** is a diagram describing an example configuration of an XOR after the F function.

FIG. **20** is a diagram describing an example configuration of an XOR after the F function.

FIG. **21** is a diagram describing an example configuration in which a secret key K is divided into four equal parts every (n/2) bits when the key is 2n bits, and are supplied while being shuffled every four rounds.

FIG. **22** is a diagram describing an example configuration in which the secret key K is divided into eight equal parts at every (n/4) bits, and two (n/4)-bit portions are supplied while being shuffled every four rounds.

FIG. **23** is a diagram describing an example configuration in which the secret key K is divided into eight equal parts every (n/2) bits when the length is 4n bits, and are supplied while being shuffled every eight rounds.

FIG. **24** is a diagram describing an example configuration in which the secret key K is divided into eight equal parts every (n/4) bits when the key is 2n bits, and two (n/4)-bit portions are supplied while being shuffled every four rounds.

FIG. **25** is a diagram describing an example configuration in which the secret key is divided into eight equal parts every (n/4) bits wherein four of these are used in a round key $RK_{r,0}$ of a left side F function, and the remaining four are used in a round key $RK_{r,1}$ of a right side F function.

FIG. **26** is a diagram describing a model in which the cyclic shift in a generalized Feistel structure with 4 data lines is changed to a round permutation (generalized Feistel structure+ with 4 data lines).

FIG. **27** is a diagram describing an example of applying a shuffling technique similar to that in FIG. **25**, as to a generalized Feistel structure with 4 data lines.

FIG. **28** is a diagram illustrating a configuration method of a key scheduling part when the secret key is (5n/4) bits, and the round key necessary for one round is (n/2) bits.

FIG. **29** is a diagram describing an example configuration when the round keys are simply input in order.

FIG. **30** is a diagram describing an example configuration when the round keys are simply input in order.

FIG. **31** is a diagram describing a configuration using round keys corresponding to one round for an n-bit block encryption having a generalized Feistel structure with d data lines of a divided number d.

FIG. **32** is a diagram describing an example configuration in which the secret key is n bits, round keys of n/4 bits are generated from dividing the n-bit secret key into four equal parts, and these are input two at a time for each round.

FIG. **33** is a diagram describing an example configuration in which the secret key is (5/4)n bits, round keys of n/4 bits are generated from dividing the n-bit secret key into five equal parts, and these are input two at a time for each round.

FIG. **34** is a diagram describing an example configuration in which the secret key is n bits, round keys of n/4 bits are generated from dividing the (5/4)n-bit secret key into five equal parts, and these are input two at a time for each round.

FIG. **35** is a diagram describing an example configuration in which the selection sequence of keys is changed in units of multiple rounds.

FIG. **36** is a diagram describing an example configuration of a round key supply set so that a replacement processing executed in units of four rounds is different each time.

FIG. **37** is a diagram describing an example configuration in which the selection sequence of keys is changed in units of multiple rounds.

FIG. **38** is a diagram describing an example configuration in which replacement is performed every three rounds, and six keys are used in the three rounds.

FIG. **39** is a diagram describing an example configuration in which the selection sequence of keys is changed in units of multiple rounds.

FIG. **40** is a diagram describing a round key supply configuration as a selector.

FIG. **41** is a diagram describing a round key supply configuration as the selector.

FIG. **42** is a diagram describing a round key supply configuration as a selector.

FIG. **43** is a diagram illustrating an example configuration of an IC module **700** as the encryption processing device.

DESCRIPTION OF EMBODIMENTS

Hereafter, an encryption processing device, an encryption processing method, and a program related to the present disclosure will be described in detail with reference to the drawings. The description will occur according to the following items.

1. Shared Key Block Encryption Overview

2. Configuration of Key Scheduling Part and Processing Overview

3. Attacks on Key Scheduling Parts and Examples of Previous Countermeasures against These Attacks

4. Replacement Type of Key Scheduling Part That Can Obtain Security against Related Key Attacks

5. Various Configuration Examples of Key Replacement Type of Key Scheduling Parts (Variation)

6. Configuration Examples of Encryption Processing Device

7. Conclusion of Configuration of the Present Disclosure

[1. Shared Key Block Encryption Overview]

First, an overview of share key block encryption will be described.

(1-1. Shared Key Block Encryption)

The following definition specifies that which designates shared key block encryption here (hereafter, block encryption).

Block encryption obtains a plaintext P and a key K as input, and outputs a ciphertext C. The bit length of the plaintext and the ciphertext is called a block size, which is written as n. n is an arbitrary integer value that is normally one value determined beforehand for each block encryption algorithm. This case in which the block length is an n block encryption is sometimes called an n-bit block encryption.

The bit length of the key is expressed as k. The key has an arbitrary integer value. The shared key block encryption algorithm can support one or multiple key sizes. For example, for some block encryption algorithm A, the block size is n=128, and so a configuration is possible which supports a key size of k=128, k=192, or k=256.

Plaintext P: n bits

Ciphertext C: n bits

Key K: k bits

FIG. **1** illustrates a diagram of an n-bit shared key block encryption algorithm E corresponding to a key length of k bits.

A decryption algorithm D corresponding to the encryption algorithm E can be defined as an inversion function E^{-1} of the encryption algorithm E, which receives the ciphertext C and key K as the input, and outputs the plaintext P. FIG. **2** illustrates a diagram of the decryption algorithm D corresponding to the encryption algorithm E illustrated in FIG. **1**.

(1-2. Internal Configuration)

The block encryption thought of as a division into two portions. One is a "key scheduling part" to which the key K is input, and outputs an expanded key K' (bit length k') by expanding the bit length according to certain previously determined steps, and the other is a "data encryption part" that receives the plaintext P and the key K' expanded from the key scheduling part, performs a data conversion, and outputs the ciphertext C.

The relationship between these two portions is illustrated in FIG. **3**.

(1-3. Data Encryption Part)

The data encryption part used in the following embodiments can be divided into processing units called round functions. The round function receives two units of data as the input, conducts processing internally, and outputs one unit of data. One part of the input data is an n-bit data currently being encrypted, which results in a configuration in which the output from the round function for some round is supplied as the input for the next round. The other part of the input data is used as data for a portion of the expanded key output from the key scheduler, and this key data is called the round key. Also, the total number of round functions is called the total round number, and is a value determined beforehand for each encryption algorithm. Here, the total round number is expressed as R.

An overview of the data encryption part is illustrated as in FIG. **4** when looking from the input side of the data encryption part in which the input data for the first round is designated as X_1, the data input in the round function for an i number of rounds is designated as X_i, and the round key is designated as RK_i.

(1-4. Round Function)

The round function can have various forms depending on the block encryption algorithm. The round function can be classified by the structure adopted by this encryption algorithm (structure). Typical structures used here as examples are SPN structures, Feistel structures, and expanded Feistel structures.

(A) SPN Structure Round Function

This structure applies linear conversion processing, non-linear conversion, and XOR calculations on the round key and all of the n-bit input data. The order of each calculation is not particularly determined. FIG. **5** illustrates an example of an SPN structure round function.

(B) Feistel Structure

The n-bit input data is divided into two units of n/2-bit data. A function (F function) is applied with one part of this data and the round key as the input, and the output and the other part of the data is XOR calculated. The result of shuffling both sides of this data becomes the output data. Though there are various types of internal configuration of the F function, but these are basically achieved similarly to the SPN structure with a combination of XOR calculations with the round key data, non-linear calculations, and linear conversions. FIG. **6** illustrates an example of a Feistel structure round function.

(C) Expanded Feistel Structure

The data division number of two regarding the Feistel structure is expanded into a format of three or more divisions with the expanded Feistel structure. If the division number is designated as d, then various expanded Feistel structures can be defined depending on d. As the size if the F function input and output is relatively smaller, this is suited for small implementations. FIG. **7** illustrates an example of an expanded Feistel structure in which d=4, and two F functions are applied in parallel within one round. Also, FIG. **8** illustrates an example of an expanded Feistel structure in which d=8, and one F function is applied within one round.

(D) Generalized Feistel Structure with d Data Lines

A d/2 number of F functions are applied in parallel within one round for expanded Feistel structures in which the division number d is an even number.

Also, a cyclic shift is used as a replacement between rounds.

Note that FIGS. **7**, **18**, and **20** illustrate a generalized Feistel structure with 4 data lines.

(1-5. Non-Linear Conversion Processing Unit)

The implementation costs tend to increase as the size of the input data increases for non-linear conversion processing units. In order to circumvent this, many configurations are used in which the corresponding data is divided into multiple units, and non-linear conversion is conducted on this data. For example, when the input size is designated as ms bits, these configurations divide an m number of data units every s bits, and perform non-linear conversions on this data in which the input and output is s bits. The non-linear conversions in these s-bit units are called S-boxes. FIG. **9** illustrates an example.

(1-6. Linear Conversion Processing Unit)

Linear conversion processing units can be defined as matrices considering their nature. The elements of the matrix can generally be expressed in various ways such as a body element of GF (2^8) and an element of GF (2). FIG. **10** illustrates an example of a linear conversion processing unit defined by a matrix of m×m, which defines the ms-bit input and output as GF (2^S).

[2. Configuration of Key Scheduling Part and Processing Overview]

Before describing the encryption processing of the present disclosure, a configuration of the key scheduling part, which is a preliminary configuration and a processing overview will be described.

The key scheduling part is a function as illustrated in FIG. **11** which (expanding key generating unit) inputs the secret key K of k bits, and generates a k'-bit expanded key (round key) K' by some determined conversion.

Generally, $k<k'$, and when the non-linear calculations called round functions are repeatedly performed by the data encryption part, the round keys supplied to each round function are designated as m bits, and the number of round function repetitions is designated as R, $k'=m\times R$. This setting is as illustrated in FIG. **12**.

For example, regarding the shared key block encryption AES,

when $k=128$,

$m=128$, and $R=11$,

and so $k'=1408$.

When $k=192$,

$m=128$, and $R=13$,

and so $k'=1664$.

The following properties (Properties 1 through 3) are desired to ensure security in the key scheduling part.

(Property 1) There is no equivalent key

Further, when $K0'=K1'$ regarding an expanded key K0' when a secret key K0 is input and an expanded key K1' when a secret key K1 ($\neq$K0) is input, K0 and K1 are called equivalent keys.

(Property 2) Has sufficient resistance against related key attacks (related key attack)

Whereas general attacks use the bias in the data between plaintext and ciphertext (difference, linearity, etc.) resulting from some fixed secret key, related key attacks use the bias in the data between plaintext and ciphertext resulting from multiple secret keys.

For example, these attacks are performed predicated on that the values of two secret keys are not known, but the difference between the secret keys is known.

There are also attacks strongly predicated on that the attacker freely selects the difference between the secret keys. It is desirable to achieve a level of security even under this predication.

In the event of normal difference attacks (difference attacks with no consideration for related keys), the attacker uses, for example, only a combination of multiple plaintexts P and ciphertexts C (=E (K, P)) encrypted by an unknown secret key K, and derives this unknown secret key K.

However, in the event of related key difference attacks, the attacker also uses a combination of multiple plaintexts P' and ciphertexts C' (=E (K(+)ΔK, P')) encrypted by a K(+)ΔK in which the attacker added an arbitrarily specified secret key difference ΔK regarding the unknown secret key K in addition to the combination of multiple plaintexts P and ciphertexts C (=E (K, P)) encrypted by an unknown secret key K to derive the unknown secret key K.

Note that the (+) represents the XOR operator.

Thus, with related key difference attacks, the information that the attacker can use is greater, and so the strength of the attacker is more than that of normal difference attacks.

(Property 3) Has sufficient resistance against slide attacks (slide attack).

For example, there is a potential for the level of security to be lost when a slid value is taken as when the round key from the input of a secret key K0 is designated as $RK_1, RK_2, \ldots, RK_R$, and the round key from the input of a secret key K1 is designated as $RK_2, RK_3, \ldots, RK_{R+1}$.

The following properties (4 through 9) are further desired in the key scheduling part regarding implementation matters.

(Property 4) Implementation is simple

(Property 5) Setup time to generate the expanded key from the secret key is short

(Property 6) Expanded keys can be generated on the fly.

(Property 7) Encryption key scheduling part and decryption key scheduling part are shared to the extent possible

(Property 8) Data encryption part and data decryption part are shared to the extent possible

(Property 9) Changes in secret key length are readily supported

Scheduling units preferably satisfy these properties in a well-balanced manner, from the perspectives of security and implementation.

[3. Attacks on Key Scheduling Parts and Examples of Previous Countermeasures Against these Attacks]

Next, attacks on key scheduling parts and example of previous countermeasures against these attacks will be described.

There are various designs of key scheduling parts for each type shared key block encryption.

There is a deep relationship with the data encryption part and the level of security and implementation performance of the key scheduling part, but it is generally thought there are tradeoffs between the level of security and implementation performance similar as with the data encryption part.

Examples of encryptions in which a complex non-linear function is introduced into the key scheduling part include CLEFIA (NPL 3: Sony Corporation, "The 128-bit Blockcipher CLEFIA Algorithm Specification", Revision 1.0, 2007) and Camellia (NPL 4: Aoki, Ichikawa, Kanda, Matsui, Moriai, Nakajima, Tokita, "128-bit Block Encryption Camellia Algorithm Specification", Version 2.0, 2001) among others. These have a high level of security against attack methods such as related key attacks occurring in the key scheduling part, but the implementation cost is comparatively high, and there is a problem in which circuit scales become very large when implementing in hardware in particular. Also, AES is an example of encryption having a key scheduling part in which a comparatively simple non-linear function has been introduced in order to raise implementation performance, but it is known that AES is fragile against related key attacks (for cases of 192- and 256-bit keys).

Further, a technique as an example of a technique to further raise the implementation performance involves a configuration of only a linear function without using a non-linear function. Within these, a key scheduling part which divides the secret key data into multiple parts, and only replaces these does not require many circuits to implement in hardware in particular, and so has a high level of implementation performance. This kind of key scheduling part is called a replacement type key scheduling part.

While the replacement type key scheduling part has a high level of implementation performance, there are many methods that have problems regarding their level of security.

For example as illustrated in FIG. **13**, the key scheduling part regarding the block encryption GOST (NPL 5: GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms, RFC 5830) has a structure to divide a 4n-bit secret key K into eight equal parts every (n/2) bits, and designate these in order as the round keys.

As the key scheduling part can be configured as only a selector for each round without any calculations against the secret key K, this has a feature in which the number of circuits necessary to implement as hardware is very small.

However, as illustrated in FIG. **14**, the GOST data structure takes the Feistel structure and performs an XOR before each F function of the round keys, so it is known that this is fragile against related key attacks, and it is also known that there in actual probability of one that an attack will be successful.

(Related Key Attacks against GOST that have a Probability of One to Succeed)

The round keys obtained from the secret key K0 are designated as $RK0_1$, $RK0_2$, . . . , and $RK0_R$, and the round keys obtained from the secret key K1 are designated as $RK1_1$, $RK1_2$, . . . , $RK1_R$. At this time, the difference between the secret key K0 and the secret key K1 is designated as the secret key difference Δ, and the difference between each round key is designated as the round key difference Δ_1, Δ_2, . . . , and Δ_R.

Here, as illustrated in FIG. **15**, it is assumed that the attacker can provide an arbitrary secret key difference Δ. At this time, regarding CLEFIA, Camellia, and others which introduce a complex non-linear function into the key scheduling part, each round key difference Δ_i is not uniquely predetermined, and so attacks on the data encryption part are difficult.

However, as illustrated in FIG. **16**, according to those having replacement type key scheduling parts, each round key difference Δ_i is that in which the secret key difference Δ is divided every m bits. Regarding GOST for example, when Δ=(d, d, d, d, d, d, d, d) (Δ is 4n bits, and d is (n/2) bits, so as n=64 in the case of GOST, this becomes 256 bits, and 32 bits, respectively), all round key differences Δ_i are equivalent to d.

At this time, when the Feistel structure is taken, and an XOR is performed before the round key supplied in each round (R1, R2, R3, . . .) to each F function, the input difference into the F function is zero by providing (d, d) as the plaintext difference as illustrated in FIG. **17**. When the input difference into the F function is zero, the output difference is always zero, and so the input difference into the next round function is (d, d).

The difference of (d, d) is propagated similarly for all round functions, and so the ciphertext difference is always (d, d).

For this reason, when the ciphertext obtained from the input of the plaintext P and secret key K into GOST is designated as C,

secret key K (+) (d, d, d, d, d, d, d, d),

plaintext P (+) (d, d),

is input to obtain a ciphertext C', which results in C'=C (+) (d, d) at a probability of one.

The probability of obtaining this kind of equality is called success probability of an identification attack using a related key attack. It is necessary to illustrate that this probability is actually sufficiently small regarding block encryption, which is strong against related key attacks.

Also, for example, regarding a type of structure called a type-2 generalized Feistel structure as illustrated in FIG. **18**, when performing an XOR on the round key before the F function using a replacement type key scheduling part, this similarly results in a configuration of a related key attack that transitions at a probability of one.

KASUMI (NPL 6: 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, V9.0.0, 2009) is an example of an encryption having another kind of replacement type key scheduling part.

However, though the key scheduling part in KASUMI has a high level of implementation performance, it is also known to be fragile against related key attacks.

[4. Replacement Type of Key Scheduling Part that can Obtain Security Against Related Key Attacks]

With consideration to the previously described problems, hereafter, configuration methods of replacement type key scheduling parts that can obtain a level of security against related key attacks without a large implementation cost will be described.

The present method first divides the secret key K into multiple equal parts, and supplies these to the data encryption part according to the following techniques.

(Technique 1) Change of round key insertion position: instead of performing an XOR on the round key before the F function as with the related art (position illustrated in FIG. **14** and FIG. **18**, for example), the XOR is taken after the F function (position illustrated in FIG. **19** and FIG. **20**, for example).

(Technique 2) Change in round key generation replacement: the round key is generated using a replacement method that is secure against related key attacks in accordance with the structure of the data encryption part.

The following advantages can be obtained by configuring the key scheduling part in this way.

(Advantage 1) High Implementation Performance

A high level of implementation performance can be expected similar to that of the related art by using a replacement type key scheduling part. There is a particular advance when implementing as hardware as the number of necessary circuits can be reduced.

(Advantage 2) Increase in Level of Security by Changing the Insertion Position of the Round Key

As with the configurations illustrated in FIG. **19** and FIG. **20**, by performing the XOR on the round key after the F function instead of before, related key attacks succeeding at a probability of one such as with the related art GOST can be prevented even though a replacement type key scheduling part is used.

Also, the use of these implementation techniques also leads to an increase in the implementation performance of the data encryption part.

(Advantage 3) Improvement in the Level of Security by Changing the Replacement Method of the Round Key

Instead of supplying the divided secret key in order, which is different from that of the methods according to the related art, they are suitably supplied in a shuffled sequence so as to have resistance against related key attacks.

This sequence is linked to the bit length and number of divisions of the secret key K and the structure of the data encryption part, and so it is necessary to design the key scheduling part for each data encryption part structure.

Regarding the Feistel structure executing one F function per round as illustrated in FIG. **19**, the length of one round key is (n/2) bits, and the secret key K is divided in to 4 equal parts of (n/2) bits when the key is 2n bits, and these parts are supplied shuffled every four rounds.

FIG. **21** illustrates this setting.

The supply processing of the round keys as illustrated in FIG. **21** will be described.

The secret key K is 2n-bit key data.

This 2n-bit secret key K is divided into four equal parts, and four round keys K1, K2, K3, and K4 are generated.

The four round keys K1, K2, K3, and K4 are (n/2)-bit key data.

These four round keys K1, K2, K3, and K4 are used in a different sequence at each unit in which one unit is four rounds.

The first four rounds as illustrated in FIG. **21**: the round keys K1 through K4 are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1
Round R2: round key K2
Round R3: round key K3
Round R4: round key K4

The next four rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K3
Round R6: round key K1
Round R7: round key K4
Round R8: round key K2

The next four rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K4
Round R10: round key K3
Round R11: round key K2
Round R12: round key K1

In this way, with four rounds as a unit, the round key input sequence different for each unit is applied.

Also, it is possible to divide at a unit which is smaller than the length of the round key, so it is also possible to divide the 2n-bit secret key K into eight equal parts every (n/4) bits, and two of these (n/4)-bit units of data are supplied while being shuffled every four rounds.

An example of this is the configuration illustrated in FIG. **22**.

The supply processing of round keys as illustrated in FIG. **22** will be described.

The secret key K is 2n-bit key data.

This 2n-bit secret key K is divided into eight equal parts, and eight keys K1, K2, K3, K4, K5, K6, K7, and K8 are generated.

The eight keys K1 through K8 are (n/4)-bit key data.

(n/2)-bit round keys made up from two keys of combined data selected from these eight keys K1 through K8: KxKy are used in a sequence different at each unit in which one unit is four rounds.

The first four rounds as illustrated in FIG. **22**: the round keys KxKy as a combination of the key data K1 through K8 are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1K2
Round R2: round key K3K4
Round R3: round key K5K6
Round R4: round key K7K8

The next four rounds: the round keys KxKy as a combination of the key data K1 through K8 are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K3K4
Round R6: round key K2K7
Round R7: round key K1K6
Round R8: round key K5K8

The next four rounds: the round keys KxKy as a combination of the key data K1 through K8 are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K2K7
Round R10: round key K4K5
Round R11: round key K3K6
Round R12: round key K1K8

In this way, with four rounds as a unit, the round key input sequence different for each unit is applied.

Also, the secret key K is divided into eight equal parts every (n/2) bits when the length is 4-n bits, and these parts are supplied while being shuffled every eight rounds.

An example of this is the configuration illustrated in FIG. **23**.

The supply processing of round keys as illustrated in FIG. **23** will be described.

The secret key K is 4n-bit key data.

This 4n-bit secret key K is divided into eight equal parts, and eight round keys K1, K2, K3, K4, K5, K6, K7, and K8 are generated.

The eight round keys K1 through K8 are (n/2)-bit key data.

These eight round keys K1 through K8 are used in a sequence different for each unit in which one unit is eight rounds.

The first eight rounds as illustrated in FIG. **23**: the round keys are input and applied by the following sequence regarding R1 through R8.

Round R1: round key K1
Round R2: round key K2
Round R3: round key K3
Round R4: round key K4
Round R5: round key K5
Round R6: round key K6
Round R7: round key K7
Round R8: round key K8

The next eight rounds: the round keys are input and applied by the following sequence regarding R9 through R16.

Round R9: round key K2
Round R10: round key K5
Round R11: round key K1
Round R12: round key K4
Round R13: round key K8
Round R14: round key K6
Round R15: round key K3
Round R16: round key K7

In this way, with eight rounds as a unit, the round key input sequence different for each unit is applied.

Also as illustrated in FIG. **20** regarding a generalized Feistel structure with 4 data lines executing two F functions simultaneously in one round, the round keys are two units of (n/4) bits. When the secret key K is 2n bits, it is divided into eight equal parts every (n/4) bits, and two units of (n/4) bits is supplied while being shuffled every four rounds.

An example of this configuration is illustrated in FIG. **24**.

The supply processing of round keys as illustrated in FIG. **24** will be described.

The secret key K is 2n-bit key data.

This 2n-bit secret key K is divided into eight equal parts, and eight round keys K1, K2, K3, K4, K5, K6, K7, and K8 are generated.

The eight round keys K1 through K8 are (n/4)-bit key data.

Two round keys selected from these eight round keys K1 through K8 are used in a sequence different for each unit in which one unit is four rounds.

The first four rounds as illustrated in FIG. **24**: two round keys are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1 and round key K2
Round R2: round key K3 and round key K4
Round R3: round key K5 and round key K6
Round R4: round key K7 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K5 and round key K1
Round R6: round key K2 and round key K6
Round R7: round key K7 and round key K3

Round R8: round key K4 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K7 and round key K5

Round R10: round key K1 and round key K3

Round R11: round key K4 and round key K2

Round R12: round key K6 and round key K8

In this way, with four rounds as a unit, the round key input sequence different for each unit is applied.

Also, as a technique to further improve implementation efficiency, four units from that divided into eight parts every (n/4) bits is used in a round key $RK_{r,0}$ regarding a left side F function, and the remaining four are used in a round key $RK_{r,1}$ regarding a right side F function.

An example of this configuration is illustrated in FIG. **25**.

The supply processing of round keys as illustrated in FIG. **25** will be described.

The secret key K is 2n-bit key data.

This 2n-bit secret key K is divided into eight equal parts, and eight round keys K1, K2, K3, K4, K5, K6, K7, and K8 are generated.

The eight round keys K1 through K8 are (n/4)-bit key data.

Two round keys selected from these eight round keys K1 through K8 are used in a sequence different for each unit in which one unit is four rounds.

According to this example, the round keys K1 through K4 are applied to the left side F function in each round regarding the Feistel structure with 4 data lines, and the round keys K5 through K8 are applied to the right side F function for each round regarding the Feistel structure with 4 data lines.

The first four rounds as illustrated in FIG. **25**: two round keys are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1 and round key K5

Round R2: round key K2 and round key K6

Round R3: round key K3 and round key K7

Round R4: round key K7 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K2 and round key K5

Round R6: round key K4 and round key K8

Round R7: round key K1 and round key K6

Round R8: round key K3 and round key K7

The next four rounds: two round keys are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K4 and round key K5

Round R10: round key K3 and round key K7

Round R11: round key K2 and round key K8

Round R12: round key K1 and round key K6

In this way, with four rounds as a unit, the round key input sequence different for each unit is applied.

According to the configuration illustrated in FIG. **25**, instead of shuffling the input order of all eight of the round keys K1 through K8 at every unit of four rounds, four of these, that is to say,

Round keys: K1 through K4

Round keys: K5 through K8

the shuffling of the input order of these is performed in units of four round keys.

According to this configuration, the cost of the selector necessary in replacement type key scheduling parts can be further reduced.

Also regarding a generalized Feistel structure with 4 data lines as illustrated in FIG. **26**, for example, the key scheduling part regarding the present method is also valid for models in which the cyclic shift is modified to a round permutation (generalized Feistel+ with 4 data lines).

The model in which the cyclic shift is modified to a round permutation regarding the generalized Feistel structure with 4 data lines as illustrated in FIG. **26** (generalized Feistel+ with 4 data lines) will be described.

The basic configuration of the configuration illustrated in FIG. **26** is a configuration in which an n-bit input data is divided into a d number of units every n/d bits regarding a d-line generalized Feistel structure, wherein an F function processing and an XOR processing is performed on these units as the basic configuration, and the configuration of the calculation with the round keys is performed against the output of the F function.

At this time,

the data sequence input into the F function is designated as the F function input side data sequence,

and the data sequence used in the XOR is designated as the XOR data sequence.

The n/d-bit data transferred in each sequence (each line) is further divided again into d/2 units (in this case, the division does not have to be equal division).

The data divided again into d/2 units for each sequence (each line) is distributed according to the following rules.

(1) The F function input data sequence is always distributed to the XOR data sequence for the next round function

(2) The XOR data sequence is always distributed to the F function input data sequence for the next round function

(3) The data sequence divided into d/2 units is distributed into the data sequence for the next round function of the d/2 position without any overlap between them.

After such a distribution, the data divided into d/2 units is combined into one unit of data.

This is repeated as many times as necessary.

The key scheduling part according to the present method is also valid regarding a model (generalized Feistel+ with 4 data lines) in which this cyclic shift has been modified to a round permutation.

Specifically, the configuration illustrated in FIG. **27**, which is a similar shuffling technique as that in FIG. **25**, is a suitable example.

That is to say, four units from the secret key K divided into eight equal parts every (n/4) bits is used in a round key $RK_{r,0}$ regarding a left side F function, and the remaining four are used in a round key $RK_{r,1}$ regarding a right side F function.

The supply processing of round keys as illustrated in FIG. **27** will be described.

The secret key K is 2n-bit key data.

This 2n-bit secret key K is divided into eight equal parts, and eight round keys K1, K2, K3, K4, K5, K6, K7, and K8 are generated.

The eight round keys K1 through K8 are (n/4)-bit key data.

Two round keys selected from these eight round keys K1 through K8 are used in a sequence different for each unit in which one unit is four rounds.

According to this example, two of the round keys selected from the round keys K1 through K8 are applied to the two F functions in each round regarding the generalized Feistel+ structure with 4 data lines in which the cyclic shift has been modified to a round permutation.

The first four rounds as illustrated in FIG. **27**: two round keys are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1 and round key K5

Round R2: round key K2 and round key K6

Round R3: round key K3 and round key K7

Round R4: round key K7 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K2 and round key K5
Round R6: round key K4 and round key K8
Round R7: round key K1 and round key K6
Round R8: round key K3 and round key K7

The next four rounds: two round keys are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K4 and round key K5
Round R10: round key K3 and round key K7
Round R11: round key K2 and round key K8
Round R12: round key K1 and round key K6

In this way, with four rounds as a unit, the round key input sequence different for each unit is applied.

Next, an example configuration is illustrated for a replacement type key scheduling part in a case when the bit length of the secret key is not an integral multiple of the bit length of the round keys necessary for one round.

For example, FIG. **28** illustrates a configuration method of a key scheduling part when the secret key is (5n/4) bits, and the round keys necessary for one round are (n/2) bits.

According to this configuration, first a replacement and expansion of this is performed so that this becomes an integral multiple of the bit length of the round keys necessary for one round. Afterwards, a configuration is taken in which this is supplied in order.

The supply processing of the round keys as illustrated in FIG. **28** will be described.

The secret key K is (5/4)n-bit key data.

First, the replacement type key scheduling part generates five round keys K1, K2, K3, K4, and K5 equivalent to the bit length of the round key on the basis of this (5/4)n-bit key data.

Two round keys selected from these round keys K1 through K5 are applied to two F functions input as the round keys applied to each round.

The first FIVE rounds as illustrated in FIG. **28**: two round keys are input and applied by the following sequence regarding R1 through R5.

Round R1: round key K3 and round key K4
Round R2: round key K1 and round key K2
Round R3: round key K3 and round key K4
Round R4: round key K5 and round key K5
Round R5: round key K1 and round key K2

Regarding the next five rounds: R6 through R10 and the following five rounds: R11 through R15, two round keys are input and applied by a similar sequence.

A detailed description of the previously described advantage "(Advantage 3) Improvement in the Level of Security by Changing the Replacement Method of the Round Key" will be performed.

First, definitions of a difference probability, an active F function, and a minimum active F function number will be performed.

Difference attacks are attacks which use that propagating from some input difference to some output difference with a high probability. That is to say, when considering security, it is necessary to indicate combinations of input differences and output differences that do not contain that which is propagated with a high probability.

Related key difference attacks are similar attacks which use that propagating from some input different and some secret key difference to some output difference with a high probability. That is to say, when considering security, it is necessary to indicate combinations of input differences, secret key differences, and output differences that do not contain that which is propagated with a high probability. Such a probability of some input difference propagating to some output difference, and a probability of some input difference and some secret key difference propagating to some output difference, are the definitions of the difference probability. As previously described, there is that in which this difference probability regarding GOST is one.

Such a difference probability is known to decrease only by a non-linear function (F function) to which a non-zero input difference is provided. The non-linear function (F function) to which this non-zero input difference is provided is the definition of the active F function. The number of active F functions is closely related to the level of security against difference attacks, and so it can be thought that if there many active F functions regarding some input difference, this will be sufficiently secure.

The number of active F functions can be determined if one input difference is determined. From that just described, it is understood that it needs to be determined how many active F functions should be indicated depending on the kind of input difference provided when considering security against difference attacks. The minimum value of the number of active F functions regarding each kind of such input differences is the definition of the minimum active F function number.

For example, regarding the generalized Feistel structure with 4 data lines in which the output of the F function on the round keys is used in the XOR as illustrated in FIG. **20**, the following (Table 1) illustrates the minimum active F function number corresponding to the number or rounds after consideration of the related key difference when the method according to the related art is adopted such that the round keys are simply input in order as illustrated in FIG. **29**.

TABLE 1

Round Number	Minimum Active F Function Number
1	0
2	0
3	0
4	0
5	0
6	1
7	1
8	2
9	2
10	2
11	2
12	3
13	3
14	3
15	3
16	4
17	4
18	4
19	4
20	5
21	5
22	5
23	5
24	6
25	6
26	6
27	6
28	7
29	7
30	7

In the case of this method, it is understood, for example, that there needs to be at least 28 rounds for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks is seven.

A processing according to the present disclosure as illustrated in FIG. **24**, for example, in a generalized Feistel structure with 4 data lines such as that illustrated in FIG. **20**, that is to say, regarding a replacement type key scheduling part, (Table 2) illustrates the corresponding minimum active F function number and number of rounds in consideration of the related key difference regarding a configuration in which processing to change the input key sequence is performed at a predetermined round unit.

TABLE 2

Round Number	Minimum Active F Function Number
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	1
9	2
10	3
11	4
12	4
13	5
14	6
15	7
16	7
17	8
18	8
19	9
20	9
21	10
22	11
23	11
24	12
25	12
26	12
27	13
28	14
29	15
30	16

According to the previously described Table 2, it is understood that at least 15 rounds are necessary for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks, for example, is seven.

According to the previous Table 1, at least 28 rounds are necessary for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks is seven.

The processing according to the present disclosure, that is to say, regarding the replacement type key scheduling part, it is understood that the number of rounds has been reduced by 13 in comparison to the method of the related art by implementing a configuration in which the processing to change the input key sequence is performed at a predetermined round unit.

In this way, the processing according to the present disclosure, that is to say, regarding the replacement type key scheduling part, it is understood that a larger number of active F functions can be ensured over the method of the related art, which does not perform this kind of key shuffling, by implementing a configuration in which the processing to change the input key sequence is performed at a predetermined round unit.

Similarly, let us consider a model of a generalized Feistel+ structure with 4 data lines as illustrated in FIG. **26** in which the cyclic shift in the generalized Feistel structure with 4 data lines is modified to a round permutation.

When adopting the method of the related art in which the round keys are simply input in order (FIG. **30**) regarding this configuration, the corresponding minimum active F function

number and the number of rounds in consideration of related key differences are the same values as in the previously described (Table 1).

In the case of this method, it is understood that at least 28 rounds are necessary for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks, for example, is seven.

Also, regarding a case in which the configuration of the key scheduling part is implemented according to the present invention as illustrated in FIG. **24** in a generalized Feistel+ structure with 4 data lines such as that illustrated in FIG. **26**, the following (Table 3) illustrates the corresponding minimum active F function number and the number of rounds with consideration to related key differences.

TABLE 3

Round Number	Minimum Active F Function Number
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	3
10	3
11	4
12	5
13	5
14	6
15	7
16	7
17	8
18	9
19	10
20	10
21	11
22	12
23	12
24	13
25	14
26	15
27	15
28	16
29	17
30	17

According to the previously described Table 3, it is understood that at least 15 rounds are necessary for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks, for example, is seven.

According to the previous Table 1, at least 28 rounds are necessary for an encryption in which the number of active F functions necessary to ensure security against related key difference attacks is seven.

The processing according to the present disclosure, that is to say, regarding the replacement type key scheduling part, it is understood that the number of rounds has been reduced by 13 in comparison to the method of the related art by implementing a configuration in which the processing to change the input key sequence is performed at a predetermined round unit.

In this way, the processing according to the present disclosure, that is to say, regarding the replacement type key scheduling part, it is understood that a larger number of active F functions can be ensured over the method of the related art, which does not perform this kind of key shuffling, by implementing a configuration in which the processing to change the input key sequence is performed at a predetermined round unit, and an encryption processing configuration with high security can be realized with few rounds.

[5. Various Configuration Examples of Key Replacement Type of Key Scheduling Parts (Variation)]

Next, various configuration examples of key replacement type key scheduling parts (variation) will be described.

Regarding an n-bit block encryption having a generalized Feistel structure with d data lines in which d designates the number of divisions, the number of (n/d)-bit round keys generally used for one round is (d/2) units as illustrated in FIG. **31**.

Therefore, the total for one round

(n/d)×(d/2)=(n/2)-bit round key data is necessary.

For example, when applying the generalized Feistel structure with 4 data lines (d=4) as illustrated in the previously described FIG. **18**, it is necessary to input two (n/4)-bit round keys each at each round unit to implement the round key input processing.

The generation and input of the round key as described below are executed depending on the length of the secret key forming the basis to generate the round keys.

When the secret key is n bits, n/4-bit round keys are generated by dividing the n-bit secret key into four equal parts, and two each are input for each round as illustrated in FIG. **32**, for example.

When the secret key is (5/4)n bits, n/4-bit round keys are generated by dividing the n-bit secret key into five equal parts, and two each are input for each round as illustrated in FIG. **33** and FIG. **34**, for example.

For example, key replacement is performed every two rounds regarding a configuration using four round keys K1, K2, K3, and K4 from the division of the secret key of n bits into four equal parts as illustrated in FIG. **32**, that is to say, the key supply sequence is changed every two rounds.

The supply processing of round keys as illustrated in FIG. **32** will be described.

The secret key K is n-bit key data.

Four round keys K1, K2, K3, and K4 are generated by dividing this n-bit secret key K into four equal parts.

The four round keys K1, K2, K3, and K4 are (n/4)-bit key data.

These four round keys K1, K2, K3, and K4 are used in sequences which are different at each unit in which one unit is four rounds.

The first two rounds as illustrated in FIG. **21**: two of the round keys K1 through K4 are input and applied by the following sequence regarding R1 through R2.

Round R1: round key K1 and round key K2

Round R2: round key K3 and round key K4

The next two rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R3 through R4.

Round R3: round key K3 and round key K1

Round R4: round key K4 and round key K2

The next two rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R5 through R6.

Round R5: round key K4 and round key K3

Round R6: round key K2 and round key K1

The next two rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R7 through R8.

Round R7: round key K2 and round key K4

Round R8: round key K1 and round key K3

The next two rounds: the round keys K1 through K4 are input and applied by the following sequence regarding R9 through R10.

Round R9: round key K1 and round key K2

Round R10: round key K3 and round key K4

Also, regarding a configuration as illustrated in FIG. **33** using five round keys K1, K2, K3, K4, and K5 from the five equal parts when dividing the secret key K of (5/4)n bits, as key replacement is performed after five key selections of these five keys K1 through K5, two key replacements are performed over five rounds. That is to say, the key supply sequence is changed twice every five rounds.

The supply processing of round keys as illustrated in FIG. **33** will be described.

The secret key K is (5/4)n-bit key data.

First, the replacement type key scheduling part generates the five round keys K1, K2, K3, K4, and K5 corresponding to the bit length of the round key based on this (5/4)n-bit key data.

Two round keys selected from these round keys K1 through K5 are applied to two F functions as the input of round keys applied to each round.

The first five rounds as illustrated in FIG. **33**: two round keys are input and applied by the following sequence regarding R1 through R5

Round R1: round key K1 and round key K2

Round R2: round key K3 and round key K4

Round R3: round key K5 and round key K5

Round R4: round key K2 and round key K3

Round R5: round key K3 and round key K4

The next five rounds: two round keys are input and applied by a similar sequence regarding R6 through R10.

Further, the key supply sequence regarding the key supply processing configuration illustrated in FIG. **33** produces the same result as that described regarding FIG. **28**.

That is to say, in either case the key supply is performed in an order of K1, K2, K3, K4, K5, K5, K5, K1, K2, K3, ...

In this way, regarding the key supply processing for the replacement type key scheduling part, the configuration may perform a processing to change the key selection sequence without performing more than two key replacements regarding a setting which the number of necessary key replacements is m' (>1) for an m number of rounds, for example.

For example, the key supply processing sequence illustrated in FIG. **33** and the key supply processing sequence illustrated in FIG. **34** is exactly the same.

That is to say, the first five rounds: two round keys are input and applied by the following sequence regarding R1 through R5.

Round R1: round key K1 and round key K2

Round R2: round key K3 and round key K4

Round R3: round key K5 and round key K5

Round R4: round key K2 and round key K3

Round R5: round key K3 and round key K4

The next five rounds: two round keys are input and applied by a similar sequence regarding R6 through R10.

Also, instead of shuffling the divided key supply order and inputting these in order into the round function as the key replacement processing, the exact same effect can be obtained by changing the selection order in units of multiple rounds. For example, the round key supply configuration previously described with reference to FIG. **24** and the key supply processing sequence illustrated in FIG. **35** is exactly the same.

That is to say, the first four rounds: two round keys are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1 and round key K2
Round R2: round key K3 and round key K4
Round R3: round key K5 and round key K6
Round R4: round key K7 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K5 and round key K1
Round R6: round key K2 and round key K6
Round R7: round key K7 and round key K3
Round R8: round key K4 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K7 and round key K5
Round R10: round key K1 and round key K3
Round R11: round key K4 and round key K2
Round R12: round key K6 and round key K8

In this way, the round key supply configuration described with reference to FIG. **24** and the key supply processing sequence illustrated in FIG. **35** are configurations that apply different round key input sequences at each unit in which one unit is four rounds.

(Multiple Types of Key Replacements)

According to the configuration previously described with reference to FIG. **24**, the same form of key replacement is performed every four rounds when the key length is 2n bits and divided into eight equal parts.

Conversely, the replacement processing executed at 4-round units may set to be of a different form every time such as with that illustrated in FIG. **36**, for example.

The key supply sequence according to the configuration illustrated in FIG. **36** is as follows. The first four rounds: two round keys are input and applied by the following sequence regarding R1 through R4.

Round R1: round key K1 and round key K2
Round R2: round key K3 and round key K4
Round R3: round key K5 and round key K6
Round R4: round key K7 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R5 through R8.

Round R5: round key K5 and round key K1
Round R6: round key K2 and round key K6
Round R7: round key K7 and round key K3
Round R8: round key K4 and round key K8

The next four rounds: two round keys are input and applied by the following sequence regarding R9 through R12.

Round R9: round key K8 and round key K5
Round R10: round key K7 and round key K3
Round R11: round key K1 and round key K2
Round R12: round key K4 and round key K6

In this way, the key supply sequence illustrated in FIG. **36** is the same as the key supply sequence illustrated in FIG. **24** up to rounds **1** through **4** and rounds **5** through **8**, but the sequence is different from round **9**.

This is because though the key replacement executing by the replacement type key scheduling part regarding the setting illustrated in FIG. **24**, that is to say, the key shuffling processing is a configuration executed at the same setting at 4-round units, but according to the configuration illustrated in FIG. **36**, the key replacement by the replacement type key scheduling part, that is to say, the key shuffling processing is executed at a different setting at 4-rounds units.

Also, instead of shuffling the order of the divided keys and inputting these in order into the round functions, a similar effect can be obtained by changing the selection sequence of keys. For example, the round key supply configuration previously described with reference to FIG. **36** and the key supply processing sequence illustrated in FIG. **37** is exactly the same.

(Portional Selections of Keys)

As previously described, according to the configurations illustrated in FIG. **24** and FIG. **36**, eight keys divided into eight equal parts are input one at a time during four rounds.

A configuration using only six keys over three rounds performing replacement every three rounds as illustrated in FIG. **38** is also possible.

The key supply sequence according to the configuration illustrated in FIG. **38** is as follows. The first three rounds: two round keys are input and applied by the following sequence regarding R1 through R3.

Round R1: round key K1 and round key K2
Round R2: round key K3 and round key K4
Round R3: round key K5 and round key K6

The next three rounds: two round keys are input and applied by the following sequence regarding R4 through R6.

Round R4: round key K7 and round key K3
Round R5: round key K8 and round key K1
Round R6: round key K2 and round key K4

The next three rounds: two round keys are input and applied by the following sequence regarding R7 through R9.

Round R7: round key K6 and round key K8
Round R8: round key K5 and round key K7
Round R9: round key K3 and round key K1

The next three rounds: two round keys are input and applied by the following sequence regarding R10 through R12.

Round R10: round key K4 and round key K5
Round R11: round key K2 and round key K6
Round R12: round key K8 and round key K7

This example designates a configuration in which the same pattern of key replacement is repeated at 3-round units.

Also, instead of shuffling the order of keys, and inputting these in order into the round functions, exactly the same effect can be obtained by changing the selection sequence of keys. For example, the round key supply configuration described with reference to FIG. **38**, for example, is exactly the same as the key supply processing sequence illustrated in FIG. **39**.

(Implementation Efficiency)

The round key supply method according to the present disclosure is similar to the method according to the related art in that the implementation efficiency is high as the processing executed is that by a replacement type key scheduling part.

For example, as illustrated in FIG. **40**, when the length of the secret key K forming the basis to generate the round keys is 2n bits, two round keys are selected every round from the keys K1 through K8 from the eight key parts.

Here, two of the keys K1 through K8 are classified as that regarding the previously described FIG. **27**,

K1, K2, K3, and K4
K5, K6, K7, and K8

Two groups of four of these round keys are set,

and so the implementation efficiency of certain implementation forms can be further improved by implementing a configuration in which key replacement in units of a predetermined round number, that is to say, the changing of key supply sequences is performed by units of these groups.

Specifically, as illustrated in FIG. **41**, a configuration including two selectors of replacement type key scheduling part is designated, and the groups of the four previously described round keys, regarding each selector, that is to say,

K1, K2, K3, and K4
K5, K6, K7, and K8

a configuration in which the selection of output keys from the group of these round keys is executed is designated.

When designing a key scheduling part such as that previously described regarding FIG. **24**, two selectors for selecting one from eight as in FIG. **40** are necessary, but by implementing a configuration of a replacement type key scheduling part as illustrated in FIG. **27**, for example, these become two small selectors for selecting one from four as in FIG. **41**.

Similarly, instead of replacing all five of the five equal parts from diving the (5/4)n-bit key, the implementation efficiency is raised by designing a replacement so that selectors with an input number of three can be used.

Specifically, as illustrated in FIG. **42**,

n/4-bit round keys K1, K2, K, K4, and K5 are generated from the secret key K of (5/4)n bits,

is a configuration in which the first selector executes a key selection of K1, K2, and K3, and the second selector executes a key selection of K3, K4, K5.

[6. Configuration Examples of Encryption Processing Device]

Finally, actual examples of encryption processing devices executing an encryption processing in accordance with the previously described embodiments will be described.

The encryption processing devices for executing the encryption processing in accordance with the previously described embodiments can be installed in various information processing devices executing encryption processing. Specifically, this can be used regarding various crises in which encryption processing is executed along with data processing and communication processing by devices such as PCs, TVs, recorders, players, communication devices, RFIDs, smart cards, sensor network devices, battery/battery authentication modules, health and medical devices, independent network devices, etc.

FIG. **43** illustrates an example configuration of an IC module **700** as an example of a device executing the encryption processing according to the present disclosure. The previously described processing can be executed in various information processing devices such as PCs, IC cards, reader-writers, and others, and the IC module **700** illustrated in FIG. **43** can be configured in these various devices.

A CPU (Central Processing Unit) **701** illustrated in FIG. **43** is a processor executing various programs such as the starting and termination of the encryption processing, control of data transmission and reception, data transfer control between each configuration element, and others. A memory **702** is made up from ROM (Read Only Memory) storing fixed data such as the program executed by the CPU **701**, calculation parameter, and so on, and RAM (Random Access Memory) used as a work region and storage area of the program executed regarding the processing of the CPU **701** and parameters that arbitrarily change during the processing of the program. Also, the memory **702** can be used as a storage region for data and such applied to conversion matrices and conversion tables (replacement tables) applied during the encryption processing, and for key data necessary during encryption processing. Further, the data storage region is desirably configured as memory having a tamper-resistant structure.

An encryption processing part **703** executes encryption processing and decryption processing in accordance with the shared key block encryption processing algorithm applying the previously described encryption processing configurations, that is to say for example, generalized Feistel structures or Feistel structures.

Further, examples illustrated here used encryption processing means as individual models, instead of provisioning these kinds of independent encryption processing modules, a configuration can be implemented in which an encryption processing program can be stored in ROM, for example, and the CPU **701** reads out and executes the program stored in ROM.

A random number generator **704** executes random number generation processing necessary during the generation of keys necessary during encryption processing.

A transmission/reception unit **705** is a data communication processing unit executing data communication with external devices, executes data communication with IC modules such as reader-writers, for example, and executes the output of ciphertext generated within the IC module, the input of data from external reader-writers and so on among others.

Further, the encryption processing device described in the previously described embodiments is not only applicable to encryption processing to encrypt plaintext as input data, but is also applicable to decryption processing to decode the ciphertext as input data back to plaintext.

Regarding both processing, the encryption processing and the decryption processing, the configurations described in the previous embodiments can be applied.

[7. Conclusion of Configuration of the Present Disclosure]

Thus, embodiments of the present disclosure have been described in detail with reference to specific embodiments. However, it will be apparent to those skilled in the art that various modifications and substitutions of the embodiments may be made without departing from the scope and spirit of the present disclosure. That is to say, the present invention has been disclosed exemplarily by embodiments, and should not interpreted restrictively. The Claims should be referenced in order to determine the scope of the present disclosure.

Further, the technologies disclosed in the present specification can take the following configurations.

(1) An encryption processing device comprising:

an encryption processing part configured to divide configuration bits of data to be data processed into a plurality of lines, and to input, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation; and

a key scheduling part configured to output round keys to a round calculation executing unit in the encryption processing part;

wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys or round key configuration data by dividing a secret key stored beforehand into a plurality of parts;

and wherein the plurality of round keys or plurality of round keys generated from a combination of the round key configuration data are output to a round calculation executing unit sequentially executing in the encryption processing part such that a constant sequence is not repeated.

(2) The encryption processing device according to (1), wherein the key scheduling part changes the input sequence of the multiple round keys corresponding to the round calculation executing unit at units of multiple rounds regarding the round calculation executing unit.

(3) The encryption processing device according to either (1) or (2), wherein the encryption processing part includes an F function executing unit configured to input the data divided into multiple lines and includes a non-linear conversion processing and a linear conversion processing, and a calculating unit configured to execute calculations applying the round keys against the output of the F function executing unit.

(4) The encryption processing device according to any one of (1) through (3), wherein the key scheduling part divides a secret key stored beforehand into multiple parts, and generates multiple round keys having the same number of bits as the round key input into the round calculation executing unit.

(5) The encryption processing device according to any one of (1) through (4), wherein the key scheduling part divides a secret key stored beforehand into multiple parts, and generates multiple round keys having a smaller number of bits as the round key input into the round calculation executing unit, and performs multiple combinations of the multiple round key configuration data, and generates a round key having the same number of bits as the round key input into the round calculation executing unit.

(6) The encryption processing device according to any one of (1) through (5), wherein the key scheduling part outputs juxtaposed multiple round keys that are applied in parallel to the round calculation executing unit regarding the round calculation executing unit sequentially executing in the encryption processing part.

(7) The encryption processing device according to any one of (1) through (6), wherein the key scheduling part includes at least one selector configured to perform a selection supply processing of keys corresponding to the round calculation executing unit.

(8) The encryption processing device according to any one of (1) through (7), wherein the key scheduling part sets multiple groups by classifying the multiple round keys or the round key configuration data, and performs control processing of the key supply sequence corresponding to the round calculation executing unit at units of the set groups.

(9) The encryption processing device according to any one of (1) through (8), wherein the key scheduling part includes selectors in units of the groups.

(10) The encryption processing device according to any one of (1) through (9), wherein the encryption processing part executes encryption processing to convert plaintext as the input data into ciphertext, and executes decryption processing to convert ciphertext as the input data into plaintext.

Further, the processing method executed in the previously described device and system, and the program executing this processing is included in the configuration of the present disclosure.

Also, a portion of the processing described in this specification can be executed as hardware, software, or combination of the two. When executing this processing by software, a program to which the processing sequence is recorded is installed and executed in memory within a computer assembled with specialized hardware, or the program can be installed and executed in a general-purpose computer capable of executing the various processing. For example, the program can be recorded onto a recording medium beforehand. Other than installing to the computer from the recording medium, the program can be received via a network such as a LAN (Local Area Network) or the Internet, and can be installed to a recording medium such as an internal hard disk.

Further, the various processing disclosed in this specification can not only be executed temporally as according to the disclosure, but can also be executed in parallel or individually as necessary or depending on the processing performance of the device executing the processing. Also, the system regarding the present specification is a logical combination configuration of multiple devices, and so each configuration of the devices is not limited to being housed within the same physical unit.

INDUSTRIAL APPLICABILITY

As previously described, according to an embodiment of the present disclosure, an encryption processing device with a high level of security is achieved by a supply control of round keys.

Specifically, included are an encryption processing part configured to divide and input configuration bits of data to be data processed into multiple lines, and to repeatedly execute data conversion processing applying a round function to each line of data as a round calculation, and a key scheduling part configured to output round keys to a round calculation executing unit in the encryption processing part, wherein the key scheduling part is a replacement type key scheduling part configured to generate a plurality of round keys by dividing a secret key stored beforehand into multiple parts, in which the generated multiple round keys are output to a round calculation executing unit sequentially executing in an encryption processing part at a setting such that a constant sequence is not repeated. According to the present configuration, an encryption processing configuration with a high level of security is achieved that has a high level of resistance to repeated key attacks or other attacks, for example.

REFERENCE SIGNS LIST

700	IC module
701	CPU (Central Processing Unit)
702	memory
703	encryption processing part
704	random number generator
705	transmission/receiving unit

The invention claimed is:

1. An encryption processing device comprising:
circuitry configured to
- divide input data into a plurality of lines, and to repeatedly execute data conversion processing applying a round function to each line of input data as a round calculation; and
- output round keys for each round of the data conversion processing;
- wherein the circuitry is configured to generate a plurality of round keys by dividing a secret key stored beforehand into a plurality of parts; and
- wherein the plurality of round keys are output such that a constant sequence is not repeated; and
- wherein the plurality of round keys are classified into a plurality of sets of round keys and one round key from each set of the plurality of sets of round keys is output for each round of the data conversion processing.

2. The encryption processing device according to claim **1**, wherein the circuitry is configured to change an input sequence of the plurality of round keys at units of a plurality of rounds.

3. The encryption processing device according to claim **1**, wherein the circuitry is further configured to input the input data divided into a plurality of lines, perform non-linear conversion processing and linear conversion processing, and execute calculations applying the round keys against the output of the non-linear conversion processing and linear conversion processing.

4. The encryption processing device according to claim **1**, wherein the circuitry is further configured to divide the secret key stored beforehand into the plurality of parts, and generate the plurality of round keys having the same number of bits as the round key input into the round calculation.

5. The encryption processing device according to claim **1**, wherein the circuitry is further configured to divide the secret key stored beforehand into the plurality of parts, and generate the plurality of round keys having a smaller number of bits as the round key input into the round calculation, and perform a plurality of combinations of the plurality of generated round keys to generate a round key having the same number of bits as the round key input into the round calculation.

6. The encryption processing device according to claim **1**, wherein the circuitry is further configured to output a juxtaposed plurality of round keys that are applied to the round calculation for sequential execution.

7. The encryption processing device according to claim **1**, wherein the circuitry includes at least one selector configured to perform a selection supply processing of keys corresponding to the round calculation.

8. The encryption processing device according to claim **1**, wherein the circuitry includes individual selectors corresponding to each set of the plurality of sets of round keys.

9. The encryption processing device according to claim **1**, wherein the circuitry is configured to execute encryption processing to convert plaintext as the input data into ciphertext, and execute decryption processing to convert ciphertext as the input data into plaintext.

10. The encryption processing device according to claim **1**, wherein the circuitry has a generalized Feistel structure.

11. The encryption processing device according to claim **1**, wherein the circuitry has a generalized Feistel structure with 4 data lines.

12. The encryption processing device according to claim **1**, wherein the circuitry has a Type-2 generalized Feistel structure.

13. An encryption processing method to be executed in an encryption processing device, the encryption processing method comprising:

dividing input data into a plurality of lines and repeatedly executing, by the encryption processing device, data conversion processing applying a round function to each line of input data as a round calculation; and

outputting round keys for each round of the data conversion processing;

wherein the outputting includes generating a plurality of round keys by dividing a secret key stored beforehand into a plurality of parts; and

wherein the plurality of round keys are output such that a constant sequence is not repeated; and

wherein the plurality of round keys are classified into a plurality of sets of round keys and one round key from each set of the plurality of sets of round keys is output for each round of the data conversion processing.

14. A non-transitory computer readable medium including a computer executable program to execute encryption processing in an encryption processing device, the computer executable program comprising:

dividing input data into a plurality of lines and repeatedly executing data conversion processing applying a round function to each line of input data as a round calculation; and

outputting round keys for each round of the data conversion processing;

wherein the outputting includes generating a plurality of round keys by dividing a secret key stored beforehand into a plurality of parts; and

wherein the plurality of round keys are output such that a constant sequence is not repeated; and

wherein the plurality of round keys are classified into a plurality of sets of round keys and one round key from each set of the plurality of sets of round keys is output for each round of the data conversion processing.

15. An information processing device comprising:

a processor configured to execute a program;

memory configured to store the program;

circuitry configured to

divide input data into a plurality of lines, and to repeatedly execute data conversion processing applying a round function to each line of input data as a round calculation; and

output round keys for each round of the data conversion processing;

wherein the circuitry is configured to generate a plurality of round keys by dividing a secret key stored beforehand into a plurality of parts;

wherein the plurality of round keys are output such that a constant sequence is not repeated; and

wherein the plurality of round keys are classified into a plurality of sets of round keys and one round key from each set of the plurality of sets of round keys is output for each round of the data conversion processing.

16. The encryption processing device according to claim **1**, wherein the circuitry is configured to perform an XOR operation on one of the plurality of round keys and an output of the round function applied to each line of the input data.

* * * * *