(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 October 2007 (25.10.2007)

PCT

(10) International Publication Number
WO 2007/121035 A2

(51) International Patent Classification:
H04L 9/28 (2006.01) *H04K 1/04* (2006.01)
H04K 1/06 (2006.01) *H04K 1/00* (2006.01)
H04L 9/00 (2006.01)

(21) International Application Number:
PCT/US2007/064712

(22) International Filing Date: 22 March 2007 (22.03.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/785,821 23 March 2006 (23.03.2006) US

(71) Applicant *(for all designated States except US)*: **EXEGY INCORPORATED** [US/US]; A corporation of the State of Delaware, Suite 300, 3668 South Geyer Road, St. Louis, MO 63127 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **TAYLOR, David E.** [US/US]; 3448 Missouri Avenue, St. Louis, MO 63118 (US). **INDECK, Ronald S.** [US/US]; 729 Gralee Drive, St. Louis, MO 63132 (US). **WHITE, Jason R.** [US/US]; Apt. 1, 902 Dogwood Creek Drive, Manchester, MO 63021 (US). **CHAMBERLAIN, Roger D.** [US/US]; 64 Notre Dame Drive, St. Louis, MO 63141 (US).

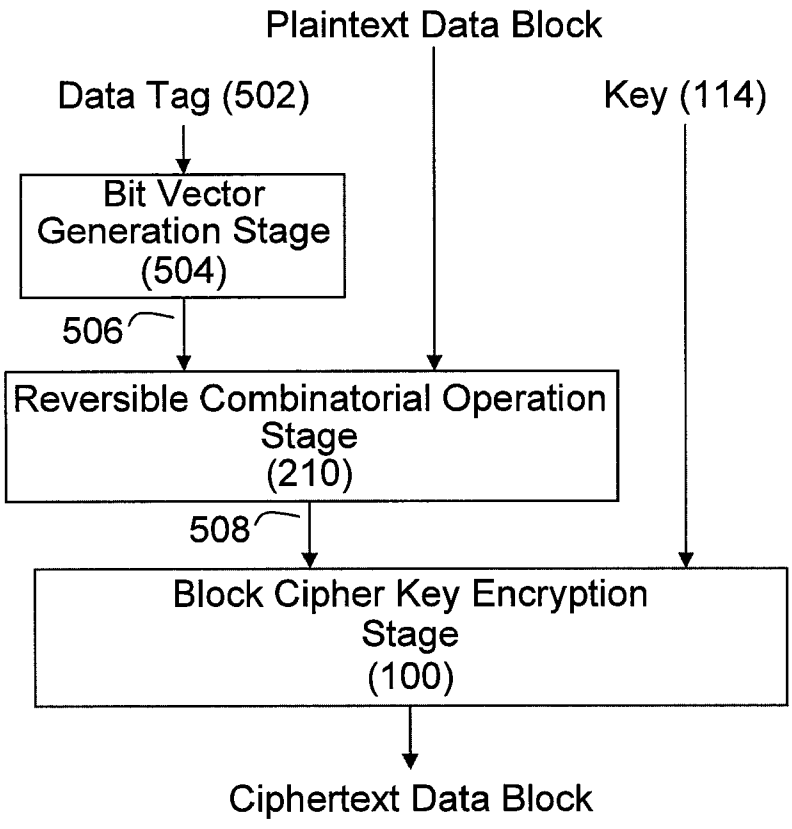
(74) Agents: **VOLK, JR., Benjamin L.** et al.; Thompson Coburn LLP, One US Bank Plaza, St. Louis, MO 63101 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR HIGH THROUGHPUT BLOCKWISE INDEPENDENT ENCRYPTION/DECRYPTION



(57) Abstract: An encryption technique is disclosed for encrypting a data segment comprising a plurality of data blocks, wherein the security and throughput of the encryption is enhanced by using blockwise independent bit vectors for reversible combination with the data blocks prior to key encryption. Preferably, the blockwise independent bit vectors are derived from a data tag associated with the data segment. Several embodiments are disclosed for generating these blockwise independent bit vectors. In a preferred embodiment, the data tag comprises a logical block address (LBA) for the data segment. Also disclosed herein is a corresponding decryption technique as well as a corresponding symmetrical encryption/decryption technique.

WO 2007/121035 A2

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Method and System for High Throughput Blockwise Independent Encryption/Decryption

Cross-Reference and Priority Claim to Related Patent Application:

This application claims priority to provisional patent application 60/785,821, filed March 23, 2006, and entitled "Method and System for High Throughput Blockwise Independent Encryption/ Decryption", the entire disclosure of which is incorporated herein by reference.

Field of the Invention:

The present invention relates to improved techniques for encrypting and decrypting data.

Background and Summary of the Invention:

The need for effective and efficient data encryption/decryption is widespread throughout today's world. Whether it be data maintained by a governmental agency that pertains to national security or data maintained by a private company that pertains to the company's trade secrets and/or confidential information, the importance of effective and efficient encryption/decryption cannot be understated.

Effective encryption/decryption is needed to preserve the integrity of the subject data. Efficient encryption/decryption is needed to prevent the act of encrypting/decrypting the subject data from becoming an overwhelming burden on the party that maintains the subject data. These needs exist in connection with both "data at rest" (e.g., data stored in nonvolatile memory) and "data in flight" (e.g., data in transit from one point to another such as packet data transmitted over the Internet).

A number of data encryption/decryption techniques are known in the art. Many of these encryption techniques utilize a block cipher (see, e.g., block cipher 100 in Figure 1). A block cipher is a cryptographic mechanism that operates on fixed length blocks of

plaintext and produces fixed length blocks of ciphertext (see, e.g., blocks 108, 110 and 112 in Figure 1). Plaintext refers to data needing encryption and ciphertext refers to data that has been encrypted. A block cipher encrypts each plaintext block using a key as per well-known key-based encryption algorithms (see, e.g., key 114 in Figure 1). The key is typically (but need not be) the same size as the plaintext block. Using different keys to encrypt the same block of plaintext typically (but need not) produces different blocks of ciphertext. Block ciphers 100 can operate on data blocks of varying sizes, with typical data block sizes ranging between 64 bits and 512 bits. For example, the Advanced Encryption Standard (AES) block cipher operates on blocks of 128 bits (16 bytes). Encrypting large segments of plaintext requires a mode of encryption operation that defines the flow of a sequence of plaintext data blocks through one or more block ciphers. Likewise, decrypting large segments of ciphertext requires a mode of decryption operation that defines the flow of a sequence of ciphertext data blocks through one or more block ciphers.

As an example of one such known mode of encryption/decryption, the electronic codebook (ECB) mode of encryption/decryption is commonly used due to its simplicity and high data throughput. Examples of the ECB mode of encryption/decryption are shown in Figure 1. With the ECB mode, a data segment needing encryption is divided into a plurality of data blocks, each data block comprising a plurality of data bits (see data blocks 102, 104 and 106 in Figure 1). Each block cipher 100 then encrypts each data block independently using key 114. At time $t=t_0$, plaintext data block 102 is encrypted by the block cipher 100 using key 114 to produce ciphertext data block 108. Subsequently, at time $t=t_1$, plaintext data block 104 is encrypted by the block cipher 100 using key 114 to produce ciphertext data block 110. Then, at time $t=t_2$, plaintext data block 106 is encrypted by the block cipher 100 using key 114 to produce ciphertext data block 112. To later decrypt the ciphertext data blocks 108, 110 and 112, these steps can then be repeated to reconstruct the original plaintext data blocks 102, 104, and 106. It is worth noting that the same block cipher 100 can be used to both encrypt and decrypt data using a key.

With ECB, the lack of sequential blockwise dependency in the encryption/decryption (i.e., feedback loops where the encryption of a given plaintext block depends on the result of encryption of a previous plaintext data block) allows implementations of the ECB mode to achieve high data throughput via pipelining and parallel processing techniques. While ECB exhibits these favorable performance characteristics, the security of ECB's encryption is susceptible to penetration because of the propagation of inter-segment and intra-segment uniformity in the plaintext to the ciphertext blocks.

For example, a 256 bit segment of plaintext containing all zeros that is to be encrypted with a 64 bit block cipher using ECB will be broken down into 4 64-bit blocks of plaintext, each 64-bit plaintext block containing all zeros. When operating on these plaintext blocks, ECB will produce a segment of ciphertext containing four identical blocks. This is an example of intra-segment uniformity. Furthermore, if another such 256-bit all zero segment is encrypted by ECB using the same key, then both of the resulting ciphertext segments will be identical. This is an example of inter-segment uniformity. In instances where intra-segment and/or inter-segment uniformity is propagated through to ciphertext, the security of the ciphertext can be compromised because the ciphertext will still preserve some aspects of the plaintext's structure. This can be a particularly acute problem for applications such as image encryption.

To address intra-segment and inter-segment uniformity issues, there are two commonly-used approaches. One approach is known as cipher block chaining (CBC). An example of the CBC mode of encryption/decryption is shown in Figure 2. The CBC mode combines the most recent ciphertext output from the block cipher with the next input block of plaintext. The first plaintext block to be encrypted is combined with an initialization vector that is a bit string whose bits have random values, thereby providing the CBC mode with inter-segment randomness.

As shown in Figure 2, At time $t=t_0$, the first plaintext data block 102 is combined with a random initialization vector (IV) 200 using a reversible combinatorial operation 210, to thereby create a

block-vector combination. This block-vector combination is then encrypted by block cipher 100 using key 114 to thereby generate ciphertext block 202. Next, at time $t=t_1$, the ciphertext block 202 is fed back to be combined with the second plaintext block 104 via XOR operation 210. The resultant block-vector combination is key encrypted by block cipher 100 to produce ciphertext block 204, which is in turn fed back for combination with the next plaintext block at time $t=t_2$ to eventually produce ciphertext block 206. Thus, as can be seen, when the CBC mode is used to encrypt a data segment comprising a plurality of data blocks, the bit vectors that are used for the reversible combinatorial operations with the plaintext data blocks that follow the first plaintext data block are bit vectors that are dependent upon the encryption operation(s) performed on each previously encrypted plaintext data block.

Preferably, the reversible combinatorial operation 210 is an XOR operation performed between the bits of the vector 200 and the block 102. The truth table for an XOR operation between bits X and Y to produce output Z is as follows:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

As is well known, the XOR operation is reversible in that either of the inputs X or Y can be reconstructed by performing an XOR operation between Z and the other of the inputs X or Y. That is, if one XORs X with Y, the result will be Z. If one thereafter XORs Z with Y, then X will be reconstructed. Similarly, if one thereafter XORs Z with X, then Y will be reconstructed.

Thus, on the decryption side, the CBC mode operates to decrypt ciphertext block 202 with the cipher block 100 using key 114 to thereby reconstruct the XOR combination of plaintext data block 102 and the initialization vector 200. Thereafter, this reconstructed combination can be XORed with the initialization vector 200 to reconstruct plaintext block 102. Next, at time $t=t_1$, the process is repeated for the next ciphertext block 204, although this time the XOR operation will be performed using ciphertext block 202 (rather than initialization vector 200) to reconstruct plaintext data block

104. Ciphertext block 202 is used in this XOR operation because it was ciphertext block 202 that was used in the XOR operation when plaintext block 104 was encrypted. Then, once again this process is repeated at time $t=t_2$, albeit with ciphertext block 204 being used for the XOR combination operation with the output from cipher block 100.

While the use of feedback by the CBC mode addresses the issue of inter-segment and intra-segment uniformity, such feedback imposes a sequential processing flow on the encryption that significantly limits the achievable throughput of the encryption engine. As such, the CBC mode cannot make ready use of pipelining because one of the inputs for the reversible combinatorial operation stage 210 of the encryption for a given data block depends upon the output of the cipher block stage 100 of the encryption performed on the previous data block. That is, because of the feedback, the reversible combinatorial operation stage in a CBC encryption engine must wait for the block cipher to complete its encryption of a given data block-bit vector combination before it can begin to process the next data block.

Furthermore, on the decryption side, the CBC mode's dependence on the sequential order of data block encryption can raise problems when one wants to retrieve only a portion of the encrypted data segment. For example, for a data segment that comprises data blocks DB_1 through DB_{20}, when that data segment is encrypted and stored for subsequent retrieval in its encrypted form, an instance may arise where there is a need to retrieve data blocks DB_6 through DB_{10}, wherein the other data blocks of the data segment are not needed. However, to be able to successfully decrypt data blocks DB_6 through DB_{10}, the retrieval operation and decryption operation will nevertheless need to operate on data blocks DB_1 through DB_5 so that decryption can be performed for data blocks DB_6 through DB_{10}.

Furthermore, when used for disk encryption, the CBC mode may be vulnerable to a "watermark attack" if the initialization vector 200 is not kept secret (such as may be the case when the initialization vector is derived from a quantity such as a disk volume number). With such an attack, an adversary can determine from the output ciphertext whether or not a specially crafted file

is stored. While there are solutions to such an attack (such as using hashing to derive the initialization vector from the data blocks in the sector), these solutions add to the computational complexity of the encryption operation and thus further degrade the throughput and/or increase the computational resources required for the encryption.

A second approach is known as the Segmented Integer Counter (SIC) mode, or more succinctly the counter (CTR) mode. Figure 3 depicts an example of the SIC/CTR mode of encryption/decryption. The SIC/CTR mode key encrypts a block comprising a combination of a random value (or nonce) and a counter value. This random value-counter combination can be achieved in any of a variety of ways (e.g., concatenation, XOR, etc.) The counter values may be any sequence of values that do not repeat over a long duration, but a simple incremental counter is believed to be the most commonly-used approach. The output of the block cipher 100 is then combined with the plaintext block using a reversible combinatorial operation 210 (e.g., XOR), with the output of the operation 210 being the ciphertext block. The SIC/CTR mode belongs to the general class of encryption modes known as a stream cipher.

As shown in Figure 3, at time $t=t_0$, the random value 300 is combined with a counter value 308 in some manner to create a random value-counter combination block 302. This block 302 is then encrypted by block cipher 100 using key 114, and the output therefrom is then XORed with plaintext block 102 to generate ciphertext block 322. Next, at time $t=t_1$, the random value 300 is combined with a next counter value 310 in some manner to create the random value-counter combination block 304. This block 304 is then encrypted by block cipher 100 using key 114, and the output therefrom is then XORed with plaintext block 104 to generate ciphertext block 324. Finally, at time $t=t_2$, the random value 300 is combined with a next counter value 312 in some manner to create the random value-counter combination block 306. This block 306 is then encrypted by block cipher 100 using key 114, and the output therefrom is then XORed with plaintext block 106 to generate ciphertext block 326.

On the decryption side, this process can then be reversed where the combination blocks 302, 304 and 306 are decrypted by block cipher 100 using key 114, with the respective outputs therefrom being XORed with the ciphertext blocks 322, 324 and 326 respectively to reconstruct plaintext blocks 102, 104 and 106.

The SIC/CTR mode of encryption/decryption also suffers from a security issue if data segments are always encrypted with the same random value 300. If an adversary is able to gather several versions of the encrypted data segment, it would be possible to derive information about the plaintext because the cipher text (C) is simply the XOR of the variable (V) based on the random number and the plaintext (P), e.g., $C = P \oplus V$, thus $C \oplus C' = P \oplus P'$.

Therefore, the inventors herein believe that a need exists in the art for a robust encryption/decryption technique that is capable of reducing both inter-segment and intra-segment uniformity while still retaining high throughput and exhibiting blockwise independence. As used herein, an encryption operation for a data segment is said to be "blockwise independent" when the encryption operations for each data block of that data segment do not rely on the encryption operation for any of the other data blocks in that data segment. Likewise, a decryption operation for a data segment is said to be "blockwise independent" when the decryption operations for each encrypted data block of that data segment do not rely on the decryption operation for any of the other data blocks in that data segment.

Toward this end, in one embodiment, the inventors herein disclose a technique for encryption wherein prior to key encryption, the plaintext data block is combined with a blockwise independent bit vector using a reversible combinatorial operation to thereby create a plaintext block-vector combination. This plaintext block-vector combination is then key encrypted to generate a ciphertext block. This process is repeated for all data blocks of a data segment needing encryption. For decryption of the cipher text blocks produced by such encryption, the inventors herein further disclose an embodiment wherein each ciphertext data block is key decrypted to reconstruct each plaintext block-vector combination. These reconstructed plaintext block-vector combinations can then be

combined (using the reversible combinatorial operation) with the corresponding randomized bit vectors that were used for encryption to thereby reconstruct the plaintext blocks.

As an improvement relative to the CBC mode of encryption/ decryption, each bit vector is blockwise independent. A bit vector is said to be blockwise independent when the value of that bit vector does not depend on any results of an encryption/decryption operation that was performed on a different data block of the data segment. Because of this blockwise independence, this embodiment is amenable to implementations that take advantage of the power of pipelined processing and/or parallel processing.

Moreover, because of the blockwise independent nature of the encryption performed by the present invention, a subset of the encrypted data segment can be decrypted without requiring decryption of the entire data segment (or at least without requiring decryption of the encrypted data blocks of the data segment that were encrypted prior to the encrypted data blocks within the subset). Thus, for a data segment that comprises data blocks DB_1 through DB_{20}, when that data segment is encrypted and stored for subsequent retrieval in its encrypted form using the present invention, a need may arise to retrieve plaintext versions of encrypted data blocks DB_6 through DB_{10} and DB_{15}, wherein the other data blocks of the data segment are not needed in their plaintext forms. A preferred embodiment of the present invention supports successful decryption of a subset of data blocks within the encrypted data segment (e.g., data blocks DB_6 through DB_{10} and DB_{15}) without requiring the decryption of the data segment's data blocks that are not members of the subset (e.g., data blocks DB_1 through DB_5, data blocks DB_{11} through DB_{14} and data blocks DB_{16} through DB_{20}). Accordingly, the present invention supports the decryption of any arbitrary subset of the encrypted data blocks of a data segment without requiring decryption of any data blocks that are non-members of the arbitrary subset even if those non-member data blocks were encrypted prior to the encryption of the data blocks within the arbitrary subset.

Similarly, even if an entire encrypted data segment is to be decrypted, the present invention supports the decryption of the encrypted data blocks in a block order independent manner. Further

still, the present invention supports the encryption of data blocks in a block order independent manner as well as supports limiting the encryption to only a defined subset of a data segment's data blocks (wherein such a subset can be any arbitrary subset of the data segment's data blocks).

Furthermore, as an improvement relative to the SIC/CTR mode of encryption/decryption, a greater degree of security is provided by this embodiment because the data that is subjected to key encryption includes the plaintext data (whereas the SIC/CTR mode does not subject the plaintext data to key encryption and instead subjects only its randomized bit vector to key encryption).

Preferably, the blockwise independent bit vector is a blockwise independent randomized bit vector. As is understood by those having ordinary skill in the art, randomization in this context refers to reproducible randomization in that the same randomized bit vectors can be reproduced by a bit vector sequence generator given the same inputs. Further still, the blockwise independent randomized bit vector is preferably generated from a data tag that is associated with the data segment needing encryption/decryption. Preferably, this data tag uniquely identifies the data segment. In a disk encryption/decryption embodiment, this data tag is preferably the logical block address (LBA) for the data segment. However, it should be noted that virtually any unique identifier that can be associated with a data segment can be used as the data tag for that data segment. It should also be noted that rather than using a single data tag associated with the data segment, it is also possible to use a plurality of data tags that are associated with the data segment, wherein each data tag uniquely identifies a different one of the data segment's constituent data blocks.

A bit vector generation operation preferably operates on a data tag to generate a sequence of blockwise independent bit vectors, each blockwise independent bit vector for reversible combination with a corresponding data block. Disclosed herein are a plurality of embodiments for such a bit vector generation operation. As examples, bit vectors can be derived from the pseudo-random outputs of a pseudo-random number generator that has been seeded

with the data tag; including derivations that employ some form of feedback to enhance the randomness of the bit vectors. Also, linear feedback shift registers and adders can be employed to derive the bit vectors from the data tag in a blockwise independent manner.

The inventors also disclose a symmetrical embodiment of the invention wherein the same sequence of operations are performed on data in both encryption and decryption modes.

One exemplary application for the present invention is to secure data at rest in non-volatile storage; including the storage of data placed on tape, magnetic and optical disks, and redundant array of independent disks (RAID) systems. However, it should be noted that the present invention can also be applied to data in flight such as network data traffic.

These and other features and advantages of the present invention will be apparent to those having ordinary skill in the art upon review of the following description and figures.

Brief Description of the Drawings:

Figure 1 depicts an example of a known ECB mode of encryption/decryption;

Figure 2 depicts an example of a known CBC mode of encryption/decryption;

Figure 3 depicts an example of a known SIC/CTR mode of encryption/decryption;

Figure 4 depicts an exemplary data segment;

Figures 5(a) and (b) depict an embodiment of the present invention in both encryption and decryption modes;

Figure 6 depicts an exemplary bit vector sequence generator;

Figures 7(a) and (b) depict exemplary encryption and decryption embodiments of the present invention;

Figures 8(a) and (b) depict exemplary encryption and decryption embodiments of the present invention showing their operations over time;

Figure 9 depicts an exemplary embodiment of a bit vector sequence generator;

Figures 10(a)-(c) depict three additional exemplary embodiments of a bit vector sequence generator;

Figure 11 depicts an exemplary embodiment of the present invention where multiple block ciphers are chained together;

Figures 12(a) and (b) depict exemplary encryption and decryption embodiments of the present invention that are hybrids of the embodiments of Figures 8(a) and (b) and the CBC mode of encryption/decryption;

Figures 12(c) and (d) depict exemplary embodiments of the bit vector sequence generator for use with the hybrid embodiments of Figures 12(a) and (b);

Figures 13(a) and (b) depict an exemplary embodiment for symmetrical encryption/decryption in accordance with the present invention;

Figures 14(a) and (b) depict an exemplary embodiment for symmetrical encryption/decryption in accordance with the present invention wherein the blockwise independent bit vectors are derived from the data segment's LBA;

Figures 15(a) and (b) depict the embodiment of Figures 14(a) and (b) showing its operation over time;

Figures 15(c) and (d) depict a symmetrical encryption/decryption counterpart to the embodiments of Figures 12(a) and (b);

Figure 16 depicts a parallel architecture for encrypting or decrypting data blocks;

Figures 17(a) and (b) depict exemplary hardware environments for the present invention; and

Figures 18(a)-(c) depict exemplary printed circuit boards on which the encryption/decryption embodiments of the present invention can be deployed.

Detailed Description of the Preferred Embodiments:

Figure 4 illustrates an exemplary data segment 400 on which the encryption/decryption technique of the present invention can be performed. The data segment 400 comprises a plurality of data blocks 102, 104, 106, … Each data block comprises a plurality of data bits and preferably has a fixed length (e.g., 64 bits, 256 bits, etc.). In an exemplary embodiment, wherein AES block ciphers are used, which as explained above operate on 16-byte data blocks,

it is preferred that the data blocks 102, 104, 106 … possess a length of 16 bytes. It should also be noted that the size of the data segment 400 is typically much larger than the size of an individual data block. For example, a data storage system may operate on "logical blocks" of data having a size of 512 bytes. In such a case, the "logical block", which can serve as the data segment 400, will be a collection of 32 16-byte data blocks.

Figure 5(a) illustrates an embodiment of the present invention wherein the encryption operation is segmented into a plurality of stages. At stage 504, the blockwise independent bit vector 506 is generated, preferably from a data tag 502 that is associated with the data segment 400. Preferably, the bit vector 506 has a length that is the same as the data blocks of the data segment, although this need not be the case. Further still, it is preferred that the blockwise independent bit vector 506 have a randomized value to thereby enhance the security of the encryption. Also, it is preferred that a different bit vector 506 be generated for each data block of a data segment that is encrypted, although this need not be the case. The bit vectors that are used in the encryption of a data segment's data blocks should be either stored for subsequent use when it is time to decrypt one or more of the data segment's data blocks or should be reproducible from a known quantity (such as the data tag) when it is time to decrypt one or more of the data segment's data blocks.

At stage 210, a reversible combinatorial operation such as a bitwise XOR operation is performed on the blockwise independent bit vector 506 and plaintext data block. This reversible combinatorial operation preferably produces a data block-bit vector combination 508.

At stage 100, a block cipher performs an encryption operation on the data block-bit vector combination 508 using key 114 as per well-known key encryption techniques (e.g., AES, the Data Encryption Standard (DES), the triple DES (3DES), etc.). The output of the block cipher stage 100 is thus a ciphertext data block that serves as the encrypted counterpart to the plaintext data block that was fed into stage 210. It should be noted that any of several well-known key management techniques can be used in connection with

managing the key(s) 114 used by the block cipher(s) 100. As such, the inventors do not consider the key management for the block cipher(s) 100 to be any limitation on the present invention. It should also be noted that "keyless" encryption techniques may also be used in the practice of the present invention (e.g., substitution ciphers that do not require a key).

Figure 5(b) depicts the decryption counterpart to Figure 5(a). In Figure 5(a), the flow of data blocks and stages is reversed such that the ciphertext data block is first key decrypted by stage 100 to reconstruct combination 508. Combination 508 is in turn combined with the same bit vector 506 that was used when creating that ciphertext data block and using the same reversible combinatorial operation 210 that was used when creating that ciphertext data block, to thereby reconstruct the plaintext data block.

As can be seen in Figures 5(a) and (b), no feedback is required between stages, thus allowing this encryption/decryption technique to be implemented in a pipelined architecture and/or a parallel processing architecture for the achievement of a high throughput when performing encryption/decryption. Thus, as a stream of data blocks are sequentially processed through the encryption/decryption stages, a high throughput can be maintained because the reversible combinatorial stage 210 can operate on a given data block while the block cipher stage 100 simultaneously operates on a different data block because the reversible combinatorial operation stage 210 does not require feedback from the block cipher stage 100 to operate.

The data tag 502 may be any data value(s) that can be associated with the data segment 400. Preferably, the data tag 502 serves as a unique identifier for the data segment 400, although this need not be the case. A preferred data tag 502 is the logical block address (LBA) for the data segment to be encrypted. An LBA for a data segment is the logical memory address for the data segment that is typically assigned by an Operating System (OS) or memory management system. However, other data tags may be used in the practice of the present invention; examples of which include file identifiers, physical memory addresses, and packet sequence numbers. The source of the data tag can be any of a variety of

sources, including but not limited to communication protocol, storage subsystem, and file management systems.

Figure 6 illustrates how a sequence of bit vectors 506 can be generated from a data tag 502. As an exemplary embodiment of bit vector generation stage 504, bit vector sequence generator 600 preferably operates to produce a plurality of blockwise independent randomized bit vectors 506_i from an input comprising data tag 502. Figures 9 and 10, to be described hereinafter, illustrate various exemplary embodiments for the bit vector sequence generator 600.

Figure 7(a) and (b) illustrate embodiments of the invention where the data segment's LBA is used as the data tag 502 for the encryption/decryption operations. Sequence generator 600 processes the LBA to produce a different blockwise independent randomized bit vector 506 for XOR combination (210) with each plaintext data block. On decryption (shown in Figure 7(b)), the sequence generator 600 operates to produce the same plurality of different bit vectors 506 from the data segment's LBA as were produced by the sequence generator 600 for encryption (see Figure 7(a)) given the same LBA input. Thus, as shown in Figure 7(b), each bit vector 506 is then used for XOR combination (210) with each decrypted ciphertext block.

Figure 8(a) illustrates the embodiment of Figure 7(a) (wherein the LBA is labeled as an initialization vector), but depicting how the encryption operation can proceed over time. Thus, at time $t=t_0$, plaintext data block 102 is reversibly combined with bit vector 506_1 produced by sequence generator 600 to generate a data block-bit vector combination that is key encrypted by a block cipher 100 to thereby produce an encrypted data block-bit vector combination 802 which serves as the ciphertext block. Subsequently, at time $t=t_1$, the sequence generator produces another bit vector 506_2 for reversible combination with plaintext data block 104. The resultant data block-bit vector combination is then key encrypted by the block cipher 100 to thereby produce an encrypted data block-bit vector combination 804 which serves as the next ciphertext block. This process then continues for subsequent clock cycles as additional data blocks of the data segment 400 are encrypted.

Figure 8(b) depicts the decryption counterpart to Figure 8(a), wherein ciphertext blocks 802, 804 and 806 are decrypted in

accordance with the embodiment of Figure 7(b) to reproduce plaintext data blocks 102, 104 and 106.

Figure 9 depicts an embodiment of the sequence generator 600 wherein a data tag 502 such as the LBA is used to seed a pseudo-random number generator (PRNG) 900. When encrypting a first data block, the bit vector 506 is initialized to be the LBA itself. Then, when encrypting subsequent data blocks, the bit vector 506 is incremented through adder 902 by the pseudo-random output from the PRNG 900. Preferably, a new pseudo-random number is generated by the PRNG 900 for each new data block of the data segment needing encryption. By using a PRNG 900 to generate counter increments for the bit vector 506, the sequence of bit vectors 506 used for encrypting different data segments (identified by their LBA) will be difficult to predict and provide more security than a simple counter. For decryption, it should be noted that the PRNG 900 should operate to produce the same sequence of pseudo-random outputs given the same data tag input, to thereby enable the generation of the same set of bit vectors 506 when decrypting the encrypted data segment (or a subset of the encrypted data segment).

As can be seen, the sequence of bit vectors 506_1, 506_2, ... 506_n produced by the sequence generator 600 of Figure 9 will be sequentially dependent in that each successive bit vector 506_i will be a function of the previous bit vector 506_{i-1} (via feedback to adder 902). This sequential nature of the bit vectors does not preclude their use in a blockwise independent encryption/decryption scheme. For example, consider a case where a data tag (such as an LBA) for a data segment comprising twenty data blocks is used as the basis for the blockwise independent bit vectors, but it is only desired to encrypt/decrypt data blocks DB_6 through DB_{10}. In such a case, the sequence generator 600 is preferably initialized with the data tag and the bit vectors for data blocks DB_1 through DB_5 are generated but discarded by the sequence generator 600. Such a configuration will require the reversible combinatorial stage 210 and the downstream encryption stage 100 to pause until the bit vector 506_6 for data block DB_6 is generated. While this pause produces a delay and degradation in throughput for the encryption/decryption technique, relative to the multiple iterations

through a block cipher as required in the conventional CBC mode of encryption, the inventors herein believe that this delay and throughput degradation is relatively minor. For example, this pause will not need to wait for data blocks DB_1 through DB_5 to be encrypted/decrypted via block cipher 100 before being able to process data block DB_6.

It should also be noted that if the encryption/decryption technique involves using a data tag that is unique to each data block to generate each data block's corresponding blockwise independent bit vector 506, the need to pause operations while cycling through unneeded bit vectors can be eliminated.

Figures 10(a)-(c) depict other examples of sequence generator embodiments. Figure 10(a) discloses a sequence generator 600 that uses the LBA 502 to seed a PRNG 900 whose pseudo-random outputs then serve as the bit vectors 506 for combination with the data segment's data blocks. As with the embodiment of Figure 9, preferably the LBA itself is used as the bit vector 506 for reversible combination with a first data block to be encrypted/decrypted.

Figure 10(b) discloses a sequence generator 600 that uses the LBA 502 to seed a linear feedback shift register (LFSR) 1000 whose outputs then serve as the bit vectors 506 for combination with the data segment's data blocks.

Figure 10(c) discloses a sequence generator 600 that uses the LBA 502 to seed a feedback counter 1002, wherein the feedback counter 1002 has a constant increment 1004, and wherein the counter's outputs then serve as the bit vectors 506 for combination with the data segment's data blocks. As with the embodiments of Figure 9 and Figure 10(a), preferably the LBA itself is used as the bit vector 506 for reversible combination with a first data block to be encrypted/decrypted. It should be noted that the sequence generator embodiment of Figure 10(c) can be configured to accommodate encryption/decryption of arbitrary subsets of data blocks within a data segment without requiring a pause while the sequence generator cycles through unneeded bit vectors. If an encryption/decryption is to begin at a data block within a data segment that is not the first data block of the data segment (e.g., data block DB_k of a data segment, wherein $k>1$), the data tag 502

(such as an LBA) that is passed to the sequence generator 600 can be computed as:

$$\text{Data Tag}' = \text{Data Tag} + k*\text{Constant}$$

wherein *Data Tag'* represents the value of the data tag 502 that is fed into the sequence generator 600, wherein *Data Tag* represents the value of the data tag that is associated with the data segment, wherein *k* represents the block number within the data segment of the data block to be encrypted/decrypted, and wherein *Constant* represents the value of the incremental constant 1004 for adder 1002. This computation can be performed either within the sequence generator (in which case it will be the value *Data Tag* that is fed into the sequence generator 600) or in a module upstream from the sequence generator. Appropriate control logic is preferably used to control whether the multiplexer passes the data tag value 502 or the output of adder 1002 on to the reversible combinatorial stage 210.

It should also be noted that the present invention need not be limited to a single combination of a blockwise independent bit vector randomizer and a block cipher. Pairs of sequence generators 600, reversible combinatorial operations 210, and block ciphers 100 can be sequentially chained as shown in Figure 11. Thus, a first sequence generator 600_1, a first reversible combinatorial operator 210_1 and a first block cipher 100_1 can operate to produce an encrypted data block-bit vector combination that is fed into a second reversible combinatorial operator 210_2 for reversible combination with a bit vector produced by a second sequence generator 600_2. The resultant encrypted data block-bit vector-bit vector combination produced by reversible combinatorial operator 210_2 can then be key encrypted by block cipher 100_2. The inventors herein believe that such chaining can enhance the security of an encryption system. Moreover, the inventors note that still greater numbers of sequence generators 600, reversible combinatorial operations 210, and block ciphers 100 can be sequentially chained to enhance security if desired by a practitioner of this embodiment of the invention. It should also readily be understood that corresponding sequential decryption chains can be used. Preferably, in such a sequential chaining embodiment, each different sequence

generator 600_i will operate to produce different set of bit vectors given the same input.

Further still, the inventors herein disclose an embodiment that hybridizes the present invention and the CBC mode of encryption/decryption. Figure 12(a) illustrates an example of such an embodiment to perform encryption. This configuration provides the flexibility to include some feedback for higher security. Note that the first output of ciphertext 1200 is not used as feedback to the second encryption operation 1202, rather it is used as feedback for encryption performed by subsequent block i where i is a feedback stride. The feedback stride can then be chosen to provide a favorable balance among security and throughput. If the feedback stride i is greater than or equal to the number of pipeline stages in the block cipher, then there is no performance penalty because there will need not be a delay in the insertion of a block into a block cipher. Furthermore, if one does choose a lower feedback stride value that would require a delay, one can introduce stall cycles in the processing. The added security provided by the technique of Figure 12(a) is that the encryption technique of Figure 12(a) does not exclusively rely on the sequence generator 600 (or the PRNG 900 in the sequence generator 600) to generate long, difficult to predict initialization sequences. Once the system begins feeding back ciphertext from previous blocks via feedback link 1206, the system gains the strength of the block cipher in producing more random initialization bit vectors. This technique essentially narrows the visibility of an observer into the "window" of the random increments produced by the PRNG 900. Thus, it is more difficult for observers to reconstruct the entire random sequence generated by the PRNG 900 (thereby making it more difficult for one to crack the encryption scheme). Figure 12(b) depicts a decryption counterpart to Figure 12(a).

Figures 12(c) and (d) depict exemplary embodiments of a sequence generator 600' that could be used to generate bit vectors for the embodiments of Figures 12(a) and (b). In the example of Figure 12(c), the sequence generator 600' comprises any of the embodiments for sequence generator 600 as described in connection with Figures 9 and 10(a)-(c). The bit vector 506 that is output by

the sequence generator 600 is preferably reversibly combined with the feedback ciphertext i from link 1206 via reversible combinatorial operator 1250 to produce bit vector 506' (which is in turn provided to the reversible combinatorial operator 210) when the conditions for the feedback stride i are met.. Sequence generator 600' also preferably includes appropriate control logic to ensure that the feedback stride i defined for the hybrid embodiment is obeyed. As an example, such control can be achieved with a multiplexer 1262 whose inputs are either null value or the feedback ciphertext i. A counter-based control circuit 1260 can define which of the inputs to multiplexer 1262 are passed to the reversible combinatorial operator 1250 such that the feedback ciphertext i is only passed on when it is time to use the ciphertext to further randomize the bit vectors.

Figure 12(d) depicts another exemplary embodiment for the sequence generator 600'. In the example of Figure 12(d), the sequence generator 600' comprises any of the embodiments for sequence generator 600 as described in connection with Figures 9 and 10(a)-(c). The sequence generator 600 will receive as an input either the data tag 502 or the feedback ciphertext i, as defined by control logic. The control logic is preferably configured to pass on the feedback ciphertext to seed the sequence generator 600 only when the conditions for the feedback stride i are met. . As an example, such control can be achieved with a multiplexer 1262 whose inputs are either the data tag 502 or the feedback ciphertext i. A counter-based control circuit 1260 can define which of the inputs to multiplexer 1262 are passed to the sequence generator 600 such that the feedback ciphertext i is only passed on when it is time to use the ciphertext to further randomize the bit vectors.

As another embodiment of the present invention, the inventors disclose a symmetrical embodiment for encryption/decryption. With "symmetrical" encryption/decryption, the same order of operations can be performed on data blocks to both encrypt and decrypt those data blocks. Thus, with a symmetrical embodiment, the same module that is used to encrypt data can be used to decrypt encrypted data. Figures 13(a) and (b) illustrate a symmetrical embodiment of the present invention. As can be seen, the same order of operations is

used by Figure 13(a) to encrypt a data block as is used by Figure 13(b) to decrypt a ciphertext data block. The symmetrical encryption/decryption engine 1300 comprises a first reversible combinatorial stage 210, a block cipher operation stage 100, and a second reversible combinatorial stage 1302. A bit vector generation stage 504 (such as the sequence generators 600 shown in Figure 9 and Figures 10(a)-(c)) operates to produce blockwise independent bit vectors 506 that are fed to both the first reversible combinatorial stage 210 and the second reversible combinatorial stage 1302.

As shown in Figure 13(a), for encryption, a plaintext data block is reversibly combined with a blockwise independent bit vector 506 by first reversible combinatorial operation stage 210 (preferably XOR logic), to thereby generate a data block-bit vector combination 508. Block cipher 100 then performs a block cipher operation on this data block-bit vector combination 508 using a key. The resultant block ciphered data block-bit vector combination 1304 is then reversibly combined with a blockwise independent bit vector 506 by second reversible combinatorial operation stage 1302 (preferably XOR logic), to thereby generate a block ciphered data block-bit vector-bit vector combination 1306, which can serve as the ciphertext for the plaintext data block.

For decryption, as shown in Figure 13(b), the same order of operations is used, albeit starting from a ciphertext data block rather than a plaintext data block. The ciphertext data block used for decryption will be a block ciphered data block-bit vector-bit vector combination 1306 that was produced during the encryption operation. First reversible combinatorial operation stage 210 operates to reversibly combine such a ciphertext data block with the same bit vector 506 that was used by the second reversible combinatorial operation stage 1302 when encrypting that ciphertext data block. The result of this reversible combination will be a reconstruction of the block ciphered data block-bit vector combination 1304. Block cipher 100 then performs a block cipher operation (decryption in this example) using the key to reconstruct the data block-bit vector combination 508. Second reversible combinatorial operation stage 210 then operates to reversibly combine the reconstructed data block-bit vector combination 508 with

the same bit vector 506 that was used by the first reversible combinatorial operation stage 210 when encrypting that ciphertext data block. The output of the second reversible combinatorial operation stage 1302 then serves as a reconstruction of the plaintext data block.

Timing logic (not shown) can be employed to synchronize the outputs of bit vectors 506 from the bit vector generation stage 504 such that the appropriate bit vector 506 is fed to the second reversible combinatorial stage 1302 for each block ciphered data block-bit vector combination 1304 (or reconstructed data block-bit vector combination 508 for the decryption mode) that is processed thereby. Such synchronization could be designed to accommodate the latency within the block cipher 100 to thereby allow the same bit vector 506 to be used for reversible combination with a given data block by first reversible combinatorial operation stage 210 as is used for later reversible combination with the block ciphered data block-bit vector combination 1304 derived from that given data block by the second reversible combinatorial operation stage 1302.

Figure 14(a) (for encryption mode) and Figure 14(b) (for decryption mode) depict an example of the symmetrical embodiment of Figures 13(a) and (b), wherein the bit vectors 506 are derived from the LBA for the data segment 400.

Figure 15(a) (for encryption mode) and Figure 15(b) (for decryption mode) depict the operation of the embodiment of Figures 14(a) and (b) over time.

It should also be noted that the symmetrical encryption/ decryption embodiments described herein can also be used in a hybrid CBC embodiment like the ones shown in Figures 12(a) and (b). An example of such a symmetrical hybrid embodiment is shown in Figures 15(c) and (d), wherein the feedback link 1502 carries the block ciphered data block-bit vector-bit vector output 1306 of the second reversible combinatorial operation stage 1302 performed for the first data block. The sequence generators 600' as shown in Figures 12(c) and (d) can be employed, although the feedback ciphertext will preferably emanate from the output of the second reversible combinatorial operator 1302 rather than the output of the block cipher 100.

As a further embodiment of the present invention, the inventors note that a parallel architecture 1600 such as the one shown in Figure 16 can be employed. With this parallel architecture, a stream of incoming data blocks 1604 (which can be either plaintext data blocks or ciphertext data blocks) are separated into a plurality of parallel streams for processing by parallel encryption/decryption engines 1602. Such encryption/decryption engines can take the form of any of the embodiments of the invention described herein such as those shown in connection with Figures 5(a) and (b), 7(a) and (b), 11, 12(a) and (b), 13(a) and (b), and 14(a) and (b). The resultant data streams produced by each parallel encryption/decryption engine 1602 can then be brought together to form the outgoing data stream 1606 (which may be either plaintext data blocks or ciphertext data blocks depending on whether the encryption/decryption engines 1602 performed encryption or decryption). It is also worth noting that each parallel engine 1602 can employ its own bit vector generation stage 504, or the same bit vector generation stage 504 can be shared by multiple (or all) of the parallel encryption engines 1602.

The encryption/decryption techniques of the present invention can be implemented in a variety of ways including but not limited to a software implementation on any programmable processor (such as general purpose processors, embedded processors, network processors, etc.), a hardware implementation on devices such as programmable logic devices (e.g., field programmable gate arrays (FPGAs)), ASICs, and a hardware and/or software implementation on devices such as chip multi-processors (CMPs), etc. For example, some CMPs include built-in hardware for encryption ciphers, in which case software on parallel processors systems for the CMPs could perform the bit vector generation and reversible combinatorial tasks while offloading the block cipher operations to the dedicated hardware.

However, the inventors herein particularly note that the present invention is highly amenable to implementation in reconfigurable logic such as an FPGA. Examples of suitable FPGA platforms for the present invention are those described in the following: U.S. patent application serial number 11/339,892 (filed January 26, 2006, entitled "Firmware Socket Module for FPGA-Based

Pipeline Processing" and published as _____), published PCT applications WO 05/048134 and WO 05/026925 (both filed May 21, 2004 and entitled "Intelligent Data Storage and Processing Using FPGA Devices"), pending U.S. patent application serial number 10/153,151 (filed May 21, 2002 entitled "Associative Database Scanning and Information Retrieval using FPGA Devices", published as 2003/0018630, now U.S. patent 7,139,743), and U.S. patent 6,711,558 (entitled "Associative Database Scanning and Information Retrieval"), the entire disclosures of each of which are incorporated by reference herein.

Figure 17(a) depicts an example of an implementation environment for the present invention. Figure 17(a) depicts a system 1700 comprising a host processor 1708 and host RAM 1710 in communication with a disk controller 1706 via bus 1712. Disk controller 1706 governs access to data store 1704 which may be any device capable of storing data. In an exemplary embodiment, data store 1704 is a mass storage medium such as a RAID system or subsystem. In such an instance, disk controller 1706 is a RAID controller.

Data flowing to or from data store 1704 can be routed through reconfigurable logic device 1702 (which may be embodied by an FPGA). One or more firmware application modules (FAMs) 1730 are deployed on the reconfigurable logic using the techniques described in the above-incorporated references. The different stages of the encryption/decryption engine of the present invention can be implemented on the reconfigurable logic device 1702 as a processing pipeline deployed on one or more of these FAMs 1730. Firmware socket module 1720 can be implemented as described in the incorporated 11/339,892 patent application to control the flow of data to and from the encryption/decryption engine(s) deployed on the reconfigurable logic device 1702 via communication paths 1732 and 1734. Data to be encrypted and stored in the data store can be routed through the reconfigurable logic device 1702 along with appropriate control instructions for the encryption. Such control information can include the data tag used to generate the blockwise independent bit vectors. Moreover, these control instructions can emanate from any source with access to system bus 1712 including

sources that connect to the system bus 1712 over a network. For example, in an embodiment wherein the data segment's LBA is used as the data tag from which the bit vectors are generated, the LBA can be passed to the FAM pipeline 1730 with the data from the data store 1704 or it can be passed to the FAM pipeline 1730 from processor 1708. Moreover, the data segments to be encrypted can emanate from any source with access to the reconfigurable logic device 1702. Encrypted data to be decrypted can also be routed through the reconfigurable logic device 1702 along with appropriate control instructions for the decryption.

Thus, when encrypting a data segment to be stored at an LBA of the data store 1704, the data blocks of the data segment can be streamed through a FAM 1730 on reconfigurable logic device 1702 that is configured to perform encryption in accordance with the teachings of the present invention (with the encryption FAM 1730 preferably deriving the blockwise independent bit vectors 506 from the LBA). The resultant ciphertext produced by the encryption FAM 1730 can then be stored in data store 1704 starting at the LBA. On decryption, the ciphertext data blocks of the encrypted data segment (or a subset thereof) can be streamed through a decryption FAM 1730 (or a symmetrical encryption/decryption FAM 1730) to reconstruct the plaintext data segment (or subset thereof). Once again, in an embodiment wherein the blockwise independent bit vectors are derived form the data segment's LBA, the LBA can also be used as the source of the bit vectors used during the decryption process.

It should also be noted that for disk or file encryption operations, it may be desirable to include the platform (e.g., FPGA or ASIC) on which the encryption/decryption engine of the present invention is deployed (or the encryption/decryption engine itself) on-board the disk controller 1706. It may also be desirable for the encryption/decryption engine to receive all data streaming to/from the disk(s), in which case control information could be added to the data streams to inform the encryption/decryption engine of which data is to be encrypted/decrypted and which data is to be passed through without modification. For example, such control information can take the form of a flag within a data set's SCSI control block (SCB).

The embodiment of Figure 17(b) depicts the system 1700 wherein bus 1712 is also connected to a network 1742 through network interface 1740. Such a network 1742 can also serve as a source or destination for data to be encrypted or decrypted (e.g., network data traffic such as network data packets that may need encryption/decryption). It should also be noted that system 1700 can be configured such that bus 1712 connects to a network 1742 (through network interface 1742) but not to a data store 1704 (through disk controller 1706) if desired by a practitioner of the present invention in view of the use(s) to which the practitioner intends to put the invention.

Figure 18(a) depicts a printed circuit board or card 1800 that can be connected to the PCI-X bus 1712 of a computer system (e.g., a commodity computer system or other) for use in encrypting/decrypting data. In the example of Figure 18(a), the printed circuit board includes an FPGA chip 1802 (such as a Xilinx Virtex 4 FPGA) that is in communication with a memory device 1804 and a PCI-X bus connector 1806. A preferred memory device 1804 comprises SRAM and DRAM memory. A preferred PCI-X bus connector 1806 is a standard card edge connector.

Figure 18(b) depicts an alternate configuration for a printed circuit board/card 1800. In the example of Figure 18(b), a private bus 1808 (such as a PCI-X bus), a disk controller 1810, and a disk connector 1812 are also installed on the printed circuit board 1800. Any commodity disk interface technology can be supported, as is understood in the art. In this configuration, the firmware socket 1720 also serves as a PCI-X to PCI-X bridge to provide the processor 1708 with normal access to the disk(s) connected via the private PCI-X bus 1808.

Figure 18(c) depicts another alternate configuration for a printed circuit board/card 1800. In the example of Figure 18(b), a private bus 1808 (such as a PCI-X bus), a network interface controller 1820, and a network connector 1822 are also installed on the printed circuit board 1800. Any commodity network interface technology can be supported, as is understood in the art. In this configuration, the firmware socket 1720 also serves as a PCI-X to

PCI-X bridge to provide the processor 1708 with normal access to the network(s) connected via the private PCI-X bus 1808.

It should be further noted that the printed circuit board/card 1800 may also be configured to support both a disk controller/ connector 1810/1812 and a network interface controller/connector 1820/1822 to connect the board 1800 to disk(s) and network(s) via private PCI-X bus 1808, if desired by a practitioner of the invention.

It is worth noting that in either of the configurations of Figures 18(a)-(c), the firmware socket 1720 can make memory 1804 accessible to the PCI-X bus, which thereby makes memory 1804 available for use by an OS kernel for the computer system as the buffers for transfers from the disk controller and/or network interface controller to the FAMs. It is also worth noting that while a single FPGA chip 1802 is shown on the printed circuit boards of Figures 18(a)-(c), it should be understood that multiple FPGAs can be supported by either including more than one FPGA on the printed circuit board 1800 or by installing more than one printed circuit board 1800 in the computer system. Further still, it should be noted that the printed circuit boards 1800 of the embodiments of Figures 18(a)-(c) can use an ASIC chip on which the encryption/decryption engines are deployed rather than an FPGA chip 1802. if desired by a practitioner of the invention.

Exemplary applications for the present invention include but are not limited to general purpose data encryption (e.g., files, images, documents, etc.), disk encryption, streaming message (e.g., packets, cells, etc.) encryption, and streaming image encryption (e.g., streaming reconnaissance imagery, etc.).

While the present invention has been described above in relation to its preferred embodiment, various modifications may be made thereto that still fall within the invention's scope. Such modifications to the invention will be recognizable upon review of the teachings herein. As such, the full scope of the present invention is to be defined solely by the appended claims and their legal equivalents.

WHAT IS CLAIMED IS:

1. A method of encrypting a plurality of data blocks, each data block comprising a plurality of data bits, the method comprising:

combining each data block with a corresponding bit vector using a reversible combinatorial operation to thereby create a plurality of data block-bit vector combinations; and

encrypting each data block-bit vector combination using a key encryption operation; and

wherein each bit vector has a value that is independent of all key encryption operations that were previously performed on the data block-bit vector combinations created from the other data blocks of the plurality of data blocks.

2. The method of claim 1 wherein the plurality of data blocks are part of a data segment, the method further comprising:

generating each bit vector based at least in part upon a data tag associated with the data segment.

3. The method of claim 2 wherein the generating step further comprises (1) seeding a pseudo-random number generator with the data tag to generate a plurality of pseudo-random outputs, and (2) generating a plurality of the bit vectors corresponding to the data blocks based at least in part upon the pseudo-random outputs.

4. The method of claim 3 wherein the combining step comprises, combining a first data block of the data segment with the data tag using the reversible combinatorial operation.

5. The method of claim 4 wherein the step of generating the bit vectors comprises incrementing the bit vector used for a previous data block of the data segment by one of the pseudo-random outputs to generate the bit vector to be combined with a next data block of the data segment.

6. The method of claim 3 wherein the step of generating the bit vectors corresponding to the data blocks based at least in part upon

the pseudo-random outputs comprises using the pseudo-random outputs as a plurality of the bit vectors.

7. The method of claim 2 wherein the generating step further comprises seeding a linear feedback shift register with the data tag to generate a plurality of the bit vectors.

8. The method of claim 2 wherein the generating step further comprises seeding a feedback counter with the data tag to generate a plurality of the bit vectors.

9. The method of claim 8 wherein the seeding step further comprises providing a constant increment to the feedback counter.

10. The method of claim 8 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the generating step comprises, prior to processing a data block of the subset, (1) calculating a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seeding the feedback counter with the calculated data tag.

11. The method of claim 2 wherein the data tag comprises a logical block address (LBA) for the data segment.

12. The method of claim 3 further comprising:
storing the encrypted data block-bit vector combinations on a disk at the LBA.

13. The method of claim 2 wherein the data tag comprises a file identifier associated with the data segment.

14. The method of claim 2 wherein the data tag comprises a physical memory address associated with the data segment.

15. The method of claim 2 wherein the data segment comprises network data traffic, and wherein the data tag comprises a packet sequence number associated with the data segment.

16. The method of claim 2 wherein the data segment has a plurality of associated data tags, wherein each data tag is uniquely associated with a different one of the data segment's data blocks, the method further comprising:

generating the bit vectors based at least in part upon the plurality of data tags.

17. The method of claim 2 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the generating step comprises, prior to processing a data block of the subset, cycling through and discarding any bit vectors corresponding to data blocks within the data segment that are not within the subset and that are prior to the data blocks within the subset.

18. The method of claim 1 wherein a plurality of the bit vectors comprise randomized bit vectors.

19. The method of claim 1 wherein each bit vector has a different value.

20. The method of claim 1 wherein the data segment comprises network data traffic.

21. The method of claim 1 further comprising:

combining each encrypted data block-bit vector combination with corresponding bit vector using a reversible combinatorial operation.

22. The method of claim 21 wherein the corresponding bit vector for each encrypted data block-bit vector combination is the same bit vector that was combined with the data block from which that encrypted data block-bit vector combination was created.

23. The method of claim 21 further comprising:
encrypting each encrypted data block-bit vector-bit vector combination using a key encryption operation.

24. The method of claim 1 wherein the reversible combinatorial operation comprises an XOR operation.

25. The method of claim 1 further comprising:
performing all of the method steps with reconfigurable logic.

26. The method of claim 1 further comprising:
performing all of the method steps with an application specific integrated circuit (ASIC).

27. A system for encrypting a plurality of data blocks, each data block comprising a plurality of data bits, the system comprising:
a bit vector generation stage configured to generate a plurality of blockwise independent bit vectors;
a reversible combinatorial operation stage configured to reversibly combine each data block with a corresponding one of the generated bit vectors to in turn generate a plurality of data block-bit vector combinations; and
an encryption stage configured to perform an encryption operation on the data block-bit vector combinations.

28. The system of claim 27 wherein the reversible combinatorial operation stage is further configured to sequentially reversibly combine each data block with its corresponding bit vector to generate the plurality of data block-bit vector combinations.

29. The system of claim 28 wherein the encryption stage is further configured to sequentially perform the key encryption operation on the data block-bit vector combinations.

30. The system of claim 27 further comprising a second reversible combinatorial operation stage configured to reversibly combine each

encrypted data block-bit vector combination with the bit vector used to generate the data block-bit vector combination corresponding to that encrypted data block-bit vector combination.

31. The system of claim 27 wherein the bit vector generation stage is further configured to generate a plurality of randomized blockwise independent bit vectors.

32. The system of claim 27 wherein the plurality of data blocks are part of a data segment, and wherein the bit vector generation stage is further configured to generate the plurality of blockwise independent bit vectors based at least in part upon a data tag associated with the data segment.

33. The system of claim 32 wherein the bit vector generation stage comprises a pseudo-random number generator that is seeded by the data tag, and wherein the bit vector generation stage is further configured to generate a plurality of the bit vectors based at least in part upon a plurality of pseudo-random number outputs of the pseudo-random number generator.

34. The system of claim 33 wherein the bit vector generation stage is configured to output the pseudo-random number outputs as a plurality of the bit vectors.

35. The system of claim 33 wherein the data tag comprises a logical block address (LBA) for the data segment.

36. The system of claim 33 wherein the bit vector generation stage further comprises an adder for adding a pseudo-random number output of the pseudo-random number generator with a blockwise independent bit vector that was reversibly combined with a previous data block to thereby generate a current blockwise independent bit vector for reversible combination with a current data block.

37. The system of claim 32 wherein the bit vector generation stage comprises a linear feedback shift register (LFSR) that is seeded by

the data tag to generate a plurality of the plurality blockwise independent bit vectors.

38. The system of claim 32 wherein the bit vector generation stage comprises a feedback counter that is seeded by the data tag and is configured to receive as a feedback input a previous blockwise independent bit vector to thereby generate a plurality of the plurality of blockwise independent bit vectors.

39. The system of claim 37 wherein the feedback counter is further configured to receive as an input a constant increment value therefor.

40. The system of claim 39 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, the system further comprising a data tag initialization stage that is configured to, prior to the reversible combinatorial operation stage processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

41. The system of claim 39 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, wherein the bit vector generation stage is further configured to, prior to the reversible combinatorial operation stage processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

42. The system of claim 32 wherein the bit vector generation stage comprises means for generating the plurality of bit vectors.

43. The system of claim 32 wherein the data tag comprises a file identifier associated with the data segment.

44. The system of claim 32 wherein the data tag comprises a physical memory address associated with the data segment.

45. The system of claim 32 wherein the data segment comprises network data traffic, and wherein the data tag comprises a packet sequence number associated with the data segment.

46. The system of claim 32 wherein the data segment has a plurality of associated data tags, wherein each data tag is uniquely associated with a different one of the data segment's data blocks, and wherein the bit vector generation stage is further configured to generate the plurality of blockwise independent bit vectors based at least in part upon the plurality of data tags.

47. The system of claim 32 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the bit vector generation stage is further configured to, prior to processing a data block of the subset, cycle through and discard any bit vectors corresponding to data blocks within the data segment that are not within the subset and that are prior to the data blocks within the subset.

48. The system of claim 27 further comprising:

a second reversible combinatorial operation stage configured to reversibly combine each encrypted data block-bit vector combination with a different one of the bit vectors to generate a plurality of encrypted data block-bit vector-bit vector combinations; and

a second encryption stage configured to perform an encryption operation on the encrypted data block-bit vector-bit vector combinations.

49. The system of claim 27 wherein each bit vector has a different value.

50. The system of claim 27 wherein the reversible combinatorial operation stage comprises XOR logic.

51. The system of claim 27 wherein the bit vector generation stage, the reversible combinatorial operation stage, and the block cipher key encryption stage are deployed in hardware.

52. The system of claim 51 wherein the hardware comprises a reconfigurable logic device.

53. The system of claim 52 wherein the bit vector generation stage, the reversible combinatorial operation stage, and the block cipher key encryption stage are deployed as a firmware pipeline on the reconfigurable logic device.

54. The system of claim 51 wherein the hardware comprises an application specific integrated circuit (ASIC).

55. The system of claim 51 wherein the hardware comprises part of a RAID controller.

56. The system of claim 27 further comprising a plurality of reversible combinatorial operation stages and a plurality of the encryption stages arranged in a plurality of parallel pipelines, each pipeline comprising a reversible combinatorial operation stage and an encryption stage, thereby allowing the data blocks to processed through the pipelines in parallel with each other.

57. The system of claim 56 wherein the bit vector generation stage is configured to provide the same bit vectors to each of the pipelines.

58. The system of claim 56 wherein each pipeline further comprises a bit vector generation stage.

59. A system for encrypting a stream of data blocks, each data block comprising a plurality of data bits, the system comprising:

a randomizer circuit for randomizing a plurality of the data blocks prior to encryption independently of the other data blocks; and

an encryptor circuit for encrypting each of the randomized data blocks.

60. The system of claim 59 wherein the randomizer circuit is further configured to randomize a plurality of the data blocks based at least in part upon a data tag associated with the data blocks.

61. The system of claim 60 wherein the randomizer circuit comprises a pseudo-random number generator that is seeded by the data tag, and wherein the randomizer circuit is further configured to generate a plurality of blockwise independent bit vectors for use to randomize a plurality of the data blocks based at least in part upon a plurality of pseudo-random number outputs of the pseudo-random number generator.

62. The system of claim 61 wherein the randomizer circuit further comprises an adder for adding a pseudo-random number output of the pseudo-random number generator with a blockwise independent bit vector that was used to randomize a previous data block to thereby generate a current blockwise independent bit vector for use to randomize a current data block.

63. The system of claim 61 wherein the randomizer circuit is further configured to output the pseudo-random number outputs as a plurality of the bit vectors.

64. The system of claim 60 wherein the randomizer circuit comprises a linear feedback shift register (LFSR) that is seeded by the data tag to generate a plurality blockwise independent bit vectors for use to randomize a plurality of the data blocks.

65. The system of claim 60 wherein the randomizer circuit comprises a feedback counter that is seeded by the data tag and is configured to receive as a feedback input a previous blockwise

independent bit vector to thereby generate a plurality of the plurality of blockwise independent bit vectors for use to randomize a plurality of the data blocks.

66. The system of claim 65 wherein the feedback counter is further configured to receive as an input a constant increment value therefor.

67. The system of claim 66 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, the system further comprising calculation circuit that is configured to, prior to the randomizer circuit processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

68. The system of claim 66 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, wherein the randomizer circuit is further configured to, prior to processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

69. The system of claim 60 wherein the randomizer circuit comprises means for generating a plurality of blockwise independent bit vectors for use to randomize a plurality of the data blocks.

70. A method of decrypting a plurality of encrypted data block-bit vector combinations of a data segment, the data segment comprising a plurality of encrypted data block-bit vector combinations, each encrypted data block-bit vector combination comprising a plurality of data bits, the method comprising:

decrypting a plurality of the encrypted data block-bit vector combinations using a decryption operation to thereby generate a plurality of decrypted data block-bit vector combinations; and

combining each decrypted data block-bit vector combination with a corresponding bit vector using a reversible combinatorial operation to thereby create a plurality of decrypted data blocks; and

wherein each bit vector has a value that is independent of all decryption operations that were previously performed on the encrypted data block-bit vector combinations for the data segment, and wherein each corresponding bit vector is the same bit vector that was reversibly combined with the data block from which the encrypted data block-bit vector combination was derived during encryption thereof.

71. The method of claim 70 further comprising:

generating each bit vector based at least in part upon a data tag associated with the data segment.

72. The method of claim 71 wherein the generating step further comprises (1) seeding a pseudo-random number generator with the data tag to generate a plurality of pseudo-random outputs, and (2) generating a plurality of the bit vectors based at least in part upon the pseudo-random outputs.

73. The method of claim 72 wherein the step of generating the bit vectors comprises incrementing the bit vector used for a previous data block of the data segment by one of the pseudo-random outputs to generate the bit vector to be combined with a next data block of the data segment.

74. The method of claim 72 wherein the step of generating the bit vectors corresponding to the data blocks based at least in part upon the pseudo-random outputs comprises using the pseudo-random outputs as a plurality of the bit vectors.

75. The method of claim 71 wherein the generating step further comprises seeding a linear feedback shift register with the data tag to generate a plurality of the bit vectors.

76. The method of claim 71 wherein the generating step further comprises seeding a feedback counter with the data tag to generate a plurality of the bit vectors.

77. The method of claim 76 wherein the seeding step further comprises providing a constant increment to the feedback counter.

78. The method of claim 77 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the generating step comprises, prior to processing a data block of the subset, (1) calculating a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seeding the feedback counter with the calculated data tag.

79. The method of claim 71 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the generating step comprises, prior to processing a data block of the subset, cycling through and discarding any bit vectors corresponding to data blocks within the data segment that are not within the subset and that are prior to the data blocks within the subset.

80. The method of claim 70 wherein the data segment comprises a sequence of encrypted data blocks, and wherein the plurality of encrypted data blocks decrypted during the decrypting step comprise any arbitrary subset of all of the encrypted data blocks of the data segment.

81. A system for decrypting a plurality of encrypted data block-bit vector combinations of a data segment, the data segment comprising a plurality of encrypted data block-bit vector combinations, each encrypted data block-bit vector combination comprising a plurality of data bits, the system comprising:

a bit vector generation stage configured to generate a plurality of blockwise independent bit vectors;

a decryption stage configured to perform a decryption operation on a plurality of the encrypted data block-bit vector combinations to thereby generate a plurality of decrypted data block-bit vector combinations; and

a reversible combinatorial operation stage configured to reversibly combine each decrypted data block-bit vector combination with a corresponding one of the bit vectors to generate a plurality of decrypted data blocks; and

wherein each corresponding bit vector is the same bit vector that was reversibly combined with the data block from which the encrypted data block-bit vector combination was derived during encryption thereof.

82. The system of claim 81 wherein the bit vector generation stage is further configured to generate the plurality of blockwise independent bit vectors based at least in part upon a data tag associated with the data segment.

83. The system of claim 82 wherein the bit vector generation stage comprises a pseudo-random number generator that is seeded by the data tag, and wherein the bit vector generation stage is further configured to generate a plurality of the bit vectors based at least in part upon a plurality of pseudo-random number outputs of the pseudo-random number generator.

84. The system of claim 83 wherein the bit vector generation stage further comprises an adder for adding a pseudo-random number output of the pseudo-random number generator with a blockwise independent bit vector that was reversibly combined with a previous data block to thereby generate a current blockwise independent bit vector for reversible combination with a current data block.

85. The system of claim 83 wherein the bit vector generation stage is further configured to output the pseudo-random number outputs as a plurality of the bit vectors.

86. The system of claim 82 wherein the bit vector generation stage comprises a linear feedback shift register (LFSR) that is seeded by the data tag to generate a plurality of the plurality blockwise independent bit vectors.

87. The system of claim 82 wherein the bit vector generation stage comprises a feedback counter that is seeded by the data tag and is configured to receive as a feedback input a previous blockwise independent bit vector to thereby generate a plurality of the plurality of blockwise independent bit vectors.

88. The system of claim 87 wherein the feedback counter is further configured to receive as an input a constant increment value therefor.

89. The system of claim 88 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, the system further comprising a data tag initialization stage that is configured to, prior to the reversible combinatorial operation stage processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

90. The system of claim 88 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, wherein the bit vector generation stage is further configured to, prior to the reversible combinatorial operation stage processing a data block of the subset, (1) calculate a data tag value such that the data tag is uniquely associated with the data block of the subset, and (2) seed the feedback counter with the calculated data tag.

91. The system of claim 82 wherein the bit vector generation stage comprises means for generating the plurality of bit vectors.

92. The system of claim 82 wherein the plurality of data blocks to be encrypted comprises a subset of the data blocks within the data segment, and wherein the bit vector generation stage is further configured to, prior to processing a data block of the subset, cycle through and discard any bit vectors corresponding to data blocks within the data segment that are not within the subset and that are prior to the data blocks within the subset.

93. The system of claim 81 wherein the decryption stage is further configured to perform the decryption operation on any arbitrary subset of the encrypted data block-bit vector combinations without requiring decryption of any encrypted data block-bit vector combinations that is not a member of the arbitrary subset.

94. The system of claim 82 further comprising a plurality of reversible combinatorial operation stages and a plurality of the decryption stages arranged in a plurality of parallel pipelines, each pipeline comprising a reversible combinatorial operation stage and a decryption stage, thereby allowing the data blocks to processed through the pipelines in parallel with each other.

95. The system of claim 94 wherein the pipelines are deployed on at least one of the group consisting of a reconfigurable logic device and an application specific integrated circuit (ASIC).

96. A method for symmetrically encrypting/decrypting a plurality of data blocks of a data segment, each data block comprising a plurality of data bits, the method comprising:

receiving a plurality of the data blocks of the data segment;

reversibly combining each of the received data blocks with a corresponding bit vector to thereby generate a plurality of data block-bit vector combinations;

performing a block cipher operation on each of the data block-bit vector combinations to thereby generate a plurality of block ciphered data block-bit vector combinations; and

reversibly combining each of the block ciphered data block-bit vector combinations with a corresponding bit vector to thereby generate a plurality of encrypted/decrypted data blocks; and

wherein the bit vectors possess values that are independent of the block cipher operations performed on the different data block-bit vector combinations.

97. The method of claim 96 further comprising:

generating the bit vectors based at least in part upon a data tag associated with the data segment.

98. The method of claim 97 wherein the step of reversibly combining each of the block ciphered data block-bit vector combinations with a corresponding bit vector comprises reversibly combining each block ciphered data block-bit vector combination with the same bit vector value that was reversibly combined with the data block from which that block ciphered data block-bit vector combination was generated.

99. The method of claim 98 further comprising:

synchronizing a delivery of the bit vectors to the step of reversibly combining each of the block ciphered data block-bit vector combinations with a corresponding bit vector to account for a latency of the block cipher operation performing step.

100. The method of claim 96 wherein the receiving step comprises receiving only a defined subset of the data blocks of the data segment.

101. A system for symmetrically encrypting/decrypting a plurality of data blocks of a data segment, each data block comprising a plurality of data bits, the system comprising:

a first reversible combinatorial operation circuit that is configured to receive a plurality of the data blocks of the data segment and reversibly combine each of the received data blocks with a corresponding bit vector to thereby generate a plurality of data block-bit vector combinations;

a block cipher operation circuit in communication with the first reversible combination operation circuit, wherein the block cipher operation circuit is configured to perform a block cipher operation on each of the data block-bit vector combinations to thereby generate a plurality of block ciphered data block-bit vector combinations; and

a second reversible combinatorial operation circuit in communication with the block cipher operation circuit, wherein the second reversible combinatorial operation circuit is configured to reversibly combine each of the block ciphered data block-bit vector combinations with a corresponding bit vector to thereby generate a plurality of encrypted/decrypted data blocks; and

wherein the bit vectors possess values that are independent of the block cipher operations performed on the different data block-bit vector combinations.

102. The system of claim 101 further comprising:

a bit vector generation circuit for generating the bit vectors based at least in part upon a data tag associated with the data segment.

103. The system of claim 102 wherein the second reversible combinatorial operation circuit is further configured to reversibly combine each block ciphered data block-bit vector combination with the same bit vector value that was reversibly combined with the data block from which that block ciphered data block-bit vector combination was generated.

104. The system of claim 103 wherein the bit vector generation circuit is further configured to synchronize a delivery of the bit vectors to the second reversible combinatorial operation circuit to account for a latency of the block cipher operation circuit.

105. The system of claim 102 wherein the bit vector generation circuit comprises means for generating the bit vectors.

106. The system of claim 101 wherein first reversible combinatorial operation circuit is further configured to receive only a defined subset of the data blocks of the data segment.

107. A method of encrypting a data segment, the data segment comprising a plurality of data blocks, each data block comprising a plurality of data bits, the method comprising:

encrypting a first plurality of data blocks of the data segment by (1) generating a first plurality of bit vectors, (2) reversibly combining each of the first plurality of data blocks with a corresponding one of the first plurality of bit vectors to thereby generate a first plurality of data block-bit vector combinations, and (3) performing an encryption operation on the first plurality of data block-bit vector combinations, wherein the first plurality of bit vectors have values that are independent of the encryption operations performed on the first plurality of data blocks; and

encrypting a second plurality of data blocks of the data segment by (1) generating a second plurality of bit vectors based at least in part upon a value of one of the first plurality of encrypted data block-bit vector combinations, (2) reversibly combining each of the second plurality of data blocks with a corresponding one of the second plurality of bit vectors to thereby generate a second plurality of data block-bit vector combinations, and (3) performing an encryption operation on the second plurality of data block-bit vector combinations.

108. The method of claim 107 wherein the step of generating a second plurality of bit vectors comprises generating the second plurality of bit vectors based at least in part upon a value of a first one of the first plurality of encrypted data block-bit vector combinations, wherein each encryption operation requires a plurality m of pipeline stages, wherein a feedback stride defines a first one of the second plurality of data blocks, and wherein the feedback stride has a value greater than or equal to the value of m.

109. The method of claim 107 wherein the step of generating the first plurality of bit vectors comprises generating the first

plurality of bit vectors based at least in part upon a data tag associated with the data segment.

110. The method of claim 109 wherein the step of generating the first plurality of bit vectors further comprises randomizing the values of the first plurality of bit vectors.

111. A method of encrypting a data segment, said data segment comprising a plurality of data block groups, each of said data block groups comprising a plurality of data blocks, the method comprising:

encrypting the data blocks of a first data block group based at least in part upon a first plurality of bit vectors that are combined with the data blocks of the first data block group, wherein the bit vectors of the first plurality of bit vectors comprise blockwise independent bit vectors;

creating a second plurality of bit vectors based at least at part on one of the previously encrypted data blocks; and

encrypting the data blocks of a data block group after the first data block group based at least in part upon the second plurality of bit vectors that are combined with the data blocks of the data block group after the first data block group.

112. A system for encrypting a plurality of data block groups, at least some of said data block groups comprising a plurality of data blocks, said system comprising:

an encryptor circuit; and

a sequence generator circuit configured to generate at least two types of bit vectors for use by the encryptor circuit, a first type of bit vector being blockwise independent and a second type of bit vector being dependent on at least one encrypted data block of a prior data block group;

wherein the encryptor circuit is configured to (1) reversibly combine the data blocks of at least one of the data groups comprising a plurality of data blocks with a plurality of bit vectors of the first type to thereby generate a first plurality of data block-bit vector combinations, and (2) encrypt the first plurality of data block-bit vector combinations.

113. The system of claim 112 wherein the encryptor circuit is further configured to (1) reversibly combine the data blocks of a different one of the data groups with a plurality of bit vectors of the second type to thereby generate a second plurality of data block-bit vector combinations, and (2) encrypt the second plurality of data block-bit vector combinations.

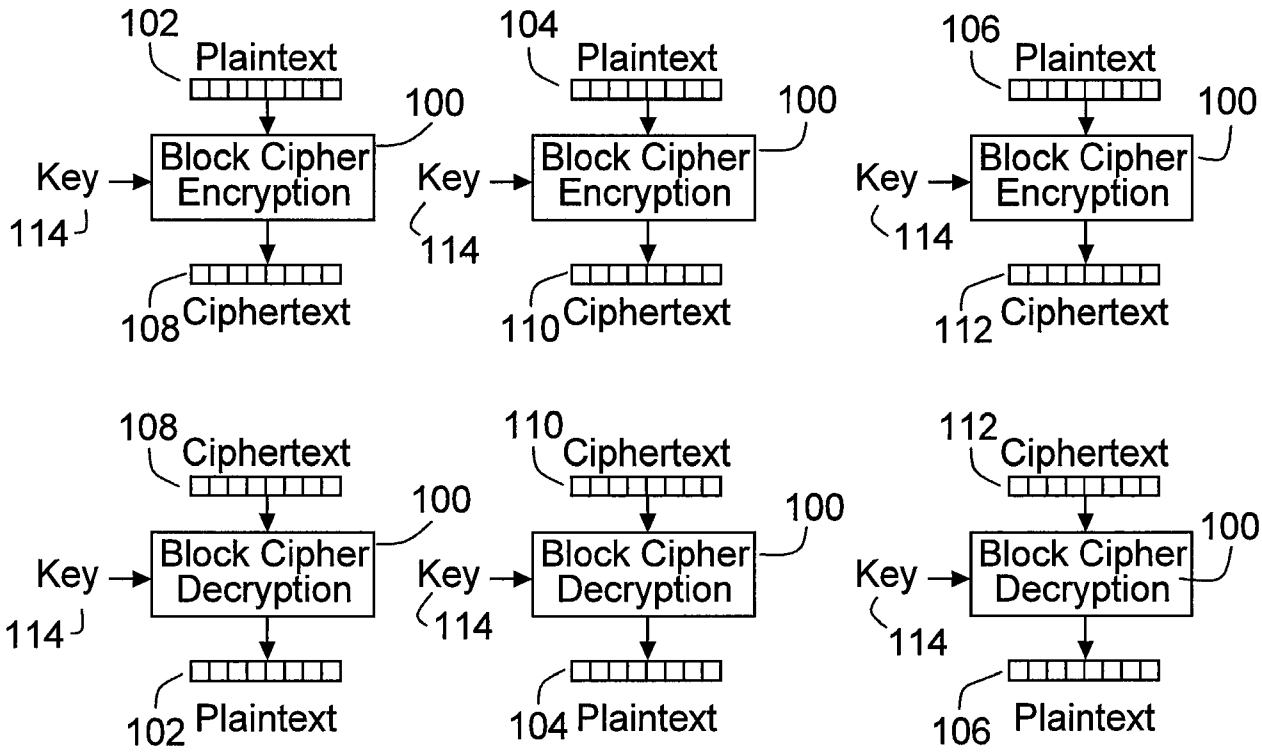


Figure 1 PRIOR ART

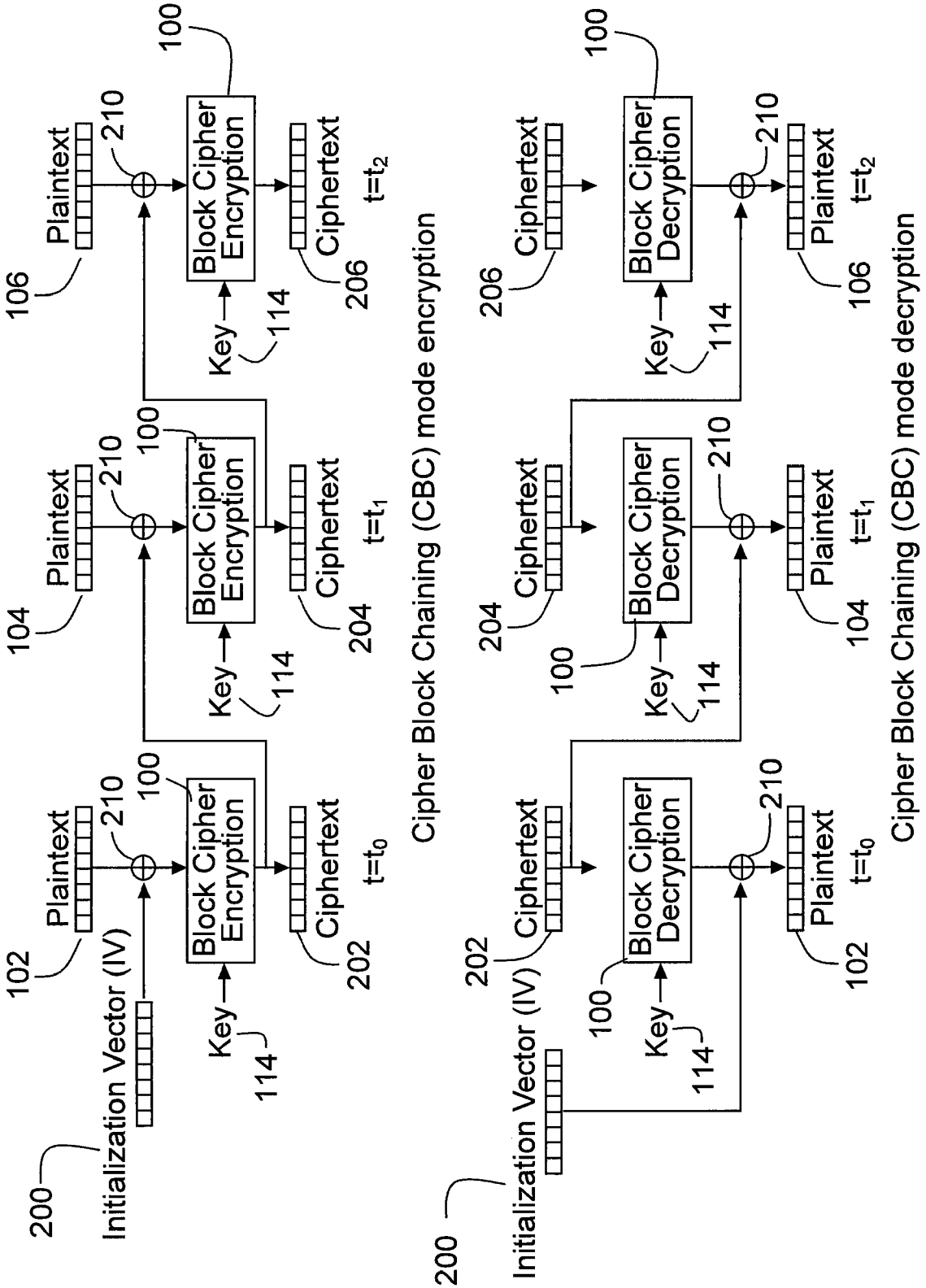


Figure 2 PRIOR ART

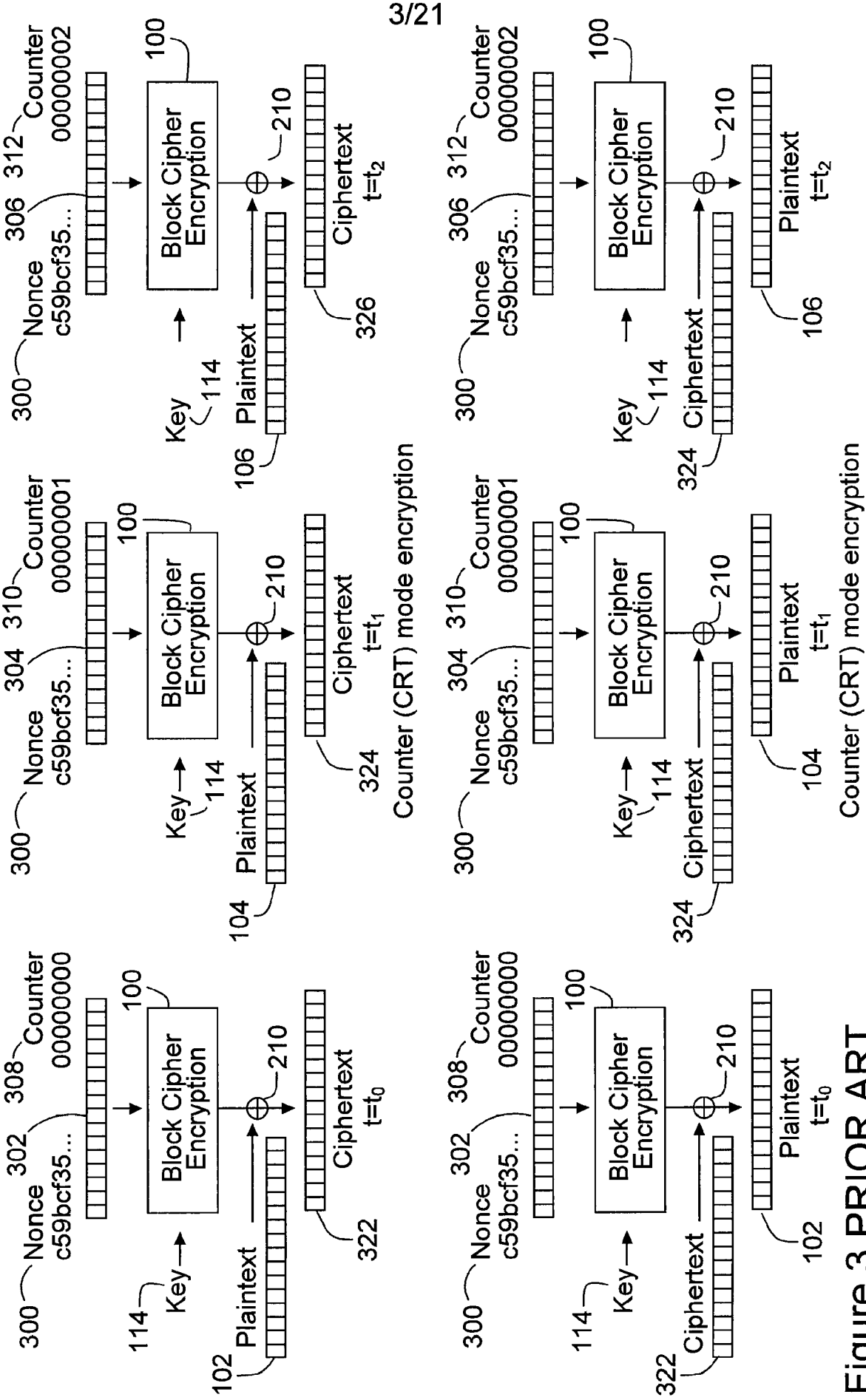


Figure 3 PRIOR ART

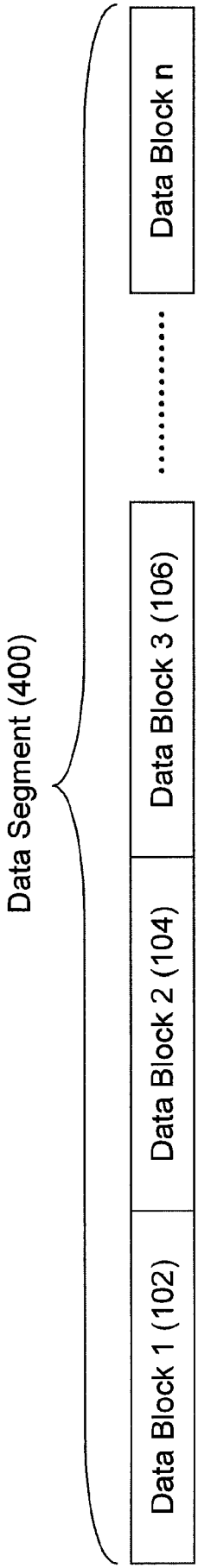


Figure 4

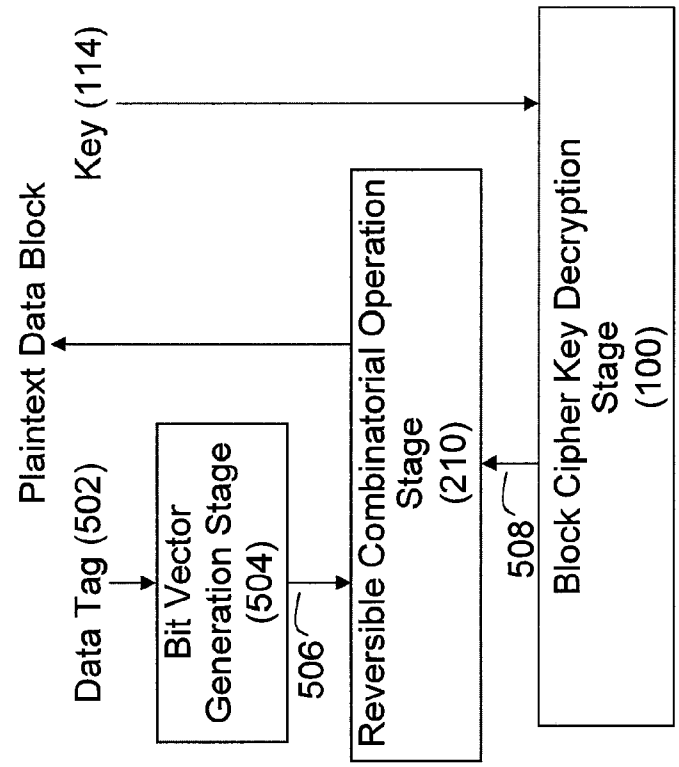


Figure 5(b)

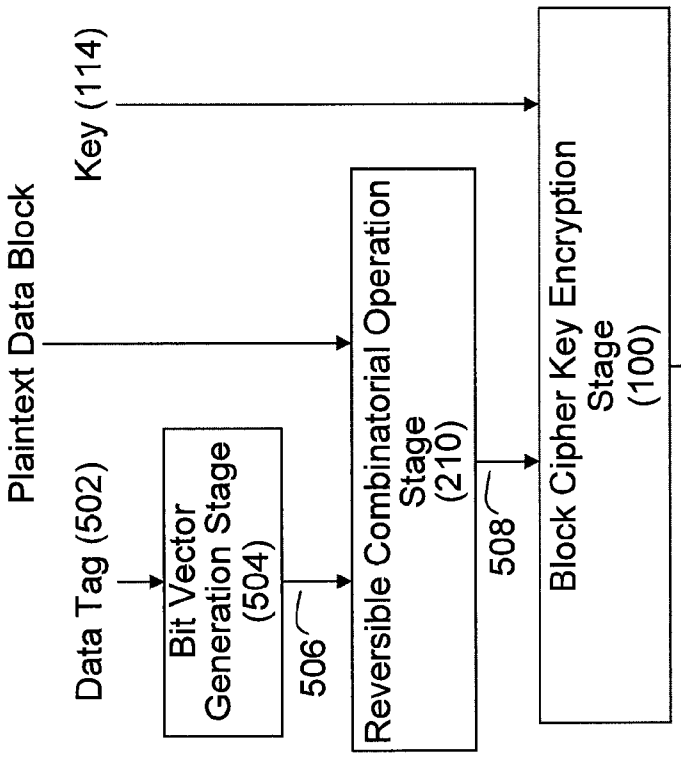


Figure 5(a)

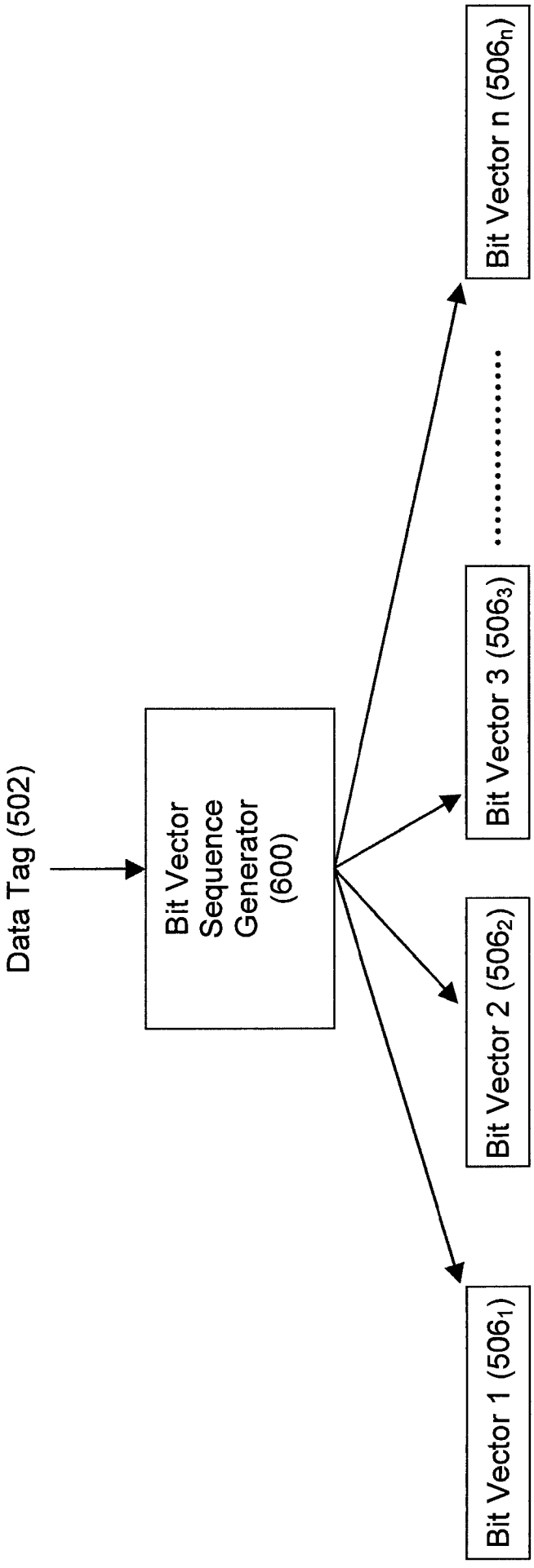

Data Tag (502)
Bit Vector Sequence Generator (600)
Bit Vector 1 (506$_1$)
Bit Vector 2 (506$_2$)
Bit Vector 3 (506$_3$)
Bit Vector n (506$_n$)

Figure 6

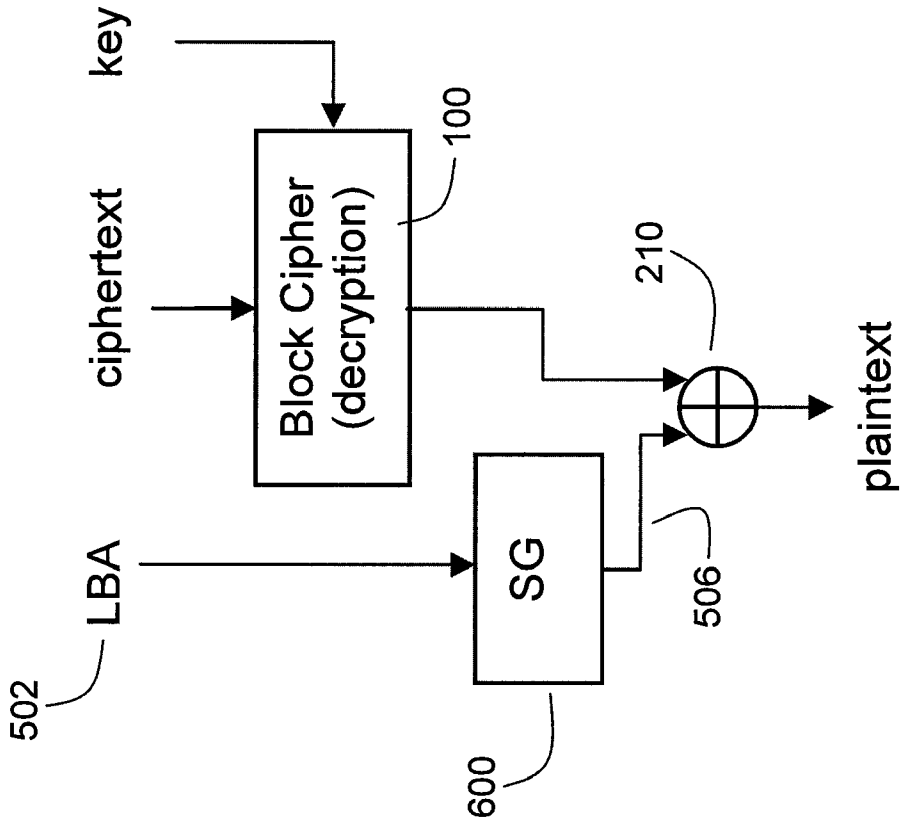


Figure 7(b)

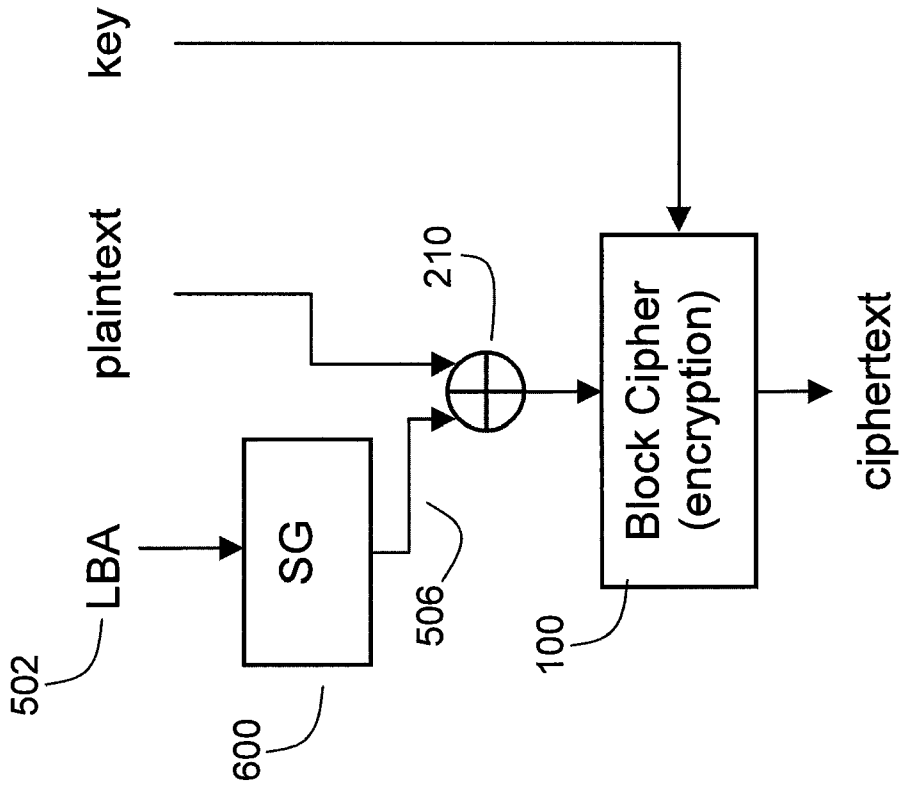
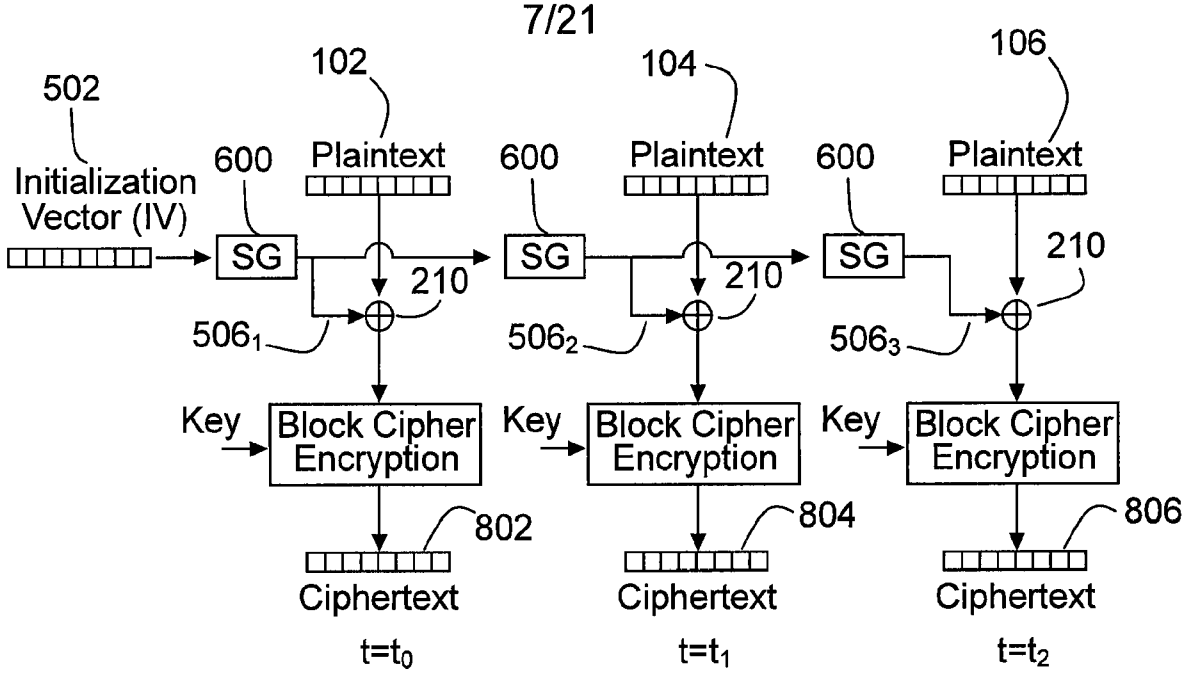
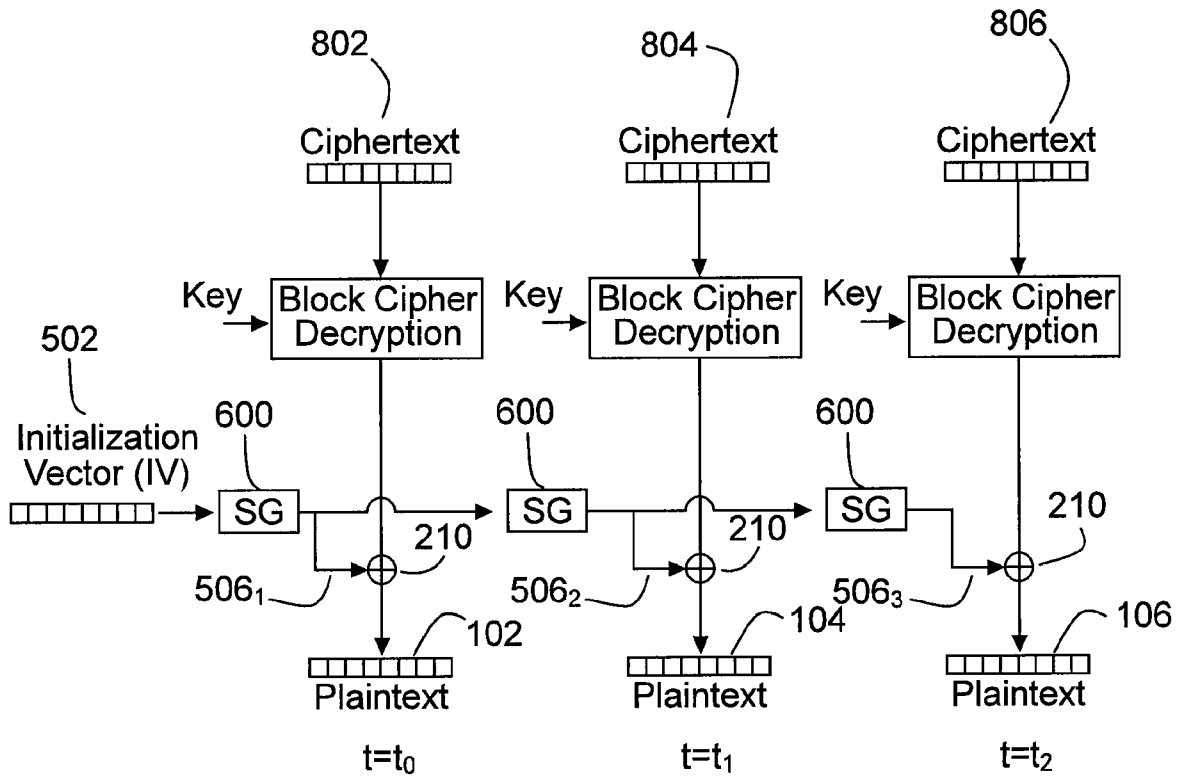


Figure 7(a)



IBR encryption figure

Figure 8(a)



IBR decryption figure

Figure 8(b)

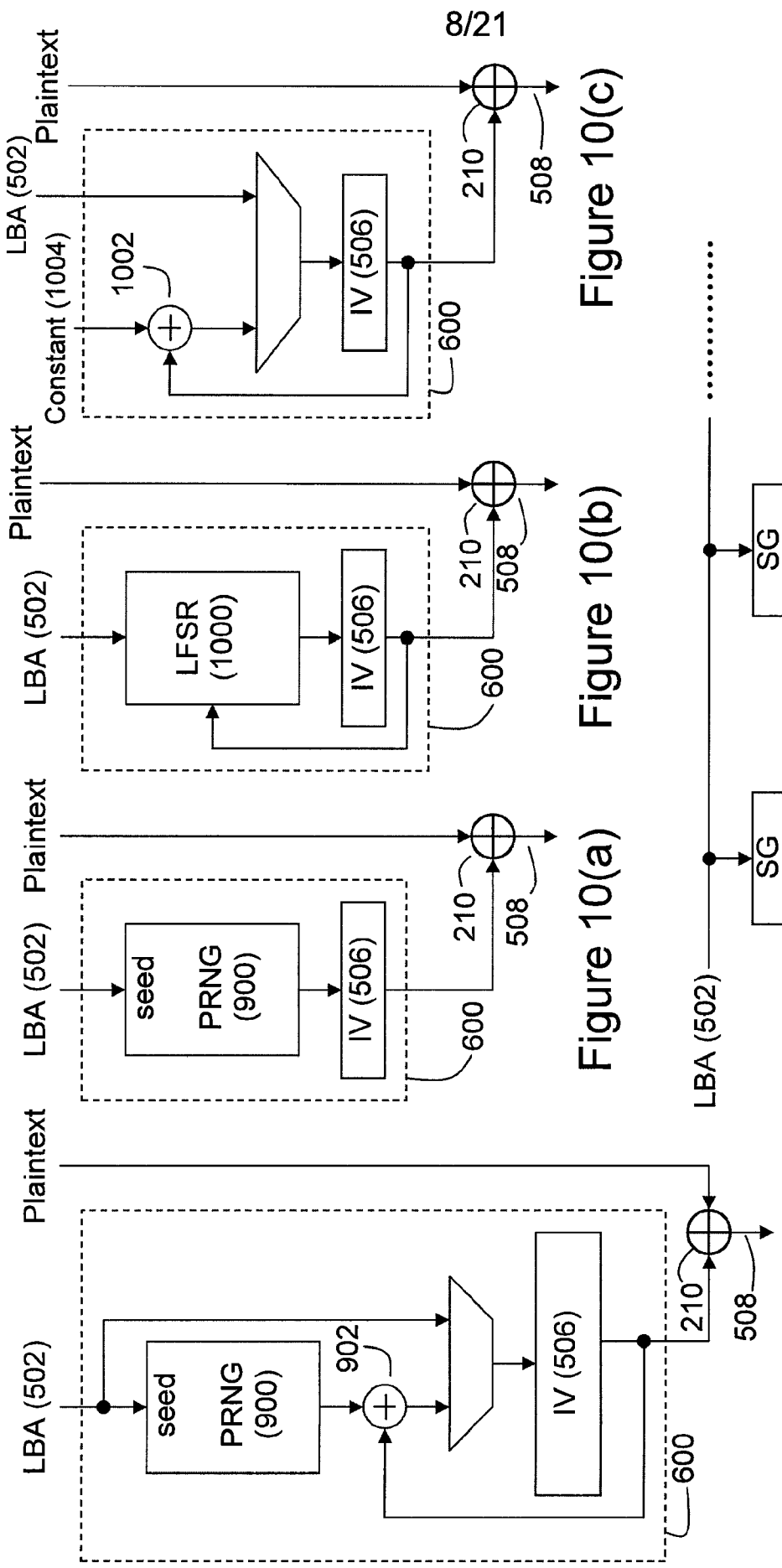


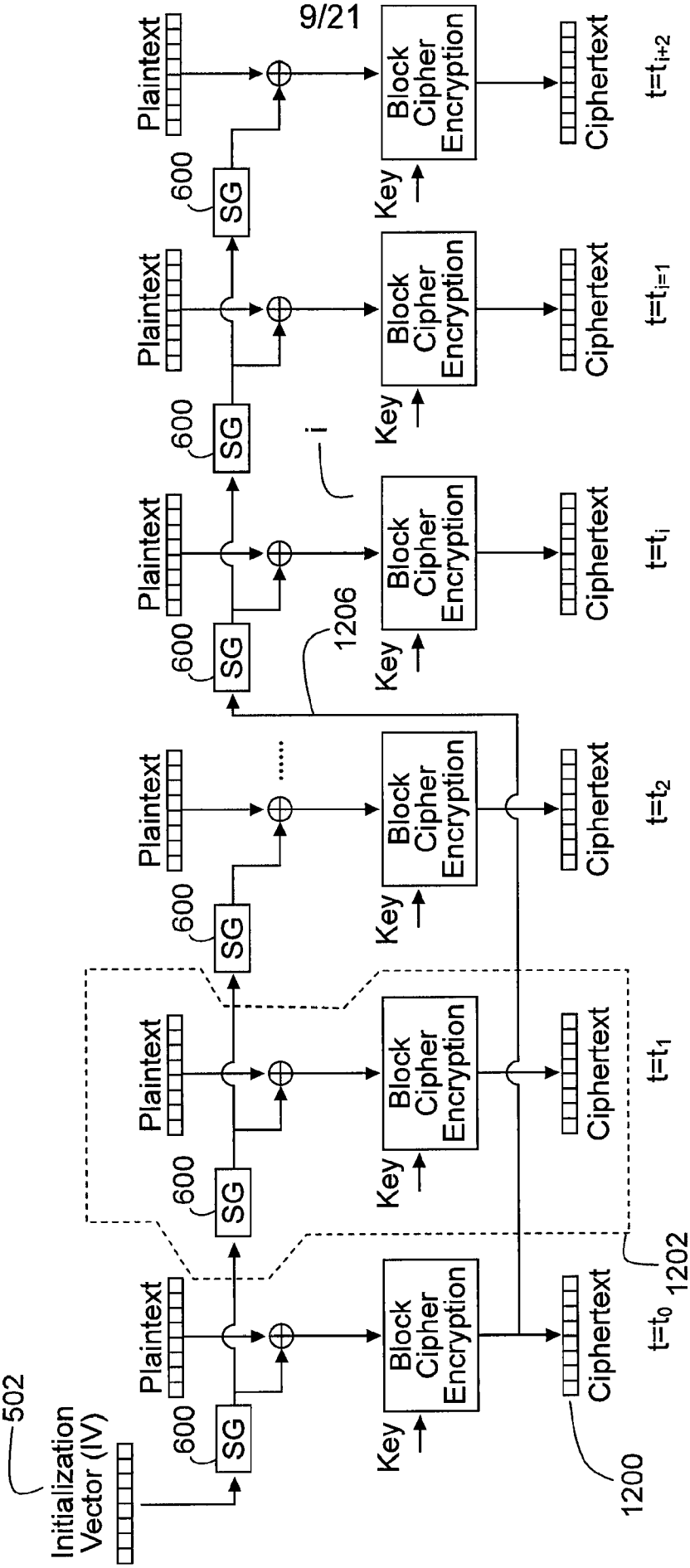
Figure 9

Figure 10(a)

Figure 10(b)

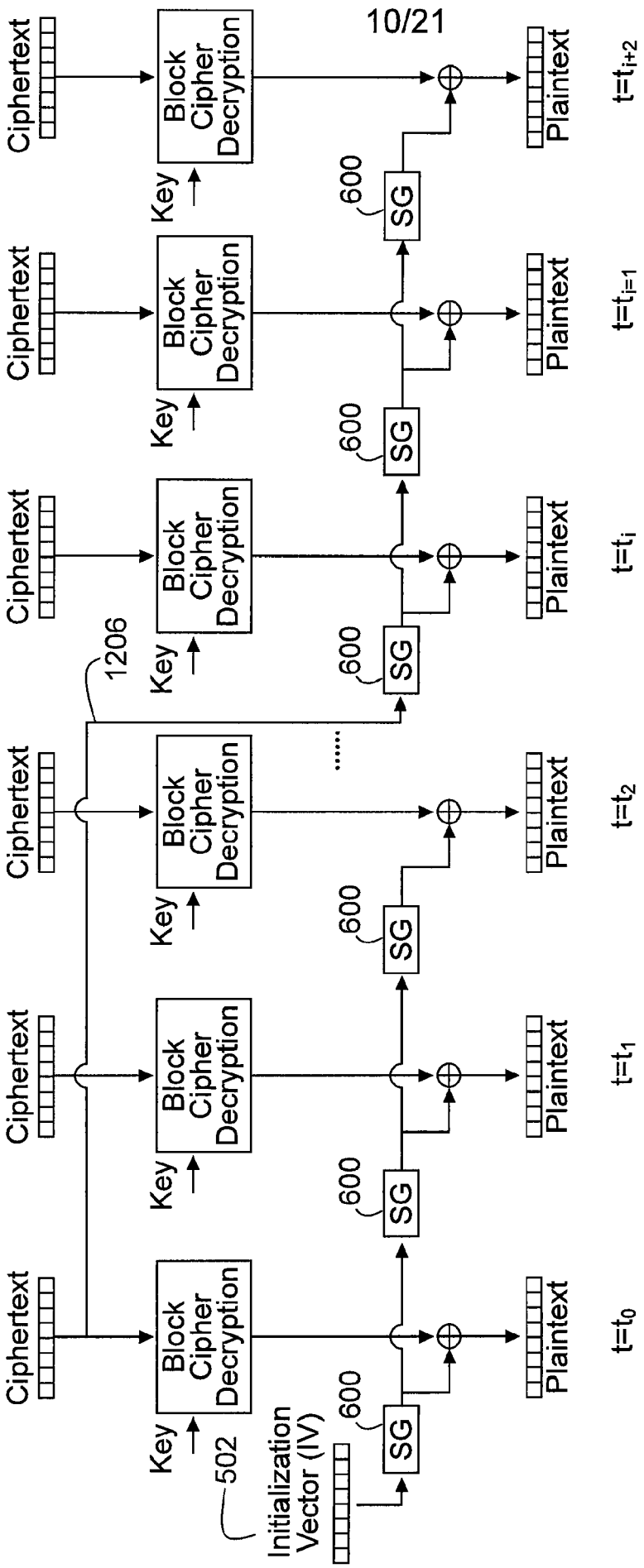
Figure 10(c)

Figure 11



IBR + CBC encryption figure

Figure 12(a)



IBR + CBC decryption figure

Figure 12(b)

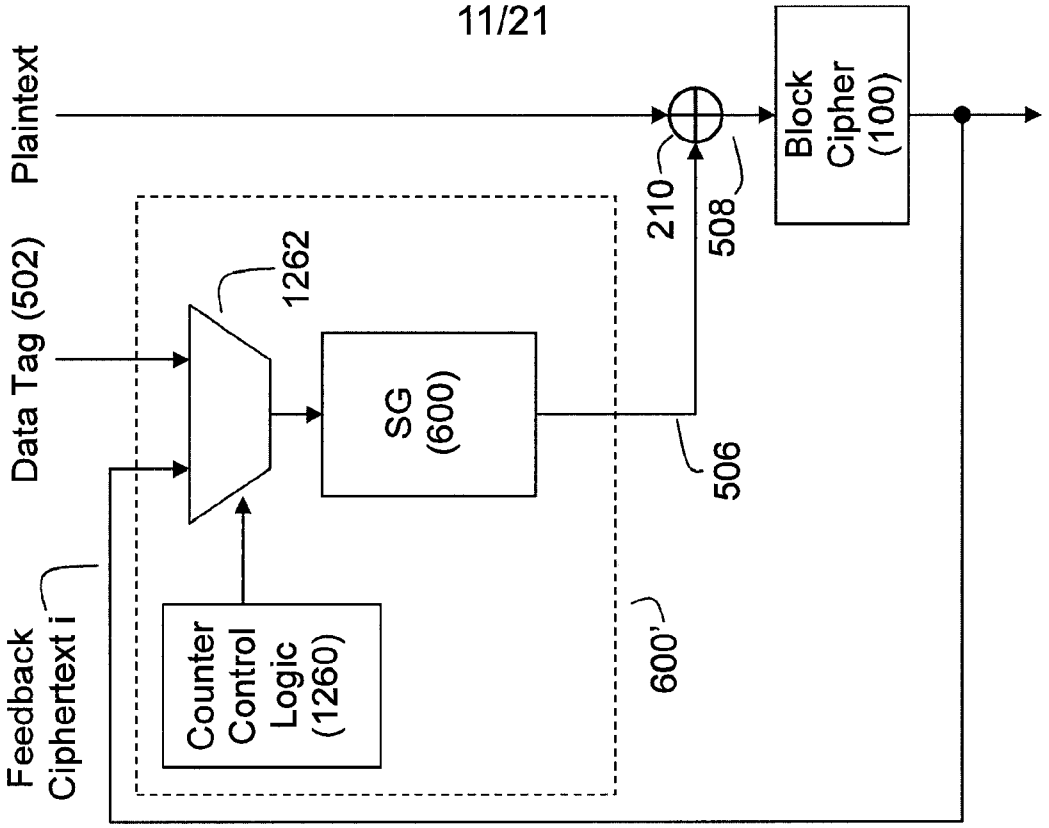


Figure 12(d)

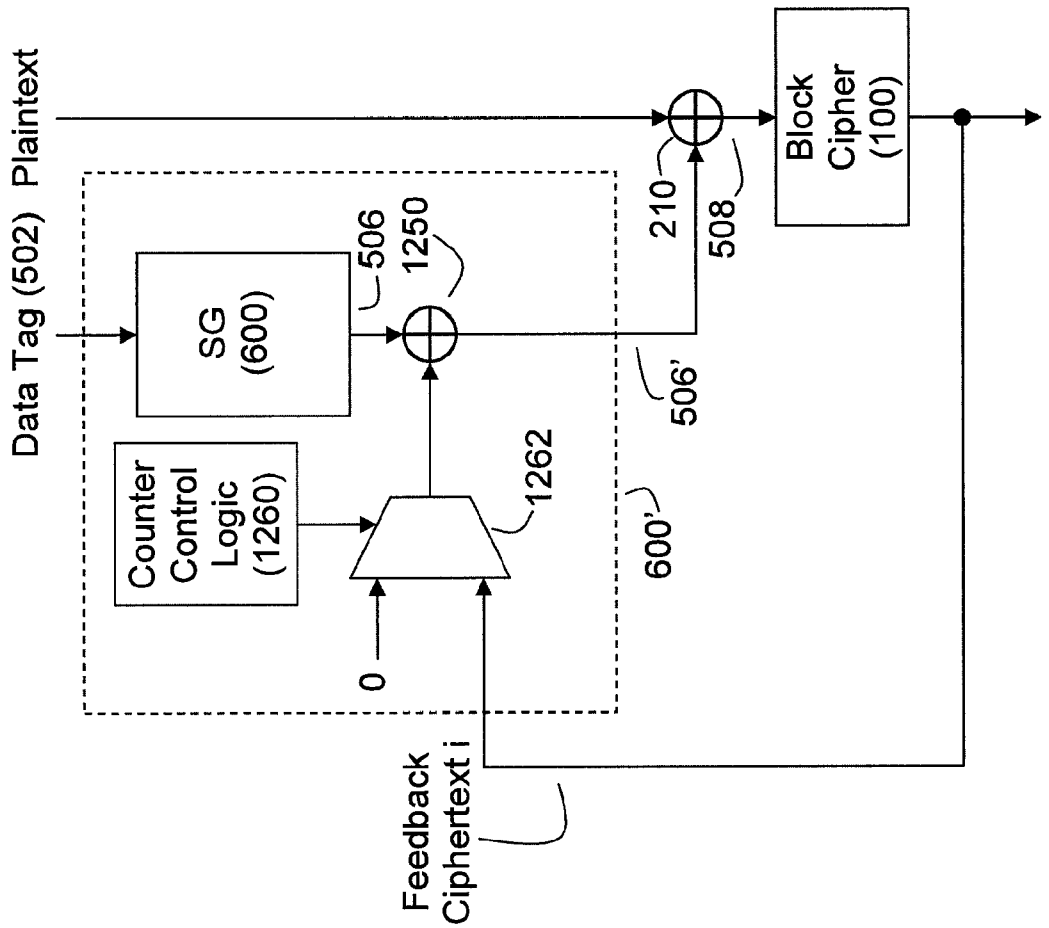


Figure 12(c)

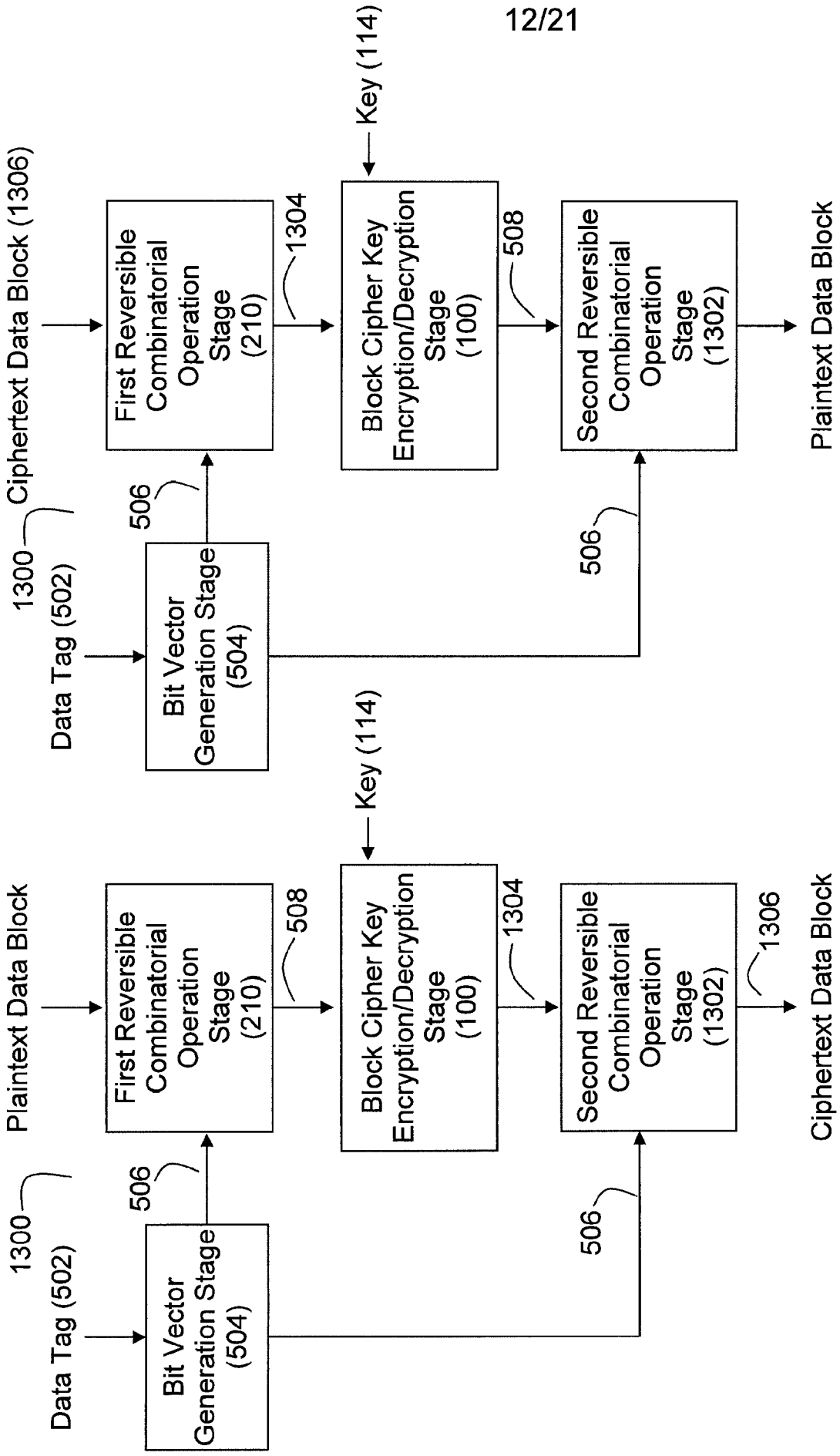


Figure 13(a)

Figure 13(b)

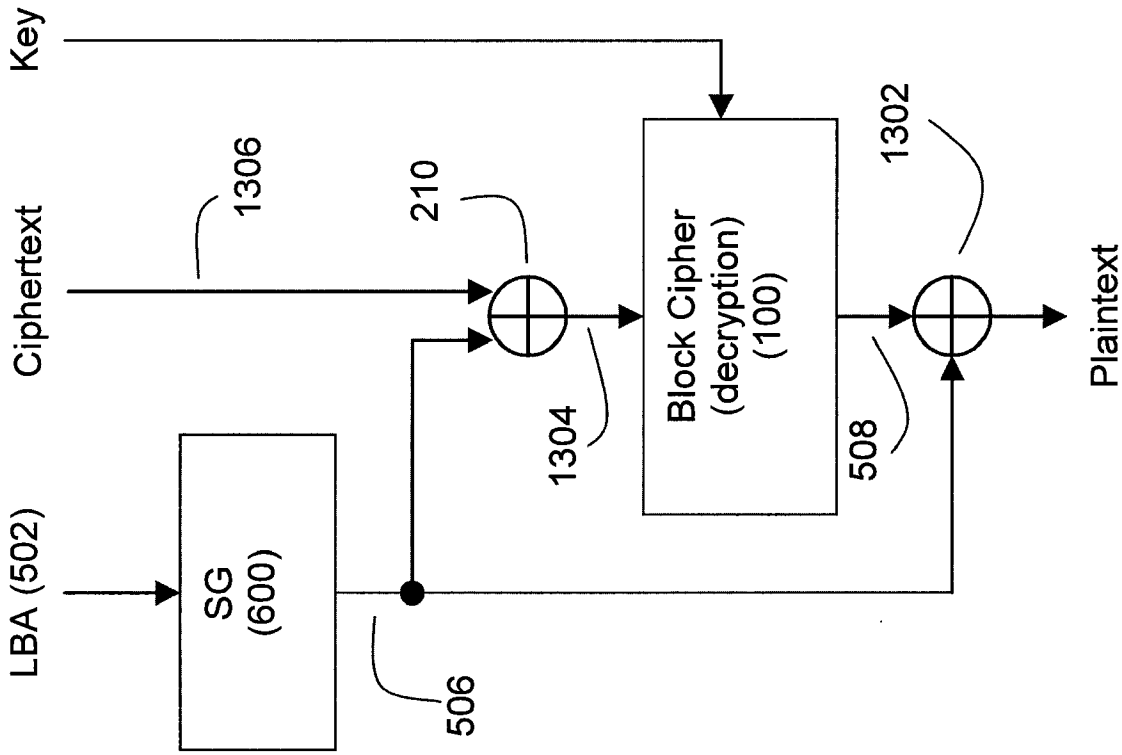


Figure 14(b)

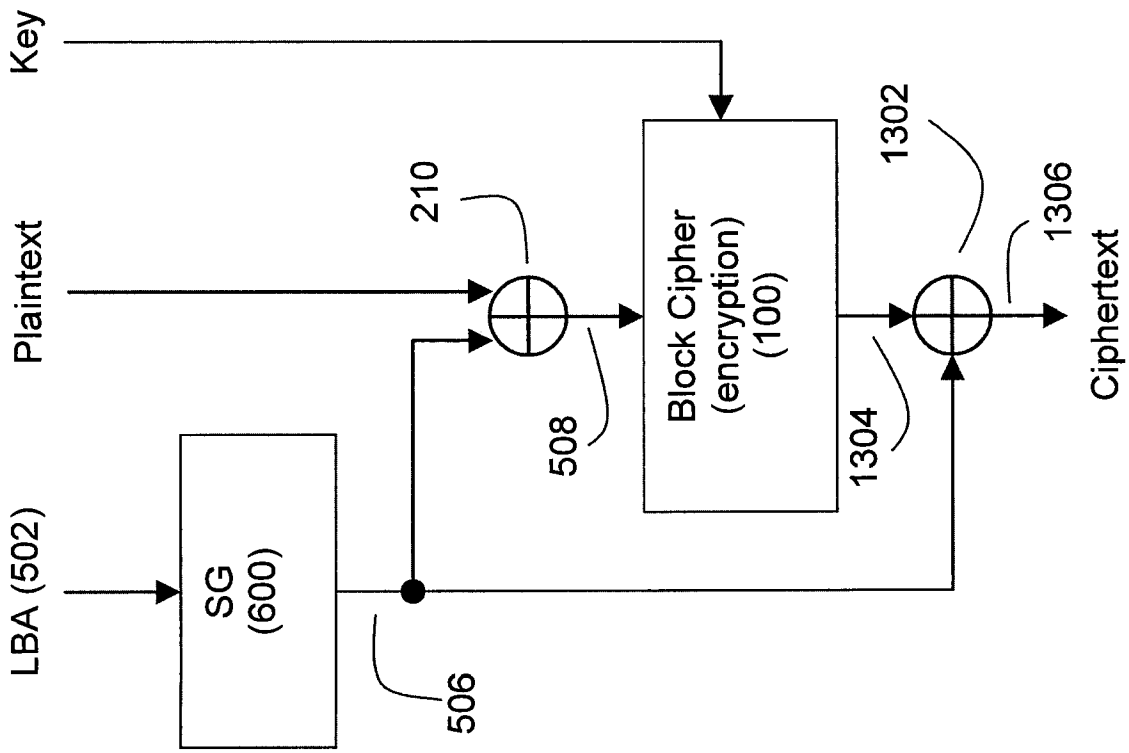


Figure 14(a)

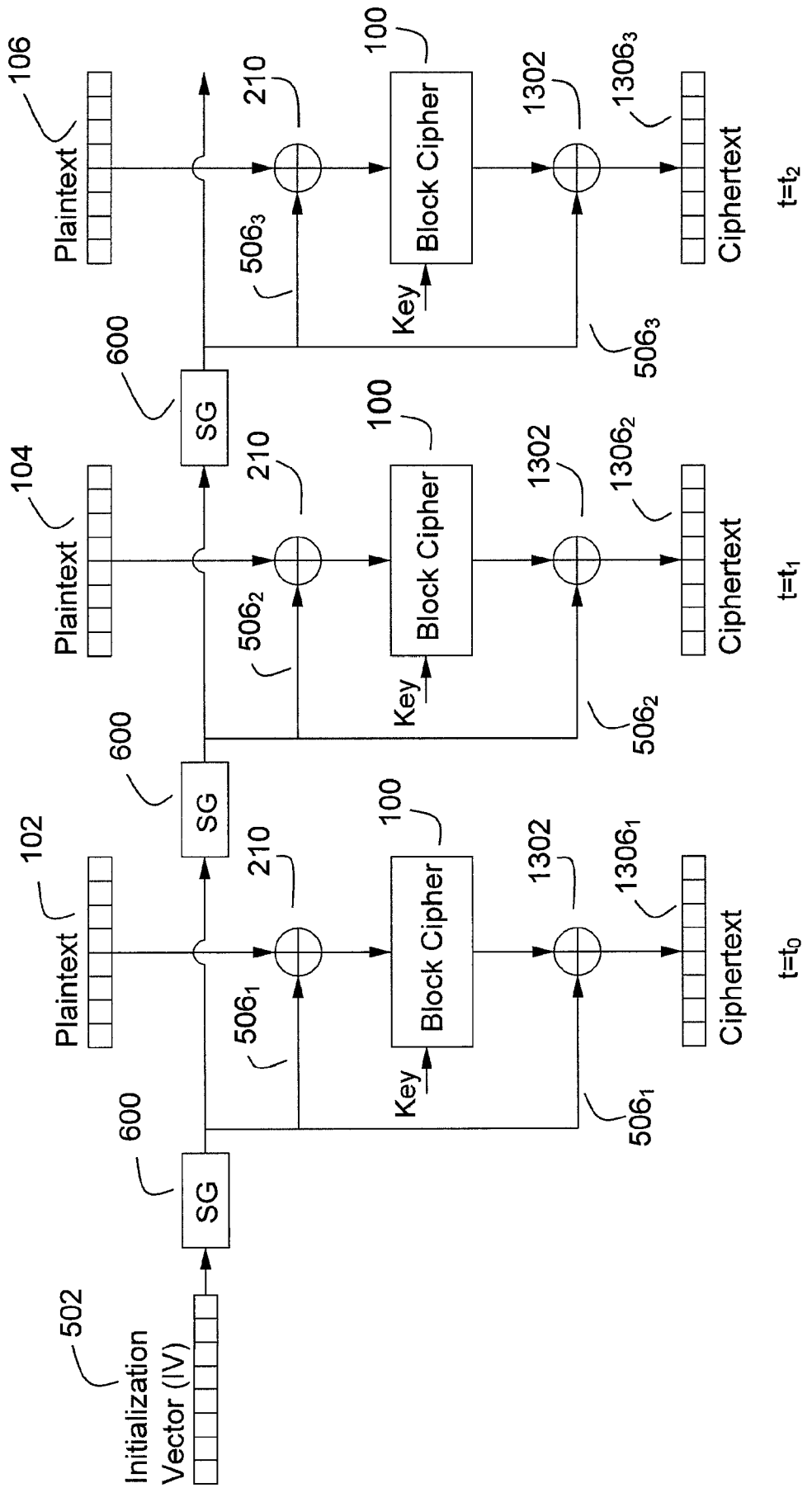

502
Initialization Vector (IV)
600
SG
102
Plaintext
210
100
Block Cipher
Key
1302
1306₁
Ciphertext
t=t₀
506₁
104
Plaintext
210
100
Block Cipher
Key
1302
1306₂
Ciphertext
t=t₁
506₂
106
Plaintext
210
100
Block Cipher
Key
1302
1306₃
Ciphertext
t=t₂
506₃

Figure 15(a)

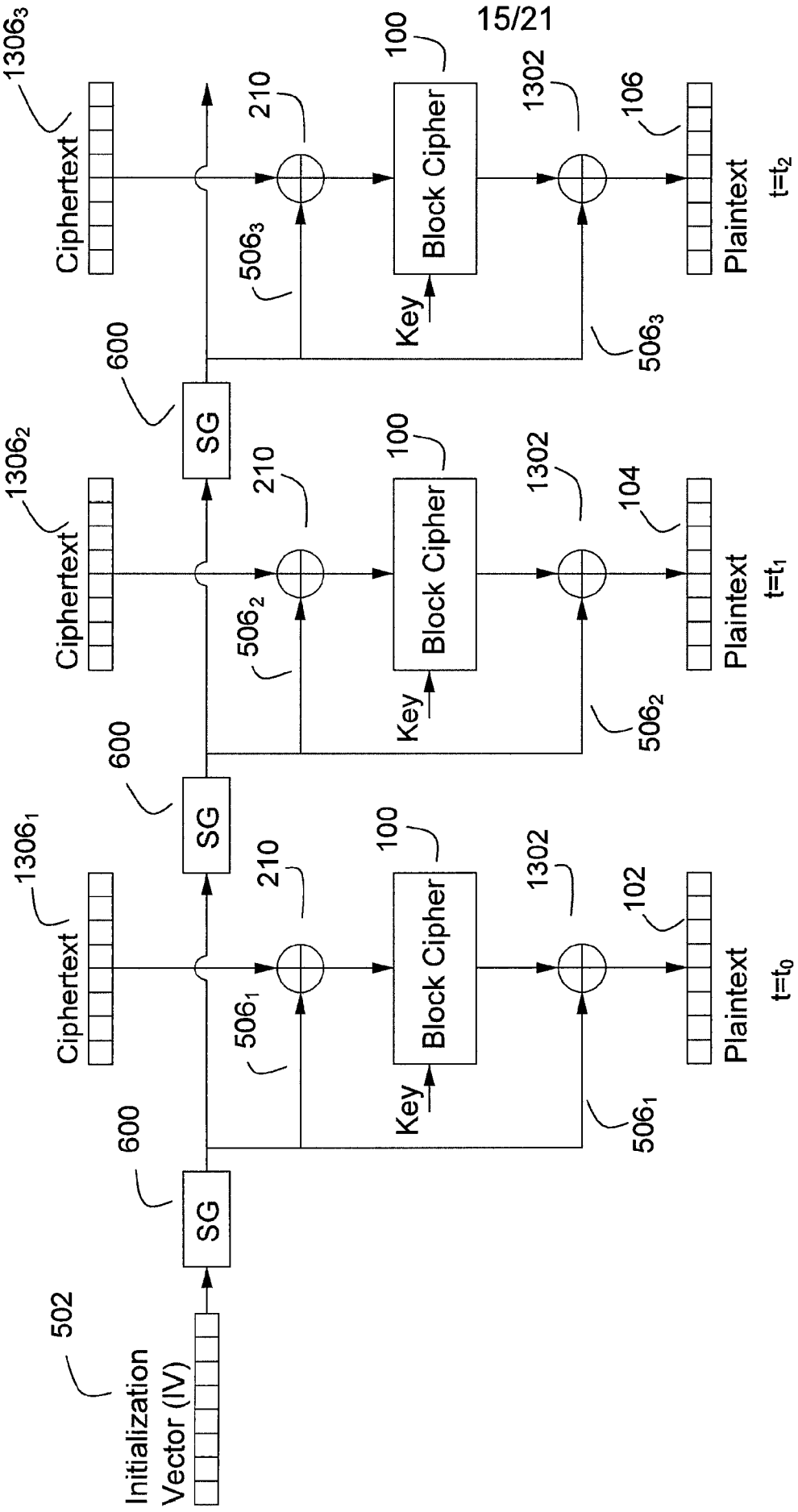
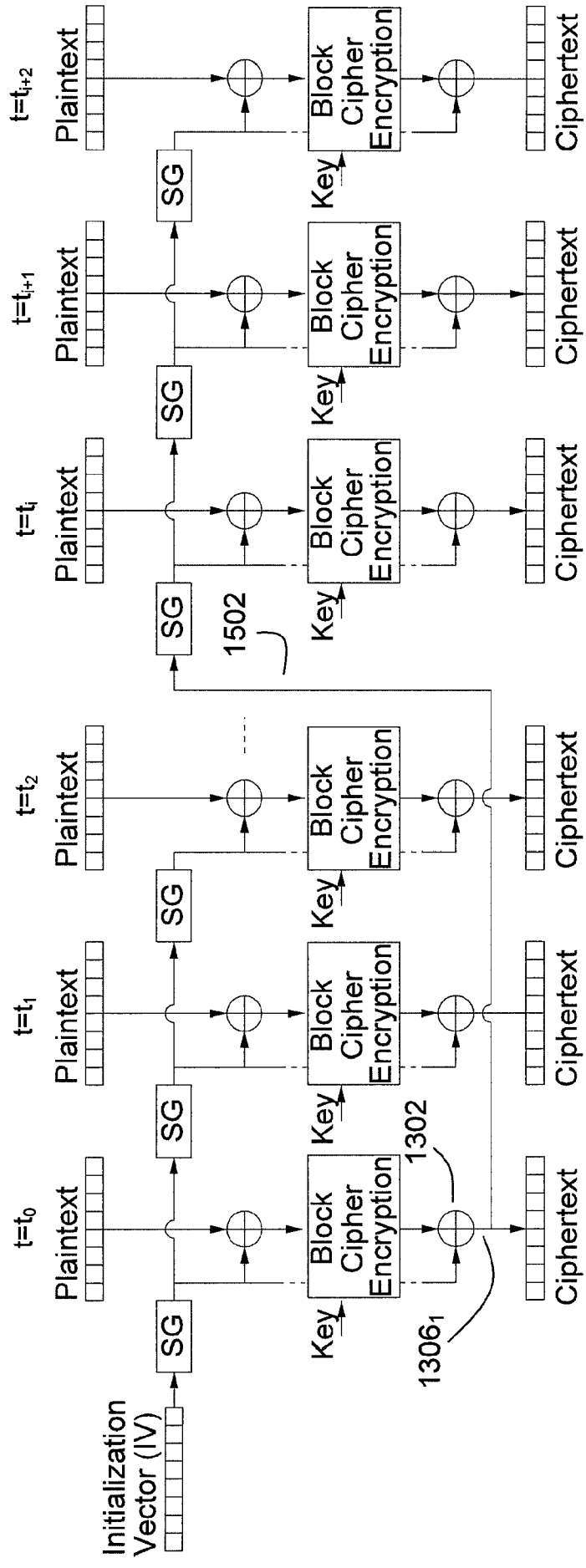
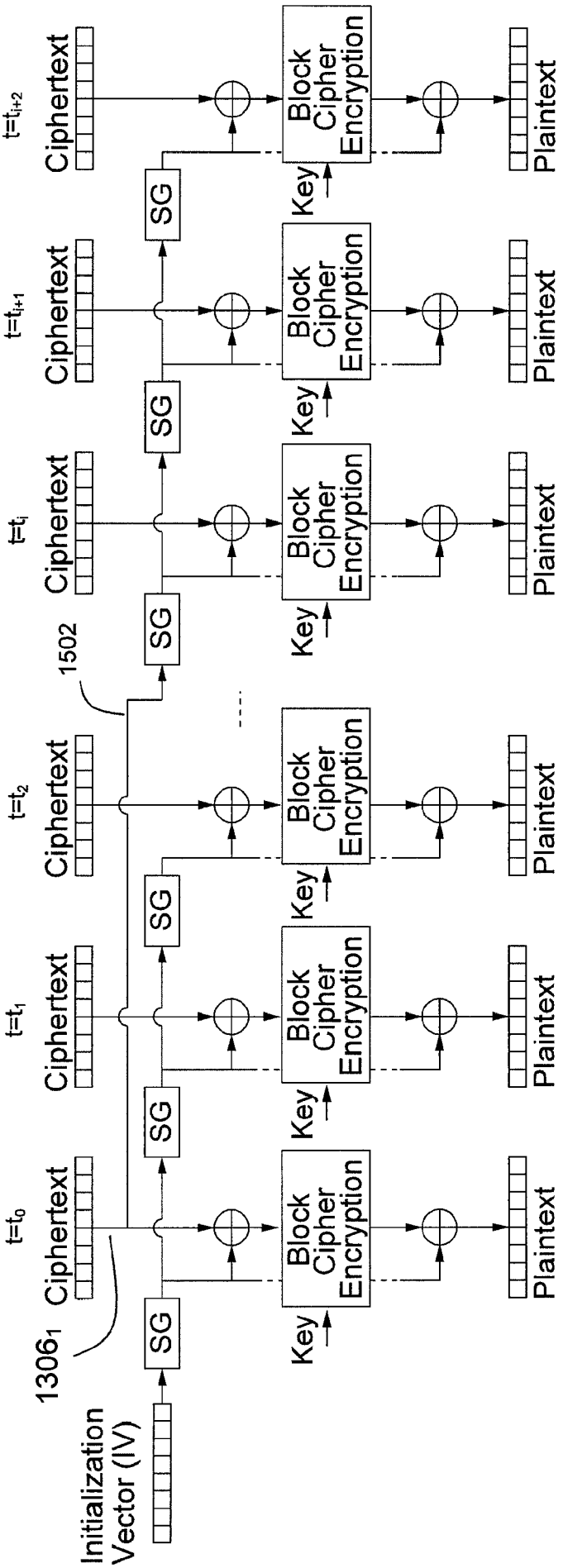

502
Initialization Vector (IV)
600
SG
1306$_1$
Ciphertext
210
506$_1$
100
Block Cipher
Key
1302
506$_1$
102
Plaintext
t=t$_0$
600
SG
1306$_2$
Ciphertext
210
506$_2$
100
Block Cipher
Key
1302
506$_2$
104
Plaintext
t=t$_1$
600
SG
1306$_3$
Ciphertext
210
506$_3$
100
Block Cipher
Key
1302
506$_3$
106
Plaintext
t=t$_2$

Figure 15(b)



Symmetric IBR + CBC encryption figure

Figure 15(c)



Symmetric IBR +CBC decryption

Figure 15(d)

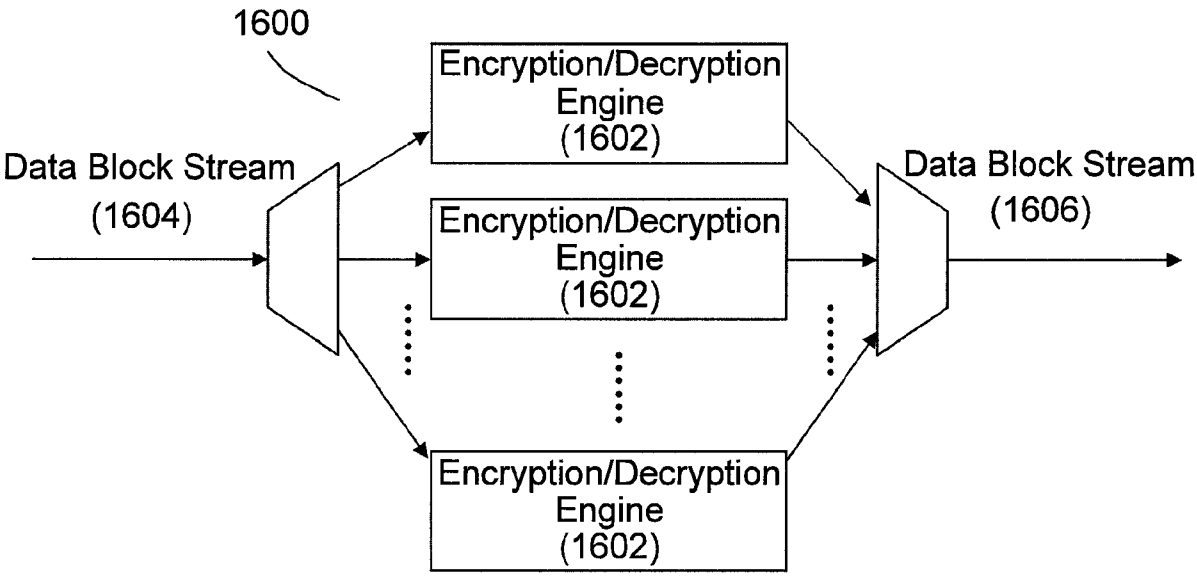


Figure 16

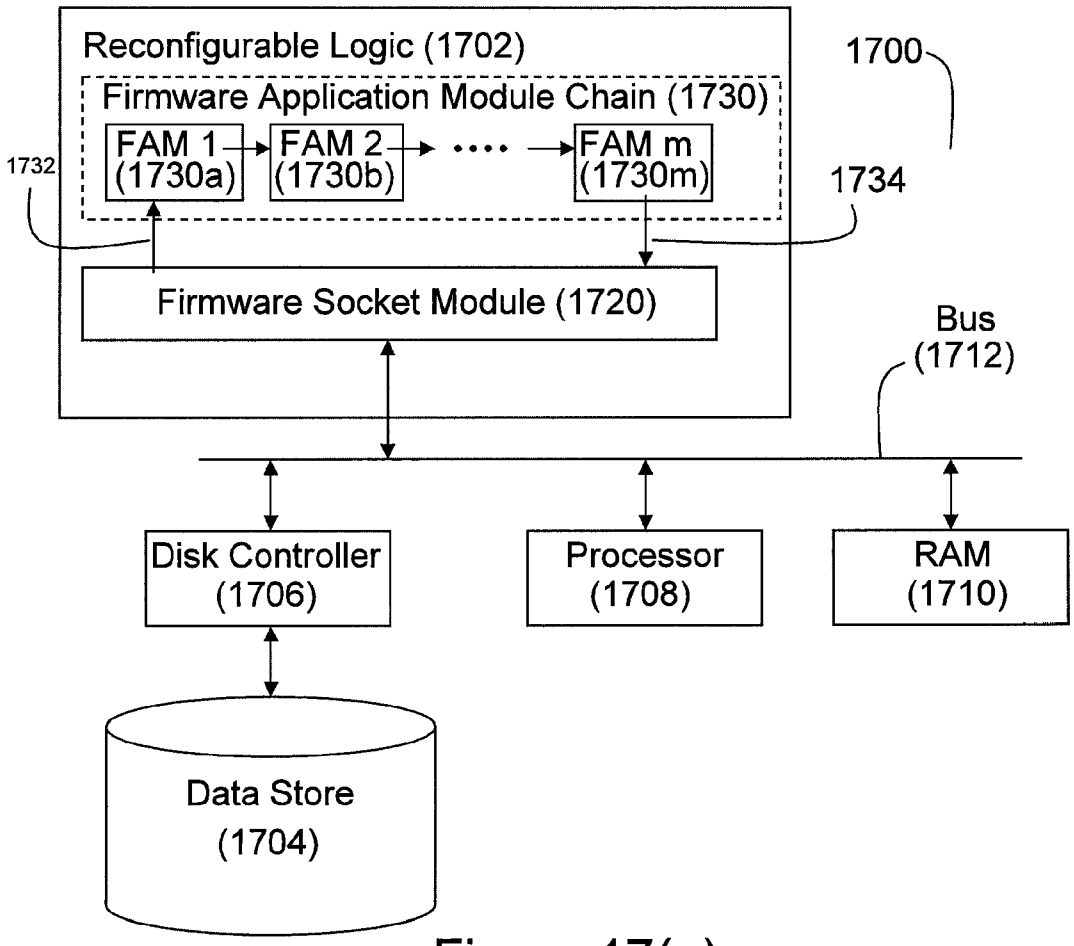


Figure 17(a)

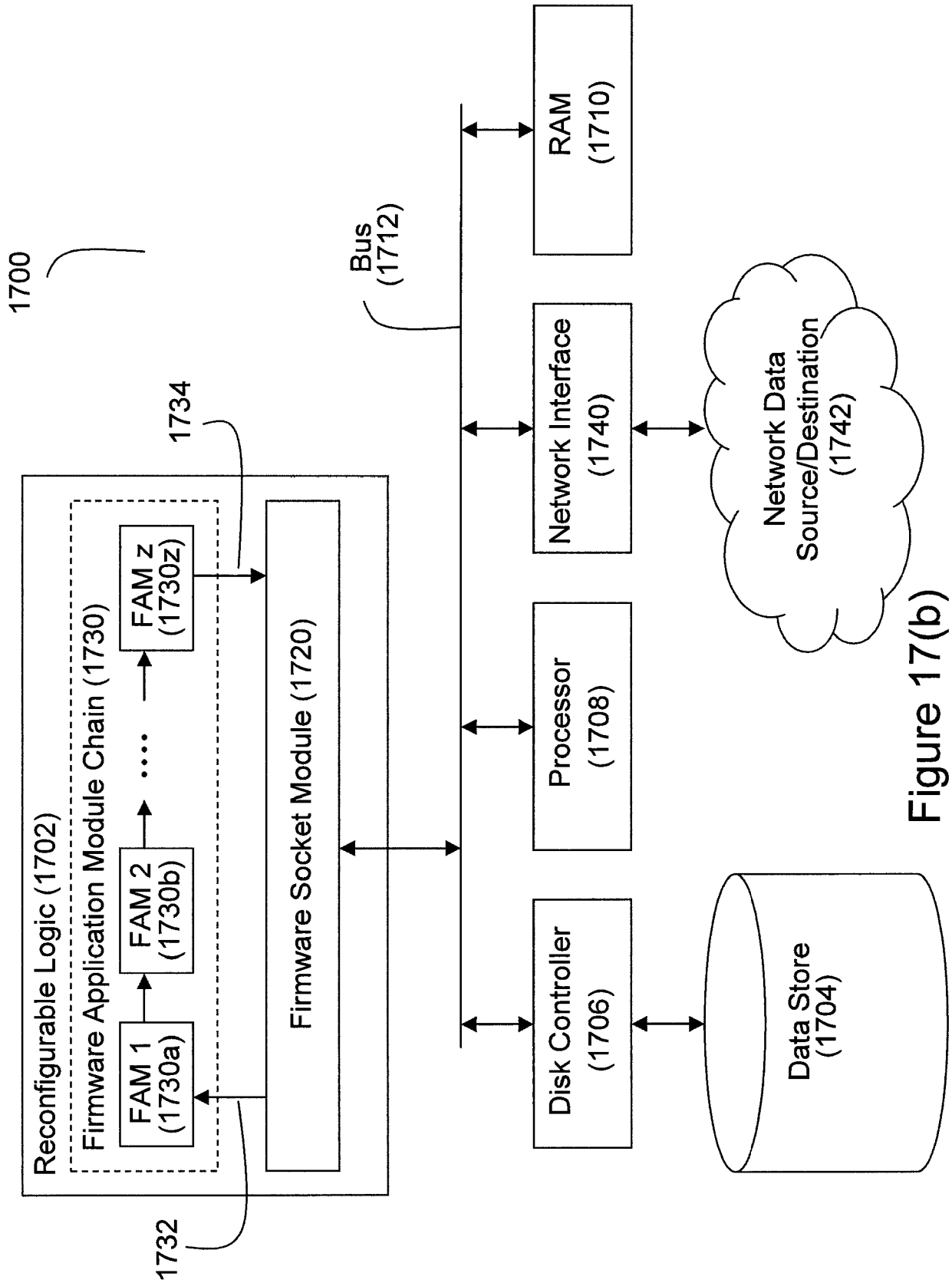

1700
Bus (1712)
RAM (1710)
1734
Reconfigurable Logic (1702)
Firmware Application Module Chain (1730)
FAM 1 (1730a)
FAM 2 (1730b)
FAM z (1730z)
Firmware Socket Module (1720)
Network Interface (1740)
Network Data Source/Destination (1742)
Processor (1708)
Disk Controller (1706)
Data Store (1704)
1732

Figure 17(b)

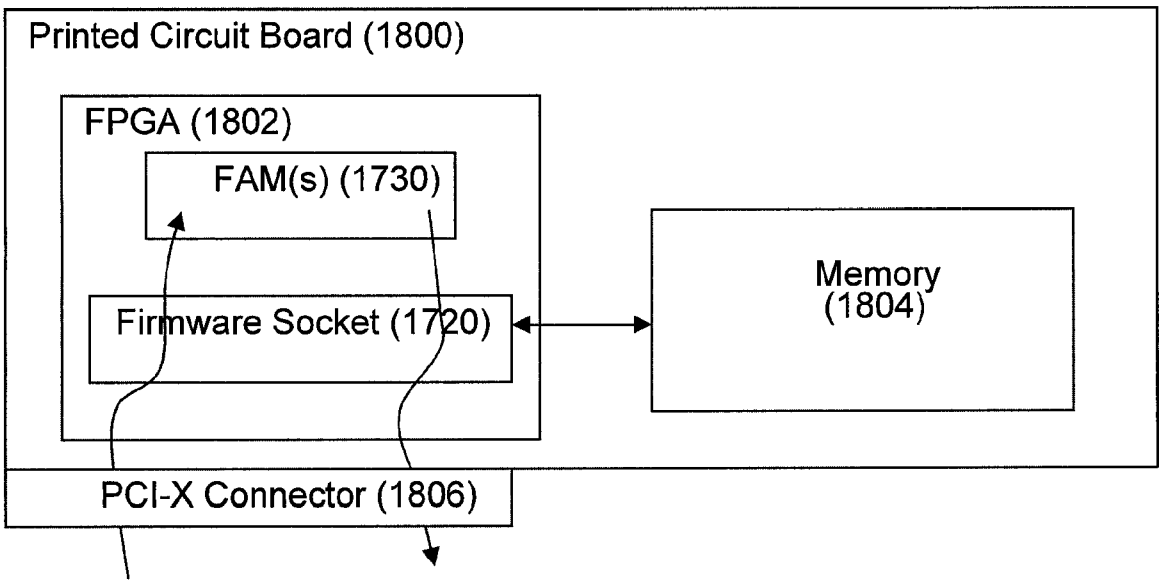


Figure 18(a)

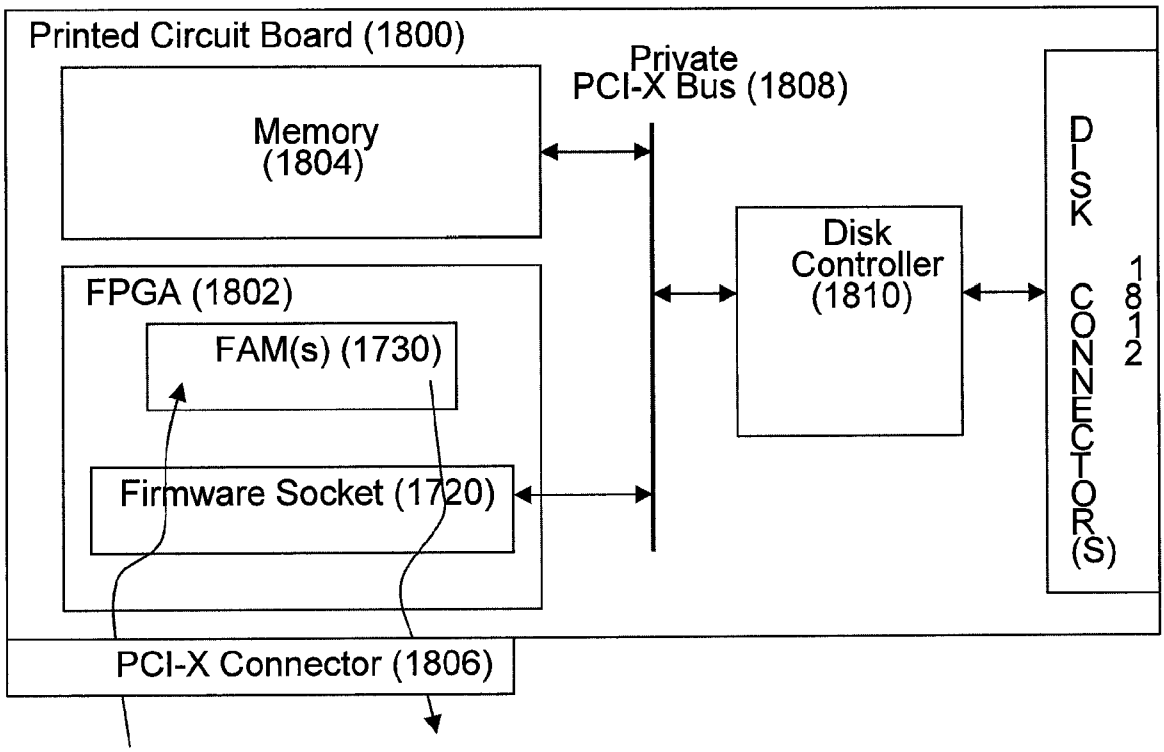


Figure 18(b)

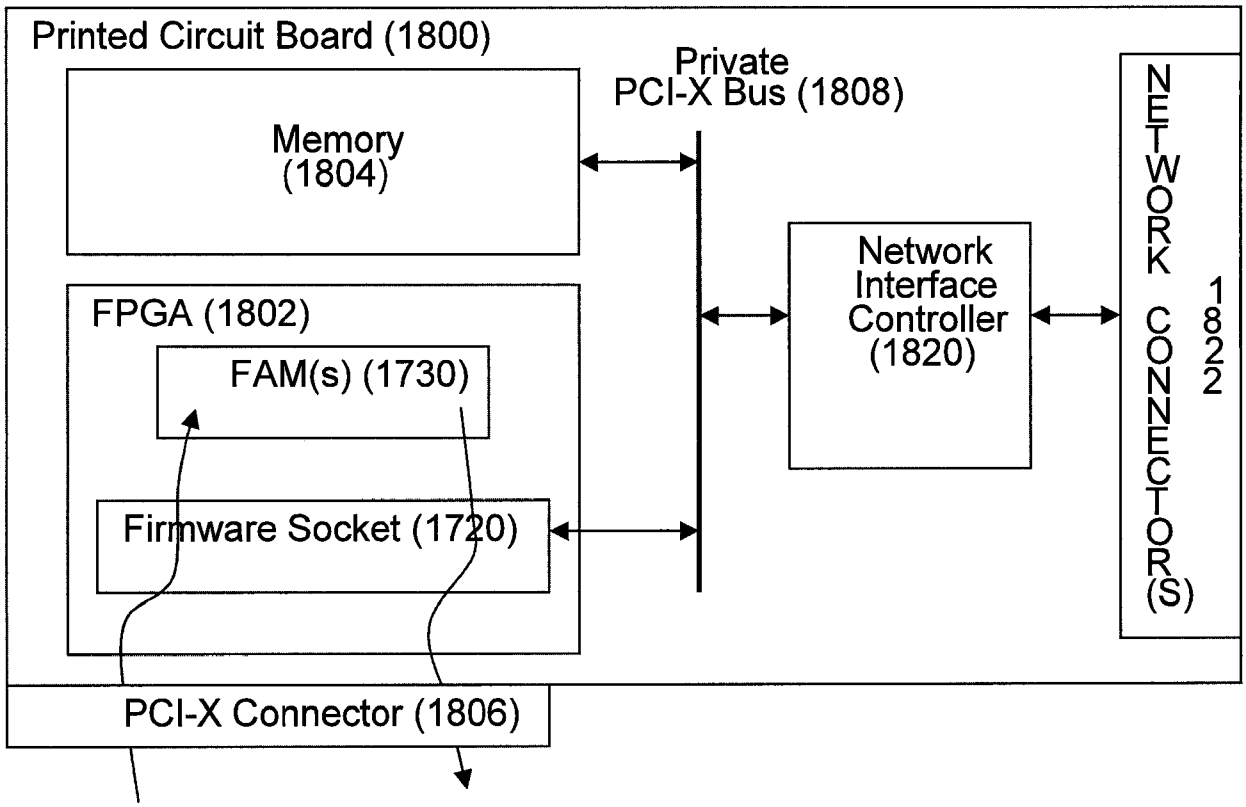

Printed Circuit Board (1800)
Memory (1804)
Private PCI-X Bus (1808)
Network Interface Controller (1820)
NETWORK CONNECTOR(S) 1822
FPGA (1802)
FAM(s) (1730)
Firmware Socket (1720)
PCI-X Connector (1806)

Figure 18(c)