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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 April 2015 (02.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/047487 A1

(51) International Patent Classification:
H04L 9/06 (2006.01)

(21) International Application Number:
PCT/US2014/043169

(22) International Filing Date:
19 June 2014 (19.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
13/929,589 27 June 2013 (27.06.2013) US

(71) Applicant: **QUALCOMM INCORPORATED** [US/US]; International IP Administration, 5775 Morehouse Drive, San Diego, California 92121-1714 (US).

(72) Inventor: **AVANZI, Roberto**; 5775 Morehouse Drive, San Diego, California 92121 (US).

(74) Agents: **KING, Eric T.** et al.; Blakely Sokoloff Taylor & Zafman LLP, 1279 Oakmead Parkway, Sunnyvale, California 94085 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS TO ENCRYPT PLAINTEXT DATA

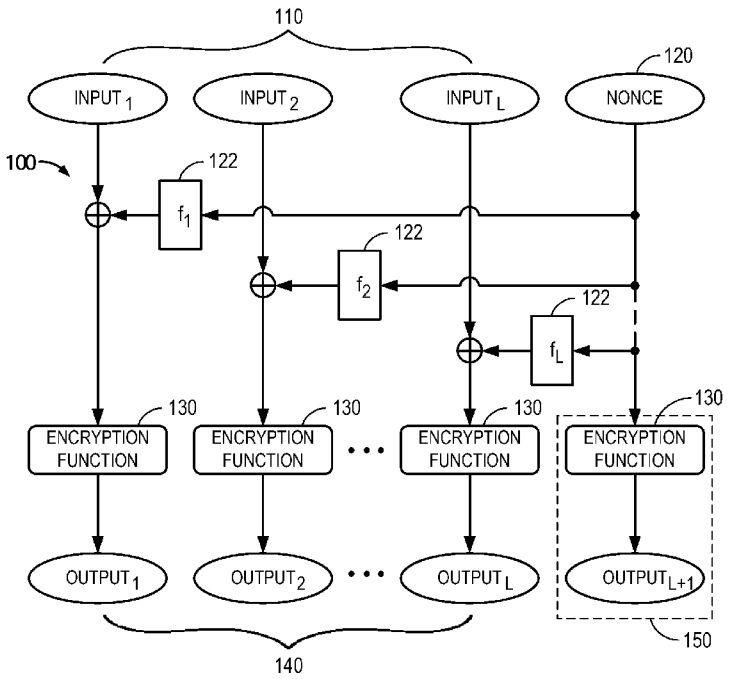

FIG. 1A

(57) Abstract: Disclosed is an apparatus and method for encrypting plaintext data. The method includes: receiving at least one plaintext data input; applying a Nonce through a function to the at least one plaintext data input to create Nonced plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs; and applying the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data. The encrypted output data is then transmitted to memory.

WO 2015/047487 A1

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

METHOD AND APPARATUS TO ENCRYPT PLAINTEXT DATA

BACKGROUND

Field

[0001] The present invention relates to a method and apparatus to encrypt plaintext data and decrypt the corresponding ciphertext data.

Relevant Background

[0002] The use of memory analyzers represents a large threat to the integrity and confidentiality of distributing content. Even if great care is devoted to protect data contained in code, the contents of memory may be captured by bus sniffing. For example, this can be used to leak raw content, even if it is distributed in an encrypted form, after it has been decrypted in a secure environment for rendering. This may be accomplished by "reading" the electric signals corresponding to the writes to the memory. Other more sophisticated attacks may even replay these signals to trick the processor into reading and processing data chosen by an attacker.

[0003] Content providers often have particular requirements for the handling of raw content. At a bare minimum, the content can never be stored in memory in the clear. In most cases, there is at least a requirement that some form of memory scrambling or encryption be applied to all memory recording to prevent physical attacks. As an example, the data written to a specific address is usually a function of the clear data, the address, and a master key. This guarantees that the same data, when written to different addresses, has a different encoding. The use of nonces to randomize the encryption of the plaintext data, when these nonces are stored and retrieved in a secure way, can be used to prevent replay attacks.

[0004] Moreover, throughput requirements for secure communication are putting current stream and block ciphers to test, and novel constructions to increase throughput while at the same time controlling power and area requirements are desirable.

[0005] Unfortunately, the current techniques are often inefficient, and a stronger level of protection, a higher throughput at the same security level and without significant increases in power and, in the case of hardware implementations, area requirements, may be desired.

SUMMARY

[0006] Aspects of the invention may relate to an apparatus and method for encrypting plaintext data. The method includes: receiving at least one plaintext data input; applying a Nonce through a function to the at least one plaintext data input to create plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs; and applying the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data. The encrypted output data is then transmitted to memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIG. 1A is a flow diagram illustrating a process in which a series of blocks of plaintext data inputs are encrypted using an encryption function and a Nonce.

[0008] FIG. 1B is a flow diagram illustrating the reverse decryption process of FIG. 1A.

[0009] FIG. 2 is a flow diagram illustrating a common structure of a block cipher based on the iterations of similar computational blocks referred to as rounds.

[0010] FIG. 3 is a flow diagram illustrating a process to encrypt a data input in expanded form using a first and second set of round functions and augmenting an intermediate step of the encryption process with a Nonce.

[0011] FIG. 4A is a flow diagram illustrating a process to encrypt a series of blocks of plaintext data inputs using the same key and a Nonce or values derived from a Nonce to modify in different ways the encryption processes of the individual blocks.

[0012] FIG. 4B is a flow diagram illustrating the reverse decryption process of FIG. 4A.

[0013] FIG. 5 is a flow diagram illustrating a process to encrypt a data input obtaining several different outputs by applying different values derived from a common Nonce to an intermediate step of the encryption process.

[0014] FIG. 6 is a diagram of an example computer hardware system to implement the data encryption techniques for the purpose of enabling the saving and restoring of encrypted memory to mass storage without having to decrypt and re-encrypt it.

DETAILED DESCRIPTION

[0015] The word "exemplary" or "example" is used herein to mean "serving as an example, instance, or illustration." Any aspect or embodiment described herein as "exemplary" or as an "example" in not necessarily to be construed as preferred or advantageous over other aspects or embodiments.

[0016] Embodiments of the invention relate to techniques to provide an enhanced mechanism for the protection of data stored in memory. In particular, methods and processes are described that extend the functionality of block ciphers in order to enhance memory encryption. Additionally, these techniques may also improve performance, throughput, and power consumption, as will be hereinafter described. These techniques may also be used to improve performance, throughput, and power consumption for the purpose of secure data storage of transmission over (wired or wireless) networks.

[0017] In one embodiment, an encryption scheme is utilized in which a series of L blocks of plaintext data inputs are encrypted using an encryption function (e.g., a block cipher). Prior to encryption with the block cipher, a Nonce is applied through a function to the plaintext data inputs. In particular, a method or process to encrypt plaintext data is disclosed that includes: receiving a plurality of plaintext data inputs; applying a Nonce through a function to the plurality of plaintext data inputs to create Nonced plaintext data outputs; applying an encryption function such as a block cipher to the Nonced plaintext data outputs to create encrypted output data; and transmitting the encrypted output data to memory.

[0018] In one embodiment, as will be more particularly described hereinafter, the method may include: receiving a plurality of plaintext data inputs; applying a Nonce through a function to the plurality of plaintext data inputs to create plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the plurality of plaintext data inputs to create intermediate Nonced data outputs; and applying the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data. The encrypted output data is then transmitted to memory.

Randomizing the Encryption of L blocks of Plaintext Data Inputs

[0019] As can be seen in FIG. 1A, in one embodiment, a method or process 100 is performed in which a plurality of plaintext data inputs (Input1-InputL) 110 are received. A Nonce 120 is applied through a function 122 to the plaintext data inputs (Input1-InputL) 110. The Nonce 120 may be used to randomize the encryption of the L blocks of plaintext data (Input1-InputL) 110. As can be seen in FIG. 1A, L blocks of plaintext data (Input1-InputL) 110 are received and a

Nonce 120 may be applied by functions (f1, f2,. . . , fL) 122 to create Nonced plaintext data outputs. In one embodiment, as will be described in more detail hereinafter, the function to apply the Nonce 120 may include an XOR function. In another embodiment, instead of an XOR function, a modular addition function may be used. An encryption function 130 (e.g., a block cipher) may then be applied to the Nonced plaintext data outputs such that encrypted output data (Output1-OutputL) 140 is outputted to memory.

[0020] It should be appreciated that the Nonce 120 may be subject to some transformations in order to avoid that equal blocks of plaintext (among the L blocks 110 processed at the same time) have the same encryption. Also, because the cryptographic key used in the encryption function 130 used in the L parallel encryptions may be the same, the key schedule does not need to be redone L times

[0021] Further, Nonce 120 may either be stored in a smaller, internal, protected area of memory, or stored in the main memory, either in a clear or encrypted manner, depending on use cases, as will be described in more detail hereinafter.

[0022] Also, for simplicity, particular encryption keys used in the encryption functions 130 are not represented. However, it should be appreciated that the encryption functions take an additional input, which is the key used by the encryption function in the encryption process. Moreover, it should also be appreciated that, in the case the encryption functions are iterated block ciphers using the same encryption key, the various vertical pipelines may share the same key schedule, where some fixed bit permutations (such as rotations) may be applied to the rounds before being used in the encryption functions 130. In a hardware implementation, these permutations should have no performance impact, because they amount to just a different wiring in the silicon.

[0023] The functions (f1, f2,. . . , fL) 122 may be mathematical functions that derive values from the Nonce 120 in order to perturbate the computation of the Nonced plaintext data outputs in a manner unpredictable for an attacker. These can be maskings with constants, different circular rotations, or other functions that may be related to the chosen encryption function 130. If the Nonce 120 has larger size than the cipher block length, then the functions may just be extractions of segments of the Nonce.

[0024] Further, methodology 100 may be parallelizable, utilizing L or L+1 implementations of the same encryption function 130 (or a different encryption function may be utilized). As can be seen in FIG. 1A, in dashed lines 150, an expansion of the ciphertext is shown as an L+1-th implementation, which outputs one additional block. Also, because the same encryption key for

the encryption function 130 may be used for each block, sub-key derivation needs to be performed only once, thus saving hardware resources.

[0025] In some embodiments, sufficient security may be provided by storing the Nonce 120 in the clear in an accessible memory area, as it plays a role similar to that of an initialization vector. A benefit of this approach is that the Nonce 120 can be shorter than the block size, and therefore it may be applied in the function operations 122 only to selected bit fields of the input blocks 110. This scheme may be useful for memory encryption. As an example, if the block cipher has a block size of 128 bits and cache lines are 128 bytes long, by setting L= 8, whole cache lines can be encrypted at once when they are spilled from the last level of the cache.

[0026] Thus, as previously described, process 100 extends the functionality of block ciphers in order to enhance memory encryption. In particular, encryption scheme 100 utilizes a series of L blocks of plaintext data inputs (Input1-InputL) 110 that are each encrypted using encryption function 130, in which prior to encryption with encryption function 130, a Nonce 120 is applied through a function 122 to the plaintext data inputs 100. The encryption function 130 may be applied to the Nonced plaintext data outputs such that encrypted output data (Output1-OutputL) 140 is outputted to memory.

Decryption works backwards. For example, with reference to FIG. 1B, the inverse function of encryption function 130 can be applied to the encrypted output data from memory (shown as Input 140), that is the corresponding decryption primitive, may be used to compute the composition of $Input_i$ and Nonce, for instance $Input_i \oplus$ Nonce, for i = 1, 2, …, L and Nonce, from which the original Inputs are recoverable (shown as Outputs 110).

Randomizing the Block Cipher

[0027] As will be described hereinafter, the plaintext data inputs 110 may first be encrypted by a first sequence of round functions that constitute the block cipher (which is the chosen encryption function), before the Nonce 120 is applied, and thereafter, the Nonce is applied, to create Nonced data outputs. The Nonced data outputs may then be encrypted by a second sequence of round functions modeling the block cipher (which is the chosen encryption function) to create the encrypted output data that is outputted to memory.

[0028] In order to model the encryption function 130 (e.g., block cipher) various constructions may be used. For example, constructions such as Luby-Rackoff constructions may be used, e.g., Feistel networks (such as Data Encryption Standard (DES)), and Substitution-Permutation (SP) networks (such as Advanced Encryption Standard (AES)). In both cases, one parameterized non-linear function is repeatedly applied to the input. Each application of this function may be

referred to as a "round" or "round function". The output of a round is the input of the next round. The plaintext is the input to the first round, and the ciphertext is the output of the last round. The round function takes a further parameter called the round key and the round keys are derived from the encryption/decryption key (e.g., the cipher key).

[0029] With reference to FIG. 2, an example of a process 200 to generate a block cipher based upon round functions is illustrated. As shown in FIG. 2, a plaintext data input 202 is inputted to a plurality of N rounds 204 of the round function, modeling the block cipher. Therefore, the block cipher is modeled by the plurality of N rounds 204 of round functions, where k1, k2, . . . kN are the round keys for Rounds 1, 2, . . . , N respectively. Output 206 is the encrypted plaintext data input 202 encrypted by the round function (modeling the block cipher) applied to the plaintext data input 202. It should be appreciated that decryption would be the exact same process in reverse.

[0030] An example implementation will be hereinafter described. For example, a performance efficient implementation of this scheme may require two parallel implementations of the same block cipher, possibly sharing the round keys. In order to reduce hardware implementation costs, the Nonce may be applied in the middle of the cipher. By means of this, the part of the cipher before the application of the Nonce must be implemented only once, and the part of the cipher following the application of the Nonce is implemented twice.

[0031] As an example, with reference to FIG. 3 which illustrates a process 300, the plaintext data input 302 may be encrypted through M of the N rounds ($1 \leq M < N$), e.g., M rounds 304 parametrised using M round keys (k1, k2, … kM). Next, the Nonce (v) 306 is applied – for instance XORing it to the output X of the M-th round - and the XORed output and the Nonce are encrypted further, independently (separate block 308) - and resuming the process with the (M + 1)-th round. As can be seen in FIG. 3, the next round of round keys k' and k'' for N-M rounds 310 may be the same set of round keys or may be slight variants of each other, such as different rotations or masked with different secret constants. Additionally, the outputs may be concatenated (block 314) resulting in Output 316.

[0032] Another example of a slightly different implementation may consist of a permutation of the bits of X and of the Nonce. For example, if X were set to = $X_{hi} \| X_{lo}$ (decomposition as concatenation of two bit strings of equal length) and v (Nonce) was set to v = $v_{hi} \| v_{lo}$ then A would be A = $X_{hi} \| vlo$ and B would be B = $X_{lo} \| v_{hi}$. Therefore, if the last N - M rounds have sufficient diffusion, then there is enough influence of both X and Nonce on both halves C and D of the Output. It should be appreciated that this is just an example, other bit permutations are

possible. However, if the block size is large enough the scheme may not lead to frequent (partial) repetitions of the same ciphertexts for the same plaintext. Therefore, it may be advisable to have the Nonce influence the whole of the next input, such as an equation like, $A = (X_{hi} \oplus v_{hi}) \| v_{lo}$ and $B = (X_{lo} \oplus v_{lo}) \| v_{hi}$. It should be appreciated that significance here is that the process be easily reversible –such that the Nonce can be recovered once the decryption process has performed N - M rounds. Further, the concatenate function 314 can be the concatenation of the two outputs of the last parallel rounds - but any another bit permutation of the two inputs can be used here. The process is beneficial in that hardware implementations of the first M rounds do not have to be duplicated - but only for the last N-M rounds. Decryption works also in this case backwards. The two "sides" C and D of Output are decrypted in parallel for the last N - M rounds, until Nonce v is recovered, the separate operation is reversed, and then the decryption of the Input is completed in M rounds.

[0033] With reference to FIG. 4A, an example of a process 400 is illustrated, generalizing the previous techniques, to simultaneously encrypt L blocks of plaintext data inputs (Input1-InputL) 410, in which a Nonce 420 is added to each block, after being suitably transformed, along with the use of various rounds. In particular, the process 400 of FIG. 4A, illustrates that the plaintext data inputs 410 may be encrypted by a first sequence of round functions (M rounds 404) before the Nonce 120 is applied, and thereafter, the Nonce 420 is applied, to create Nonced data outputs. As can be seen in FIG. 4A, the L blocks of plaintext data (Input1-InputL) 410 are received and functions (f1, f2,. . . , fL) 422 are applied to a Nonce 420 to create differently Nonced data outputs. In one embodiment, the function to apply the Nonce 420 may include a XOR function. Alternatively, other easily invertible functions, such as modular additions or subtractions may be used to apply the (values derived from the) Nonce. The Nonced data outputs may then be encrypted by a second sequence (N-M rounds 406) of round functions to create the encrypted output data 440 that is outputted to memory. It should be appreciated that by utilizing the M and N-M rounds 404 and 406 that the full encryption function is thereby modeled and applied. Further, methodology 400 may be parallelizable, utilizing L (in the case the nonce is not encrypted) or L+1 (in the case the nonce is encrypted) implementations of the round functions 404 and 406 to create the encrypted output data that is outputted to memory.

[0034] It should be appreciated that function (f1, f2, . . . , fL) 422 perform substantially the same roles as described with reference to FIG. 1A. However, the fact that the functions are not implemented until the (M + 1)-th round 406 of the underlying block cipher allows a more complex derivation from the Nonce. In the case of AES implementation, some alteration of the

AES key scheduling procedure could be adopted to generate the functions. In one embodiment, the functions could be computed in parallel with the first M rounds 404 of the block cipher. It may be beneficial to not feed all the same round keys to the various rounds, but to also apply some fixed permutations and/or maskings to them, which are unique for each vertical pipeline. Also, it may be beneficial, depending on use case requirements, to just store the Nonce 420 in the clear in an accessible or in a protected memory area, as it may play a role similar to that of an initialization vector, and may still be secure enough.

[0035] Decryption works backwards. For example, with reference to FIG. 4B, inputs 440 are the outputs of the encryption of FIG. 4A and the outputs 410 should correspond to the original inputs (i.e., the original plaintext inputs).

Resource Savings

[0036] It should be appreciated that all of the previous schemes have been based on the idea that the plaintext is encrypted directly by the encryption function. However, several modes of operations for block ciphers use the encryption primitive to generate a key stream that is XORed to the plaintext to derive the ciphertext - for instance Counter (CTR) mode. Examples of this type of encryption will be hereinafter described. When attempting to save resources for key stream generation, it needs to be ensured that too much saving does not occur at the expense of security – i.e., the various key stream blocks must appear uncorrelated to each other. For instance, it may be tempting to reuse a block from the "key stream" to encrypt several input blocks – in a memory encryption scenario this could easily solve the problem of the area of the memory encryption circuits. However, if two blocks of plaintext P1 and P2, are both XORed, with the same pad π, the ciphertext blocks would be $C1 = P1 \oplus \pi$ and $C2 = P2 \oplus \pi$, which satisfy $P1 \oplus P2 = C1 \oplus C2$. This can reveal significant information about the plaintext, and is thus unsuitable to store critical information. However, it may be beneficial to use common hardware to compute only the first rounds of two or more blocks of the key stream, and then perform the last rounds separately. The security of such a method depends on the cryptanalysis of reduced round versions of the used cipher and the predictability of intermediate values after some rounds.

[0037] An example of this is displayed with reference to FIG. 5. In this example embodiment process 500, the Input 502 and the Nonce (v) 520 are values used to generate L key stream blocks. The Input 502 is not the plaintext. Similarly, Output1, Output2,…OutputL 540 are not the ciphertext, but L blocks of the ciphertext are XORed to these values as in the CTR mode of operation (or are used in a more complex way in some variants of other modes of encryption

which only use the encryption primitive of the block cipher). In other aspects, FIG. 5 is similar to FIG. 4A, including a first round (M rounds 504) of round keys before the Nonce 520 is applied, and thereafter, the Nonce 520 is applied, to create Nonced data outputs. Nonce 520 may be applied by functions (f1, f2,. . . , fL) 522 to create Nonced outputs. The function to apply the Nonce 520 may include an XOR function. The Nonced outputs may then be encrypted by a second round (N-M rounds 506) of round keys to create encrypted output 540.

If AES (e.g., AES-128) is chosen as the block cipher, then M = 3 or 4 in view of current cryptanalytic results may be used. The rationale being that AES-128 reduced to 6 or 7 rounds is still considerably difficult to attack and then only if the attacker can control the input – which is not possible in this situation. For example, suppose the use case is memory encryption, where whole cache lines are encrypted, these are 128 bytes, so we need 8 blocks (L=8). This means that, for M = 3, a total of 3 + 8 *7 = 59 rounds of AES need to be implemented in HW, in place of 80, leading to an area and power saving of about 26%. For M = 4, the number of rounds of AES that are implemented is 4 + 8* 6 = 52, for a saving of about 35%. The savings may be a bit larger if the key schedule for the last N - M rounds is common to all the pipelines – perhaps with just some fixed bit permutations of the round keys in the parallel pipelines, but probably not more than that – as this should be more than offset by the logic for deriving from the nonce different values to be XORed to the inputs to the (M + 1)-th round.

Computation of Nonces

[0038] In one embodiment, each time a new block (or set of L blocks) needs to be written to memory, the Nonce may be refreshed. If the block cipher has sufficient diffusion (or it has sufficient diffusion in the last N - M rounds), then it may be sufficient to just shift the Nonce by, for example, s bits, and then append s new fresh random bits to the Nonce. For example, this may be computed for Nonce (v) as $v \leftarrow (v << s) \oplus r$, where r is a string of s bits. Further, the fresh bits can be shifted in from the most significant position, or v can be partitioned in various sub-registers that are independently shifted and refreshed. However, if this strategy is used, the Nonces should not be stored in the clear, but encrypted, because storing them in the clear could possibly make future Nonces partially predictable thereby possibly helping cryptanalysis. It should further be noted that the Nonce can either: (a) be a value independent of the physical memory address where the data will be stored; or (b) be dependent from that address. For the latter case, it could be the concatenation of: (i) the physical memory address and (ii) of a random value, a (encrypted) counter, or a value computed by the methods described above or a by a different method.

Example Hardware

[0039] Example computer hardware 600 that may implement the previously described methods and processes is illustrated in FIG. 6. The computer system 600 is shown comprising hardware elements that can be electrically coupled via busses (or may otherwise be in communication, as appropriate). The hardware elements may include at least one main processor 602 (e.g., central processing unit (CPU)) as well as other processors 604. It should be appreciated that these processors may be general-purpose processors and/or one or more special-purpose processors (such as digital signal processing chips, graphics acceleration processors, and/or the like). The processors may be coupled to respective memory management units (MMUs) 610, which may in turn be coupled through caches 612 (e.g., caches may or may not be present and/or may be separate or incorporated into other elements) (surrounded by dashed lines), to an encryptor processing unit 620 and/or to memory 630 and/or storage devices 640. As will be described hereinafter, encryptor 620 may utilize the previously described methods and processes to extend the functionality of cipher blocks in order to enhance memory encryption for data to be stored in memory.

[0040] It should be appreciated that computer 600 may include other devices (not shown), such as: input devices (e.g., keyboard, mouse, keypad, microphone, camera, etc.); and output devices (e.g., display device, monitor, speaker, printer, etc.). Computer 600 may further include (an/or be in communication with) one or more memory elements, storage devices 630,640, which may comprise local and/or network accessible storage, and/or can include, without limitation, a disk drive, a drive array, an optical storage device, solid-state storage device such as a random access memory ("RAM") and/or a read-only memory ("ROM"), which can be programmable, flash-updateable, and/or the like. Computer 600 may also include a communication subsystem, which may include a modem, a network card (wireless or wired), an infrared communication device, a wireless communication device and/or chipset (such as a Bluetooth device, an 802.11 device, a Wi-Fi device, a WiMax device, cellular communication device, etc.), and/or the like. The communications subsystem may permit data to be exchanged with a network, other computer systems, and/or any other devices described herein. It should be appreciated that computer 600 may be a mobile device, non-mobile device, wireless device, wired device, etc., and may have wireless and/or wired connections, and may be any type of electronic or computing device.

[0041] In one embodiment, if data is to be stored at an encrypted location (decision block 650), then encryptor 620 (e.g., a device to encrypt data) may implement the previously described process (with additional reference to FIG. 1A) including: receiving a plurality of plaintext data

inputs (Input1-InputL) 110; applying a Nonce 122 through functions (f1, f2,. . . , fL) 122 to create Nonced plaintext data outputs that are randomized; and applying an encryption function 130 to the Nonced plaintext data outputs such that encrypted output data (Output1-OutputL) 140 is outputted to memory 630. This data may further be stored in storage 640. In other embodiments, as previously described, to apply an encryption function, encryptor 620 may encrypt the plaintext data inputs utilizing a first sequence of round functions modeling the encryption function before the Nonce is applied. After, this the Nonce is applied, to create the Nonced data outputs. The Nonced data outputs may then be encrypted by a second sequence of round functions modeling the encryption function to create the encrypted output data that is outputted to memory 630. Examples of these implementations are illustrated in FIGs. 2-5, as previously described in detail.

[0042] However, if at decision block 650, the data is determined not to be stored at an encrypted location, then the data may be normally stored to memory 630 and/or normal memory mapping input/outs and control 655 may utilized to implement direct memory access (DMA) control to storage 640.

[0043] Generally, when memory encryption is available, its contents need be decrypted before they are written to storage device in a virtual memory system. However, to accommodate this, according to embodiments of the invention, a DMA data transfer channel may be used to read the actual, encrypted contents of the memory 630 (e.g., RAM, DDR RAM, etc.) and can be used to write them to a sector of the storage device 640 (e.g., a hard drive or a flash memory), as well as, to read from a sector and place the contents directly into memory 630. Thus, these memory encryption methods may be independent of the physical addresses and pages can be swapped out and back in without additional encryption/decryption overhead.

[0044] A benefit of the previously described system is that memory contents do not need to be decrypted and re-encrypted each time they are moved to the swap file and back to memory, which results in significant power savings and in time savings. Further, the techniques described herein, not only offer good direct protection against physical or electrical memory attacks – i.e. against direct reading of the memory – but also offer resistance against attacks that use the bus traffic as a side channel, as repeated writes of the same or correlated data to the same location are effectively randomized. Furthermore, the techniques described herein require a relatively small additional hardware implementation. Also, the techniques described herein are generic enough such that they can be applied to essentially any commonly-used block cipher. Additionally, the input and output sizes of each round do not all have to be equal and masking operations have to

be adapted only minimally in these cases. Moreover, the direct DMA channel for saving encrypted memory can also bring significant savings in power consumption and time.

[0045] Further, as previously described, the Nonce may be stored in the main memory 630, either in a clear unencrypted manner or in an encrypted manner, depending on the implementation. Alternatively, as previously described, the Nonce may be stored in a small, protected area of a specialized memory.

[0046] Also, it should be appreciated that, in one example, if a fixed key is chosen randomly at device boot, the corresponding key schedule may be pre-computed at the time. As a particular example, there could be a master key, or a dependency on memory address, if required, that could be placed in the key. As a further example, the Nonce could be: a fixed value (in which case all derived constants, such as: the outputs of functions (f1, f2,. . . ,fL) can be pre-computed), a per page value, or could be dependent on the physical memory address. These example schemes may be used for simplification purposes.

[0047] It should be appreciated that techniques to provide an enhanced mechanism for the protection of data stored in memory by extending the functionality of block ciphers, as previously described, may be implemented as software, firmware, hardware, combinations, thereof, etc. In one embodiment, the previous described functions may be implemented by one or more processors (e.g., encryptor 620 or other processors) of a computer 600 to achieve the previously desired functions (e.g., the method operations of Figures 1-5). Moreover, as previously described with reference to Figures 1-5, decryption simply works backwards.

[0048] It should be appreciated that aspects of the invention previously described may be implemented in conjunction with the execution of instructions by processors of the devices, as previously described. Particularly, circuitry of the devices, including but not limited to processors, may operate under the control of a program, routine, or the execution of instructions to execute methods or processes in accordance with embodiments of the invention. For example, such a program may be implemented in firmware or software (e.g. stored in memory and/or other locations) and may be implemented by processors and/or other circuitry of the devices. Further, it should be appreciated that the terms processor, microprocessor, circuitry, controller, etc., refer to any type of logic or circuitry capable of executing logic, commands, instructions, software, firmware, functionality, etc

[0049] It should be appreciated that when the devices are mobile or wireless devices that they may communicate via one or more wireless communication links through a wireless network that are based on or otherwise support any suitable wireless communication technology. For

example, in some aspects the wireless device and other devices may associate with a network including a wireless network. In some aspects the network may comprise a body area network or a personal area network (e.g., an ultra-wideband network). In some aspects the network may comprise a local area network or a wide area network. A wireless device may support or otherwise use one or more of a variety of wireless communication technologies, protocols, or standards such as, for example, 3G, LTE, Advanced LTE, 4G, CDMA, TDMA, OFDM, OFDMA, WiMAX, and WiFi. Similarly, a wireless device may support or otherwise use one or more of a variety of corresponding modulation or multiplexing schemes. A wireless device may thus include appropriate components (e.g., air interfaces) to establish and communicate via one or more wireless communication links using the above or other wireless communication technologies. For example, a device may comprise a wireless transceiver with associated transmitter and receiver components (e.g., a transmitter and a receiver) that may include various components (e.g., signal generators and signal processors) that facilitate communication over a wireless medium. As is well known, a mobile wireless device may therefore wirelessly communicate with other mobile devices, cell phones, other wired and wireless computers, Internet web-sites, etc.

[0050] The teachings herein may be incorporated into (e.g., implemented within or performed by) a variety of apparatuses (e.g., devices). For example, one or more aspects taught herein may be incorporated into a computer, a wired computer, a wireless computer, a phone (e.g., a cellular phone), a personal data assistant ("PDA"), a tablet, a mobile computer, a mobile device, a non-mobile device, a wired device, a wireless device, a laptop computer, an entertainment device (e.g., a music or video device), a headset (e.g., headphones, an earpiece, etc.), a medical device (e.g., a biometric sensor, a heart rate monitor, a pedometer, an EKG device, etc.), a user I/O device, a fixed computer, a desktop computer, a server, a point-of-sale (POS) device, an entertainment device, a set-top box, an ATM, or any other suitable electronic/computing device. These devices may have different power and data requirements

[0051] In some aspects a wireless device may comprise an access device (e.g., a Wi-Fi access point) for a communication system. Such an access device may provide, for example, connectivity to another network (e.g., a wide area network such as the Internet or a cellular network) via a wired or wireless communication link. Accordingly, the access device may enable another device (e.g., a WiFi station) to access the other network or some other functionality.

[0052] Those of skill in the art would understand that information and signals may be represented using any of a variety of different technologies and techniques. For example, data, instructions, commands, information, signals, bits, symbols, and chips that may be referenced throughout the above description may be represented by voltages, currents, electromagnetic waves, magnetic fields or particles, optical fields or particles, or any combination thereof.

[0053] Those of skill would further appreciate that the various illustrative logical blocks, modules, circuits, and algorithm steps described in connection with the embodiments disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, and steps have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

[0054] The various illustrative logical blocks, modules, and circuits described in connection with the embodiments disclosed herein may be implemented or performed with a general purpose processor, a digital signal processor (DSP), an application specific integrated circuit (ASIC), a field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, or any combination thereof designed to perform the functions described herein. A general purpose processor may be a microprocessor, but in the alternative, the processor may be any conventional processor, controller, microcontroller, or state machine. A processor may also be implemented as a combination of computing devices, e.g., a combination of a DSP and a microprocessor, a plurality of microprocessors, one or more microprocessors in conjunction with a DSP core, or any other such configuration.

[0055] The steps of a method or algorithm described in connection with the embodiments disclosed herein may be embodied directly in hardware, in a software module executed by a processor, or in a combination of the two. A software module may reside in RAM memory, flash memory, ROM memory, EPROM memory, EEPROM memory, registers, hard disk, a removable disk, a CD-ROM, or any other form of storage medium known in the art. An exemplary storage medium is coupled to the processor such the processor can read information from, and write information to, the storage medium. In the alternative, the storage medium may be integral to the processor. The processor and the storage medium may reside in an ASIC. The

ASIC may reside in a user terminal. In the alternative, the processor and the storage medium may reside as discrete components in a user terminal.

[0056] In one or more exemplary embodiments, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software as a computer program product, the functions may be stored on or transmitted over as one or more instructions or code on a computer-readable medium. Computer-readable media includes both computer storage media and communication media including any medium that facilitates transfer of a computer program from one place to another. A storage media may be any available media that can be accessed by a computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to carry or store desired program code in the form of instructions or data structures and that can be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if the software is transmitted from a web site, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, then the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. Disk and disc, as used herein, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and blu-ray disc where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0057] The previous description of the disclosed embodiments is provided to enable any person skilled in the art to make or use the present invention. Various modifications to these embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments without departing from the spirit or scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown herein but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

WHAT IS CLAIMED IS:

1. A method to encrypt plaintext data comprising:

receiving at least one plaintext data input;

applying a Nonce through a function to the at least one plaintext data input to create Nonced plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs;

applying the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data; and

transmitting the encrypted output data to memory.

2. The method of claim 1, wherein the Nonce is stored in an encrypted manner.

3. The method of claim 1, wherein the Nonce is stored in an unencrypted manner.

4. The method of claim 1, wherein the encryption function applied to the data outputs is the same encryption function.

5. The method of claim 1, wherein the function includes an XOR function.

6. The method of claim 1, wherein the function is a mathematical function that derives values from the Nonce to perturbate the Nonced plaintext data outputs in an unpredictable manner.

7. The method of claim 6, wherein the function includes binary or arithmetic addition constants, circular rotations or arbitrary permutations of the bits representing the input to said function.

8. The method of claim 6, wherein the function is related to the encryption function.

9. The method of claim 1, wherein applying the encryption function includes encrypting the plaintext data inputs by a first sequence of round functions modeling the encryption function before the Nonce is applied, and thereafter, the Nonce is applied, to create Nonced data outputs.

10. The method of claim 9, wherein the Nonced data outputs are encrypted by a second sequence of round functions modeling the encryption function to create the encrypted output data.

11. The method of claim 1, further comprising decrypting the encrypted output data from memory.

12. A non-transitory computer-readable medium including code that, when executed by a processor, causes the processor to:

receive at least one plaintext data input;

apply a Nonce through a function to the at least one plaintext data input to create Nonced plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs;

apply the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data; and

transmit the encrypted output data to memory.

13. The computer-readable medium of claim 12, wherein the Nonce is stored in an encrypted manner.

14. The computer-readable medium of claim 12, wherein the Nonce is stored in an unencrypted manner.

15. The computer-readable medium of claim 12, wherein the encryption function applied to the data outputs is the same encryption function.

16. The computer-readable medium of claim 12, wherein the function includes an XOR function.

17. The computer-readable medium of claim 12, wherein the function is a mathematical function that derives values from the Nonce to perturbate the Nonced plaintext data outputs in an unpredictable manner.

18. The computer-readable medium of claim 17, wherein the function includes binary or arithmetic addition constants, circular rotations or arbitrary permutations of the bits representing the input to said function.

19. The computer-readable medium of claim 17, wherein the function is related to the encryption function.

20. The computer-readable medium of claim 12, further comprising code to apply the encryption function by encrypting the plaintext data inputs by a first sequence of round functions modeling the encryption function before the Nonce is applied, and thereafter, the Nonce is applied, to create Nonced data outputs.

21. The computer-readable medium of claim 20, further comprising code to encrypt the Nonced data outputs by a second sequence of round functions modeling the encryption function to create the encrypted output data.

22. The computer-readable medium of claim 12, further comprising code to decrypt the encrypted output data from memory.

23. A device to encrypt plaintext data comprising:

a processor to:

receive at least one plaintext data input;

apply a Nonce through a function to the at least one plaintext data input to create Nonced plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs;

apply the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data; and

transmit the encrypted output data to memory.

24. The device of claim 23, wherein the Nonce is stored in an encrypted manner.

25. The device of claim 23, wherein the Nonce is stored in an unencrypted manner.

26. The device of claim 23, wherein the encryption function applied to the data outputs is the same encryption function.

27. The device of claim 23, wherein the function includes an XOR function.

28. The device of claim 23, wherein the function is mathematical function that derives values from the Nonce to perturbate the Nonced plaintext data outputs in an unpredictable manner.

29. The device of claim 28, wherein the function includes binary or arithmetic addition constants, circular rotations or arbitrary permutations of the bits representing the input to said function.

30. The device of claim 28, wherein the function is related to the encryption function.

31. The device of claim 23, wherein applying the encryption function includes encrypting the plaintext data inputs by a first sequence of round functions modeling the encryption function before the Nonce is applied, and thereafter, the Nonce is applied, to create Nonced data outputs.

32. The device of claim 31, wherein the Nonced data outputs are encrypted by a second sequence of round functions modeling the encryption function to create the encrypted output data.

33. The device of claim 23, wherein the processor further decrypts the encrypted output data from memory.

34. A device to encrypt plaintext data comprising:
means for receiving at least one plaintext data input;

means for applying a Nonce through a function to the at least one plaintext data input to create Nonced plaintext data outputs and/or to intermediate values of a portion of an encryption function applied to the at least one plaintext data input to create intermediate Nonced data outputs;

means for applying the encryption function to at least one of the Nonced plaintext data outputs and/or the intermediate Nonced data outputs to create encrypted output data; and

means for transmitting the encrypted output data to memory.

35. The device of claim 34, wherein the Nonce is stored in an encrypted manner.

36. The device of claim 34, wherein the Nonce is stored in an unencrypted manner.

37. The device of claim 34, wherein the encryption function applied to the data outputs is the same encryption function.

38. The device of claim 34, wherein the function includes an XOR function.

39. The device of claim 34, wherein the function is a mathematical function that derives values from the Nonce to perturbate the Nonced plaintext data outputs in an unpredictable manner.

40. The device of claim 39, wherein the function includes binary or arithmetic addition constants, circular rotations or arbitrary permutations of the bits representing the input to said function.

41. The device of claim 39, wherein the function is related to the encryption function.

42. The device of claim 34, wherein applying the encryption function includes encrypting the plaintext data inputs by a first sequence of round functions modeling the encryption function before the Nonce is applied, and thereafter, the Nonce is applied, to create Nonced data outputs.

43. The device of claim 42, wherein the Nonced data outputs are encrypted by a second sequence of round functions modeling the encryption function to create the encrypted output data.

44. The device of claim 34, further comprising means for decrypting the encrypted output data from memory.

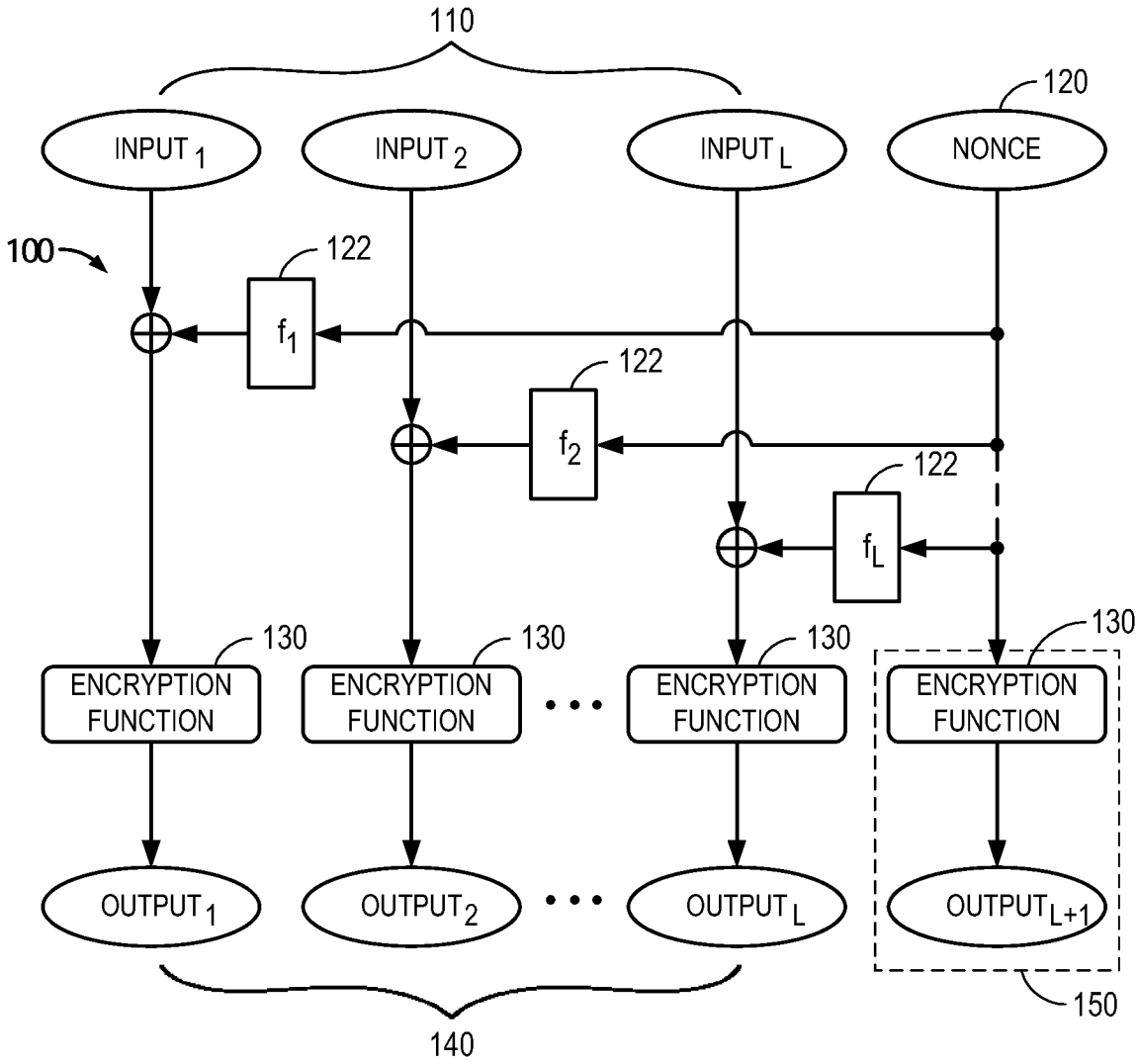
110
120
INPUT 1
INPUT 2
INPUT L
NONCE
100
122
f1
122
f2
122
fL
130
130
130
130
ENCRYPTION FUNCTION
ENCRYPTION FUNCTION
ENCRYPTION FUNCTION
ENCRYPTION FUNCTION
OUTPUT 1
OUTPUT 2
OUTPUT L
OUTPUT L+1
140
150

FIG. 1A

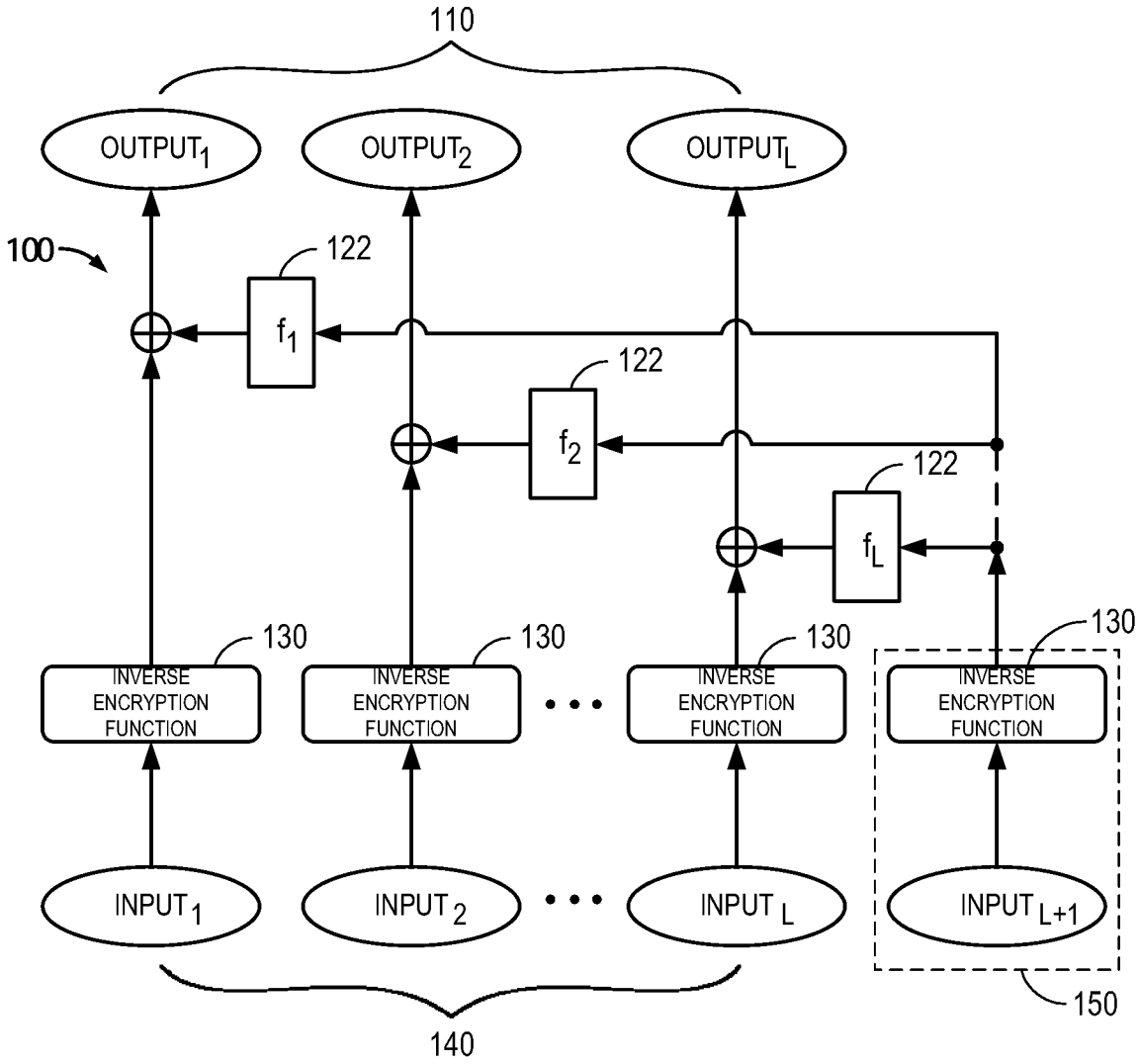
110
OUTPUT1
OUTPUT2
OUTPUTL
100
122
f1
122
f2
122
fL
130
130
130
130
INVERSE ENCRYPTION FUNCTION
INVERSE ENCRYPTION FUNCTION
INVERSE ENCRYPTION FUNCTION
INVERSE ENCRYPTION FUNCTION
INPUT1
INPUT2
INPUT L
INPUT L+1
140
150

FIG. 1B

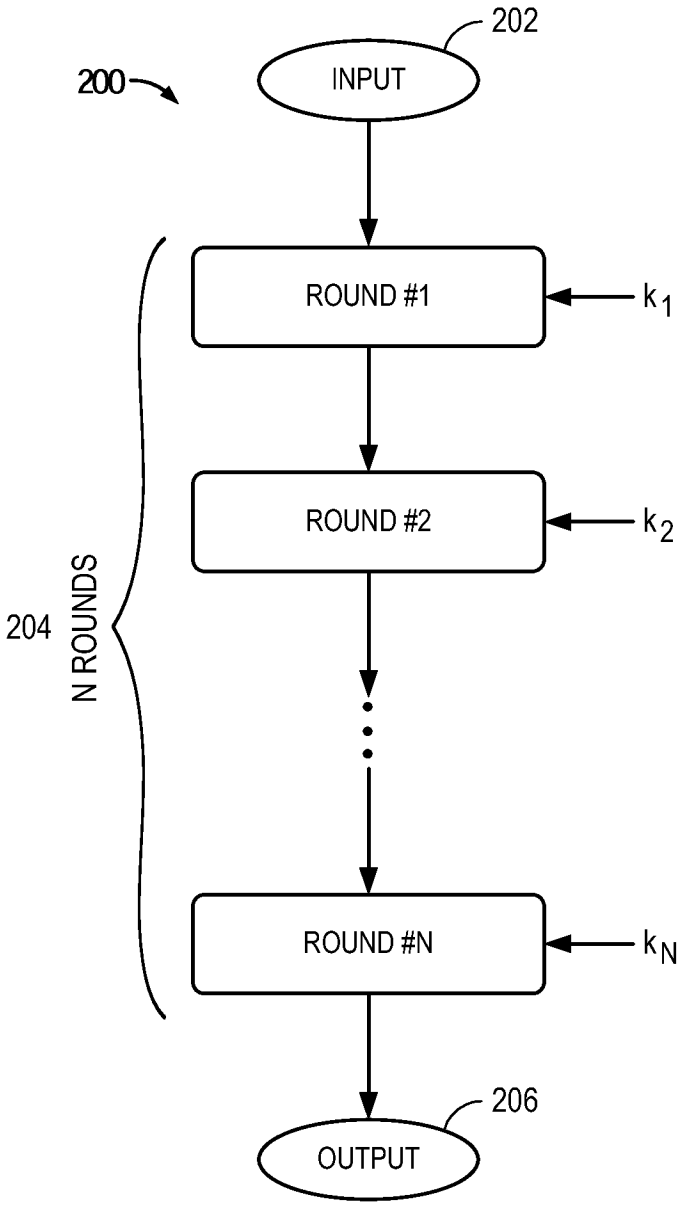
200
202
INPUT
ROUND #1
k_1
ROUND #2
k_2
204
N ROUNDS
ROUND #N
k_N
206
OUTPUT

FIG. 2

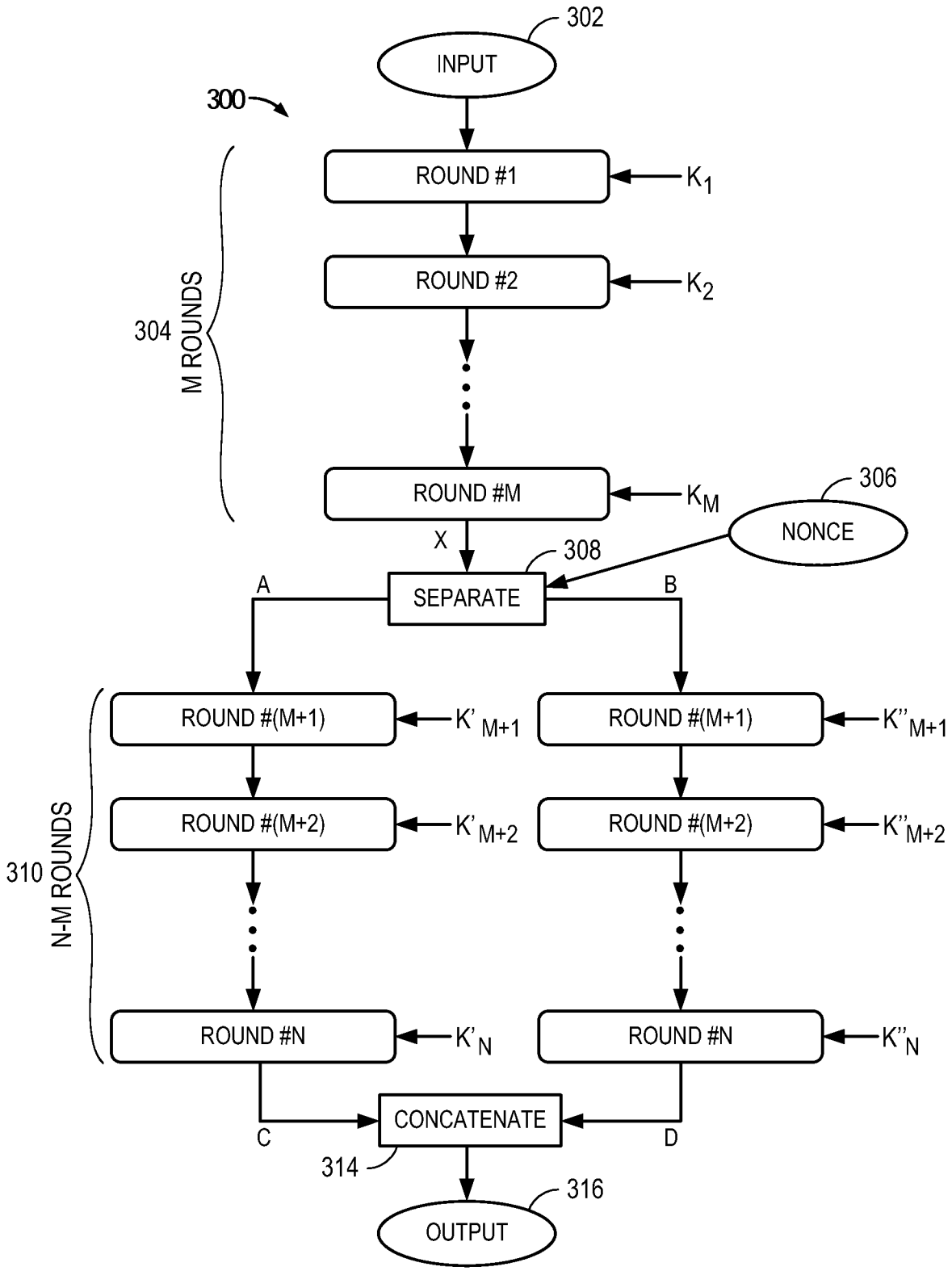
300
302
INPUT
ROUND #1
K_1
ROUND #2
K_2
304
M ROUNDS
ROUND #M
K_M
306
NONCE
X
308
SEPARATE
A
B
ROUND #(M+1)
K'_{M+1}
ROUND #(M+1)
K''_{M+1}
ROUND #(M+2)
K'_{M+2}
ROUND #(M+2)
K''_{M+2}
310
N-M ROUNDS
ROUND #N
K'_N
ROUND #N
K''_N
CONCATENATE
C
D
314
316
OUTPUT

FIG. 3

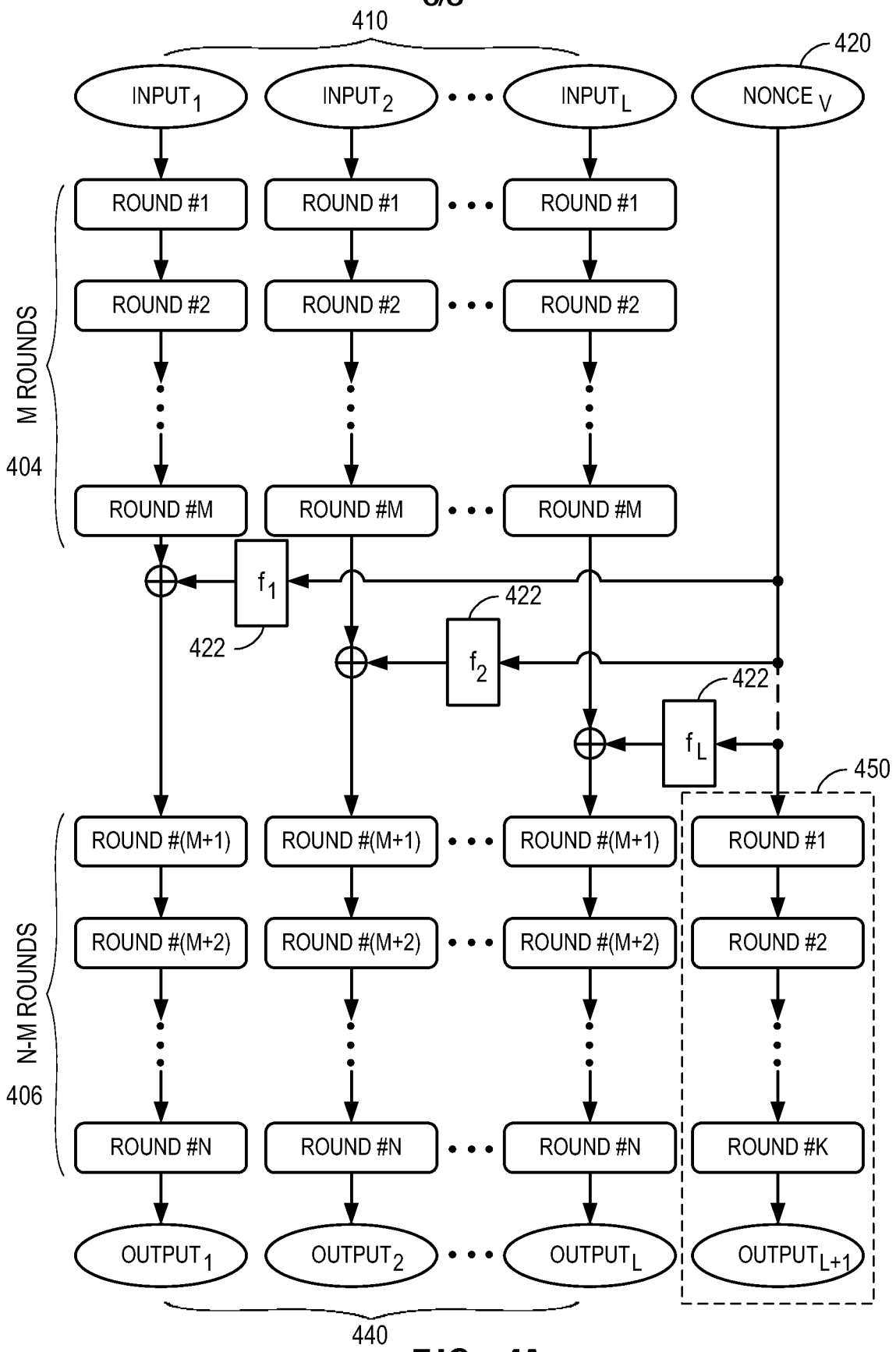
410
420
INPUT 1
INPUT 2
INPUT L
NONCE V
ROUND #1
ROUND #2
ROUND #M
M ROUNDS
404
f 1
f 2
f L
422
422
422
450
ROUND #(M+1)
ROUND #(M+2)
ROUND #N
ROUND #1
ROUND #2
ROUND #K
N-M ROUNDS
406
OUTPUT 1
OUTPUT 2
OUTPUT L
OUTPUT L+1
440

FIG. 4A

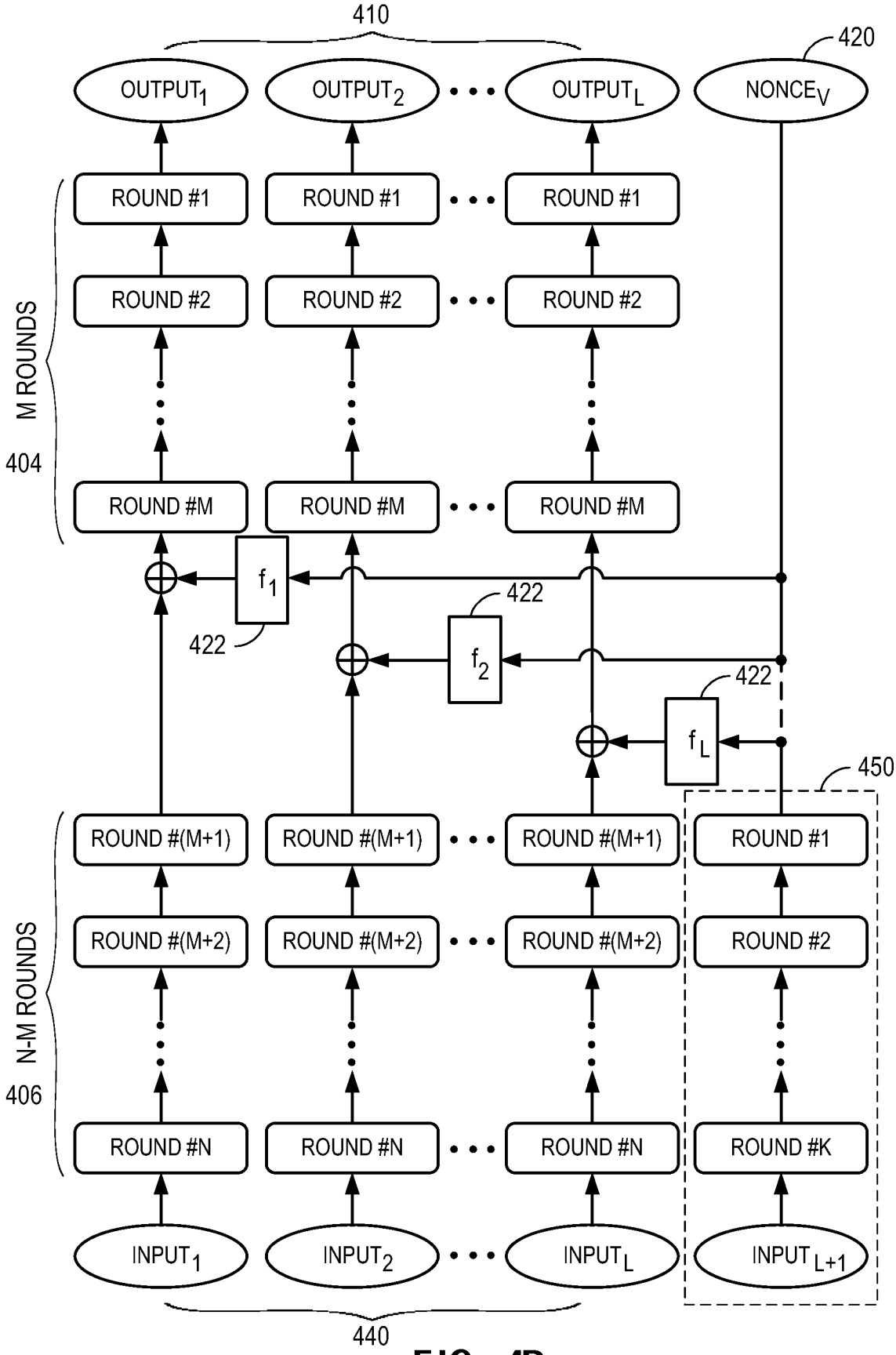
410
420
OUTPUT1
OUTPUT2
OUTPUTL
NONCEV
ROUND #1
ROUND #2
ROUND #M
M ROUNDS
404
f1
f2
fL
422
450
ROUND #(M+1)
ROUND #(M+2)
ROUND #N
ROUND #K
N-M ROUNDS
406
INPUT1
INPUT2
INPUTL
INPUT L+1
440

FIG. 4B

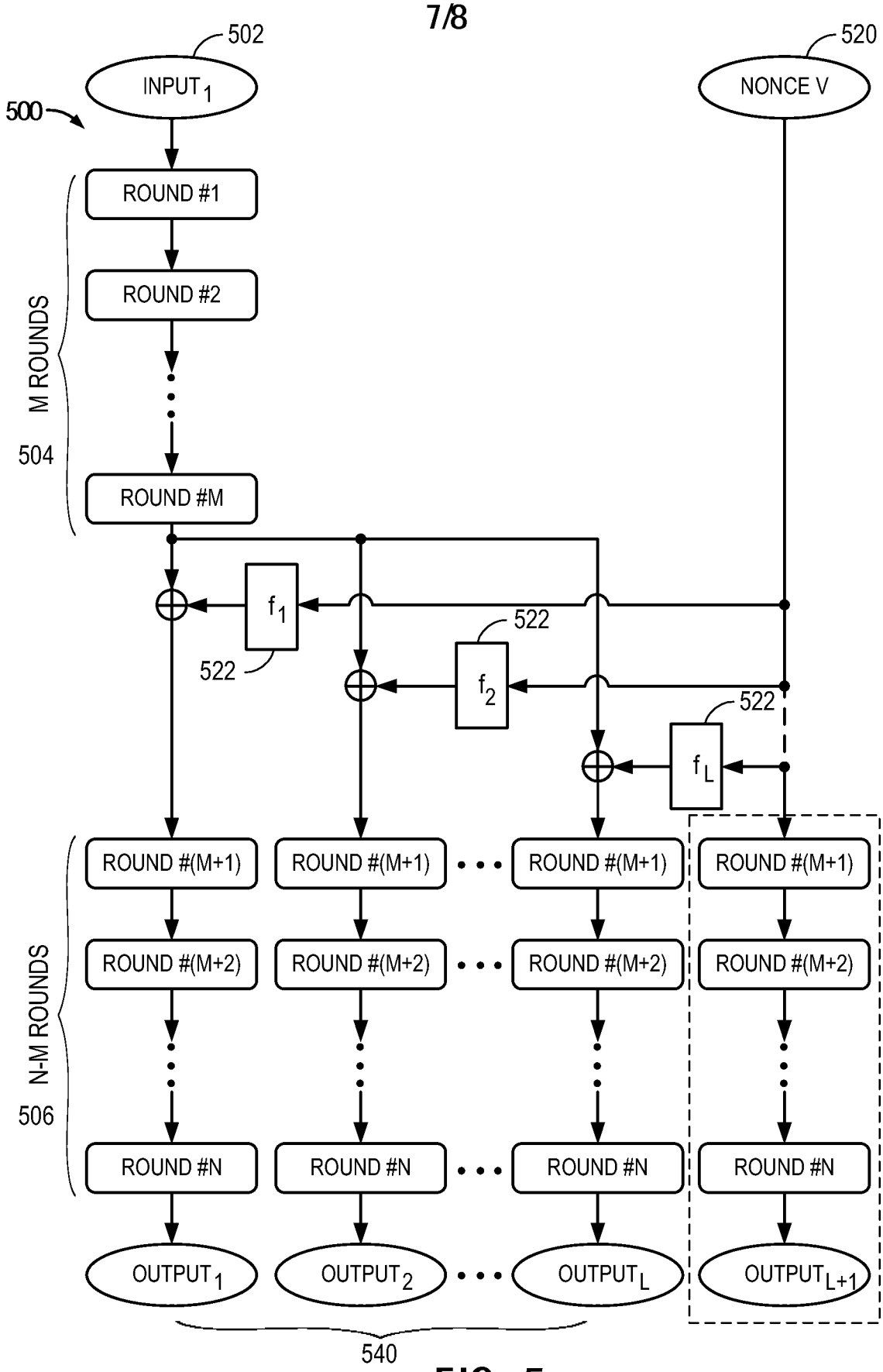
502
INPUT 1
500
520
NONCE V
ROUND #1
ROUND #2
M ROUNDS
504
ROUND #M
f1
522
522
f2
522
fL
ROUND #(M+1)
ROUND #(M+1)
ROUND #(M+1)
ROUND #(M+1)
ROUND #(M+2)
ROUND #(M+2)
ROUND #(M+2)
ROUND #(M+2)
N-M ROUNDS
506
ROUND #N
ROUND #N
ROUND #N
ROUND #N
OUTPUT 1
OUTPUT 2
OUTPUT L
OUTPUT L+1
540

FIG. 5

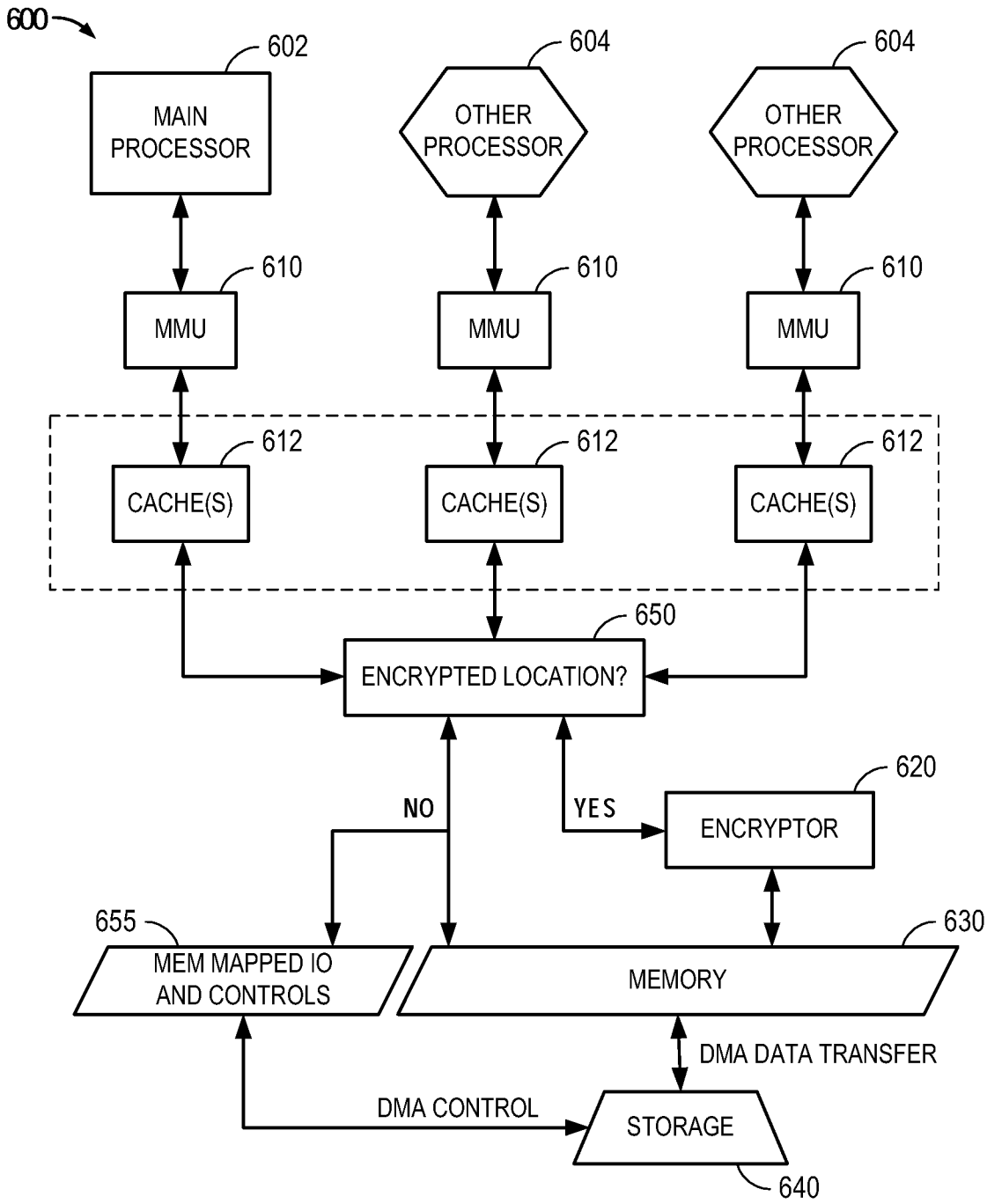
600
602
MAIN PROCESSOR
604
OTHER PROCESSOR
604
OTHER PROCESSOR
610
MMU
610
MMU
610
MMU
612
CACHE(S)
612
CACHE(S)
612
CACHE(S)
650
ENCRYPTED LOCATION?
NO
YES
620
ENCRYPTOR
655
MEM MAPPED IO AND CONTROLS
630
MEMORY
DMA DATA TRANSFER
DMA CONTROL
STORAGE
640

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/043169

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2010/031057 A1 (BEAVER DONALD ROZINAK [US] ET AL) 4 February 2010 (2010-02-04) paragraphs [0002] - [0009], [0037] - [0046]; figure 7 -----	1-8, 11-19, 22-41,44
X	US 2011/191588 A1 (ROGAWAY PHILLIP W [US]) 4 August 2011 (2011-08-04) the whole document -----	1-8, 11-19, 22-41,44
X	US 7 797 751 B1 (HUGHES JAMES P [US] ET AL) 14 September 2010 (2010-09-14) column 1, line 1 - column 2, line 59 -----	1-8, 11-19, 22-41,44

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 March 2015	17/03/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Prins, Leendert

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/043169

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010031057 A1	04-02-2010	NONE	
US 2011191588 A1	04-08-2011	US 2006285684 A1	21-12-2006
		US 2007189524 A1	16-08-2007
		US 2011191588 A1	04-08-2011
		US 2013077780 A1	28-03-2013
US 7797751 B1	14-09-2010	NONE	