

HOMOMORPHIC ENCRYPTION METHOD AND ASSOCIATED DEVICES AND SYSTEM

TECHNICAL FIELD

[001] The disclosed technology concerns methods, devices and systems for
5 homomorphic encryption.

BACKGROUND

[002] Homomorphic encryption, which allows one to perform calculations or data processing on encrypted data, without decrypting it first, has attracted a lot of attention recently.

10 [003] Indeed, digital data processing of personal data has become ubiquitous in our everyday lives. Confidentiality and privacy protection of such data has thus become critical, as such personal data tends to circulate more and more in our digital environment.

[004] In this context, homomorphic encryption is a very promising solution, as it enables
15 to process data while preserving very securely data anonymization and privacy at the same time, as the data is not decrypted during its processing.

[005] Homomorphic encryption is usually based on the "learning with error" encryption scheme, in which the encrypted message $c=(a,b)$ is derived from the unencrypted message m according to the following formula: $b=m+e+a.s$, where:

- 20
- s is a secret key,
 - a is randomly chosen vector, for projecting the secret key s , and
 - e is a random noise component, added to $m+a.s$.

[006] To decrypt the message, someone possessing the secret key s may compute the quantity $b-a.s$ (equal to $m+e$), and then round the result to remove the noise component
25 e and retrieve the message m . Of course, the noise term e has to be, and to remain small enough, if one wants to be able to retrieve the message m .

[007] When two encrypted messages are summed up, or even worse, when they are homomorphically multiplied together, one obtains an encrypted message, which is an encrypted version of the sum, or product of the two initial, unencrypted messages,
30 whose noise component is higher than for the two initial encrypted messages.

[008] So, to prevent the noise term from increasing and increasing in the course of data processing, a refreshing procedure, usually named “bootstrapping”, is executed repeatedly, usually after each operation on the encrypted message c . This procedure produces a refreshed version of c , that is an encrypted message c' which is decrypted
5 as m (when decrypted using s), and whose noise component is smaller than the one of c .

[009] The bootstrapping procedure usually comprises the homomorphic (that is: in encrypted form) computation of $X^q.V(X)$ where:

- $V(X)$ is the following polynomial: $1+X+X^2+X^3+\dots+X^{N-1}$,
- 10 - and q is equal to the integer number that is the nearest to $2N(b-a.s)$:

$$q = \lfloor 2N(b - a.s) \rfloor$$

[010] The constant term of $X^q.V(X) \bmod (X^N+1)$, that is $\text{coef}_0(X^q.V(X))$, is a refreshed version of c : it is an encrypted version of m , but with a noise component e' smaller than the noise component e of c .

15 [011] One may note that the initial noise e is somehow washed out when (homomorphically) computing q , thanks to the rounding operation $\lfloor \cdot \rfloor$. In this operation, the factor $2N$ plays the role of an expansion factor, that enables to spread the possible values of $b-a.s$ (impaired by some noise) sufficiently, before rounding.

[012] Such a bootstrapping procedure is very useful and ingenious, but also very time-
20 consuming (in particular because the operations involved have to be carried on homomorphically), all the more that it has to be executed repeatedly.

[013] The homomorphic encryption scheme succinctly described above, which combines “learning with error” and a bootstrapping procedure, is usually carried on with binary messages (that is, messages whose values are either 0 or 1). Indeed, in this
25 technical field, most of the tools have been specifically developed and optimized for such messages.

[014] To perform homomorphic operations on a longer message, the message is decomposed into binary messages (classical binary decomposition), each binary message is encrypted separately, and the operation is then carried on,
30 homomorphically, bit by bit. But in this process, one has to take into account the

(encrypted) carry resulting from each binary addition, and to propagate this carry to the next binary operation. The process is thus a serial one, which is time consuming. Besides, each binary operation is carried homomorphically, which multiplies the number of homomorphic (and thus time-consuming) operations.

5 SUMMARY

[015] In this context, a homomorphic encryption technology, that enables to process efficiently messages longer than binary ones, is disclosed.

[016] An aspect of this technology concerns a method for determining an encrypted message c_i , which is an encrypted version of a message m_i , c_i being equal to (a_i, b_i) where b_i is determined from m_i and from a secret key s , b_i being equal to $m_i + e_i + a_i \cdot s$ with a_i being a randomly selected vector for projecting s and e_i being a random noise component added to $m_i + a_i \cdot s$,

wherein the message m_i belongs to a discrete set consisting of p_i distinct elements, with p_i being an odd number and wherein the elements of said discrete set are such that twice the difference between any two of these elements is not an integer number, said discrete set being the discrete torus $T_{p_i} = \frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or the set $T_{p_i, N}[X]$ of polynomials of degree $N-1$ whose coefficients belong to T_{p_i} , or another discrete set in bijective relationship with T_{p_i} .

[017] The inventors have developed a homomorphic cryptographic technique for such “long” messages (longer than binary messages), enabling, among other things, adding, multiplying and refreshing (i.e.: bootstrapping) these messages while they remain encrypted. These operations are carried on without decomposing the message into binary, independently encrypted, smaller messages. In other words, the encrypted message is treated as a whole, with no decomposition onto smaller fields.

[018] So, the burden of dealing with encrypted carries and serial processing of the numerous and encrypted bits is avoided.

[019] It may be noted that the fact that a homomorphic processing is possible for long messages (without decomposing it), and the fact that this processing can be fast and efficient is far from being an immediate, straightforward result.

[020] Indeed, as mentioned above, most of the homomorphic cryptography tools currently available are adapted specifically to binary messages. So, the inventors had to go against the usual practice and prejudices and had to move forward in the development of specific (and elaborated) tools intended for longer messages, without
 5 knowing whether a direct homomorphic processing of long messages was actually possible or not, or whether it could be executed efficiently or not. In particular, the inventors had to elaborate a new homomorphic multiplication scheme and a new bootstrapping method (presented below), to develop such an encryption protocol. Only once these quite complex tools had been developed could the inventors be sure that
 10 such a treatment could actually be carried out.

[021] And, in the process of developing these new tools, the inventors realized that working with messages belonging to the discrete Torus T_{p_i} , or to the set of polynomials $T_{p_i, N}[X]$, or to another equivalent discrete set, *with:*

- p_i being an odd number,

15 - *and such that for any couple grouping two of the elements of said set, twice the difference between these two elements is not an integer number*

was very fruitful, as it turns out that these specific features make different operations, involved in the multiplication and bootstrapping process mentioned above, feasible (as will be described in more details later).

20 [022] The fact that a direct homomorphic processing of a "long" message is not only possible, but that it can also be achieved *efficiently*, in a limited time, is also far from obvious, at first glance.

[023] Indeed, when the message m_i is long, that is when the integer number p_i is quite large, for example equal to 17, 21, or more, one would expect at first sight to have to
 25 use polynomials whose degree $N-1$ is high, during bootstrapping operations. Indeed, as explained above, in the calculation of the quantity q , the integer N acts as a kind of expansion coefficient. So, when p_i is large, as the possible values of the message m_i (distributed in the interval $]-1/2; 1/2[$) are then close to each other, one expects to have to choose a number N that is large, in order to be able to spread these values sufficiently

apart, before the rounding operation involved in the calculation of the quantity q . And a large N considerably increases the calculation time.

[024] More precisely, a rough estimate leads to the conclusion that N would have to be of the order of 10000 (with $p_i=20$ and $n=500$, for instance), or greater, for the bootstrapping operation to work properly (that is with a low probability of losing or degrading the message m during this operation). And such values of N are almost prohibitive, in terms of computation time.

[025] But in fact, a complete, acute probability calculation shows that such a bootstrapping operation can be successfully performed with polynomials whose degree $N-1$ is much smaller than what the first approximate estimate mentioned above suggests. More precisely, it can be shown that an integer number N of the order of 1000 (e.g. 1024) turns out to be sufficient for typical values of p_i (about 20-40, for example).

[026] According to the disclosed technology, N may be selected as small as 50 times p_i , or below, or even below 30 times p_i , to take advantage of this surprisingly favorable scaling and accelerate the bootstrapping processing.

[027] The method for bootstrapping developed by the inventors, mentioned above, is a method for bootstrapping an encrypted message $c_i=(a_i, b_i)$ determined as described above, this method comprising a determination of a refreshed encrypted message $c'_i=(a'_i, b'_i)$, which is an encrypted version of a value $g(m_i)$ of a function g applied to the message m_i , c'_i having a noise component e'_i smaller than e_i .

[028] The determination of c'_i comprises a step of determining homomorphically the constant coefficient $\text{coef}_0(X^q.W_g(X))$ of $X^q.W_g(X) \bmod (X^{N+1})$,

where $W_g(X) = \sum_{j=0}^{N-1} w_j.X^j$ is a polynomial of degree $N-1$, whose coefficients w_j are given by the following formula:

$$(-1)^{\lfloor 2Nm_i \rfloor / N!} . w_j = g(m_i), \text{ where } j = N[\lfloor 2Nm_i \rfloor / N] - \lfloor 2Nm_i \rfloor$$

with $\lfloor 2Nm_i \rfloor$ being the integer number that is the nearest to $2Nm_i$ and with $\lceil \lfloor 2Nm_i \rfloor / N \rceil$ being the ceiling function applied to $\lfloor 2Nm_i \rfloor / N$,

and where q is the integer number that is the nearest to $2N(b_i - a_i.s)$: $q = \lfloor 2N(b_i - a_i.s) \rfloor$.

[029] So, during this bootstrapping operation, in addition to reducing (refreshing) the noise component of a cipher text, an arbitrary function g can be applied to the corresponding message m_i .

[030] This accelerates considerably numerous data processing routines. Indeed, to
 5 apply such a function g to a message, instead of: carrying on an elementary operation (homomorphically), then bootstrapping the result (to counter the noise increase due to the preceding operation), carrying on another elementary operation, then bootstrapping the result, and so on, one just has to execute one time the special bootstrapping procedure presented above.

10 [031] In this bootstrapping procedure, the degree $N-1$ of the polynomial $Wg(X)$ may be selected so as to be above p_i , or even above 5 times p_i .

[032] The disclosed technology concerns also a method for homomorphically multiplying two messages m_{1i} and m_{2i} , each belonging to the discrete torus T_{p_i} or to the set of polynomials $T_{p_i, N}[X]$, the method comprising a determination of an encrypted
 15 message c_{3i} which is an encrypted version of the product of m_{1i} by m_{2i} , c_{3i} being determined from encrypted versions c_{1i} and c_{2i} of the messages m_{1i} and m_{2i} , without decrypting c_{1i} or c_{2i} , the method comprising homomorphically determining

$$\frac{(p_i + 1)}{2} \cdot \left[\frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} + m_{2i})^2 - \frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} - m_{2i})^2 \right]$$

20 Where $p_i \cdot (m_{1i} + m_{2i})^2$ and $p_i \cdot (m_{1i} - m_{2i})^2$, or $\frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} + m_{2i})^2$ and $\frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} - m_{2i})^2$, or $\left[\frac{(p_i + 1)}{2} \right]^2 \cdot p_i \cdot (m_{1i} + m_{2i})^2$ and $\left[\frac{(p_i + 1)}{2} \right]^2 \cdot p_i \cdot (m_{1i} - m_{2i})^2$ are determined homomorphically by executing the method for bootstrapping presented above, the function g being the function
 $g_{sq'} : m_i \rightarrow p_i \cdot (m_i)^2$, or $g_{sq} : m_i \rightarrow \frac{(p_i + 1)}{2} \cdot p_i \cdot (m_i)^2$, or, respectively, $g_{sq''} : m_i \rightarrow$
 25 $\left[\frac{(p_i + 1)}{2} \right]^2 \cdot p_i \cdot (m_i)^2$.

[033] It may be noted that the tools presented above, for homomorphic processing of messages m_i belonging to such a discrete set of p_i elements (with p_i odd), can, in

addition to their own benefits, be applied in an extremely beneficial manner to a cryptographic technic for even longer messages, decomposed into smaller components m_i , based on the Chinese remainder theorem decomposition, which, again, enables to avoid the burden of dealing with carries (so, in other words, the tools mentioned above are some kinds of basic bricks of this Chinese-remainder processing method, presented in more detail below).

[034] The disclosed technology concerns also a method for determining an encrypted message $c_{i,i}$, which is an encrypted version of a value $f(m1_i, m2_i)$ of a two-variable function f applied to messages $m1_i$ and $m2_i$, $m1_i$ and $m2_i$ belonging each to the discrete set mentioned above (for instance the torus T_{pi} or the set of polynomials $T_{pi,N}[X]$),

$c_{i,i}$ being determined from encrypted messages $c1_i$ and $c2_i$ without decrypting $c1_i$ or $c2_i$, $c1_i$ and $c2_i$ being encrypted versions of $m1_i$ and $m2_i$ respectively, that have been determined according to the method for encrypting presented above,

the determination of $c_{i,i}$ comprising:

- 15 - determining $c1'_i$ and $c2'_i$, from $c1_i$ and $c2_i$ respectively, by executing the method for bootstrapping presented above, the function g employed during this bootstrapping being a function g_{map} that maps said discrete set onto a set $\underline{S}$ whose elements $\underline{u}_j$ belong each to the torus $T =]-1/2, 1/2]$ and are such that, for any set of indexes (i, j, k, l) with $(i, j) \neq (k, l)$, (k, l) being different from (j, i) : $\alpha \cdot \underline{u}_i + \beta \cdot \underline{u}_j$ is different from $\alpha \cdot \underline{u}_k + \beta \cdot \underline{u}_l$ modulo $1/2$, with α and β being two constant and integer, given numbers,

$c1'_i$ and $c2'_i$ being the encrypted versions of $\underline{m1}_i = g_{map}(m1_i)$ and $\underline{m2}_i = g_{map}(m2_i)$ respectively, where $\underline{m1}_i$ and $\underline{m2}_i$ belong each to $\underline{S}$,

- determining homomorphically, from $c1'_i$ and $c2'_i$, an encrypted message c_s which is an encrypted version of $\alpha \cdot \underline{m1}_i + \beta \cdot \underline{m2}_i$,
- 25 - determining $c_{i,i}$ from c_s by executing the method for bootstrapping presented above, the function g employed during this bootstrapping being a one-variable function g_f from T to T_{pi} , defined by the following condition:

for any couple of messages $m1_j$, $m2_j$ belonging each to said discrete set,

$$30 \quad g_f(\alpha \cdot \underline{m1}_j + \beta \cdot \underline{m2}_j) = f(m1_j, m2_j)$$

that is:

$$g_f[\alpha \cdot g_{\text{map}}(m1_j) + \beta \cdot g_{\text{map}}(m2_j)] = f(m1_j, m2_j) .$$

[035] According to an aspect of this technology, the elements of the set $\underline{S}$ are such that for any set of indexes (i,j,k,l) with $(i,j) \neq (k,l)$, $\alpha \cdot \underline{m}_i + \beta \cdot \underline{m}_j$ is different from $\alpha \cdot \underline{m}_k + \beta \cdot \underline{m}_l$ modulo $1/2$, even when $(k,l) = (j,i)$. In this case, one may have, in particular : $\alpha=1$ and $\beta=-1$, c_s being determined by computing the homomorphic difference $\ominus$ between $c1'_i$ and $c2'_i$: $c_s = c1'_i \ominus c2'_i$, or $\alpha=-1$ and $\beta=1$, c_s being determined by computing the homomorphic difference $\ominus$ between $c2'_i$ and $c1'_i$: $c_s = c2'_i \ominus c1'_i$.

10 [036] According to another aspect of this technology, the function f is symmetric, $f(m1_i, m2_i)$ being equal to $f(m2_i, m1_i)$ for any couple of messages $m1_i$ and $m2_i$, α and β are each equal to 1, and c_s is determined by computing the homomorphic sum $\oplus$ of $c1'_i$ and $c2'_i$: $c_s = c1'_i \oplus c2'_i$.

[037] The disclosed technology also concerns such a method for determining an encrypted message $c_{i,i}$, which is an encrypted version of a value $f(m1_i, m2_i)$ of a two-variable function f , as such, independently from the bootstrapping method in question. Indeed, as the skilled person will appreciate, the method for homomorphically evaluating f , based on the intermediate mapping of said discrete set (e.g.: $\mathbf{T}_{pi}$) onto $\underline{S}$, can be achieved independently from this specific bootstrapping procedure, using any method for homomorphically mapping said discrete set on $\underline{S}$ (the mapping method being possibly different from the one described above), and using then any method for homomorphically applying a function like the function g_f to the quantity $\alpha \cdot \underline{m1}_i + \beta \cdot \underline{m2}_i$.

[038] According to an aspect of the disclosed technology, any of the methods presented above is executed by a computer (programmed or otherwise arranged to execute the method in question), that is an electronic device or system (possibly distributed among several distant, remote devices) comprising at least one processor for executing logical operations, and one memory device for storing data.

[039] In particular, an aspect of the technology concerns an encrypting device, comprising one or more processors and at least one memory, the encrypting device being programmed to make the processor or processors executing the following steps:

- 5 - receiving data representative of an unencrypted message m_i , that belongs to the discrete set mentioned above,
- processing said data according to the method for encrypting presented above, to determine encrypted data representative of the encrypted message c_i , the secret key s employed during said processing, stored at least transitorily in the memory of the encrypting device, being accessed
10 by the processor or processors during said processing.

[040] An aspect of the disclosed technology concerns a cryptographic system comprising such an encrypting device and a processing device,

- the processing device comprising one or more processors and at least one memory, the processing device being programmed to make the processor or processors
15 executing the following steps:
- receiving data, representative of an encrypted message c_i produced by the encrypting device,
 - processing said data, according to the method for bootstrapping that has been presented above, to determine refreshed data, representative of the
20 encrypted message c'_i , said processing being achieved without decrypting c_i and without using, reading or accessing the secret key s .

[041] In particular, the secret key s may be completely absent from the processing device (which is a separate device distinct from the encrypting device), and remain absent from the processing device at all time.

- 25 [042] The disclosed technology also concerns such a processing device, as such, independently from the encrypting device.

[043] The processing device may also be programmed to execute the method for multiplying presented above, and the method for determining an encrypted message $c_{f,i}$, which is an encrypted version of a value $f(m1_i, m2_i)$ of the two-variable function f .

[044] The disclosed technology concerns also a computer-readable program product comprising instructions that, when executed by a computer, make the computer to execute any of the method presented above.

[045] The disclosed technology concerns also a method for determining an encrypted message C, which is an encrypted version of a message x belonging to $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$, p being equal to $\prod_{i=1}^r p_i$ where the integer numbers p_i are pairwise coprime, the method comprising:

- decomposing x into its components $x_i, i=1 \dots r$, with x_i equal to x modulo p_i ,
- for each component x_i whose associated factor p_i is odd, determining an encrypted version c_i of a message m_i , according to the method for encrypting that has been presented above m_i being, among the elements of said discrete set, the element that is associated to x_i by a given bijective relationship between $\mathbb{Z}_{p_i}$ and said discrete set,
- if a component x_i is associated to a factor p_i equal to 2, determining an encrypted version c_i of a quantity Q_i associated to x_i and that can takes two distinct values c_i being equal to (a_i, b_i) with $b_i = Q_i + e_i + a_i \cdot s$,
- returning $C = (c_1, \dots, c_i, \dots, c_r)$.

[046] An aspect of this technology concerns a method for bootstrapping the encrypted message $C = (c_1, \dots, c_i, \dots, c_r)$ determined according the encrypting method that has just been described, this (generalized) bootstrapping method comprising:

- for each c_i whose associated factor p_i is odd, determining a refreshed encrypted message c'_i , from c_i , by executing the method for bootstrapping that has been presented before,
- returning $C' = (c'_1, \dots, c'_i, \dots, c'_r)$.

[047] This technology concerns also a method for determining an encrypted message $C_3 = (c_{31}, \dots, c_{3i}, \dots, c_{3r})$, which is an encrypted version of the product of two messages x_1 and x_2 , each belonging to $\mathbb{Z}_p$,

C_3 being determined from encrypted messages C_1 and C_2 , without decrypting C_1 or C_2 , C_1 and C_2 being encrypted versions of x_1 and x_2 respectively, $C_1 = (c_{11}, \dots, c_{1i}, \dots, c_{1r})$ and $C_2 = (c_{21}, \dots, c_{2i}, \dots, c_{2r})$ having been determined from x_1 and x_2

respectively, according to the method for encrypting that has just been presented, wherein said discrete set is T_{p_i} or $T_{p_i, N}[X]$, and

wherein each component c_{3i} of C_3 is determined from the corresponding components c_{1i} and c_{2i} of C_1 and C_2 , according to the method for multiplying that was
5 described above for messages belonging to the discrete torus T_{p_i} , when c_{1i} and c_{2i} are associated to a factor p_i that is odd.

[048] The disclosed technology concerns such a method for bootstrapping a message C , or such a method for multiplying homomorphically two messages x_1 and x_2 , in itself, independently from the bootstrapping method (for messages belonging to T_{p_i}) that has
10 been presented above. Indeed, as the skilled person will appreciate, these two methods, for processing encrypted messages, associated to unencrypted messages x belonging to $\mathbf{Z}_p$ (with p being equal to $\prod_{i=1}^r p_i$ where the integer numbers p_i are pairwise coprime), take advantage of the Chinese remainder theorem to process the encrypted “sub-messages” c_i , $i=1..r$, independently from each other, without having to deal with, or
15 anyhow considering (encrypted) carries. This technique could clearly be applied using other “sub-routines” than the one presented above, for the homomorphic processing of the sub-messages m_i . In particular, any multiplying or bootstrapping method for elements belonging to the discrete set mentioned above (e.g.: the discrete torus T_{p_i}) could be used, instead of the one described above.

20 [049] According to the disclosed technology, these encrypting and processing techniques (based on a “Chinese remainder” decomposition), could be executed by a computer.

[050] The disclosed technology concerns also an electronic device, comprising at least one processor and one memory, programmed or otherwise configured to execute one of
25 these methods. The disclosed technology concerns also a computer-readable program product comprising instructions, whose execution by a computer make the computer to execute anyone of these methods.

BRIEF DESCRIPTION OF THE DRAWINGS

[051] Other characteristics and benefits of the disclosed technology will become clear from the description which is given below, by way of example and non-restrictively, in reference to the figure, in which:

[052] Figure 1 represents, as a block-diagram, operations carried on to multiply
5 homomorphically two messages.

[053] Figure 2 represents, as a block-diagram, operations carried on to evaluate homomorphically a two-variable function.

[054] Figure 3 represents, as a block-diagram, an encryption step of a long message x , based on a Chinese remainder decomposition.

10 [055] Figure 4 represents, as a block-diagram, operations carried on to add homomorphically two encrypted messages obtained previously according to the encryption step of figure 3.

[056] Figure 5 represents, as a block-diagram, operations carried on to multiply homomorphically two encrypted messages, obtained previously according to the
15 encryption step of figure 3, said multiplication comprising several sub-multiplications carried on according to the process represented in figure 1.

[057] Figure 6 schematically illustrate a cryptographic system according to the disclosed technology.

DETAILED DESCRIPTION

20 [058] As mentioned above, the disclosed technology concerns an homomorphic encryption technique that enables to carry on operations on encrypted messages without decrypting them, these encrypted messages being associated to unencrypted messages that are each longer than a single bit, and that are processed without being decomposed into individual binary messages.

25 [059] The disclosed technology concerns more particularly a technique for encrypting and processing messages m_i , that belong to a discrete set consisting of p_i distinct elements, with p_i being an odd number, these messages being encrypted and processed directly, without decomposing these messages into smaller (for instance binary) messages that would be encrypted separately. This technology is presented below in

the case where the discrete set in question is the discrete torus $\mathbf{T}_{pi} = \frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or the set $\mathbf{T}_{pi,N}[X]$ of polynomials of degree N-1 whose coefficients belong to $\mathbf{T}_{pi}$. Still, as the skilled person will appreciate, this technology can be applied as well (possibly, with minor modifications for the homomorphic addition or multiplication) to another discrete set of messages in bijective relationship with $\mathbf{T}_{pi}$, like $\mathbb{Z}_{p_i} = \mathbb{Z}/p_i\mathbb{Z}$ or $\frac{1}{p_i} \left\{ -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2}, \frac{p_i-1}{2} + 1 \right\}$ (that is $\mathbf{T}_{pi}$ with an offset of $+1/p_i$; it may be noted that for this alternative example, the different operations described below – including the bootstrapping procedure, but also the homomorphic multiplication – can be implemented in the same manner as in the case of $\mathbf{T}_{pi}$, even if the values may have to be shifted to reintegrate them in the interval $[-1/2, 1/2[$) for instance.

[060] The disclosed technology concerns also a technique for encrypting and processing even longer messages x , using a specific decomposition of the message x into smaller messages m_i , this decomposition being based on the Chinese remainder theorem decomposition and enabling to process the sub-messages m_i separately from each other, without having to take into account possible carries.

[061] The encryption technique for encrypting and processing a message m_i , without decomposing it, will be presented first. The technique for encrypting and processing a longer message x , composed of several such messages m_i , based on a Chinese remainder theorem decomposition, will be presented then. Finally, electronic devices and systems for implementing such methods will be presented.

[062] Homomorphic encryption and processing of a message m_i belonging to $\mathbf{T}_{pi}$ or to $\mathbf{T}_{pi,N}[X]$.

[063] As mentioned above, this homomorphic encryption technique is applied to messages m_i that belong to the discrete torus $\mathbf{T}_{pi} = \frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or to the set $\mathbf{T}_{pi,N}[X]$ of polynomials of degree N-1 whose coefficients belong to $\mathbf{T}_{pi}$.

[064] More precisely, the set of polynomials $T_{pi,N}[X]$ is the finite field composed of the polynomials of degree $N-1$ defined modulo X^N+1 , and whose coefficients belong to T_{pi} : $T_{pi,N}[X]=T_{pi}[X]/(X^N+1)$.

[065] Remarkably, the integer number p_i is odd (which makes some of the operations
5 described below possible).

[066] Encryption

[067] An unencrypted message m_i , belonging to T_{pi} , is encrypted according to the "learning with error" scheme: the encrypted message $c_i=(a_i,b_i)$ (or, in other words, the cypher text c_i), which is the encrypted version of m_i , is determined by:

- 10 - randomly choosing a vector a_i , and
 - computing the quantity $b_i=m_i+e_i+a_i.s$ where s is a secret key and where e_i is a noise component, added to $m_i+a_i.s$.

[068] When m_i belongs to T_{pi} , the noise component e_i belongs to the torus $T=[-1/2,1/2]$. And when m_i belongs to $T_{pi,N}[X]$, e_i belongs to the set $T_N[X]$ of polynomials of degree $N-1$, defined modulo (X^N+1) and whose coefficients belong each to T . e_i is randomly
15 chosen according to a predefined statistical distribution. At least one feature of this distribution, for instance its standard deviation (or the standard deviation of each of its coefficients, when e_i is a polynomial), is determined depending on the integer number p_i . For instance, the standard deviation of this distribution is all the smaller than p_i is high
20 (this standard deviation may be inversely proportional to p_i , for example).

[069] The secret key s is a vector having n components, each belonging to the set of integers Z , or to the set of polynomials of degree $N-1$, defined modulo (X^N+1) and whose coefficients belong each to Z . In other words, s belongs to Z^n , or to $Z_N[X]^n$, where $Z_N[X]=Z[X]/(X^N+1)$. The secret key s may more particularly belong to B^n , or to $B_N[X]^n$, where
25 $B=\{0,1\}$ (B being the set constituted by the binary values 0 and 1).

[070] The vector a_i has also n components, and is such that the scalar product $a_i.s$ belongs either to T_{pi} (when m_i belongs to T_{pi}), or to $T_{pi,N}[X]$ (when m_i belongs to $T_{pi,N}[X]$). For instance, if m_i belongs of T_{pi} , and if s belongs to B^n , a_i belongs to T_{pi}^n (each of the n components of a_i belongs to T_{pi}).

[071] In the following, for the sake of clarity, we consider that m_i belongs to T_{pi} , and that s belongs to $\mathbf{Z}^n$ or $\mathbf{B}^n$.

[072] Besides, in the following, we consider more specifically that s belongs to $\mathbf{B}^n$ (which simplifies some of the operations involved in the bootstrapping procedure
5 described below).

[073] To decrypt the encrypted message $c_i=(a_i,b_i)$, a decryption module or a decryption device that has access to the secret key s computes $b_i-a_i.s$ (which is equal to m_i+e_i), and then rounds it to the nearest value belonging to the discrete Torus T_{pi} , thus removing the noise component e_i and obtaining m_i .

10 [074] The encrypted message c_i , obtained by encrypted m_i using the secret key s , may be also noted $\tilde{\varphi}_s(m_i)$.

[075] Homomorphic operations

[076] An objective of the instant technology is to carry on operations homomorphically, that is to say to carry on operations on one, two or more messages $m1_i, m2_i, \dots$, but
15 using their encrypted version $c1_i, c2_i, \dots$ (instead of the unencrypted messages $m1_i, m2_i, \dots$) and without decrypting them, without knowing or having access to the secret key s .

[077] More generally, determining homomorphically a given quantity is meant determining an encrypted version of this quantity, on the basis of encrypted messages,
20 without using or determining unencrypted versions (unencrypted counterparts) of these encrypted messages, and without using (in particular without accessing) the secret key s .

[078] In practice, it is particularly interesting to be able to achieve two-operands homomorphic operations, like a homomorphic addition, multiplication or comparison of
25 two messages (which is very useful for sorting encrypted data without decrypting them).

[079] In the case of a two-operands operation, the usual ("normal") operation, whose operands are $m1_i$ and $m2_i$, is noted Op and its result is noted $m3_i$: $m3_i=Op(m1_i,m2_i)$. The corresponding homomorphic operation (that is, the homomorphic counterpart of Op) is noted Hop . So, the homomorphic operation Hop is an operation:

- whose operands are two cipher texts c_{1i} and c_{2i} , which are encrypted versions of two messages m_{1i} and m_{2i} respectively (each belonging to T_{pi});
- which produces a cipher text $c_{3i} = \text{Hop}(c_{1i}, c_{2i})$, that is determined from c_{1i} and c_{2i} without decrypting c_{1i} or c_{2i} , without using the secret key s (in particular, without
5 knowing or accessing the secret key s);
- the cipher text c_{3i} being an encrypted version of $m_{3i} = \text{Op}(m_{1i}, m_{2i})$.

[080] In other words, when decrypting c_{1i} , c_{2i} and c_{3i} using the secret key s , one gets respectively m_{1i} , m_{2i} and $m_{3i} = \text{Op}(m_{1i}, m_{2i})$.

Homomorphic addition

- 10 [081] The homomorphic addition is noted $\oplus$, ($\text{Hop} = \oplus$). The homomorphic addition of two cipher texts, $c_{3i} = c_{1i} \oplus c_{2i}$, is readily achieved by summing the two cipher texts $c_{1i} = (a_{1i}, b_{1i})$ and $c_{2i} = (a_{2i}, b_{2i})$ component by component: $c_{3i} = (a_{3i}, b_{3i})$ with $a_{3i} = a_{1i} + a_{2i}$ and $b_{3i} = b_{1i} + b_{2i}$.

- [082] Indeed, $b_{3i} - s \cdot a_{3i} = m_{1i} + m_{2i} + e_{1i} + e_{2i}$ is rounded as $m_{1i} + m_{2i}$ (provided that $e_{1i} + e_{2i}$
15 is small enough). So, $c_{3i} = c_{1i} \oplus c_{2i}$ is indeed decrypted as $m_{3i} = m_{1i} + m_{2i}$.

[083] A homomorphic subtraction $\ominus$ can be achieved similarly by subtracting two cipher texts component by component.

- [084] A homomorphic multiplication for messages belonging to the discrete torus is much more complicated to develop and to implement. It will be presented later, after
20 having described a new bootstrapping technique for messages belonging to the discrete torus (as this bootstrapping technique is used to carry on the homomorphic multiplication in question).

[085] Method for bootstrapping an encrypted message c_i

- [086] A method for bootstrapping an encrypted message $c_i = (a_i, b_i)$, that had been
25 determined, as described above, by encrypting a message m_i using the secret key s , is presented below.

[087] This method comprises a homomorphic determination of a refreshed encrypted message $c'_i = (a'_i, b'_i)$, which is an encrypted version of a value $g(m_i)$ of a function g

applied to the message m_i , c_i' having a noise component e_i' smaller than the noise component e_i of c_i . The result c_i' of this bootstrapping method, applied to c_i , is noted $G(c_i)$.

[088] So, this new bootstrapping method is a bootstrapping method adapted to 5 messages m_i that belong to the discrete torus T_{pi} . It enables to refresh encrypted messages c_i associated to such messages m_i , that is to say to reduce the noise components e_i of these cypher texts c_i . And it enables also, at the same time, to apply an arbitrary function g to the message m_i . In other words, this method enables to homomorphically determine the value $g(m_i)$ of the function g , applied to m_i .

10 [089] The function g can be any function from T_{pi} to T_{pi} . In other words, g maps elements of T_{pi} to elements of T_{pi} (or elements of $T_{pi,N}[X]$ to elements of $T_{pi,N}[X]$).

[090] The determination of c_i' comprises a step of determining homomorphically the constant coefficient $\text{coef}_0(X^q.W_g(X))$ of a polynomial $X^q.W_g(X) \bmod (X^{N+1})$, where q is the integer number that is the nearest to $2N(b_i - a_i.s)$: $q = \lfloor 2N(b_i - a_i.s) \rfloor$.

15 [091] The homomorphic determination of $\text{coef}_0(X^q.W_g(X))$ is carried on in a way similar to prior art techniques, like described in section 6 of Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène, "TFHE: Fast fully homomorphic encryption over the torus", Journal of Cryptology, 33(1):34–91, 2020, for instance, *but using a specially designed polynomial $W_g(X)$* , associated to the function g considered.

20 [092] The coefficients of the polynomial $W_g(X)$, of degree $N-1$, are noted w_j :

$$W_g(X) = \sum_{j=0}^{N-1} w_j \cdot X^j$$

[093] They belong each to T_{pi} and are determined as follow.

[094] Let's consider a function f_w , from $\mathbf{Z}$ to T_{pi} , defined as:

$$f_w: z \rightarrow f_w(z) = (-1)^{\lfloor z/N \rfloor} \cdot w_{j'}, \text{ where } j' = N \lfloor z/N \rfloor - z$$

25 [.] denoting the ceiling function.

[095] The coefficients w_j of $W_g(X)$ are then defined, depending on g , such that, for any m_i belonging to T_{pi} :

$$f_w([2Nm_i]) = g(m_i)$$

[096] So, the coefficients w_j are related to the function g by the following equation:

$$(-1)^{[2Nm_i]/N} \cdot w_{j'} = g(m_i), \text{ where } j' = N[2Nm_i/N] - [2Nm_i]$$

[097] It can be proven that, given that p_i is odd, and that twice the difference between
5 any two of the elements of the discrete set considered (namely T_{p_i} , here) is not an integer number:

- a) it is possible, for any function g from T_{p_i} to T (and more generally for any function from any discrete set of p_i elements such as defined above, to T), to construct a corresponding function f_w , and thus a polynomial $W_g(X)$, that fulfills
10 the conditions given above, and that
- b) thanks to this particular choice of polynomial (i.e.: thanks to the use of the particular polynomial $W_g(X)$), homomorphically determining $\text{coef}_0(X^q \cdot W_g(X))$ actually leads to the determination of an encrypted version of $g(m_i)$.

[098] The detailed (and quite elaborate) proof of these results is not presented here, for
15 the sake of conciseness and as it not directly useful in view of implementing the instant technology.

[099] Still, it is noted that point a) is far from being immediate. Indeed, it can be readily noted that, for any z belonging to Z ,

$$f_w(z + N) = -f_w(z) = f_w(z - N)$$

20 [0100] So the function f_w defined above is quite constrained, and a detailed analysis is in fact required to ensure that an adequate f_w function can actually be constructed for *any* function g (provided that the conditions above, regarding p_i and the elements of the set, are fulfilled).

[0101] It may be noted that this bootstrapping technique, presented above in the case
25 where the discrete set with p_i elements is T_{p_i} (or $T_{p_i, N}[X]$), can be applied as well in the case of another discrete set consisting of p_i elements (with p_i odd, and such that twice the difference between any two of the elements of this set is not an integer number). In

particular, the formula giving the values of the coefficients w_j of the polynomial W_g remains the same.

[0102] As mentioned in the section entitled “summary”, the degree $N-1$ of the polynomial $W_g(X)$ can be chosen so that N is below 50 times p_i , or even below 30 times p_i , which is
 5 high enough to avoid enables erasing or degrading the message m_i , while leading to a reasonably fast execution of this bootstrapping procedure.

[0103] Besides, N may be chosen as equal to or higher p_i , or even than 5 times p_i . The coefficients w_j of $W_g(X)$ are then numerous enough to adjust the polynomial $W_g(X)$ to any, arbitrary function g , from T_{p_i} to T_{p_i} .

10 [0104] Some details regarding the homomorphic evaluation of $\text{coef}_0(X^q.W_g(X))$ in itself are presented now. They are outlined briefly, as this evaluation is carried on in a way similar to prior art techniques (such as the one described in ***Ref1), apart from the specific choice of the polynomial $W_g(X)$.

[0105] As known in the art, a polynomial such as $X^q.W_g(X)$ modulo (X^N+1) can be
 15 decomposed into several successive modular products :

$$X^q.W_g(X) = X^{-\bar{a}_n s_n} \cdot \left(\dots \left(X^{-\bar{a}_1 s_1} \cdot (X^{-\bar{b}}.W_g(X)) \right) \dots \right)$$

where $\bar{b} - \sum_{k=1}^n \bar{a}_k s_k \approx [2N(b_i - a_i.s)] = q$.

[0106] In the equation above, each product “.” is a modular product: the left term, for instance $X^{-\bar{b}}$, or $X^{-\bar{a}_1 s_1}$, belongs to $\mathbf{Z}_N[X]$, while the right term, for instance $W_g(X)$, or
 20 $(X^{-\bar{b}}.W_g(X))$, belongs to $T_{p_i, N}[X]$. Such a product is completely similar to the product $y.m_i$ of an integer number y with a number m_i belonging to T_{p_i} , which is usually designated as a modular product (due to the modular structure of T_{p_i} , defined modulo 1).

[0107] It is known that a modular product $y.m_i$, or $X^i.W_g(X)$, can be computed homomorphically in the form of a so-called external product $\chi_{s'}(y) \boxtimes \tilde{\varphi}_s(m_i)$ where
 25 $\tilde{\varphi}_s(m_i) = c_i$ is the encrypted version of the message m_i (c_i being computed as explained above), and where $\chi_{s'}(z)$ is the so-called TGSW encrypted form of z (TGSW standing for “Gentry, Sahai, Waters” Torus encryption, the TGSW scheme having been suggested in the article “Homomorphic encryption from learning with errors:

Conceptually-simpler, asymptotically-faster, attribute-based”, by C. Gentry, A. Sahai, and B. Waters, InCrypto’13, 2013).

[0108] Besides, as known in the art, a quantity like $\chi_{s'}(X^{-\bar{a}_k s_k}) \boxtimes \tilde{\varphi}_s(m)$ can be computed without knowing the secret key component s_k , $k=1\dots n$ (more generally, without
5 knowing the secret key s).

[0109] Indeed, as the secret key components s_k , $k=1\dots n$ belong each to $\mathbf{B}=\{0,1\}$, the quantity $X^{-\bar{a}_k s_k}$ is equal to $(X^{-\bar{a}_k} - 1)s_k + 1$ which is encrypted as $(X^{-\bar{a}_k} - 1) \cdot \chi_{s'}(s_k) + H_B^l$.

[0110] And so, an encrypted version of $X^q.W_g(X)$ modulo (X^{N+1}) can be computed
10 without knowing or having access to the secret key s as it can be computed using an encrypted versions $\chi_{s'}(s_k)$ of the secret key components s_k , sometimes designated as a Bootstrap Key BK_k in the literature. Usually, the Bootstrap Key $BK_k = \chi_{s'}(s_k)$, which is an encrypted version of the secret key component s_k , encrypted using another key s' , is a public key.

15 [0111] The way to compute the TGSW encrypted version $\chi_{s'}(y)$ of an integer number y or, equivalently, to compute the TRGSW (GSW Torus ‘Ring’) encrypted version $\chi_{s'}(P)$ of a polynomial P belonging to $\mathbf{Z}_N[X]$, is recalled below. The way to compute the external product $\boxtimes$ is also recalled.

[0112] $\chi_{s'}(y)$ is defined as being equal to $Z + yH_B^l$ with:

20 - $Z = \begin{pmatrix} z_1 \\ \vdots \\ z_{l(n+1)} \end{pmatrix}$, where the z_j , $j=1\dots l(n+1)$ are $l(n+1)$ distinct encrypted versions

of 0, the null value 0 being encrypted with s' : $z_j = \tilde{\varphi}_{s'}(0)$, and where l is a given, fixed integer number; and

- H_B^l a matrix whose dimension is $(n+1) \times (n+1) \times l$ and whose coefficients belong to the Torus $\mathbf{T}$, H_B^l being equal to:

25 $H_B^l = \begin{pmatrix} h & 0 & \dots \\ 0 & h & \ddots \\ \vdots & \ddots & \ddots \end{pmatrix}$ with $h = \begin{pmatrix} 1/B \\ \vdots \\ 1/B^l \end{pmatrix}$, B being an integer number higher than or equal to 2.

[0113] The external product $C \boxtimes c$, between:

- an encrypted message $c = \tilde{\varphi}_{s'}(m)$, $c \in \mathbf{T}^{n+1}$, corresponding to a message m belonging to the discrete torus $\mathbf{T}_{\text{pi}}$, more generally to the torus $\mathbf{T}$, and
- an encrypted version $C = \chi_{s'}(y)$ of an integer number y , $C \in \mathcal{M}_{(n+1)l, n+1}(\mathbf{T})$,

5 is computed as explained below.

[0114] The $n+1$ components c_j , $j=1..n+1$ of the encrypted message $c = \tilde{\varphi}_{s'}(m)$ are each decomposed, using a kind of numeral system of base B (like the one used to compute $\chi_{s'}$):

$$c_j \approx \sum_{u=1}^l \frac{c_j^u}{B^u} \bmod(1), \quad 0 \leq |c_j^u| \leq B-1$$

10 [0115] Then, one computes

$$C \boxtimes c = \sum_{B,l} (c) C$$

where $\sum_{B,l}(c) C$ denotes the usual row-vector matrix product and where $\sum_{B,l}(c)$ is a matrix of dimension $(n+1) \times l$ whose expression is:

$$\sum_{B,l}(c) = (\sigma_{B,l}(c_1), \dots, \sigma_{B,l}(c_j), \dots, \sigma_{B,l}(c_{n+1}))$$

15 with $\sigma_{B,l}(c_j) = (c_j^1, \dots, c_j^u, \dots, c_j^l)^T$.

[0116] The result, $C \boxtimes c$, which belongs to $\mathbf{T}^{n+1}$, is an encrypted version of $y.m$. In other words, decrypting this quantity using s gives $y.m$.

[0117] This technique can be applied to any message m belonging to the torus $\mathbf{T}$. So, it can also be applied, like here, to messages belonging to the discrete torus $\mathbf{T}_{\text{pi}}$.

20 [0118] Homomorphic multiplication

[0119] First, it is noted that the usual, natural multiplication cannot be used as such for multiplying two messages belonging to the discrete torus $\mathbf{T}_{\text{pi}}$.

[0120] For elements belonging to the discrete torus $\mathbf{T}_{\text{pi}}$, a multiplication, noted $\times_T$, can however be defined, according to the following equation:

$$m3_i = m1_i \times_T m2_i := (x1_i \cdot x2_i \bmod(p_i)) / p_i = (m1_i \cdot m2_i) \cdot p_i \bmod(1)$$

where $m1_i = x1_i / p_i$ and $m2_i = x2_i / p_i$ each belong each to T_{p_i} , $x1_i$ and $x2_i$ being integer numbers, and where “.” is the usual, natural multiplication.

[0121] Clearly, the product $m3_i = m1_i \times_T m2_i$ belongs also to T_{p_i} . Besides, this multiplication between elements of the discrete torus complies with the modular nature of T_{p_i} . Indeed, for any couple of messages $m1_i, m2_i$ belonging to $T_{p_i}^2$:

$$(m1_i + 1) \times_T m2_i = ((x1_i + p_i) \cdot x2_i \bmod(p_i)) / p_i = (x1_i \cdot x2_i \bmod(p_i)) / p_i$$

as $x2_i$ is an integer number, and so, $(m1_i + 1) \times_T m2_i = m1_i \times_T m2_i$.

[0122] The homomorphic multiplication procedure described here is noted $\odot$.

10 [0123] It produces a cyphertext $c3_i = c1_i \odot c2_i$ which is an encrypted version of $m3_i = m1_i \times_T m2_i$, $c1_i$ and $c2_i$ being encrypted versions of $m1_i$ and $m2_i$ respectively.

[0124] To determine $c3_i$, that is to say to determine homomorphically the quantity $(x1_i \cdot x2_i) / p_i \bmod(1)$, the bootstrapping procedure described above is exploited, as well as the fact that $xy = ((x+y)^2 - (x-y)^2) / 4$.

15 [0125] This identity is employed to switch from the two-operands multiplication operation xy to a single-operand squaring operation, this squaring operation being carried on homomorphically thanks to this specific bootstrapping procedure (thanks to an appropriate choice of the function g).

[0126] More precisely, to determine homomorphically
20 $m1_i \times_T m2_i = (x1_i \cdot x2_i) / p_i \bmod(1)$, one determines homomorphically the quantity M below :

$$M = \frac{(p_i + 1)}{2} \cdot \left[\frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i + m2_i)^2 - \frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i - m2_i)^2 \right]$$

where $(m1_i + m2_i)^2 = (m1_i + m2_i) \cdot (m1_i + m2_i)$, “.” being the usual, natural multiplication.

25 [0127] Indeed,

$$\begin{aligned}
& \frac{(p_i + 1)}{2} \cdot \left[\frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i + m2_i)^2 - \frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i - m2_i)^2 \right] \\
&= \frac{(p_i + 1)}{2} \cdot \left[\frac{(p_i + 1)}{2} \cdot \frac{(x1_i + x2_i)^2}{p_i} - \frac{(p_i + 1)}{2} \cdot \frac{(x1_i - x2_i)^2}{p_i} \right] \\
&= \frac{(p_i + 1)}{2} \cdot 2 \cdot \frac{x1_i \cdot x2_i}{p_i} = \frac{x1_i \times x2_i}{p_i}
\end{aligned}$$

as $(p_i + 1)/2$ is the inverse of 2, in $\mathbb{Z}_{p_i}$.

- 5 [0128] In this respect, it is noted that it is useful to have p_i odd, as it allows 2 to be inversible, in $\mathbb{Z}_{p_i}$.

[0129] To determine homomorphically the quantity

$$\frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i + m2_i)^2 - \frac{(p_i + 1)}{2} \cdot p_i \cdot (m1_i - m2_i)^2$$

- one computes $G_{sq}(c1_i \oplus c2_i) \ominus G_{sq}(c1_i \ominus c2_i)$ where G_{sq} denotes the bootstrapping
 10 procedure described above, in the particular case where the function g is the function:

$$g_{sq} : m_i = x_i/p_i \rightarrow \frac{(p_i + 1)}{2} \cdot p_i \cdot (m_i)^2 = \frac{(p_i + 1)}{2} \cdot [(x_i)^2/p_i \bmod(1)]$$

[0130] And so, finally $c3_i$ is determined, homomorphically, from $c1_i$ and $c2_i$, according to the following equation, where $\boxtimes$ is the external product mentioned above (see figure 1):

$$c3_i = c1_i \odot c2_i = \chi_s \left(\frac{p_i + 1}{2} \right) \boxtimes [G_{sq}(c1_i \oplus c2_i) \ominus G_{sq}(c1_i \ominus c2_i)]$$

- 15 [0131] As the skilled person will appreciate, several modifications can be made to this homomorphic multiplication procedure, without departing from the disclosed technology. For instance, instead of using the bootstrapping “function” G_{sq} to determine homomorphically the quantity M , one could use another bootstrapping “function”, noted $G_{sq'}$, which corresponds to the bootstrapping procedure described above in the particular
 20 case where the function g is the function:

$$g_{sq'} : m_i = x_i/p_i \rightarrow m_i \times_T m_i = p_i \cdot (m_i)^2 = (x_i)^2/p_i \bmod(1)$$

[0132] In this case the encrypted product $c3_i$ is determined, homomorphically, according to the following equation:

$$c3_i = \chi_s \left(\frac{p_i + 1}{2} \right) \boxtimes \left[\left(\chi_s \left(\frac{p_i + 1}{2} \right) \boxtimes G_{sq'}(c1_i \oplus c2_i) \right) \ominus \left(\chi_s \left(\frac{p_i + 1}{2} \right) \boxtimes G_{sq'}(c1_i \ominus c2_i) \right) \right]$$

[0133] Still, using G_{sq} will enable a faster computation than $G_{sq'}$, as it prevents from computing two extra external products $\boxtimes$.

[0134] And for even faster computation, $c3_i$ could be determined homomorphically, from
5 $c1_i$ and $c2_i$, according to the following equation:

$$c3_i = c1_i \odot c2_i = G_{sq''}(c1_i \oplus c2_i) \ominus G_{sq''}(c1_i \ominus c2_i)$$

where $G_{sq''}$ denotes the bootstrapping function presented above in the case where the function g is the function:

$$g_{sq''} : m_i = x_i/p_i \rightarrow \left[\frac{(p_i + 1)}{2} \right]^2 \cdot p_i \cdot (m_i)^2 = \left[\frac{(p_i + 1)}{2} \right]^2 \cdot [(x_i)^2/p_i \bmod(1)]$$

10

[0135] Homomorphic evaluation of a two-variable function

[0136] The disclosed technology concerns also a method for evaluating a two-variable function, that is an arbitrary two-operands function, homomorphically.

[0137] Being able to evaluate such a quantity is very useful for homomorphic data
15 bases processing. Indeed, it enables, inter alia, to compare two ciphers texts $c1_i$ and $c2_i$ without decrypting them (for instance to sort the data base). To this end, the two-variable function employed could be the Max, or Min function. In this respect, one may note that a function returning (in an encrypted form) the maximum, or the minimum of a set of two messages is a two-variable function. And due to the modular nature of the messages
20 employed here, the maximum or minimum of two messages cannot be determined by simply looking at the sign of the difference of these two messages.

[0138] As explained below, to evaluate homomorphically a two-variable function f , each of the two inputs of this function (which belong each to $\mathbf{T}_{pi}$) is mapped on a specifically designed set of values $\underline{S}$. The elements $\underline{\mu}_i$ of $\underline{S}$ belong each to $\mathbf{T}$, and such that, for any
25 set of indexes (i,j,k,l) with $(i,j) \neq (k,l)$, (k,l) being also different from (j,i) :

25

$\alpha.\underline{\mu}_i + \beta.\underline{\mu}_j$ is different from $\alpha.\underline{\mu}_k + \beta.\underline{\mu}_l$, modulo $1/2$, where α and β are two constant and integer, given numbers.

So:

$$\forall (i,j) \neq (k,l), \text{ with } (j,i) \neq (l,k) : \alpha.\underline{\mu}_i + \beta.\underline{\mu}_j \neq \alpha.\underline{\mu}_k + \beta.\underline{\mu}_l \quad \text{and} \quad \alpha.\underline{\mu}_i + \beta.\underline{\mu}_j \neq \alpha.\underline{\mu}_k + \beta.\underline{\mu}_l + 1/2$$

[0139] Then, taking advantage of this property, the two-operand function f is transformed into a one-operand function g_f by first computing a linear combination of these two messages (multiplied respectively by the coefficients α and β), in order to obtain a single operand (the two messages having been preliminarily mapped onto $\underline{S}$). The one-operand function g_f is then homomorphically evaluated, by using the specific bootstrapping procedure described above.

[0140] More precisely, the two input messages $m1_i$ and $m2_i$ are first mapped onto corresponding elements of $\underline{S}$, thanks to a mapping function g_{map} :

$$g_{\text{map}} : \quad \forall m \in \mathbf{T}_{pi} \rightarrow \underline{m} = g_{\text{map}}(m) \in \underline{S}$$

15 where $\underline{m}$ is one of the elements of $\underline{S}$.

[0141] g_{map} is chosen among the different possible functions that map $\mathbf{T}_{pi}$ on $\underline{S}$.

[0142] It may be noted that it can be proven that it is always possible to build a set $\underline{S}$ fulfilling the condition above (and fulfilling also one the additional conditions given below for $\underline{S}$). In practice, the set $\underline{S}$ can be chosen among different sets fulfilling these conditions. Preferably, the set S that is chosen is the one for which the values $\underline{\mu}_j$ of the set are the more spaced apart from each other.

[0143] The “mapped” messages associated respectively to $m1_i$ and $m2_i$ are noted $\underline{m1}_i$ and $\underline{m2}_i$ respectively: $\underline{m1}_i = g_{\text{map}}(m1_i)$, $\underline{m2}_i = g_{\text{map}}(m2_i)$.

[0144] The “one-operand” function g_f corresponding to f is then defined as:

$$25 \quad \forall (m1_j, m2_j) \in \mathbf{T}_{pi}^2 \quad g_f(\alpha.\underline{m1}_j + \beta.\underline{m2}_j) = f(m1_j, m2_j)$$

that is:

$$\forall (m1_j, m2_j) \in \mathbf{T}_{pi}^2 \quad g_f[\alpha.g_{\text{map}}(m1_j) + \beta.g_{\text{map}}(m2_j)] = f(m1_j, m2_j)$$

[0145] The fact that two different couples of messages $(m1_i, m2_i)$ and $(m1'_i, m2'_i)$ always lead to two different values of the intermediate quantity $\alpha \cdot \underline{m1}_i + \beta \cdot \underline{m2}_i$ (respectively, $\alpha \cdot \underline{m1}'_i + \beta \cdot \underline{m2}'_i$), due to the specific structure of $\underline{S}$, is one of the elements that enables to apply the method to any, arbitrary two variable function f from $\mathbf{T}_{pi} \times \mathbf{T}_{pi}$ to $\mathbf{T}_{pi}$, or even from $\mathbf{T}_{pi} \times \mathbf{T}_{pi}$ to $\mathbf{T}$.

[0146] The encrypted version of $f(m1_i, m2_i)$ is noted

$$c_{f,i} = F(c1_i, c2_i)$$

[0147] When $c_{f,i}$ is decrypted using the secret key s , it gives $f(m1_i, m2_i)$.

[0148] $c_{f,i}$ is determined, homomorphically, from the ciphertexts $c1_i$ and $c2_i$ (without decrypting $c1_i$ or $c2_i$), according to the following equation:

$$c_{f,i} = F(c1_i, c2_i) = G_f \left[\left(\alpha \cdot G_{map}(c1_i) \right) \oplus \left(\beta \cdot G_{map}(c2_i) \right) \right]$$

where G_f denotes the bootstrapping function described above in the particular case where g is g_i , and

where G_{map} denotes the bootstrapping function described above, in the particular case where g is g_{map} .

[0149] It may be noted that the unciphered values of α and β are used directly to compute $c_{f,i}$. In other words, the coefficients α and β are not ciphered.

[0150] This method can be applied to any function f (in particular, whether f is symmetric or not).

[0151] When the function f is symmetric, α and β are chosen equal to each other. For instance, they may be chosen each as being equal to 1. In this case, $c_{f,i}$ is determined homomorphically from $c1_i$ and $c2_i$ according to the following equation (see figure 2):

$$c_{f,i} = G_f[G_{map}(c1_i) \oplus G_{map}(c2_i)]$$

[0152] This way to compute $c_{f,i}$ is beneficial as there is no multiplication by α or β to be carried on, and as the noise terms are not amplified, as they are multiplied by 1.

[0153] It may be noted that the $\text{Max}(c1_i, c2_i)$ function, for instance, whose output is the cipher text $c1_i$, or $c2_i$, that is associated to the highest of the two messages $m1_i$ and $m2_i$ (highest, with respect to usual order of T_{pi} values), is a symmetric function.

[0154] When the function f is not symmetric, α and β are chosen different from each other, and the elements of $\underline{S}$ are such that, for any set of indexes (i,j,k,l) with $(i,j) \neq (k,l)$, $\alpha.\underline{u}_i + \beta.\underline{u}_j$ is different from $\alpha.\underline{u}_k + \beta.\underline{u}_l$ modulo $\frac{1}{2}$, even when $(k,l) = (j,i)$.

[0155] In this case, α and β can be chosen as being equal to +1 and -1 respectively, for instance. $c_{f,i}$ is then determined homomorphically from $c1_i$ and $c2_i$ according to the following equation

$$c_{f,i} = G_f[G_{map}(c1_i) \ominus G_{map}(c2_i)]$$

Again, this kind of implementation is beneficial, as it involves no multiplication by a coefficient higher than 1, and so no noise amplification.

[0156] This method for evaluating homomorphically a two-variable function has been presented above in a case where the function f is a function from $T_{pi} \times T_{pi}$ to T_{pi} , or from $T_{pi} \times T_{pi}$ to T . Still, it can be implemented identically when the two variables of the function f each belongs to a discrete set of p_i elements (possibly different from T_{pi} , but consisting of p_i elements, with p_i odd, and such that twice the difference between any two of the elements of this set is not an integer number), the values of function f belonging to T .

20 [0157] **Carry-less processing of a message m, by decomposing it into orthogonal sub-messages m_i .**

[0158] As mentioned above, the disclosed technology also concerns a technique for encrypting and processing 'very long' messages x (or m), using a specific decomposition of the message x into smaller messages x_i , this decomposition being based on the Chinese remainder theorem decomposition and enabling to process the sub-messages x_i , or, for instance, their counterparts x_i/p_i , separately from each other, without having to take into account possible carries.

[0159] This technique can be applied to any message belonging to $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$, p being equal to $\prod_{i=1}^r p_i$, where the integer numbers p_i are pairwise coprime.

[0160] More precisely, for any message x belonging to $\mathbb{Z}_p$, the corresponding encrypted message C is determined by executing the following operations (figure 3):

- Step D: decomposing x into its components x_i , $i=1\dots r$, with x_i equal to x modulo p_i (the Chinese remainder theorem stating that such a decomposition is always possible),
- Step E: for each component x_i whose associated factor p_i is odd, determining an encrypted version c_i of a message m_i , according to the encryption method presented above, m_i being, among the elements of the discrete set of p_i elements mentioned above (e.g.: T_{p_i}), the element that is associated to x_i by a given bijective relationship between $\mathbb{Z}_{p_i}$ and said discrete set,
- if a component x_i is associated to a factor p_i equal to 2, determining an encrypted version c_i of a quantity Q_i associated to x_i and that can takes two distinct values, c_i being equal to (a_i, b_i) with $b_i = Q_i + e_i + a_i \cdot s_i$
- returning $C = (c_1, \dots, c_i, \dots, c_r)$.

[0161] Below, we consider the case in which the discrete set consisting of p_i distinct elements is T_{p_i} . The component x_i may be associated (bijectively) to a corresponding element m_i of T_{p_i} according to the following equation, for instance: $m_i = x_i/p_i$, or according to the following equation $m_i = -x_i/p_i$, or, alternatively, according to the following equation $m_i = (x_i + 1)/p_i$.

[0162] Besides, we consider below that the same secret key s is used on the different fields $\mathbb{Z}_{p_1}, \dots, \mathbb{Z}_{p_i}, \dots, \mathbb{Z}_{p_r}$. In other words, here, the same secret key s is used to cipher the different messages $m_1, \dots, m_i, \dots, m_r$. Still, in alternative, different keys could be used to cipher the different (sub-) messages $m_1, \dots, m_i, \dots, m_r$.

[0163] It may be noted that, in other embodiments, the discrete set in question may be another set than T_{p_i} , and that another operation than the one mentioned above could be used to map $\mathbb{Z}_{p_i}$ onto this discrete set (for instance, m_i could be obtained by multiplying x_i by any integer number that is coprime with p_i , and then be divided by p_i).

[0164] Besides, here, the decomposition of x is carried on such that the only even factor p_i , should there be one, is two. So, except from, possibly, a factor two, all the other

factors p_i are odd. And so, their corresponding messages m_i can all be processed, individually, according to the methods described in detail above, for elements belonging to T_{p_i} , except, possibly, for one of them, associated to a factor p_i equal to two, which is processed, for instance, according to existing, known techniques for binary messages.

5 [0165] When processing such an encrypted message $C=(c_1,...c_i,...c_r)$, or when processing a couple $(C1,C2)$ of two such messages $C1=(c1_1,...c1_i,...c1_r)$ and $C2=(c2_1,...c2_i,...c2_r)$, the components $c_1,...c_i,...c_r$ of C , or the couples of components $(c1_1, c2_1) \dots (c1_i, c2_i), \dots (c1_r, c2_r)$ are processed, homomorphically, independently from each other (that is, without taking into account any carry).

10 [0166] For instance, the homomorphic addition of two such messages $C1$ and $C2$ is achieved as follow (see figure 4):

$$C1 \oplus C2 = (c1_1 \oplus c2_1, \dots, c1_i \oplus c2_i, \dots, c1_r \oplus c2_r)$$

[0167] Similarly, the homomorphic multiplication of two such messages $C1$ and $C2$ is achieved as follow, based on the $T_{p_i} \times T_{p_i}$ homomorphic multiplication

15 $\odot$ presented above (see figure 5):

$$C1 \odot C2 = (c1_1 \odot c2_1, \dots, c1_i \odot c2_i, \dots, c1_r \odot c2_r)$$

Similarly, a refreshed version C' of such an encrypted message $C=(c_1,...c_i,...c_r)$ is determined by computing the following quantity $C' = (G(c_1), \dots, G(c_i), \dots, G(c_r))$.

[0168] Electronic devices and system

20 [0169] Figure 6 represents schematically a cryptographic system 1 comprising an encrypting device 2 and a processing device 3, for processing encrypted messages without decrypting them.

[0170] The encrypting device 2 and the processing device 3 comprise each at least one processor, and at least one memory.

25 [0171] The encrypting device 2 is programmed, or otherwise arranged for executing the following steps:

- receiving data representative of an unencrypted message m_i , that belongsto a discrete set consisting of p_i distinct elements, with p_i being an odd number, said discrete set being the discrete torus $T_{p_i} =$

$\frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or the set $T_{p_i, N}[X]$ of polynomials of degree $N-1$ whose coefficients belong to T_{p_i} , or to another discrete set in bijective relationship with T_{p_i} ,

- 5 - processing said data according to the method for encrypting such messages, that has been presented above, to determine encrypted data representative of the encrypted message c_i , the secret key s employed during said processing, for instance stored, at least temporarily, in the memory of the encrypting device, being accessed by the processor or processors during said processing.

10 [0172] The processing device 3, which is distinct from the encrypting device 2, is configured for receiving the encrypted message c_i from the encrypting device 2, and, possibly, to receive other such encrypted messages.

[0173] The processing device 3 is programmed, or otherwise arranged to execute the following steps:

- 15 - receiving data, representative of the encrypted message c_i produced by the encrypting device 2,
- processing said data, according to the method for bootstrapping described above, to determine refreshed data, representative of the encrypted message c'_i , said processing being achieved without decrypting c_i and
- 20 without using, reading or accessing the secret key s .

[0174] In particular, during this bootstrapping, the unencrypted message m_i corresponding to c_i does not appear nor result from an operation, nor it is used, read or accessed. The unencrypted message m_i , just as the secret key s , thus remains absent from the processing device 3 at all time.

25 [0175] The processing device could also be programmed to execute the homomorphic multiplication method presented above and/or to homomorphically evaluate of a two-variable function like the one presented above.

[0176] The processing device 3 could also be programmed to execute one or several of the homomorphic processing methods for encrypted messages like the message

$C=(c_1,...c_i,...c_r)$, that has been presented above. And the encrypting device 2 could be programmed to produce such encrypted messages, from unencrypted messages x belonging to $\mathbf{Z}_p$.

CLAIMS

1. Method for determining an encrypted message c_i , which is an encrypted version
 5 of a message m_i , c_i being equal to (a_i, b_i) where b_i is determined from m_i and from a secret key s , b_i being equal to $m_i + e_i + a_i \cdot s$ with a_i being a randomly selected vector for projecting s and e_i being a random noise component added to $m_i + a_i \cdot s$,

wherein the message m_i belongs to a discrete set consisting of p_i distinct elements, with p_i being an odd number, the elements of said discrete set being such that twice the
 10 difference between any two of these elements is not an integer number, said discrete set being the discrete torus $T_{p_i} = \frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or the set $T_{p_i, N}[X]$ of polynomials of degree $N-1$ whose coefficients belong to T_{p_i} , or another discrete set in bijective relationship with T_{p_i} .

15 2. A Method for bootstrapping the encrypted message $c_i = (a_i, b_i)$ determined according to claim 1, the method for bootstrapping comprising a determination of a refreshed encrypted message $c_i' = (a_i', b_i')$, which is an encrypted version of a value $g(m_i)$ of a function g applied to the message m_i , c_i' having a noise component e_i' smaller than e_i .

20

3. Method according to claim 2, wherein the determination of c_i' comprises a step of determining homomorphically the constant coefficient $\text{coef}_0(X^q \cdot W_g(X))$ of $X^q \cdot W_g(X) \bmod (X^N + 1)$, where

- $W_g(X) = \sum_{j=0}^{N-1} w_j \cdot X^j$ is a polynomial of degree $N-1$ whose coefficients w_j are given
 25 by the following formula:

$$(-1)^{\lfloor \lfloor 2Nm_i \rfloor / N \rfloor} \cdot w_j = g(m_i), \text{ where } j = N \lfloor \lfloor 2Nm_i \rfloor / N \rfloor - \lfloor 2Nm_i \rfloor$$

with $\lfloor 2Nm_i \rfloor$ being the integer number that is the nearest to $2Nm_i$ and with $\lfloor \lfloor 2Nm_i \rfloor / N \rfloor$ being the ceiling function applied to $\lfloor 2Nm_i \rfloor / N$, and where

- q is the integer number that is the nearest to $2N(b_i - a_i \cdot s)$: $q = \lfloor 2N(b_i - a_i \cdot s) \rfloor$.

30

4. Method according to claim 3, wherein N is below 50 times p_i , or even below 30 times p_i .

5. Method according to claim 3 or 4, wherein N is above p_i , or even above 5 times p_i .

6. Method for homomorphically multiplying two messages m_{1i} and m_{2i} , each belonging to the discrete torus T_{p_i} or to the set of polynomials $T_{p_i, N}[X]$, the method comprising a determination of an encrypted message c_{3i} which is an encrypted version
10 of the product of m_{1i} by m_{2i} , c_{3i} being determined from encrypted messages c_{1i} and c_{2i} without decrypting c_{1i} or c_{2i} , c_{1i} and c_{2i} being encrypted versions of m_{1i} and m_{2i} respectively, that have been determined according to the method of claim 1,

the determination of c_{3i} comprising homomorphically determining

$$\frac{(p_i + 1)}{2} \cdot \left[\frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} + m_{2i})^2 - \frac{(p_i + 1)}{2} \cdot p_i \cdot (m_{1i} - m_{2i})^2 \right]$$

15 Where $p_i \cdot (m_{1i} + m_{2i})^2$ and $p_i \cdot (m_{1i} - m_{2i})^2$, or $\frac{(p_i+1)}{2} \cdot p_i \cdot (m_{1i} + m_{2i})^2$ and $\frac{(p_i+1)}{2} \cdot p_i \cdot (m_{1i} - m_{2i})^2$, or $\left[\frac{(p_i+1)}{2}\right]^2 \cdot p_i \cdot (m_{1i} + m_{2i})^2$ and $\left[\frac{(p_i+1)}{2}\right]^2 \cdot p_i \cdot (m_{1i} - m_{2i})^2$ are determined homomorphically by executing the method for bootstrapping according to any of claims 2 to 5, the function g being the function $g_{sq'} : m_i \rightarrow p_i \cdot (m_i)^2 \bmod(1)$, or $g_{sq} : m_i \rightarrow \frac{(p_i+1)}{2} \cdot [p_i \cdot (m_i)^2 \bmod(1)]$, or,
20 respectively, $g_{sq''} : m_i \rightarrow \left[\frac{(p_i+1)}{2}\right]^2 \cdot [p_i \cdot (m_i)^2 \bmod(1)]$.

7. Method for determining an encrypted message $c_{i,j}$, which is an encrypted version of a value $f(m_{1i}, m_{2i})$ of a two-variable function f applied to messages m_{1i} and m_{2i} , m_{1i} and m_{2i} belonging each to said discrete set,

25 $c_{i,j}$ being determined from encrypted messages c_{1i} and c_{2i} without decrypting c_{1i} or c_{2i} , c_{1i} and c_{2i} being encrypted versions of m_{1i} and m_{2i} respectively, that have been determined according to the method of claim 1,

the determination of $c_{i,j}$ comprising:

34

- determining $c1'_i$ and $c2'_i$, from $c1_i$ and $c2_i$ respectively, by executing the method for bootstrapping according to any of claims 2 to 5, the function g employed during this bootstrapping being a function g_{map} that maps said discrete set onto a set $\underline{S}$ whose elements $\underline{u}_j$ belong each to the torus $\mathbb{T} =]-1/2, 1/2]$ and are such that,
- 5 for any set of indexes (i, j, k, l) with $(i, j) \neq (k, l)$, (k, l) being different from (j, i) :
 $\alpha \cdot \underline{u}_i + \beta \cdot \underline{u}_j$ is different from $\alpha \cdot \underline{u}_k + \beta \cdot \underline{u}_l$ modulo $1/2$,
 with α and β being two constant and integer, given numbers,
 $c1'_i$ and $c2'_i$ being the encrypted versions of $\underline{m1}_i = g_{\text{map}}(m1_i)$ and $\underline{m2}_i = g_{\text{map}}(m2_i)$ respectively, where $\underline{m1}_i$ and $\underline{m2}_i$ belong each to $\underline{S}$,
- 10 - determining homomorphically, from $c1'_i$ and $c2'_i$, an encrypted message c_s which is an encrypted version of $\alpha \cdot \underline{m1}_i + \beta \cdot \underline{m2}_i$,
- determining $c_{f,i}$ from c_s by executing the method for bootstrapping according to any of claims 2 to 5, the function g employed during this bootstrapping being a one-variable function g_f defined by the following condition: for any couple of
- 15 messages $m1_j, m2_j$ belonging each to said discrete set,

$$g_f(\alpha \cdot \underline{m1}_j + \beta \cdot \underline{m2}_j) = f(m1_j, m2_j)$$

that is:

$$g_f[\alpha \cdot g_{\text{map}}(m1_i) + \beta \cdot g_{\text{map}}(m2_i)] = f(m1_i, m2_i).$$

- 20 8. Method according to claim 7, wherein the elements of the set $\underline{S}$ are such that for any set of indexes (i, j, k, l) with $(i, j) \neq (k, l)$, $\alpha \cdot \underline{u}_i + \beta \cdot \underline{u}_j$ is different from $\alpha \cdot \underline{u}_k + \beta \cdot \underline{u}_l$ modulo $1/2$, even when $(k, l) = (j, i)$.

9. Method according to claim 8, wherein $\alpha=1$ and $\beta=-1$, c_s being determined by
- 25 computing the homomorphic difference $\ominus$ between $c1'_i$ and $c2'_i$: $c_s = c1'_i \ominus c2'_i$, or wherein $\alpha=-1$ and $\beta=1$, c_s being determined by computing the homomorphic difference $\ominus$ between $c2'_i$ and $c1'_i$: $c_s = c2'_i \ominus c1'_i$.

10. Method according to claim 7, wherein the function f is symmetric, $f(m1_i, m2_i)$
- 30 being equal to $f(m2_i, m1_i)$ for any couple of messages $m1_i$ and $m2_i$, and wherein α and β

35

are each equal to 1, c_s being determined by computing the homomorphic sum $\oplus$ of $c1'_i$ and $c2'_i$: $c_s = c1'_i \oplus c2'_i$.

11. Method for determining an encrypted message C , which is an encrypted version of a message x belonging to $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$, p being equal to $\prod_{i=1}^r p_i$ where the integer numbers p_i are pairwise coprime, the method comprising:

- (D) decomposing x into its components x_i , $i=1 \dots r$, with x_i equal to x modulo p_i ,
- (E) for each component x_i whose associated factor p_i is odd, determining an encrypted version c_i of a message m_i , according to the method of claim 1, m_i being, among the elements of said discrete set, the element that is associated to x_i by a given bijective relationship between $\mathbb{Z}_{p_i}$ and said discrete set,
- if a component x_i is associated to a factor p_i equal to 2, determining an encrypted version c_i of a quantity Q_i associated to x_i and that can takes two distinct values, c_i being equal to (a_i, b_i) with $b_i = Q_i + e_i + a_i \cdot s$,
- returning $C=(c_1, \dots, c_i, \dots, c_r)$.

12. Method for bootstrapping the encrypted message $C=(c_1, \dots, c_i, \dots, c_r)$ determined according to claim 11, this method comprising:

- for each c_i whose associated factor p_i is odd, determining a refreshed encrypted message c'_i , from c_i , by executing the method for bootstrapping according to any of claims 2 to 5,
- returning $C'=(c'_1, \dots, c'_i, \dots, c'_r)$.

25

13. Method for determining an encrypted message $C3=(c3_1, \dots, c3_i, \dots, c3_r)$, which is an encrypted version of the product of two messages $x1$ and $x2$, each belonging to $\mathbb{Z}_p$, $C3$ being determined from encrypted messages $C1$ and $C2$, without decrypting $C1$ or $C2$, $C1$ and $C2$ being encrypted versions of $x1$ and $x2$ respectively, $C1=(c1_1, \dots, c1_i, \dots, c1_r)$ and $C2=(c2_1, \dots, c2_i, \dots, c2_r)$ having been determined from $x1$ and $x2$

30

36

respectively, according to the method of claim 11, wherein said discrete set is T_{p_i} or $T_{p_i, N}[X]$,

wherein each component c_{3i} of C_3 is determined from the corresponding components c_{1i} and c_{2i} of C_1 and C_2 , according to the method for multiplying of claim 6,
5 when c_{1i} and c_{2i} are associated to a factor p_i that is odd.

14. Encrypting device (2), comprising one or more processors and at least one memory, the encrypting device being programmed to make the processor or processors executing the following steps:

- 10 - receiving data representative of an unencrypted message m_i , that belongs to a discrete set consisting of p_i distinct elements, with p_i being an odd number and wherein the elements of said discrete set are such that twice the difference between any two of these elements is not an integer number, said discrete set being the discrete torus $T_{p_i} = \frac{1}{p_i} \left\{ -\frac{p_i-1}{2}, -\frac{p_i-1}{2} + 1, \dots, \frac{p_i-1}{2} - 1, \frac{p_i-1}{2} \right\}$, or the set $T_{p_i, N}[X]$ of polynomials of degree $N-1$ whose
15 coefficients belong to T_{p_i} , or to another discrete set in bijective relationship with T_{p_i} ,
- processing said data according to the method of claim 1, to determine encrypted data representative of the encrypted message c_i , the secret key
20 s employed during said processing, stored at least transitorily in the memory of the encrypting device, being accessed by the processor or processors during said processing.

15. Cryptographic system (1) comprising the encrypting device (2) of claim 14 and a
25 processing device (3),

the processing device (3) comprising one or more processors and at least one memory, the processing device being programmed to make the processor or processors executing the following steps:

- receiving data, representative of the encrypted message c_i that has been
30 produced by the encrypting device (2),

- processing said data, according to the method for bootstrapping of any of claims 2 to 5, to determine refreshed data, representative of the encrypted message c'_i , said processing being achieved without decrypting c_i and without using, reading or accessing the secret key s .

5

16. Computer-readable program product comprising instructions that, when executed by a computer, make the computer to execute the bootstrapping method according to any of claim 2 to 5.

- 10 17. Computer-readable program product comprising instructions that, when executed by a computer, make the computer to execute the encrypting method according to claim 11.