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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 March 2008 (13.03.2008)

PCT

(10) International Publication Number
WO 2008/031109 A2

(51) International Patent Classification:
G11C 8/00 (2006.01)

(21) International Application Number:
PCT/US2007/078070

(22) International Filing Date:
10 September 2007 (10.09.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/517,641 8 September 2006 (08.09.2006) US

(71) Applicant *(for all designated States except US)*: **ATMEL CORPORATION** [US/US]; 2329 Orchard Parkway, San Jose, CA 95131 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **FRONTE, Daniel** [IT/FR]; 18 Avenue du Petit Barthelemy, Batiment, Le Champsaur, F-13090 Aix-en-Provence (FR). **PAYRAT, Eric** [FR/FR]; 29, Lotissement la Marie, F-13109 Simiane-Collongue (FR). **PEREZ, Annie**; 5 Rue Louis Gibert, F-13004 Marseilles (FR).

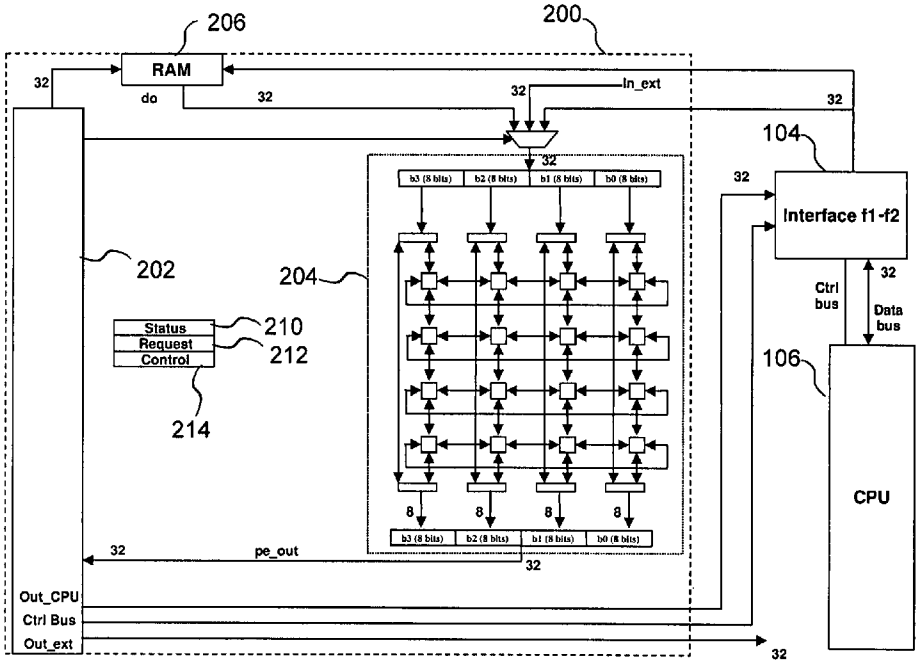
(74) Agents: **SAWYER, Joseph, A.** et al.; Sawyer Law Group LLP, 2465 E. Bayshore Road, Suite No. 406, Palo Alto, CA 94303 (US).

(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, DO, 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, 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*

(54) **Title:** SYSTEM AND METHOD FOR ENCRYPTING DATA



(57) **Abstract:** A system and method for encrypting data. The system includes a controller and a processing element (PE) array coupled to the controller. The PE array is operative to perform one or more of encryption functions and decryption functions using an encryption algorithm. According to the system and method disclosed herein, by utilizing the PE array, the system encrypts and decrypts data efficiently and flexibly.

WO 2008/031109 A2

ABSTRACT

[073] A system and method for encrypting data. The system includes a controller and a processing element (PE) array coupled to the controller. The PE array is operative to perform one or more of encryption functions and decryption functions using an encryption algorithm. According to the system and method disclosed herein, by utilizing the PE array, the system encrypts and decrypts data efficiently and flexibly.

SYSTEM AND METHOD FOR ENCRYPTING DATA

FIELD OF THE INVENTION

[001] The present invention relates to computer systems, and more particularly to a system and method for encrypting data.

BACKGROUND OF THE INVENTION

[002] Data encryption is well known and is typically used to provide security for data that may be transmitted within or across communication systems such as the Internet. Cryptographic chips may be used to implement data encryption functions. For example, a cryptographic chip may perform cryptographic functions involving electronic keys that may be required to execute functions or code on a given device. Such functions may include, for example, accessing data in a memory device. A problem with conventional data encryption solutions is that they themselves may have security vulnerabilities. For example, a cryptographic chip may be vulnerable to side-channel attacks. A side-channel attack is an attack on information gained from implementation of a cryptosystem. Such information may include timing information, electro-mechanical information, power information can be exploited to acquire sensitive data from a system.

[003] Accordingly, what is needed is an improved system and method for encrypting data. The present invention addresses such a need.

SUMMARY OF THE INVENTION

[004] A system for encrypting data is disclosed. The system includes a controller and a processing element (PE) array coupled to the controller. The PE array is operative

to perform one or more of encryption functions and decryption functions using an encryption algorithm.

[005] According to the system and method disclosed herein, by utilizing the PE array, the system encrypts and decrypts data efficiently and flexibly.

BRIEF DESCRIPTION OF THE DRAWINGS

[006] Figure 1 is a block diagram of a system for encrypting data in accordance with the present invention.

[007] Figure 2 is a block diagram of a crypto-processor, which may be used to implement the crypto-processor of Figure 1, in accordance with the present invention.

[008] Figure 3 is a more detailed block diagram of the crypto-processor of Figure 2, in accordance with the present invention.

[009] Figure 4 is a block diagram of the process element (PE) array of Figure 3 in accordance with the present invention.

[010] Figure 5 is a block diagram of the left/east-most column of a PE array, including input/output (I/O) units, which may be used to implement the I/O units of Figure 4, respectively, in accordance with the present invention.

[011] Figure 6 is a block diagram of a PE, which may be used to implement a PE of Figure 4, in accordance with the present invention.

[012] Figure 7 is a block diagram of two PEs in accordance with one embodiment of the present invention.

[013] Figure 8 is a block diagram of the sequencer of Figure 2 in accordance with the present invention.

[014] Figure 9 is a table showing exemplary operation code of the crypto-processor of Figure 2, in accordance with the present invention.

[015] Figure 10 is a flow chart showing a method for encrypting data in accordance with the present invention.

[016] Figure 11 is a block diagram of an Advanced Encryption Standard (AES) matrix in accordance with the present invention.

[017] Figure 12 is a block diagram of a PE matrix in accordance with the present invention.

[018] Figure 13 is a flow chart showing a method for performing cryptographic transformations in accordance with the present invention.

[019] Figure 14 is a block diagram illustrating a sub-byte transformation, which may be used to implement the sub-byte transformation of Figure 13, in accordance with the present invention.

[020] Figure 15 is an exemplary S-Box table in accordance with the present invention.

[021] Figure 16 is a block diagram illustrating a portion of the RAM in accordance with the present invention.

[022] Figure 17 is a block diagram illustrating a shift-row transformation, which may be used to implement the shift-row transformation of Figure 13, in accordance with the present invention.

[023] Figure 18 is a block diagram illustrating a mix-columns transformation, which may be used to implement the mix-columns transformation of Figure 13, in accordance with the present invention.

[024] Figure 19 is an exemplary logarithm table in accordance with the present invention.

[025] Figure 20 is an exemplary anti-logarithm table in accordance with the present invention.

[026] Figure 21 is a block diagram of the finite state machine (FSM) of Figure 8, in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[027] The present invention relates to computer systems, and more particularly to a system and method for encrypting data. The following description is presented to enable one of ordinary skill in the art to make and use the invention, and is provided in the context of a patent application and its requirements. Various modifications to the preferred embodiment and the generic principles and features described herein will be readily apparent to those skilled in the art. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features described herein.

[028] A system and method in accordance with the present invention for encrypting data are disclosed. The system includes a cryptographic processor that is built around a systolic array of processing elements (PEs). In one embodiment, a systolic array is a matrix of processors that substantially simultaneously execute the same operations. In another embodiment, the systolic array may separately perform different operations. The cryptographic processor may encrypt or decrypt data by using an encryption algorithm. In one embodiment, the cryptographic processor architecture implements the Advanced Encryption Standard (AES) algorithm to encrypt/decrypt data. By utilizing the systolic array of PEs, the cryptographic processor encrypts and decrypts data efficiently and flexibly. To more particularly describe the features of the present invention, refer now to the following description in conjunction with the accompanying figures.

[029] Although the present invention disclosed herein is described in the context of an AES algorithm, the present invention may apply to other types of encryption algorithms, as well as other applications, and still remain within the spirit and scope of the present invention.

[030] Figure 1 is a block diagram of a system 100 for encrypting data in accordance with the present invention. The system 100 includes a cryptographic processor (or crypto-processor) 102, an interface unit 104, and a central processing unit (CPU) 106. In operation, the crypto-processor 102 encrypts and/or decrypts data by using an encryption algorithm, which in one embodiment may involve the AES algorithm to encrypt/decrypt data. One embodiment of the crypto-processor 102 is described in more detail below in Figure 2.

[031] Figure 2 is a block diagram of a crypto-processor 200, which may be used to implement the crypto-processor 200 of Figure 1, in accordance with the present invention. The crypto-processor 200 includes a controller or sequencer 202, a processing elements (PE) array 204, and a local memory 206. In operation, the sequencer 202 controls the PE array 204 and also manages data exchanged between the PE array 204 and the CPU 106 (Figure 1). Accordingly, the PE array 204 is configured to receive input commands and data (to encrypt/decrypt) from the sequencer 202 and is also configured to receive/transmit encrypted/decrypted data to the sequencer 202. In one embodiment, the local memory 206 may be a random access memory (RAM) unit or any other suitable memory unit. One embodiment of the sequencer 202 is described in more detail below in connection with Figure 8.

[032] Figure 3 is a more detailed block diagram of the crypto-processor 200 of Figure 2, in accordance with the present invention. The crypto-processor 200 includes the sequencer 202, the PE array 204, the local memory (e.g., RAM) 206, a status register 210, a request register 212, and a control register 214. The crypto-processor 200 is operatively coupled to the interface unit 104 and the CPU 106. In operation, the crypto-processor 102 may exchange data with both the CPU 106 and/or an external device (not shown). The external device may be, for example, a network card that transmits/receives a data stream to be encrypted/decrypted by the crypto-processor 200. In one embodiment, the crypto-processor 200 may use a different clock from that of the CPU 106. The presence of a control interface between the CPU 106 and the crypto-processor 200 allows the use a different clock (e.g., higher clock frequency for the crypto processor 200 than for the CPU 106). Accordingly, the interface unit 104 enables communication between the CPU 106.

[033] Because the crypto-processor 200 utilizes the PE array 204 to encrypt or decrypt data, the crypto-processor 200 is protected against side-channel attacks. As described above, a side-channel attack is an attack on information gained from implementation of a cryptosystem. For example, information such as timing information, electro-mechanical information, and power information can be exploited to acquire sensitive data from a system. In accordance with the present invention, the PE array 204 is operative to prevent side-channel attacks because the PE elements substantially simultaneously perform the same functions as one another and create noise (e.g., electrical and algorithmic noise). In one embodiment, the PE array is operative to perform operations that generate noise to distract or eliminate the ability of an agent from detecting information sensitive data.

Processor Elements Array

[034] Figure 4 is a block diagram of the PE array 204 of Figure 3 in accordance with the present invention. The PE array 204 includes PEs 220a, 220b, 220c, 220d, etc., 2-dimensional input/output (2D I/O) structures 222 and 224, and I/O units 226 and 228, etc. As Figure 4 illustrates, in one embodiment, the PE array 204 is configured in a 4X4 matrix. Although the present invention disclosed herein is described in the context of a PE array having 16 PEs arranged in a 4x4 matrix, the present invention may apply to any number of PEs having other configurations, and still remain within the spirit and scope of the present invention. In one embodiment, all of the PEs are identical such that they perform the same operations. This enables the crypto-processor 200 to function more efficiently. As Figure 4 illustrates, in one embodiment, the PEs 220 are coupled to each other in a tore fashion. A tore fashion means that the eastern frontier of the array is linked with the western frontier (like in a

cylindrical fashion, with the main axis parallel with the north-south direction), and the northern frontier of the array is linked with the southern frontier (like in a cylindrical fashion, with the main axis parallel with the east-west direction). Also, in one embodiment, each PE 220 has 4 data input/outputs (I/Os) to/from each cardinal direction: north, east, south, and west, where the 4 data I/Os connect to 4 respective neighboring PEs. In one embodiment, each I/O data is 1 byte. In one embodiment, the PEs 220 receive data from one cardinal direction (e.g., from the north) and transmits data to another cardinal direction (e.g., to the South). Figure 5 is a block diagram of the left/east-most column of a PE array, including I/O structures 502 and 504, which may be used to implement the I/O structures 226 and 228 of Figure 4, respectively, in accordance with the present invention.

[035] In one embodiment, the PEs 220 may perform the same operations and may perform these operations substantially simultaneously. This is the case while in a Single Instruction Multiple Data (SIMD) mode. In another embodiment, the PEs 220 may separately perform different operations. This is the case in a Multiple Instruction Multiple Data (MIMD) mode. For instance, data may be shifted in all of the PEs 220 except for those stored in one line of the PE array 204. As described in more detail below, an "Enable_PE" signal enables or disables the PEs 220. For example, the "Enable_PE" signal may activate the PEs for a shift operation only, and disable the others. In one implementation, the enable signals may be managed in order to increase the algorithmic noise. For instance, one shift operation of data of the PE array may be split into two arrays, thereby managing enable signals twice. This may in some cases slow down the operation, but it enables the tracking of a side-channel hacker.

Processor Element

[036] Figure 6 is a block diagram of a PE 600, which may be used to implement a PE 220 of Figure 4, in accordance with the present invention. The PE 600 includes I/O ports 602, 604, 606, and 608 (having respective I/O enable units 612, 614, 616, and 618), input-select multiplexor (or mux) 620, an operations unit 622, which includes a clock (not shown). The PE 600 also includes a register mux 624, a register 626, and an output mux 628. As described above, referring again to Figure 3, the PE array 204 is configured to receive input commands and data (to encrypt/decrypt) from the sequencer 202, and is also configured to receive and transmit encrypted/decrypted data to the sequencer 202. The I/O enable units 612, 614, 616, and 618 control whether their respective I/O ports 602, 604, 606, and 608 function as inputs or outputs to receive or transmit instructions or data, and the input-select mux 610 selects appropriate I/Os inputs from which to receive instructions and/or data for encryption/decryption. The operations unit 612 receives the instructions and data for encryption/decryption and then encrypts or decrypts the data using an encryption algorithm, the process of which is described in detail below, beginning at Figure 10.

[037] In operation, the input-select mux 620 performs a select-PE-in function, where the input-select mux 620 selects an input from which to load data (e.g., from a northern, eastern, western or southern input). The operations unit 622 performs various functions/instructions used in data encryption and decryption. For example, the operations unit 622 may perform select operations, where the operations unit 622 selects exclusive-OR (XOR) and addition operations. The operations unit 622 performs encryption-related and/or decryption-related operations such as read/write operations. For example, in a shift_W2E operation, the operations unit 622 may read from a western/eastern PE and/or may write to a western/eastern PE. In a shift_N2S

operation, the operations unit 622 may read from a northern/southern PE and/or write to a northern/southern PE. The register mux 624 performs a select-register-in function, where the register 624 selects between data computed by the operations unit 622 or previously computed data (e.g., from another PE). Some of the data may be stored in the register 626. The output mux 628 performs a select-PE-out function, where the output mux 628 selects previously registered data or non-registered data stored in the register 626.

[038] Figure 7 is a block diagram of two PEs 600a and 600b, in accordance with one embodiment of the present invention. In one embodiment, a given PE 600 may execute one instruction at a time. When a control signal is activated, the PE 600 determines whether other instructions are in progress. If not, the PE 600 performs the requested instruction. Otherwise, the PE 600 performs the previous instruction. Two algebraic operations are available: the "XOR" and the "addition" over 8 bits, between the data registered data and the incoming data.

[039] In one embodiment, each PE 600 performs the following inner functions/instructions (of no particular order):

- Load data from near-neighbor cells;
- Load data from the RAM, or the CPU, or an external source (for array frontier's PE only); Referring to Figure 1200, examples of the array frontier's PEs may include:

PE_{11}, PE_{12}, PE_{13}, PE_{14} (northern frontier)

PE_{41}, PE_{42}, PE_{43}, PE_{44} (southern frontier)

[040] Although the present invention disclosed herein has been described in the context of the inputs of the PE array being located at the northern frontier and the

outputs of the PE array being located at the southern frontier, the inputs and outputs of the PE array may be at any other array frontier, and still remain within the spirit and scope of the present invention. For example, the inputs may alternatively be located at the southern array frontier, the western array frontier (PE_{11}, PE_{21}, PE_{31}, PE_{41}), or eastern array frontier (PE_{14}, PE_{24}, PE_{34}, PE_{44}). Similarly, the outputs may alternatively be located at the northern, western, or eastern array frontiers.

- Write data into near neighbor cells;
- Write data into the RAM, or the CPU, or an external source (for array frontier's PE only);
- XOR; and
- Addition.

[041] If the algorithm requires that an XOR operation be performed on the output values of the PEs 600a and 600b (same row, different column) and that the result be stored the in the PE 600b, the sequencer 202 may provide the following commands: enable PE 600a and PE 600b (disable all other PEs), select_PE_out on registered data, shift_W2E, and select_PE_in on the east input.

The Sequencer

[042] Figure 8 is a block diagram of the sequencer 202 of Figure 2 in accordance with the present invention. In one embodiment, the sequencer 202 includes finite state machine (FSM) 216 and storage elements 218. In operation, the sequencer 202 generates command signals for the PE array 204 and the instructions for the CPU 106. The sequencer 202 also manages, exchanges, and transmits data for the PE array 204.

[043] In one implementation, the status, request, and control registers 210, 212, and 214 function as a communication interface between the CPU 106 and the crypto processor 200. The status and request registers 210 and 212 are utilized when the

sequencer 202 makes a request to the CPU 106. The following is an example of an exchange protocol:

• the sequencer 202 writes the type of the request for the CPU 106 to perform (e.g., a memory access, a computation, etc.), and the request is written into the request register 212 of the communication interface;

• the request register 212 is read by the CPU 106;

• the CPU 106 writes its status (e.g., free, busy, acknowledge) into the status register 210; when the request is accomplished, the CPU 106 sets the status register 210 to the acknowledge status; and

• the sequencer 202 reads the status of its request in the status register 210.

The control register 214 is set by the sequencer 202, and the control register 214 gives its status (e.g., work in progress, encryption/decryption complete, etc.) to the CPU 106.

[044] Figure 9 is a table showing exemplary operation code of the crypto-processor 200 of Figure 2, in accordance with the present invention. In one embodiment, the crypto-processor 200 performs the following operations.

[045] In one embodiment, a load row operation copies a 32-bit word from the CPU 106, the RAM 206 or from an external source, and write it into a row of the PE array 204. In one embodiment, parameters may include:

• "address": if the RAM 206 is selected as source;

• "XOR": to XOR the word with the word previously stored in the selected row of the PE array 204; and

• "add": to concatenate the word with the word previously stored in the selected row of the PE array 204.

[046] In one embodiment, a load column operation copies a word (e.g., a 32-bit word) from the CPU 106, the RAM 206 or from an external source, and write it into a column of the PE array 204. In one embodiment, parameters may include:

• "address": if the RAM 206 is selected as source;

• "XOR": to XOR the word with the word previously stored in the selected column of the PE array 204;

• "add": to concatenate the word with the word previously stored in the selected column of the PE array 204.

[047] In one embodiment, a load index A operation copies a word (e.g., a 32-bit word) from the RAM 206 at the address given by the "address" parameter + the offset given register A, into a row of the PE array 204. In one embodiment, parameters may include:

• "Address";

• "XOR": to XOR the word with the word previously stored in the selected row of the PE array 204;

• "add": concatenate the word with the word previously stored in the selected row of the PE array 204.

[048] Store Row: copy a word (e.g., a 32-bit word) from a row of the PE array 204 and write it into the CPU 106, the RAM 206, or from an external source. In one embodiment, parameters may include:

• "address": if the RAM 206 is selected as destination; and

• "CPU_request": if the CPU 106 is selected as destination.

[049] Store Column: copy a word (e.g., a 32-bit word) from a column of the PE array 204 and write it into the CPU 106, the RAM 206 or from an external source. In one embodiment, parameters may include:

- "address": if the RAM 206 is selected as destination; and
- "CPU_request": if the CPU 106 is selected as destination.

[050] Store index A: write a word (e.g., a 32-bit word) to the register A). The sequencer 202 includes several registers. In one embodiment, the sequencer 202 may include the following registers:

- A, 32 bits;
- B, 10 bits;
- I, 32 bits, it contains the micro instruction which is executed;
- Counter_address, 10 bits, to address up to 1024 memory locations; and
- Current-state, Next_state, each one of 5 bits, they are needed by the FSM.

Sequencer inputs and outputs

[051] In one embodiment, the inputs of the sequencer 202 accept data from up to three possible sources:

- a 32-bit word from the PE array 204;
- a 32-bit word from the RAM 206; and
- the acknowledge signal from the CPU 106 (status register).

[052] In one embodiment, the sequencer 202 outputs the following data:

- Clock (1 bit);

• Sel_RAM: to select the input of the RAM 206 between the Data_out (from the sequencer itself), the output of the CPU and the pe_out (the output of the PE Array 204);

• Address: to address the RAM 206;

• Sel_PE-Array_in: to select the input of the PE array 200 between the RAM 206, the CPU, and the external source;

• Load: to load data from pe_in (the source is selected by Sel_PE-array_in) into the PE (the ones that are enabled);

• Enable: to enable the PE. The width of this bus is 4x4 bits (one wire for each PE);

• shift W2E: to shift the byte of each PE into its neighbor, from west to east (or vice-versa), only for enabled PE;

• shift N2S: to shift the data of each PE into its neighbor, from north to south (or vice-versa), only for enabled PE;

• Sel_mux: to manage the muxes inside each PE;

• Sel_PE-Array_out: to select the output of the PE-Array between the RAM 206, the CPU, and the external device (via the sequencer);

• Control (bus): to manage the communication with the CPU 106;

• Request (register): to manage the communication with the CPU 106; and

• Control (register): to manage the communication with the CPU 106.

Data Encryption by the Crypto-Processor

[053] In one embodiment, the sequencer 202 reads the instruction to be executed from the RAM 206. The core of the sequencer 202 contains an FSM 216 (Figure 8) that manages the output of the RAM 206. The data stored in the RAM 206 may have the format described by the Table shown in Figure 9. In one embodiment, some of

the operational code may be described as follows, where the "x" most significant bits (MSBs) code the instructions, which are executed by the FSM 216:

Operation code	Source	Destination	Parameters
X bits (MSB)	n-x bits (LSB)		
$b_n\ b_{n-1}\ b_{n-2} \ldots b_{n-x+1}$	$b_{n-x}\ b_{n-x-1} \ldots b_1\ b_0$		

[054] Figure 10 is a flow chart showing a method for encrypting data in accordance with the present invention. Referring to both Figures 3 and 10 together, the process begins in a step 1002 where the crypto-processor 200 provides a PE array 204 for storing data. Next, in a step 1004, the crypto-processor 200 performs one or more of encryption functions and decryption functions using an encryption algorithm. In one embodiment, the crypto-processor 200 performs cryptographic transformations on the data stored in the PE array. The cryptographic transformations are described in detail below in connection with Figure 13.

[055] Figure 11 is a block diagram of an AES matrix 1100 in accordance with the present invention. In one embodiment, the AES matrix includes 128-bit data blocks arranged in a 4x4 matrix. The AES specifies a Federal Information Processing Standards (FIPS) approved cryptographic algorithm that may be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits. These different versions may be referred to as AES-128, AES-192 and AES-256, respectively. In one embodiment, the crypto-processor 102 utilizes AES-128 as the AES algorithm, where the plain text consists of 128-bit data blocks and each block may be managed as a matrix of 4x4 bytes. Although the present invention disclosed herein is described in the context of

128-bit data blocks, the present invention may apply to other size data blocks (e.g., 192-bit, 256-bit, etc.) and still remain within the spirit and scope of the present invention.

[056] Figure 12 is a block diagram of a PE matrix 1200 in accordance with the present invention. As Figure 12 illustrates, the PE matrix data block is arranged in a 4x4 matrix. In one embodiment, the crypto-processor 102 maps AES transformations with the PE array, where each element of the AES matrix 1100 may be mapped to a PE of the PE matrix 1200. As such, every PE is mapped to a byte of plain text. In one embodiment, referring to Figures 4 and 12, a byte of data stored in a PE (e.g., PE_{11}) of Figure 12 corresponds to a byte of data stored in a PE (e.g., 220a) of Figure 4.

[057] Figure 13 is a flow chart showing a method for performing cryptographic transformations in accordance with the present invention. Referring to both Figures 3 and 13 together, the process begins in a step 1302 where the crypto-processor 200 performs a sub-bytes transformation. Next, in a step 1304, the crypto-processor 200 performs a shift-rows transformation. Next, in a step 1306, the crypto-processor 200 performs a mix-columns transformation. Finally, in a step 1308, the crypto-processor 200 performs an add-round-key transformation. In one embodiment, the encryption process includes 10 rounds, where each round involves the above transformations. The following paragraphs describe each transformation in detail.

Sub-Bytes Transformation

[058] Figure 14 is a block diagram illustrating a sub-byte transformation, which may be used to implement the sub-byte transformation of box 1302 of Figure 13, in

accordance with the present invention. Each byte stored in the PE array 1400 is substituted with a corresponding byte in the substitution box table (S-Box table) 1402. An S-Box is a non-linear substitution table used in the Sub-Byte transformations. In one embodiment, the S-Box table may be stored in the RAM 206. Figure 15 is an exemplary S-Box table in accordance with the present invention. In one embodiment, the FSM 216 (Figure 8) may manage one word of 4 bytes and may provide all needed commands for the substitution of all 4x4 1-byte data. In one embodiment, the sequencer 202 selects the PE array output (e.g., at its Sel_PE-Array_out output). For each received word, the FSM 216 substitutes a single byte at a time. For each byte in the PE array, the FSM 216 searches for a corresponding byte in the S-Box table. In one embodiment, since the S-Box table is stored in the RAM, the FSM 216 searches the following address: address = Sbox_table_address + offset. The offset is given by the byte that needs a sub-byte transformation.

[059] Figure 16 is a block diagram illustrating a portion of the RAM 206 in accordance with the present invention. If the byte "10(16)" is to be substituted, for example, the S-Box tables are recorded at the address "FF00(16)": address = FF00 + 10 = FF10. The following is an example of a S-Box transformation. In the expression RAM [$FF10_{(16)}$] = $CA_{(16)}$, both 10_H and 10_{16} mean 10 in hexadecimal format, i.e., 16 in decimal format. The data 10_H is substituted using the S-Box table. The S-Box table substitutes 10_H with CA_H. In this example, S-Box tables are stored in the RAM at the address $FF00_H$. The FSM reads the RAM at the address given at the statement 56. More precisely, the FSM activates the enable_ram signal, and computes the RAM'S address in the following way: Address = $FF00_H + 10_H$ = $FF10_H$. In the RAM, at the above address, there is a written the value CA_H, because

the 17th data of the S-Box table is CA_H (the table starts at 0, so the data 'n' is the 'n-1' element, see the figure 15). Accordingly, RAM [$FF10_H$] = CA_H.

[060] Then data RAM[address] is written into the PE-array. In one embodiment, the FSM 216 actives a Sel_PE-Array_in function to select the RAM 206 as input to the PE array 204 and actives a shift_NS function to shift all bytes from north to south. As such, the FSM 216 may write and read simultaneously to or from the PE array 204. These instructions are repeated until all 4x4 bytes are substituted.

Shift-Rows Transformation

[061] Figure 17 is a block diagram illustrating a shift-row transformation, which may be used to implement the shift-row transformation of box 1304 of Figure 13, in accordance with the present invention. In one embodiment, each row of the matrix 1400 is left shifted by 0, 1, 2, or 3 positions, respectively, for rows 0, 1, 2 or 3. In one embodiment, a row of PEs 220 in the PE array 204 is wrapped around in a cylindrical fashion (e.g., s03 is connected to s00, s13 is connected to s10 etc.) Each row of the PE array 204 may be considered a circular shift register, which is particularly useful in the Shift-Rows transformation. In one embodiment, the FSM 216 activates the Shift_W2E signal to shift all bytes from west to east of PE array 204. The Shift_W2E may be active during 4 cycles. Also, the enable signals activate only the PE that need to be shifted (e.g., at the first cycle only the 2^{nd}, 3^{rd}, and 4th rows of the PE are enabled; at the second cycle only the 3rd and 4th rows of the PE are enabled; and at the third cycle the 4th row of the PE is enabled).

Mix-Columns Transformation

[062] Figure 18 is a block diagram illustrating a mix-columns transformation, which may be used to implement the mix-columns transformation of box 1306 of Figure 13, in accordance with the present invention. As Figure 18 illustrates, the mix-column transformation operates on the state column-by-column. The mix-columns transformation linearly combines all the data in each whole column. More specifically, in one embodiment, 4 vectors are applied to transform the 4 columns linearly.

[063] The crypto-processor 200 performs the following matrix multiplication:

$$S'(x) = A(x) * S(x) \qquad (1)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \qquad (2)$$

[064] S(x) is the data transformed by the PE array 204, and A(x) is the matrix of multiplicative vectors, which is shown in (2). The above multiplication may be performed by using logarithm and anti-logarithm tables. For example,

$$c = a * b \qquad (3)$$

[065] can be computed by using logarithm tables in the following way:

$$c = \text{Log}^{-1}((\text{Log } a) + (\text{Log } b)) \qquad (4)$$

[066] The crypto-processor 200 exploits logarithms in order to perform (4). Figure 19 is an exemplary logarithm table, and Figure 20 is an exemplary anti-logarithm table, in accordance with the present invention. In one embodiment, the mix-columns transformation computes each row separately. In order to compute the matrix multiplication of expression (1) and exploiting the expression (4), all of the bytes of

the PE array 204 are substituted by using the logarithm tables (addition rather than a multiplication).

[067] Figure 21 is a block diagram of the FSM 216 of Figure 8, in accordance with the present invention. In one embodiment, the FSM 216 extracts the logarithm of all bytes in the PE array 204 and writes the substituted bytes to the PE array 204. A copy of the logarithms is also written to the RAM 206 for further computations. For example, the multiplicative vectors, shown in the matrix (2) above are copied and added from the RAM 206 into the PE array 204. At this point, all of the bytes in the PE array 204 may be XORed by columns and written into the first row of PEs. In one embodiment, in order to do that, all bytes are shifted from southern to northern PEs. For example, in order to compute the data of PE11, the FSM 216 provides commands in order to XOR the data stored in PE21 with the data stored in PE11. The FSM 216 then stores the result into PE11. The data stored in PE31 is then copied and XORed with the data stored in PE11. Similarly, the data stored in PE41 is XORed with the data stored in PE11. Next, PE11 is computed:

$$PE_{11} = PE_{11} \wedge PE_{21} \wedge PE_{31} \wedge PE_{41}$$

[068] In one embodiment, the computation of the data stored in PE11 may be substantially simultaneously computed with the data stored in PE12, PE13, and PE14. Next, the first row of the PE array 204 is computed and the FSM 216 may compute the other three rows. The results of these computations may be stored in the RAM 206. In order to do so, the FSM 216 may copy the PE logarithms previously saved in the RAM 206. Once all 4x4 bytes will be computed, according to the expression (4) above, the FSM 216 may substitute them by using the antilogarithm tables.

Add-Round-Key Transformation

[069] During the add-round-key transformation of box 1308 of Figure 13, the final transformation of a given round, combines the key value with the transformed data. In one embodiment, the keys are loaded from the RAM 206 into the PE array 204 and XORed with data stored in the PE array 204. In one embodiment, the FSM 216 selects the RAM 206 as the source of the PE array 204 and enables PEs row-by-row to receive the keys, XORing them.

[070] According to the system and method disclosed herein, the present invention provides numerous benefits. For example, embodiments of the present invention are efficient, flexible, and secure.

[071] A system and method for encrypting data has been disclosed. The system includes a cryptographic processor that is built around a systolic array of PEs. In one embodiment, a systolic array is a matrix of processors that substantially simultaneously execute the same operations. The cryptographic processor may encrypt or decrypt data by using an encryption algorithm. In one embodiment, the cryptographic processor architecture implements the AES algorithm to encrypt/decrypt data. By utilizing the systolic array of PEs, the cryptographic processor encrypts and decrypts data efficiently and flexibly.

[072] The present invention has been described in accordance with the embodiments shown. One of ordinary skill in the art will readily recognize that there could be variations to the embodiments, and that any variations would be within the spirit and scope of the present invention. For example, the present invention can be implemented using hardware, software, a computer readable medium containing program instructions, or a combination thereof. Software written according to the

present invention is to be either stored in some form of computer-readable medium such as memory or CD-ROM, or is to be transmitted over a network, and is to be executed by a processor. Consequently, a computer-readable medium is intended to include a computer readable signal, which may be, for example, transmitted over a network. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.

CLAIMS

What is claimed is:

1. A system comprising:

a controller; and

a processing element (PE) array coupled to the controller, wherein the PE array is operative to perform one or more of encryption functions and decryption functions using an encryption algorithm.

2. The system of claim 1 wherein the PE array is systolic.

3. The system of claim 1 wherein the PE array comprises a matrix of processors that substantially simultaneously execute the same operations.

4. The system of claim 1 wherein the PE array comprises a matrix of processors, wherein each processor comprises:

a multiplexor for selecting an input from which to load data; and

an operations unit for performing one or more of encryption-related and decryption-related operations.

5. The system of claim 1 wherein the PE array is operative to prevent side-channel attacks.

6. The system of claim 1 wherein the PE array is operative to perform operations that generate noise to distract or eliminate the ability of an agent from detecting information sensitive data.

7. The system of claim 1 wherein the encryption algorithm is based at least in part on an Advanced Encryption Standard (AES) algorithm.

8. The system of claim 1 wherein the controller is a sequencer.

9. A method comprising:

providing a processing element (PE) array; and

utilizing the PE array to perform one or more of encryption functions and decryption functions using an encryption algorithm.

10. The method of claim 9 wherein the utilizing comprises performing cryptographic transformations on data.

11. The method of claim 10 wherein the cryptographic transformations performing comprises:

performing at least one sub-bytes transformation;

performing at least one shift-rows transformation;

performing at least one mix-columns transformation; and

performing at least one add-round-key transformation.

12. The method of claim 9 wherein the PE array is systolic.

13. The method of claim 9 further comprising utilizing the PE array to prevent side-channel attacks.

14. The method of claim 9 wherein the encryption algorithm is based at least in part on an Advanced Encryption Standard (AES) algorithm.

15. A computer-readable medium containing program instructions for conserving power, the program instructions which when executed by a computer system cause the computer system to execute a method comprising:

providing a processing element (PE) array; and

utilizing the PE array to perform one or more of encryption functions and decryption functions using an encryption algorithm.

16. The computer-readable medium of claim 15 wherein the utilizing comprises program instructions for performing cryptographic transformations on data.

17. The computer-readable medium of claim 16 wherein the cryptographic transformations performing comprises program instructions for:

performing at least one sub-bytes transformation;

performing at least one shift-rows transformation;

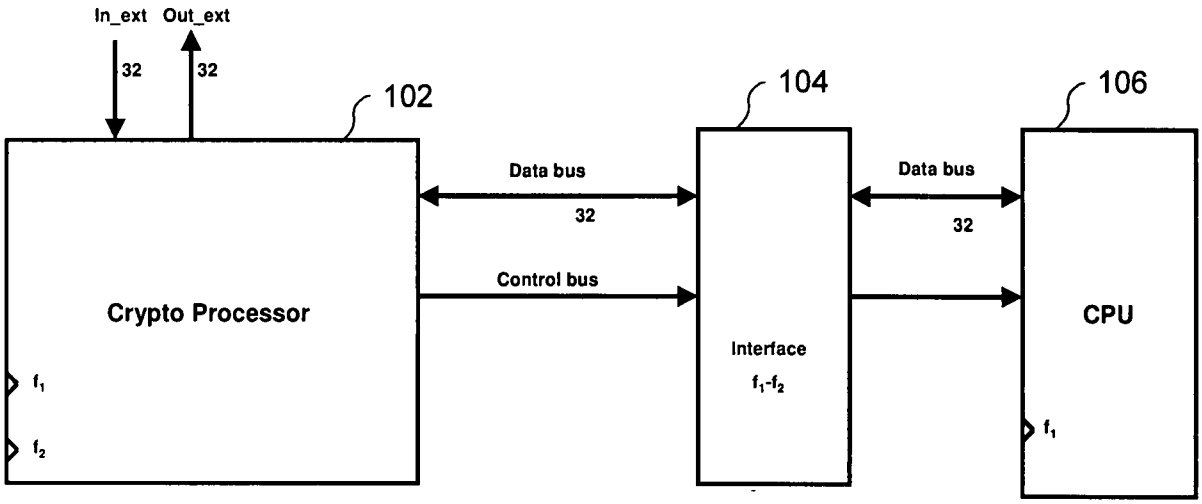
performing at least one mix-columns transformation; and

performing at least one add-round-key transformation.

18. The computer-readable medium of claim 15 wherein the PE array is systolic.

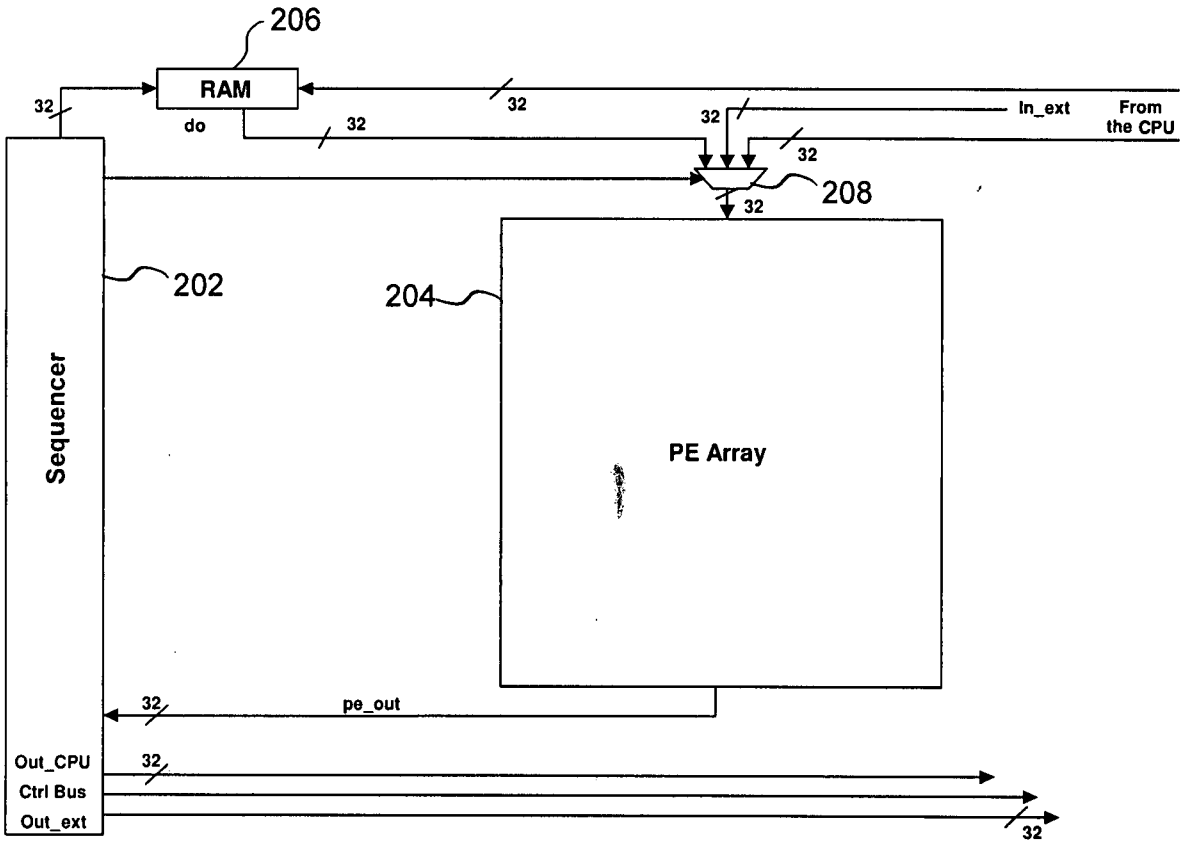
19. The computer-readable medium of claim 15 further comprising program instructions for utilizing the PE array to prevent side-channel attacks.

20. The computer-readable medium of claim 15 wherein the encryption algorithm is based at least in part on an Advanced Encryption Standard (AES) algorithm.


In_ext
Out_ext
32
32
102
104
106
Data bus
32
Data bus
32
Control bus
Crypto Processor
Interface
f_1-f_2
CPU
f_1
f_2
f_1

100

FIG. 1


206
RAM
32
do
32
32
32
In_ext
From the CPU
32
208
32
202
204
Sequencer
PE Array
32
pe_out
Out_CPU
Ctrl Bus
Out_ext
32
32

200

FIG. 2

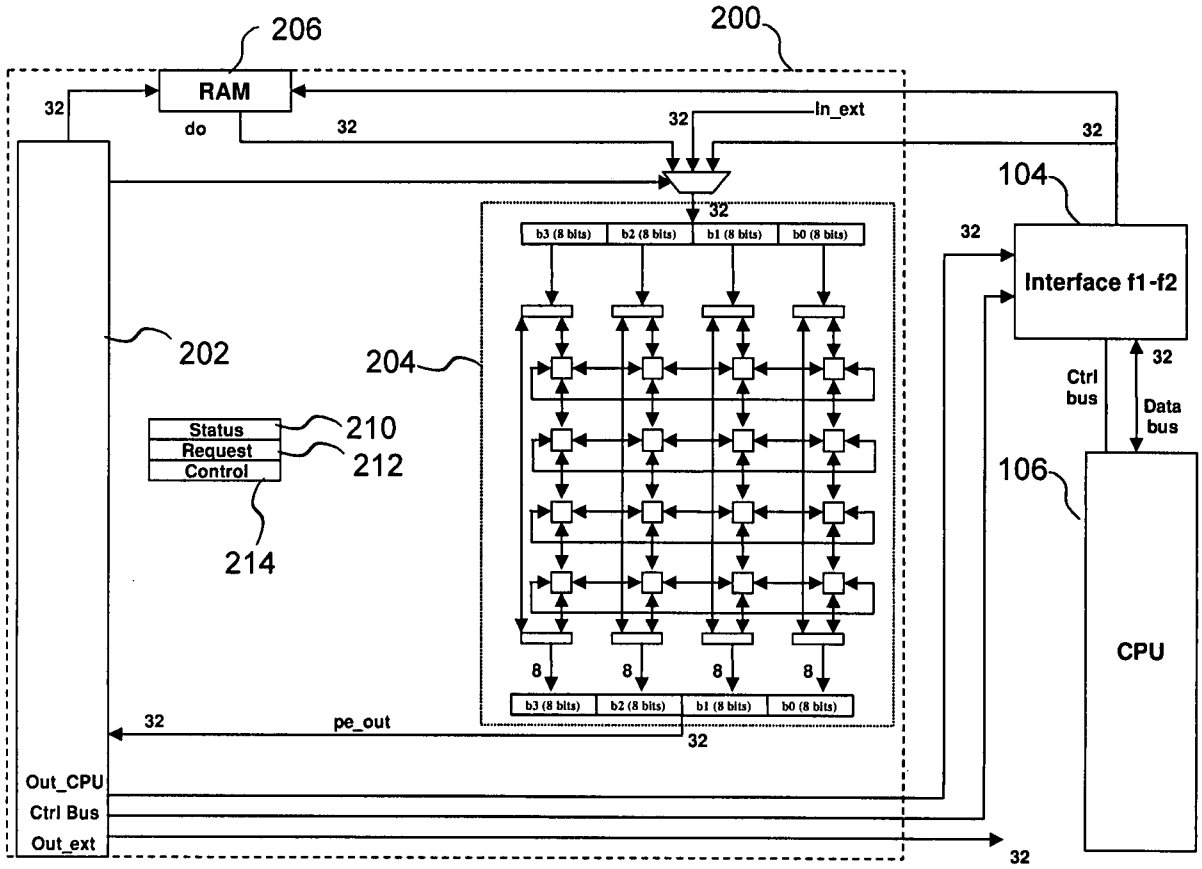

206
200
RAM
32
do
32
32
In_ext
32
104
32
Interface f1-f2
32
Ctrl bus
Data bus
32
106
CPU
202
204
Status
Request
Control
210
212
214
b3 (8 bits)
b2 (8 bits)
b1 (8 bits)
b0 (8 bits)
8
8
8
8
32
pe_out
32
Out_CPU
Ctrl Bus
Out_ext
32

FIG. 3

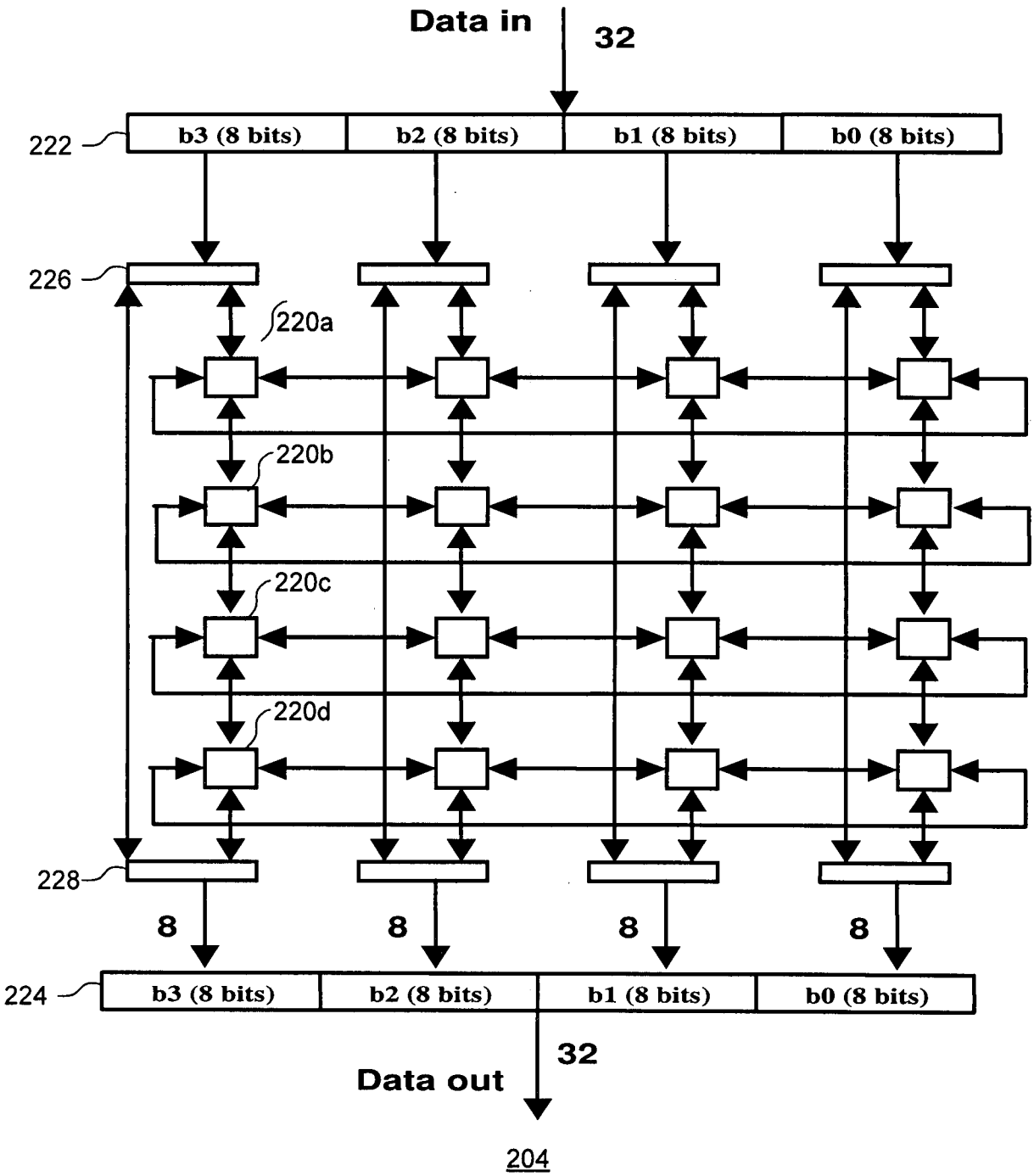

Data in
32
222
b3 (8 bits)
b2 (8 bits)
b1 (8 bits)
b0 (8 bits)
226
220a
220b
220c
220d
228
8
8
8
8
224
b3 (8 bits)
b2 (8 bits)
b1 (8 bits)
b0 (8 bits)
32
Data out
204

FIG. 4

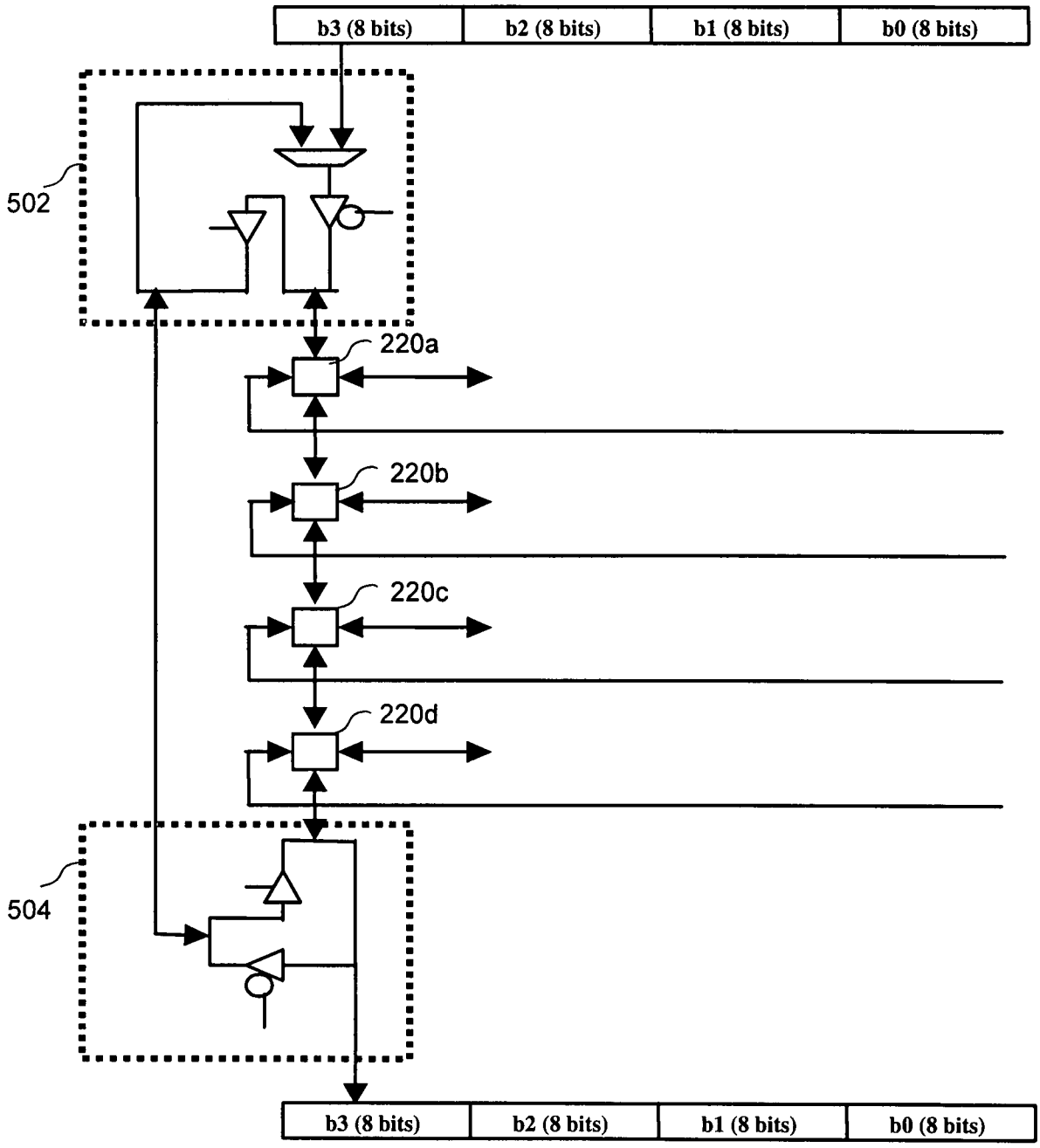
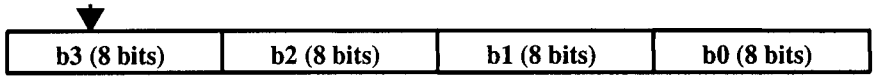

b3 (8 bits)
b2 (8 bits)
b1 (8 bits)
b0 (8 bits)
502
220a
220b
220c
220d
504
b3 (8 bits)
b2 (8 bits)
b1 (8 bits)
b0 (8 bits)

FIG. 5



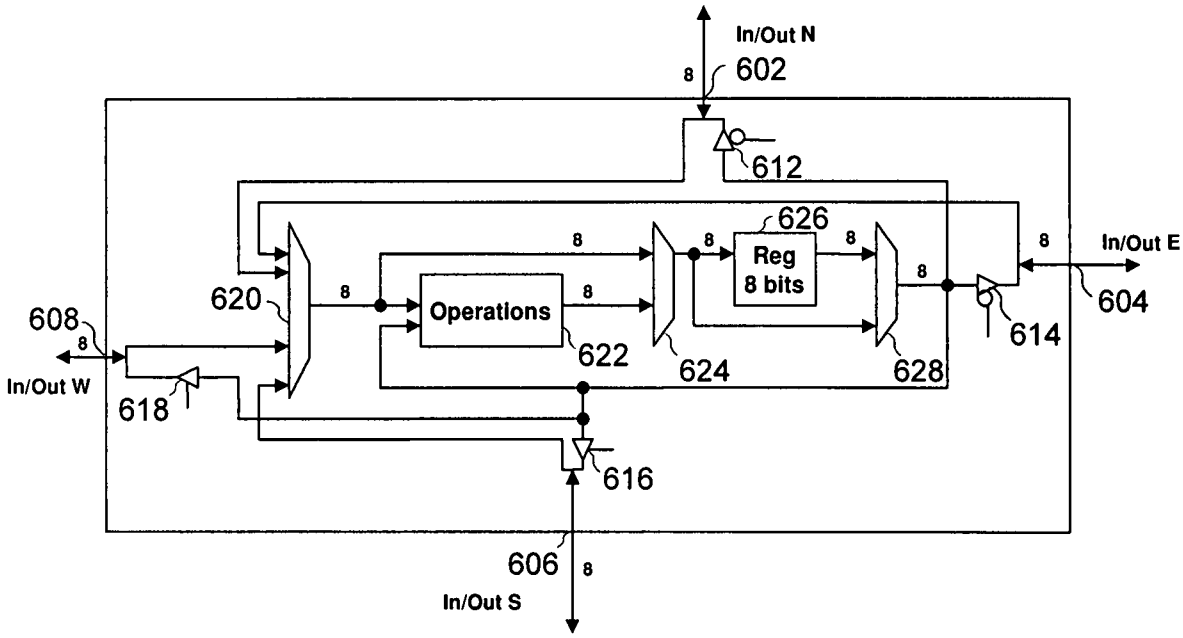

In/Out N
8
602
612
626
Reg
8 bits
8
In/Out E
604
620
Operations
622
624
628
614
608
In/Out W
618
616
606
In/Out S
600

FIG. 6

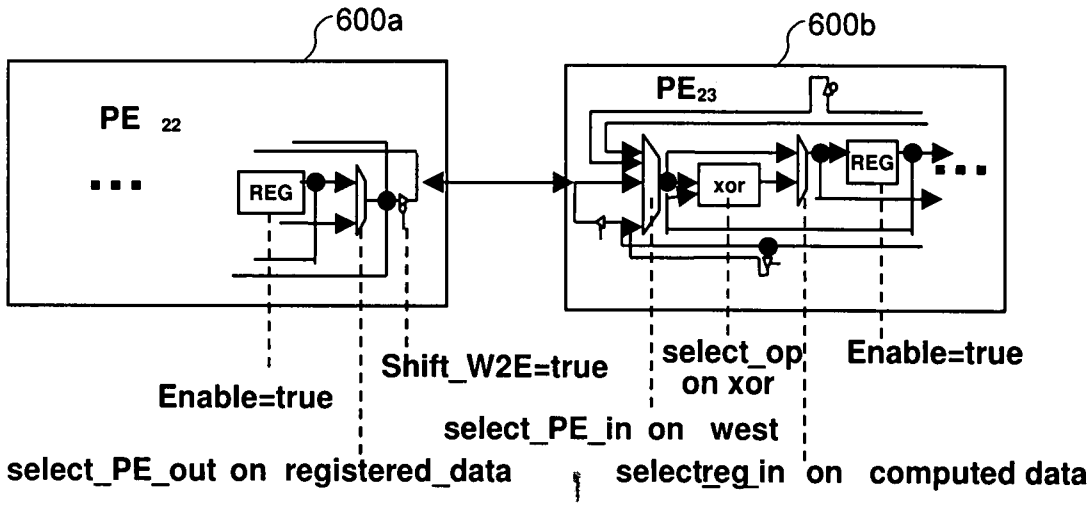

600a
600b
PE 22
PE23
REG
xor
REG
Enable=true
Shift_W2E=true
select_op
on xor
Enable=true
select_PE_in on west
select_PE_out on registered_data
selectreg in on computed data

FIG. 7

FIG. 7

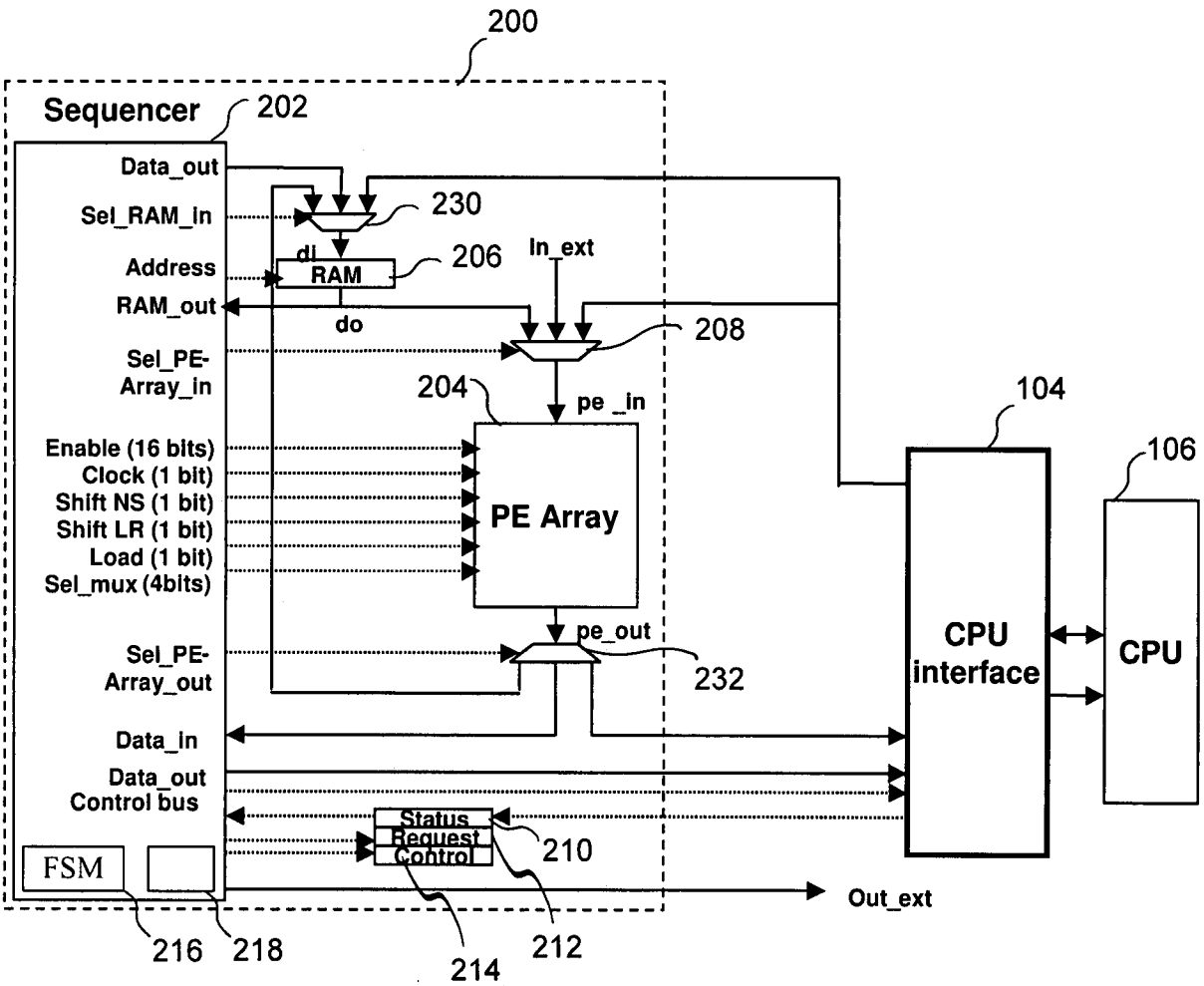

200
Sequencer
202
Data_out
Sel_RAM_in
230
Address
di
RAM
206
In_ext
RAM_out
do
208
Sel_PE-
Array_in
204
pe _in
104
106
Enable (16 bits)
Clock (1 bit)
Shift NS (1 bit)
Shift LR (1 bit)
Load (1 bit)
Sel_mux (4bits)
PE Array
pe_out
Sel_PE-
Array_out
232
CPU
interface
CPU
Data_In
Data_out
Control bus
Status
Request
Control
210
FSM
Out_ext
216
218
214
212

FIG. 8

	Name	SRC	DST	Parameters
Data Transfers	**Load Row**	CPU, RAM, ext	Row #	Autoincrement, xor, add, Address
	Load Col	CPU, RAM, ext	Col #	Autoincrement, xor, add, Address
	Load index A		Row #	Autoincrement, xor, add, Address
	Store Row	Row #	CPU, RAM, ext	CPU_request, Address
	Store Col	Col #	CPU, RAM, ext	CPU_request, Address
	Store index A	data		
Op Array	**Shift**			# times, PE enables, NSWE
	Xor			PE enables
	Nop			

FIG. 9

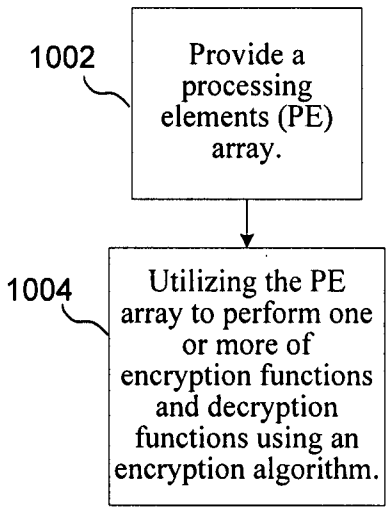

1002
Provide a processing elements (PE) array.
1004
Utilizing the PE array to perform one or more of encryption functions and decryption functions using an encryption algorithm.

FIG. 10

d_{11}	d_{12}	d_{13}	d_{14}
d_{21}	d_{22}	d_{23}	d_{24}
d_{31}	d_{32}	d_{33}	d_{34}
d_{41}	d_{42}	d_{43}	d_{44}

1100

FIG. 11

PE_{11}	PE_{12}	PE_{13}	PE_{14}
PE_{21}	PE_{22}	PE_{23}	PE_{24}
PE_{31}	PE_{32}	PE_{33}	PE_{34}
PE_{41}	PE_{42}	PE_{43}	PE_{44}

1200

FIG. 12

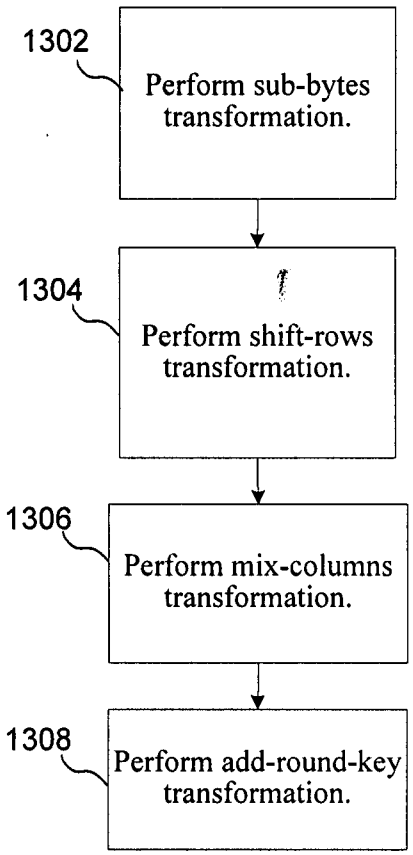

1302
Perform sub-bytes transformation.
1304
Perform shift-rows transformation.
1306
Perform mix-columns transformation.
1308
Perform add-round-key transformation.

FIG. 13

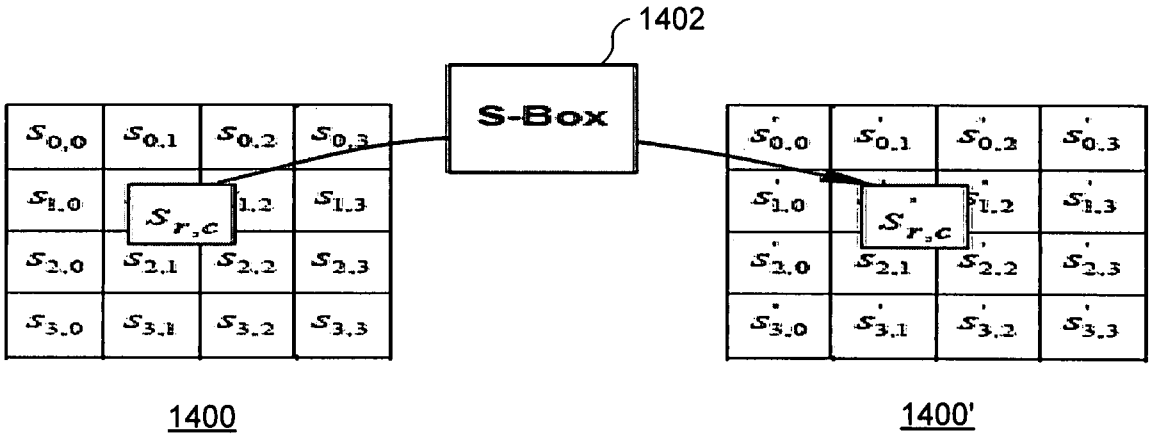

1402
S-Box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{r,c}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{r,c}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$
1400
1400'

FIG. 14

Sbox Table

63 7C 77 7B F2 6B 6F C5 30 01 67 2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 AD D4 A2 AF 9C A4 72 C0 B7 FD 93 26 36 3F F7 CC 34 A5 E5 F1 71 D8 31 15 04 C7 23 C3 18 96 05 9A 07 12 80 E2 EB 27 B2 75 09 83 2C 1A 1B 6E 5A A0 52 3B D6 B3 29 E3 2F 84 53 D1 00 ED 20 FC B1 5B 6A CB BE 39 4A 4C 58 CF D0 EF AA FB 43 4D 33 85 45 F9 02 7F 50 3C 9F A8 51 A3 40 8F 92 9D 38 F5 BC B6 DA 21 10 FF F3 D2 CD 0C 13 EC 5F 97 44 17 C4 A7 7E 3D 64 5D 19 73 60 81 4F DC 22 2A 90 88 46 EE B8 14 DE 5E 0B DB E0 32 3A 0A 49 06 24 5C C2 D3 AC 62 91 95 E4 79 E7 C8 37 6D 8D D5 4E A9 6C 56 F4 EA 65 7A AE 08 BA 78 25 2E 1C A6 B4 C6 E8 DD 74 1F 4B BD 8B 8A 70 3E B5 66 48 03 F6 0E 61 35 57 B9 86 C1 1D 9E E1 F8 98 11 69 D9 8E 94 9B 1E 87 E9 CE 55 28 DF 8C A1 89 0D BF E6 42 68 41 99 2D 0F B0 54 BB 16

FIG. 15

RAM [FF10$_{(16)}$] = CA$_{(16)}$

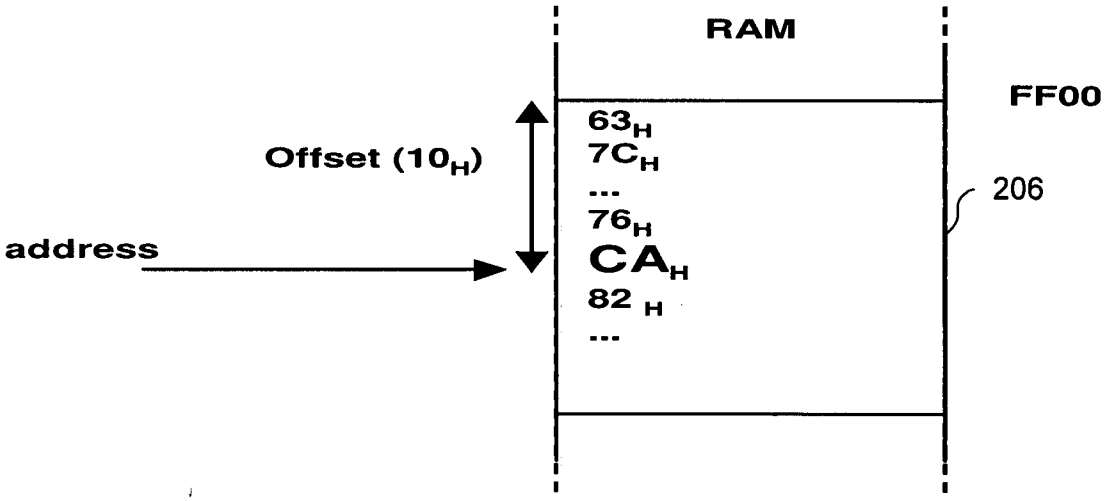


FIG. 16

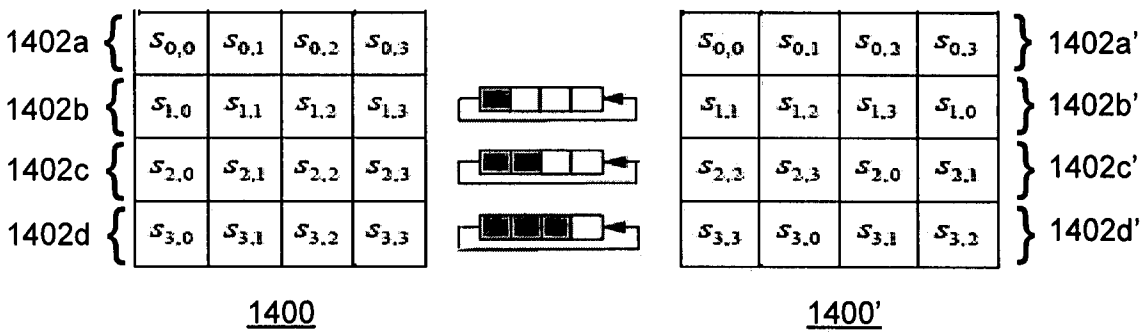

1402a
1402b
1402c
1402d
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
1400
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$
1402a'
1402b'
1402c'
1402d'
1400'

FIG. 17

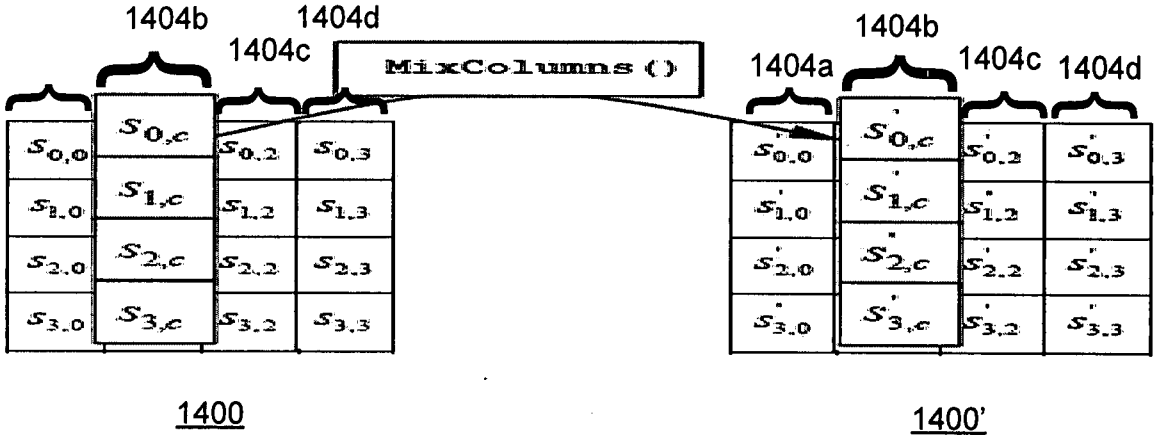

1404b
1404d
1404c
MixColumns ()
1404a
1404b
1404c
1404d
1400
1400'

FIG. 18

Logarithm Table

0 19 1 32 2 1a c6 4b c7 1b 68 33 ee df 3 64 4 e0 e 34 8d 81 ef 4c 71 8
c8 f8 69 1c c1 7d c2 1d b5 f9 b9 27 6a 4d e4 a6 72 9a c9 9 78 65 2f 8a 5 21 f
e1 24 12 f0 82 45 35 93 da 8e 96 8f db bd 36 d0 ce 94 13 5c d2 f1 40 46 83 38
66 dd fd 30 bf 6 8b 62 b3 25 e2 98 22 88 91 10 7e 6e 48 c3 a3 b6 1e 42 3a 6b
28 54 fa 85 3d ba 2b 79 a 15 9b 9f 5e ca 4e d4 ac e5 f3 73 a7 57 af 58 a8 50
f4 ea d6 74 4f ae e9 d5 e7 e6 ad e8 2c d7 75 7a eb 16 b f5 59 cb 5f b0 9c a9
51 a0 7f c f6 6f 17 c4 49 ec d8 43 1f 2d a4 76 7b b7 cc bb 3e 5a fb 60 b1 86
3b 52 a1 6c aa 55 29 9d 97 b2 87 90 61 be dc fc bc 95 cf cd 37 3f 5b d1 53 39
84 3c 41 a2 6d 47 14 2a 9e 5d 56 f2 d3 ab 44 11 92 d9 23 20 2e 89 b4 7c b8
26 77 99 e3 a5 67 4a ed de c5 31 fe 18 d 63 8c 80 c0 f7 70 7

FIG. 19

Anti-Logarithm Table

3 5 f 11 33 55 ff 1a 2e 72 96 a1 f8 13 35 5f e1 38 48 d8 73 95 a4 f7 2 6
a 1e 22 66 aa e5 34 5c e4 37 59 eb 26 6a be d9 70 90 ab e6 31 53 f5 4 c
14 3c 44 cc 4f d1 68 b8 d3 6e b2 cd 4c d4 67 a9 e0 3b 4d d7 62 a6 f1 8
18 28 78 88 83 9e b9 d0 6b bd dc 7f 81 98 b3 ce 49 db 76 9a b5 c4 57 f9
10 30 50 f0 b 1d 27 69 bb d6 61 a3 fe 19 2b 7d 87 92 ad ec 2f 71 93 ae
e9 20 60 a0 fb 16 3a 4e d2 6d b7 c2 5d e7 32 56 fa 15 3f 41 c3 5e e2 3d
47 c9 40 c0 5b ed 2c 74 9c bf da 75 9f ba d5 64 ac ef 2a 7e 82 9d bc df
7a 8e 89 80 9b b6 c1 58 e8 23 65 af ea 25 6f b1 c8 43 c5 54 fc 1f 21 63
a5 f4 7 9 1b 2d 77 99 b0 cb 46 ca 45 cf 4a de 79 8b 86 91 a8 e3 3e 42
c6 51 f3 e 12 36 5a ee 29 7b 8d 8c 8f 8a 85 94 a7 f2 d 17 39 4b dd 7c
84 97 a2 fd 1c 24 6c b4 c7 52 f6 1

FIG. 20

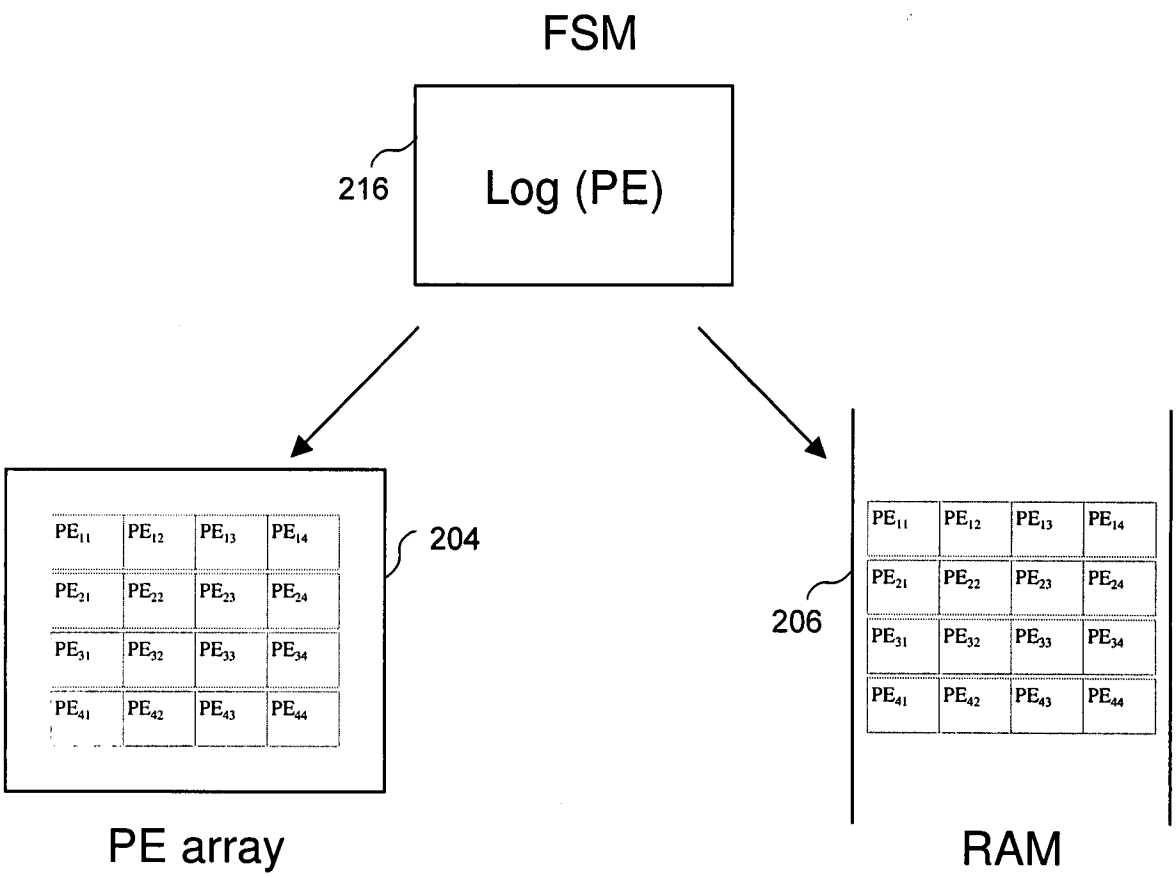

FSM
216
Log (PE)
204
206
PE11 PE12 PE13 PE14
PE21 PE22 PE23 PE24
PE31 PE32 PE33 PE34
PE41 PE42 PE43 PE44
PE array
RAM

FIG. 21