A method for accelerating pipelined cryptographic implementations

Bogdanov, Andrey; LAURIDSEN, Martin, Mehl; TISCHHAUSER, Elmar, Wolfgang

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 September 2016 (15.09.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/142330 A1

(51) International Patent Classification:
H04L 9/06 (2006.01) *G06F 9/38* (2006.01)
G09C 1/00 (2006.01)

(21) International Application Number:
PCT/EP2016/054761

(22) International Filing Date:
7 March 2016 (07.03.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15157994.3 6 March 2015 (06.03.2015) EP

(71) Applicant: **DANMARKS TEKNISKE UNIVERSITET** [DK/DK]; Anker Engelunds Vej 101A, 2800 Kgs. Lyngby (DK).

(72) Inventors: **BOGDANOV, Andrey**; Amerika Plads 3A 2mf, 2100 København Ø (DK). **LAURIDSEN, Martin, Mehl**; Venøgade 24, st. tv., 2100 København Ø (DK). **TISCHHAUSER, Elmar, Wolfgang**; Rundforbiparken 3, 1tv, 2850 Nærum (DK).

(74) Agent: **ZACCO DENMARK A/S**; Arne Jacobsens Allé 15, 2300 København S (DK).

(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, 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, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE,

[Continued on next page]

(54) Title: A METHOD FOR ACCELERATING PIPELINED CRYPTOGRAPHIC IMPLEMENTATIONS

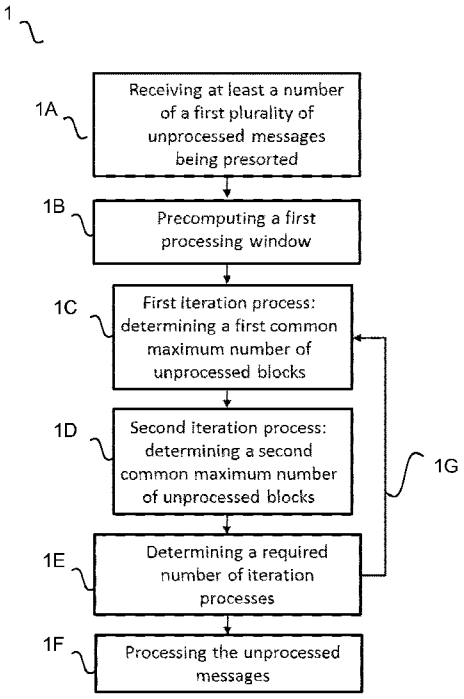

Fig. 1

(57) Abstract: The present invention relates to a system and a method for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline, where the unprocessed messages comprise respective block lengths indicating a number of unprocessed blocks which the unprocessed message may be divided into, the method comprises: receiving at least a number of the first plurality of unprocessed messages being pre-sorted, precomputing a first processing window by; determining through a first iteration process a first common maximum number of unprocessed blocks to be processed for respective unprocessed messages to be processed, determining through a second iteration process a second common maximum number of unprocessed blocks to be processed for the remaining respective unprocessed messages to be processed after the first iteration, and determining a required number of iteration processes to be performed in order to process each unprocessed block of each unprocessed message, processing the unprocessed messages associated with the first processing window, wherein the respective unprocessed messages of each respective iteration process are processed by respective multiple cryptographic processes in parallel.

WO 2016/142330 A1

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).

Published:

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

A Method for Accelerating Pipelined Cryptographic Implementations

FIELD OF INVENTION

The present invention relates to a system and a computer-implemented method of processing a plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline.

BACKGROUND OF THE INVENTION

In recent years, encryption has become commonplace and very widespread with the advent of the modern global communication infrastructure. Individuals, governments and businesses, alike nowadays, use encryption on a daily basis. In comparison to unencrypted communication, encryption always incurs a certain performance penalty, which tends to be particularly noticeable on the server side, since many connections to clients have to be dealt with simultaneously.

In order to meet the arising performance needs, microprocessor manufactures have been starting to incorporate dedicated hardware support for encryption functionality in their products. One prominent and industry-wide influential example of this is the implementation of the widely used Intel Corporation' Advanced Encryption Standard – New Instructions (AES-NI) instruction set. Since then almost every Central Processing Unit (CPU) incorporates this native support for AES encryption. AES is a very widely standardized and used encryption algorithm.

Furthermore, since AES is currently the dominant block cipher and is used in various protocols, the new instructions are valuable for a wide range of applications.

With such hardware support, client and servers alike can benefit from a performance improvement of a factor of around 10 compared to conventional software implementations. This however only applies if the encryption algorithm is used in a mode of operation, which allows for parallelization in

order to make use of the available instruction pipelining. Many of the most widely used and standardized modes of operation, most notably including Cipher Block Chaining (CBC), are inherently serial and thus cannot benefit in full from the available hardware support. For example, an implementation of CBC mode with AES-NI achieves a throughput of 4.5 cycles per byte (cpb) on contemporary platforms, while a full parallel mode would have a throughput of 0.65 cpb. Besides CBC, many other standard modes of operation, such as Cipher Feedback Block (CFB), Output Feedback Block (OFB), Cipher Block Chaining Message Authentication Code (CBC-MAC), Cryptographic Message Authentication Code (CMAC), counter with CBC-MAC (CCM) are limited by their serial nature.

PRIOR ART

The white paper "Processing Multiple Buffers in Parallel to Increase Performance on Intel Architecture Processors" by Vinodh Gopal et al., 1 July 2010, discloses a scheduling mechanism for the processing of jobs of varying sizes. The scheduler can be either in-order in which case the jobs are returned in the same order as they are submitted; alternatively, the scheduler is out-order in which case the return order can be arbitrary. The scheduler works by starting jobs whenever enough jobs are ready, and process them in a pipeline for the minimal time possible until a job can be returned. In the in-order situation there is a non-optimal use of the pipeline since messages are processed as they arrive in any unknown order and each with unpredictable length. In the out-order situation, In the out-order situation, the scheduler waits for enough messages to arrive that the pipeline can be filled in an optimal way. However, this involves the problem of prolonged latency times and a potential excessive memory consumption in case of an unfortunate distribution of message lengths.

Various documents disclose methods to improve the throughput of a cryptographic algorithm by parallel processing of several blocks in a pipeline.

For instance, the article “Intel Advanced Encryption Standard (AES) New Instructions Set rev. 3.0.1”, by Shay Gueron, 1 September 2012 discloses a method that receives a predefined size of a group of messages and processes them together. This method may perform well for a fixed number of messages of identical length, but its performance will degrade significantly when message lengths are not the same, as is the usual case. This is a problem because the distribution of message lengths will vary greatly both over time and for different applications - thus making the method less useful.

The article “Performance of Interleaved Cipher Block Chaining in CCMP” by Zadia Codabux-Rossan appearing in “Novel Algorithms and Techniques in Telecommunications and Networking”, 12 December 2009 and the article “Reducing Packet Loss in CBC Secured VoIP using Interleaved Encryption” by Dansereau et al. appearing in IEEE conference proceedings, 1 May 2006 are related to a modification to the CBC encryption mode. However, none of these articles discloses an algorithm that considers message inputs of varying length.

The article “Use of the AES instruction set” by Ryad Benadjila, 18, October 2012 discloses information about usage of the AES-NI instructions, including example latency and throughput for various Intel platforms. It also discloses examples of parallel encryption of several inputs in the AES-NI pipeline. However, nor does this article consider message inputs of varying length.

KR 2002/0071328-A discloses pipelined execution in connection with CBC. However, nor does this patent document consider message inputs of varying length.

US 7,603,549-B1 discloses parallel processing of several message inputs to be processed by a cryptographic algorithm in a pipeline. However, nor does this patent document consider message inputs of varying length.

The article “White Paper Breakthrough AES Performance with Intel AES New Instructions”, 1 January 2010, discloses processing of data blocks for

decryption using AES-CBC, which is inherently parallel, in a pipeline, thereby improving the throughput. However, nor does this patent document consider message inputs of varying length.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a computer implemented method, which exploits the available parallelism offered by simultaneous serial processing of plurality of unprocessed messages to improve the speed of processing unprocessed messages in a single pipelined cryptographic process.

According to the present invention, the above and other objects may be provided by a computer implemented method for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline, wherein the unprocessed messages has respective block lengths indicating a number of unprocessed blocks which the unprocessed message may be divided into, the method comprising:

receiving at least a number of the first plurality of unprocessed messages;

precomputing a list of processing windows by:

- determining through a first iteration process a first common maximum number of unprocessed blocks to be processed for respective unprocessed messages to be processed, wherein the determined first common maximum number of unprocessed blocks is used to define, for a first processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing;
- determining through a second iteration process a second common maximum number of unprocessed blocks, if any, to be processed for the remaining respective unprocessed messages to be processed after the first iteration, wherein the determined second common

maximum number of unprocessed blocks, if any, is used to define, for a second processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing; and

- determining a required number of iteration processes to be performed in order to process each unprocessed block of each unprocessed message; and

processing the unprocessed messages associated with windows in the list of windows at least comprising the first processing window, wherein the unprocessed blocks, defined by at least the first processing window, of respective unprocessed messages of each respective iteration process are processed window-by-window by respective multiple cryptographic processes in parallel.

Since the processing window determines a first common maximum number of unprocessed blocks and then determines a second common maximum number of unprocessed blocks for the remaining unprocessed messages after the first iteration, parallel processing of the unprocessed messages can be performed at an optimal or close to optimal trade-off between latency and pipeline utilization. Especially, messages with a particularly long or short block length are not penalized compared to messages with an average block length. Thus, it is possible to guarantee a certain throughput and a certain latency.

In some embodiments the unprocessed messages are received in a pre-sorted manner, whereas in other embodiments the unprocessed messages are pre-sorted as a step performed in connection with or as a step of precomputing the first processing window. By pre-sorting is meant to re-order the unprocessed messages in descending or ascending order of length.

By unprocessed message is meant that a message, which is unprocessed, has not yet been processed by the cryptographic processes.

A processing window thereby specifies the common maximum number of unprocessed blocks for all messages associated with the processing window and then a further window specifies the common maximum number of remaining unprocessed blocks for all messages associated with the processing window and so on for as many windows needed to process all blocks of all unprocessed messages. This process results in a list of windows. This list of windows specifies the order of processing the blocks of the unprocessed messages by the respective multiple cryptographic processes in parallel.

In one or more embodiments, the computer-implemented method is configured to schedule the processing of unprocessed blocks of unprocessed data or unprocessed messages in order to make full use of the instruction pipeline even for serial algorithms.

In one or more embodiments, the unprocessed messages may be either data and/or information.

In one or more embodiments, a message may be divided into blocks which comprises a part of the message.

In one or more embodiments, an iteration process may be a process wherein the unprocessed messages are analysed in order to determine a common maximum number of unprocessed blocks to be processed for respective unprocessed messages.

In one or more embodiments, pipelining allows Computer Processing Unit (CPU) to execute multiple data-independent instances of a same instruction, e.g. an instruction configured to AES-NI, such as aesenc, aesenclast, aesdec, or aesdeclast, in an overlapping fashion. This is done by subdividing the instruction into steps called pipeline stages, with each stage processing its part, i.e. an unprocessed block of an unprocessed message, of one instruction at a time, i.e. a sequentially processing of the unprocessed blocks. The performance of a pipelined instruction is characterized by its latency L

(number of cycles to complete one instruction) and throughput Tc (number of instructions that can be issued per cycle).

When employing a parallelizable design to a pipelined cryptographic process, the prevailing distribution of message lengths makes it hard to achieve the best performance. Arranging the plurality of unprocessed messages in parallel already in the implementation of an algorithm, comprising the computer implemented method, opens up the possibilities of increasing the performance in the cases of both sequential modes and the availability of multiple shorter or medium-sized messages. In this case, the performance penalty of serial execution can potentially be hidden by filing the pipeline with a sufficient number of operations on independent data. In a second case, there is a potential of increasing performance by keeping the pipeline filled also for overhead operations such as block cipher or multiplication calls during initialization or tag generation.

In one or more embodiments, the processing of multiple messages may be on a single core or a multiple core CPU.

Consider the scenario where a number of messages of varying lengths need to be processed by a serial encryption mode of operation. Unprocessed blocks of respective unprocessed messages have to be processed in an interleaved fashion in order to make use of the available inter-messages parallelism. Having messages of different lengths imply that generally, the pipeline cannot always be filled completely. At the same time, the goal to arrange or schedule the unprocessed blocks, according to the computer implemented method, such that pipeline usage is maximized has to be weighed against the computational cost of making such scheduling decisions.

According to an embodiment of the invention, a system for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline is provided. The system comprises;

a pipelined cryptographic processing unit,

a control logic unit,

a programmed unit configured to receive a first plurality of unprocessed messages from the pipelined cryptographic processing unit, where the unprocessed messages comprise respective block lengths indicating a number of unprocessed blocks which the unprocessed message is divided into, and at least a number of the first plurality of unprocessed messages are pre-sorted, the programmed unit is further configured to;

precompute a first processing window by;

- determining through a first iteration a first common maximum number of unprocessed blocks to be processed for respective unprocessed messages to be processed,
- determining through a second iteration a second common maximum number of unprocessed blocks to be processed for the remaining respective unprocessed messages to be processed after the first iteration, and
- determining a required number of iteration processes to be performed in order to process the unprocessed blocks of the unprocessed messages,

process the unprocessed messages associated with the first processing window, wherein the respective unprocessed messages of each respective iteration are processed by respective multiple cryptographic processes in parallel,

wherein the programmed unit transmits processed unprocessed messages to the control logic unit.

In one or more embodiments, the pipelined cryptographic processing unit may be configured to an Advanced Encryption Standard (AES), a Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3.

In one or more embodiments, the control logic unit may either be separated from the programmed unit or implemented into the programmed unit. The control logic unit may be an Arithmetic Logic unit (ALU) or a Central Processing Unit.

In one or more embodiments, the system may be implemented into a server or a computable device.

The advantage of arranging or scheduling the unprocessed messages before processing the messages is the improved speed of processing the messages in a serial cryptographic process or a partially serial cryptographic process.

The partially serial cryptographic process may be a combination of serial and parallel cryptographic processes.

Table 1 below shows test results, with the usage of AES-NI, of trivial sequential processing with and without the method for scheduling the processing of unprocessed blocks. It is clearly seen that the advantage of present invention is the improved speed of the processing in serial cryptographic operation mode.

Mode	Sequential processing (cpb)	Sequential with scheduling (cpb)	Speed-up
AES-ECB	0.65	-	
AES-CTR	0.78	-	
AES-CBC	4.47	0.87	5.14
AES-OFB	4.48	0.88	5.09
AES-CFB	4.45	0.89	5.00
AES-CMAC	4.29	0.84	5.10

Table 1: performance comparison (cycles/byte) of sequential processing with and without the scheduling

For example, for AES-CBC the speed of the processing, when including the scheduling, is improved with a speed-up factor of 5.14.

In one or more embodiments, the number of the first plurality of unprocessed messages may be determined by a parallelism degree (Par) which may be determined as:

$$Par=L*Tc,$$

wherein L denotes the latency (in cycles) and Tc denotes the throughput (in instructions/cycles) of the pipelined instruction in the single pipeline.

In one or more embodiments, the parallelism degree may vary depending on the application. Alternatively, the parallelism degree may vary depending on the load of incoming unprocessed messages.

The advantage of increasing the number of the first plurality of unprocessed messages to be processed in parallel is that the speed of processing the unprocessed messages improves even more.

Table 2 shows test results, with the usage of AES-NI, of a CBC cryptographic process with the scheduling of the unprocessed messages. In table 2, two distributions for message lengths are considered; one where all messages are 2048 bytes long, and one where all messages are "realistic" (i.e. messages which are realistic in a bimodal distribution of internet traffic). It is seen that by increasing the number of parallelism degree the relative speed increases. For example, when the parallelism degree is set to eight the speed of the processing (i.e. throughput of the messages) will be improved with a factor of 6.74 for messages of 2048 bytes long and a factor of 5.15 for realistic messages (bimodal distribution of internet traffic). However, it is seen that the increase of speed by increasing the parallelism degree reduces when going above Par = 7.

	Sequential	Parallelization degree						
		2	3	4	5	6	7	8
2 K msg.(cpb)	4.38	2.19	1.47	1.11	0.91	0.76	0.66	0.65
Relative speed-up	X1.00	X2.00	X2.98	X3.95	X4.81	X5.76	X6.64	X6.74
"realistic" msg. (cpb)	4.38	2.42	1.73	1.37	1.08	0.98	0.87	0.85
Relative speed up	X1.00	X1.81	X2.53	X3.20	X4.06	X4.47	X5.03	X5.15

Table 2: performance of CBC encryption (cpb) and relative speed-up for scheduling the unprocessed messages with different parallelism degree for fixed messenger length of 2048 bytes and realistic message lengths.

In one or more embodiments, the processing of the unprocessed messages by the respective multiple cryptographic processes may be configured to perform one or more of the following processes:

- an encryption process or a decryption process,

- an authenticated encryption process (with associated data) or an authenticated decryption process, and/or a verification process of integrity and authenticity,
- a message authentication process or a message verification process of integrity, and/or an authenticity,
- a hashing process,
- a pseudo- random number generation process,
- a password based key derivation functions process, or
- a post-processing of true random numbers process.

In one or more embodiments, the processing of the unprocessed blocks of the respective unprocessed messages within the respective cryptographic processes may either be done serially or partially serial.

The advantage of this is that the method of the present invention is configurable to any serial mode of operation, such as CBC, CFB, OFB, CBC-MAC, CMAC, OMAC, GMAC, GOST, CLOC, SILC, VMAC, UMAC, KDF1, KDF2, KDF3, HKDF, PBKDF1, PBKDF2, Davies-Meyer, Matyas-Meyer-Oseas, Miyaguchi-Preneel or Sponge.

The advantage of combining the scheduling and a serial mode of operation is that it gives a far better speed up result compared to a mode of operation being partially serial. A further advantages is that serial mode of operation can now be performed in parallel in a single pipeline.

Furthermore, the advantage of this is that the method of the present invention is configurable to any partial serial mode of operation, such as GCM, CCM, EAX, OCB3, OTR, COPA, or POET

The advantage of combining the scheduling and a partially serial mode of operation is that it gives good speed-ups by parallelizing (using again the independency of several unprocessed messages) the serial parts of the

modes (typically initialization and finalization). This will be especially prominent in use cases where the unprocessed messages are short, i.e for GCM using AES-NI, the length of a short message might be equal or less than 128 bytes. The length of a short message may depend on the application, but mostly the length of a short message may be within a range of 2 bytes to 128 bytes, 32 bytes to 256 bytes, or 128 bytes to 256 bytes.

A partial serial cryptographic process may be a combination of at least a serial cryptographic process with at least a parallel cryptographic process, and the computer implemented method may be configured to at least a serial cryptographic. Thereby, the advantage is that a partial serial or a partial parallel cryptographic process would experience an improved speed-up.

In one or more embodiments, the processing of unprocessed blocks of a respective unprocessed message within a respective cryptographic process may be done sequentially, and each unprocessed block may be processed during a single operation.

The advantage of processing the unprocessed blocks sequentially is that the speed of processing is reduced compared to random processing of the unprocessed blocks.

In one or more embodiments, the pipeline may be configured to Advanced Encryption Standard (AES), Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3.

In one or more embodiments, the pre-sorted unprocessed messages may be sorted by decreasing length or increasing length.

The sorting of the unprocessed messages simplifies the processing, and thereby, improves the speed of the processing of unprocessed messages even more. The sorting can be implemented via an optimal sorting network for the value of Par chosen by the implementation of the present invention. Alternatively, a low-overhead algorithm, like "*Insertion Sort*", can be used.

In one or more embodiments, firstly, a first list of processing windows is computed and then, secondly, a second list of processing windows is computed. The precomputing of a second list of processing window may be initiated if the unprocessed messages, associated with the first list of processings windows, have been processed by the respective cryptographic processes, and if a second plurality of unprocessed messages with respective block lengths is being received or has been received.

In some embodiments the list of windows is reused over further incoming messages. This reduces overhead in precomputing the list of windows.

In some embodiments the method comprises a step of measuring the distribution of lengths of messages at different points in time and use this measure to decide whether to reuse the list of windows.

In one or more embodiments, the first processing window may be reused if the unprocessed messages, configured to the first processing window, have been processed by the respective cryptographic processes, and, if receiving a second plurality of unprocessed messages with similar respective block lengths as the respective block lengths of the first plurality of unprocessed messages, or if receiving the first plurality of unprocessed messages with respective block lengths.

In one or more embodiments, the first processing window may be reused if the unprocessed messages, configured to the first processing window, have been processed by the respective cryptographic processes being a two-pass scheme, such as CCM.

In one or more embodiments, the first processing window may be reused if the unprocessed messages, configured to the first processing window, have been processed by the respective cryptographic processes.

The advantage of reusing a processing window is that the speed of processing a first or a second plurality of unprocessed messages would be even more improved.

In one or more embodiments, the processing window may determine through the second iteration process a maximum number of unprocessed blocks to be processed of the remaining unprocessed message to be processed, and the processing of the unprocessed message of each iteration process associated with the processing window is processed by a respective cryptographic process.

In one or more embodiments, the system may comprise a pipeline which may be configured to Advanced Encryption Standard (AES), Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3

In one or more embodiments, a computer readable medium may be configured with the system as described above.

It is an advantage that the speed improvement is significant when one or more embodiments of the present invention is processed on an Intel based chip architecture. Intel based chip architecture may be produced by Intel corporation or Intel.

The advantage of the one or more embodiments of the present invention is that it is flexible, which means that it can be implemented in any applications using the various protocols of AES.

BRIEF DESCRIPTION OF THE DRAWING

Fig.1 shows a flow diagram illustrating a computer implemented method of processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline,

Fig. 2 illustrates an example of processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline,

Fig. 3A and 3B show a flow diagram of a second embodiment and a third embodiment of the invention, respectively,

Fig. 4 illustrates the system for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline,

Fig. 5 shows an example of a system where a computer implemented method, is implemented into an existing cryptographic system.

DETAILED DESCRIPTION OF THE DRAWING:

The present invention will now be described more fully hereinafter with reference to the accompanying drawings, in which exemplary embodiments of the invention are shown. The invention may, however, be embodied in different forms and should not be construed as limited to the embodiments set forth herein. Rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art.

In the following the processing of the unprocessed messages by the respective multiple cryptographic processes may be configured to following process;

- an encryption process or a decryption process,
- an authenticated encryption process (with associated data) or an authenticated decryption process, and/or a verification process of integrity and authenticity,
- a message authentication process or a message verification process of integrity, and/or an authenticity,

- a hashing process,
- a pseudo- random number generation process,
- a password based key derivation functions process, or
- a post-processing of true random numbers process.

In the following, none of the embodiments, described below, are limited to a single configuration of a cryptographic process.

In the following, the unprocessed messages may comprise data or information.

Fig.1 shows a flow diagram illustrating a computer implemented method 1 of processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline 4. The method 1 may be also be denoted as a schedule process.

In the following, the unprocessed messages T comprise respective block lengths B_L indicating a number of unprocessed blocks B which the unprocessed message T is divided into. In a first step, the method receives at least a number of unprocessed messages T of the first plurality of unprocessed messages k which are pre-sorted by decreasing length or increasing length 1A.

The processing of the unprocessed messages in parallel, using multiple cryptographic processes in a single pipeline, is determined through a first processing window 2, 1B. The first processing window 2 is precomputed, by firstly, determining through a first iteration process I_1, a first common maximum number of unprocessed blocks B_1 to be processed for respective unprocessed messages T_1, 1C.

Secondly, through a second iteration process I_2 a second common maximum number of unprocessed blocks B_2 is determined for the remaining respective unprocessed messages T_2 after the first iteration I_1, 1D.

Additionally or alternatively the processing window 2 may be determined through the second iteration process I_2 a maximum number of unprocessed blocks B to be processed of the remaining unprocessed message to be processed.

Thirdly, the first iteration process I_1 and/or the second iteration process I_2 may be repeated in at least one additional iteration process so that a number of iteration processes Ix, required to perform the processing of each unprocessed block of each unprocessed message, may be determined (1E, 1G).

Then, the unprocessed messages T are processed in accordance with the first processing window 2, wherein the respective unprocessed messages T of each respective iteration process I are processed by respective multiple cryptographic processes in parallel, 1F.

Additionally or alternatively, the processing of the unprocessed message of each iteration process, associated with the processing window 2, is processed by a respective cryptographic process.

Fig. 2 illustrates an example of processing a first plurality of unprocessed messages k in parallel using multiple cryptographic processes in a single pipeline 4. In this specific example, four unprocessed messages (M_1, M_2, M_3, and M_4) of the first plurality of unprocessed messages k are received and sorted by increasing block length (B_L).

In this particular example, a first unprocessed message M_1 is divided into three unprocessed blocks ($M_{1,1}$, $M_{1,2}$, and $M_{1,3}$) with a certain data capacity. The first unprocessed message M_1 has a first block length B_{L1} of three. A second unprocessed message M_2 is divided into five unprocessed blocks ($M_{2,1}$, $M_{2,2}$, $M_{2,3}$, $M_{2,4}$ and $M_{2,5}$) having a second block length B_{L2} of five. A third unprocessed message M_3 is divided into eight unprocessed blocks ($M_{3,1}$, $M_{3,2}$, $M_{3,3}$, $M_{3,4}$, $M_{3,5}$, $M_{3,6}$, $M_{3,7}$, and $M_{3,8}$) having a third block length B_{L3} of eight. A fourth unprocessed M_4 message is divided into eight unprocessed

blocks ($M_{4,1}$, $M_{4,2}$, $M_{4,3}$, $M_{4,4}$, $M_{4,5}$, $M_{4,6}$, $M_{4,7}$, and $M_{4,8}$) having a fourth block length B_{L4} of eight.

The number of received unprocessed messages T may be determined by a parallelism degree (*Par*), $ar = L * Tc$, wherein L denotes the latency (in cycles) and Tc denotes the throughput (in instructions/cycles) of the pipelined instruction.

The processing of the unprocessed messages (M_1, M_2, M_3, and M_4) are subdivided into a number of iteration processes (I_1, I_2, and I_3), wherein each iteration process is configured to process as many consecutive unprocessed blocks B as possible for as many respective messages T as possible. In the first iteration process I_1, the maximum number of unprocessed messages T_1, having a common maximum number of unprocessed blocks B_1, may be determined by the parallelism degree Par.

In the next iteration process the degree of parallelism reduces since at least one message will be exhausted, i.e. each block of the respective message is processed.

In this specific example, the precomputing of a first list of processing windows 2 comprises a first iteration process I_1, where, for a first processing window, a first common maximum number of unprocessed blocks B_1 is set to three for respective four unprocessed messages T_1. Each row of the table (i.e. as referred to be reference numeral 2) represents a window and the whole table represents the list of processing windows.

In the following, the precomputation of each window in the list of windows is also referred to as an iteration process, I.

Furthermore, a second iteration process I_2 is required. In the second iteration I_2, a second common maximum number of unprocessed blocks B_2 is set to two for respective three unprocessed messages T_2.

In this situation, the parallelism degree will be reduced from four to three, during the transfer from first to second iteration process.

Furthermore, a third iteration I_3 is required, wherein a third common maximum number of unprocessed blocks B_3 is set to three for respective two unprocessed messages T_3.

In this situation, the parallelism degree will be reduced from three to two during the transfer from second to third iteration process.

The required number of iteration processes Ix, in order to complete the processing of the unprocessed messages T, is set to three.

After precomputing the processing window 2, the unprocessed messages T are processed, in accordance with the processing window 2, by respective multiple cryptographic processes (CBC_1, CBC_2 and CBC_3) in parallel and within the single pipeline 4.

In this specific example, the unprocessed messages T_1 for the first iteration process I_1 are processed by respective cryptographic processes in parallel. Each cryptographic process, in this specific example, is configured to a Cipher Block Chaining (CBC) mode of operation. The cryptographic processing would be repeated for the remaining iteration processes, i.e. the second and the third iteration process (I_2, I_3).

In a further embodiment, the cryptographic processes could be configured to other types of serial mode of operation, such as CFB, OFB, CBC-MAC, CMAC, OMAC, GMAC, GOST, CLOC, SILC, VMAC, UMAC, KDF1, KDF2, KDF3, HKDF, PBKDF1, PBKDF2 Davies-Meyer, Matyas-Meyer-Oseas, Miyaguchi-Preneel or Sponge.

Alternatively, in a further embodiment, the cryptographic processes could be configured to a partially serial mode of operation, such as GCM, CCM, EAX, OCB3, OTR, COPA, or POET.

Furthermore, the pipeline 4 may be configured to an Advanced Encryption Standard (AES), Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3.

In this particular example, the respective unprocessed messages T are being processed by respective CBC processes (CBC_1, CBC_2, CBC_3, and CBC_4), and each unprocessed block B is being processed in each block cipher (E_{k1}, E_{k2}, or E_{k3}).

The unprocessed blocks B of respective unprocessed message T are processed sequentially and in parallel with other unprocessed blocks B_x of other respective unprocessed messages T_x.

Alternatively, the unprocessed blocks of respective unprocessed message are processed sequentially and in parallel with other unprocessed blocks Bx of a second respective unprocessed message Tx.

Each output (O_1, O_2, O_3, or O_4) of respective CBC processes transmit the processed messages to a second unit, e.g. a logic unit (Not shown in Fig. 2).

Fig. 3A and 3B show a flow diagram of a second embodiment and a third embodiment of the invention, respectively.

Fig. 3A shows a method 1, wherein a plurality of unprocessed messages k is received and stored in array M[k] with corresponding block length $B_L[k]$, 3A1. The multiple messages are sorted by decreasing block length $B_L[k]$ and stored in array L[k] with corresponding unprocessed messages M[k], 3A2.

A length of array L[k] of above zero 3A3 indicates that at least one unprocessed message is received. Based on a parallelism degree Par and the received plurality of unprocessed messages k a number of the plurality of unprocessed messages (5, r), to be processed, is determined, 3A4.

The number of the plurality of unprocessed messages (5, r), to be processed, are initialized according to a cryptographic process which may be used for processing each unprocessed message (5, r), 3A5. Then, a processing

window is calculated 3A6, and how to calculate the processing window is described in Fig. 1 and Fig. 2.

After the processing window is calculated, the unprocessed messages will be processed 3A7 and returned to a second unit, e.g. a logic unit, 3A8.

Fig 3B illustrates a method 1, wherein a front end 3B1 comprises steps 3A1 to step 3A5, from Fig. 3A, in a similar order, i.e. 3A2 and 3A4 may switch places such that the number of the plurality of unprocessed messages (5,r) to be processed may be determined before sorting the unprocessed messages (5,r).

A first processing window is calculated 3B2, and if the iteration process I is less or equal the required number of iteration processes Ix, 3B3, the unprocessed blocks B of the respective unprocessed messages T of iteration process I are processed in respective cryptographic processes, 3B4. If the iteration process I is above the required number of iteration process Ix, the processed messages are finalized according to the cryptographic process used for processing each unprocessed message T, 3B5.

In one or more embodiments, the processed messages may be removed from the array L[k] and returned to a second unit, e.g. a logic unit, 3B6. The remaining unprocessed messages from the array L[k] may be processed according to the first processing window 2 or according to a second processing window 2x, 3B2.

In one or more embodiments, a second processing window 2X may be precomputed if the unprocessed messages T, configured to the first processing window 2, have been processed by the respective cryptographic processes, and if receiving a second plurality of unprocessed messages kx with respective block lengths B_L.

In one or more embodiments, the first processing window 2 may be reused if the unprocessed messages T, configured to the first processing window 2,

have been processed by the respective cryptographic processes, and, if receiving either a second plurality of unprocessed messages kx with respective block lengths or the first plurality of unprocessed messages k with respective block lengths.

The processed messages may be returned to a second unit, e.g. a logic unit.

Fig. 4 shows a system 10 for processing a first plurality of unprocessed messages k in parallel using multiple cryptographic processes in a single pipeline 4. The system 10 comprises a pipelined cryptographic processing unit 4x1, a control logic unit 4x2, and a programmed unit 4x3 configured to receive a first plurality of unprocessed messages k and/or information from the pipelined cryptographic processing unit 4x1. The information could include initialization information indicating, to the pipelined cryptographic processing unit 4x1, that the processing of the received unprocessed messages T is ready to be processed. Furthermore, the information could include a finalization information indicating, to the pipelined cryptographic processing unit 4x1, that the processing of the received unprocessed messages T is finished.

The unprocessed messages T comprise respective block lengths B_L indicating a number of unprocessed blocks B which the unprocessed message T is divided into, and at least a number of the first plurality of unprocessed messages k are pre-sorted.

The programmed unit 4x3 is configured with one or more embodiments described in Figs. 1, 2, 3A and 3B. In this specific example, the programme unit 4x3 is configured to precompute a first processing window 2, by firstly, determining through a first iteration I_1 a first common maximum number of unprocessed blocks B_1 to be processed for respective unprocessed messages T_1 to be processed. Secondly, determining through a second iteration I_2 a second common maximum number of unprocessed blocks B_2 to be processed for the remaining respective unprocessed messages T_2 to be

processed after the first iteration I_1. Thirdly, determining a required number of iteration processes Ix to be performed in order to process the unprocessed blocks B of the unprocessed messages T,

Then, the unprocessed messages T associated with the first processing window 2 are processed, wherein the respective unprocessed messages T of each respective iteration Ix are processed by respective multiple cryptographic processes in parallel. Then, the processed messages are transmitted to the control logic unit 4x2.

Furthermore, the control logic unit 4x2 is configured to control the programmed unit 4x3. The control logic unit 4x2 may receive information from the programmed unit 4x3. The information could be an initialization information indicating, to the control logic unit 4x2, that the processing of the received unprocessed messages T is ready to be processed. Furthermore, the information could be a finalization information indicating, to the control logic unit 4x2, that the processing of the received unprocessed messages T is finished.

The communication between the pipelined cryptographic processing unit 4x1 and the control logic unit 4x2 may involve other types of cryptographic processes not relating to the programmed unit 4x3.

The pipelined cryptographic processing unit 4x1 may be an AES, and the control unit 4x2 may for example be an Arithmetic Logic unit (ALU) or a Central Processing Unit.

Fig 5 shows an example of a system 20 where a computer implemented method 10, is implemented into an existing cryptographic system 20, wherein the existing cryptographic system 20 comprises an existing cryptographic process including a cryptographic software 5a2 in connection with a pipelined cryptographic processing unit 5a3.

In this specific example, a data source 5x1 could be a server, a web server or any kind of a server or a computable device, providing a plurality of unprocessed messages k to an existing cryptographic software 5a2. The Cryptographic software 5a2 is configured to communicate with the computer implemented method 5b2 and the existing pipelined cryptographic processing unit 5a3. Furthermore, the Cryptographic software 5a2 is configured to divide the plurality of unprocessed of messages into a first part of unprocessed messages and a second part of unprocessed messages. The first part is transmitted to the computer implemented method 5b2 and the second part is transmitted directly to the pipelined cryptographic processing unit 5a3. Thereby, the system 20 may combine the already implemented cryptographic process, included in the cryptographic software 5a2, with the implemented method 5b2 in order to process the unprocessed messages or data.

The first part and the second part may both be transmitted to either the computer implemented method 5b2 or directly to the pipelined cryptographic processing unit 5a3. Thereby, the system 20 may be configured to only process the unprocessed messages in either the existing cryptographic process (5a2, 5a3) or in the implemented method (5b2, 5a3).

In this specific example, the pipelined cryptographic processing unit 5a3 may be included in a central processing unit.

In some embodiments there is provided a computer implemented method of processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline, where the unprocessed messages comprise respective block lengths indicating a number of unprocessed blocks which the unprocessed message is divided into, wherein the method comprises; receiving at least a number of the first plurality of unprocessed messages; precomputing a first processing window by: determining through a first iteration process a first common maximum number of unprocessed blocks to be processed for respective unprocessed

messages to be processed, determining through a second iteration process a second common maximum number of unprocessed blocks to be processed for the remaining respective unprocessed messages to be processed after the first iteration, and determining a required number of iteration processes to be performed in order to process each unprocessed block of each unprocessed message; and processing the unprocessed messages associated with the first processing window, wherein the respective unprocessed messages of each respective iteration process are processed by respective multiple cryptographic processes in parallel.

When receiving at least a number of the first plurality of unprocessed messages, the unprocessed messages may be pre-sorted. The messages may be presorted or be sorted as a step of precomputing one or both of the first processing window and any subsequent windows such as a second processing window.

In some embodiments the precomputing of a second processing window is initiated if the unprocessed messages, configured to the first processing window, have been processed by the respective cryptographic processes, and if receiving a second plurality of unprocessed messages with respective block lengths.

In some embodiments the first processing window is reused if the unprocessed messages, configured to the first processing window, have been processed by the respective cryptographic processes, and if receiving a second plurality of unprocessed messages with similar respective block lengths as the respective block lengths of the first plurality of unprocessed messages, or if receiving the first plurality of unprocessed messages with respective block lengths.

1	A computer implemented method of processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline.
2	Processing window.
2x	Second processing window.
4	A pipeline.
5	A number of a plurality of unprocessed message.
10	A system for processing plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline.
20	A second system comprising the computer implemented method and an existing cryptographic process.
T	Unprocessed messages of common maximum number of unprocessed blocks.
T_x	Other unprocessed messages or another unprocessed message.
B	A common maximum number of unprocessed blocks.
B_x	Other unprocessed blocks of another unprocessed message.
T1, T2, and T3	A first, second and third unprocessed messages, respectively.
B1, B2, and B3	A first, a second and a third common maximum number of unprocessed blocks, respectively.
B_L	A block length.
B_{L1},B_{L2}, B_{L3}, and B_{L4}	A first block length, a second block length, a third block length and a fourth block length.
I	Iteration number
Ix	Required number of iteration processes.

I_1	First Iteration number.
I_2	Second Iteration number.
I_3	Third Iteration number.
M_x	Message number.
$M_{x,y}$	Block number y of message x.
CBC_1, CBC_2, and CBC_3	A first Cipher Block Chaining, a second Cipher Block Chaining, and a third Cipher Block Chaining, respectively.
E_{k1},E_{k2}, and E_{k3}	A first block cipher, a second block cipher and a third block cipher, respectively.
k	Plurality of unprocessed messages.
kx	Second plurality of unprocessed messages.
1A	Receiving at least a number of unprocessed messages of the first plurality of unprocessed messages being pre-sorted.
1B	Precomputing a first processing window.
1C	Determining a first common maximum number of unprocessed messages of a first iteration.
1D	Determining a second common maximum number of unprocessed messages of a second iteration.
1E	Determining the required number of iterations in order to process each unprocessed block of each unprocessed message.
1F	Processing the unprocessed messages in parallel.
1G	Repeat the first and/or second iteration.
3A1	Receive k unprocessed messages.
3A2	Sort the unprocessed messages.
3A3	Check if received unprocessed messages.

3A4	Determine a parallelism degree.
3A5	Perform initialization of unprocessed messages.
3A6	Precompute a processing window.
3A7	Parallel processing of unprocessed messages.
3A8	Return processed messages.
3B1	Comprises 3A1 to 3A5.
3B2	Precompute or reuse a processing window.
3B3	Check the number of iteration process.
3B4	Process the unprocessed blocks.
3B5	Finalization of processed messages.
3B6	Remove the processed messages from the plurality of unprocessed messages.
3B7	Return processed messages.
4x1	A pipelined cryptographic processing unit
4x2	A control logic
4x3	A programmed unit
5x1	A data source
5x2	A software unit
5a2	A cryptographic software
5b2	A computer implemented method
5x3	A hardware unit
5a3	A pipelined cryptographic processing unit

CLAIMS

1. A computer implemented method for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline, wherein the unprocessed messages has respective block lengths indicating a number of unprocessed blocks which the unprocessed message may be divided into, the method comprising:

receiving at least a number of the first plurality of unprocessed messages;

precomputing a list of processing windows by:

- determining through a first iteration process a first common maximum number of unprocessed blocks to be processed for respective unprocessed messages to be processed, wherein the determined first common maximum number of unprocessed blocks is used to define, for a first processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing;
- determining through a second iteration process a second common maximum number of unprocessed blocks, if any, to be processed for the remaining respective unprocessed messages to be processed after the first iteration, wherein the determined second common maximum number of unprocessed blocks, if any, is used to define, for a second processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing; and
- determining a required number of iteration processes to be performed in order to process each unprocessed block of each unprocessed message; and

processing the unprocessed messages associated with windows in the list of windows at least comprising the first processing window, wherein the

unprocessed blocks, defined by at least the first processing window, of respective unprocessed messages of each respective iteration process are processed window-by-window by respective multiple cryptographic processes in parallel.

2. A method according to claim 1, wherein the number of the first plurality of unprocessed messages is determined by a parallelism degree (*Par*) which is determined as

$$Par = L * Tc,$$

wherein L denotes the latency (in cycles) and Tc denotes the throughput (in instructions/cycles) of a pipelined instruction in the single pipeline.

3. A method according to any of the previous claims, wherein the processing of the unprocessed messages by the respective multiple cryptographic processes are configured to perform the following processes;

- an encryption process or a decryption process,
- an authenticated encryption process (with associated data) or an authenticated decryption process, and/or a verification process of integrity and authenticity,
- a message authentication process or a message verification process of integrity, and/or an authenticity,
- a hashing process,
- a pseudo- random number generation process,
- a password based key derivation functions process, or

- a post-processing of true random numbers process.

4. A method according to any of the previous claims, wherein the processing of the unprocessed blocks of the respective unprocessed messages within the respective cryptographic processes is either done serially or partially serial.

5. A method according to any of the previous claims, wherein the processing of unprocessed blocks of a respective unprocessed message within a respective cryptographic process is done sequentially, and each unprocessed block is processed during a single operation.

6. A method according to any of the previous claims, wherein each cryptographic process is configured to perform a serial mode of operation, such as CBC, CFB, OFB, CBC-MAC, CMAC, OMAC, GMAC, GOST, CLOC, SILC, VMAC, UMAC, KDF1, KDF2, KDF3, HKDF, PBKDF1, PBKDF2, Davies-Meyer, Matyas-Meyer-Oseas, Miyaguchi-Preneel or Sponge.

7. A method according to claims 1 to 5, wherein each cryptographic process is configured to perform a partially serial mode of operation, such as GCM, CCM, EAX, OCB3, OTR, COPA, or POET

8. A method according to any of the previous claims, wherein the pipeline is configured to perform operations according to Advanced Encryption Standard (AES), Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3.

9. A method according to any of the previous claims, wherein the pre-sorted unprocessed messages are sorted by decreasing length or increasing length.

10. A method according to any of the previous claims, wherein the precomputing of a second list of processing windows is initiated if the unprocessed messages associated with the first list of processing windows, have been processed by the respective cryptographic processes, and if a second plurality of unprocessed messages with respective block lengths has been received or is being received.

11. A method according to any of the previous claims, wherein the first list of processing windows is reused if the unprocessed messages, associated with the first list of processing windows, have been processed by the respective cryptographic processes, and;

- if receiving a second plurality of unprocessed messages with similar respective block lengths as the respective block lengths of the first plurality of unprocessed messages, or
- if receiving a plurality of unprocessed messages with respective block lengths being the same as the as the respective block lengths of the first plurality of unprocessed messages.

12. A method according to any of the previous claims, wherein the processing window is determining through the second iteration process a maximum number of unprocessed blocks to be processed of the remaining unprocessed message to be processed, and the processing of the unprocessed message of each iteration process associated with the processing window is processed by a respective cryptographic process.

13. A system for processing a first plurality of unprocessed messages in parallel using multiple cryptographic processes in a single pipeline, wherein the system comprises;

a pipelined cryptographic processing unit;

a control logic unit;

a programmed unit configured to receive a first plurality of unprocessed messages from the pipelined cryptographic processing unit, where the unprocessed messages comprise respective block lengths indicating a number of unprocessed blocks which the unprocessed message is divided into, the programmed unit is further configured to:

- determine through a first iteration a first common maximum number of unprocessed blocks to be processed for respective unprocessed messages to be processed, wherein the determined first common maximum number of unprocessed blocks is used to define, for a first processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing;

- determine through a second iteration a second common maximum number of unprocessed blocks, if any, to be processed for the remaining respective unprocessed messages to be processed after the first iteration, wherein the determined second common maximum number of unprocessed blocks, if any, is used to define, for a second processing window, in the list of processing windows, how many unprocessed blocks of which unprocessed messages to include for processing; and

- determine a required number of iteration processes to be performed in order to process the unprocessed blocks of the unprocessed messages,

- process the unprocessed messages associated with windows in the list of windows at least comprising the first processing window, wherein the unprocessed blocks, defined by at least the first processing window, of respective unprocessed messages of each respective iteration are processed window-by-window by respective multiple cryptographic processes in parallel,

wherein the programmed unit transmits processed unprocessed messages to the control logic unit.

14. A system according to claim 13, wherein the pipeline is configured to Advanced Encryption Standard (AES), Secure Hash Algorithm – 1 (SHA-1), SHA-2, or SHA-3

15. A computer readable medium configured with the system according to claims 13-14.

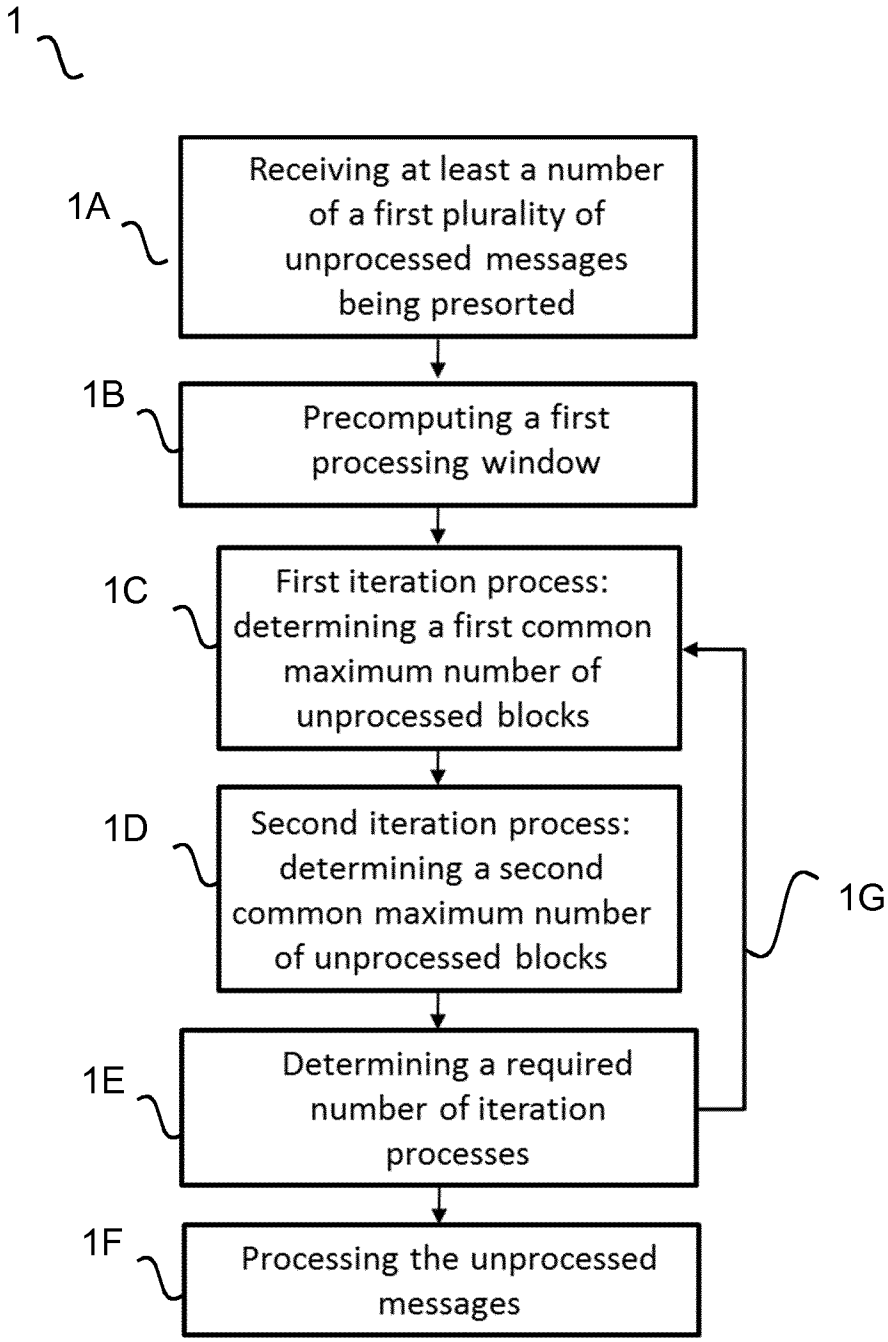
1
1A
Receiving at least a number of a first plurality of unprocessed messages being presorted
1B
Precomputing a first processing window
1C
First iteration process: determining a first common maximum number of unprocessed blocks
1D
Second iteration process: determining a second common maximum number of unprocessed blocks
1G
1E
Determining a required number of iteration processes
1F
Processing the unprocessed messages

Fig. 1

2/6

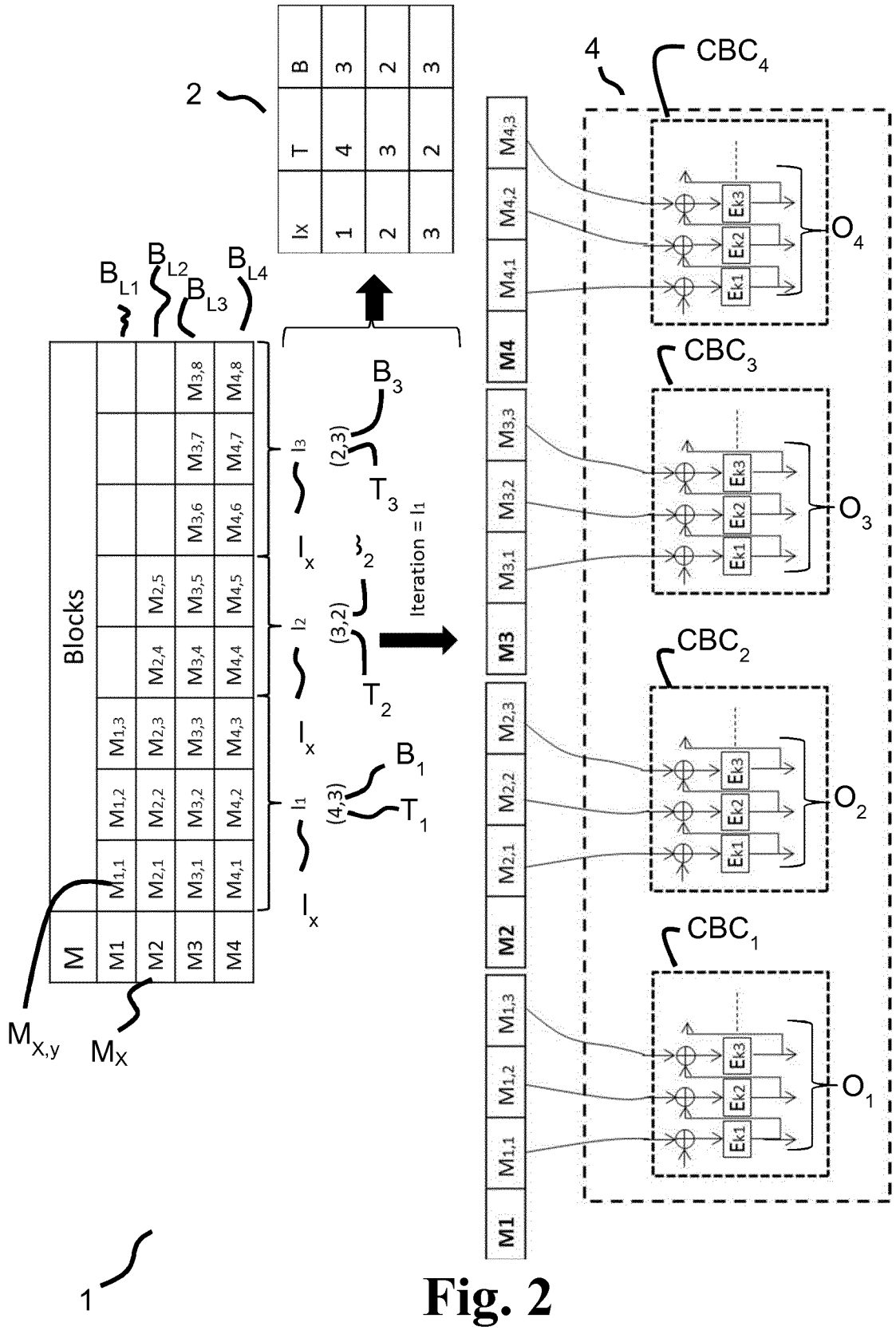
2
Ix | T | B
1 | 4 | 3
2 | 3 | 2
3 | 2 | 3
Blocks
M | M1 | M2 | M3 | M4
M1,1 | M2,1 | M3,1 | M4,1
M1,2 | M2,2 | M3,2 | M4,2
M1,3 | M2,3 | M3,3 | M4,3
M2,4 | M3,4 | M4,4
M2,5 | M3,5 | M4,5
M3,6 | M4,6
M3,7 | M4,7
M3,8 | M4,8
BL1
BL2
BL3
BL4
Ix
I1
I2
I3
(4,3)
(3,2)
(2,3)
T1
T2
T3
B1
B2
B3
Iteration = I1
MX,y
MX
1
4
CBC4
CBC3
CBC2
CBC1
M1
M2
M3
M4
Ek1
Ek2
Ek3
O1
O2
O3
O4

Fig. 2

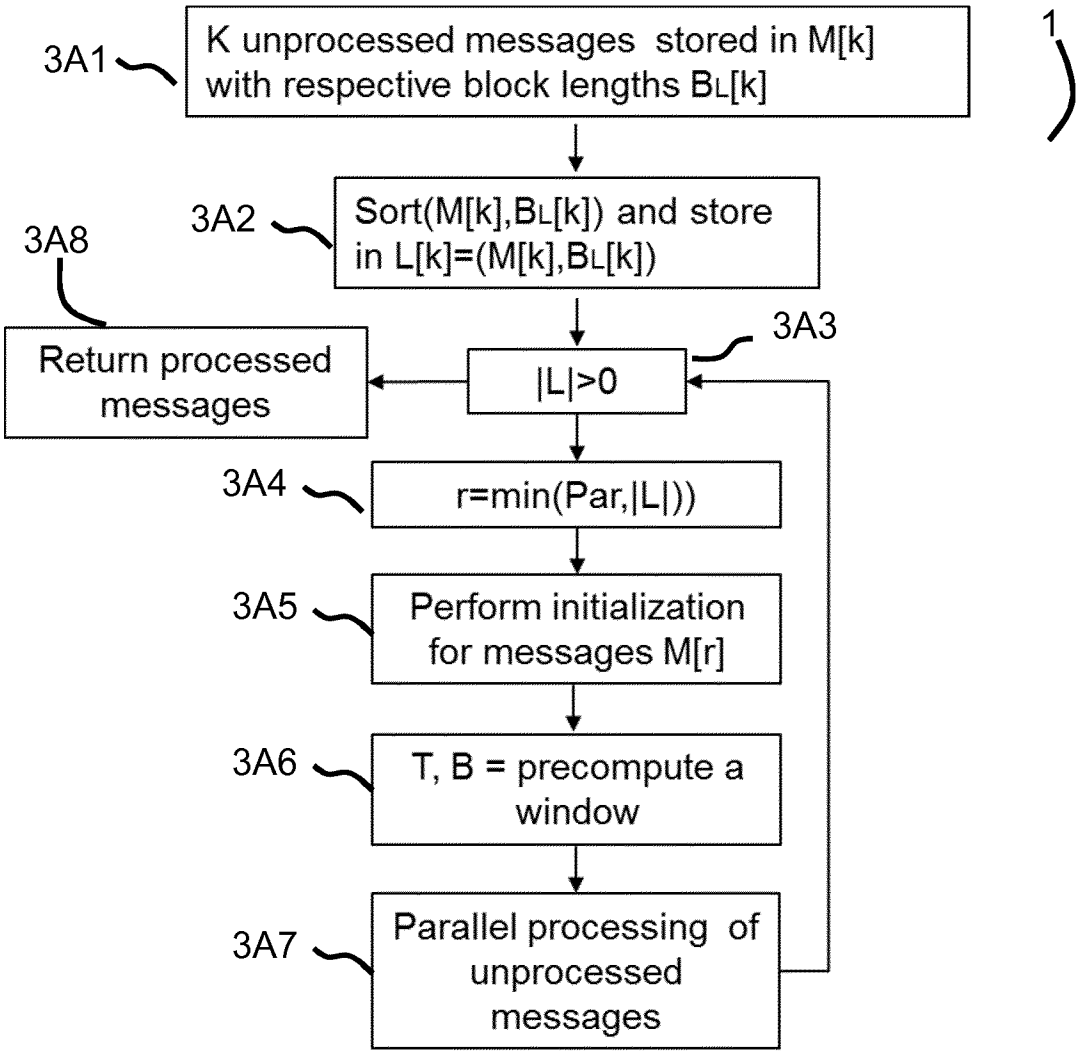
1
3A1
K unprocessed messages stored in M[k] with respective block lengths BL[k]
3A2
Sort(M[k],BL[k]) and store in L[k]=(M[k],BL[k])
3A3
|L|>0
3A8
Return processed messages
3A4
r=min(Par,|L|))
3A5
Perform initialization for messages M[r]
3A6
T, B = precompute a window
3A7
Parallel processing of unprocessed messages

Fig. 3A

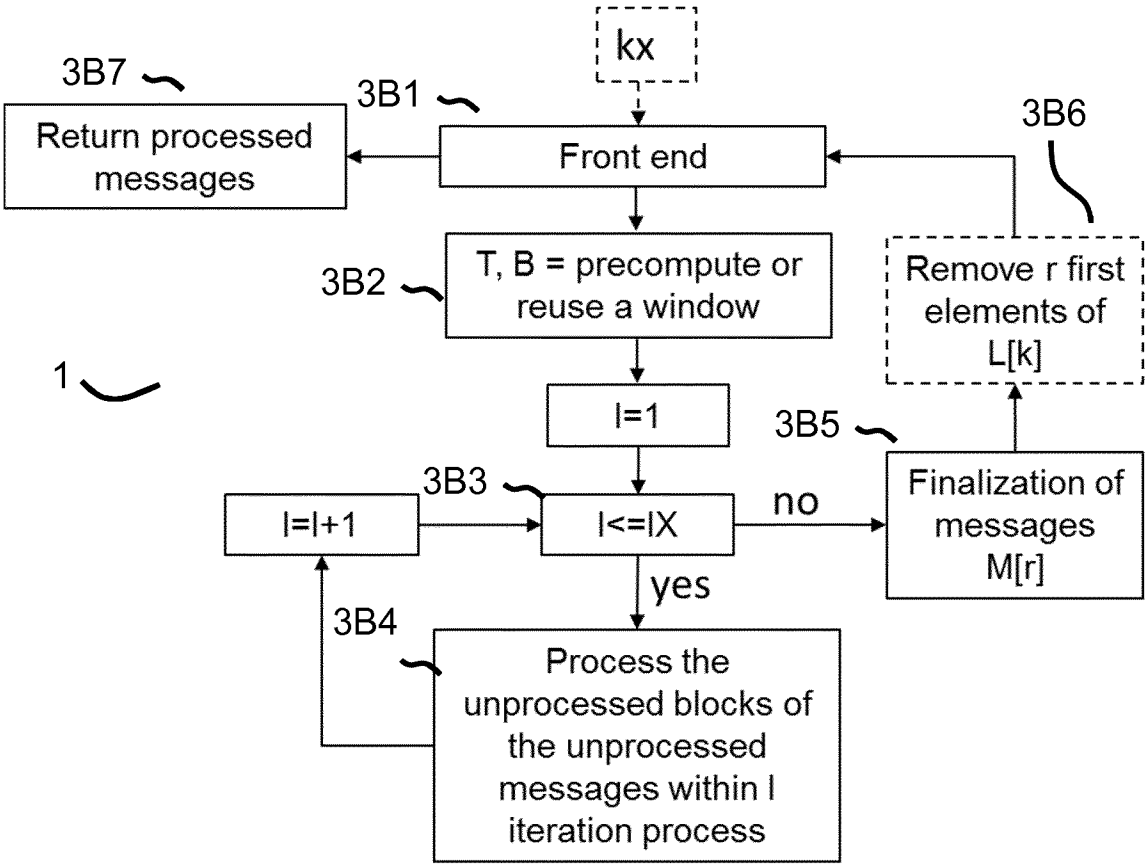
kx
3B7
3B1
3B6
Return processed messages
Front end
3B2
T, B = precompute or reuse a window
Remove r first elements of L[k]
1
I=1
3B5
3B3
I=I+1
I<=IX
no
Finalization of messages M[r]
yes
3B4
Process the unprocessed blocks of the unprocessed messages within I iteration process

Fig. 3B

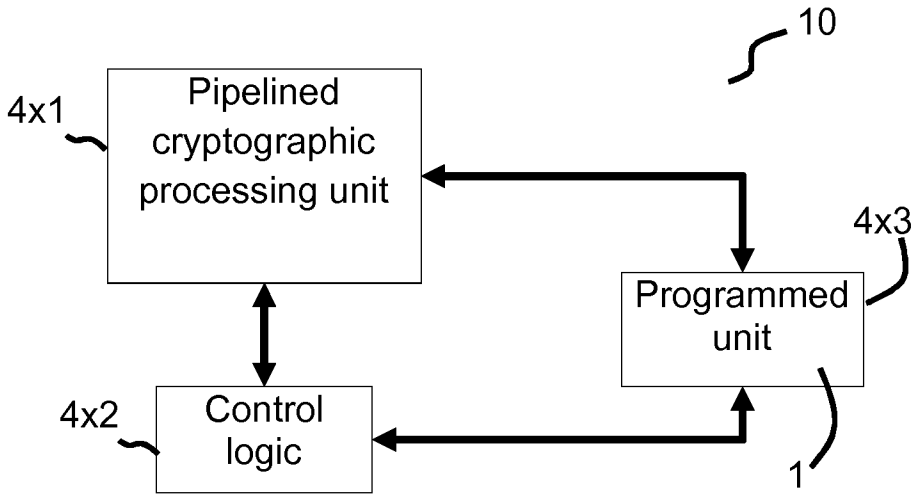
10
4x1
Pipelined cryptographic processing unit
4x3
Programmed unit
4x2
Control logic
1

Fig. 4

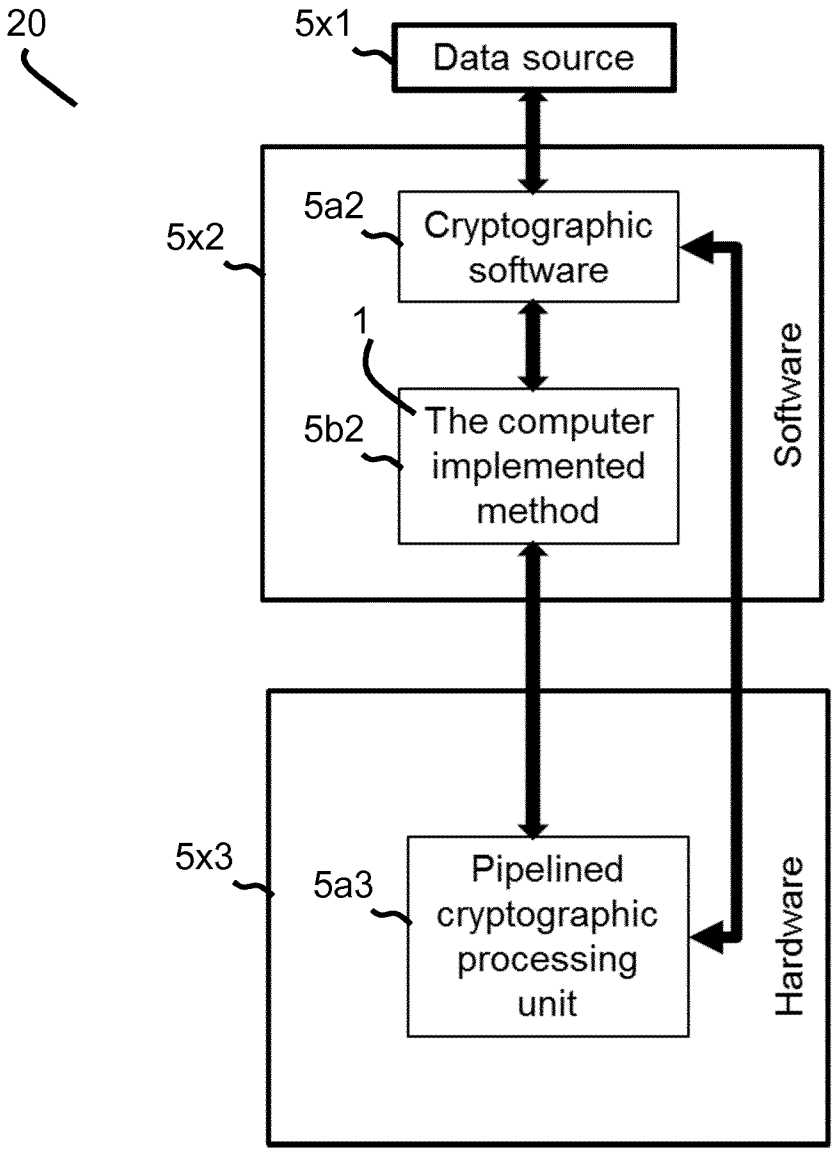
20
5x1
Data source
5x2
5a2
Cryptographic software
1
5b2
The computer implemented method
Software
5x3
5a3
Pipelined cryptographic processing unit
Hardware

Fig. 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2016/054761

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/06 G09C1/00 G06F9/38
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 G09C G06F

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	Vinodh Gopal ET AL: "Processing Multiple Buffers in Parallel to Increase Performance on Intel Architecture Processors White Paper", , 1 July 2010 (2010-07-01), XP055209448, Retrieved from the Internet: URL:http://www.intel.de/content/dam/www/public/us/en/documents/white-papers/communications-ia-multi-buffer-paper.pdf [retrieved on 2015-08-25] last two paragraphs page 6, section Performance (page 10 to 14); page 20 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C.

[X] 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
24 May 2016	03/06/2016

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	Manet, Pascal

1

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Shay Gueron: "Intel Advanced Encryption Standard (AES) New Instructions Set rev 3.01", , 1 September 2012 (2012-09-01), XP055210876, Retrieved from the Internet: URL:https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf [retrieved on 2015-09-02] page 48 - page 49; figure 37	1-15
X	Zadia Codabux-Rossan ET AL: "Perfromance of Interleaved Cipher Block Chaining in CCMP" In: "Novel Algorithms and Techniques in Telecommunications and Networking", 12 December 2009 (2009-12-12), Springer, XP055210749, ISBN: 978-9-04-813661-2 pages 53-58, section IV	1-15
A	Ryad Benadjila: "Use of the AES instruction set", , 18 October 2012 (2012-10-18), XP055209451, Retrieved from the Internet: URL:https://www.cosic.esat.kuleuven.be/ecrypt/AESday/slides/Use_of_the_AES_Instruction_Set.pdf [retrieved on 2015-08-25] slides 50 to 52 and 54	2
X	KR 2002 0071328 A (FIELDCOM COMPANY LTD [KR]; MISSION TELECOM COMPANY LTD [KR]) 12 September 2002 (2002-09-12) abstract; figures 2,3	1-15
X	US 7 603 549 B1 (KAY RONY [US]) 13 October 2009 (2009-10-13) column 7, lines 3-36; figure 7	1-15
X	DANSEREAU R M ET AL: "Reducing Packet Loss in CBC Secured VoIP using Interleaved Encryption", ELECTRICAL AND COMPUTER ENGINEERING, CANADIAN CONFERENCE ON, IEEE, PI, 1 May 2006 (2006-05-01), pages 1320-1324, XP031004765, ISBN: 978-1-4244-0038-6 section 3	1-15

-/--

1

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
T	Kahraman Akdemir ET AL: "White Paper Breakthrough AES Performance with Intel AES New Instructions", , 1 January 2010 (2010-01-01), XP055210880, Retrieved from the Internet: URL:http://software.intel.com/sites/default/files/m/d/4/1/d/8/10TB24_Breakthrough_AES_Performance_with_Intel_AES_New_Instructions.final.secure.pdf [retrieved on 2015-09-02] page 5, paragraph 2 -----	2
X,P	ANDREY BOGDANOV ET AL: "Comb to Pipeline: Fast Software Encryption Revisited", INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH,, vol. 20160119:132339, 19 January 2016 (2016-01-19), pages 1-26, XP061020018, [retrieved on 2016-01-19] the whole document section 3 -----	1-15

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2016/054761

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
KR 20020071328 A	12-09-2002	NONE	
US 7603549 B1	13-10-2009	NONE	