

## **SYSTEM AND METHOD FOR PERFORMING CRYPTOGRAPHIC OPERATIONS**

### **Field of the Invention**

This invention relates to a system and method for performing cryptographic operations on a message using a computing device, whereby the numerical representation of the message is squared as part of the cryptographic operations.

### **Summary of the Prior Art**

Most computer applications require the manipulation of large numbers such as the addition or subtraction of a large number or as more commonly used in cryptographic operations, the multiplication of large numbers followed by the modulus of the result. Cryptographic operations are nowadays widely adopted for the scrambling of private messages or for the securing of online transactions between parties. At the very minimum, most cryptographic systems require very large numbers, e.g. large numbers comprising 1024 bits or larger, to be multiplied together.

An example of a cryptographic scheme that requires the use of big number squaring is the commonly used Rivest, Shamir and Adleman ("RSA") cryptographic algorithm. A user of a RSA scheme will first create a public key by multiplying two large prime numbers together. The public key along with an auxiliary value is then made public. Anyone having access to the public key may then utilize this key to encrypt a message under this scheme. If the generated public key is sufficiently large, only someone with knowledge of the prime numbers may decrypt the encrypted message. Hence, it is in the interest of a RSA scheme's user to utilize prime numbers that are as large as possible during the generation of the public key for the RSA scheme. The downside to this is that the amount of processing

power required for the generation of the public key and for the decryption of the cipher text increases as the sizes of the prime numbers increases.

Existing methods of computing a square of a number involves the simple multiplication of each integer within the two identical numbers. The obtained result is then summed at the end. However, such simple methods are only useful when small numbers are to be squared. In cryptographic operations, the numbers that are to be squared typically involve numbers of 1024-bits or more. Hence, those skilled in the art have proposed various ways of optimizing the squaring process. One method that has been proposed by those skilled in the art for squaring a number involves the de-duplication of simple multiplication terms which repeat themselves, e.g.  $(A_1 \times A_0) = (A_0 \times A_1)$ . The result obtained from this de-duplication process is then multiplied by two to obtain the final result which is then subsequently summed with the individually squared integers of the number.

However, these methods do not allow the parallel computing power of modern processing units to be fully harnessed as these methods proposed by those skilled in the art involve multiplication methods that do not allow parallel or simultaneous computing of the various multiplication terms.

For the above reasons, those skilled in the art are constantly striving to come up with a system and method that allows cryptographic operations such as the squaring of arbitrary sized messages to be carried out efficiently using processing units with parallel computing capabilities.

### **Summary of the Invention**

We provide systems and methods to improve the execution of cryptographic operations on a message A as set out by the embodiments in accordance with the invention.

A first advantage of embodiments of systems and methods in accordance with embodiments of the invention is that the proposed system and method allows a message of any arbitrary size or length to be efficiently squared as part of a cryptographic operation.

A second advantage of embodiments of systems and methods in accordance with embodiments of the invention is that the process of computing the non-squaring terms of the message may be distributed across a plurality of processors configured in parallel. This allows the overall calculation time to be greatly reduced.

A third advantage of embodiments of systems and methods in accordance with embodiments of the invention is that once a cryptographic operation of squaring a message has been partially completed, a modulo operation may then be applied to the results using a Barrett or Montgomery modular reduction algorithm.

According to a first aspect of the invention, a computer implemented method for performing cryptographic operations on a message A using a computing device having at least a first processor, the method comprising the computing device: retrieving the message A from a register, wherein the message A comprises contiguous ordered digits, each digit being characterized by a weight  $w$  and identified as  $A[w]$ , where  $w$  is an integer from 0 to  $(n-1)$ , and where  $n$  represents the maximum number of digits in the message; performing cryptographic operations on the message by squaring the message A, wherein the squaring the message A comprises: computing and summing, by the first processor, non-squaring terms for the message A, wherein values for the computed and summed non-squaring terms are stored in a register  $R[ ]$ , wherein register  $R[ ]$  comprises a  $(n \times 2)$ -bit sized array; and computing, by the first processor, squaring terms for the message A and adding, by the first processor, the computed squared terms to the values stored in register  $R[ ]$  wherein the

values stored in register  $R[ ]$  are multiplied by two before the values are added to the computed squared terms, to generate the square of message A.

With reference to the first aspect, in a first possible implementation manner of the first aspect, wherein before the step of computing the squaring terms for the message A, the method comprises the step of: determining if a value in  $R[2 \cdot w + 1]$  of the register  $R[ ]$  has been set; and if the value in  $R[2 \cdot w + 1]$  has been set, commencing the computing of the squaring term for a digit represented by  $A[w]$  simultaneously as the step of computing and summing non-squaring terms for the message A is being carried out.

With reference to the first aspect, in a second possible implementation manner of the first aspect, the computing and the summing the non-squaring terms for the message A include: initializing  $(MSD, LSD) = 0$  and all terms in  $R[ ] = 0$ ; for each integer  $i$  with  $0 \leq i < (n/2)$ , performing by the first processor the following operations:

for each integer  $j$  with  $(i+1) \leq j < (n-i)$  do,

set  $(MSD, LSD) = A[i] \times A[j] + MSD + R[j+i]$ ; and

set  $R[j+i] = LSD$ ;

set count =  $i$ ;

when  $(count < 2 \cdot i)$  do,

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase count by 1;

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase  $j$  by 1;

set  $R[j+count] = R[j+count] + MSD$ ;

returning register R with set values; and

wherein the computing the squaring terms for the message A and adding the computed squared terms to the terms stored in register R include: initializing (MSD, LSD) = 0;

for each integer i with  $0 \leq i < n$ , performing by the first processor the following operations:

set (MSD,LSD) =  $A[i] \times A[i] + \text{MSD}$ ;

set (MSD', LSD) =  $R[i \cdot 2] \cdot 2 + \text{LSD}$ ;

set  $R[i \cdot 2] = \text{LSD}$ ;

set (MSD,LSD) =  $R[i \cdot 2 + 1] \cdot 2 + \text{MSD}' + \text{MSD}$ ; and

set  $R[i \cdot 2 + 1] = \text{LSD}$ ;

returning values stored in register R as the square of the message A,

where, MSD is a most significant digit of an integer and LSD is a least significant digit of the integer.

With reference to the second possible implementation manner of the first aspect, in a third possible implementation manner of the first aspect, the computing and the summing the non-squaring terms for the message A further comprises: determining, by the first processor, if a value of  $R[2 \cdot i - 1]$  in the register R has been set; and in response to a determination that the value of  $R[2 \cdot i - 1]$  has been set, completing the remainder of the operations for the value of i using the first processor and simultaneously, increasing the value of i by 1 wherein, for each integer i whereby  $i < (n/2)$ , performing by a second processor the following operations:

for each integer j with  $(i+1) \leq j < (n-i)$  do,

set (MSD, LSD) =  $A[i] \times A[j] + \text{MSD} + R[j+i]$ ; and

set  $R[j+i] = \text{LSD}$ ;

set count = i;

```

when (count < 2·i) do,
    set (MSD, LSD) = A[j] x a[count] + MSD + R[j+count];
    set R[j+count] = LSD;
    increase count by 1;
    set (MSD,LSD) = A[j] x A[count] + MSD + R[j+count];
    set R[j+count] = LSD;
increase j by 1; and
    returning register R with set values.

```

With reference to the first aspect, in a fourth possible implementation manner of the first aspect the method further comprises applying a Montgomery modular reduction for a modulus value of  $m$  to the square of the message  $A$ .

With reference to the first aspect, in a fifth possible implementation manner of the first aspect the method further comprises applying a Barrett or Montgomery modular reduction for a modulus value of  $m$  to the square of the message  $A$ .

### **Brief Description of the Drawings**

The above advantages and features in accordance with this invention are described in the following detailed description and are shown in the following drawings:

Figure 1 illustrating a block diagram representative of processing systems providing embodiments in accordance with embodiments of the invention;

Figure 2 illustrating an exemplary message  $A$  that is squared in accordance with embodiments of the invention;

Figure 3 illustrating an exemplary message A comprising four digits that is squared in accordance with embodiments of the invention;

Figure 4 illustrating a flow diagram of a process for performing cryptographic squaring operations in accordance with embodiments of the invention.

### **Detailed Description**

This invention relates to a system and method for performing cryptographic operations on a message using a computing device, whereby the numerical representation of the message is squared as part of the cryptographic operations.

One skilled in the art will recognize that many functional units in this description have been labelled as modules throughout the specification. The person skilled in the art will also recognize that a module may be implemented as circuits, logic chips or any sort of discrete component. Further, one skilled in the art will also recognize that a module may be implemented in software which may then be executed by a variety of processors. In embodiments of the invention, a module may also comprise computer instructions or executable code that may instruct a computer processor to carry out a sequence of events based on instructions received. The choice of the implementation of the modules is left as a design choice to a person skilled in the art and does not limit the scope of this invention in any way.

Further, a person skilled in the art will recognize that messages transmitted by computing systems are usually expressed as numbers and these numbers are commonly expressed as binary notations. However, for ease of describing the invention, the digits in the message described in this description, i.e. message A, will be described as base 10 (i.e. decimal) numbers. It is further assumed that each processor that is used to perform

multiplication operations on the digits in message A may only multiply single digits at any one time.

In addition to the above, it should also be noted that the use of algorithms and symbolic representations of operations on data bits within a computer or processor memory are the means used by those skilled in the art to describe their work. In this description, an algorithm is interpreted to be a self-consistent sequence of steps that produces a desired outcome and that the steps are those that require physical manipulations of physical quantities. Typically, these quantities are in the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. Commonly, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like.

Unless specifically stated otherwise, it is appreciated that throughout the description, terms such as “calculating” or “determining” or “displaying” or “processing” or “computing” or the like, refer to the action and processes of a computer system, or of a similar electronic computing device, that is capable of manipulating and transforming data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the memory of the computer system or registers.

Figure 1 illustrates a block diagram of a computer system or module for implementing embodiments in accordance with embodiments of the invention. One skilled in the art will recognize that the exact configuration of the module of the computer system may be different and the exact configuration of each module or computer system may vary and Figure 1 is provided by way of example only.

In embodiments of the invention, computer system 100 comprises controller 101 and user interface 102. User interface 102 is arranged to enable manual interactions between a user and module 100 and for this purpose includes the input/output components required for the user to enter instructions to control module 100. A person skilled in the art will recognize that components of user interface 102 may vary from embodiment to embodiment but will typically include one or more of display 140, keyboard 135 and track-pad 136.

Controller 101 is in data communication with user interface 102 via bus 115 and includes memory 120, processor 105 mounted on a circuit board that processes instructions and data for performing the method of this embodiment, an operating system 106, an input/output (I/O) interface 130 for communicating with user interface 102 and a communications interface, in this embodiment in the form of a network card 150. Network card 150 may, for example, be utilized to send data from electronic device 100 via a wired or wireless network to other processing devices or to receive data via the wired or wireless network. Wireless networks that may be utilized by network card 150 include, but are not limited to, Wireless-Fidelity (Wi-Fi), Bluetooth, Near Field Communication (NFC), cellular networks, satellite networks, telecommunication networks, Wide Area Networks (WAN) and etc.

Memory 120 and operating system 106 are in data communication with CPU 105 via bus 110. The memory components include both volatile and non-volatile memory and more than one of each type of memory, including Random Access Memory (RAM) 120, Read Only Memory (ROM) 125 and a mass storage device 145, the last comprising one or more solid-state drives (SSDs). Memory 120 also includes secure storage 146 for securely storing secret keys, or private keys. It should be noted that the contents within secure storage 146 are only accessible by a super-user or administrator of module 100 and may not be

accessed by any user of module 100. One skilled in the art will recognize that the memory components described above comprise non-transitory computer-readable media and shall be taken to comprise all computer-readable media except for a transitory, propagating signal. Typically, the instructions are stored as program code in the memory components but can also be hardwired. Memory 120 may include a kernel and/or programming modules such as a software application that may be stored in either volatile or non-volatile memory.

Herein the term “processor” is used to refer generically to any device or component that can process such instructions and may include: a microprocessor, microcontroller, programmable logic device or other computational device. That is, processor 105 may be provided by any suitable logic circuitry for receiving inputs, processing them in accordance with instructions stored in memory and generating outputs (for example to the memory components or on display 140). In this embodiment, processor 105 may be a single core or multi-core processor with memory addressable space. In one example, processor 105 may be multi-core, comprising—for example—an 8 core CPU.

Figure 2 depicts an exemplary method of performing cryptographic operations on message A by squaring message A in accordance with embodiments of the invention. The squaring of message A may be carried out by computer system 100 and it should be noted that message A may be preloaded into the computer system or may have been received by the computer system during an earlier operation. As illustrated in Figure 2, it can be seen that message A comprises contiguous ordered digits whereby each digit in message A may be characterized by a weight “w” and may be identified as  $A[w]$ , where w is an integer from 0 to (n-1), and n represents the maximum number of digits in message A. In this example, it can be seen that n is equal to six as there are six digits in message A and message A may be characterized as follows:  $A = \{ A[5], A[4], A[3], A[2], A[1], A[0] \}$ . Although this example

illustrates that message A comprises of only six digits, one skilled in the art will recognize that message A may comprise of any arbitrary number of digits without departing from this invention and that this example is for illustration purposes only.

It can also be said that message A is contained in an  $A[ ]$  array that has a series of locations running from 0 to  $(n-1)$  where  $n$  represents the number of digits in message A. In the example illustrated in Figure 2, it can be seen that the  $A[ ]$  array comprises of six locations running from 0 to 5 (although locations 0-9 are shown for illustration purposes).

Non-squaring terms 210 for message A are then computed. This is done by multiplying each digit in message A with another digit except itself. For example, when the digit represented by  $A[0]$  is selected, this digit is then multiplied with all the other digits in message A, i.e.  $A[0] \times A[1]$ ;  $A[0] \times A[2]$ ;  $A[0] \times A[3]$ ;  $A[0] \times A[4]$ ;  $A[0] \times A[5]$ , except itself, i.e.  $A[0]$ . This process is then repeated until all the digits have been selected thereby producing an unarranged version of non-squaring terms 210.

A location index of each result obtained from each multiplication operation within non-squaring terms 210 may then be derived from the weights or locations of the digits involved in each multiplication operation. In particular, this location index for each result is obtained from the sum of the weights or locations of the digits involved in each multiplication operation. Each result is then arranged in non-squaring term 210 based on its location index whereby the least significant bit of the result is located directly below a location indicated by its location index. For example, for a multiplication operation involving digits  $A[0] \times A[1]$ , the sum of its weights is "1". This means that its location index is "1" and as such, the least significant bit of this result will be located under location "1" while the most significant bit of this result will be located under location "2" as illustrated in Figure 2.

For clarity, another example is provided. In this other example, for another multiplication operation involving digits  $A[3] \times A[4]$ , the sum of its weights is “7”. This means that its location index is “7” and as such, the least significant bit of this result will be located under location “7” while the most significant bit of this result will be located under location “8”. The location index for each result is then computed. Based on the calculated location indexes, the results are then arranged to produce non-squaring terms 210 as illustrated in Figure 2.

Non-squaring terms 210 are then summed in accordance with normal procedures to produce register  $R[ ]$  wherein register  $R[ ]$  comprises a  $(n \times 2)$ -bit sized array. In the example illustrated in Figure 2, this means that register  $R[ ]$  comprises an array having 12 locations as the maximum number of digits in message A is 6, i.e.  $n = 6$ . Register  $R[ ]$  is then multiplied by two to produce register 215.

As the non-squaring terms have now been summed together and multiplied by two, the next step is to then compute the squaring terms for message A. Squaring terms 205 are computed by multiplying each digit in message A with itself. For example, when the digit represented by  $A[0]$  is selected, this digit is then multiplied by itself, i.e.  $A[0] \times A[0]$ . This process is then repeated until all the digits have been selected and multiplied by themselves thereby producing an unarranged version of squaring terms 205. Similarly, a location index of each of squaring terms 205 may be derived from the weights or locations of the digits involved in each multiplication operation. In particular, this location index for each result is obtained from the sum of the weights or locations of the digits involved in each multiplication operation. Each result is then arranged in squaring term 205 based on its location index whereby the least significant bit of the result is located directly below a location indicated by its location index. For example, for a multiplication operation involving digits  $A[0] \times A[0]$ , the

sum of its weights is “0”. This means that its location index is “0” and as such, the least significant bit of this result will be located under location “0” while the most significant bit of this result will be located under location “1” as illustrated in Figure 2. Based on the calculated location indexes, the results are then arranged to produce squaring terms 205 as illustrated in Figure 2.

Register 215 is then simply summed with squaring terms 205 to produce register 220. The result of this summation as stored in register 200 represents the square of message A. The details of the methods described above can probably be best understood by referring to the exemplary numbers illustrated in Figure 3.

Figure 3 depicts an exemplary method of squaring message A in accordance with embodiments of the invention. As illustrated in Figure 3, it can be seen that message A may be characterized as follows:  $A = \{ A[3] = 2, A[2] = 3, A[1] = 5, A[0] = 4 \}$  and that the total number of digits in message A is 4, i.e.  $n = 4$ , whereby each digit corresponds to a location between 0 to 3.

Non-squaring terms 310 for message A are then computed. This is done by multiplying each digit in message A with another digit except itself. This produces the following:  $A[1] \times A[0] = 20$ ;  $A[2] \times A[0] = 12$ ;  $A[3] \times A[0] = 08$ ;  $A[2] \times A[1] = 15$ ;  $A[3] \times A[1] = 10$ ;  $A[3] \times A[2] = 06$ .

A location index of each result obtained from each multiplication operation within non-squaring terms 310 may then be derived from the sum of the weights or locations of the digits involved in each multiplication operation. Each result is then arranged in non-squaring term 310 based on its location index whereby the least significant bit of the result is located directly below a location indicated by its location index.

For example, for a multiplication operation involving digits  $A[0] \times A[1]$ , the sum of its weights is “1” while the result of its multiplication is “20”. This means that its location index is “1” and as such, the least significant bit of this result which is “0” will be located under location “1” while the most significant bit of this result which is “2” will be located under location “2” as illustrated in Figure 2. As another example, for another multiplication operation involving digits  $A[2] \times A[3]$ , the sum of its weights is “5” while the result of its multiplication is “6”. This means that its location index is “5” and as such, the least significant bit of this result which is “6” will be located under location “5” while the most significant bit of this result which is “0” will be located under location “6”. The location index for each result is then computed and then based on the calculated location indexes, the results are then arranged to produce the arrangement of non-squaring terms 310 as illustrated in Figure 3.

Non-squaring terms 310 are then summed in accordance with normal procedures to produce register  $R[ ]$  wherein register  $R[ ]$  comprises a  $(n \times 2)$ -bit sized array which in this example is an 8-bit sized array having 8 locations. The value of register  $R$  is then multiplied by two to produce register 315.

As the non-squaring terms have now been summed together and multiplied by two, the next step is to then compute the squaring terms for message  $A$ . Squaring terms 305 are computed by multiplying each digit in message  $A$  with itself. This produces the following:  $A[0] \times A[0] = 16$ ;  $A[1] \times A[1] = 25$ ;  $A[2] \times A[2] = 09$ ;  $A[3] \times A[3] = 04$ .

A location index of each of squaring terms 305 may then be derived from the sum of the weights or locations of the digits involved in each multiplication operation. Each result is then arranged in squaring term 305 based on its location index whereby the least significant bit of the result is located directly below a location indicated by its location index. For

example, for a multiplication operation involving digits  $A[0] \times A[0]$ , the sum of its weights is “0”. This means that its location index is “0” and as such, the least significant bit of this result which is “6” will be located under location “0” while the most significant bit of this result which is “1” will be located under location “1” as illustrated in Figure 3. Based on the calculated location indexes, the results are then arranged to produce the arrangement of squaring terms 305 as illustrated in Figure 3.

Register 315 is then summed with squaring terms 305 to produce register 320 which then contains the result of the square of message A.

In accordance with embodiments of the invention, a further cryptographic operation may then be carried out on the result of the square of message A (as contained within register 220) by applying a modulus P to the result. The modular reduction of the result of the square of message A may be carried out using a Montgomery or Barrett reduction. The detailed workings of these two modular reductions are omitted for brevity as the workings of these two reductions are known to those skilled in the art.

An example algorithm that may be implemented to compute the non-squaring terms 210 as illustrated in Figure 2 is shown below in Algorithm 1. It should be noted that the integers MSD, LSD, i, j, n, and count may be values or variables that may be stored in the memory of the computer system.

---

#### Algorithm 1

---

|         |                                                                                                                            |
|---------|----------------------------------------------------------------------------------------------------------------------------|
| Input:  | message A where each digit is characterized by a weight w and identified as $A[w]$ , where w is an integer from 0 to (n-1) |
| Output: | register R[ ]                                                                                                              |
| Step 1: | (MSD, LSD) = 0; R[ ] = 0;                                                                                                  |

```

Step 2:   for (i=0; i < (n/2); i++) {
           for (j=i+1; j < n-i; j++) {
               (MSD, LSD) = A[ i ] x A[ j ] + MSD + R [j+i];
               R[j+i] = LSD;
           }

           count = i;
           while (count < 2·i)    {
               (MSD, LSD) = A[ j ] x A[ count ] + MSD + R [j+count];
               R [j+count] = LSD;
               count = count + 1;
               (MSD, LSD) = A[ j ] x A[count] + MSD + R [j+count];
               R [j+count] = LSD;
               j++;
           }

           R [j+count] = R [j+count] + MSD
       }

```

---

An example algorithm that may be implemented to compute the squaring terms 205 and adding the computed squaring terms 205 to register R that has been multiplied by two is shown below in Algorithm 2. It should be noted that the integers MSD, MSD', LSD, i, and n may be values or variables that may be stored in the memory of the computer system and that register R is obtained as the output from Algorithm 1.

---

#### Algorithm 2

---

Input:        message A where each digit is characterized by a weight w and identified as A[w], where w is an integer from 0 to (n-1); and  
                register R as obtained from Algorithm 1

Output:       register R[ ]

Step 1:        (MSD, LSD) = 0;

```

Step 2:      for (i=0; i < n; i++)    {
                (MSD, LSD) = A[ i ] x A[ i ] + MSD;
                (MSD', LSD) = R[ i·2 ]·2 + LSD;
                R[ i·2 ] = LSD;
                (MSD, LSD) = R [i·2 + 1] ·2 + MSD' + MSD; and
                R [i·2 + 1] = LSD;
            }

```

---

In accordance with embodiments of the invention, the computer system carrying out the cryptographic operations may comprise two or more processors. Referring to Figure 2, if the computer system comprises five processors that are configured to work in parallel, the computer system may utilize these five processors to compute each outer  $i$ -th loop during the computation of non-squaring terms 210 after a small amount of time has lapsed between each  $i$ -th loop. It is necessary to allow a small amount of time to lapse first because the computation of an  $(i+1)^{\text{th}}$  loop requires certain input from the  $i$ -th loop. One skilled in the art will recognize that any number of processors that are configured to work in parallel may be utilized without departing from this invention.

In particular, for this example, a first processor may be used to compute loop 251 for the outer  $0^{\text{th}}$  loop, a second processor may be used to compute loop 252 for the outer  $1^{\text{th}}$  loop, a third processor may be used to compute loop 253 for the outer  $2^{\text{th}}$  loop, a fourth processor may be used to compute loop 254 for the outer  $3^{\text{rd}}$  loop, and a fifth processor may be used to compute loop 255 for the outer  $4^{\text{th}}$  loop provided that a small amount of time is allowed to lapse between each loop. In particular, the computation of an  $(i+1)^{\text{th}}$  loop using another processor may begin once the  $i$ -th loop has completed the calculation for  $R[2 \cdot i - 1]$  in register R during the calculation of the non-squaring terms 210. By doing so, this increases the speed at which the non-squaring terms may be obtained.

In accordance with embodiments of the invention, the computer system carrying out the cryptographic operations may comprise of at least two processors. A first processor may be tasked to carry out the computation of the non-squaring terms for a message A as described above in Algorithm 1 while a second processor, in response to a determination that a value in  $R[2 \cdot w + 1]$  of the register R[ ] has been set, may be tasked to commence computing the squaring term for a digit represented by  $A[w]$  simultaneously (as described in Algorithm 2 above) as the step of computing and summing non-squaring terms for the remainders of the terms in message A is being carried out.

In accordance with an embodiment of the invention, a computer implemented method for performing cryptographic operations on a message A using a computing device having at least a first processor, comprises the following two steps:

Step 1, for performing cryptographic operations on a message A using a computing device having at least a first processor;

Step 2, performing cryptographic operations on the message by squaring the message A, wherein the squaring the message A comprises:

computing and summing, by the first processor, non-squaring terms for the message A, wherein values for the computed and summed non-squaring terms are stored in a register R[ ], wherein register R[ ] comprises a  $(n \times 2)$ -bit sized array; and

computing, by the first processor, squaring terms for the message A and adding, by the first processor, the computed squared terms to the values stored in register R[ ], wherein the values stored in register R[ ] are multiplied by two before the values are added to the computed squared terms, to generate the square of message A.

In order to provide such a system or method, a process is needed for performing cryptographic operations on a message A. The following description and Figure 4 describes embodiments of processes that provide processes in accordance with this invention.

Figure 4 illustrates process 400 that is performed by a module or computer processor installed within computing device to perform cryptographic operations on a message A in accordance with embodiments of this invention. Process 400 begins at step 405 whereby process 400 retrieves message A from a register or a memory within the computing device. It should be noted that the retrieved message A comprises a series of contiguous ordered digits whereby each digit in message A is characterized by a weight "w" and may be identified by a term  $A[w]$ . Further, the weight "w" may comprise any integer between "0" and "n-1" where n represents the maximum number of digits contained within message A.

Process 400 then proceeds to perform cryptographic operations on message A. In embodiments of the invention, the cryptographic operation performed at this step, i.e. step 410, comprises the step of squaring message A.

Process 400 begins the squaring process at step 415 by computing non-squaring terms of message A. The computed non-squaring terms are then summed together and stored within a register R[ ]. The computation of the non-squaring terms at this step may be carried out using Algorithm 1 as described above.

At the next step, step 420, process 400 then computes the squaring terms of message A. After the result in register R has been multiplied by two, process 400 then adds the computed squaring terms to register R to produce the square of message A. Process 400 then ends.

The above is a description of embodiments of a system and process in accordance with the present invention as set forth in the following claims. It is envisioned that others may and will design alternatives that fall within the scope of the following claims.

**CLAIMS:**

1. A computer implemented method for performing cryptographic operations on a message A using a computing device having at least a first processor, the method comprising the computing device:

retrieving the message A from a register, wherein the message A comprises contiguous ordered digits, each digit being characterized by a weight  $w$  and identified as  $A[w]$ , where  $w$  is an integer from 0 to  $(n-1)$ , and where  $n$  represents the maximum number of digits in the message;

performing cryptographic operations on the message by squaring the message A, wherein the squaring the message A comprises:

computing and summing, by the first processor, non-squaring terms for the message A, wherein values for the computed and summed non-squaring terms are stored in a register  $R[ ]$ , wherein register  $R[ ]$  comprises a  $(n \times 2)$ -bit sized array; and

computing, by the first processor, squaring terms for the message A and adding, by the first processor, the computed squared terms to the values stored in register  $R[ ]$  wherein the values stored in register  $R[ ]$  are multiplied by two before the values are added to the computed squared terms, to generate the square of message A.

2. The computer implemented method according to claim 1 wherein before the step of computing the squaring terms for the message A, the method comprises the step of:

determining if a value in  $R[2 \cdot w + 1]$  of the register  $R[ ]$  has been set; and

if the value in  $R[2 \cdot w + 1]$  has been set, commencing the computing of the squaring term for a digit represented by  $A[w]$  simultaneously as the step of computing and summing non-squaring terms for the message  $A$  is being carried out.

3. The computer implemented method according to claim 1,

wherein the computing and the summing the non-squaring terms for the message  $A$  include:

initializing  $(MSD, LSD) = 0$  and all terms in  $R[] = 0$ ;

for each integer  $i$  with  $0 \leq i < (n/2)$ , performing by the first processor the following operations:

for each integer  $j$  with  $(i+1) \leq j < (n-i)$  do,

set  $(MSD, LSD) = A[i] \times A[j] + MSD + R[j+i]$ ; and

set  $R[j+i] = LSD$ ;

set count =  $i$ ;

when  $(count < 2 \cdot i)$  do,

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase count by 1;

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase  $j$  by 1;

set  $R[j+count] = R[j+count] + MSD$ ;

returning register  $R$  with set values; and

wherein the computing the squaring terms for the message  $A$  and adding the computed squared terms to the terms stored in register  $R$  include:

initializing  $(MSD, LSD) = 0$ ;

for each integer  $i$  with  $0 \leq i < n$ , performing by the first processor the following operations:

set  $(MSD, LSD) = A[i] \times A[i] + MSD$ ;

set  $(MSD', LSD) = R[i \cdot 2] \cdot 2 + LSD$ ;

set  $R[i \cdot 2] = LSD$ ;

set  $(MSD, LSD) = R[i \cdot 2 + 1] \cdot 2 + MSD' + MSD$ ; and

set  $R[i \cdot 2 + 1] = LSD$ ;

returning values stored in register  $R$  as the square of the message  $A$ ,

where,  $MSD$  is a most significant digit of an integer and  $LSD$  is a least significant digit of the integer.

4. The computer implemented method according to claim 3 wherein the computing and the summing the non-squaring terms for the message  $A$  further comprises:

determining, by the first processor, if a value of  $R[2 \cdot i - 1]$  in the register  $R$  has been set; and

in response to a determination that the value of  $R[2 \cdot i - 1]$  has been set, completing the remainder of the operations for the value of  $i$  using the first processor and simultaneously, increasing the value of  $i$  by 1 wherein,

for each integer  $i$  whereby  $i < (n/2)$ , performing by a second processor the following operations:

for each integer  $j$  with  $(i+1) \leq j < (n-i)$  do,

set  $(MSD, LSD) = A[i] \times A[j] + MSD + R[j+i]$ ; and

set  $R[j+i] = LSD$ ;

set count =  $i$ ;

when (count < 2·i) do,

set (MSD, LSD) =  $A[j] \times a[\text{count}] + \text{MSD} + R[j+\text{count}]$ ;

set  $R[j+\text{count}] = \text{LSD}$ ;

increase count by 1;

set (MSD,LSD) =  $A[j] \times A[\text{count}] + \text{MSD} + R[j+\text{count}]$ ;

set  $R[j+\text{count}] = \text{LSD}$ ;

increase j by 1; and

returning register R with set values.

5. The computer implemented method according to claim 1 further comprising:

applying a Montgomery modular reduction for a modulus value of m to the square of the message A.

6. The computer implemented method according to claim 1 further comprising:

applying a Barrett modular reduction for a modulus value of m to the square of the message A.

7. A system for performing cryptographic operations on a message A comprising:

a first processing unit; and

a non-transitory media readable by the first processing unit, the non-transitory media storing instructions that when executed by the first processing unit, cause the first processing unit to:

retrieve the message A from a register, wherein the message A comprises contiguous ordered digits, each digit being characterized by a weight w and identified

as  $A[w]$ , where  $w$  is an integer from 0 to  $(n-1)$ , and where  $n$  represents the maximum number of digits in the message;

perform cryptographic operations on the message by squaring the message  $A$ , wherein the squaring the message  $A$  comprises:

instructions for directing the first processing unit to:

compute and sum non-squaring terms for the message  $A$ , wherein values for the computed and summed non-squaring terms are stored in a register  $R[ ]$ , wherein register  $R[ ]$  comprises a  $(n \times 2)$ -bit sized array;

compute squaring terms for the message  $A$ ; and

add the computed squared terms to the values stored in register  $R[ ]$ , wherein the values stored in register  $R[ ]$  are multiplied by two before the values are added to the computed squared terms, to generate the square of message  $A$ .

8. The system according to claim 7 wherein before the instructions for directing the first processing unit to compute the squaring terms for the message  $A$ , the system comprises instructions for directing the first processing unit to:

determine if a value in  $R[2 \cdot w + 1]$  of the register  $R[ ]$  has been set; and

if the value in  $R[2 \cdot w + 1]$  has been set, commence the computing of the squaring term for a digit represented by  $A[w]$  simultaneously as the instructions for directing the first processing unit to compute and sum non-squaring terms for the message  $A$  is being carried out.

9. The system according to claim 7,

wherein instructions for directing the first processing unit to compute and sum the non-squaring terms for the message A include:

instructions for directing the first processing unit to

initialize  $(MSD, LSD) = 0$  and all terms in  $R[ ] = 0$ ;

for each integer  $i$  with  $0 \leq i < (n/2)$ , perform the following operations:

for each integer  $j$  with  $(i+1) \leq j < (n-i)$  do,

set  $(MSD, LSD) = A[i] \times A[j] + MSD + R[j+i]$ ; and

set  $R[j+i] = LSD$ ;

set  $count = i$ ;

when  $(count < 2 \cdot i)$  do,

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase  $count$  by 1;

set  $(MSD, LSD) = A[j] \times A[count] + MSD + R[j+count]$ ;

set  $R[j+count] = LSD$ ;

increase  $j$  by 1;

set  $R[j+count] = R[j+count] + MSD$ ;

return register  $R$  with set values; and

wherein the computing the squaring terms for the message A and adding the computed squared terms to the terms stored in register  $R$  include:

instructions for directing the first processing unit to:

initialize  $(MSD, LSD) = 0$ ;

for each integer  $i$  with  $0 \leq i < n$ , perform the following operations:

set  $(MSD, LSD) = A[i] \times A[i] + MSD$ ;

set  $(MSD', LSD) = R[i-2] \cdot 2 + LSD$ ;

set  $R[i \cdot 2] = \text{LSD}$ ;

set  $(\text{MSD}, \text{LSD}) = R[i \cdot 2 + 1] \cdot 2 + \text{MSD}' + \text{MSD}$ ; and

set  $R[i \cdot 2 + 1] = \text{LSD}$ ;

return values stored in register R as the square of the message A,

where, MSD is a most significant digit of an integer and LSD is a least significant digit of the integer.

10. The system according to claim 9 wherein the instructions for directing the first processing unit to compute and sum the non-squaring terms for the message A further comprises:

instructions for directing the first processing unit to:

determine if a value of  $R[2 \cdot i - 1]$  in the register R has been set; and

in response to a determination that the value of  $R[2 \cdot i - 1]$  has been set, complete the remainder of the operations for the value of i using the first processing unit and simultaneously increasing the value of i by 1;

a second processing unit;

the non-transitory media readable by the second processing unit, the non-transitory media storing instructions that when executed by the second processing unit, cause the second processing unit to:

in response to the determination that the value of  $R[2 \cdot i - 1]$  has been set,

for each integer i whereby  $i < (n/2)$ , perform the following operations:

for each integer j with  $(i+1) \leq j < (n-i)$  do,

set  $(\text{MSD}, \text{LSD}) = A[i] \times A[j] + \text{MSD} + R[j+i]$ ; and

set  $R[j+i] = \text{LSD}$ ;

set count = i;

when (count < 2·i) do,

set (MSD, LSD) =  $A[j] \times a[\text{count}] + \text{MSD} + R[j+\text{count}]$ ;

set  $R[j+\text{count}] = \text{LSD}$ ;

increase count by 1;

set (MSD,LSD) =  $A[j] \times A[\text{count}] + \text{MSD} + R[j+\text{count}]$ ;

set  $R[j+\text{count}] = \text{LSD}$ ;

increase j by 1; and

return register R with set values.

11. The system according to claim 7 further comprising:

instructions for directing the first processing unit to:

apply a Montgomery modular reduction for a modulus value of m to the square of the message A.

12. The system according to claim 7 further comprising:

instructions for directing the first processing unit to:

apply a Barrett modular reduction for a modulus value of m to the square of the message A.