



(22) Date de dépôt/Filing Date: 2001/02/15

(41) Mise à la disp. pub./Open to Public Insp.: 2001/09/09

(45) Date de délivrance/Issue Date: 2004/05/25

(30) Priorités/Priorities: 2000/03/09 (2000-070994) JP;
2000/07/06 (2000-210690) JP

(51) Cl.Int.⁷/Int.Cl.⁷ H04L 9/20

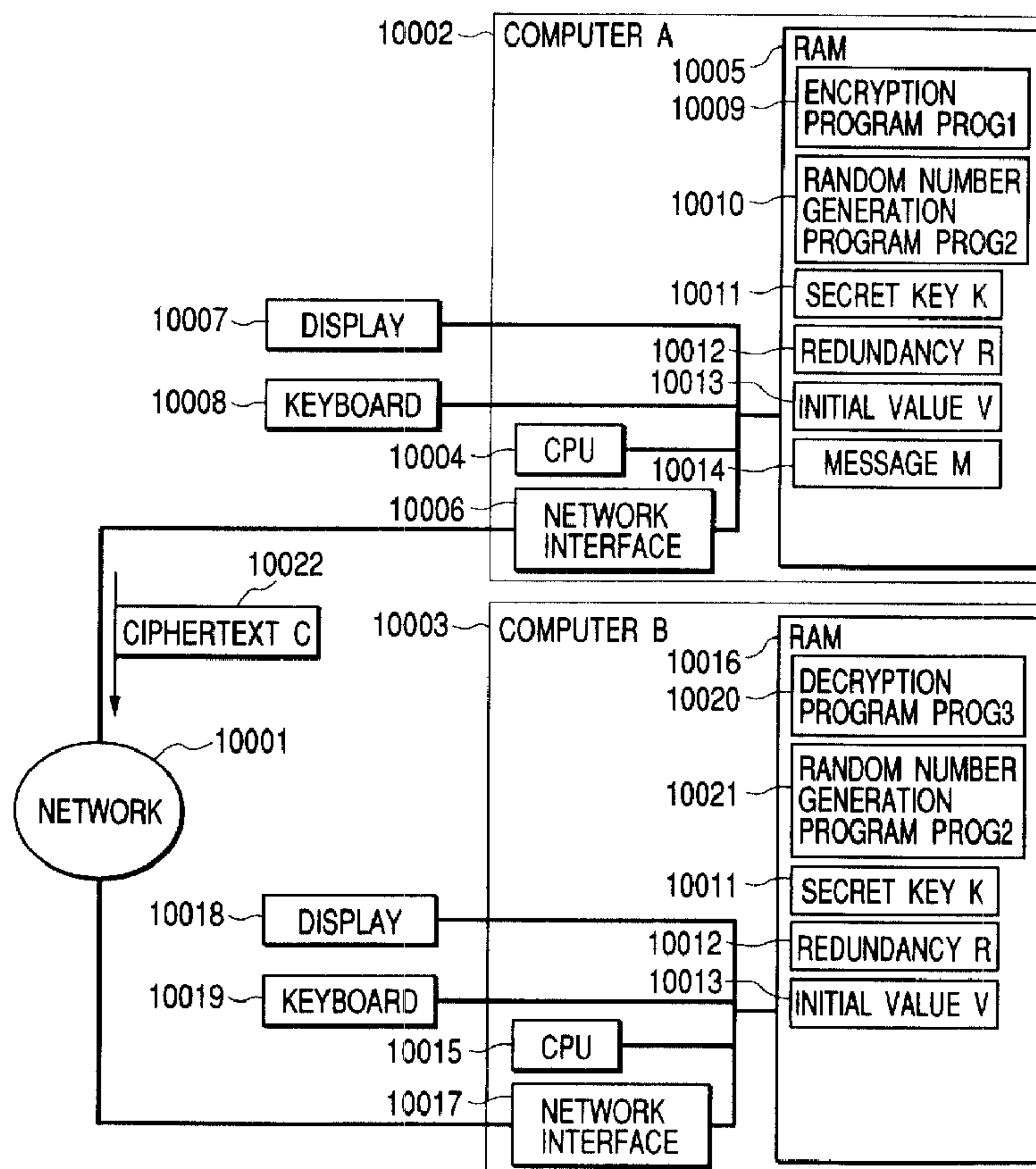
(72) Inventeurs/Inventors:
FURUYA, SOICHI, JP;
TAKARAGI, KAZUO, JP;
KURUMATANI, HIROYUKI, JP;
TAKAHASHI, MASASHI, JP;
MIYAZAKI, KUNIHICO, JP;
SATO, HISAYOSHI, JP;
WATANABE, DAI, JP

(73) Propriétaire/Owner:
HITACHI, LTD., JP

(74) Agent: KIRBY EADES GALE BAKER

(54) Titre : METHODE ET APPAREIL POUR LE CHIFFREMENT DE CLES SYMETRIQUES

(54) Title: METHOD AND APPARATUS FOR SYMMETRIC-KEY ENCRYPTION



(57) Abrégé/Abstract:

The present invention provides a symmetric-key cryptographic technique capable of realizing both high-speed cryptographic processing having a high degree of parallelism and alteration detection. The present invention performs the steps of: dividing



(57) Abrégé(suite)/Abstract(continued):

plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length; generating a random number sequence based on a secret key; generating a random number block corresponding to one of the plurality of plain text blocks from the random number sequence; outputting a feedback value obtained as a result of operation on the one of the plurality of plain text blocks and the random number block, the feedback value being fed back to another one of the plurality of plain text blocks; and performing an encryption operation using the one of the plurality of plain text blocks, the random number block, and a feedback value obtained as a result of operation on still another one of the plurality of plain text blocks to produce a ciphertext block.

ABSTRACT

The present invention provides a symmetric-key cryptographic technique capable of realizing both high-speed cryptographic processing having a high degree of parallelism and alteration detection. The present invention performs the steps of: dividing plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length; generating a random number sequence based on a secret key; generating a random number block corresponding to one of the plurality of plain text blocks from the random number sequence; outputting a feedback value obtained as a result of operation on the one of the plurality of plain text blocks and the random number block, the feedback value being fed back to another one of the plurality of plain text blocks; and performing an encryption operation using the one of the plurality of plain text blocks, the random number block, and a feedback value obtained as a result of operation on still another one of the plurality of plain text blocks to produce a ciphertext block.

METHOD AND APPARATUS FOR SYMMETRIC-KEY ENCRYPTION

BACKGROUND OF THE INVENTION

5 The present invention relates to a technique for ensuring security of confidential information.

Traditional cryptographic processing apparatuses employ a block cipher or a stream cipher for concealing data. Various types of block ciphers have been proposed
10 including DES and IDEATM. DES and IDEATM are described by Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp. 250-259, pp. 263-266.

The security of the total cryptographic process of each block cipher and its characteristics are discussed
15 based on a block-cipher operation mode employed, such as ECB, CBC, CFB, OFB, or the counter mode. However, only the iaPCBC mode is known to be capable of performing both cryptographic processing and detection of an alteration at the same time, and other modes cannot detect alterations by
20 themselves. Block-cipher operation modes are described by Schneider, Applied Cryptography, Second Edition, John Wiley & Sons, Inc., 1996, pp. 189-209.

The iaPCBC mode is described by Gligor, Donescu, "Integrity-Aware PCBC Encryption Schemes," Preproceedings in Secure Protocol Workshop, Cambridge, 1999, to appear in Lecture Notes in Computer Science series, Springer-Verlag.

5 The iaPCBC mode is an operation mode that uses a block cipher. Regarding encryption, the iaPCBC mode can perform neither parallel processing nor preprocessing, making it very difficult to implement the iaPCBC mode in an extremely high speed processing environment.

10 On the other hand, there is a system that generates a cryptographic checksum called a "message authentication code" (hereinafter referred to as "MAC") in order to detect alterations. By implementing a MAC generation process as an independent mechanism, and executing the process during
15 cryptographic processing in one of the above block-cipher operation modes, it is possible to perform both cryptographic processing and detection of an alteration at the same time. However, in this case it is necessary to share two completely independent cryptographic keys, one
20 for encryption and the other for alteration detection. Furthermore, data to be encrypted must be processed twice, once for encryption and a second time for MAC generation. As a result, a realized cryptographic system may be

complicated or may not be suitable for processing data having an extended length. In addition, the processing speed of the block cipher is slower than the current communication speed, which means that it is difficult to
5 apply any technique using a combination of the block cipher and MAC to processing of the order of gigabit-per-second or terabit per-second. MAC is described by Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp. 352-368.

10 In contrast with the block cipher, a stream cipher is an encryption mechanism that uses one of various proposed cryptographic pseudorandom number generators. The stream cipher was not able to detect alterations by itself regardless of security or characteristics of each
15 implementation. Well-known stream ciphers, or pseudorandom number generators used for stream ciphers include SEAL, a linear feedback shift register using a nonlinear combination generator, a linear feedback shift register using a nonlinear filter, and a clock-controlled linear
20 feedback shift register. SEAL is described by Schneider, Applied Cryptography, Second Edition, John Wiley & Sons, Inc., 1996, pp. 398-400.

On the other hand, Systems based on the above feedback shift registers are described by Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp. 203-212. A technique using a combination of a stream cipher and a MAC can also perform both cryptographic processing and detection of an alteration at the same time. Furthermore, processing of a stream cipher is 2 to 20 times faster than that of a block cipher. However, as is the case with the combination of a block cipher and MAC, every MAC generation system (meaning every combination of a stream cipher and MAC) requires sharing of two different keys, and processing of a message twice. When considered in detail, the MAC generation system requires a particular mechanism in addition to that necessary for the stream cipher itself, and considerable computational complexity. For example, MAC generation systems such as HMAC and UMAC require a safe hash function having guaranteed cryptographically-collision-free one-way characteristics. This means that it is necessary to implement the above safe function in addition to a stream cipher. HMAC is described by Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, p. 355, Example 9.67 while UMAC is described by Black, Halevi, Krawczyk, Krovetz, Rogaway, "UMAC: Fast and Secure Message Authentication,"

Advances in Cryptology, - CRYPTO '99 Lecture Notes in
Computer Science, Vol. 1666, Springer-Verlag, 1999.

Generally, however, hash functions such as SHA-1 and
MD5 are very complicated, and are not easy to implement.

5 These hash functions are described by Menezes, van
Oorschot, Vanstone, Handbook of Applied Cryptography, CRC
Press, 1996, pp. 347-349.

The security of hash functions has not yet been
studied adequately in contrast with study of the security
10 of block ciphers. Therefore, a user may not be able to
incorporate a hash function because the user cannot rely on
the hash function. In regards to MAC generation systems,
MMH uses only a pseudorandom number generator, and
requires a very small amount of additional resources
15 such as circuits and programs to add an alteration
detection function to the cryptographic process.

However, MMH requires a pseudorandom number sequence
whose length is as long as that of the message, taking
long time to generate necessary random numbers. MMH is
20 described by Halevi, Krawczyk, "MMH: Software Message
Authentication in the Gbit/Second Rates," Fast Software
Encryption, 4th International Workshop, FSE

'97, Lecture Notes in Computer Science, Vol. 1267,
Springer-Verlag, 1997. As described above, the prior art
techniques are unsatisfactory in terms of ensuring of
security and high-speed processing, and therefore a safer
5 and faster cryptographic processing technique is required.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a
safer and faster symmetric-key cryptographic processing
10 technique.

An additional object of the present invention is to
provide a symmetric-key cryptographic method that is
capable of performing alteration detection and decryption
at the same time, and whose safety for data confidentiality
15 and data alteration protection is provable.

A further object of the present invention is to
provide a symmetric-key cryptographic method that
advantageously has preprocessing and parallel processing
functions, and is capable of processing at high speed,
20 capitalizing on the high-speed processing characteristics
of the pseudorandom number generator.

Another object of the present invention is to provide
a symmetric-key cryptographic method whose processing speed
is not only faster than that of the conventional block

cipher, but can be made still faster as the amount of resources employed is increased, and that can attain a high level of parallel operation for high-speed processing.

Yet another object of the present invention is to
5 provide a symmetric-key cryptographic method whose processing speed does not drop even when a very short message is processed.

A further object of the present invention is to provide a symmetric-key cryptographic method that can be
10 implemented by adding a very small circuit or program to stream cipher equipment.

An additional object of the present invention is to provide a symmetric-key cryptographic method capable of processing each block using a pseudorandom number sequence
15 as a key stream, and detecting an alteration at the same time.

In accordance with one aspect of the present invention there is provided a symmetric-key encryption method performed in a cryptographic system having input means for
20 inputting at least a message, a secret data, and a parameter opened to the public, and output means for outputting an encrypted data, said encryption method comprising the steps of: dividing a plain text composed of redundancy data and a message to generate a plurality of

plain text blocks each having a predetermined length;
generating a random number sequence based on a secret key;
generating a plurality of random number blocks from said
random number sequence, each of said random number blocks
5 corresponding to one of said plurality of plain text
blocks; outputting a first feedback value to a second one
of said plurality of plain text blocks obtained as a result
of an operation on a first one of said plurality of plain
text blocks and a first one of said random number blocks;
10 and performing an encryption operation using said first one
of said plurality of plain text blocks, said first one of
said random number blocks, and a second feedback value
obtained as a result of an operation on a third one of the
plurality of plain text blocks, to produce a ciphertext
15 block.

In accordance with another aspect of the present
invention there is provided a symmetric-key decryption
method performed in a cryptographic system having input
means for inputting at least a message, an encrypted data,
20 and a parameter opened to the public, and output means for
outputting a decrypted data, said decryption method
comprising the steps of: dividing a ciphertext to generate
a plurality of ciphertext blocks each having a
predetermined length; generating a random number sequence

based on a secret key; generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of ciphertext blocks; outputting a first feedback value to a second one of said plurality of ciphertext blocks obtained as a result of an operation on a first one of said plurality of ciphertext blocks and a first one of said random number blocks; and performing a decryption operation using said first one of said plurality of ciphertext blocks, said first one of said random number blocks, and a second feedback value obtained as a result of an operation on a third one of the plurality of ciphertext blocks, to produce a plain text block.

In accordance with yet another aspect of the present invention there is provided a symmetric-key encryption apparatus for use in a cryptographic system having input means for inputting at least a message, a secret data, and a parameter opened to the public, and output means for outputting an encrypted data, said encryption apparatus comprising: a circuit for dividing a plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length; a circuit for generating a random number sequence based on a secret key; a circuit for generating a plurality of random

number blocks from said random number sequence, each of
said random number blocks corresponding to one of said
plurality of plain text blocks; a circuit for outputting a
first feedback value to a second one of said plurality of
5 plain text blocks obtained as a result of an operation on a
first one of said plurality of plain text blocks and a
first one of said random number blocks; and a circuit for
performing an encryption operation using said first one of
said plurality of plain text blocks, said first one of said
10 random number blocks, and a second feedback value obtained
as a result of an operation on a third one of the plurality
of plain text blocks, to produce a ciphertext block.

In accordance with still yet another aspect of the
present invention there is provided a symmetric-key
15 decryption apparatus for use in a cryptographic system
having input means for inputting at least a message, an
encrypted data, and a parameter opened to the public, and
output means for outputting a decrypted data, said
decryption apparatus comprising: a circuit for dividing a
20 ciphertext to generate a plurality of ciphertext blocks
each having a predetermined length; a circuit for
generating a random number sequence based on a secret key;
a circuit for generating a plurality of random number
blocks from said random number sequence, each of said

random number blocks corresponding to one of said plurality of ciphertext blocks; a circuit for outputting a first feedback value to a second one of said plurality of ciphertext blocks obtained as a result of an operation on a first one of said plurality of ciphertext blocks and a first one of said random number blocks; and a circuit for performing a decryption operation using said first one of said plurality of ciphertext blocks, said first one of said random number blocks, and a second feedback value obtained as a result of an operation on a third one of the plurality of ciphertext blocks, to produce a plain text block.

In accordance with still yet another aspect of the present invention there is provided a computer-readable recording medium having recorded thereon statements and instructions for use in the execution in a computer of said symmetric-key encryption method.

In accordance with still yet another aspect of the present invention there is provided a computer-readable recording medium having recorded thereon statements and instructions for use in the execution in a computer of said symmetric-key decryption method.

In accordance with still yet another aspect of the present invention there is provided a computer program product, comprising: a memory having computer-readable code

embodied therein for implementing a symmetric-key encryption method performed in a cryptographic system having input means for inputting at least a message, a secret data, and a parameter opened to the public, and
5 output means for outputting an encrypted data, said computer program product comprising: code means for dividing a plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length; code means for generating a
10 random number sequence based on a secret key; code means for generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of plain text blocks; code means for outputting a first feedback value to
15 a second one of said plurality of plain text blocks obtained as a result of an operation on a first one of said plurality of plain text blocks and a first one of said random number blocks; and code means for performing an encryption operation using said first one of said plurality
20 of plain text blocks, said first one of said random number blocks, and a second feedback value obtained as a result of an operation on a third one of the plurality of plain text blocks, to produce a ciphertext block.

As for implementation cost, the present invention can avoid additional implementation that is difficult to make, such as the additional implementation of a hash function.

These and other benefits are described throughout the
5 present specification. A further understanding of the nature and advantages of the invention may be realized by reference to the remaining portions of the specification and the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a system configuration employed in
embodiments of the present invention;

Fig. 2 is a flowchart of a plain text preparation
5 subroutine;

Fig. 3 is a flowchart of a random number generation
subroutine;

Fig. 4 is a flowchart of an encryption subroutine;

Fig. 5 is a flowchart of the decryption program shown
10 in Fig. 1;

Fig. 6 is a flowchart of the ciphertext preparation
subroutine shown in Fig. 5;

Fig. 7 is a flowchart of the decryption subroutine
shown in Fig. 5;

15 Fig. 8 is a flowchart of the plain text extraction
subroutine shown in Fig. 5;

Fig. 9 is a flowchart of the redundancy extraction
subroutine shown in Fig. 5;

Fig. 10 is a diagram showing data blocks in
20 encryption;

Fig. 11 is a diagram showing data blocks in the
decryption shown in Fig. 7;

Fig. 12 is a flowchart of the random number generation
2 subroutine according to a second embodiment of the
present invention;

Fig. 13 is a flowchart of the encryption 2 subroutine
5 of the second embodiment;

Fig. 14 is a flowchart of the second embodiment;

Fig. 15 is a flowchart of the decryption 2 subroutine
of the second embodiment;

Fig. 16 is a diagram showing data blocks in the
10 encryption according to the second embodiment;

Fig. 17 is a diagram showing data blocks decryption
according to the second embodiment;

Fig. 18 is a flowchart of the encryption program
according to a third embodiment of the present invention;

Fig. 19 is a flowchart of the random number generation
15 3 subroutine of the third embodiment;

Fig. 20 is a flowchart of the encryption 3 subroutine
of the third embodiment;

Fig. 21 is a flowchart of the decryption of the third
20 embodiment;

Fig. 22 is a flowchart of the decryption 3 subroutine
of the third embodiment;

Fig. 23 is a diagram showing data blocks in the
encryption according to the third embodiment;

Fig. 24 is a diagram showing data blocks in the decryption according to the third embodiment;

Fig. 25 is a flowchart of the parallel encryption program according to a fifth embodiment of the present
5 invention;

Fig. 26 is a flowchart of the parallel decryption program of the fifth embodiment;

Fig. 27 is a diagram showing data blocks in the encryption according to the fifth embodiment;

10 Fig. 28 is a diagram showing data blocks in the decryption according to the fifth embodiment;

Fig. 29 is a flowchart of the random number generation 4 subroutine according to a fourth embodiment of the present invention;

15 Fig. 30 is a flowchart of the plain text preparation 2 subroutine of the fourth embodiment;

Fig. 31 is an explanatory diagram showing a padding program of operation on a message according to the fourth embodiment;

20 Fig. 32 is a flowchart of the decryption program of the fourth embodiment;

Fig. 33 is a flowchart of the plain text extraction 2 subroutine shown in Fig. 32;

Fig. 34 is an explanatory diagram showing an

extraction operation on decrypted text according to the fourth embodiment;

Fig. 35 is a diagram showing the configuration of a system for cryptocommunications according to a sixth
5 embodiment of the present invention;

Fig. 36 is a diagram showing the configuration of an encryption apparatus employed in a cryptocommunication system according to a seventh embodiment of the present invention;

10 Fig. 37 is a diagram showing the configuration of a contents delivery system according to an eighth embodiment of the present invention;

Fig. 38 is a diagram showing the configuration of a system according to a ninth embodiment of the present
15 invention; and

Fig. 39 is a diagram showing the configuration of an encryption/decryption router according to a tenth embodiment of the present invention.

20 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS
(First Embodiment)

Fig. 1 shows the configuration of a computer system including a computer A 10002 and a computer B 10003 connected to each other through a network 10001 for

cryptocommunications from the computer A 10002 to the computer B 10003. The computer A 10002 has an operation unit (hereinafter referred to as "CPU") 10004, a memory unit (volatile or nonvolatile, hereinafter referred to as "RAM") 10005, a network interface 10006 therein, with a display 10007 and a keyboard 10008 externally connected thereto for the user to operate the computer A 10002. The RAM 10005 stores an encryption program PROG1 10009, a random number generation program PROG2 10010, a secret key K 10011, that is shared only between the computers A 10002 and B 10003, a redundancy R 10012 and an initial value V 10013 both of which are data shared between the computers A 10002 and B 10003, and encryption-target data 10014 to be transmitted to the computer B 10003. The computer B 10003 has a CPU 10015, a RAM 10016, and a network interface 10017 with a display 10018 and a keyboard 10019 externally connected thereto for the user to operate the computer B 10003. The RAM 10016 stores a decryption program PROG3 10020, a random number generation program PROG2 10021, the secret key K 10011, the redundancy R 10012, and the initial value V 10013.

The computer A 10002 executes the encryption program PROG1 10009 to generate ciphertext C 10022 from a message M 10014 and transmits the generated ciphertext C 10022 to the

network 10001 through the network interface 10006.
Receiving the ciphertext C 10022 through the network
interface 10017, the computer B 10003 executes the
decryption program PROG3 10020, and if no alteration is
5 detected, the computer B 10003 stores the decryption
results in the RAM 10016.

Each program employed can be introduced into each RAM
by receiving the program from another computer in the form
of a transmission signal, which is a transmission medium on
10 the network 10001, or by using a portable medium such as a
CD or an FD. Each program can be configured so that it
runs under control of the operating system (not shown) of
each computer.

The encryption program PROG1 10009 is read out from
15 the RAM 10005, and executed by the CPU 10004 in the
computer A 10002. The encryption program PROG1 10009
internally calls a random number generation program PROG2
10010 as a subroutine to process the input secret key K
10011, the redundancy R 10012, the initial value V 10013,
20 and the message M 10014 so as to output ciphertext C 10022.

The decryption program PROG3 10020 is read out from
the RAM 10016, and executed by the CPU 10015 in the
computer B 10003. The decryption program PROG3 10020
internally calls a random number generation program PROG2

10021 as a subroutine to process the input key, the redundancy R 10012, the initial value V 10013, and ciphertext C 10022 so as to output a message or an alteration detection alarm.

- 5 Description will be made of the process flow of the encryption program PROG1 10009.

 Step 20002 (a data setting subroutine): waits for input of an initial value V, a redundancy R, and a secret key K.

- 10 Step 20003 (a plain text preparation subroutine): waits for input of plain text, adds predetermined padding and a redundancy to the given plain text, and divides the padded plain text into a series of plain text blocks P_i ($1 \leq i \leq n$) each having 64 bits and outputs these plain text
15 blocks.

 Step 20004 (a random number generation subroutine) outputs pseudorandom number sequences A_i and B_i ($1 \leq i \leq n$) based on the secret key K.

- Step 20005 (an encryption subroutine): uses the
20 pseudorandom number sequences A_i and B_i , the series of plain text blocks P_i ($1 \leq i \leq n$), and the initial value V to output a series of ciphertext blocks C_i ($1 \leq i \leq n$).

 Step 20006: concatenates the series of ciphertext blocks C_i ($1 \leq i \leq n$) obtained at step 20005 one after another

sequentially to output ciphertext C.

In this specification, the term "padding" used above refers to input of additional data to main data. In the case of padding of digital data, the additional data is often concatenated to the main data, simply bits to bits.

Description will be made of the process flow of the plain text preparation subroutine with reference to Fig. 2.

Step 20202: waits for input of an encryption-target message M. The message M is either input from the keyboard or read out from a RAM, or introduced from another medium.

Step 20203: adds padding indicating the length of the message. Specifically, this step adds 64-bit binary data indicating the length of the message M to the head of the message M.

Step 20204: adds padding to the message so that the length of the message is a multiple of a predetermined number. Specifically, the padded data is set to have an integer multiple of 64 bits for subsequent processing. When the length of the message M to which the data indicating the length is added at step 20203 is L bits, this step adds $(64 - L \pmod{64})$ number of 0s to the end of the message M.

Step 20205 (addition of redundancy data): further adds

a redundancy R of 64 bits to the end of the message.

Step 20206 (division of message data into plain text blocks): divides the data obtained at step 20205 into blocks $P_1, P_2, \dots, P_n$, each having 64 bits.

5 Description will be made of the process flow of the random number generation subroutine with reference to Fig. 3.

Step 20302 (input of necessary parameters): obtains number n of blocks making up the padded message, and secret
10 key K .

Step 20303 (generation of a pseudorandom number sequence A): calls the random number generation program PROG2 to generate a pseudorandom number sequence having 64*n bits and output it as a pseudorandom number sequence
15 A .

Step 20304 (division of random number sequence A blocks): divides the pseudorandom number sequence A blocks $A_1, A_2, \dots, A_n$, each having 64 bits.

Step 20305 (initialization of a counter i):
20 initializes a counter so that $i=1$.

Step 20306 (generation of a random number B_i): executes PROG2 using the secret key K to generate a random number B_i having 64 bits.

Step 20307: if the random number B_i generated at step

20306 is 0, returns to step 20306.

Step 20308: if $i=n$, performs step 20310.

Step 20309: increments the counter i and returns to step 20306.

5 Description will be made of the process flow of the encryption subroutine with reference to Fig. 4.

Step 20402: sets an initial value F_0 so that $F_0=V$.

Step 20403: sets a counter so that $i=1$.

Step 20404: calculates a feedback value F_i by the
10 formula $F_i = P_i \wedge A_i$.

Step 20405: calculates a ciphertext block C_i by the formula $C_i = (F_i * B_i) \wedge F_{i-1}$.

Step 20406: if $i=n$, performs step 20408.

Step 20407: increments the counter i and returns to
15 step 20404.

Description will be made of the process flow of the decryption program PROG3 10020 with reference to Fig. 5.

Step 20502 (a data setting subroutine): waits for input of the initial value V , the redundancy R , and the
20 secret key K .

Step 20503 (a ciphertext preparation subroutine): waits for input of ciphertext C' , and divides the given ciphertext C' into a series of ciphertext blocks C'_i ($1 \leq i \leq n$) each having 64 bits and outputs them.

Step 20504 (a random number generation subroutine):
outputs pseudorandom number sequences A_i and B_i ($1 \leq i \leq n$)
based on the secret key K .

Step 20505 (a decryption subroutine): uses the
5 pseudorandom number sequences A_i and B_i , the series of
ciphertext blocks C'_i ($1 \leq i \leq n$), and the initial value V to
output a series of plain text blocks P'_i ($1 \leq i \leq n$).

Step 20506 (a plain text extraction subroutine):
combines the series of plain text blocks P'_i into three
10 data strings L' , M' , and Z' .

Step 20507 (a redundancy extraction subroutine):
divides Z' into R' and T' .

Step 20508: if $T=0$ and $R'=R$, proceeds to step 20510.

Step 20509: outputs a rejection indication and
15 proceeds to step 25011.

Step 20510: stores M' into a RAM.

At step 20509 or 20510, the decryption program outputs
a result (acceptance/rejection or the encryption result) to
the display 10018 as a notification to the user.

20 Description will be made of the process flow of the
ciphertext preparation subroutine with reference to Fig. 6.

Step 20602: waits for input of ciphertext C' .

Step 20603: divides the ciphertext C' into blocks C'_1 ,
 C'_2 , ..., C'_n , each having 64 bits.

Description will be made of the process flow of the decryption subroutine with reference to Fig. 7.

Step 20702: sets an initial value F'_0 so that $F'_0 = V$.

Step 20703: initializes a counter so that $i=1$.

5 Step 20704: calculates a feedback value F'_i by the formula $F'_i = (C'_i \wedge F'_{i-1}) / B_i$.

Step 20705: calculates a plain text block P'_i by the formula $P'_i = F'_i \wedge A_i$.

Step 20706: if $i=n$, performs step 20708.

10 Step 20707: increments the counter i and returns to step 20704.

Description will be made of the process flow of the plain text extraction subroutine with reference to Fig. 8.

15 Step 20802: sets L' to the first 64-bit plain text block.

Step 20803: sets M' to the L' number of bits starting from the most significant bit of P'_2 included in the series of decrypted plain text blocks.

20 Step 20804: after L' and M' are removed from the series of decrypted plain text blocks, sets Z' to the remaining decrypted plain text blocks (data).

Description will be made of the process flow of the redundancy extraction subroutine with reference to Fig. 9.

Step 20902: sets R' to the lower 64 bits of Z' .

Step 20903: after R' is removed from Z' , sets T' to the remaining data.

Fig. 10 is an explanatory diagram showing the encryption process. The encircled plus " $(+)$ " denotes an
 5 exclusive OR logic operation between two pieces of data each having a width of 64 bits, while the encircled X mark " (X) " denotes a multiplication operation between two pieces of data each having a width of 64 bits in the finite field $F2^{64}$.

10 The message M 20931 is added with data 20930 indicating the length, appropriate padding 20932, and a redundancy R 20933 to produce plain text P 20934.

The produced plain text P 20934 is divided into blocks P_1 20935, P_2 20936, P_3 20937, ..., P_n 20938, each having 64
 15 bits.

P_1 20935 and A_1 20940 are exclusive-ORed to produce a feedback value F_1 20941 which is then multiplied by B_1 20942 in a finite field. The result is exclusive-ORed with an initial value F_0 20939 to obtain a ciphertext block C_1
 20 20943.

Similarly, P_2 20936 and A_2 20946 are exclusive-ORed to produce a feedback value F_2 20945 which is then multiplied by B_2 20946 in a finite field. The result is exclusive-ORed with the feedback value F_1 20941 to obtain a

ciphertext block C_2 20947.

The above procedure is repeated up to P_n 20938, obtaining ciphertext blocks C_1 20943, C_2 20947, C_3 20951, ..., C_n 20955. The ciphertext blocks are concatenated one
5 after another in that order to obtain ciphertext C 20956.

Fig. 11 is an explanatory diagram showing the decryption process. The encircled slash " $(/)$ " denotes a division operation between two pieces of data each having a width of 64 bits in the finite field $F2^{64}$. In the figure,
10 data introduced to the encircled slash symbol from top is the dividend, while data introduced from left is the divisor.

Ciphertext C' 20960 is divided into blocks C'_1 20962, C'_2 20963, C'_3 20964, ..., C'_n 20965, each having 64 bits.

15 C'_1 and an initial value F'_0 20961 are exclusive-ORed, and the result is divided by B_1 20966. The division result is set as a feedback value F'_1 20967. The feedback value F'_1 20967 and A_1 20968 are exclusive-ORed to obtain a plain text block P'_1 20969.

20 The other blocks C'_2 20963, C'_3 20964, ..., C'_n 20965 are also processed in the same way as C'_1 20962 to obtain plain text blocks P'_1 20969, P'_2 20972, P'_3 20977, P'_n 20981, which are then concatenated one after another to produce plain text P' 20982. The plain text P' 20982 is

divided into L' 20983, M' 20984, and Z' 20985.

Furthermore, Z' 20985 is divided into T' 20988 and R' 20989
so as to check the redundancy R' 20989.

The first embodiment uses a pseudorandom number
5 sequence whose length is about twice as long as that of the
message for cryptographic processes. Even though
pseudorandom-number processing is faster than block-cipher
processing, it has the highest cryptographic process
computational complexity. Therefore, it is desirable to
10 reduce the number of random numbers used.

(Second Embodiment)

As described below, a second embodiment of the present
invention employs a function different from that used by
the first embodiment. By employing this second embodiment
15 function, the number of random numbers used can be reduced
by using the same divisor for each iteration in the
decryption process. This makes it possible to perform the
division operation at substantially the same speed as that
of a multiplication operation if the reciprocal is
20 calculated beforehand, resulting in very efficient
processing.

The second embodiment employs an encryption program
PROG1A and a decryption program PROG3A instead of the
encryption program PROG1 and the decryption PROG3,

respectively.

The encryption program PROG1A replaces the random number generation subroutine 20004 and the encryption subroutine 20005 employed in the encryption program PROG1
5 10009 in Fig. 1 by a random number generation 2 subroutine 21004 and an encryption 2 subroutine 21005, respectively.

Description will be made of the process flow of the random number generation 2 subroutine 21004 with reference to Fig. 12.

10 Step 21102 (input of necessary parameters): obtains number n of message blocks making up a padded message and a secret key K .

Step 21103 (generation of pseudorandom number sequence A): calls the random number generation program PROG2 to
15 generate a pseudorandom number sequence having $64 \cdot n$ bits and outputs n the number as a pseudorandom number sequence A.

Step 21104 (division of pseudorandom number sequence A into blocks): divides the pseudorandom number sequence A
20 into blocks $A_1, A_2, \dots, A_n$, each having 64 bits.

Step 21105 (generation of random number B): executes PROG2 using the secret key K to generate a random number B having 64 bits.

Step 21106: if the value of B generated at step 21105

is 0, returns to step 21105.

Description will be made of the process flow of the encryption 2 subroutine 21005 with reference to Fig. 13.

Step 21202: sets an initial value F_0 so that $F_0=V$.

5 Step 21203: sets a counter so that $i=1$.

Step 21204: calculates a feedback value F_i by the formula $F_i=P_i^{A_i}$.

Step 21205: calculates a ciphertext block C_i by the formula $C_i=(F_i*B)^{F_{i-1}}$.

10 Step 21206: if $i=n$, performs step 21208.

Step 21207: increments the counter i and returns to step 21204.

Description will be made of the process flow of the decryption program PROG3A corresponding to PROG1A with
15 reference to Fig. 14.

The decryption program PROG3A replaces the random number generation subroutine 20504 and the decryption subroutine 20505 employed in the decryption program PROG3 10020 by a random number generation 2 subroutine 21304 and
20 a decryption 2 subroutine 21305, respectively.

Step 21302 (a data setting subroutine): waits for input of the initial value V , the redundancy R , and the secret key K .

Step 21303 (a ciphertext preparation subroutine):
waits for input of ciphertext C' , and divides the given
ciphertext C' into a series of ciphertext blocks C'_i
($1 \leq i \leq n$) each having 64 bits and outputs them.

5 Step 21304 (a random number generation subroutine):
outputs pseudorandom number sequences A_i ($1 \leq i \leq n$) and B in
response to the secret key K .

Step 21305 (a decryption subroutine): uses the
pseudorandom number sequences A_i and B , the series of
10 ciphertext blocks C'_i ($1 \leq i \leq n$), and the initial value V to
output a series of plain text blocks P'_i ($1 \leq i \leq n$).

Step 21306 (a plain text extraction subroutine):
combines the series of plain text blocks P'_i into three
data strings L' , M' , and Z' .

15 Step 21307 (a redundancy extraction subroutine):
divides Z' into R' and T' .

Step 21308: if $T=0$ and $R'=R$, proceeds to step 21310.

Step 21309: outputs a rejection indication and
proceeds to step 21311.

20 Step 21310: stores M' into a RAM.

Description will be made of the process flow of the
decryption 2 subroutine 21305 in Fig. 14 with reference to
Fig. 15.

Step 21402: sets an initial value F'_0 so that $F'_0=V$.

Step 21403: calculates $1/B$ beforehand.

Step 21404: initializes a counter so that $i=1$.

Step 21405: calculates a feedback value F'_i by the
 5 formula $F'_i=(C'_i \wedge F'_{i-1}) * (1/B)$.

Step 21406: calculates a plain text block P'_i by the
 formula $P'_i=F'_i \wedge A_i$.

Step 21407: if $i=n$, performs step 21409.

Step 21408: increments the counter i and returns to
 10 step 21405.

Fig. 16 is an explanatory diagram showing the
 encryption process employed by the above method of
 increasing the processing speed.

The message M 21421 is added with data 21420
 15 indicating the length, appropriate padding 21422, and a
 redundancy R 21423 to produce plain text P 21424.

The produced plain text is divided into blocks P_1
 21425, P_2 21426, P_3 21427, ..., P_n 21428, each having 64
 bits.

20 P_i 21425 and A_i 21431 are exclusive-ORed to produce a
 feedback value F_i 21432 that is multiplied by B 21429 in a
 finite field. The result is exclusive-ORed with an initial
 value F_0 21430 to obtain a ciphertext block C_i 21433.

Similarly, P_2 21426 and A_2 21434 are exclusive-ORed to

produce a feedback value F_2 21435 that is then multiplied by B 21429 in a finite field. The result is exclusive-ORed with the feedback value F_1 21432 to obtain a ciphertext block C_2 21436.

5 The above procedure is repeated up to P_n 21428, obtaining ciphertext blocks C_1 21433, C_2 21436, C_3 21439, ..., C_n 21442. The ciphertext blocks are concatenated one after another in that order to obtain ciphertext C 21443.

Fig. 17 is an explanatory diagram showing the
10 corresponding decryption process.

Ciphertext C' 21450 is divided into blocks C'_1 21453, C'_2 21454, C'_3 21455, ..., C'_n 21456, each having 64 bits.

C'_1 and an initial value F'_0 21451 are exclusive-ORed, and the result is multiplied by $1/B$ 21452. The
15 multiplication result is set as a feedback value F'_1 21457. The feedback value F'_1 21457 and A_1 21458 are exclusive-ORed to obtain a plain text block P'_1 21459.

The other blocks C'_2 21454, C'_3 21455, ..., C'_n 21456 are also processed in the same way as C'_1 21453 to obtain
20 plain text blocks P'_1 21459, P'_2 21462, P'_3 21465, ..., P'_n 21468. These plain text blocks are then concatenated one after another to produce plain text P' 21476. The plain text P' 21476 is divided into L' 21469, M' 21470, and Z'

21471. Furthermore, Z' 21471 is divided into T' 21474 and R' 21475 so as to check the redundancy R' 21475.

The second embodiment uses a 64-bit redundancy, and therefore employs addition and multiplication in the finite
5 field $F2^{64}$.

With enhanced efficiency provided by this embodiment, it is possible to realize high-speed cryptographic processing. An implementation example written in the C programming language achieved a processing speed of 202
10 Mbit/sec in encryption processing using a 64-bit processor with a clock frequency of 600 MHz. On the other hand, a processing speed of 207 Mbit/sec was observed in decryption processing.

The above implementation uses such operations as
15 pseudorandom number generation, exclusive OR, and multiplication in the finite field $F2^{64}$, which are efficiently implemented especially by hardware. For example, it is estimated that with a gate array fabricated in a 0.35 μm process, the above operations can be
20 implemented by adding an additional circuit having 3 k gates for the pseudorandom number generator. Furthermore, the pseudorandom number generator can be implemented using parallel processing, making it easy to realize a parallel processing device (including the pseudorandom number

generator) having a processing speed as high as required. Thus, it is possible to realize a processing speed of 9.6 Gbit/sec at maximum by adding a circuit having about 36 k gates to a parallel pseudorandom number generator.

5 (Third Embodiment)

As described below, a third embodiment of the present invention uses another high-speed processing function to provide processing at higher speed with the same security level as those of the first and the second embodiments. In
10 another aspect, the third embodiment can provide higher security equivalent to $F2^{128}$ if operations in the finite field $F2^{64}$ employed in the first and second embodiments are also used.

In the aspect of providing processing at higher speed
15 described above, the third embodiment uses an operation in the finite field $F2^{32}$ twice. Since multiplication in the field $F2^{64}$ generally requires a computational amount (computational complexity) four times as much as that for the finite field $F2^{32}$, the third embodiment requires only
20 half $((1/4)*2)$ of the computational amount (computational complexity) required by an operation in the finite field $F2^{64}$, thereby doubling the processing speed.

In the aspect of enhancing security, the third embodiment can use both an operation in the finite field

$F2^{64}$ and a 64-bit feedback value twice to reduce the alteration success rate from 2^{-64} of the above method to 2^{-128} .

The third embodiment employs an encryption program
5 PROG1B and a decryption program PROG3B instead of the encryption program PROG1 and the decryption program PROG3.

The encryption program PROG1B replaces the random number generation subroutine (step 20004) and the encryption subroutine (step 20005) employed in the
10 encryption program PROG1 10009 in Fig. 1 by a random number generation 3 subroutine 21504 and an encryption 3 subroutine 21505. Description will be made of the process flow of the encryption program PROG1B with reference to Fig. 18.

15 Step 21502 (a data setting subroutine): waits for input of an initial value V, a redundancy R, and a secret key K.

Step 21503 (a plain text preparation subroutine):
waits for input of plain text, adds predetermined padding
20 and a redundancy to the given plain text, and divides the padded plain text into a series of plain text blocks P_i ($1 \leq i \leq n$) each having 32 bits and outputs these plain text blocks.

Step 21504 (random number generation 3 subroutine):
outputs pseudorandom number sequences A_i ($1 \leq i \leq n$), Ba , and
Bb based on the secret key K .

Step 21505 (encryption 3 subroutine): uses the
5 pseudorandom number sequences A_1 , Ba , and Bb , the series of
plain text blocks P_i ($1 \leq i \leq n$), and the initial value V to
output a series of ciphertext blocks C_i ($1 \leq i \leq n$).

Step 21506: concatenates the series of ciphertext
blocks C_i ($1 \leq i \leq n$) obtained at step 21505 one after another
10 sequentially to output ciphertext C .

Description will be made of the process flow of the
random number generation 3 subroutine 21504 with reference
to Fig. 19.

Step 21602 (input of necessary parameters): obtains
15 number n of message blocks making up the padded message the
secret key K .

Step 21603 (generation of pseudorandom number sequence
A): calls the random number generation program PROG2 to
generate a pseudorandom number sequence having $32 \cdot n$ bits
20 and outputs it as a pseudorandom number sequence A .

Step 21604 (division of random number sequence A into
blocks): divides the pseudorandom number sequence A into
blocks $A_1, A_2, \dots, A_n$, each having 32 bits.

Step 21605 (generation of random number Ba): executes PROG2 using the secret key K to generate a random number Ba having 32 bits.

Step 21606: if the value of the random number Ba
5 generated at step 21605 is 0, returns to step 21605.

Step 21607 (generation of random number Bb): executes PROG2 using the secret key K to generate a random number Bb having 32 bits.

Step 21608: if the value of the random number Bb
10 generated at step 21607 is 0, returns to step 21607.

Description will be made of the process flow of the encryption 3 subroutine 21505 with reference to Fig. 20. The symbols "*" and "^" denote multiplication and addition, respectively, in the finite field $F2^{32}$.

15 Step 21702: sets initial values FA_0 and FB_0 so that $FA_0=FB_0=V$.

Step 21703: initializes a counter so that $i=1$.

Step 21704: calculates a feedback value FA_i by the formula $FA_i=P_i \wedge A_i$.

20 Step 21705: calculates a feedback value FB_i by the formula $FB_i=(FA_i * Ba) \wedge FB_{i-1}$.

Step 21706: calculates a ciphertext block C_i by the formula $C_i= (FB_i * Bb) \wedge FB_{i-1}$.

Step 21707: if $i=n$, performs step 21709.

Step 21708: increments the counter i and returns to step 21704.

Description will be made of the process flow of the decryption program PROG3B with reference to Fig. 21. The decryption program PROG3B replaces the random number generation subroutine 20504 and the decryption subroutine 20505 employed in the decryption program PROG3 10020 by a random number generation 3 subroutine 21804 and a decryption 3 subroutine 21805, respectively.

10 Step 21802 (a data setting subroutine): waits for of the initial value V , the redundancy R , and the key K .

Step 21803 (a ciphertext preparation subroutine): waits for input of ciphertext C' , and divides the given ciphertext C' into a series of ciphertext blocks C'_i (15 $(1 \leq i \leq n)$ each having 32 bits and outputs them.

Step 21804 (a random number generation subroutine): outputs pseudorandom number sequences A_i ($1 \leq i \leq n$), Ba , and Bb based on the secret key K .

20 Step 21805 (a decryption subroutine): uses the pseudorandom number sequences A_i , Ba , Bb , the series of ciphertext blocks C'_i ($1 \leq i \leq n$), and the initial value V to output a series of plain text blocks P'_i ($1 \leq i \leq n$).

Step 21806 (a plain text extraction subroutine):
combines the series of plain text blocks P'_i into three
data strings L' , M' , Z' .

Step 21807 (a redundancy extraction subroutine):
5 divides Z' into R' and T' .

Step 21808: if $T=0$ and $R=R'$, proceeds to step 21810.

Step 21809: outputs a rejection indication and
proceeds to step 21811.

Step 21810: stores M' into a RAM.

10 Description will be made of the process flow of the
decryption 3 subroutine 21805 in Fig. 21 with reference to
Fig. 22. The symbol $"/$ denotes division in the finite
field $F2^{32}$.

Step 21902: sets initial values FA'_0 and FB'_0 so that
15 $FA'_0=FB'_0=V$.

Step 21903: calculates $1/Ba$ and $1/Bb$ beforehand.

Step 21904: initializes a counter so that $i=1$.

Step 21905: calculates a feedback value FB'_i by the
formula $FB'_i = (C'_i \wedge FB'_{i-1}) * (1/Bb)$.

20 Step 21906: calculates a feedback value FA'_i by the
formula $FA'_i = (FB'_i \wedge FA'_{i-1}) * (1/Ba)$.

Step 21907: calculates a plain text block P'_i by the
formula $P'_i = FA'_i \wedge A_i$.

Step 21908: if $i=n$, performs step 21910.

Step 21909: increments the counter i and returns to step 21905.

Fig. 23 is an explanatory diagram showing the encryption process employed by the above method of increasing the processing speed.

The message M 21921 is added with data L 21920 indicating the length, appropriate padding 21922, and a redundancy R 21923 to produce plain text P 21924.

The produced plain text P 21924 is divided into blocks P_1 21925, P_2 21926, P_3 21927, ..., P_n 21928, each having 32 bits.

P_1 21925 and A_1 21933 are exclusive-ORed to produce a feedback value FA_1 21934 that is then multiplied by Ba 21929 in a finite field. The result is exclusive-ORed with an initial value FA_0 21930 to obtain a feedback value FB_1 21935. The obtained feedback value FB_1 21935 is multiplied by Bb 21931 in a finite field, and the multiplication result is exclusive-ORed with an initial value FB_0 21932 to obtain a ciphertext block C_1 21936.

Similarly, P_2 21926 and A_2 21937 are exclusive-ORed to produce a feedback value FA_2 21938 that is multiplied by Ba 21929 in a finite field. The result is exclusive-ORed with the feedback value FA_1 21934 to obtain a feedback value FB_2 21939. The obtained FB_2 21939 is multiplied by

Bb 21931 in a finite field, and the multiplication result is exclusive-ORed with the feedback value FB_1 21935 to obtain a ciphertext block C_2 21940.

The above procedure is repeated up to P_n 21928, obtaining ciphertext blocks C_1 21936, C_2 21940, C_3 21944, ..., C_n 21950. The ciphertext blocks are concatenated one after another in that order to obtain ciphertext C 21951.

Fig. 24 is an explanatory diagram showing the corresponding decryption process.

Ciphertext C' 21960 is divided into blocks C'_1 21961, C'_2 21962, C'_3 21963, ..., C'_n 21964, each having 32 bits.

C'_1 and an initial value FB'_0 21965 are exclusive-ORed, and the result is multiplied by $1/Bb$ 21966. The multiplication result is set as a feedback value FB'_1 21969. The feedback value FB'_1 21969 is exclusive-ORed with an initial value FA'_0 21967, and the result is multiplied by $1/Ba$ 21968 to generate a feedback value FA'_1 21970. The feedback value FA'_1 21970 is exclusive-ORed with A_1 21971 to obtain a plain text block P'_1 21972.

The other blocks C'_2 21962, C'_3 21963, ..., C'_n 21964 are also processed in the same way as C'_1 21961 to obtain plain text blocks P'_1 21972, P'_2 21976, P'_3 21980, P'_n 21985. These plain text blocks are then concatenated one after another to produce plain text P' 21986. The plain

text P' 21986 is divided into L' 21987, M' 21988, and Z' 21989. Furthermore, Z' 21989 is divided into T' 21992 and R' 21993 so as to check the redundancy R' 21993.

(Fourth Embodiment)

5 As described below, a fourth embodiment of the present invention provides a cryptographic method capable of properly starting encryption/decryption processing without using information on the length of a message to be processed. Accordingly, the fourth embodiment makes it
10 possible to perform cryptographic processing of data (message) of a stream type whose entire length is not known beforehand.

The fourth embodiment replaces the random number generation 2 subroutine and the plain text preparation
15 subroutine in the encryption program PROG1A, and the decryption program PROG3A employed in the second embodiment by a random number generation 4 subroutine, a plain text preparation 2 subroutine, and a decryption program PROG6, respectively.

20 Description will be made of the process flow of the random number generation 4 subroutine with reference to Fig. 29.

Step 40212 (input of necessary parameters): obtains the number n of message blocks making up a padded message,

and a secret key K.

Step 40213 (generation of pseudorandom number sequence A): calls the random number generation program PROG2 to generate a pseudorandom number sequence having $64 \cdot n$ bits
5 and outputs it as a pseudorandom number sequence A.

Step 40214 (division of pseudorandom number sequence A into blocks): divides the pseudorandom number sequence A into blocks $A_1, A_2, \dots, A_n$, each having 64 bits.

Step 40215 (generation of random number B): executes
10 PROG2 using the secret key K to generate a random number B having 64 bits.

Step 40216: if the value of B generated at step 40215 is 0, returns to step 40215.

Step 40217 (generation of random number Q): executes
15 PROG2 using the secret key K to generate a random number Q having 64 bits.

Next, description will be made of the process flow of the plain text preparation 2 subroutine with reference to Figs. 30 and 31.

20 Step 40202: waits for input of an encryption-target message M 40300. The message is either input from the keyboard 10008 or read out from a RAM, or introduced from another medium.

Step 40203: adds padding to the message so that the length of the message is a multiple of a predetermined number. Specifically, the padded data (message) is set to have an integer multiple of 64 bits for subsequent
5 processing. When the length of the message M 40300 is L bits, this step adds $(64 - L(\bmod 64))$ number of 0s to the end of the message M 40300.

Step 40204 (addition of secret data): further adds 64-bit secret data Q 40302 to the end of the message M 40300.
10 The secret data Q 40302 can be known by only a person who holds or has obtained its key (or the key data). The secret data may be a random number generated from the secret key K. The above step 40217 performs this process of generating secret data.

15 Step 40205 (addition of redundancy data): still further adds a redundancy R 40303 of 64 bits to the end of the message M 40300.

Step 40206 (division of message data into plain text blocks): divides the data P 40304 (the padded message)
20 obtained at step 40205 into blocks $P_1, P_2, \dots, P_n$, each having 64 bits.

Description will be made of the process flow of the decryption program PROG6 with reference to Figs. 32 and 34.

Step 40402 (a data setting subroutine): waits for input of the initial value V , the redundancy R , and the secret key K .

Step 40403 (a ciphertext preparation subroutine):
5 waits for input of ciphertext C' , and divides the given ciphertext C' into a series of ciphertext blocks C'_i ($1 \leq i \leq n$) each having 32 bits and outputs them.

Step 40404 (random number generation 4 subroutine):
outputs pseudorandom number sequences A_i ($1 \leq i \leq n$) and B
10 based on the secret key K .

Step 40405 (decryption 3 subroutine): uses the pseudorandom number sequences A_i , B , and Q , the series of the ciphertext blocks C'_i ($1 \leq i \leq n$), and the initial value V to output a series of plain text blocks P'_i ($1 \leq i \leq n$).

15 Step 40406 (plain text extraction 2 subroutine): combines the series of plain text blocks P'_i 40601 into three data strings M' 40602, Q' 40603, and R' 40604.

Step 40407: if Q' 40603 = Q 40302 and R' 40604 = R 40303, proceeds to step 40409.

20 Step 40408: outputs a rejection indication and proceeds to step 40410.

Step 40409: stores M' into a RAM.

Step 40410: ends the process.

Next, description will be made of the process flow of the plain text extraction 2 subroutine with reference to Fig. 33.

Step 40502: removes the last 128 bits of the decrypted
5 plain text, and sets a plain text block M' to the remaining decrypted text.

Step 40503: sets Q' to the upper 64 bits of the removed last 128 bits obtained at step 40502.

Step 40504: sets R' to the lower 64 bits of the
10 removed last 128 bits.

(Fifth Embodiment)

The above first through fourth embodiments of the present invention have a single-processor configuration, that is, they do not employ parallel processing. However,
15 fifth embodiment of the present invention, shows that the present invention can be easily applied to parallel processing.

The system configuration (not shown) of the fifth embodiment is different from that shown in Fig. 1 in that
20 the computer A 10002 employs two CPUs, CPU 1 30004 and CPU 2 30005, instead of the single CPU 10004, and the RAM 10005 stores a parallel encryption program PROG4 30016 in addition to the components shown in Fig. 1. Furthermore, the computer B 10003 employs two CPUs, CPU 1 30017 and CPU

2 30018, instead of the single CPU 10015, and the RAM 10016 stores a parallel decryption program PROG5 30025 in addition to the components shown in Fig. 1.

The computer A 10002 executes the parallel encryption
5 program PROG4 30016 to generate ciphertext C 10022 from a message M 10014 and transmit the generated ciphertext C 10022. Receiving the ciphertext C 10022, the computer B 10003 executes the parallel decryption program PROG5 30025 and if no alteration is detected, the computer B 10003
10 stores the decryption results into the RAM 10016.

The CPUs 1 30004 and 2 30005 implement the parallel encryption program PROG4 30016 by executing the program read out from the RAM 10005 in the computer A 10002. The parallel encryption program PROG4 30016 internally calls
15 and executes the encryption program PROG1 10009 and the random number generation program PROG2 10010 as its subroutines.

The CPUs 1 30017 and 2 30018 executes the parallel decryption program PROG5 30025 read out from the RAM 10016
20 in the computer B 10003. The parallel decryption program PROG5 30025 calls and executes the decryption program PROG3 10020 and the random number generation program PROG2 10021 as its subroutines.

The other configurations and operations of the system are the same as those shown in Fig. 1.

Description will be made of the process flow of the parallel encryption program PROG4 30016 with reference to
5 Fig. 25. The expression " $A\|B$ " denotes concatenation of two bit-strings A and B.

Step 40002: divides a message M into two parts, M_1 and M_2 , in message processing performed by the CPU 1.

Step 40003: uses an initial value $V+1$, a redundancy
10 $R+1$, a secret key K, and the plain text M_1 to output ciphertext C_1 in encryption processing by the encryption program PROG1 10009 executed by CPU 1.

Step 40004: uses an initial value $V+2$, a redundancy
 $R+2$, the secret key K, and the plain text M_2 to output
15 ciphertext C_2 in encryption processing by the encryption program PROG1 10009 executed by CPU 2.

Step 40005: uses an initial value V, a redundancy R, the secret key K, and plain text $(R_1\|R_2)$ to output ciphertext C_3 in encryption processing by the encryption
20 program PROG1 10009 executed by CPU 1.

Step 40006: generates ciphertext $C(C=C_1\|C_2\|C_3)$.

Step 40007: stores the ciphertext C into a memory.

Description will be made of the process flow of the parallel decryption program PROG5 30025 with reference to Fig. 26.

Step 40102: divides ciphertext C' into three parts,
5 C'_1 , C'_2 , and C'_3 . C'_3 has 192 bits, and C'_1 and C'_2 has the same length, where $C' = C'_1 \| C'_2 \| C'_3$.

Step 40103: uses the initial value $V+1$ and the secret key K to decrypt the ciphertext block C'_1 into a message block M'_1 and the redundancy $R+1$ in decryption processing
10 by the decryption program PROG3 10020 executed by the CPU 1, and stores the message block M'_1 and the redundancy $R+1$.

Step 40104: uses the initial value $V+2$ and the secret key K to decrypt the ciphertext block C'_2 into a message block M'_2 and the redundancy $R+2$ in decryption processing
15 by the decryption program PROG3 10020 executed by CPU 2, and stores the message block M'_2 and the redundancy $R+2$.

Step 40105: if at least one of the decryption results obtained at steps 40103 and 40104 is a reject, performs step 40111.

20 Step 40106: uses the initial value V and the secret key K to decrypt the ciphertext block C'_3 into a block and the redundancy R in decryption processing by the decryption program PROG3 10020 executed by the CPU 1, and stores the decryption result (the decrypted block) and the redundancy

R.

Step 40107: if the decryption results obtained at step 40106 is a reject, performs step 40111.

Step 40108: if the decrypted block obtained at step 40106 is not equal to $(R+1) \parallel (R+2)$, performs step 40111.

Step 40109: lets $M' = M'_1 \parallel M'_2$ (M' : decryption result).

Step 40110: stores M' into a memory and performs step 40112.

Step 40111: outputs a rejection indication.

As described above, the fifth embodiment can provide parallel cryptographic processing using two separate processors.

Fig. 27 is an explanatory diagram showing the encryption process employed by the above parallel cryptographic processing method.

M_1 40141 and M_2 40142 obtained as a result of dividing a message M 40140 are added with redundancies $R+1$ and $R+2$, respectively, and denoted as blocks 40143 and 40144. The blocks 40143 and 40144 are encrypted by use of encryption processes 40146 and 40147 to obtain ciphertext blocks C_1 40149 and C_2 40150, respectively. Further, a combination of the redundancies $R+1$ and $R+2$, which is set as a message, and another redundancy R are encrypted to obtain a ciphertext block C_3 40151.

The ciphertext blocks C_1 40149, C_2 40150, and C_3 40151 are concatenated one after another to output ciphertext C 40152.

Fig. 28 is an explanatory diagram showing the
5 corresponding parallel decryption process.

Ciphertext C' 40160 is divided into three blocks, C'_1 40161, C'_2 40162, and C'_3 40163. The obtained blocks C'_1 40161, C'_2 40162, and C'_3 40163 are decrypted by decryption processes 40164, 40165, and 40166 to obtain
10 plain text blocks 40167, 40168, and 40169, respectively.

If the obtained plain text blocks are accepted, and the redundancies included in the plain text blocks 40167 and 40168 are identical to the message portions of the plain text block 40169, and furthermore the redundancy
15 included in the plain text block 40169 is equal to the one shared beforehand, the message portions M'_1 40170 and M'_2 40171 are extracted from the plain text blocks 40167 and 40168, respectively, and concatenated to obtain a message M' 40172.

20 Any CPU capable of executing a program can be used for the above embodiments whether it is a general-purpose CPU or a dedicated one. Even though the above embodiments are each implemented by execution of programs by a CPU (or CPUs), dedicated hardware can be used for each process

employed, providing high speed and low cost.

Any of known pseudorandom number generators can be applied to the above embodiments. The known pseudorandom number generators include a pseudorandom generator using a
5 linear feedback shift register (LFSR) with a nonlinear filter, a nonlinear feedback shift register, a combining generator, a shrinking generator, a clock-controlled pseudorandom number generator, a Geffe generator, an alternating step generator, RC4TM, SEAL, PANAMA, the OFB
10 mode of the block cipher, the counter mode of the block cipher, and other pseudorandom number generators using hash functions.

(Sixth Embodiment)

The above first through fifth embodiments each
15 provides a cryptographic processing method. A sixth embodiment of the present invention, on the other hand, shows that the present invention can be applied to various information systems.

Fig. 35 is a diagram showing the configuration of a
20 system in which computers A 50016 and B 50017 are connected through a network 50009 for cryptocommunications from the computer A 50016 to the computer B 50017. The computer A 50016 has a CPU 50007, a RAM 50001, and a network interface device 50008. The RAM 50001 stores key-exchange protocol

software 50002 for executing a key-exchange protocol, a public key 50004 of the authentication center, a secret-key generation program 50003, an encryption program 50006, and communication data 50005 (corresponding to the message M in
5 each embodiment described above) to be transmitted using cryptocommunications. The computer B 50017 has a CPU 50014, a RAM 50010, and a network interface device 50015 therein. The RAM 50010 stores key-exchange protocol software 50011 and a decryption program 50013.

10 The computer A executes the secret-key generation program 50003 to generate a secret key used for cryptocommunications with the computer B 50017. The computers A 50016 and B 50017 execute the key-exchange protocol software 50002 and 50011, respectively, to share
15 the secret key generated by the computer A.

After sharing the secret key, the computer A 50016 executes the encryption program 50006 of the present invention to encrypt the communication data 50005 at high speed. The computer A 50016 then transmits the encryption
20 results to the computer B 50017 through the network 50009 using the network interface device 50008.

The computer B 50017 executes the decryption program 50013 of the present invention to decrypt received ciphertext at high speed to restore the communication data.

This embodiment shows that the present invention can provide high-speed and safe cryptocommunications even when available hardware resources are limited. That is, the present invention is capable of realizing a highly safe
5 cryptocommunication system which is faster than the conventional cryptographic method, and provides confidentiality as well as a mathematically proven alteration detection function.

(Seventh Embodiment)

10 The above sixth embodiment performs cryptographic processing by use of software. A seventh embodiment of the present invention, on the other hand, shows that the present invention can be realized by hardware implementation.

15 Fig. 36 is a diagram showing the configuration of an encryption apparatus employed in a cryptocommunication system using a network. The computer 50110 has a RAM 50101, a CPU 50104, and a network interface device 50105, and is connected to a network 50106. The RAM 50101 stores
20 communication data 50103 (corresponding to the message M in each embodiment described above) to be encrypted and a communication program 50102, and the CPU 50104 executes the communication program 50102 to output the communication data 50103 to the network interface device 50105. The

network interface device 50105 includes a secret-key generation circuit 50107, an encryption circuit 50109, and a key-exchange protocol circuit 50108, and has a public key 50110 of the authentication center stored in its memory
5 area. According to the execution of the communication program 50102, the network interface device 50105 generates a secret key by use of the secret-key generation circuit 50107, and exchanges the generated secret key with another device on the network using the key-exchange protocol
10 circuit 50108 so as to share the generated secret key with the communication destination device. The encryption circuit 50109 in the network interface device 50105 encrypts the input communication data 50103 at high speed using the generated and then shared secret key to generate
15 ciphertext, which is then output to the network 50106.

This embodiment shows that the present invention can provide safe and fast cryptocommunications using limited hardware resources. Particularly, if this embodiment is combined with the cryptographic processing method of the
20 second embodiment, more efficient and safe cryptocommunications can be realized. This is because addition and multiplication in the finite field F_2^{64} employed in the second embodiment are suitable for hardware implementation. The decryption process can also be

implemented by hardware in the same way.

As shown by this embodiment, the present invention can provide a cryptographic method whose hardware implementation requires a small number of gates or performs
5 very high-speed processing.

(Eighth Embodiment)

By using a computer capable of performing cryptographic processing employed in the sixth or seventh embodiment, it is possible to easily realize a contents
10 delivery protected by encryption. An eighth embodiment of the present invention shows an example of a contents delivery.

As shown in Fig. 37, a storage device (whose medium is not limited to a specific type, that is, it is possible to
15 use a semiconductor storage device, a hard disk, a magnetic storage device such as one using tape, or an optical storage device such as a DVD or an MO) storing contents 50201 as digital information is connected to a computer 50202 capable of performing encryption processing according
20 to the present invention. An information reproduction device 50205 (an MPEG reproduction device, a digital TV, a personal computer, etc.) which is to reproduce contents and may be located in a physically remote place is connected to an external coding device 50204 capable of performing

decryption processing according to the present invention.

The computer 50202 and the external coding device are connected to each other through a network 50203.

The contents 50201 is encrypted by the computer 50202
5 capable of encryption, and then transmitted to the network
50203. The external coding device 50204 capable of
decryption decrypts the encrypted contents, and outputs the
decryption results to the information reproduction device
50205. The information reproduction device 50205 stores
10 and reproduces input information.

The contents 50201 handled by the information
reproduction device 50205 include not only electronic files
but also multimedia data such as computer software, sound,
and image. Contents which require real-time processing,
15 such as sound and movie, can be encrypted or decrypted at
high speed by applying the present invention, making it
possible to secure smooth real-time transmission.
Furthermore, the receiving device can detect data
corruption due to alteration or noise during the
20 transmission, ensuring communications free of transmission
errors.

(Ninth Embodiment)

The eighth embodiment delivers contents by
transmission through a network. When it is necessary to

deliver a very large amount of information, however, it is more efficient to deliver ciphertext on a DVD, etc. beforehand, and then transmit the decryption key at the time of permitting the decryption of the ciphertext. Such
5 a system is provided by a ninth embodiment.

As shown in Fig. 38, contents are distributed to the consumer as ciphertext, using a medium such as a DVD-ROM 50307, beforehand. The consumer enters information 50306 (money transfer information) on payment for contents using
10 a contents-key exchange program 50305 running on the consumer's personal computer 50304. The contents-key exchange program 50305 then obtains a key from a contents-key table in a key server 50302 through a network 50303. A decryption program 50308 decrypts the ciphertext contents
15 recorded on the DVD-ROM 50307 using the obtained key. The decryption results are output to the information reproduction device 50309 which then reproduces the contents.

This embodiment may be configured such that the
20 contents are not output to the information reproduction device 50309, and the personal computer 50304 itself reproduces them. In a typical example, the contents is a program to be executed on a personal computer. The above reproduction method of using a personal computer is

efficient in such a case. When ciphertext contents recorded on a DVD-ROM can be divided into several parts, and each part is encrypted using a different key, it is possible to control keys transmitted to the contents-key acquisition program 50305 so as to limit contents which can be decrypted by the consumer.

The ninth embodiment was described assuming that data recorded on the DVD-ROM 50307 is to be read out. Generally, a very large amount (a few tens of megabytes to a few hundreds of megabytes) of data is stored on the DVD-ROM 50307, and therefore a fast cryptographic processing method is required for processing such data. Since the present invention can provide high-speed decryption, the present invention is suitably applied to distribution of charged contents using a DVD medium.

(Tenth Embodiment)

In a tenth embodiment of the present invention, the present invention is applied to a router which controls packet transfer on a network. This router encrypts packets differently depending on the destination router of each packet at the time of their transmission to the network.

Fig. 39 is a diagram showing the configuration of a cryptographic router. The network router 50401 has a routing table 50402, a packet exchanger 50403, network

interfaces A 50404, B 50405, and C 50406, and an internal parallel encryption/decryption device 50410 therein. The network interfaces A 50404, B 50405, and C 50406 are connected to external networks A 50407, B 50408, and C
5 50409, respectively.

The internal parallel encryption/decryption device 50410 has a secret-key table 50411, a router-key storage area 50412, and a parallel encryption/decryption circuit 50413 therein.

10 A packet sent from the network A 50407 is transmitted to the internal parallel encryption/decryption device 50410 through the network interface A 50404. After recognizing that the received packet is originated from the network A 50407, the internal parallel encryption/decryption device
15 50410 refers to the secret-key table 50411 to obtain the secret key corresponding to the network A 50407, stores the obtained secret key in the router-key storage area 50412, and decrypts the packet using the parallel encryption/decryption circuit 50413. The internal parallel
20 encryption/decryption device 50410 then transmits the decryption results to the packet exchanger 50403.

The following description assumes that this decrypted packet should be transmitted to the network B. The packet exchanger 50403 transfers the packet to the internal

parallel encryption/decryption device 50410. The internal parallel encryption/decryption device 50410 refers to the secret-key table 50411 to obtain the secret key corresponding to the network B 50408, stores the obtained
5 secret key in the router-key storage area 50412, and encrypts the packet using the parallel encryption/decryption circuit 50413. The internal parallel encryption/decryption device 50410 then transmits the encryption results to the network interface B 50405 which,
10 in turn, transmits the received encrypted packet to the network B 50408.

This embodiment is applied to an application used in an environment in which a large quantity of hardware resources are available and which requires
15 cryptocommunications at very high speed. In the CBC mode of the block cipher in which parallel processing is difficult to employ, it is difficult to enhance its processing speed even when a large quantity of hardware resources are available. In contrast, parallel processing
20 is very easy to employ in the present invention (providing a high level of parallel operation) since the pseudorandom

number generation process is independent from the plain
text and ciphertext processing. That is, the present
invention can attain a higher communication speed in the
environment in which a large quantity of hardware resources
5 suitable for parallel processing are available.

What is claimed is:

1. A symmetric-key encryption method performed in a cryptographic system having input means for inputting at least a message, a secret data, and a parameter opened to the public, and output means for outputting an encrypted data, said encryption method comprising the steps of:

dividing a plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length;

generating a random number sequence based on a secret key;

generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of plain text blocks;

outputting a first feedback value to a second one of said plurality of plain text blocks obtained as a result of an operation on a first one of said plurality of plain text blocks and a first one of said random number blocks; and

performing an encryption operation using said first one of said plurality of plain text blocks, said first one of said random number blocks, and a second

feedback value obtained as a result of an operation on a third one of the plurality of plain text blocks, to produce a ciphertext block.

5 2. The symmetric-key encryption method as claimed in claim 1, wherein said encryption operation uses one or more said random number blocks whose total length is longer than a length of said ciphertext block.

10 3. The symmetric-key encryption method as claimed in claim 2, wherein said plain text further includes secret data of a predetermined length.

 4. The symmetric-key encryption method as claimed in
15 claim 2, wherein said encryption operation performs at least one of a binary operation and a monadic operation using one of said plurality of plain text blocks one or more times according to a predetermined procedure, combines a plurality of obtained ciphertext blocks and outputs the
20 combined plurality of ciphertext blocks as ciphertext.

 5. The symmetric-key encryption method as claimed in claim 2, wherein said encryption operation includes multiplication and addition in a finite field.

6. The symmetric-key encryption method as claimed in claim 2, wherein said encryption operation includes a combination of a cyclic shift operation and arithmetic multiplication.

5

7. The symmetric-key encryption method as claimed in claim 2, wherein said symmetric-key encryption method employs a pseudorandom-number generating means for generating said random number sequence based on said secret
10 key.

8. The symmetric-key encryption method of claim 7, further comprising the steps of:

dividing said message into a plurality of message
15 blocks;

generating a number of random number sequences equal to the number of said plurality of message blocks using said pseudorandom-number generating means; and

assigning a first one of said message blocks, and
20 a first one of said random number sequences to a first operation unit;

assigning a second one of said message blocks, and a second one of said random number sequences to a second operation unit; and

performing parallel processing by said first and second operation units.

9. A symmetric-key decryption method performed in a cryptographic system having input means for inputting at least a message, an encrypted data, and a parameter opened to the public, and output means for outputting a decrypted data, said decryption method comprising the steps of:

dividing a ciphertext to generate a plurality of ciphertext blocks each having a predetermined length;

generating a random number sequence based on a secret key;

generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of ciphertext blocks;

outputting a first feedback value to a second one of said plurality of ciphertext blocks obtained as a result of an operation on a first one of said plurality of ciphertext blocks and a first one of said random number blocks; and

performing a decryption operation using said first one of said plurality of ciphertext blocks, said first one of said random number blocks, and a second

feedback value obtained as a result of an operation on a third one of the plurality of ciphertext blocks, to produce a plain text block.

5 10. The symmetric-key decryption method as claimed in claim 9, wherein said decryption operation uses one or more said random number blocks whose total length is longer than a length of said one of the plurality of ciphertext blocks.

10 11. The symmetric-key decryption method as claimed in claim 10, further comprising steps of:

concatenating a plurality of said plain text blocks to generate plain text;

15 extracting redundancy data included in said plain text; and

checking said redundancy data to detect whether said ciphertext has been altered.

12. The symmetric-key decryption method as claimed in claim 11, further comprising steps of:

extracting secret data included in said plain text; and

checking said redundancy data and said secret data to detect whether said ciphertext has been altered.

13. A symmetric-key encryption apparatus for use in a cryptographic system having input means for inputting at least a message, a secret data, and a parameter opened to the public, and output means for outputting an encrypted data, said encryption apparatus comprising:

a circuit for dividing a plain text composed of redundancy data and a message to generate a plurality of plain text blocks each having a predetermined length;

a circuit for generating a random number sequence based on a secret key;

a circuit for generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of plain text blocks;

a circuit for outputting a first feedback value to a second one of said plurality of plain text blocks obtained as a result of an operation on a first one of said plurality of plain text blocks and a first one of said random number blocks; and

a circuit for performing an encryption operation using said first one of said plurality of plain text blocks, said first one of said random number blocks, and a second feedback value obtained as a result of an operation on a third one of the plurality of plain text blocks, to

produce a ciphertext block.

14. The symmetric-key encryption apparatus as claimed in claim 13, wherein said encryption operation circuit uses
5 one or more said random number blocks whose total length is longer than a length of said ciphertext block.

15. The symmetric-key encryption apparatus as claimed in claim 14, wherein said plain text further includes
10 secret data of a predetermined length.

16. The symmetric-key encryption apparatus of claim 14, wherein said encryption operation circuit includes:

15 a circuit for performing at least one of a binary operation and a monadic operation using one of said plurality of plain text blocks one or more times according to a predetermined procedure; and

a circuit for combining a plurality of obtained
20 ciphertext blocks, and outputting the combined plurality of ciphertext blocks as ciphertext.

17. The symmetric-key encryption apparatus as claimed in claim 14, wherein said encryption operation circuit performs multiplication and addition in a finite field.

5 18. The symmetric-key encryption apparatus as claimed in claim 14, wherein said encryption operation circuit includes a cyclic shift operation circuit and an arithmetic multiplication circuit.

10 19. The symmetric-key encryption apparatus as claimed in claim 14, further comprising: a pseudorandom-number generator for generating said random number sequence based on said secret key.

15 20. The symmetric-key encryption apparatus of claim 19, further comprising:

a circuit for dividing said message into a plurality of message blocks;

a circuit for generating a number of random
20 number sequences equal to the number of said plurality of message blocks using said pseudorandom-number generating means; and

a circuit for assigning a first one of said message blocks, and a first one of said random number sequences to a first operation unit;

a circuit for assigning a second one of said message blocks, and a second one of said random number sequences to a second operation unit; and

a circuit for performing parallel processing by said first and second operation units.

10 21. A symmetric-key decryption apparatus for use in a cryptographic system having input means for inputting at least a message, an encrypted data, and a parameter opened to the public, and output means for outputting a decrypted data, said decryption apparatus comprising:

15 a circuit for dividing a ciphertext to generate a plurality of ciphertext blocks each having a predetermined length;

a circuit for generating a random number sequence based on a secret key;

20 a circuit for generating a plurality of random number blocks from said random number sequence, each of said random number blocks corresponding to one of said plurality of ciphertext blocks;

a circuit for outputting a first feedback value
to a second one of said plurality of ciphertext blocks
obtained as a result of an operation on a first one of said
plurality of ciphertext blocks and a first one of said
5 random number blocks; and

a circuit for performing a decryption operation
using said first one of said plurality of ciphertext
blocks, said first one of said random number blocks, and a
second feedback value obtained as a result of an operation
10 on a third one of the plurality of ciphertext blocks, to
produce a plain text block.

22. The symmetric-key decryption apparatus as claimed
in claim 21, wherein said decryption operation circuit uses
15 one or more said random number blocks whose total length is
longer than a length of said one of the plurality of
ciphertext blocks.

23. The symmetric-key decryption apparatus as claimed
20 in claim 22, further comprising:

a circuit for concatenating a plurality of said
plain text blocks to generate plain text;

a circuit for extracting redundancy data included
in said plain text; and

a circuit for checking said redundancy data to detect whether said ciphertext has been altered.

24. The symmetric-key decryption apparatus as claimed
5 in claim 23, further comprising: a circuit for extracting secret data included in said plain text, wherein said circuit for detecting whether said ciphertext has been altered checks said secret data and said redundancy data to detect whether said ciphertext has been altered.

10

25. A computer-readable recording medium having recorded thereon statements and instructions for use in the execution in a computer of said symmetric-key encryption method of claim 1.

15

26. The medium storing a program as claimed in claim 25, wherein said encryption operation uses one or more said random number block whose total length is longer than a length of said ciphertext block.

20

27. The medium storing a program as claimed in claim 26, wherein said plain text further includes secret data of a predetermined length.

28. The medium of claim 26, wherein said encryption operation performs at least one of a binary operation and a monadic operation using one of said plurality of plain text blocks one or more times according to a predetermined
5 procedure, combines a plurality of obtained ciphertext blocks, and outputs the combined plurality of ciphertext blocks as ciphertext.

29. The medium storing a program as claimed in
10 claim 26, wherein said encryption operation includes multiplication and addition in a finite field.

30. The medium storing a program as claimed in claim 26, wherein said encryption operation includes a
15 cyclic shift operation and arithmetic multiplication.

31. The medium storing a program as claimed in claim 26, wherein said symmetric-key encryption method further comprises a step of: generating pseudorandom
20 numbers to generate said random number sequence based on said secret key.

32. The medium of claim 31, wherein said symmetric-key encryption method further comprises the steps of:

dividing said message into a plurality of message blocks;

5 generating a number of random number sequences equal to the number of said plurality of message blocks using said pseudorandom-number generating means; and

assigning a first one of said message blocks, and a first one of said random number sequences to a first
10 operation unit;

assigning a second one of said message blocks, and a second one of said random number sequences to a second operation unit; and

performing parallel processing by said first and
15 second operation units.

33. A computer-readable recording medium having recorded thereon statements and instructions for use in the execution in a computer of said symmetric-key decryption
20 method of claim 9.

34. The medium storing a program as claimed in claim 33, wherein said decryption operation uses one or more said random number blocks whose total length is longer

than a length of said one of the plurality of ciphertext blocks.

35. The medium of claim 34, wherein said symmetric-
5 key decryption method further comprises the steps of:

concatenating a plurality of said plain text
blocks to generate plain text;

extracting redundancy data included in said plain
text; and

10 checking said redundancy data to detect whether
said ciphertext has been altered.

36. The medium of claim 35, wherein said symmetric-
key decryption method further comprises the steps of:

15 extracting secret data included in said plain
text; and

checking said redundancy data and said secret
data to detect whether said ciphertext has been altered.

20 37. A computer program product, comprising: a memory
having computer-readable code embodied therein for
implementing a symmetric-key encryption method performed in
a cryptographic system having input means for inputting at
least a message, a secret data, and a parameter opened to

the public, and output means for outputting an encrypted data, said computer program product comprising:

code means for dividing a plain text composed of redundancy data and a message to generate a plurality of
5 plain text blocks each having a predetermined length;

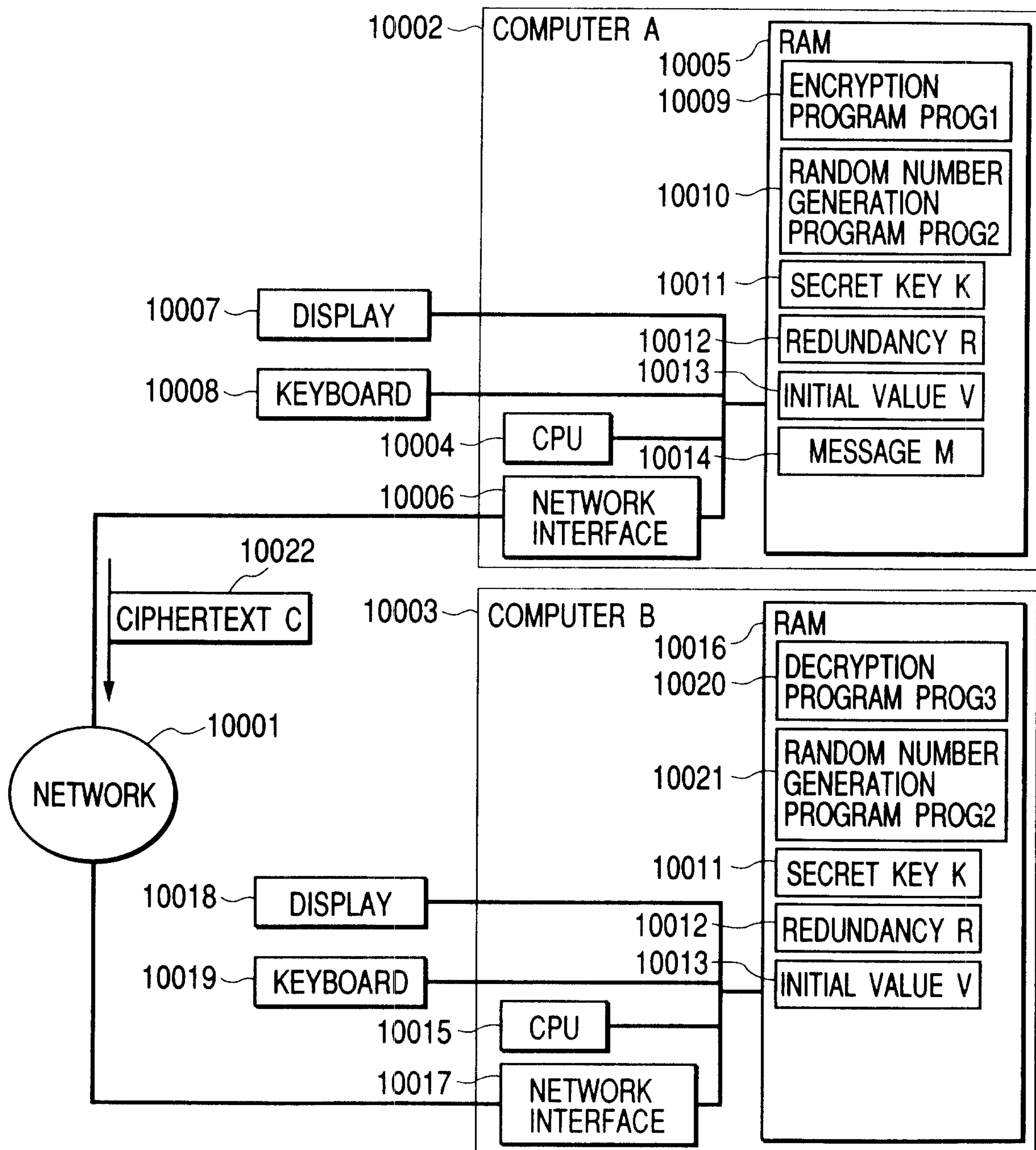
code means for generating a random number sequence based on a secret key;

code means for generating a plurality of random number blocks from said random number sequence, each of
10 said random number blocks corresponding to one of said plurality of plain text blocks;

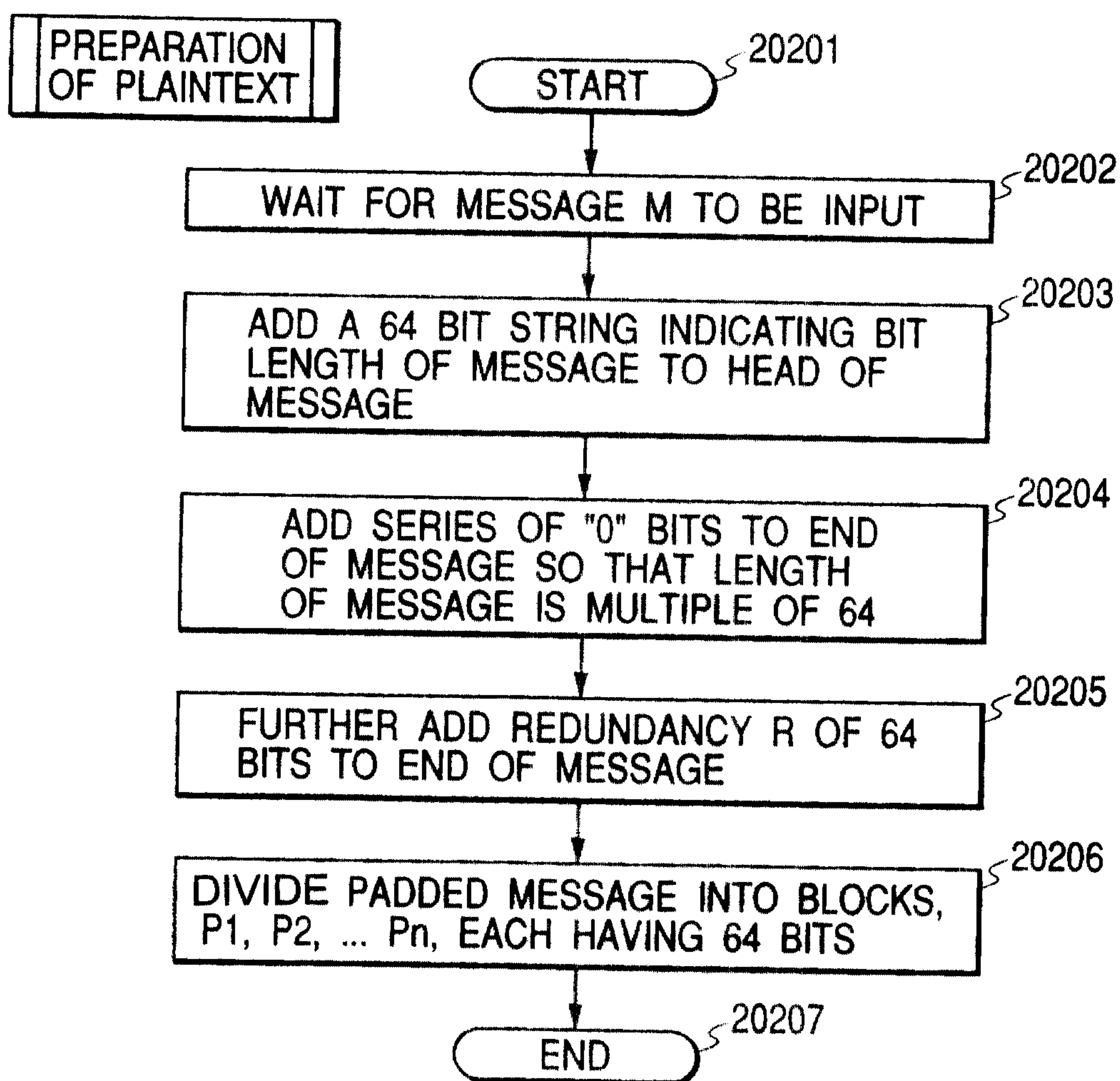
code means for outputting a first feedback value to a second one of said plurality of plain text blocks obtained as a result of an operation on a first one of said
15 plurality of plain text blocks and a first one of said random number blocks; and

code means for performing an encryption operation using said first one of said plurality of plain text blocks, said first one of said random number blocks, and a
20 second feedback value obtained as a result of an operation on a third one of the plurality of plain text blocks, to produce a ciphertext block.

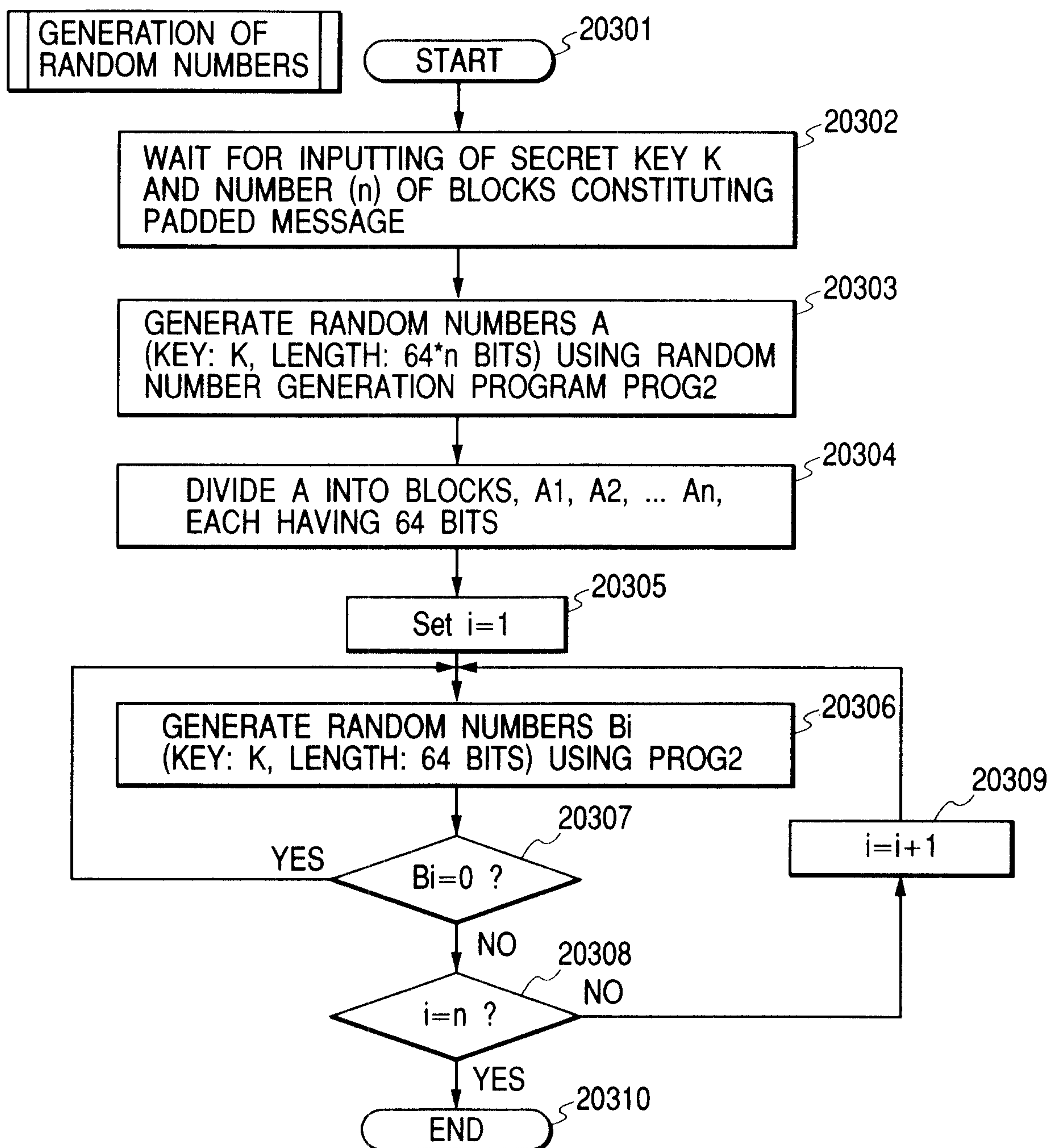
1 / 34

FIG. 1

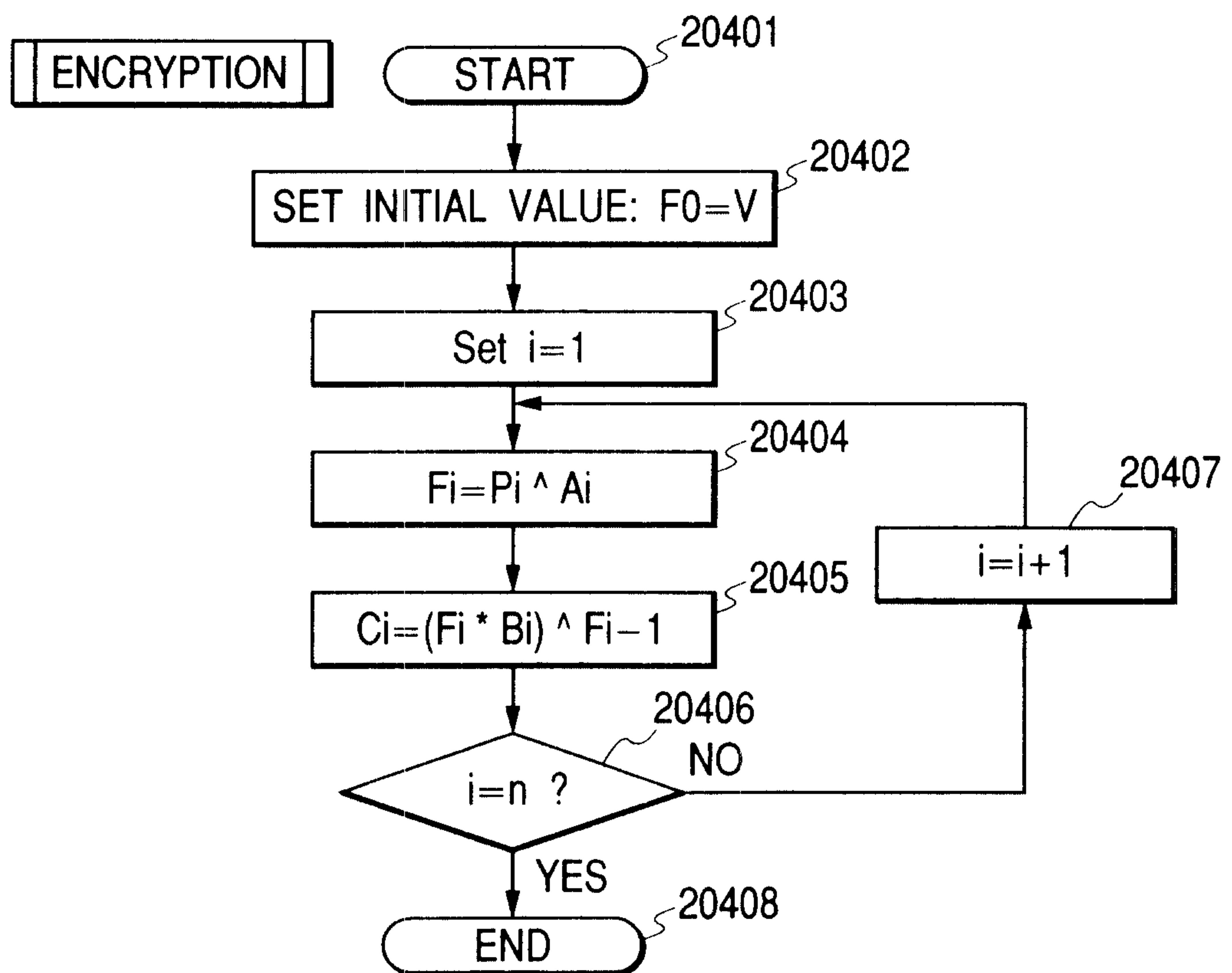
2 / 34

FIG. 2

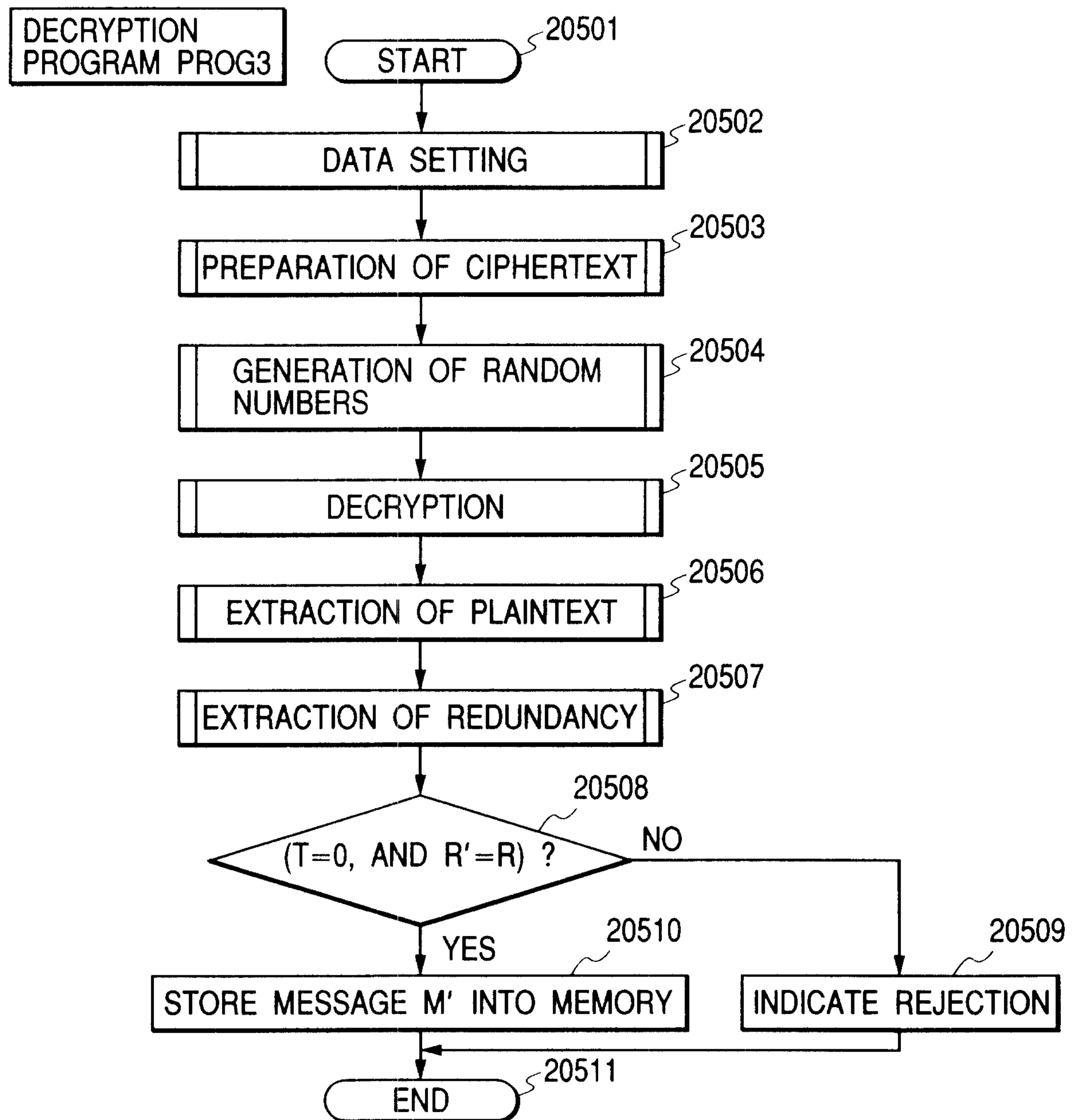
3 / 34

FIG. 3

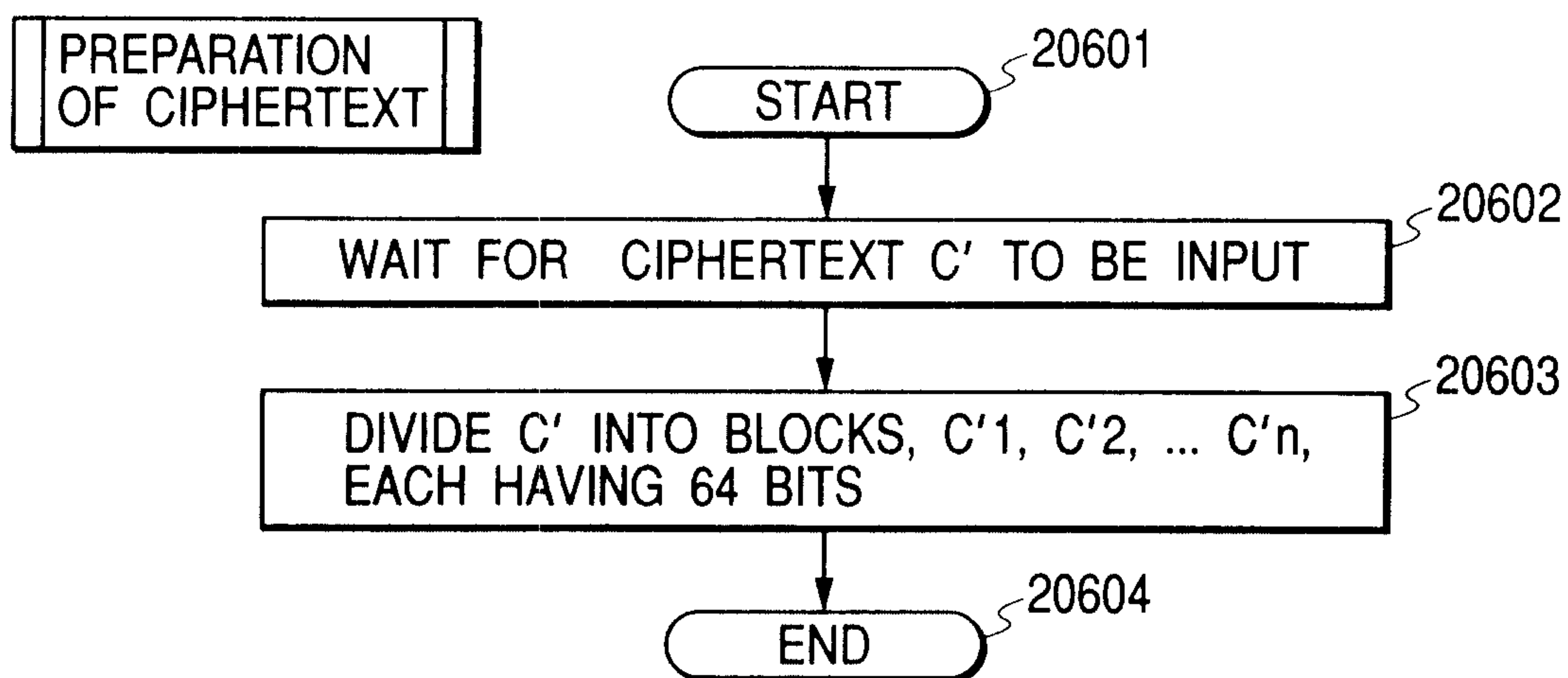
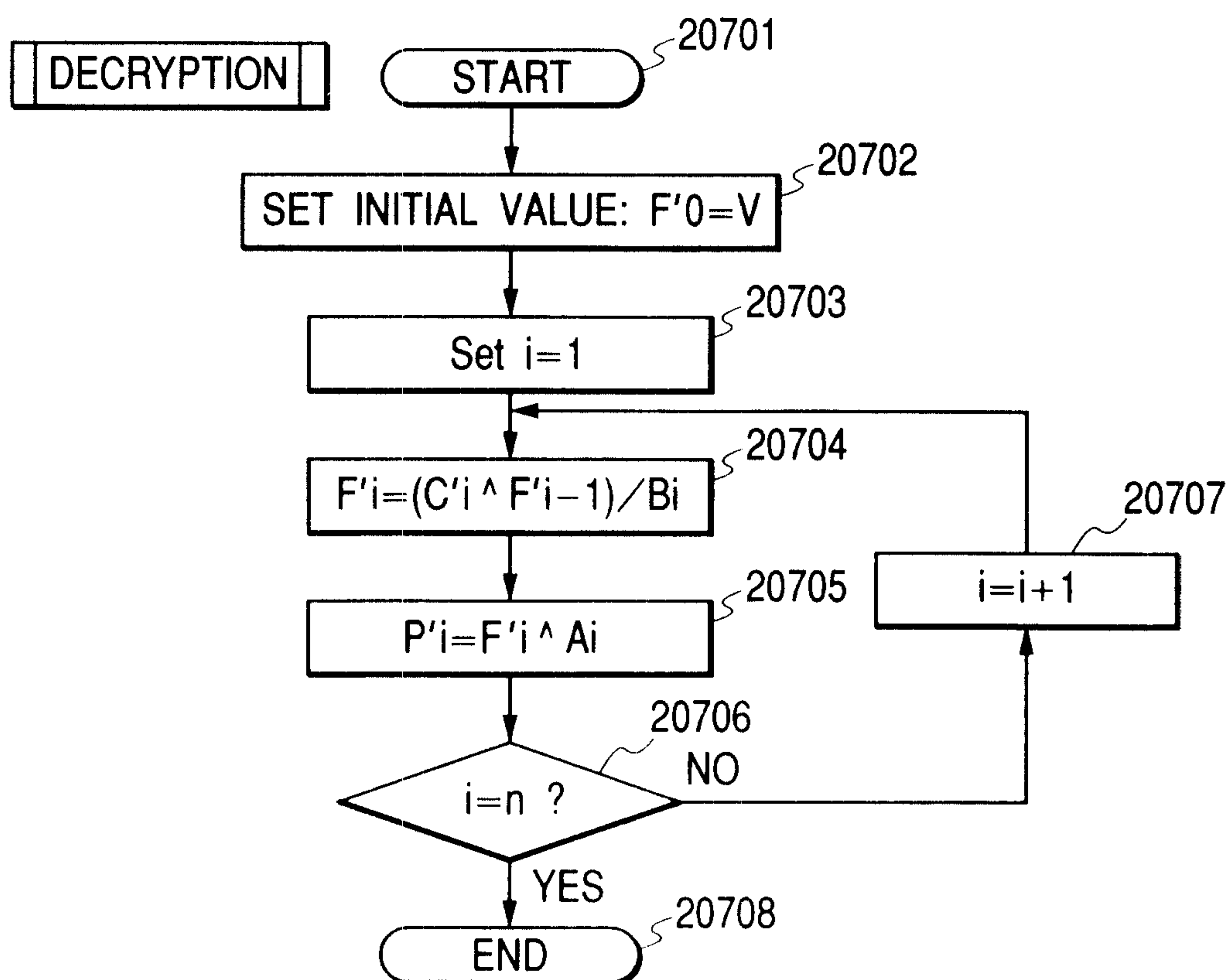
4 / 34

FIG. 4

5 / 34

FIG. 5

6 / 34

FIG. 6**FIG. 7**

7 / 34

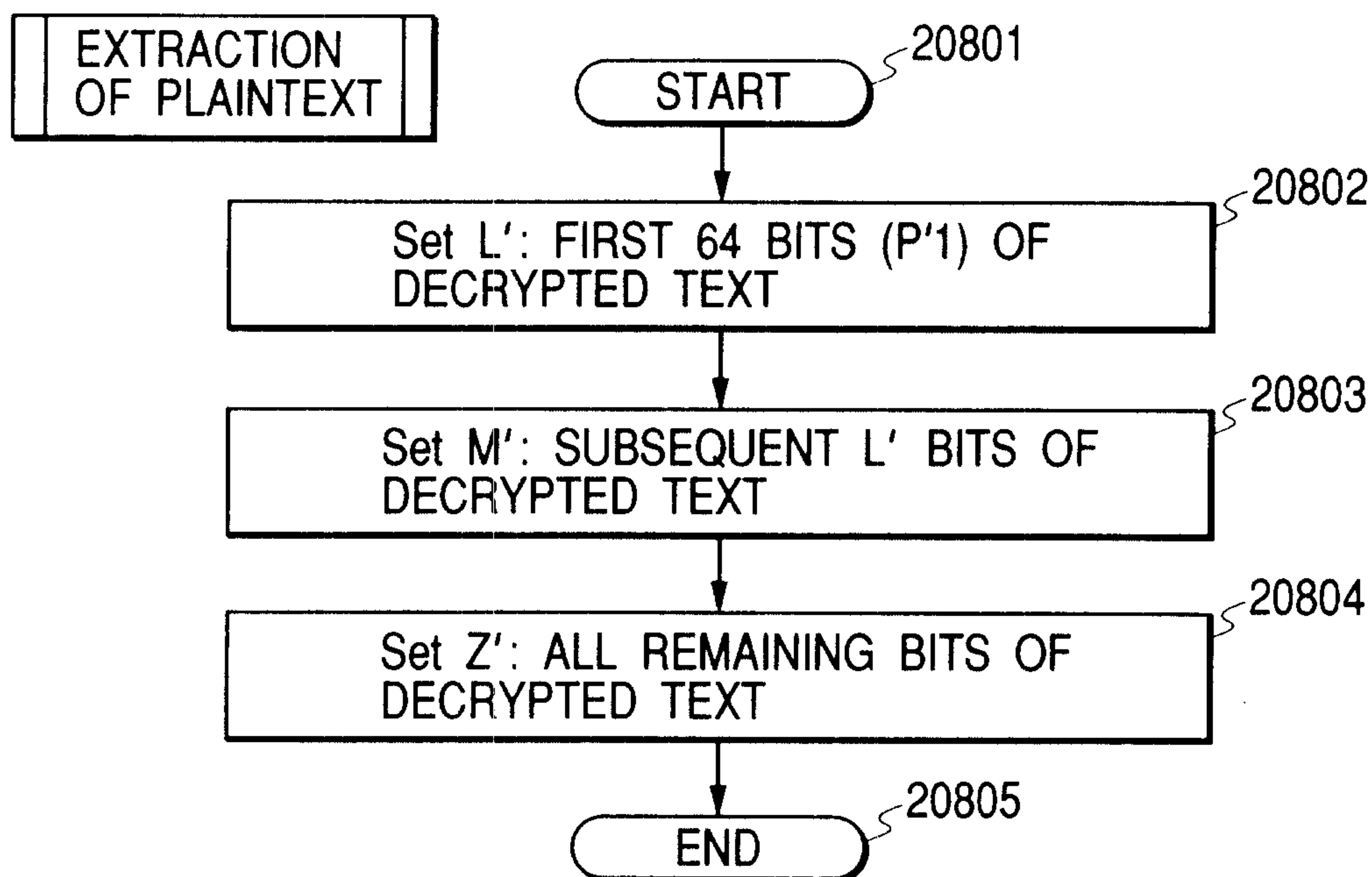
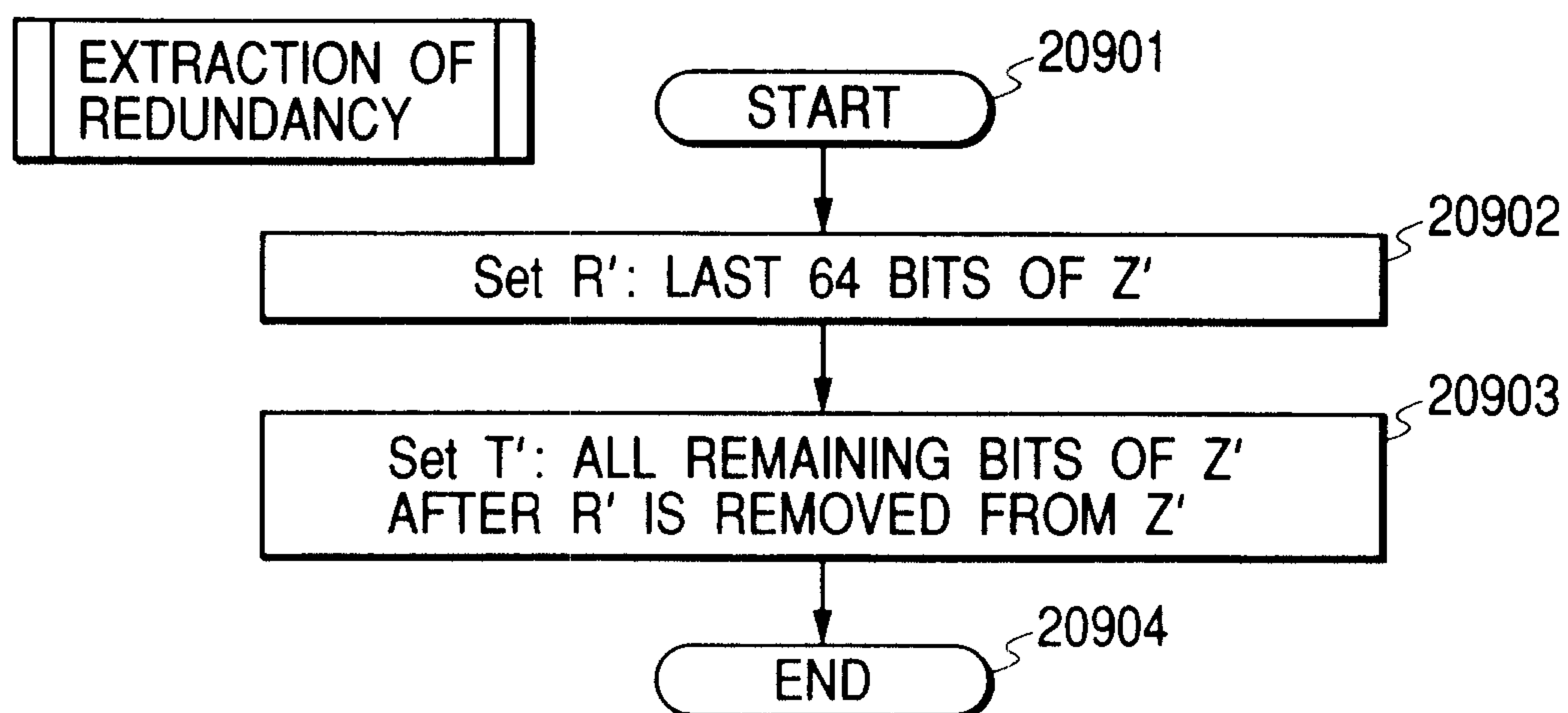
FIG. 8**FIG. 9**

FIG. 10

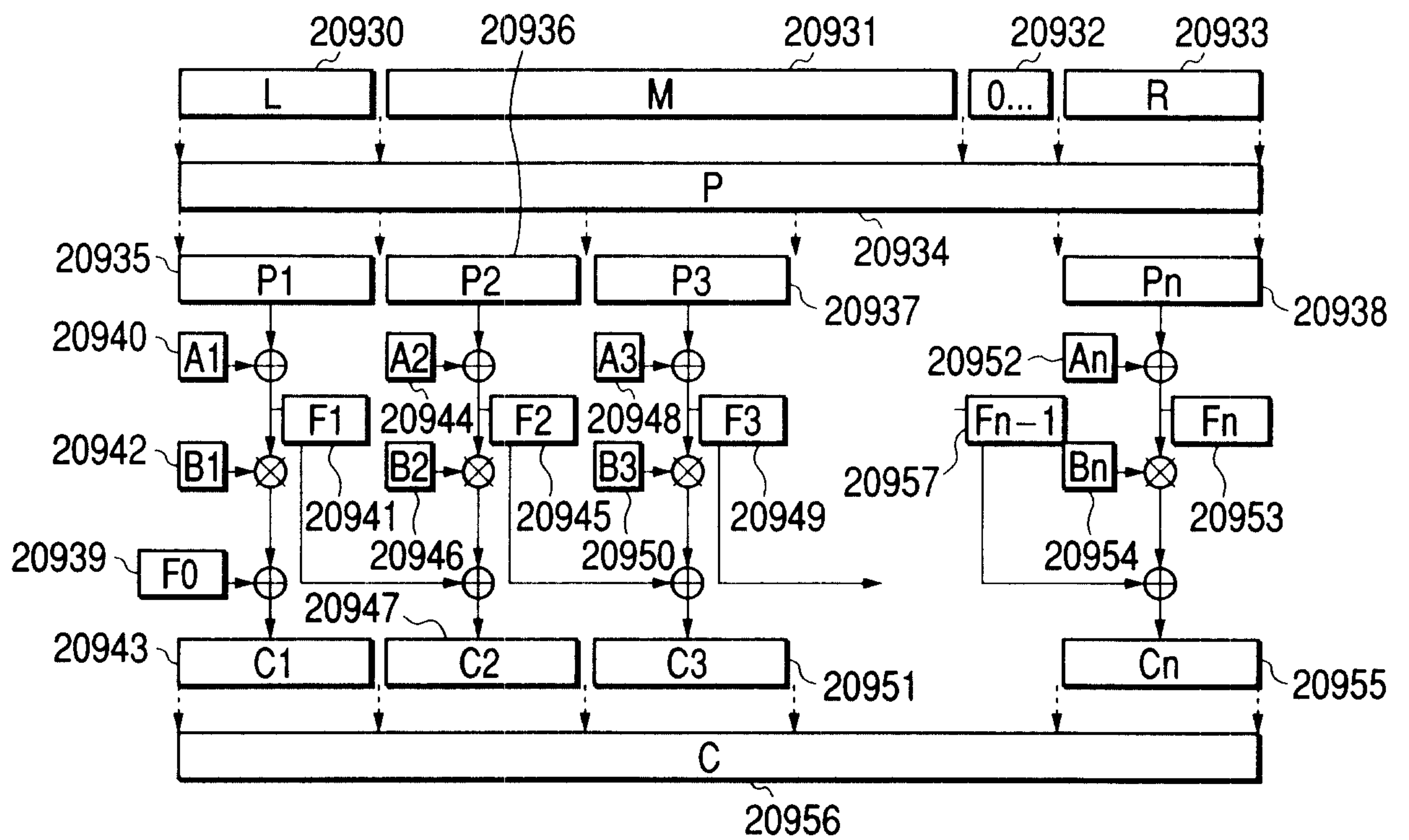
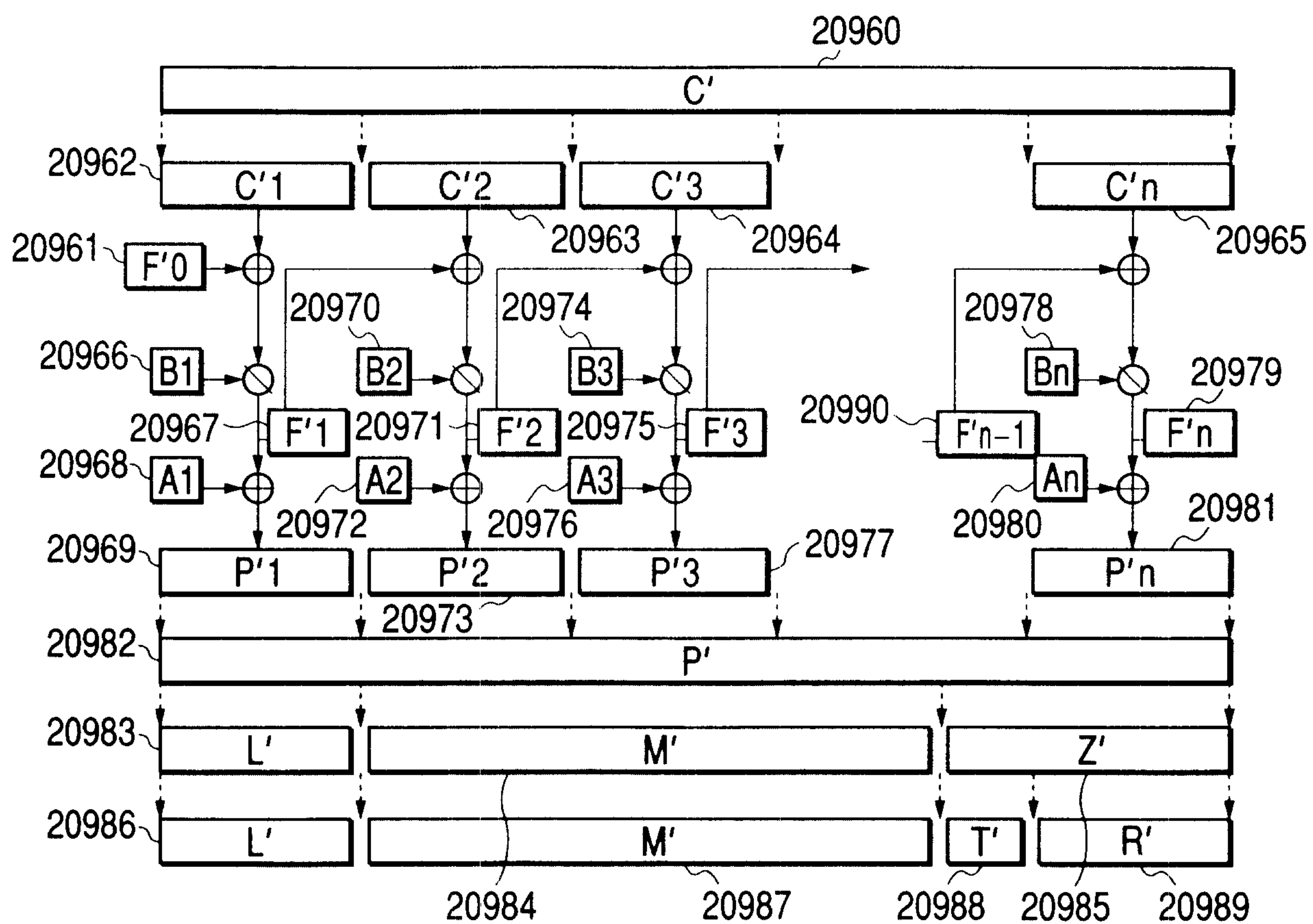
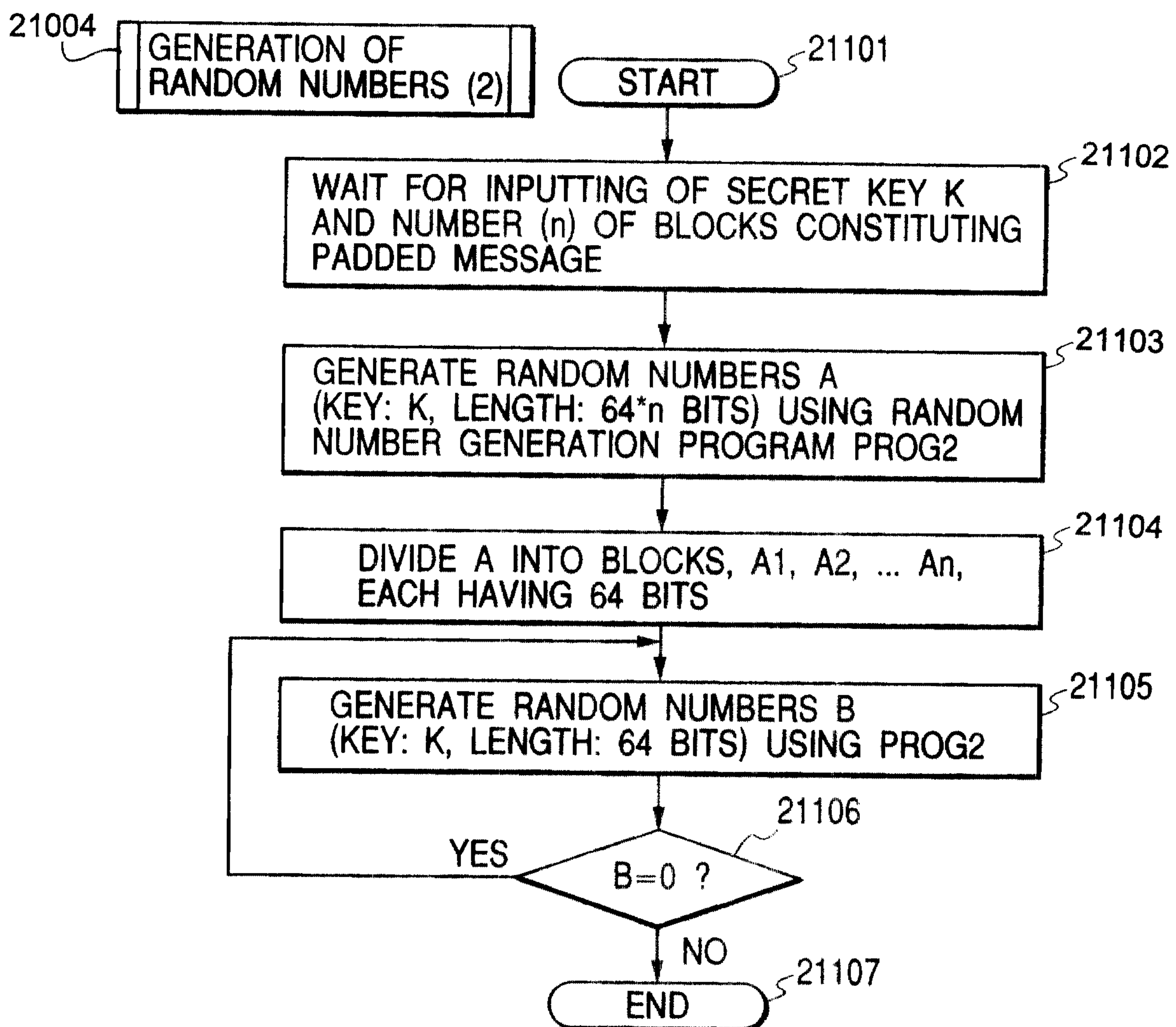
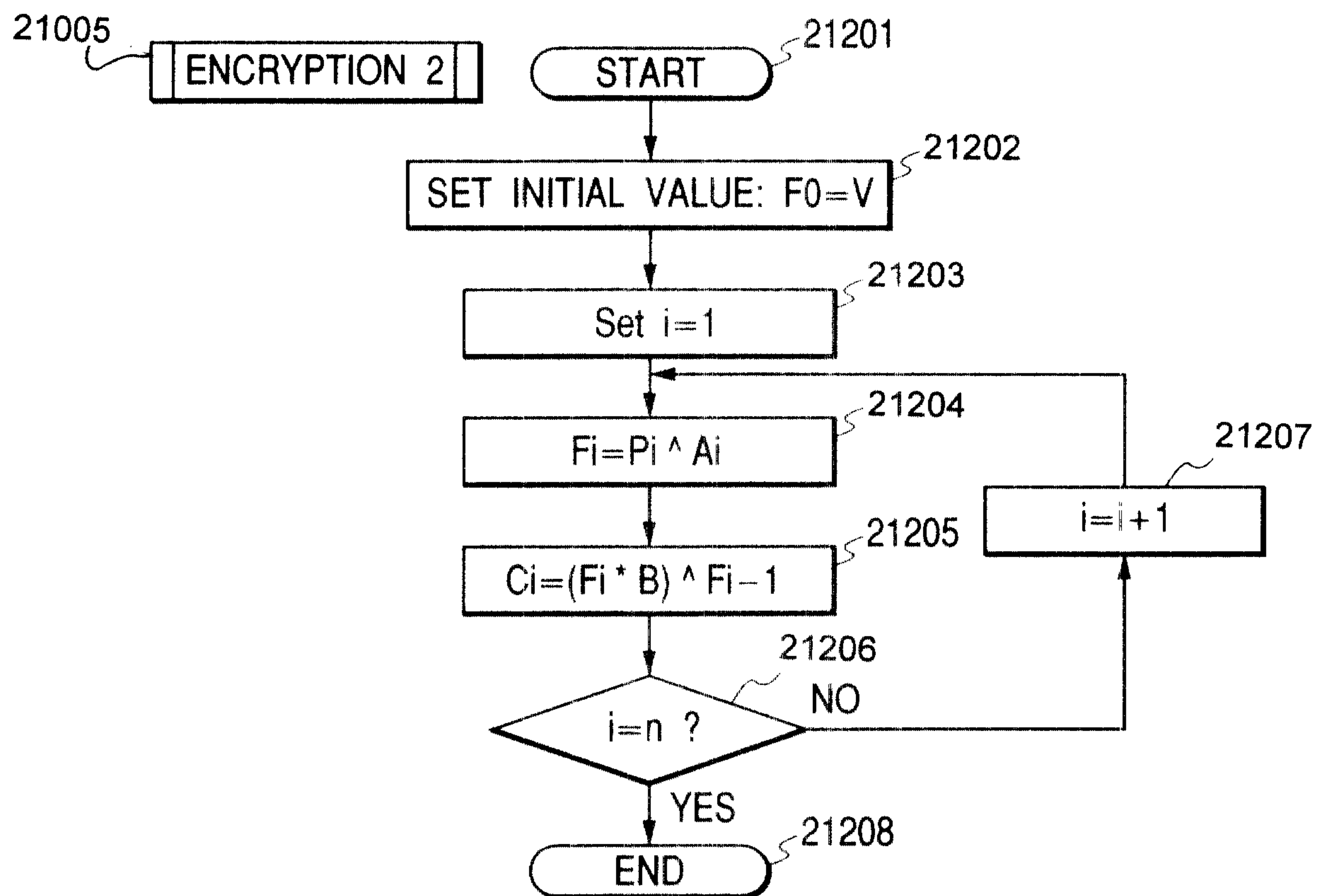


FIG. 11

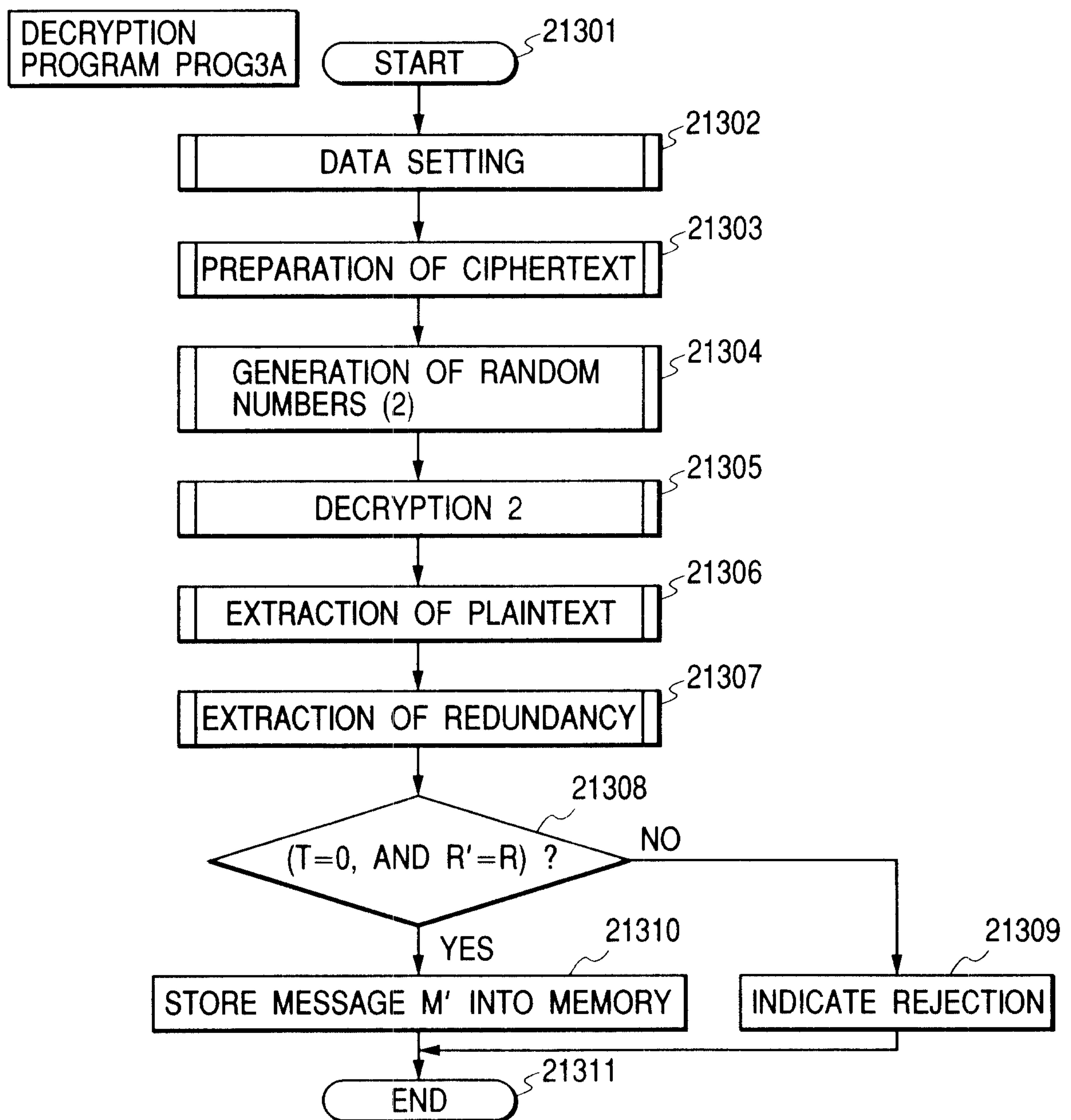
10 / 34

FIG. 12

11 / 34

FIG. 13

12 / 34

FIG. 14

13 / 34

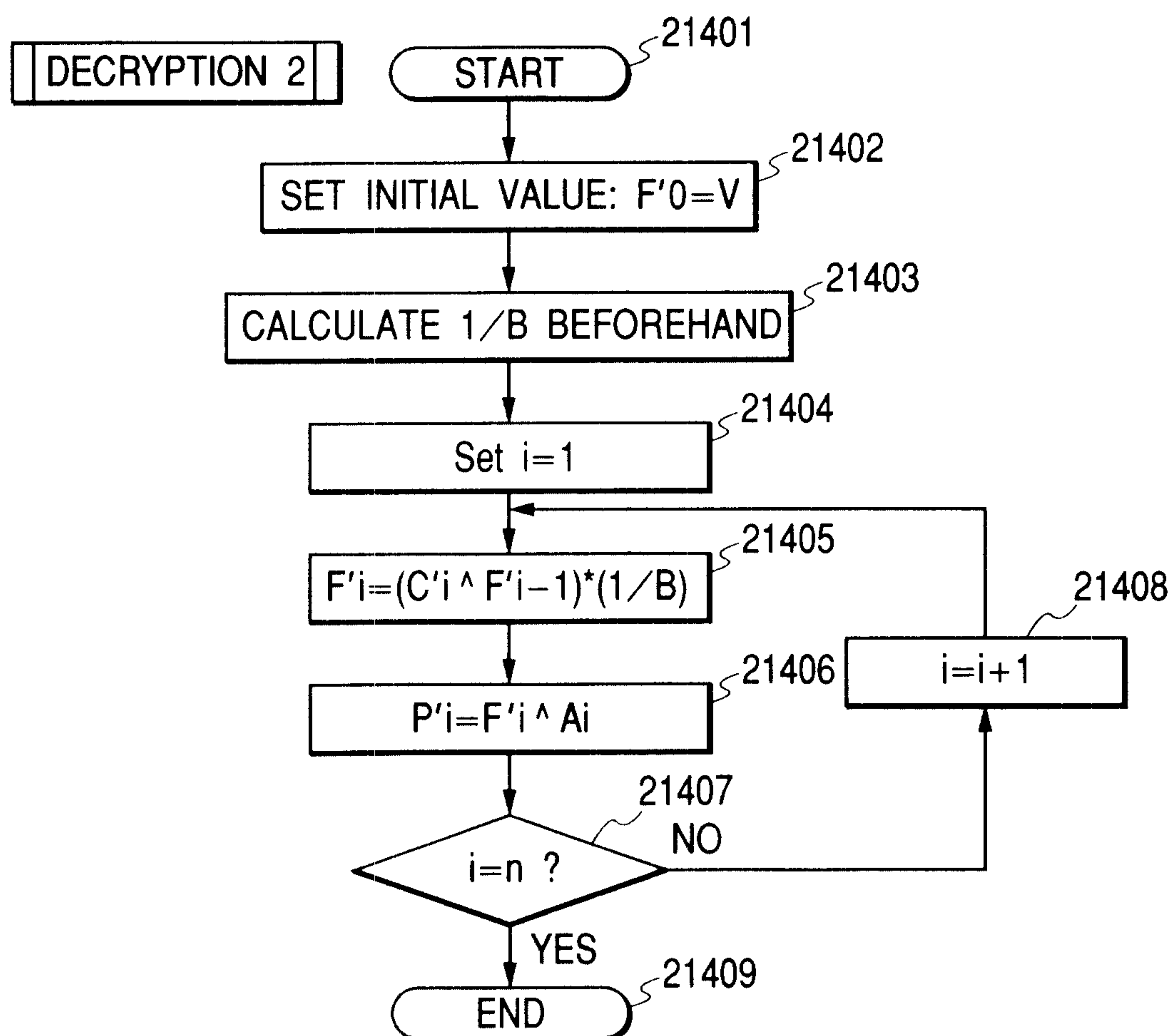
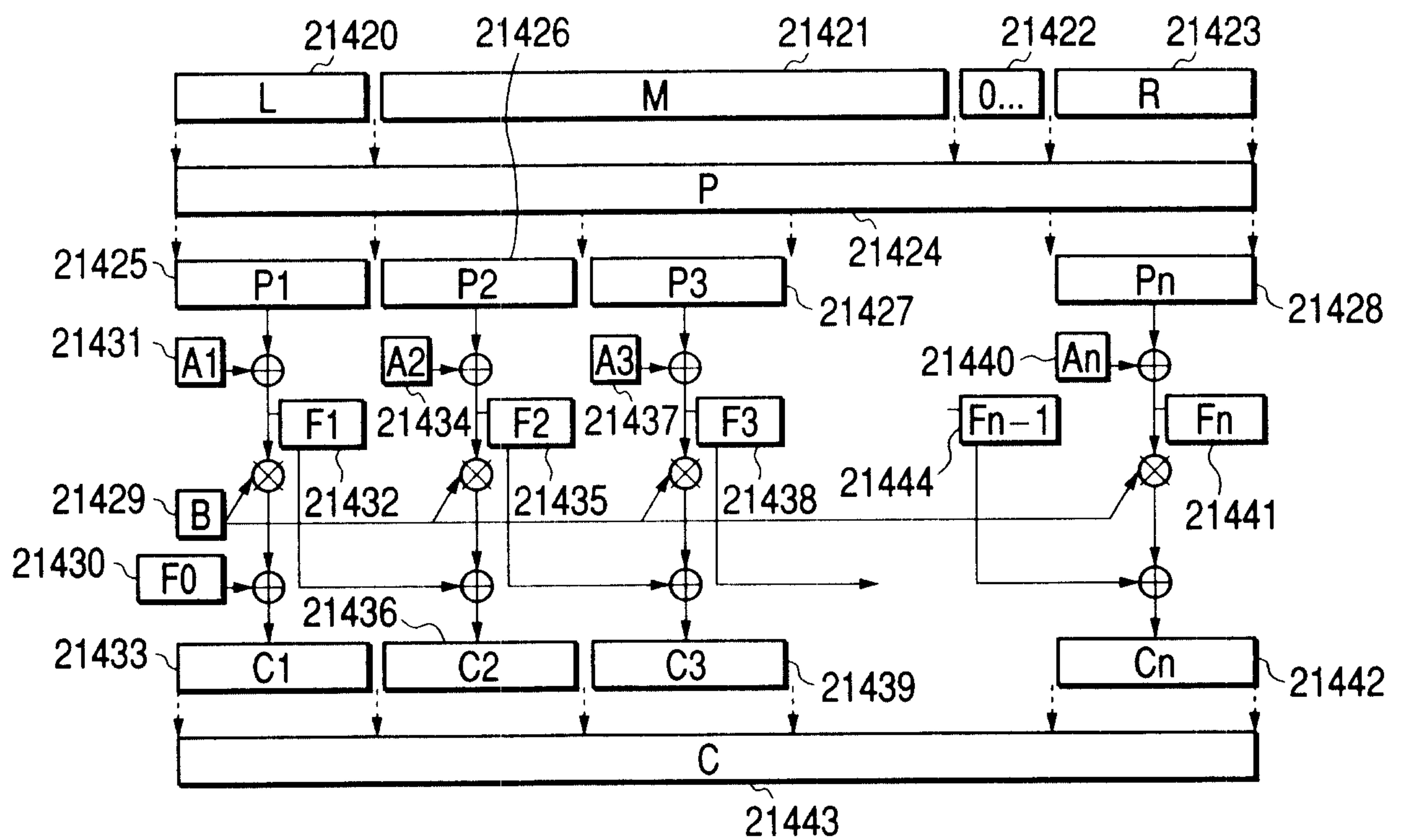
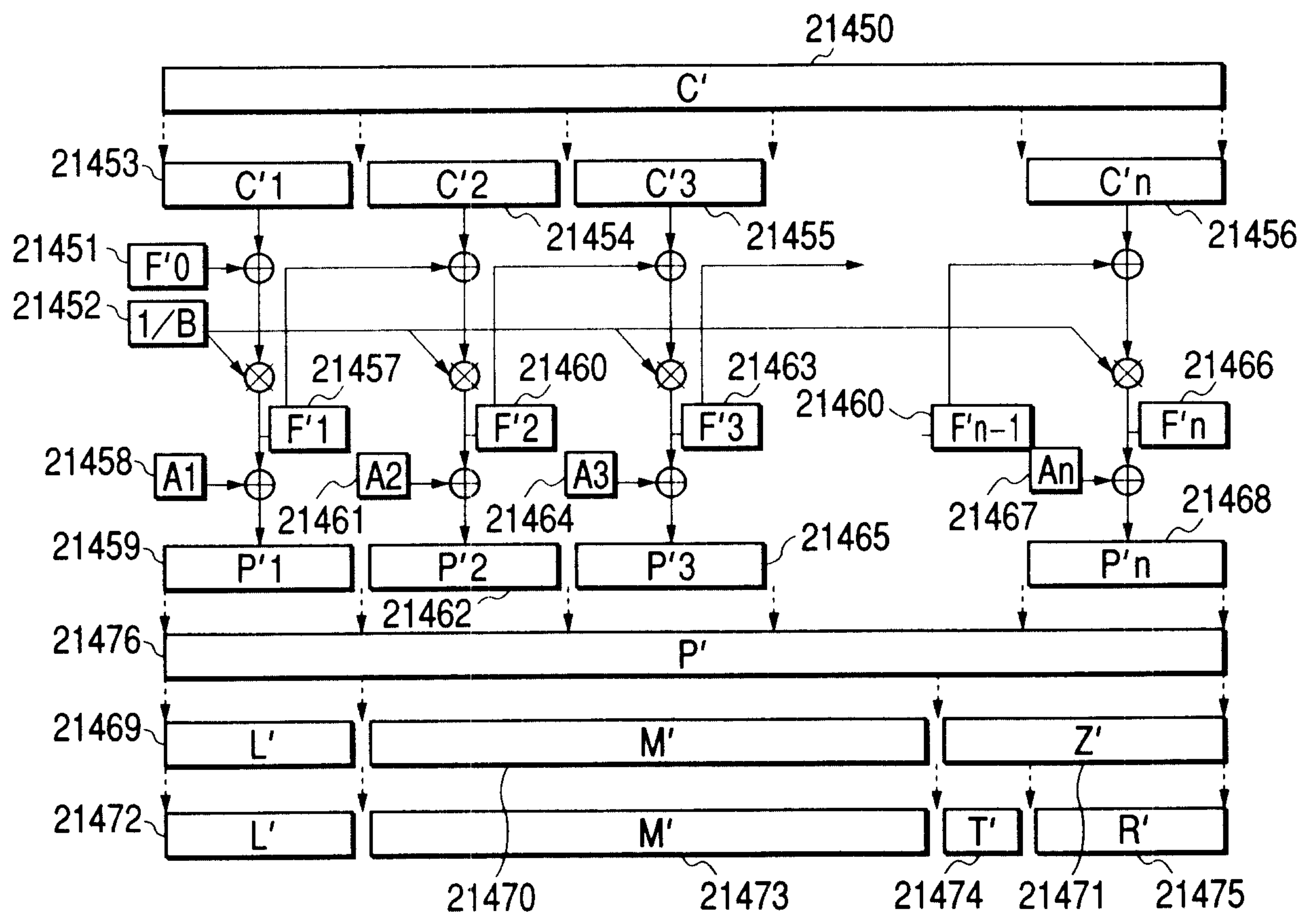
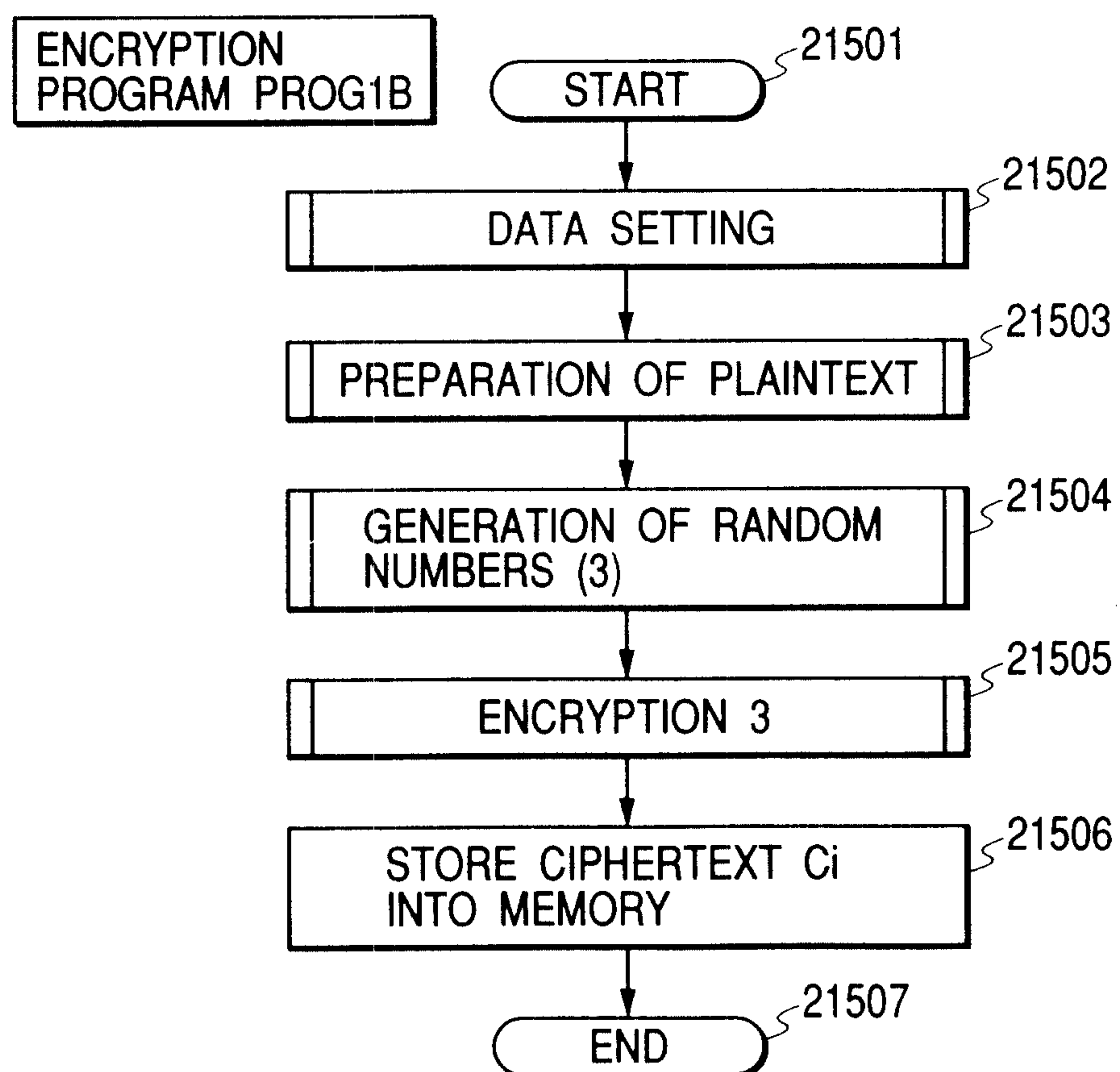
FIG. 15

FIG. 16

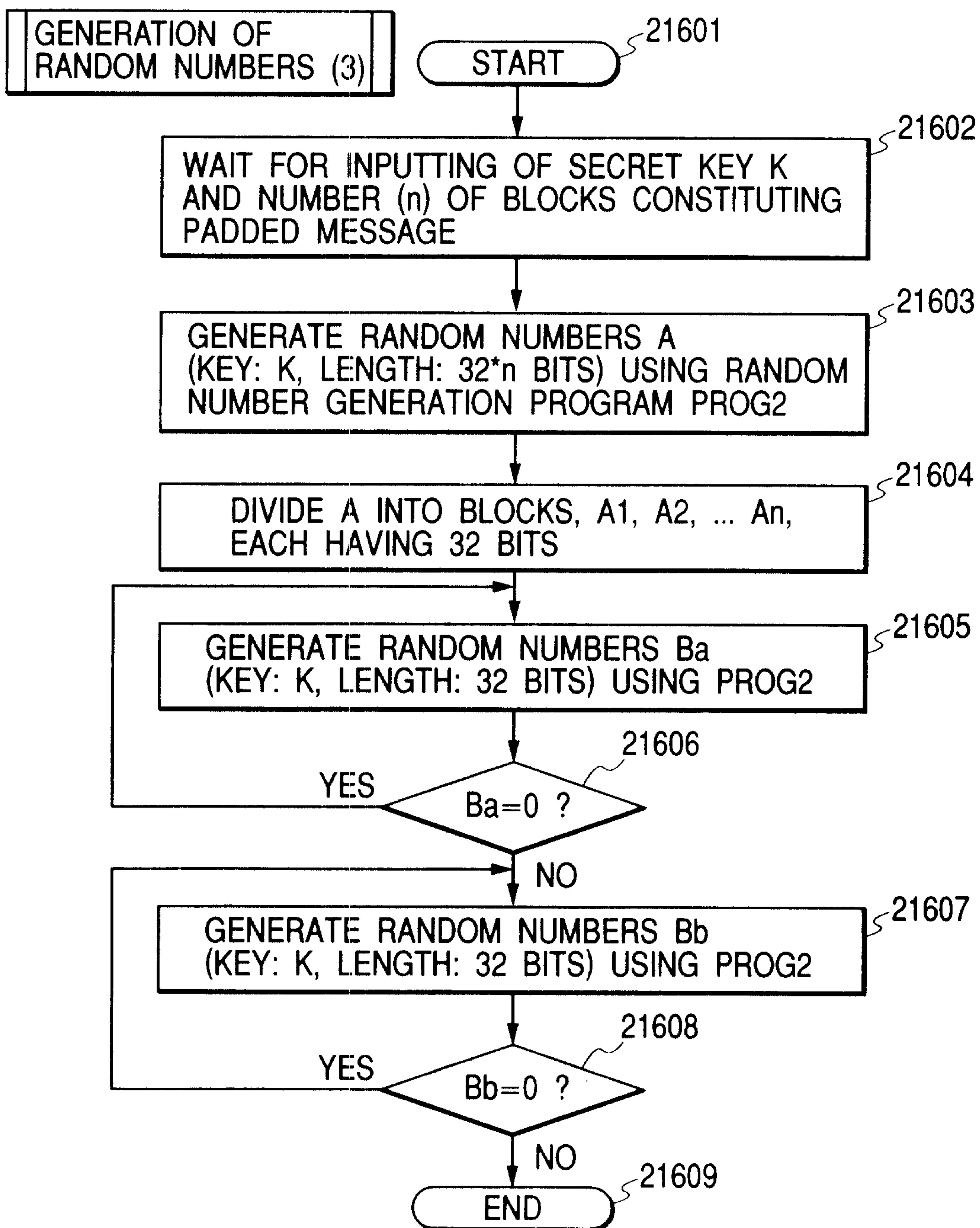
15 / 34

FIG. 17

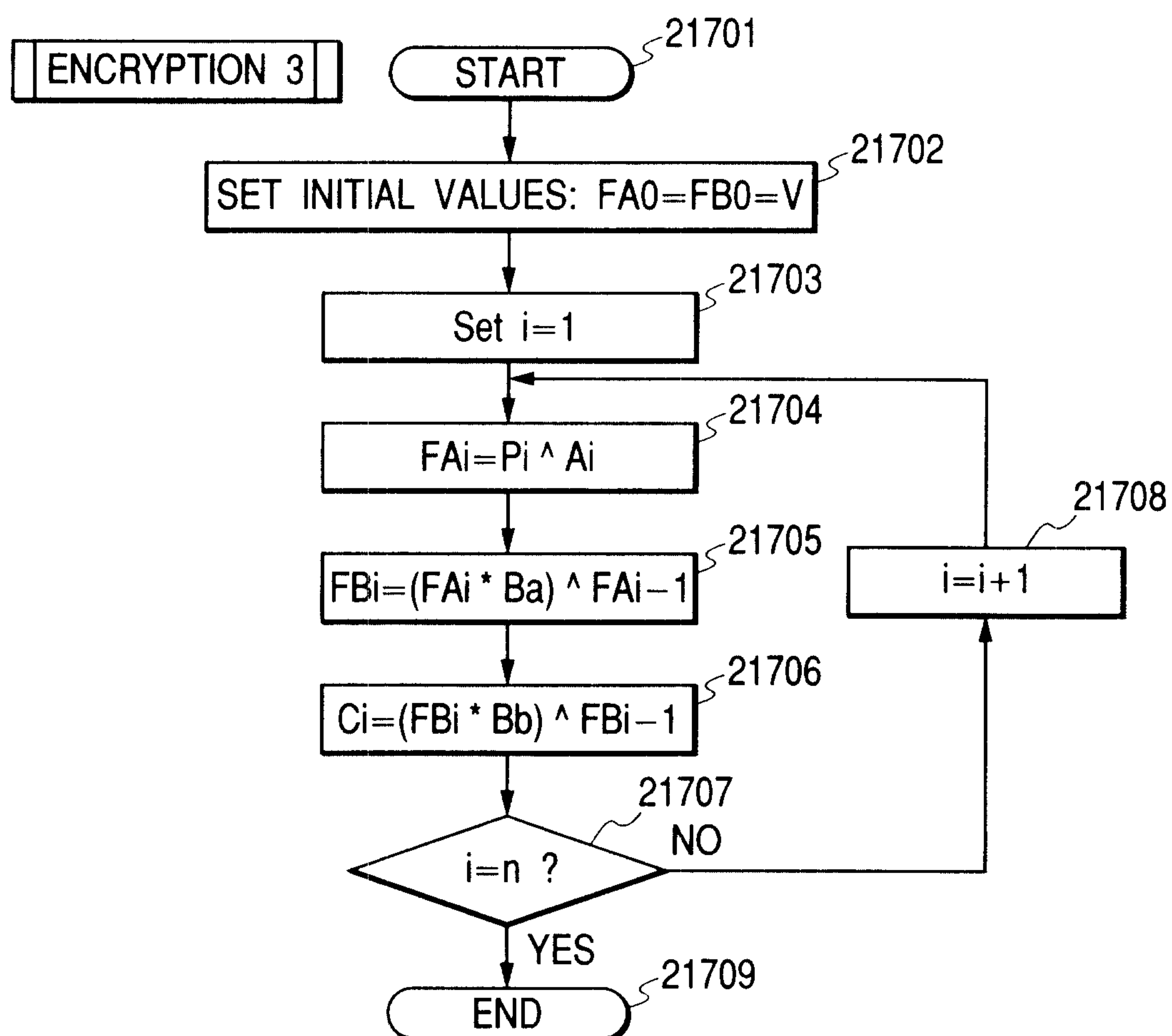
16 / 34

FIG. 18

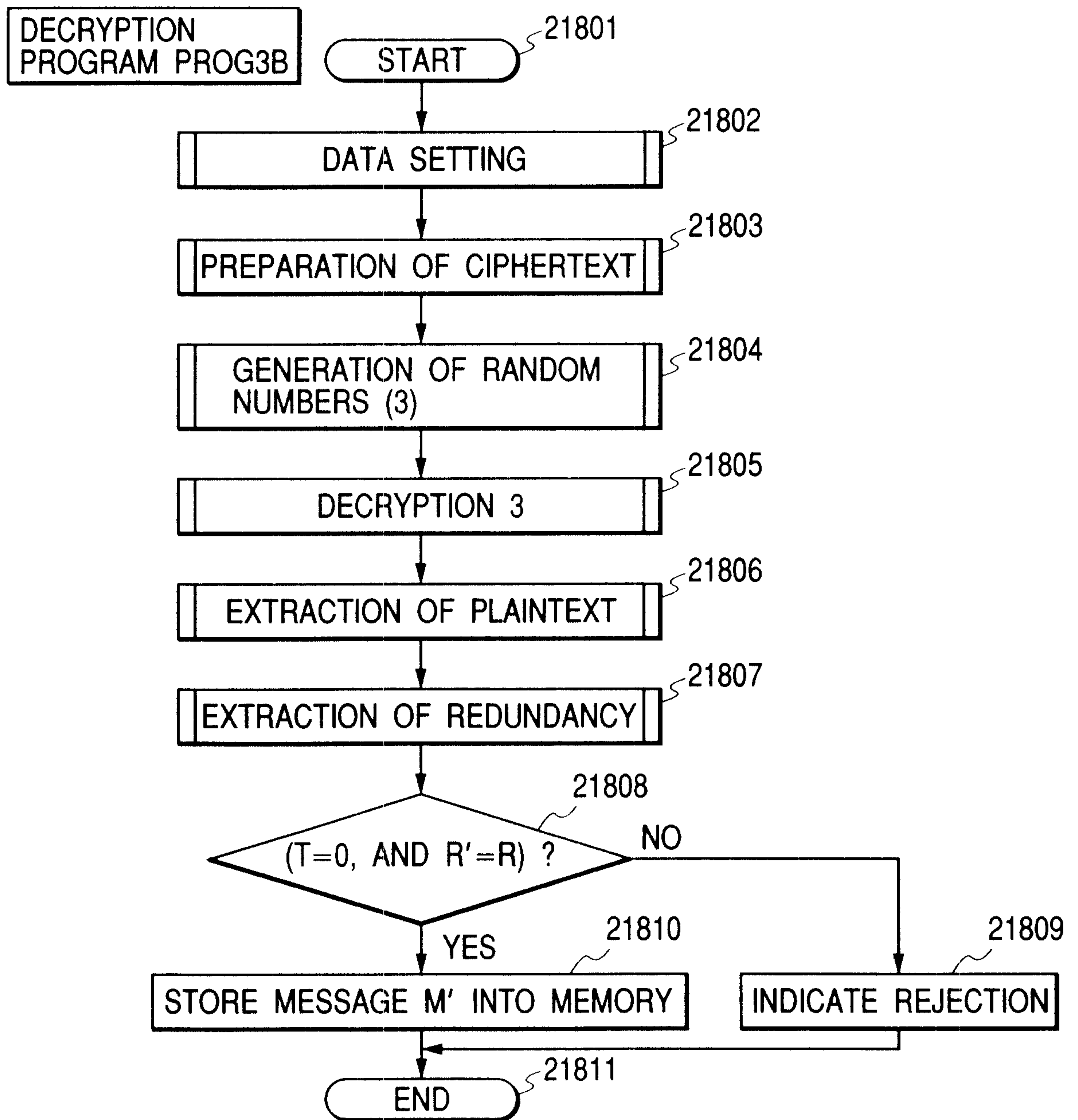
17 / 34

FIG. 19

18 / 34

FIG. 20

19 / 34

FIG. 21

20 / 34

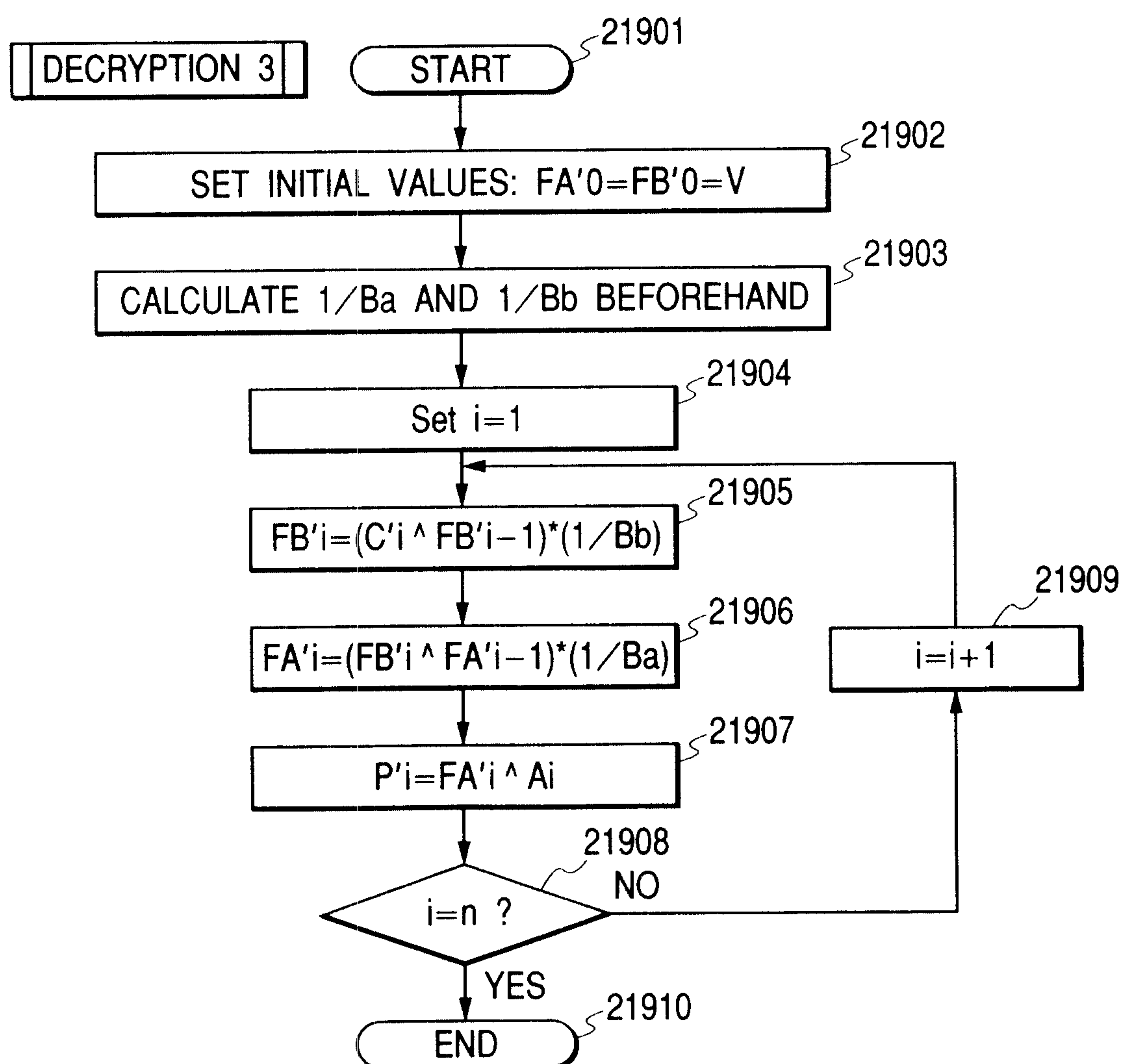
FIG. 22

FIG. 23

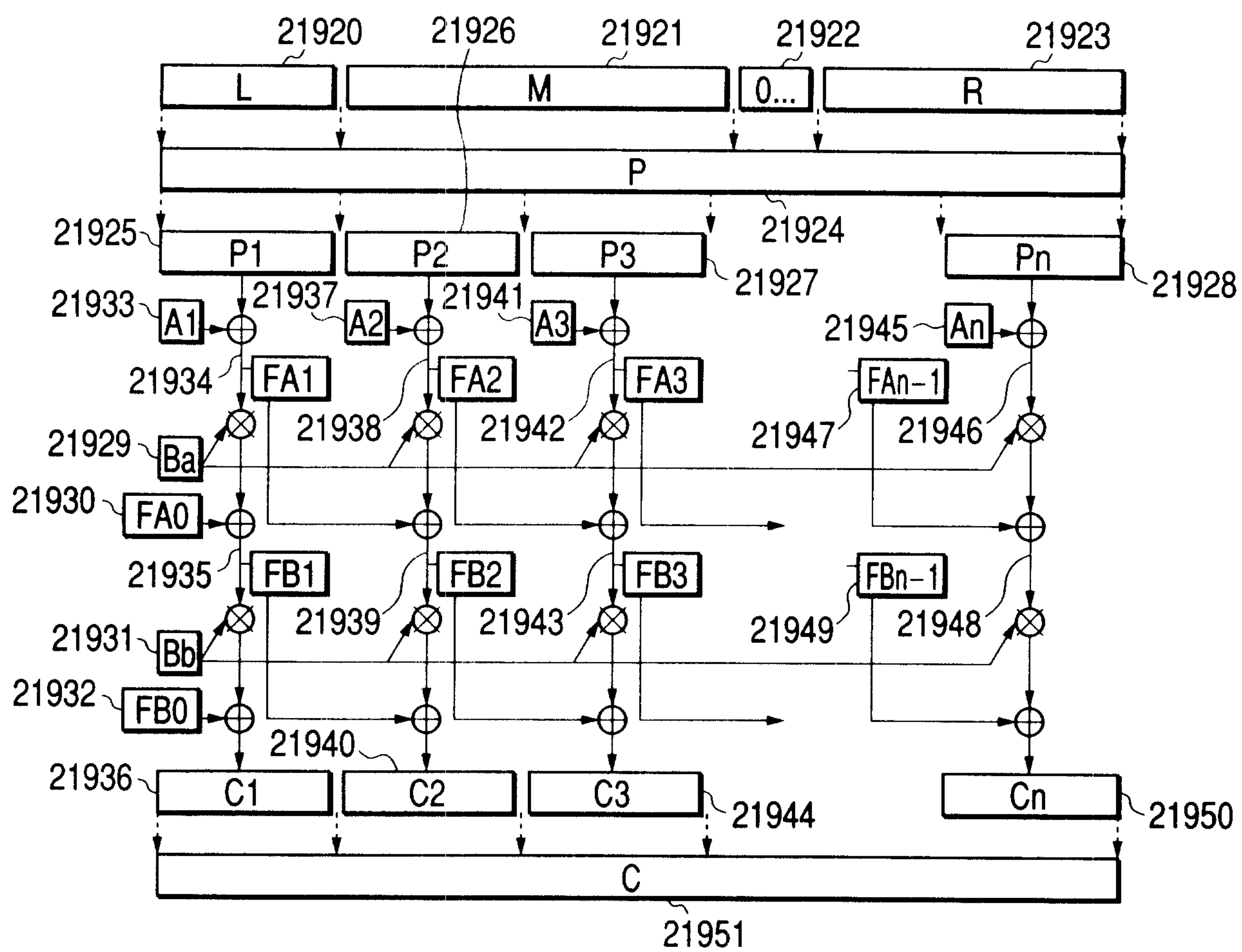
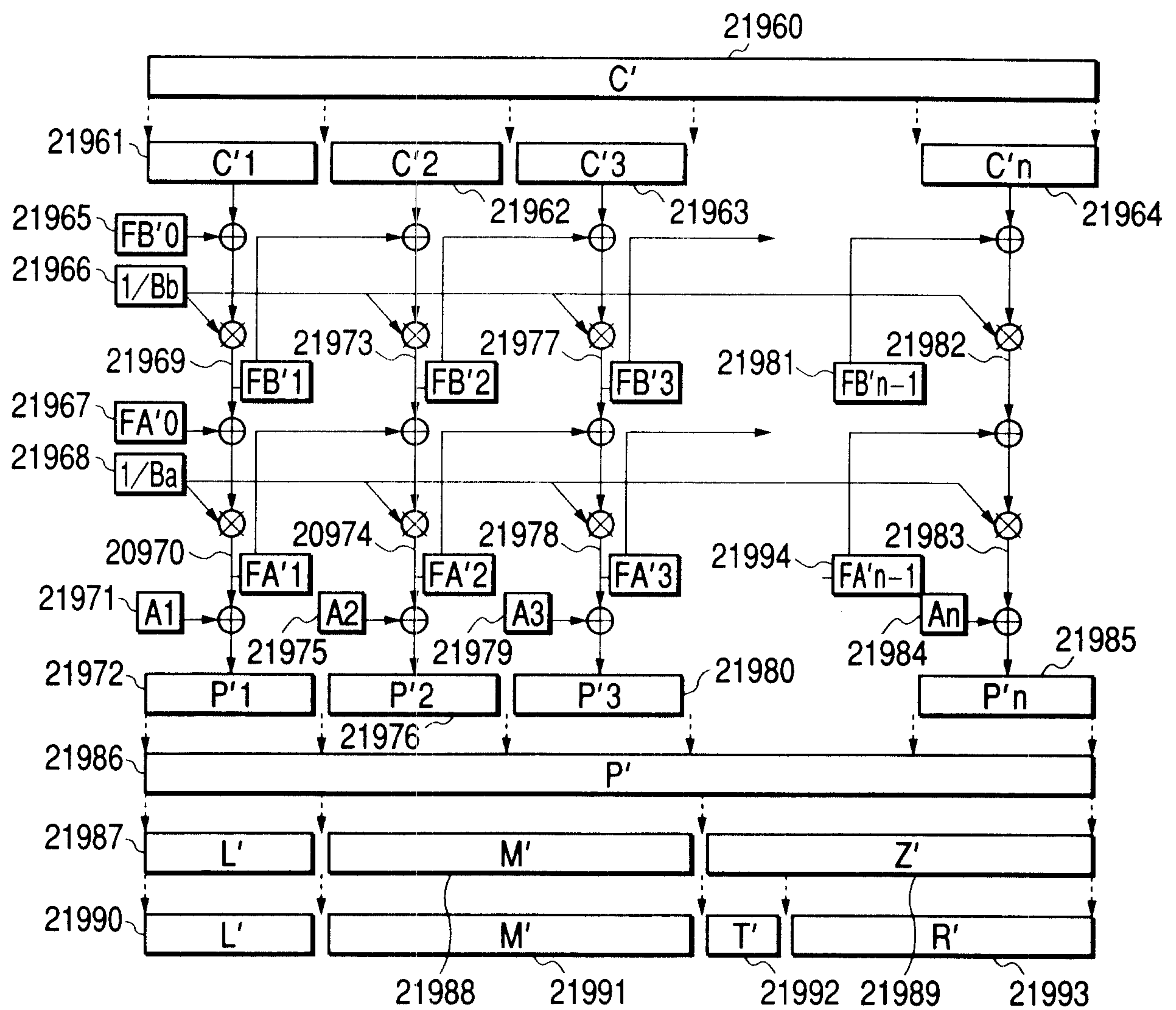
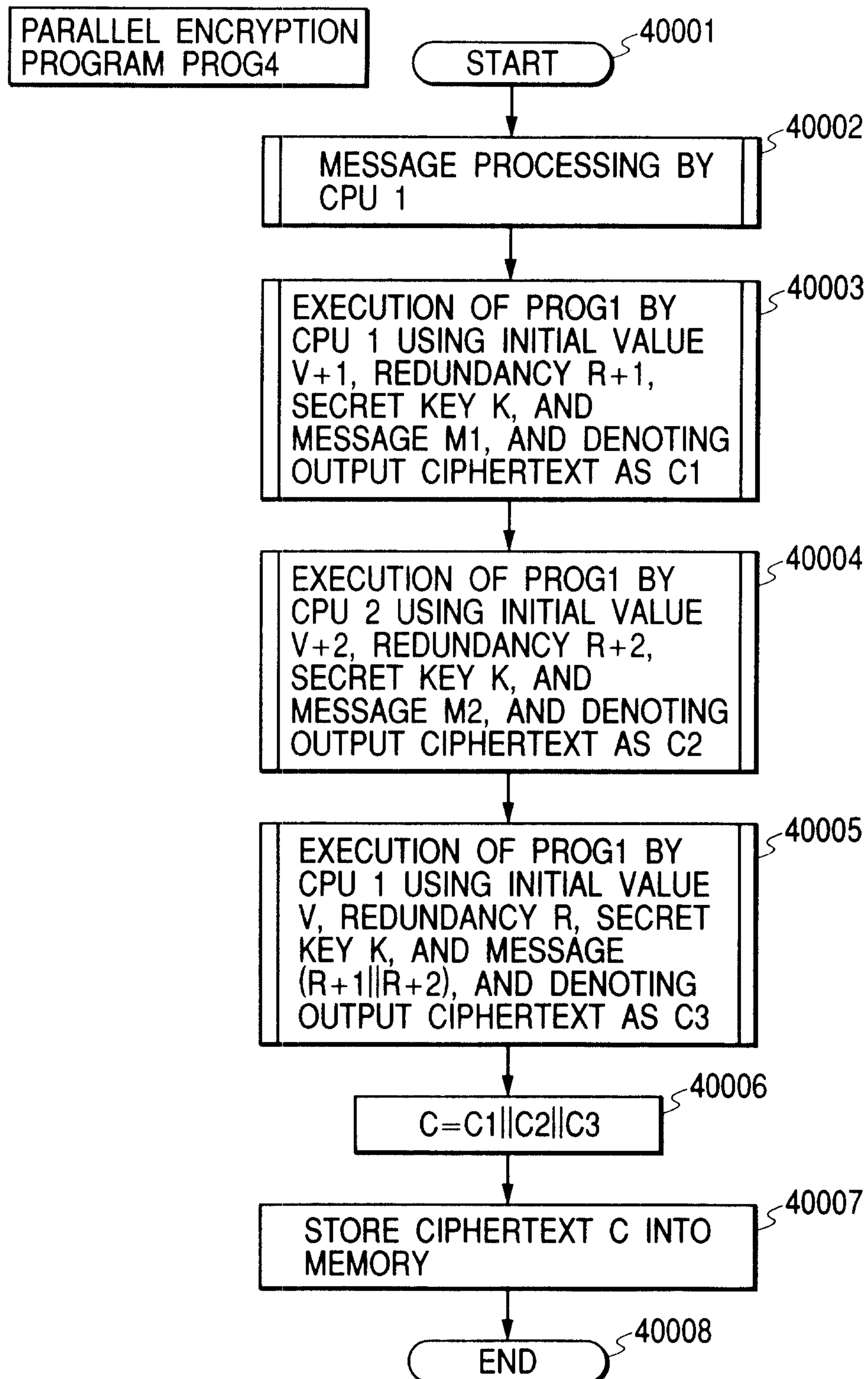


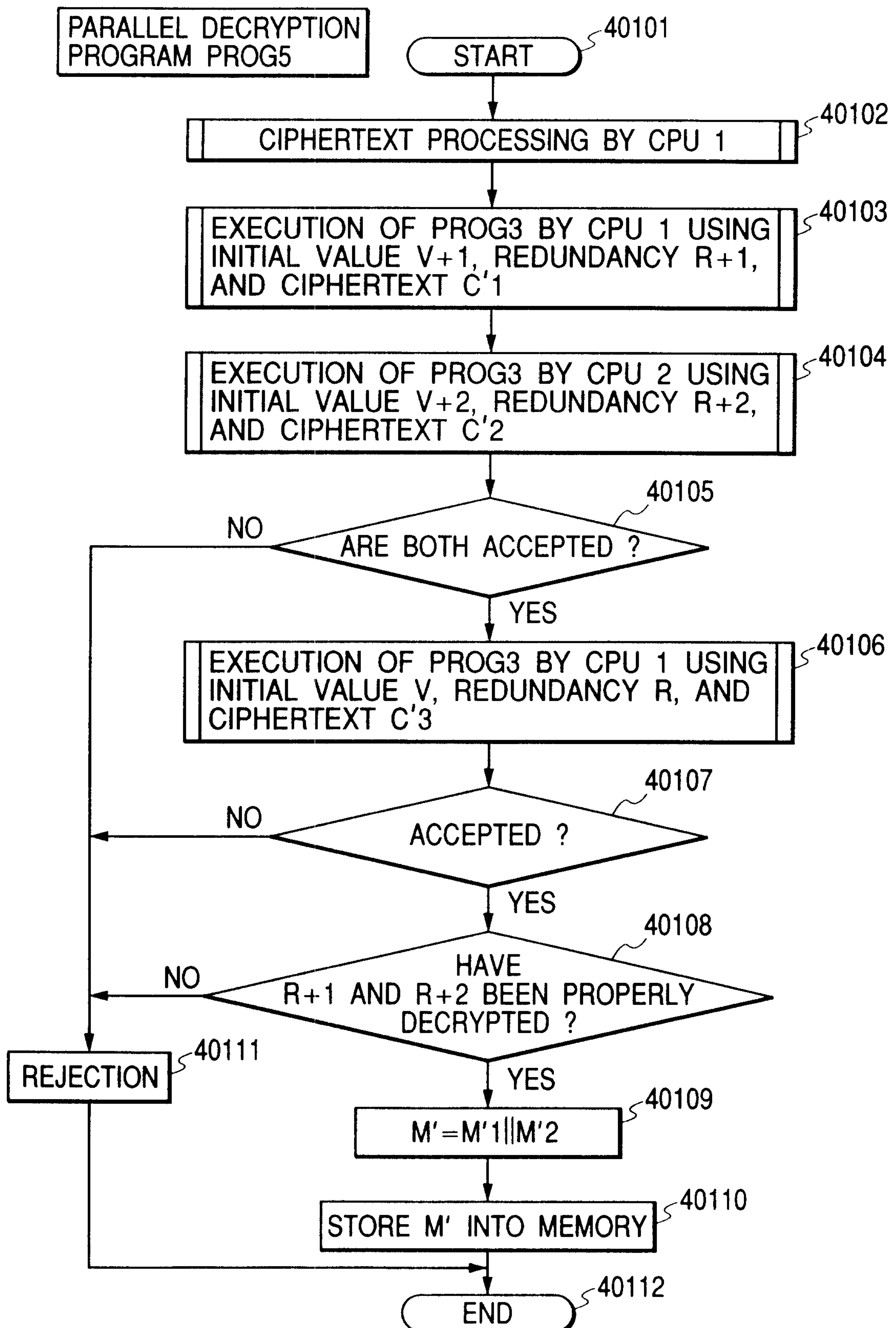
FIG. 24



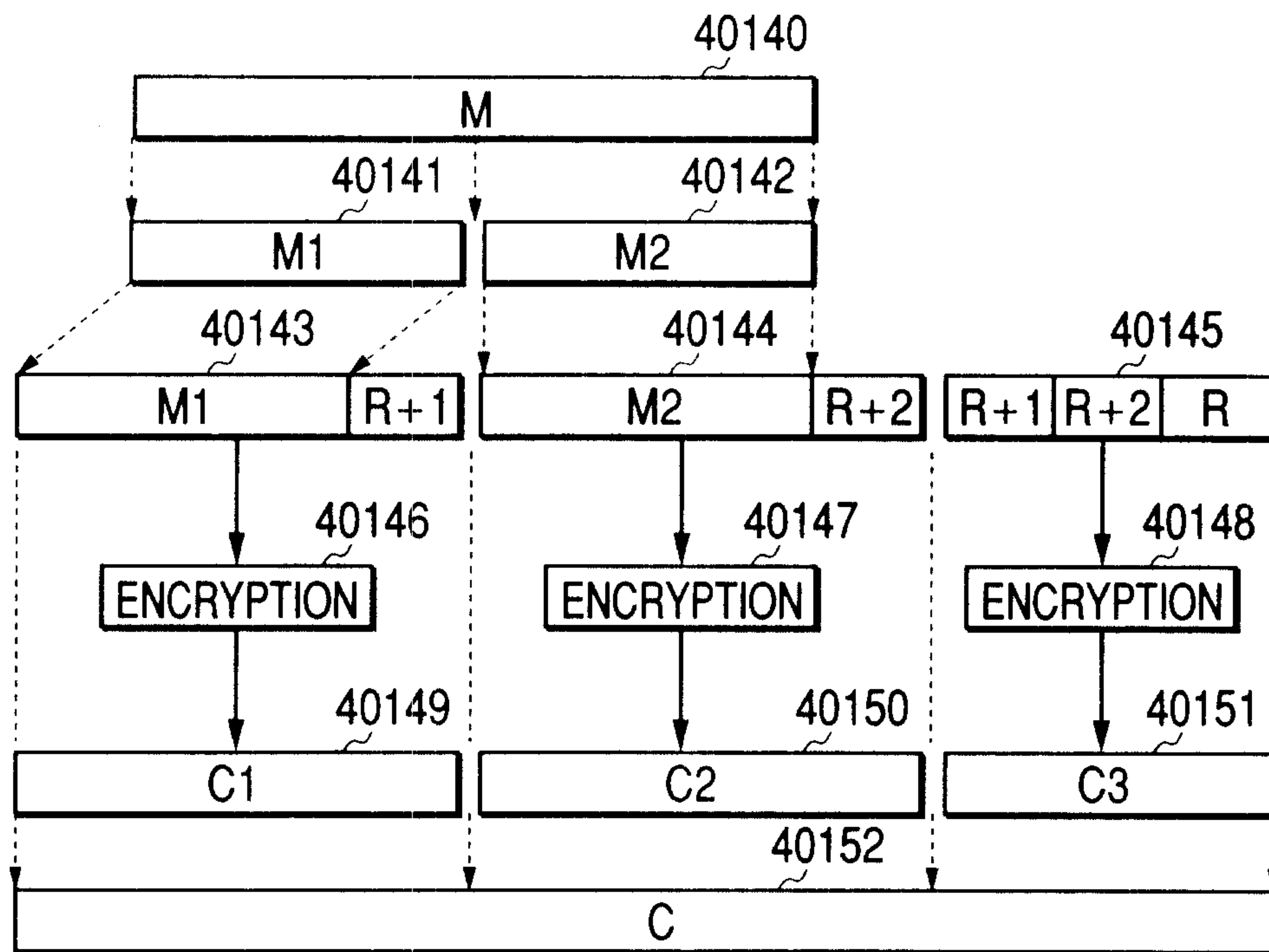
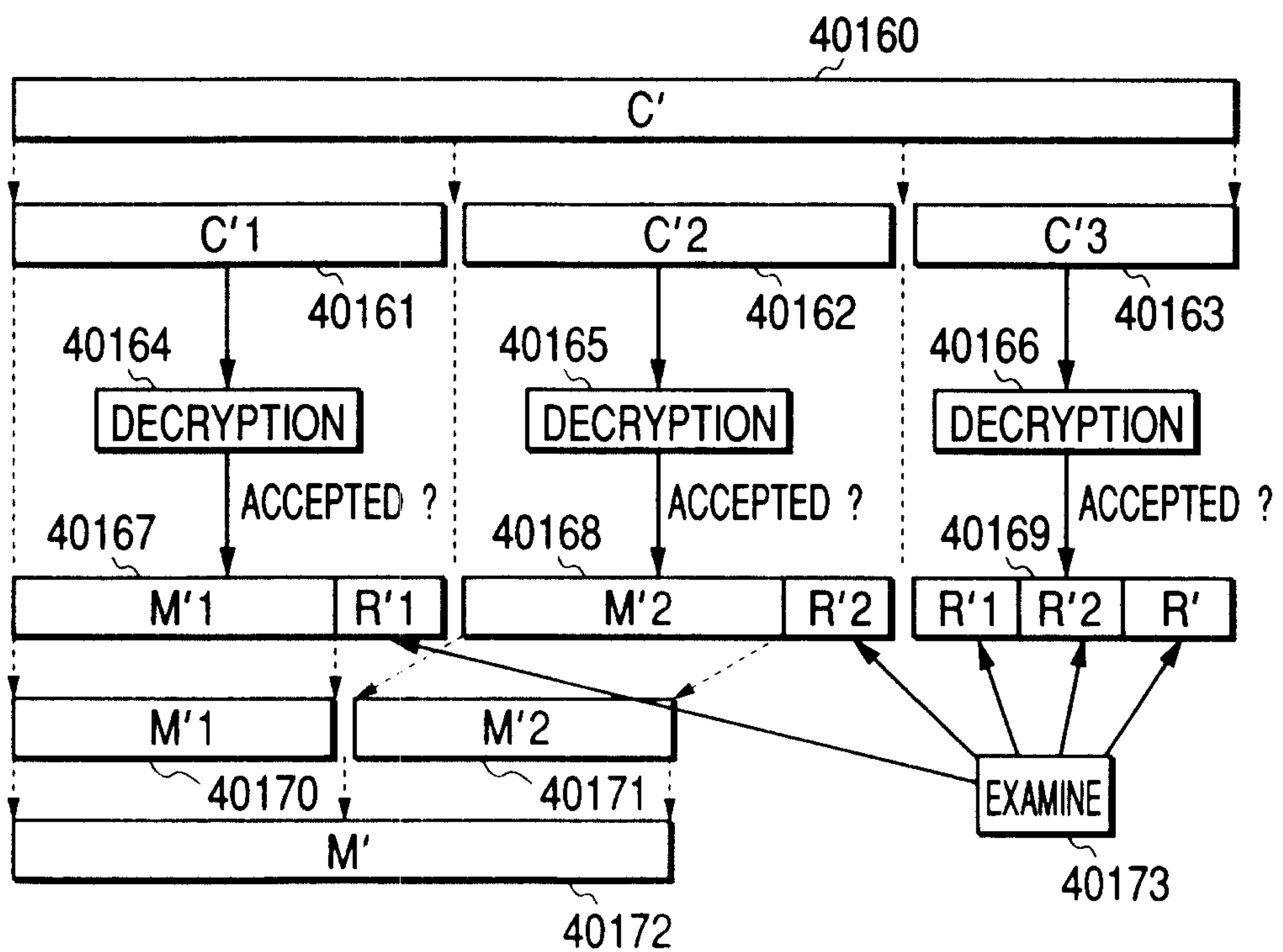
23 / 34

FIG. 25

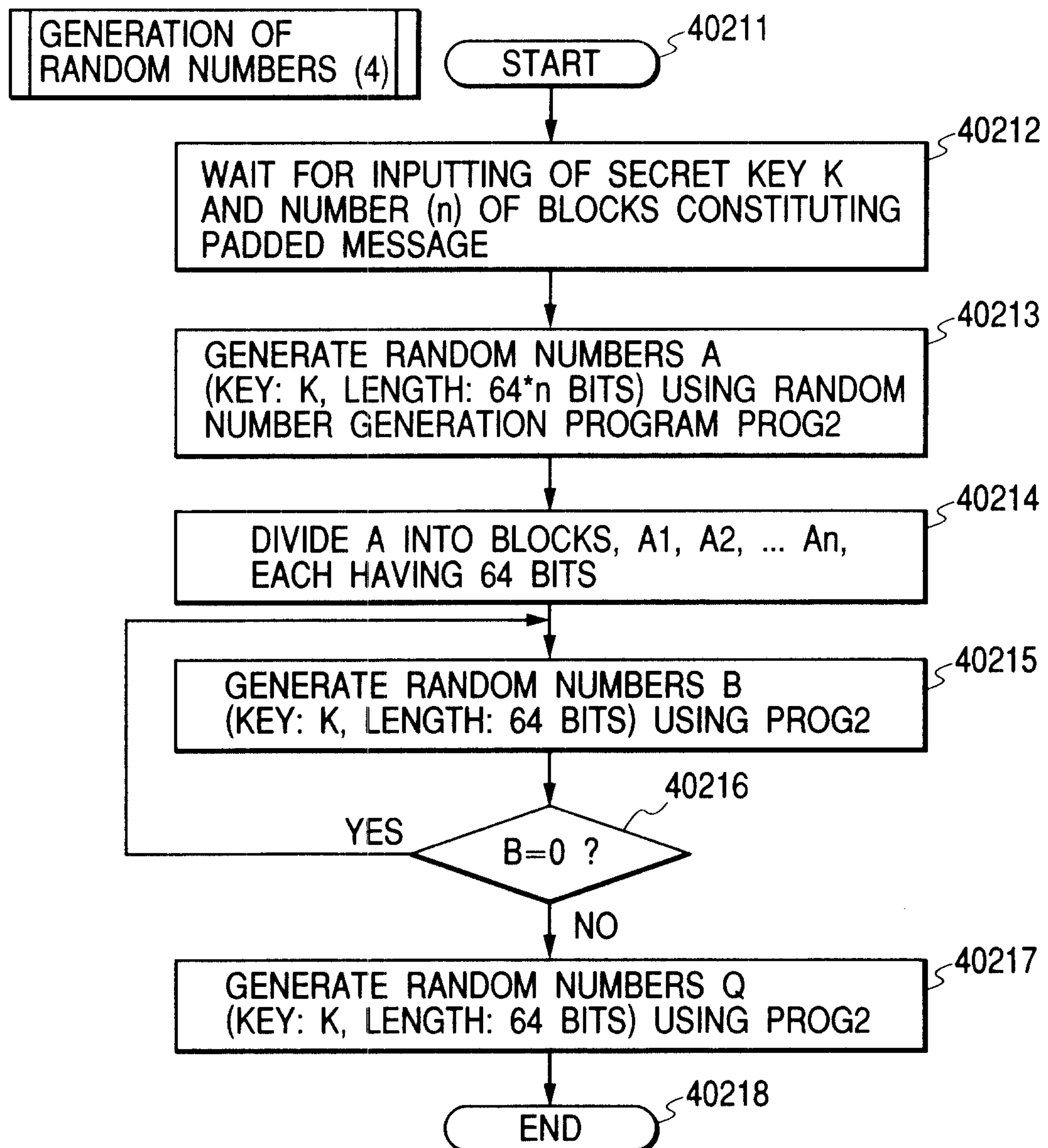
24 / 34

FIG. 26

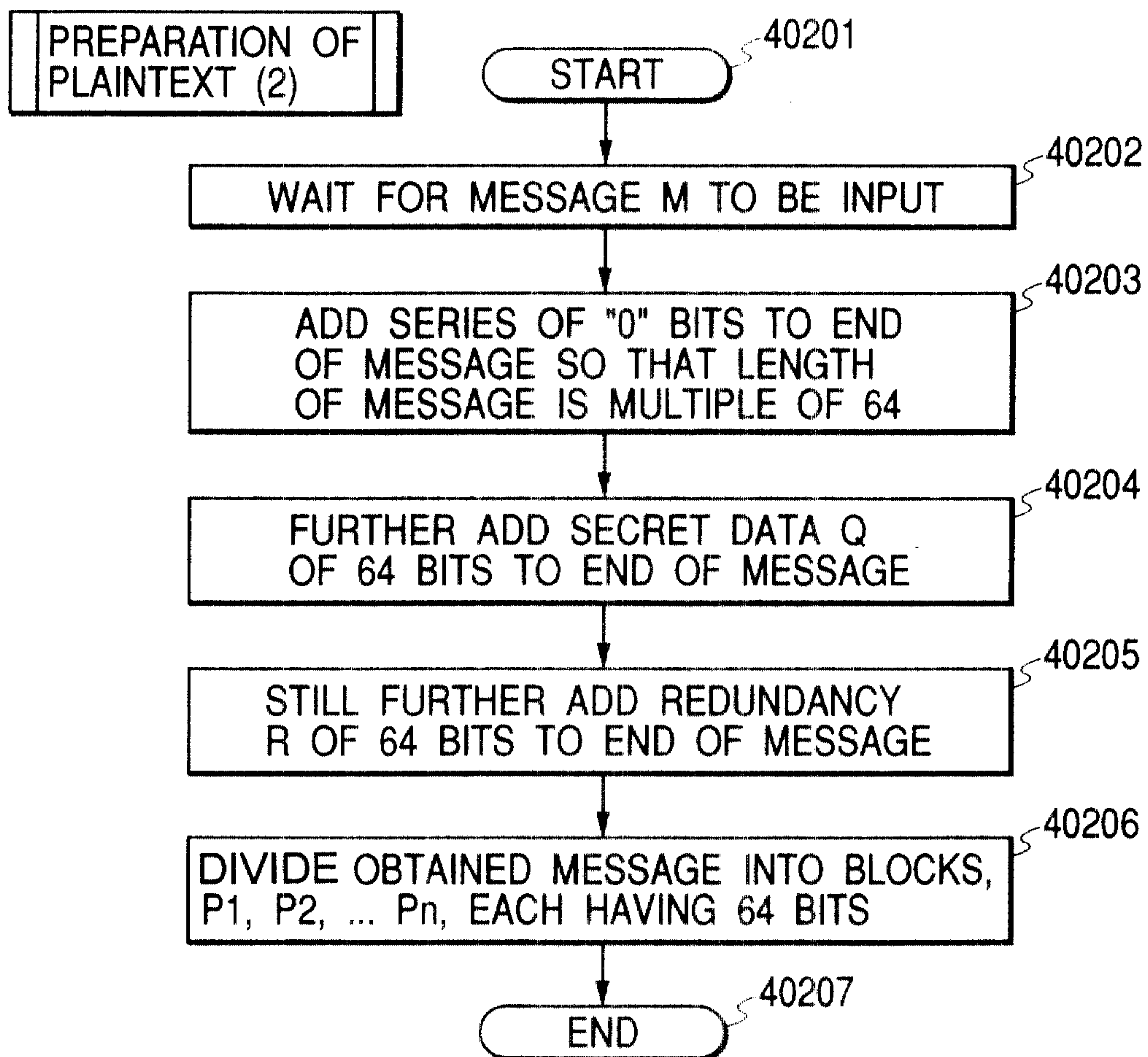
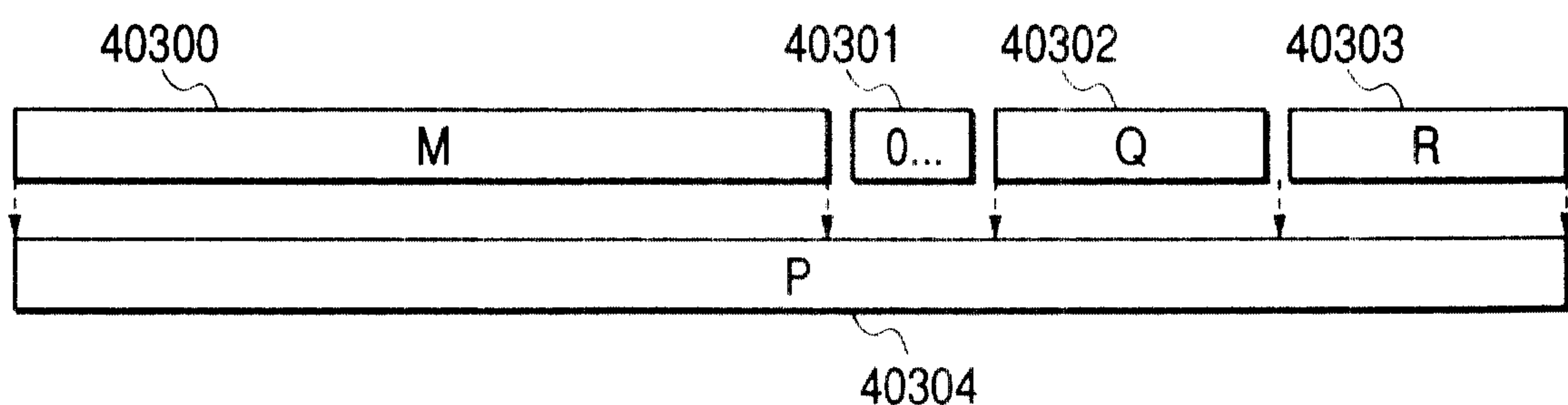
25 / 34

FIG. 27**FIG. 28**

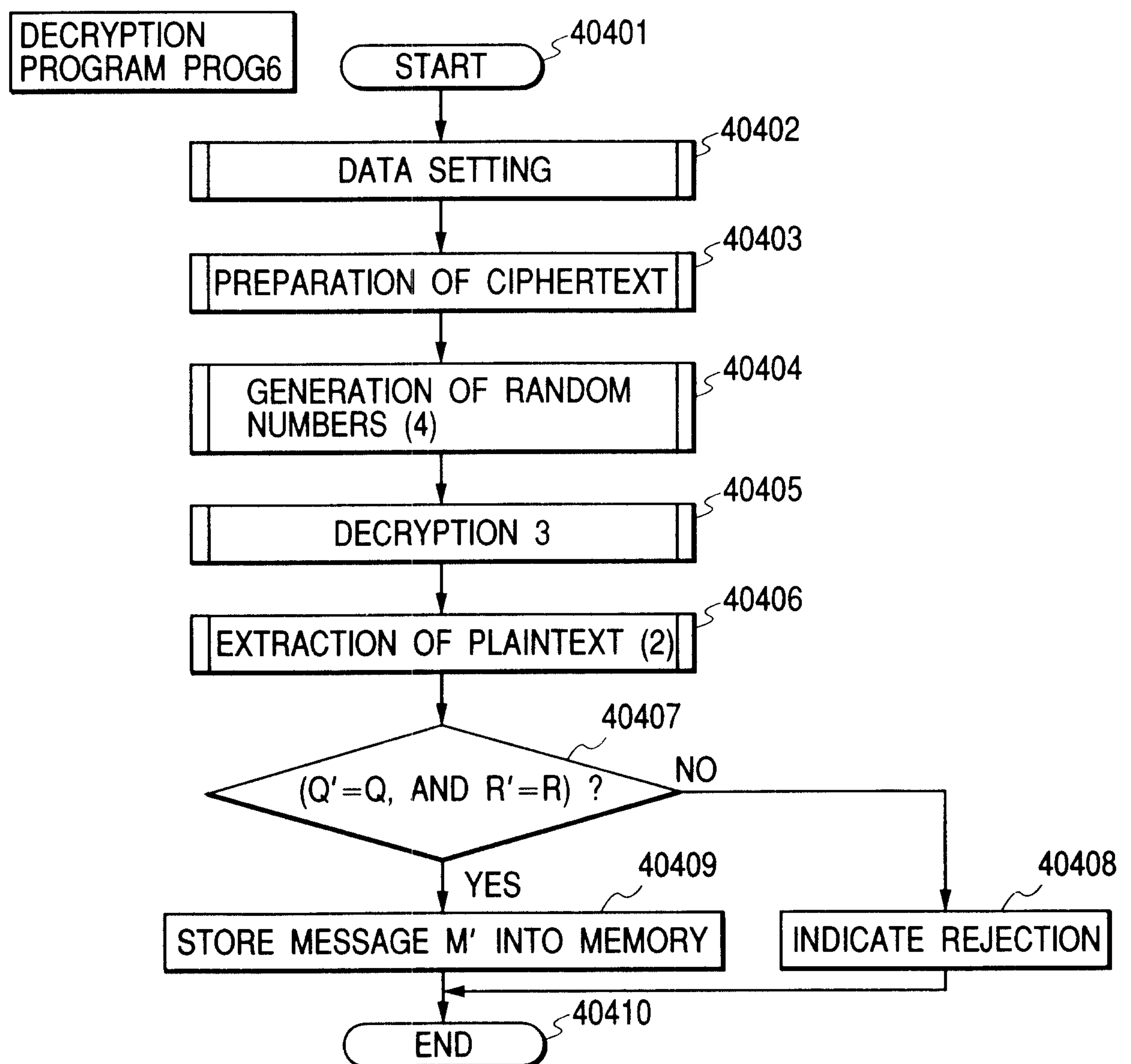
26 / 34

FIG. 29

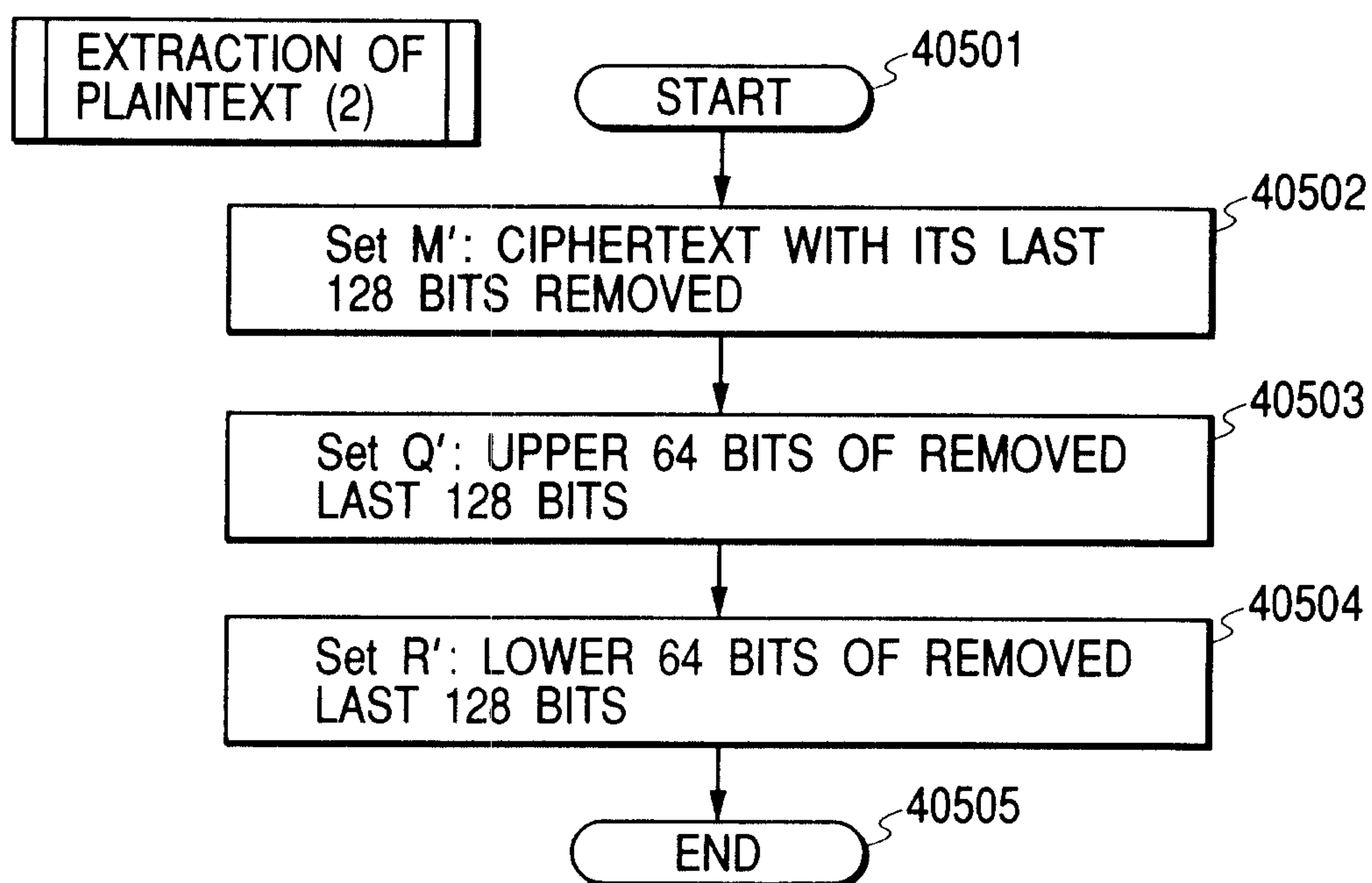
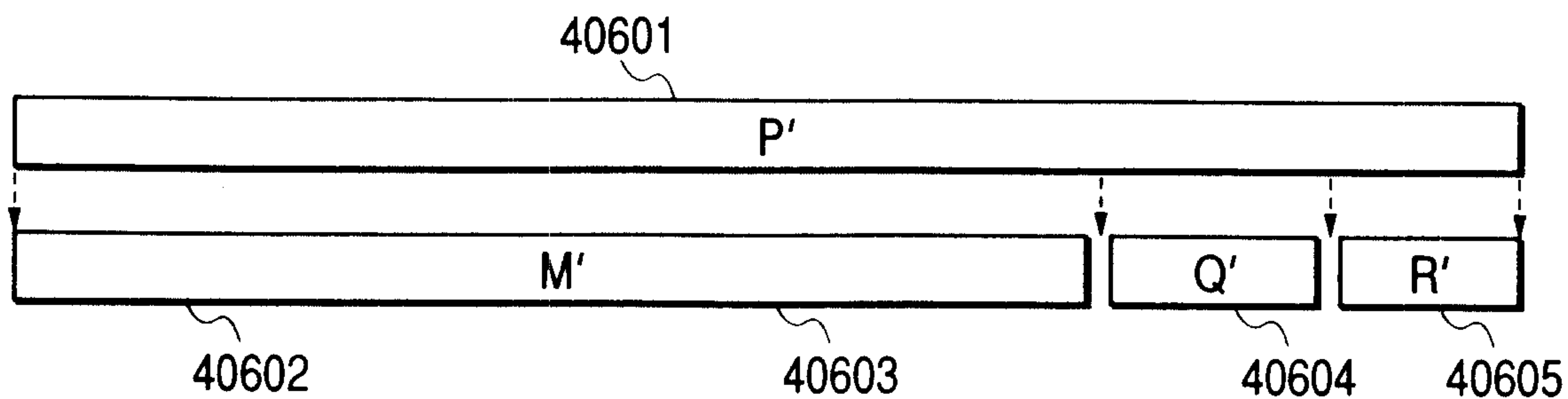
27 / 34

FIG. 30**FIG. 31**

28 / 34

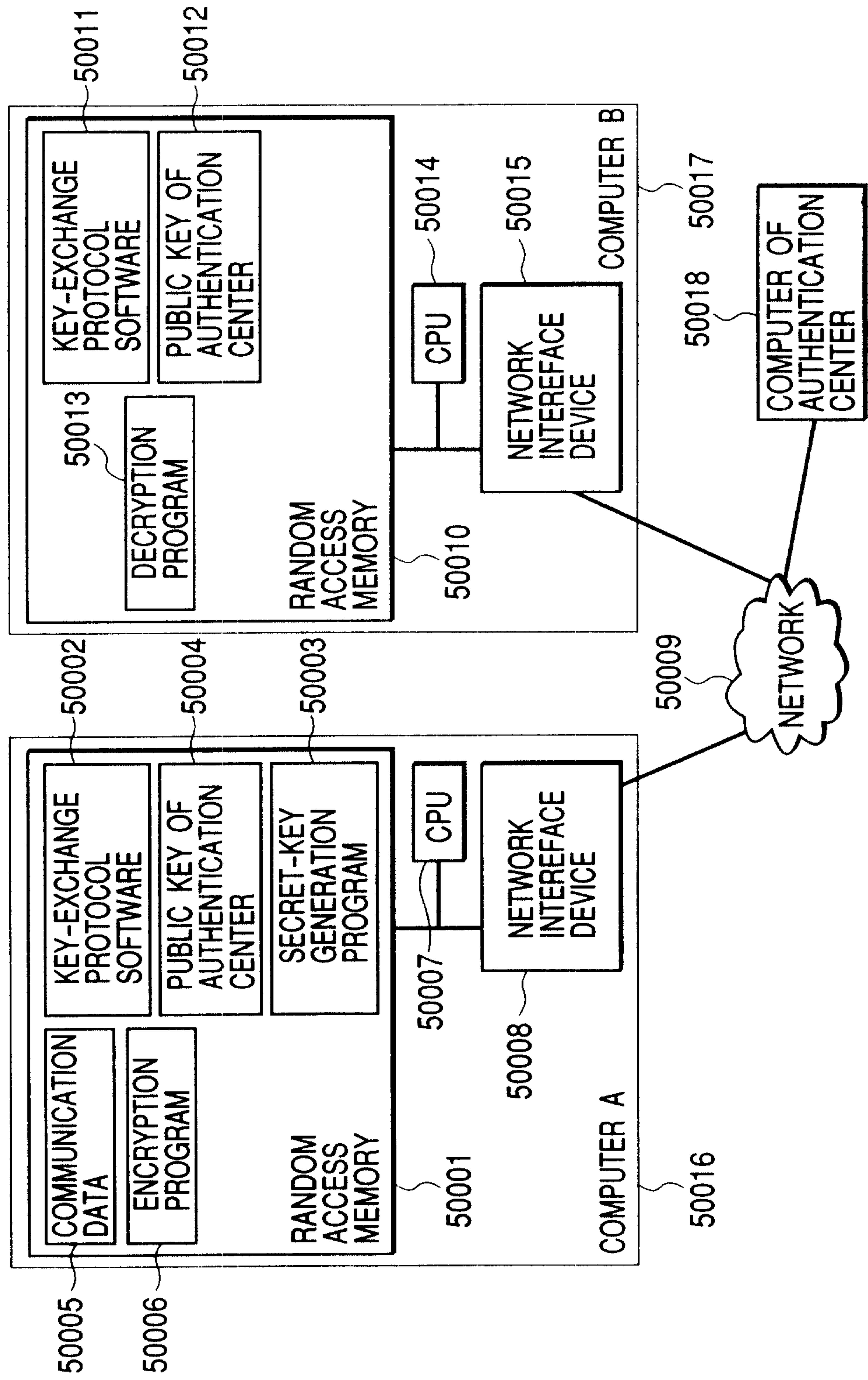
FIG. 32

29 / 34

FIG. 33**FIG. 34**

30 / 34

FIG. 35



31 / 34

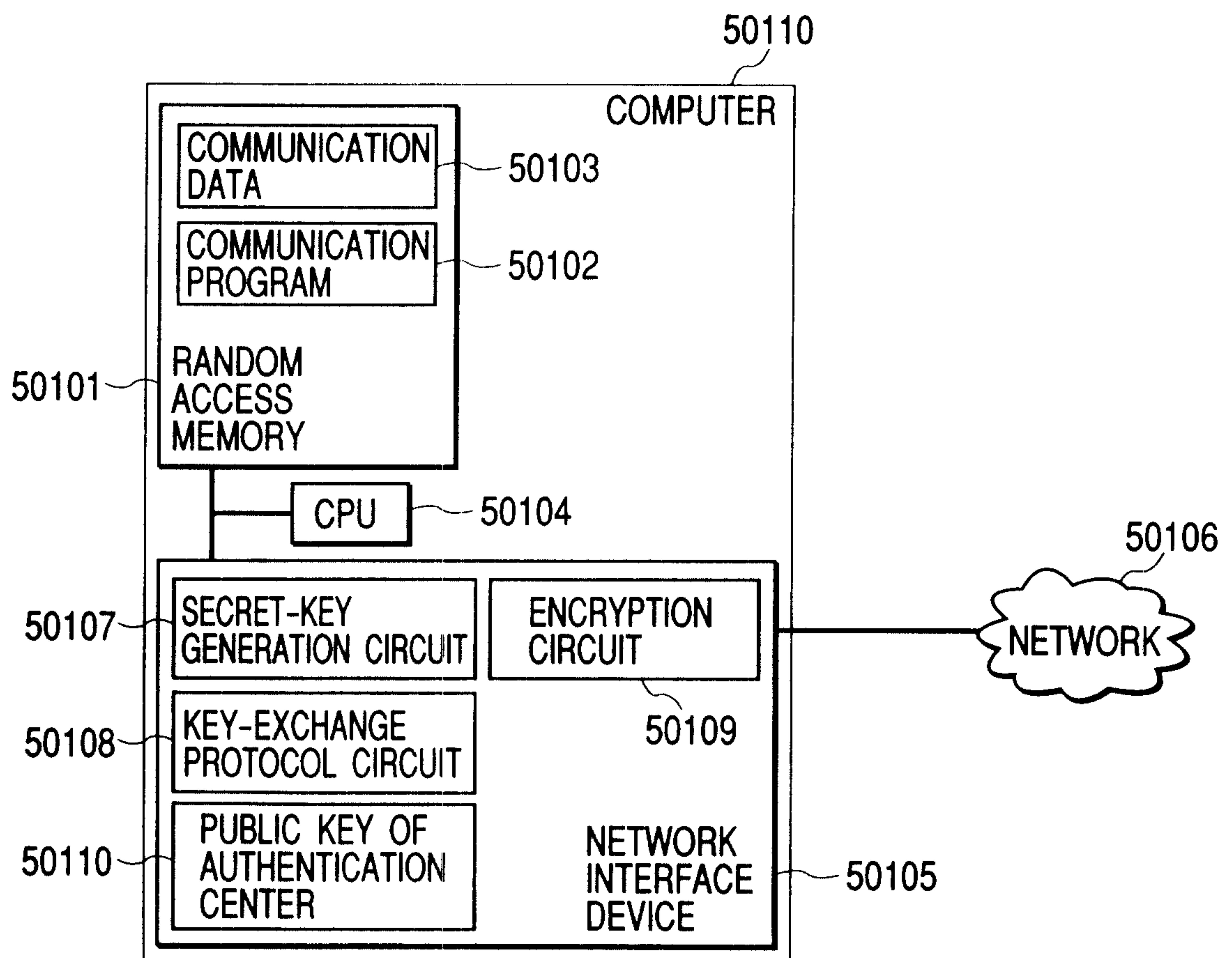
FIG. 36

FIG. 37

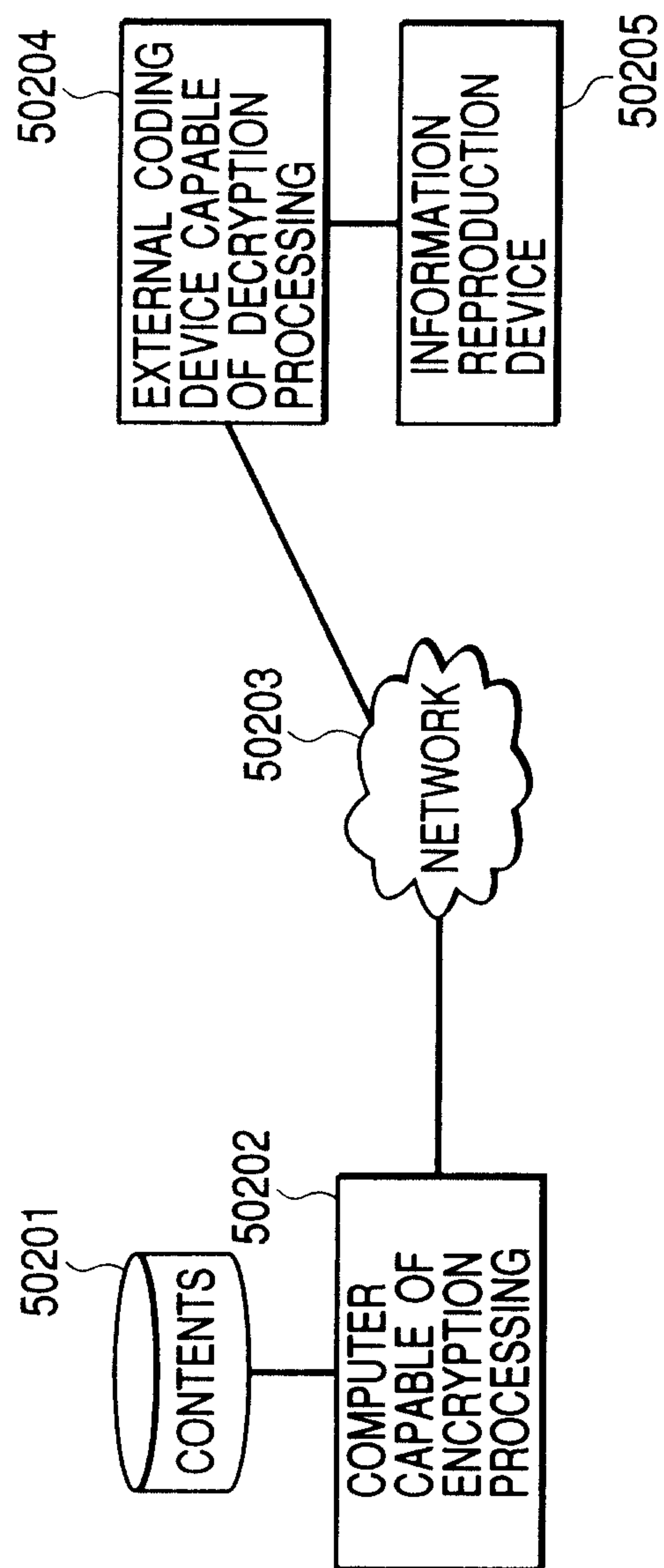
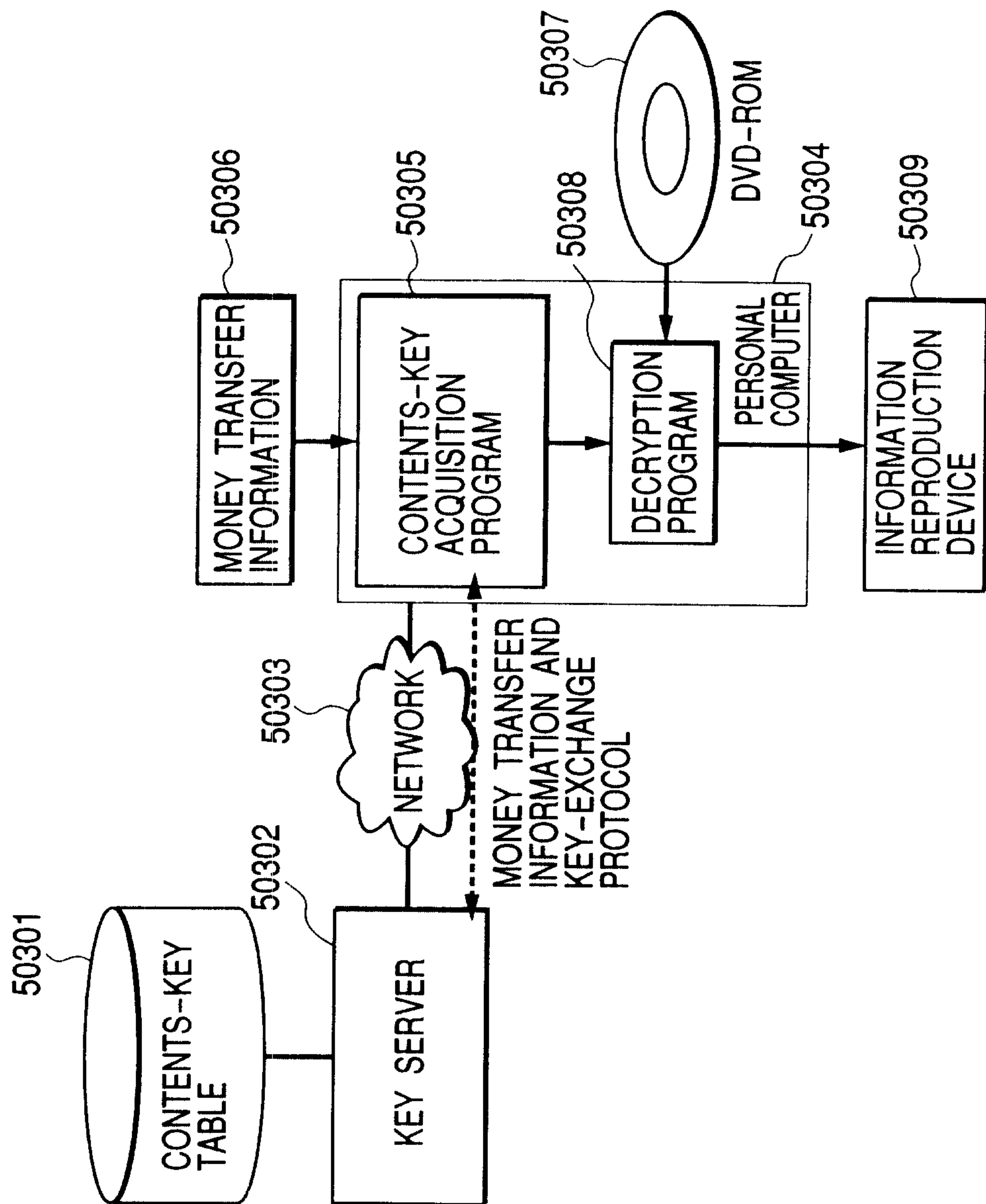


FIG. 38



34 / 34

FIG. 39

