(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
19 février 2004 (19.02.2004)

PCT

(10) Numéro de publication internationale
WO 2004/015559 A2

(51) Classification internationale des brevets[7] : G06F 7/72

(21) Numéro de la demande internationale :
PCT/FR2003/050022

(22) Date de dépôt international : 29 juillet 2003 (29.07.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/09942 5 août 2002 (05.08.2002) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **EVERBEE NETWORKS** [FR/FR]; 41, boulevard des Capucines, F-75002 Paris (FR).

(72) Inventeur; et
(75) Inventeur/Déposant *(pour US seulement)* : **STEHLE, Jean-Luc** [FR/FR]; 300, rue de Vaugirard, F-75015 Paris (FR).

(74) Mandataire : **GRYNWALD, Albert**; Cabinet Grynwald, 127, rue du Faubourg Poissonnière, F-75009 Paris (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclaration en vertu de la règle 4.17 :
— *relative à la qualité d'inventeur (règle 4.17.iv)) pour US seulement*

Publiée :
— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(54) **Title:** METHOD FOR ACCELERATING CALCULATIONS IN MODULAR ARITHMETIC

(54) **Titre :** PROCEDE POUR ACCELERER DES CALCULS EN ARITHMETIQUE MODULAIRE

(57) **Abstract:** The invention a method for accelerating exponentiation calculations in arithmetic modulo, a number N stored on q words. The exponentiations are especially involved in cryptography protocols implemented by means of computer resources. According to said method, a first algorithm is suitable for replacing an argument, stored on 2q words, by a result which is congruent modulo N to said argument and the q low-order words of which are null and a first operator takes two entries each stored on q words and outputs a number W, stored on q words, the product by R of which is congruent modulo N to the product of both entries, whereby R is a power of two higher than N. Said method allows computation power and memory space to be saved.

(57) **Abrégé :** L'invention concerne un procédé permettant d'accélérer les calculs d'exponentiation en arithmétique modulo un nombre N stocké sur q mots. Les exponentiations interviennent notamment dans des protocoles de cryptographie mis en œuvre à l'aide de ressources informatiques. Le procédé comporte : un premier algorithme ayant pour objet de remplacer un argument, stocké sur 2q mots, par un résultat qui est congru modulo N audit argument et dont les q mots de poids faibles sont nuls, un premier opérateur prenant deux entrées stockées chacune sur q mots et fournissant en sortie un nombre W, stocké sur q mots, dont le produit par R est congru modulo N au produit des deux entrées. R est une puissance de deux supérieure à N. Le procédé permet d'économiser de la puissance de calcul et de l'espace mémoire.

WO 2004/015559 A2

PROCEDE POUR ACCELERER DES CALCULS EN ARITHMETIQUE MODULAIRE

Préambule de la description

Domaine concerné, problème posé

L'invention présentée ici est un procédé permettant d'accélérer les calculs en arithmétique modulaire, ce qui permet entre autres d'accélérer certaines opérations utilisées dans des protocoles cryptographiques.

Le problème du logarithme discret en arithmétique modulaire est utilisé aussi bien pour des protocoles d'authentification et d'échanges de clés que pour des systèmes de cryptographie asymétriques (comportant une clé publique et une clé privée).

Les principes de ces protocoles sont bien décrits par exemple dans le brevet US # 4 200 770 « Cryptographic Apparatus and Method » de Hellman, Diffie et Merkle du 29 avril 1980 ou dans le brevet US # 4 405 829 « Cryptographic Communication System and Method » de Rivest, Shamir et Adleman du 20 septembre 1983. Le système décrit dans ce second brevet est appelé système RSA de cryptographie à clé publique ou tout simplement RSA. Ces brevets sont tous deux aujourd'hui du domaine public.

La sécurité des dits protocoles repose sur l'impossibilité de résoudre en un temps raisonnable le problème

du logarithme discret en arithmétique modulaire, dont nous rappelons ici le principe.

L'arithmétique modulo N est une technique qui consiste à considérer que deux nombres entiers représentent la même entité mathématique dès que leur différence est un multiple entier d'un nombre N, appelé le module. On dira alors que ces deux nombres sont congrus modulo N.

Dans les applications cryptographiques, ce module N est en général très grand (plusieurs centaines de chiffres décimaux). On utilise couramment des nombres à 512 ou 1024 bits ou plus.

Les entités mathématiques sur lesquelles on travaille sont appelées les entiers modulo N (entiers définis à l'addition près d'un multiple entier du module N). L'ensemble des entiers modulo N peut être assimilé à l'ensemble des entiers compris entre 0 et N-1. L'addition et la multiplication y sont définies de façon similaire à leur définition en arithmétique classique, à ceci près que lorsqu'un résultat dépasse N-1, il est remplacé par le reste de sa division par N (nous écrirons par la suite « reste modulo N »). Nous ne considérerons ici que la multiplication. L'ensemble des entiers modulo N, dont on aura enlevé les éléments non premiers à N, forme un groupe pour l'opération de multiplication, qu'on appellera groupe multiplicatif des entiers modulo N.

On peut réaliser en un temps raisonnable des additions et des multiplications modulo N. Le temps de calcul d'une addition est approximativement proportionnel au nombre de chiffres nécessaires pour représenter N. Le temps de calcul de la multiplication est approximativement proportionnel au carré de ce nombre de chiffres.

Notons cependant que, partant de deux nombres a et b définis modulo N (en pratique ce seront des entiers compris entre 0 et N-1), la multiplication de a par b donne, en général, un nombre bien plus grand que N . Il faudra alors en calculer le reste modulo N en vue d'obtenir un nombre compris entre 0 et

N-1. Cette opération sera appelée la « réduction modulo N ». Elle implique de faire une division par N. Or une division est, en général, bien plus longue à calculer qu'une multiplication, et ce temps de calcul augmente avec l'ordre de grandeur de N. Le résultat final de ces opérations sera noté a.b [mod N] et se lira « a multiplié par b modulo N ».

Étant donne un nombre X, on appellera inverse de X modulo N un nombre Y tel que le produit X.Y [mod N] soit égal à 1. Le calcul de l'inverse d'un nombre en arithmétique modulo N se fait par l'algorithme d'Euclide généralisé et nécessite en général un temps de calcul significativement plus élevé que celui d'une multiplication. Cet algorithme est présenté en détail par exemple dans « Introduction à l'algorithmique » de Cormen, Leiserson, Rivest dont la traduction française est parue en 1994 chez Dunod. L'édition originale est parue en 1990 chez « The MIT Press, Cambridge Massachusetts » sous le titre « Introduction to Algorithms ».

L'exponentiation, pour une base a et un exposant b est la multiplication entre eux de b nombres égaux chacun à a. On réservera le nom d'exponentiation modulaire à l'exponentiation réalisée dans le groupe multiplicatif des entiers modulo N. Le résultat de cette opération est sera noté ci-après a^b [mod N] et sera appelé « a puissance b modulo N ».

Le calcul de a^b [mod N] est plus complexe qu'une simple addition ou multiplication, mais il peut s'effectuer en un temps raisonnable, et l'objet de la présente invention est un procédé permettant d'accélérer ce calcul.

En revanche, si on connaît la base a et le résultat de l'exponentiation modulaire a^b [mod N], il est en général impossible (pour un module N assez grand), de retrouver en un temps raisonnable l'exposant b. Ce problème est connu sous le nom de « problème du logarithme discret ».

Le nom provient de ce que, si on travaillait sur des nombres réels classiques, connaissant a et c = a^b, l'exposant b serait simplement donné par la relation b = Log c / Log a

Malheureusement (et heureusement pour les cryptographes), il n'existe pas d'équivalent du logarithme dans les groupes utilisés en cryptographie et en particulier en arithmétique modulo N .

L'impossibilité de résoudre le problème du logarithme discret en un temps raisonnable fait que l'exponentiation modulaire est une excellente fonction « à sens unique ». Le calcul direct (calculer a^b [mod N] connaissant a et b) est très rapide, quelques fractions de secondes sur les micro-ordinateurs du commerce, pour un module N de plusieurs centaines de bits. Le temps nécessaire au calcul inverse (calculer b connaissant a et a^b [mod N]) peut se chiffrer en milliard d'années, même avec des moyens de calculs importants.

C'est cette fonction d'exponentiation modulaire qui est utilisée dans certaines applications cryptographiques. On bâtit un protocole dans lequel l'utilisateur normal ne fait que les calculs d'exponentiation, calculs raisonnablement rapides sur des machines peu puissantes. Un pirate souhaitant accéder frauduleusement aux informations devra, quant à lui, résoudre le problème inverse, celui du logarithme discret ce qui lui est impossible, même avec les moyens très importants dont pourrait par exemple disposer une organisation criminelle puissante.

En particulier, en ce qui concerne le système à clés publiques RSA, si on disposait d'une technique efficace pour résoudre le problème du logarithme discret, on pourrait retrouver les clés privées à partir des clés publiques, ce qui réduirait à néant la sécurité fournie par ce système.

Le calcul d'une multiplication en arithmétique modulo N se ramène, nous l'avons dit, à une multiplication de deux grands nombres entiers, (temps proportionnel au carré du nombre de chiffres de N), suivi d'une réduction modulo N consistant à calculer le reste d'une division en nombres entiers par N. Le temps de calcul de cette réduction est équivalent à celui de plusieurs multiplication en grands nombres.

L'invention a plus particulièrement pour objet un procédé permettant d'accélérer le calcul de multiplications en arithmétique modulo N, N désignant un entier comportant notamment plusieurs centaines de chiffres décimaux. L'invention concerne également un procédé permettant d'accélérer le calcul des exponentiations en arithmétique modulo N. Les multiplications et les exponentiations modulaires interviennent notamment dans des protocoles de cryptographie mis en œuvre à l'aide de ressources informatiques. Le procédé est plus particulièrement conçu pour économiser les ressources informatiques, telles que puissance de calcul et/ou espace mémoire.

Solution

Première variante de réalisation : Multiplication modulaire

Dans le cas d'une première variante de réalisation le procédé est plus particulièrement conçu pour permettre d'accélérer le calcul de multiplications en arithmétique modulo N.

Étapes préliminaires

Dans le cas de cette première variante de réalisation le procédé comprend les étapes préliminaires suivantes :

- l'étape de stocker le nombre N sur q mots,
- l'étape de soustraire le nombre N d'une puissance de 2 supérieure à N, notée R, pour obtenir un nombre, noté d.

Le nombre R est la première puissance de 2 ne pouvant pas être stockée sur q mots. Le procédé comprend en outre l'étape préliminaire de stocker le nombre d dans une mémoire de la ressource informatique.

De préférence, le nombre d doit être beaucoup plus petit que N, afin d'accélérer les calculs et d'économiser les ressources informatiques. De préférence les valeurs de N appropriées sont telles le nombre d tienne sur peu de mots, et de manière idéale sur un seul mot.

Premier opérateur

Le procédé comporte un premier opérateur prenant en entrée deux nombres E1, E2 stockés chacun sur q mots et fournissant en sortie un nombre W tel que le produit par R du nombre W soit congru modulo N au produit des entrées E1, et E2. Les dites entrées E1, E2 sont ci-après dénommées les entrées du premier opérateur, et le nombre W est ci-après dénommé le résultat du premier opérateur.

Forme de réalisation du premier algorithme

Le procédé comporte un premier algorithme ayant pour objet de remplacer un argument, stocké sur 2q mots, par un résultat qui est congru modulo N à l'argument et dont les q mots de poids faibles sont nuls. Cet argument est ci-après dénommé l'argument du premier algorithme. Le résultat est ci-après dénommé le résultat du premier algorithme.

Il résulte de la combinaison des traits techniques que le résultat du premier algorithme est un multiple de R.

Étapes principales

Le procédé comprend en outre les étapes principales suivantes pour multiplier entre eux modulo N deux nombres A et B,

- l'étape de stocker en mémoire les restes modulo N du produit par R des nombres A et B, les restes stockés en mémoire sont respectivement dénommés le nombre F et le nombre G,
- l'étape de mettre en œuvre le premier opérateur, les entrées E1 et E2 prenant respectivement comme valeur les nombres F et G,
- l'étape de stocker le résultat W du premier opérateur sous la forme d'un nombre à 2q mots, ci-après dénommé H.

Il résulte de la combinaison des traits techniques que ce nombre H est égal au reste modulo N du produit des nombres A, B et R.

Le procédé comprend en outre les étapes principales suivantes :

- l'étape de mettre en œuvre le premier algorithme avec comme argument le nombre H, stocké sur 2q mots, pour fournir un nombre V congru modulo N au nombre H et dont les q mots de poids faibles sont nuls,

- l'étape de lire le nombre à q mots C constitué par les q mots de poids fort du nombre V.

Le nombre C ou le nombre C-N est égal au reste modulo N du produit des nombres A et B.

Forme de réalisation du premier algorithme

De préférence selon l'invention, le premier algorithme consiste à itérer q fois une opération ayant pour effet de remplacer le nombre obtenu au terme de l'itération précédente par un nombre qui lui est congru modulo N et tel que le nombre obtenu au terme de la ième itération ait tous ses mots de poids les plus faibles nuls jusqu'au ième. La ième opération comprend les quatre étapes suivantes :

(a) on multiplie le ième mot, dans l'ordre des poids croissants, du nombre issu de l'itération précédente, ou, lors de la première itération, le mot de poids faible de l'argument du premier algorithme, par une constante prédéfinie de manière à obtenir un résultat dont on ne conserve que le mot de poids faible,

(b) on obtient au terme de l'étape précédente (a) un résultat, dénommé x, qu'on multiplie par le nombre d,

(c) on décale le résultat x de la première étape (a) de q+i crans vers les poids forts et on ajoute le nombre ainsi décalé au nombre issu de l'itération précédente ou lors de la première itération à l'argument du premier algorithme,

(d) on obtient au terme de l'étape (b) un résultat, dénommé y, qu'on décale de i crans dans le sens des poids forts puis qu'on soustrait au nombre issu du résultat de l'addition effectuée à l'étape (c) précédente.

Le résultat de cette soustraction constitue le résultat de l'itération en cours.

On obtient au terme des quatre étapes de la ième itération un nombre qui est congru modulo N à l'argument du premier algorithme et dont tous les mots de poids les plus faibles sont nuls jusqu'au ième.

Forme de réalisation du premier opérateur

De préférence selon l'invention, le premier opérateur comprend les étapes suivantes :

- l'étape de multiplier entre elles les entrées E1 et E2 du premier opérateur, les entrées E1 et E2 ayant pris pour valeurs les nombres F et G, le résultat de cette multiplication est un nombre T stocké sur 2q mots,

- l'étape de mettre en œuvre le premier algorithme avec comme argument le nombre T pour fournir comme résultat un nombre dénommé U, stocké sur 2q mots,

- l'étape de lire le nombre à q mots constitué par les q mots de poids fort du nombre U, le nombre à q mots constitue le résultat W du premier opérateur.

Il résulte de la combinaison des traits techniques que les q mots de poids faible de U sont nuls. Il résulte également de la combinaison des traits techniques que le résultat W multiplié par R est congru modulo N au produit des entrées E1 et E2.

Autre forme de réalisation du premier opérateur

De préférence selon une autre forme de réalisation du premier opérateur, le premier opérateur comprend les étapes préliminaires suivantes :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 du premier opérateur, l'entrée E1 ayant pris pour valeur le nombre F.

Dans le cas de cette forme de réalisation le premier opérateur comprend une étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier le mot ainsi sélectionné par l'entrée E2 du premier opérateur, l'entrée E2 ayant pris pour valeur le nombre G, le résultat de cette multiplication est un nombre stocké sur q+1 mots et est dénommé P,

- (b) la sous-étape d'additionner le nombre P, calculé à l'étape précédente, au contenu de la mémoire Z, le bit de retenue est positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de la mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre N, le résultat de cette multiplication est un nombre dénommé Q stocké sur q+1 mots,

- (f) la sous-étape d'additionner le nombre Q, calculé à l'étape (e) précédente, au contenu de la mémoire Z,

- (g) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente.

A la suite de ces sous étapes, le mot de poids faible de la mémoire Z contient la valeur zéro.

L'étape principale consistant également à itérer les autres sous-étapes suivantes :

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de la mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de la mémoire Z le contenu du bit de retenue,

- (j) la sous-étape de mettre à zéro le bit de retenue.

Le premier opérateur comprend en outre une autre étape principale. Cette autre étape principale consiste lire le mot

suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j), autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, la mémoire Z contient un nombre, compris entre zéro et 2N, dont le produit par R est égal, modulo N, au produit des entrées E1 et E2.

Dans le cas de cette forme de réalisation du premier opérateur, le premier opérateur comprend en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de la mémoire Z,

- l'étape, lorsque ce contenu est supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction est stocké dans la mémoire Z.

Le résultat W du premier opérateur, au sens de la présente invention, est égal au contenu de la mémoire Z.

Autre forme de réalisation du premier opérateur

De préférence selon une autre forme de réalisation du premier opérateur, le premier opérateur comprend l'étape préliminaire suivante :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 du premier opérateur, l'entrée E1 ayant pris pour valeur le nombre F.

Dans le cas de cette autre forme de réalisation du premier opérateur, le premier opérateur comprend une étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier le mot ainsi sélectionné par l'entrée E2 du premier opérateur, l'entrée E2 ayant pris pour valeur le nombre G, le résultat de cette

multiplication est un nombre stocké sur q+1 mots et est dénommé P,

- (b) la sous-étape d'additionner le nombre P, calculé à l'étape précédente, au contenu de la mémoire Z, le bit de retenue est positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de la mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) les sous-étapes :

• d'additionner le résultat m de l'étape précédente au contenu du mot de poids fort de la mémoire Z,

• de stocker le résultat de cette addition dans le même mot de poids fort,

• en cas de dépassement de capacité de cette addition, de mettre à 1 le bit de retenue,

- (f) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre d, le résultat de cette multiplication est un nombre dénommé Q,

- (g) la sous-étape de soustraire le nombre Q, calculé à l'étape précédente, au contenu de la mémoire Z, puis de stocker dans la mémoire Z le résultat de cette soustraction.

Il résulte de la combinaison de ces traits techniques que le mot de poids faible de la mémoire Z contient la valeur zéro.

L'étape principale consiste également à itérer les sous-étapes suivantes :

- (h) la sous-étape de décaler d'un mot, dans la direction des mots de poids faibles, le contenu des mots de la mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de la mémoire Z le contenu du bit de retenue,

- (j) la sous-étape de mettre à zéro le bit de retenue.

Le premier opérateur comprend en outre l'étape principale de lire le mot suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j) autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, la mémoire Z contient un nombre H dont le produit par R est égal, modulo N, au produit des nombres E1 et E2.

Dans le cas cette autre forme de réalisation, le premier opérateur comprend en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de la mémoire Z,

- l'étape, lorsque ce contenu supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction est stocké dans la mémoire Z.

Le résultat W du premier opérateur est égal au contenu de la mémoire Z.

On notera que le stockage du nombre Q (voir sous-étape (f)) nécessite un mot de plus que le nombre de mots nécessaire pour stocker d. Par conséquent on économise d'autant plus de temps de calcul et de mémoire que d tient sur peu de mots, à l'idéal sur un mot.

De préférence, selon l'invention, pour calculer le reste modulo N, dénommé le nombre F, respectivement dénommé le nombre G , du produit par R d'un nombre donné A, respectivement B :

- on multiplie le nombre A, respectivement le nombre B, par le reste modulo N du carré de R, pour obtenir un nombre dénommé T_A, respectivement T_B, stocké sur 2q mots,

- on applique le premier algorithme avec comme argument le nombre T_A, respectivement le nombre T_B, pour obtenir

comme résultat un nombre V_A, respectivement un nombre V_B, qui lui est congru modulo N et dont les q mots de poids faibles sont nuls.

Le nombre F, respectivement le nombre G, est alors le nombre à q mots formés des q mots de poids forts du nombre V_A, respectivement du nombre V_B.

Exponentiation modulaire

L'invention concerne également une seconde variante de réalisation permettant d'accélérer le calcul des exponentiations en arithmétique modulo N. Dans le cas de cette seconde variante de réalisation, le procédé a pour objet d'élever un nombre A, ci après dénommé la base, à une puissance p modulo N. p étant un nombre entier supérieur ou égal à 2, ci-après dénommé l'exposant, et étant représenté sous forme binaire.

Variante de réalisation mettant en œuvre un premier algorithme

Étapes préliminaires

Le procédé comprend les étapes préliminaires suivantes :

- l'étape de stocker le nombre N sur q mots,
- l'étape de soustraire le nombre N d'une puissance de 2 supérieure à N, notée R, pour obtenir un nombre, noté d.

Le nombre R est la première puissance de 2 ne pouvant pas être stockée sur q mots.

Le procédé comprend l'étape préliminaire de stocker le nombre d dans une mémoire de la ressource informatique.

On notera que les étapes préliminaires ci-dessus sont exécutées une fois pour toutes, une fois la valeur de N fixée : elles ne dépendent ni de la base A ni de l'exposant p

Le procédé comprend en outre les étapes préliminaires suivantes :

- l'étape de stocker en mémoire le reste modulo N du produit par R de la base A, le reste stocké en mémoire est dénommé le nombre F,

- l'étape d'initialiser une zone mémoire de 2q mots, ci-après dénommée la zone mémoire Y, avec une copie nombre F.

On notera que les étapes préliminaires ci-dessus dépendent de la base A.

Le procédé comporte un premier algorithme ayant pour objet de remplacer un argument, stocké sur 2q mots, par un résultat qui est congru modulo N à cet argument et dont les q mots de poids faibles sont nuls. Cet argument est ci-après dénommé l'argument du premier algorithme. Le résultat est ci-après dénommé le résultat du premier algorithme.

Il résulte de la combinaison des traits techniques que le résultat du premier algorithme est un multiple de R

Le premier algorithme mis en œuvre dans le cas de la présente deuxième variante réalisation est le même que celui qui a été décrit, notamment sous une forme de réalisation particulière, lors de l'exposé de la première variante de réalisation. On pourra donc se reporter également, en tant que de besoin, à la précédente description du premier algorithme.

Le procédé comprend les étapes principales suivantes :

- l'étape de sélectionner, en commençant par les bits de poids forts, le premier bit suivant le premier bit non nul de l'exposant p,

- l'étape consistant à mettre en œuvre, selon le cas, les opérations ci-après définies :

Cas où le bit sélectionné est égal à zéro :

Dans le cas où le bit sélectionné est égal à zéro :

- (a) on calcule le carré du contenu de la zone mémoire Y,
- (b) on stocke dans la zone mémoire Y, le résultat du calcul précédent,
- (c) on met en œuvre le premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir comme résultat un nombre U_0 congru modulo N à cet argument et dont les q mots de poids faibles sont nuls,

- (d) on stocke dans les q mots de poids faible de la zone mémoire Y, les q mots de poids forts du résultat U_0 du calcul précédent,
- (e) on met à zéro le contenu des q mots de poids forts de la mémoire Y. Cas où le bit sélectionné est égal à un.

Dans le cas où le bit sélectionné est égal à un, on réalise les mêmes opérations (a) à (e) que dans le cas où l'exposant sélectionné est égal à zéro et en outre on multiplie le contenu de la mémoire Y, au terme de l'opération (e), par le nombre F,

- (g) on stocke dans la zone mémoire Y, le résultat du calcul précédent,
- (h) on met en œuvre le premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir comme résultat un nombre U_1 congru modulo N à cet argument et dont les q mots de poids faibles sont nuls,
- (i) on stocke dans les q mots de poids faible de la zone mémoire Y, les q mots de poids forts du résultat U_1 du calcul précédent,
- (j) on met à zéro le contenu des q mots de poids forts de la mémoire Y.

Le procédé comprend en outre l'étape principale suivante :

- l'étape de lire le bit suivant de l'exposant p, dans l'ordre des poids décroissants, et
- lorsque le bit est égal à zéro on itère les opérations (a) à (e),
- lorsque le bit est égal à un on itère les opérations (a) à (j).

Les itérations sont effectuées autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné tous les bits de l'exposant, la mémoire Y contient un nombre qui est congru

modulo N au produit du nombre R par le résultat de l'exponentiation A puissance p.

Le procédé comprend en outre l'étape de mettre en œuvre le premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir un nombre S congru modulo N à cet argument et dont les q mots de poids faibles sont nuls.

Il résulte de la combinaison des traits techniques que les q mots de poids fort du nombre S représentent un nombre compris entre zéro et 2N et congru modulo N au résultat de l'exponentiation A puissance p.

Le procédé comprend en outre les étapes finales suivantes :

- l'étape de lire le nombre à q mots D constitué par les q mots de poids fort du nombre S,
- l'étape de comparer ce nombre au nombre N
- l'étape, lorsque D est supérieur à N, de retrancher le nombre N du nombre D.

Il résulte de la combinaison des traits techniques que le nombre D auquel on a éventuellement soustrait N est égal au résultat de l'exponentiation A puissance p modulo N qu'on souhaite calculer.

Variante de réalisation mettant en œuvre un premier opérateur

De préférence selon une variante de réalisation, le procédé comporte un premier opérateur. Le procédé comprend les étapes préliminaires suivantes :

- l'étape de stocker en mémoire le reste modulo N du produit par R du nombre A, le reste stocké en mémoire est dénommé le nombre F,
- l'étape d'initialiser une zone mémoire de q mots, ci-après dénommée la zone mémoire Y, avec le nombre F.

Le premier opérateur prend en entrée deux nombres E1, E2 stockés chacun sur q mots et fournissant en sortie un nombre W, stocké sur q mots, tel que le produit par R du nombre W soit congru modulo N au produit des entrées E1, et E2. Les dites

entrées E1, E2 sont ci-après dénommées les entrées du premier opérateur et le nombre W est ci-après dénommé le résultat du premier opérateur.

Le procédé comprend les étapes principales suivantes :

- l'étape de sélectionner, en commençant par les bits de poids forts, le premier bit suivant le premier bit non nul de l'exposant p,

- l'étape consistant à mettre en œuvre, selon le cas, les opérations ci-après définies.

Cas où le bit sélectionné est égal à zéro

Dans le cas où le bit sélectionné est égal à zéro, les opérations mises en œuvre sont les suivantes :

(a) l'étape de mettre en œuvre le premier opérateur, les deux entrées E1 et E2 du premier opérateur prenant toutes deux une valeur égale au contenu de la zone mémoire Y,

(b) l'étape de stocker le résultat W fourni par le premier opérateur dans la zone mémoire Y,

Cas où le bit sélectionné est égal à un

Dans le cas où le bit sélectionné est égal à un, on réalise les mêmes opérations (a) et (b) que dans le cas où l'exposant sélectionné est égal à zéro et en outre on réalise les deux étapes suivantes :

(c) l'étape de mettre en œuvre le premier opérateur les entrées E1 et E2 de cet opérateur prenant l'une pour valeur le nombre F et l'autre une valeur égale au contenu de la zone mémoire Y,

(d) l'étape de stocker le résultat W fourni par le premier opérateur dans la zone mémoire Y.

Le procédé comprend en outre l'étape principale suivante :

- l'étape de lire le bit suivant de l'exposant p, dans l'ordre des poids décroissants, et

• lorsque le bit est égal à zéro on itère les opérations (a) à (b),

• lorsque le bit est égal à un on itère les opérations (a) à (d).

Les itérations sont effectuées autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné tous les bits de l'exposant, la mémoire Y contient un nombre qui est congru modulo N au produit du nombre R par le résultat de l'exponentiation A puissance p.

Le procédé comprend en outre l'étape finale de mettre en œuvre le premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir un nombre S congru modulo N à cet argument et dont les q mots de poids faibles sont nuls.

Il résulte de la combinaison des traits techniques que les q mots de poids fort du nombre S représentent un nombre congru modulo N au résultat de l'exponentiation A puissance p.

Le procédé comprend en outre les étapes finales suivantes :

- l'étape de lire le nombre à q mots D constitué par les q mots de poids fort du nombre S,
- l'étape de comparer ce nombre au nombre N,
- l'étape, lorsque D est supérieur à N, de retrancher le nombre N du nombre D.

Il résulte de la combinaison des traits techniques que le nombre D auquel on a éventuellement soustrait N est égal au résultat de l'exponentiation A puissance p modulo N qu'on souhaite calculer.

Forme de réalisation du premier algorithme

De préférence, selon une forme de réalisation particulière du premier algorithme, le premier algorithme consiste à itérer q fois une opération ayant pour effet de remplacer le nombre obtenu au terme de l'itération précédente par un nombre qui lui est congru modulo N et tel que le nombre obtenu au terme de la ième itération ait tous ses mots de poids

les plus faibles nuls jusqu'au ième. La ième opération comprend les quatre étapes suivantes :

(a) on multiplie le ième mot, dans l'ordre des poids croissants, du nombre issu de l'itération précédente, ou lors de la première itération le mot de poids faible de l'argument du premier algorithme, par une constante prédéfinie de manière à obtenir un résultat dont on ne conserve que le mot de poids faible,

(b) on obtient au terme de l'étape précédente (a) un résultat, dénommé x, qu'on multiplie par le nombre d,

(c) on décale le résultat x de la première étape (a) de q+i crans vers les poids forts et on ajoute le nombre ainsi décalé au nombre issu de l'itération précédente ou lors de la première itération à l'argument du premier algorithme,

(d) on obtient au terme de l'étape (b) un résultat, dénommé y, qu'on décale de i crans dans le sens des poids forts puis qu'on soustrait au nombre issu du résultat de l'addition effectuée à l'étape (c) précédente.

Le résultat de cette soustraction constitue le résultat de l'itération en cours.

Il résulte de la combinaison des traits techniques qu'on obtient au terme des quatre étapes de la ième itération un nombre qui est congru modulo N à l'argument du premier algorithme et dont tous les mots de poids les plus faibles sont nuls jusqu'au ième.

Autre forme de réalisation du premier opérateur

De préférence selon une forme de réalisation particulière du premier opérateur, le premier opérateur comprend les étapes suivantes :

- l'étape de multiplier entre elles les entrées E1 et E2 du premier opérateur pour obtenir un nombre T stocké sur 2q mots,

- l'étape de mettre en œuvre le premier algorithme avec comme argument le nombre T pour fournir comme résultat un nombre dénommé U, stocké sur 2q mots,

- l'étape de lire le nombre à q mots constitué par les q mots de poids fort du nombre U, le nombre à q mots constituant le résultat W du premier opérateur.

Il résulte de la combinaison des traits techniques que les q mots de poids faible de U sont nuls. Il résulte également de la combinaison des traits techniques que la sortie W multipliée par R est congrue modulo N au produit des entrées E1 et E2.

Autre forme de réalisation du premier opérateur

Étape préliminaire

De préférence selon une autre forme de réalisation du premier opérateur, le premier opérateur comprend les étapes préliminaires suivantes :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 du premier opérateur.

Étape principale

Le premier opérateur comprend l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier le mot ainsi sélectionné par l'entrée E2, le résultat de cette multiplication est un nombre stocké sur q+1 mots et est dénommé P,

- (b) la sous-étape d'additionner le nombre P, calculé à l'étape précédente, au contenu de la mémoire Z, le bit de retenue est positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de la mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre N, le résultat de cette multiplication est un nombre dénommé Q stocké sur q+1 mots,

- (f) la sous-étape d'additionner le nombre Q, calculé à l'étape (e) précédente, au contenu de la mémoire Z,

- (g) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente.

Le mot de poids faible de la mémoire Z contient la valeur zéro.

L'étape principale comprend également les sous-étapes suivantes :

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de la mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de la mémoire Z le contenu du bit de retenue,

- (j) la sous-étape de mettre à zéro le bit de retenue.

Le premier opérateur comprend en outre l'étape principale de lire le mot suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j), autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, la mémoire Z contient un nombre, compris entre zéro et 2N, dont le produit par R est égal, modulo N, au produit des nombres E1 et E2.

Étape principale finale

Le premier opérateur comprend en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de la mémoire Z,

- l'étape, lorsque ce contenu est supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction est stocké dans la mémoire Z.

Le résultat W du premier opérateur est égal au contenu de la mémoire Z.

Autre forme de réalisation du premier opérateur

Étape préliminaire

De préférence selon une autre forme de réalisation du premier opérateur, le premier opérateur comprend les étapes préliminaires suivantes :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 du premier opérateur.

Étape principale

Le premier opérateur comprend l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier le mot ainsi sélectionné par l'entrée E2 du premier opérateur, le résultat de cette multiplication est un nombre stocké sur q+1 mots et est dénommé P,

- (b) la sous-étape d'additionner le nombre P, calculé à l'étape précédente, au contenu de la mémoire Z, le bit de retenue est positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans la mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de la mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) les sous-étapes :

• d'additionner le résultat m de l'étape précédente au contenu du mot de poids fort de la mémoire Z,

• de stocker le résultat de cette addition dans le même mot de poids fort,

• en cas de dépassement de capacité de cette addition, de mettre à 1 le bit de retenue,

- (f) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre d, le résultat de cette multiplication est un nombre dénommé Q,

- (g) la sous-étape de soustraire le nombre Q, calculé à l'étape précédente, au contenu de la mémoire Z, puis de stocker dans la mémoire Z le résultat de cette soustraction.

Le mot de poids faible de la mémoire Z contient la valeur zéro.

L'étape principale consiste également à itérer les sous-étapes suivantes :

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de la mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de la mémoire Z le contenu du bit de retenue,

- (j) la sous-étape de mettre à zéro le bit de retenue.

Le premier opérateur comprend en outre l'étape principale de lire le mot suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j) autant de fois que nécessaire.

Il résulte de la combinaison des traits techniques qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, la mémoire Z contient un nombre H dont le produit par R est égal, modulo N, au produit des nombres E1 et E2.

Etape principale finale

le premier opérateur comprend en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de la mémoire Z,

- l'étape, lorsque ce contenu supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction est stocké dans la mémoire Z.

Le résultat W du premier opérateur est égal au contenu de la mémoire Z.

Ainsi qu'on l'a déjà noté, le stockage du nombre Q (voir étape (f)) nécessite un mot de plus que le nombre de mots nécessaire pour stocker d. par conséquent, on économise d'autant plus de temps de calcul et de mémoire que d tient sur peu de mots, à l'idéal un seul mot.

De préférence, selon l'invention, le procédé est tel que pour calculer le reste modulo N du produit par R de la base A, dénommé le nombre F :

- on multiplie le nombre A par le reste modulo N du carré de R, pour obtenir un nombre dénommé T, stocké sur 2q mots,

- on applique le premier algorithme avec comme argument le nombre T, pour obtenir comme résultat un nombre V, qui lui est congru modulo N et dont les q mots de poids faibles sont nuls.

Le nombre F est alors le nombre à q mots formés des q mots de poids forts du nombre V.

Description détaillée

D'autres caractéristiques et avantages de l'invention apparaîtront à la lecture de la description de variantes de réalisation de l'invention données à titre d'exemple indicatif et non limitatif.

Avant de décrire comment on réalise en pratique les opérations en arithmétique modulo N, décrivons en détail comment se font les calculs sur des grands nombres

On suppose que les calculs seront réalisés sur un processeur qui fait des opérations élémentaires, additions et multiplications, sur des mots de w bits, par exemple w=32, pour un processeur à 32 bits. On notera B le nombre 2^w, 2 puissance w, et on considère qu'un mot de w bits représente un nombre

entier compris entre 0 et B-1, ou encore un nombre en arithmétique modulo B. Un tel nombre sera appelé un scalaire.

Une addition élémentaire prend deux scalaires, donc deux mots de w bits, et les additionne. Le résultat occupe w+1 bits, compte tenu d'une éventuelle retenue. Il sera stocké sous la forme d'un mot de w bits, contenant le résultat compte non tenu de la retenue, donc le résultat en arithmétique modulo B, accompagné d'un bit indiquant s'il y a ou non retenue, ce bit est parfois appelé bit d'overflow c'est-à-dire bit de dépassement de capacité.

Une multiplication élémentaire prend deux scalaires, donc deux mots de w bits, les multiplie entre eux et stocke le résultat, qui occupe alors 2w bits, sur deux mots de w bits, en séparant les w bits les plus significatifs et les w bits les moins significatifs.

Évaluons le nombres d'opérations élémentaires nécessaires, additions et/ou multiplications, pour une addition et une multiplication en grands nombres. Ce nombre n'est pas équivalent au nombre d'instructions ou au nombre de tops d'horloge nécessaires à un processeur pour réaliser le calcul, car on ne tient pas compte de la richesse potentielle du jeu d'instruction. Celui-ci peut permettre, dans certains cas, au cours de la même instruction, d'ajouter le résultat d'une multiplication au contenu d'un registre, ou de deux registres en séparant le mot de poids fort et le mot de poids faible du résultat. Cela permet donc de réaliser une ou deux additions supplémentaires en même temps que la multiplication demandée, c'est-à-dire dans le même cycle d'horloge et sans nécessiter de temps additionnel. C'est en particulier le cas pour des implémentations dans le silicium en utilisant des processeurs comme le processeur ARM.

Étant donné un grand nombre T, dont l'écriture binaire nécessite r fois w bits, on utilisera, pour son stockage dans un système informatique, une suite de r mots de w bits, notés T[0], T[1],... , T[r-1], en convenant que T est égal à la somme

des produits T[i].(B^i) ,où le point désigne la multiplication et l'expression B^i représente B puissance i, le nombre i variant entre 0 et r-1, et B étant égal à 2^w . Cette représentation sera appelée la représentation binaire standard du nombre T. On conviendra d'écrire l'expression, T[r-1], ... , T[1], T[0] , pour désigner le nombre T dans cette représentation informatique. On prendra garde à l'ordre d'écriture des mots. Cette notation est analogue à la notation décimale classique où par exemple le nombre à quatre chiffres décimaux 4893 représente la somme 4.10^3 + 8.10^2 + 9.10 + 3 . Le nombre B et les mots T[i], dans la représentation introduite ici, jouent les mêmes rôles que le nombre 10 et que les chiffres dans la notation décimale classique. Le nombre i est appelé l'indice ou le poids du mot T[i] dans la représentation de T. La représentation sur r mots de w bits permet de représenter tous les nombres entiers compris entre 0 et B^r - 1, c'est-à-dire 2^(r.w) - 1 . Dans la suite, nous dirons que T est un nombre à r mots, un scalaire étant par définition un nombre à un seul mot.

Dans cette représentation les mots T[j] d'indice élevé seront appelés « mots de poids fort » ou « mots les plus significatifs » et ceux d'indice j faible seront appelés « mots de poids faible » ou « mots les moins significatifs ».

Une addition entre deux grands nombres, respectivement à r et à s mots, en supposant s inférieur ou égal à r, se ramène à s additions mot par mot, les mots étant pris dans l'ordre croissant de leur poids, avec éventuelle propagation d'une retenue qui sera prise en compte à l'étape suivante.

Une méthode simple et naturelle pour réaliser la multiplication entre un nombre T de r mots et un nombre U de s mots consiste à utiliser une zone mémoire V de r+s mots, mémoire qui fonctionnera comme un accumulateur et qui, à la fin du calcul, contiendra le résultat final. Cette zone V sera, au préalable, initialisée à zéro. Puis, pour tout couple formé d'un mot de T et d'un mot de U, donc au total r.s fois, on multiplie entre eux ces deux mots, multiplication élémentaire, résultat

sur deux mots, les deux mots résultats étant ajoutés aux mots adéquats de la zone mémoire V . On a donc au total r.s multiplications et 2 r.s additions, compte non tenu des reports de bits de retenue. Notons que, les r+s mots de la zone V étant au préalable initialisée à zéro, les premières additions sur ces mots sont en fait des affectations, et en toute rigueur, le nombre exact d'additions est égal à 2 r.s - (r+s).

Si on utilise un processeur adéquat, une programmation astucieuse permet de faire pratiquement toutes les additions en même temps que les multiplications. Dans un tel cas, une évaluation approximative du temps de calcul nécessaire, estimé en nombre de cycles d'horloge du processeur utilisé, nécessite uniquement de prendre en compte les multiplications.

En particulier, une multiplication entre un nombre stocké sur r mots et un scalaire, stocké sur s=1 mot, représente r multiplications élémentaires et (r-1) additions élémentaires, pour la prise en compte des retenues, le résultat étant stocké sur r+1 mots.

Une multiplication entre deux nombres représentés chacun sur r mots de w bits (ou l'élévation au carré d'un mot de w bits) représente dans le cas général r^2 (r au carré) multiplications élémentaires et 2r.(r-1) additions élémentaires.

Analysons maintenant le problème des opérations en arithmétique modulo N où N est un grand nombre impair, dont l'écriture binaire nécessite par exemple 512 ou 1024 bits, ce qui correspond à environ 150 ou 300 chiffres décimaux. Ce nombre N sera représenté comme un nombre à q mots. Par exemple si on travaille sur un processeur à w=32 bits et si N est un nombre à 512, respectivement 1024, bits, on aura q = 16, respectivement 32.

On définit alors le nombre R comme étant la première puissance de 2 qui ne peut pas être stockée sur q mots. Ce nombre est égal à 2 à la puissance le produit qw c'est-à-dire à B puissance q, soit 2 puissance 512 ou 2 puissance 1024 dans les

exemples ci-dessus. Ce nombre est, par construction, supérieur à N.

On appelle d la différence R-N , et on note r_d le nombre minimal de mots nécessaire au stockage informatique de la valeur de d, en représentation binaire standard. Ce nombre r_d est toujours inférieur ou égal à q.

Dans les applications pratiques, on se place habituellement dans le cas où le bit de poids fort de N est égal à 1 (N est effectivement un nombre à 512 ou 1024 bits), et on a alors les inégalités $0 < d < R/2 < N < R$. En général d est bien plus petit que la borne ci-dessus, et r_d peut être significativement inférieur à q . Dans le cas idéal r_d sera égal à 1 .

On appellera e le nombre stocké sur un seul mot et tel que le produit d.e ait son mot de poids faible égal à 1. (Le nombre e est donc l'inverse modulo B du mot de poids faible de d). N étant impair, il en est de même de d, donc d est premier à B, ce qui prouve l'existence et l'unicité de e .

On appellera f le reste modulo N du carré de d. Notons que lorsque d est petit, et en particulier lorsque r_d est inférieur à q/2, le nombre f est tout simplement égal au carré du nombre d.

Les nombres d, e et f sont calculés à l'avance et stockés dans la mémoire du processeur utilisé pour effectuer les calculs.

Nous décrivons pour commencer une méthode permettant de remplacer un nombre T par un nombre U qui est congru à T modulo N et dont le mot de poids faible, d'indice 0, est nul. On suppose que T est écrit sur 2q mots, les mots de poids forts étant, si nécessaire, nuls.

Dans un premier temps, on multiplie par e le mot de poids faible de T, et on ne garde que le mot de poids faible du résultat. Ce mot de poids faible sera noté x. Le nombres x est donc le reste par B du produit T[0].e .

D'une part, le mot x est additionné au mot d'indice q de T (avec propagation de l'éventuelle retenue sur le mot d'indice q+1), et d'autre part le mot x est multiplié par d pour obtenir un nombre y qui sera soustrait au nombre T. Notons que le nombre y occupe au maximum r_d+1 mots.

Soit U la nouvelle valeur de T après ces deux opérations. On démontre que ce nombre U est congru modulo N au nombre initial T dont on était parti. Par ailleurs le mot de poids faible de U est nul, cela résulte de ce que le produit d.e est congru à 1 modulo B. Notons que le fait de savoir d'avance que ce mot est nul peut être utilisé lors de l'implémentation informatique de l'algorithme.

De la même façon, si, dans le nombre T, les i mots de poids les plus faibles sont nuls, c'est-à-dire si les mots T[0], T[1], ..., T[i-1], avec les notations décrites précédemment, sont nuls, on peut par une méthode similaire, remplacer le nombre T par un nombre qui lui est congru modulo N et dont le mot d'indice i est nul. Pour ce faire, on calcule le nombre x égal au mot de poids faible du produit e.T[i], et le nombre y égal au produit x.d . Le nombre x sera additionné au mot d'indice q+i de T, avec éventuelle propagation de retenue, et le nombre y sera décalé de i crans dans la direction des poids forts avant d'être soustrait du nombre T ainsi modifié. Soit U la nouvelle valeur de T après ces deux opérations. On démontre que ce nombre U est congru modulo N au nombre initial T dont on était parti, et que par ailleurs les i+1 mot de poids faible de U sont nuls.

De proche en proche, en itérant au plus q fois l'opération précédente, on peut ainsi remplacer un nombre T par un nombre U qui lui est congru modulo N et dont les q mots de poids faible sont nuls.

La méthode ainsi décrite, composée d'une suite d'itérations, constitue un premier algorithme qui permet de passer d'un nombre T à nombre V multiple de R et congru à T

modulo N. Ce premier algorithme est d'autant plus rapide que le nombre r_d de mots nécessaires pour stocker d est faible.

Ce premier algorithme est l'un des éléments de base de la présente invention.

Pour effectuer la multiplication modulo N de deux nombres A et B, on calcule dans un premier temps deux nombres F et G à q mots et respectivement congrus modulo N aux produits A.R et B.R. Pour ce faire, on multiplie A, respectivement B, par f pour obtenir un nombre dénommé T_A, respectivement T_B, stocké sur 2q mots. On applique alors à T_A, respectivement T_B, le premier algorithme décrit ci-avant, pour obtenir un nombre V_A, respectivement V_B, congru, modulo N, à T_A, respectivement T_B, et multiple de R.

En extrayant les q mots de poids fort de ces deux résultats on obtient les nombres F et G recherchés. On calcule le produit F.G qu'on note T. Ce nombre, stocké sur 2q bits, est congru modulo N à ABRR. On applique le premier algorithme et on lit les q mots de poids fort pour obtenir un nombre congru modulo N à ABR, et on répète la même opération pour obtenir un nombre congru modulo N à AB. Si ce nombre est inférieur à N, c'est le résultat cherché, à savoir le reste par N du produit des nombres A et B. Si ce nombre n'est pas inférieur à N, on peut démontrer qu'il est inférieur à 2N et il suffira de lui soustraire N pour obtenir le résultat cherché.

La méthode décrite ici prend tout son intérêt lorsque r_d est petit car le premier algorithme est alors rapide, de même que les produits par f qui est alors un nombre à 2 r_d mots. La seule opération longue reste le calcul du produit F.G .

La méthode peut encore être améliorée si on imbrique le calcul du produit F.G et l'application du premier algorithme au résultat de cette multiplication. Cette opération sera réalisée en considérant l'un des nombres facteurs, par exemple G, dans son ensemble, mais en utilisant mot après mot les mots formant l'autre facteur, par exemple F, en commençant par les mots de poids faible.

En pratique, on procède comme suit. Dans une étape préliminaire, on initialise à 0 une zone mémoire Y formée de q+1 mots et d'un bit de retenue. Cette zone mémoire fonctionnera d'une part comme un registre à décalage et d'autre part comme un accumulateur dans lequel s'additionneront les résultats intermédiaires de calcul. On itère alors les opérations suivantes pour un indice i variant de 0 à q-1.

Lors de l'itération d'indice i, on multiplie le mot d'indice i de F par le nombre G pour obtenir un résultat P, sur q+1 mots, qui sera ajouté au contenu de la mémoire Y. Notons que si le jeu d'instructions du processeur utilisé le permet, l'opération d'addition dans un accumulateur peut être réalisée dans la même instruction élémentaire que la multiplication. C'est le cas par exemple lorsqu'on utilise un processeur de type ARM. L'ensemble de ces opérations, multiplication d'un nombre à q mots par un scalaire et addition du résultat au contenu d'une zone mémoire peut alors s'effectuer en q instructions élémentaires du processeur.

On multiplie ensuite le contenu du mot de poids faible de la mémoire Y par le nombre e défini plus haut, et on ne conserve que le mot de poids faible du résultat, multiplication mot-mot sans retenue. Ce mot de poids faible sera appelé m. D'une part m sera ajouté au mot de poids q de Y, avec stockage d'une éventuelle retenue dans le bit retenue de Y. D'autre part le produit de m par d, nombre à r_d+1 mots, sera soustrait à la mémoire Y. A la suite de ces deux opérations, le mot de poids faible de Y est nul. On décale alors le contenu de mots de Y d'un mot vers la droite, le bit de retenue étant recopié dans le mot de poids fort. Cette opération achève l'itération en cours.

Après les q itérations, la mémoire Y contient un nombre inférieur à 2N et dont le produit par R est congru modulo N au produit F.G . Si Y est supérieur ou égal à N, on lui soustrait N .

La mémoire Y contient maintenant le résultat final de toutes ces opérations, à savoir le reste par N d'un nombre dont le produit par R est égal au produit F.G .

L'ensemble des opérations décrites ici constitue un opérateur qui prend comme arguments les deux nombres F et G et fournit comme résultat le reste par N d'un nombre dont le produit par R est égal au produit F.G .

Un des intérêts de l'imbrication de la multiplication avec le premier algorithme est l'économie de mémoire permise par cette méthode. En effet il suffit des q+1 mots, plus un bit, de la mémoire Y pour stocker les produits intermédiaires, au lieu des 2q mots nécessaires au stockage du produit F.G dans le cas où ce produit est calculé préalablement à l'application du premier algorithme.

Voyons maintenant comment la méthode développée ici peut être utilisée pour le calcul des exponentielles modulaires. Remarquons tout d'abord que le calcul d'une exponentiation A^p [mod N] est plus complexe que celui d'une simple multiplication. Un calcul direct de A^p = a.a.a. ... .a, avec p facteurs, représente p-1 multiplications modulo N ce qui, dès que p devient grand, devient très rapidement prohibitif en temps de calcul. Lorsque p atteint l'ordre de grandeur des valeurs habituellement utilisées dans les applications, nombres à plus de 100 chiffres décimaux, le temps de calcul dépasse très significativement l'âge de l'univers, même si l'on faisait travailler en parallèle tous les ordinateurs de la planète.

Une méthode indirecte permet néanmoins de calculer A^p [mod N] en un temps acceptable. Le principe de cette méthode consiste à calculer, par des élévations au carré successive, multiplication d'un nombre par lui-même, modulo N, la liste des puissances de A , modulo N, dont l'exposant est une puissance de 2 , c'est-à-dire (A, A^2 [mod N], A^4 [mod N], A^8 [mod N], A^16 [mod N], A^32 [mod N], A^64 [mod N], et ainsi de suite ...). On examine alors le développement binaire de l'exposant p, c'est sous cette forme que p est stocké en machine. On retient

dans la liste précédente les éléments qui correspondent à des bits à 1 dans ce développement et on les multiplie entre eux, toujours modulo N. Si l'exposant s'écrit sur r bits dont t bits à 1, il faut donc r-1 élévations au carré, suivies de t-1 multiplications pour calculer le résultat.

A titre d'exemple illustratif, prenons le cas de l'exposant 181 qui se décompose en 128+32+16+4+1 s'écrit donc en binaire 1011 0101 . Par 7 élévations au carré successives, on peut calculer la liste des puissances de A d'exposants successifs 2, 4, 8, 16, 32, 64 et 128 . En remarquant que le résultat recherché A^181 est égal au produit des 5 facteurs A^128 [mod N], A^32 [mod N], A^16 [mod N], A^4 [mod N] et A^1 = A , on voit qu'il suffit de 4 multiplications modulo N pour obtenir le résultat final recherché. Soit au total 11 multiplications modulo N au lieu de 180. Le gain est encore considérablement plus spectaculaire lorsque l'exposant est un très grand nombre.

Les algorithmes habituellement implémentés dans les applications informatiques s'inspirent de cette méthode. Leur principe est du domaine public et est décrit par exemple dans « Applied Cryptography » de Bruce Schneier (John Wiley 1994) ou dans « The Art of Computer Programming Volume 2 Semi numerical Algorithms» de D. Knuth (Addison Wesley, 1981 pour la seconde édition).

D'une façon générale, une exponentiation en arithmétique modulo N peut se ramener à approximativement 3/2 fois r multiplications modulo N, où r est le nombre de bits nécessaires à la représentation binaire de l'exposant p de l'exponentielle, soit 768 ou 1536 multiplications modulo N si on travaille avec des exposants à 512 ou 1024 bits. Chacune de ces multiplications modulo N se ramène à une multiplication en grands nombres, temps de calcul proportionnel au carré du nombre de mots nécessaires pour représenter N dans le processeur de calcul, suivi d'une réduction modulo N. Si on effectue cette réduction de façon classique, il faut effectuer une division en

grands nombres, ce qui, en temps de calcul, est équivalent à plusieurs multiplications en grands nombres.

Les méthodes objet de la présente invention permettent d'accélérer significativement les calculs d'exponentiation modulaire.

Le nombre A le nombre destiné à être élevé à la puissance p modulo N est stocké sur q mots. L'exposant p est simplement stocké en mémoire par son développement binaire, en utilisant autant de bits ou de mots que nécessaire. L'algorithme mis en œuvre, on lira successivement les bits de ce développement.

Dans une étape préliminaire, on calcule et on stocke en mémoire un nombre F, stocké sur q mots et égal au reste par N du produit A.R, avec les notations utilisées précédemment. Ce calcul est effectué en utilisant la même méthode que dans le cas de la multiplication.

On se réserve une zone mémoire de 2q mots, notée Y, et destinée à recueillir les résultats partiels des calculs. Dans un premier temps, cette zone mémoire est initialisée au nombre F calculé ci-dessus. Ce nombre F, stocké sur q mots, servira à initialiser les q mots de poids faible de Y, les q mots de poids forts étant initialisés à zéro.

On sélectionnera alors, bit à bit, les bits de l'exposant p, en commençant par les bits de poids fort, et pour chaque bit on effectuera la séquence d'opérations suivantes.

Au début de cette séquence d'opérations, les q mots de poids forts de la zone mémoire Y contiennent la valeur 0. Cette mémoire représente donc un nombre stocké sur q mots, donc un nombre inférieur à R.

On calcule alors le carré de ce nombre, carré dont le stockage nécessite maintenant 2q mots. Soit V_0 le résultat de ce calcul. On applique à V_0 l'algorithme décrit plus haut sous le nom de premier algorithme, qui a pour effet de calculer un nombre U_0 congru modulo N à V_0 et dont les q mots de poids faibles sont nuls. On stocke alors dans les q mots de poids

faibles de la mémoire Y les q mots de poids fort de U_0 et on met à zéro les mots de poids fort de Y. Le contenu de Y est donc obtenu en échangeant les q mots de poids fort et les q mots de poids faibles de U_0.

A ce stade, comme au début de la séquence, les q mots de poids forts de la zone mémoire Y contiennent donc la valeur 0.

Lorsque le bit de p sélectionné au début de la séquence est égal à 0, on réitère la séquence en sélectionnant le bit suivant dans p, dans le sens des poids décroissants.

Lorsque le bit est de p sélectionné au début de la séquence est égal à 1, on effectue en outre les opérations suivantes.

On multiplie le contenu de la mémoire Y, donc les q mots de poids faible, car les autres sont nuls, par le nombre F. Soit V_1 le résultat de ce calcul. Le stockage de V_1 nécessite maintenant 2q mots. La suite des opérations est similaire à ce qui est fait plus haut. On applique à V_1 l'algorithme décrit plus haut sous le nom de premier algorithme, qui a pour effet de calculer un nombre U_1 congru modulo N à V_1 et dont les q mots de poids faibles sont nuls. On stocke alors dans les q mots de poids faibles de la mémoire Y les q mots de poids fort de U_1 et on met à zéro les mots de poids fort de Y. Le contenu de Y est donc obtenu en échangeant les q mots de poids fort et les q mots de poids faibles de U_1.

A ce stade, comme au début de la séquence, les q mots de poids forts de la zone mémoire Y contiennent donc la valeur 0, et on réitère la séquence depuis le début après avoir sélectionné le bit suivant dans p, dans le sens des poids décroissants.

A la fin de ces opérations, après avoir épuisé tous les bits de l'exposant p, la mémoire Y contient un nombre qui est congru modulo N au produit de R par A puissance p. On applique à nouveau le premier algorithme en prenant comme argument le contenu de la mémoire Y pour obtenir un nombre S

congru modulo N à cet argument et dont les q mots de poids faibles sont nuls. Soit D le nombre à q mots formé des mots de poids forts de S. Ce nombre est positif, est inférieur à 2N et est congru modulo N au nombre A puissance p. S'il est inférieur à N, c'est nombre qu'on souhaitait calculer, reste de la division par N du nombre A puissance p. S'il est supérieur ou égal à N, il suffit de lui retrancher N pour trouver le résultat souhaité.

Le procédé de calcul de l'exponentiation modulaire décrit ci-dessus, applique à plusieurs reprises une suite d'opérations consistant à effectuer tout d'abord une multiplication entre deux grands nombres, élévation au carré de Y, ou multiplication de Y par F, puis à appliquer au résultat de cette multiplication le premier algorithme décrit précédemment et consistant à remplacer ledit résultat par un nombre qui lui est congru modulo N et dont les q mots de poids faibles sont nuls, et finalement à extraire le nombre formé des q mots de poids forts du résultat du premier algorithme.

De même que nous avons pu imbriquer ces opérations dans le cas de la multiplication, nous pouvons aussi les imbriquer ici, et remplacer la suite d'opérations ci-dessus par l'application de l'opérateur décrit précédemment, qui étant donné deux arguments, selon le cas Y et le même Y ou Y et F, fournit comme résultat le reste par N d'un nombre dont le produit par R est égal au produit des arguments de l'opérateur.

Cela a pour conséquence entre autres une économie de mémoire, car la mémoire de travail Y ne nécessite plus alors que q+1 bits (plus un bit de parité) au lieu des 2q bits nécessaires si on n'imbrique pas les opérations.

REVENDICATIONS

1. Procédé permettant d'accélérer le calcul de multiplications en arithmétique modulo N, N désignant un entier comportant notamment plusieurs centaines de chiffres décimaux ; lesdites multiplications intervenant notamment dans des protocoles de cryptographie mis en œuvre à l'aide de ressources informatiques ; ledit procédé étant plus particulièrement conçu pour économiser lesdites ressources informatiques, telles que puissance de calcul et/ou espace mémoire ;

ledit procédé comprenant les étapes préliminaires suivantes :

- l'étape de stocker ledit nombre N sur q mots,
- l'étape de soustraire ledit nombre N d'une puissance de 2 supérieure à N, notée R, pour obtenir un nombre, noté d ;

ledit nombre R étant la première puissance de 2 ne pouvant pas être stockée sur q mots ;

ledit procédé comprenant en outre l'étape préliminaire suivante :

- l'étape de stocker ledit nombre d dans une mémoire de ladite ressource informatique ;

ledit procédé comportant un premier opérateur prenant en entrée deux nombres E_1, E_2 stockés chacun sur q mots et fournissant en sortie un nombre W tel que le produit par R dudit nombre W soit congru modulo N au produit desdites entrées E_1, et E_2 les dites entrées E_1, E_2 étant ci-après dénommées les entrées dudit premier opérateur, et le nombre W étant ci-après dénommé le résultat dudit premier opérateur ;

ledit procédé comportant un premier algorithme ayant pour objet de remplacer un argument, stocké sur 2q mots, par un résultat qui est congru modulo N audit argument et dont les q mots de poids faibles sont nuls ; ledit argument étant ci-après dénommé l'argument dudit premier algorithme ; ledit résultat étant ci-après dénommé le résultat dudit premier algorithme ;

de sorte que le résultat dudit premier algorithme est un multiple de R ;

ledit procédé comprenant en outre les étapes principales suivantes pour multiplier entre eux modulo N deux nombres A et B,

- l'étape de stocker en mémoire les restes modulo N du produit par R des nombres A et B, lesdits restes stockés en mémoire étant respectivement dénommés le nombre F et le nombre G,

- l'étape de mettre en œuvre le premier opérateur, les entrées E1 et E2 prenant respectivement comme valeur les nombres F et G,

- l'étape de stocker le résultat W du premier opérateur sous la forme d'un nombre à 2q mots, ci-après dénommé H ;

de sorte que ce nombre H est égal au reste modulo N du produit des nombres A, B et R ;

ledit procédé comprenant en outre l'étape principale suivante :

- l'étape de mettre en œuvre ledit premier algorithme avec comme argument ledit nombre H, stocké sur 2q mots, pour fournir un nombre V congru modulo N audit nombre H et dont les q mots de poids faibles sont nuls ;

- l'étape de lire le nombre à q mots C constitué par les q mots de poids fort dudit nombre V ;

ledit nombre C ou le nombre C-N étant égal au reste modulo N du produit des nombres A et B.

2. Procédé selon la revendication 1 ; ledit premier algorithme consistant à itérer q fois une opération ayant pour effet de remplacer le nombre obtenu au terme de l'itération précédente par un nombre qui lui est congru modulo N et tel que le nombre obtenu au terme de la ième itération ait tous ses mots de poids les plus faibles nuls jusqu'au ième ; ladite ième opération comprenant les quatre étapes suivantes :

(a) on multiplie le ième mot, dans l'ordre des poids croissants, du nombre issu de l'itération précédente, ou, lors de la première itération, le mot de poids faible de l'argument dudit premier algorithme, par une constante prédéfinie de manière à obtenir un résultat dont on ne conserve que le mot de poids faible,

(b) on obtient au terme de l'étape précédente (a) un résultat, dénommé x, qu'on multiplie par ledit nombre d,

(c) on décale le résultat x de la première étape (a) de q+i crans vers les poids forts et on ajoute le nombre ainsi décalé au nombre issu de l'itération précédente ou lors de la première itération à l'argument dudit premier algorithme,

(d) on obtient au terme de l'étape (b) un résultat, dénommé y, qu'on décale de i crans dans le sens des poids forts puis qu'on soustrait au nombre issu du résultat de l'addition effectuée à l'étape (c) précédente ;

le résultat de cette soustraction constitue le résultat de l'itération en cours ;

de sorte qu'on obtient au terme desdites quatre étapes de la ième itération un nombre qui est congru modulo N audit argument dudit premier algorithme et dont tous les mots de poids les plus faibles sont nuls jusqu'au ième.

3. Procédé selon la revendication 1 ou 2 ; ledit premier opérateur comprend les étapes suivantes :

l'étape de multiplier entre elles les entrées E1 et E2 dudit premier opérateur, ayant pris pour valeurs les nombres F et G, le résultat de cette multiplication étant un nombre T stocké sur 2q mots,

- l'étape de mettre en œuvre ledit premier algorithme avec comme argument ledit nombre T pour fournir comme résultat un nombre dénommé U, stocké sur 2q mots,

- l'étape de lire le nombre à q mots constitué par les q mots de poids fort dudit nombre U, ledit nombre constituant le résultat W dudit premier opérateur ;

de sorte que les q mots de poids faible de U sont nuls ;

de sorte que ledit résultat W multiplié par R est congru modulo N au produit desdites entrées E1 et E2 ;

4. Procédé selon la revendication 1 ou 2 ; ledit premier opérateur comprenant les étapes préliminaires suivantes :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,
- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 dudit premier opérateur, ladite entrée E1 ayant pris pour valeur le nombre F ;

ledit premier opérateur comprenant l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier ledit mot ainsi sélectionné par l'entrée E2 ayant pris pour valeur le nombre G, le résultat de cette multiplication étant un nombre stocké sur q+1 mots et étant dénommé P,
- (b) la sous-étape d'additionner ledit nombre P, calculé à l'étape précédente, au contenu de ladite mémoire Z, ledit bit de retenue étant positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,
- (c) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente,
- (d) la sous-étape de multiplier le contenu du mot de poids faible de ladite mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,
- (e) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre N, le résultat de cette multiplication étant un nombre dénommé Q stocké sur q+1 mots,
- (f) la sous-étape d'additionner ledit nombre Q, calculé à l'étape (e) précédente, au contenu de la mémoire Z,
- (g) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente ;

de sorte que le mot de poids faible de ladite mémoire Z contient la valeur zéro ;

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de ladite mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de ladite mémoire Z le contenu dudit bit de retenue,

- (j) la sous-étape de mettre à zéro ledit bit de retenue ;

ledit premier opérateur comprenant en outre l'étape principale suivante :

- l'étape de lire le mot suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j), autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, ladite mémoire Z contient un nombre, compris entre zéro et 2N, dont le produit par R est égal, modulo N, au produit des entrées E1 et E2 ;

ledit premier opérateur comprenant en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de ladite mémoire Z,

- l'étape, lorsque ce contenu est supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction étant stocké dans la mémoire Z ;

le résultat W dudit premier opérateur étant égal au contenu de la mémoire Z.

5. Procédé selon la revendication 1 ou 2 ;

ledit premier opérateur comprenant l'étape préliminaire suivante :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 dudit premier opérateur, ladite entrée E1 ayant pris pour valeur le nombre F ;

ledit premier opérateur comprenant l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier ledit mot ainsi sélectionné par ladite entrée E2 dudit premier opérateur, ladite entrée E2 ayant pris pour valeur le nombre G, le résultat de cette multiplication étant un nombre stocké sur q+1 mots et étant dénommé P,

- (b) la sous-étape d'additionner ledit nombre P, calculé à l'étape précédente, au contenu de ladite mémoire Z, ledit bit de retenue étant positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de ladite mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) les sous-étapes :

• d'additionner le résultat m de l'étape précédente au contenu du mot de poids fort de ladite mémoire Z,

• de stocker le résultat de cette addition dans le même mot de poids fort,

• en cas de dépassement de capacité de cette addition, de mettre à 1 ledit bit de retenue,

- (f) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre d, le résultat de cette multiplication étant un nombre dénommé Q,

- (g) la sous-étape de soustraire ledit nombre Q, calculé à l'étape précédente, au contenu de ladite mémoire Z, puis de stocker dans ladite mémoire Z le résultat de cette soustraction ;

de sorte que le mot de poids faible de ladite mémoire Z contient la valeur zéro ;

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de ladite mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de ladite mémoire Z le contenu dudit bit de retenue,

- (j) la sous-étape de mettre à zéro ledit bit de retenue ;

ledit premier opérateur comprenant en outre l'étape principale suivante :

- l'étape de lire le mot suivant de ladite entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j) autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, ladite mémoire Z contient un nombre H dont le produit par R est égal, modulo N, au produit des nombres E1 et E2 ;

ledit premier opérateur comprenant en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de ladite mémoire Z,

- l'étape, lorsque ce contenu supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction étant stocké dans la mémoire Z ;

le résultat W dudit premier opérateur étant égal au contenu de la mémoire Z ;

6. Procédé selon l'une quelconque des revendications 1 à 5 ; ledit procédé étant tel que pour calculer le reste modulo N, dénommé le nombre F, respectivement dénommé le nombre G , du produit par R d'un nombre donné A, respectivement B :

- on multiplie le nombre A, respectivement le nombre B, par le reste modulo N du carré de R, pour obtenir un nombre dénommé T_A, respectivement T_B, stocké sur 2q mots,

- on applique ledit premier algorithme avec comme argument ledit nombre T_A, respectivement ledit nombre T_B, pour obtenir comme résultat un nombre V_A, respectivement un nombre V_B, qui lui est congru modulo N et dont les q mots de poids faibles sont nuls ;

ledit nombre F, respectivement ledit nombre G, est alors le nombre à q mots formés des q mots de poids forts dudit nombre V_A, respectivement dudit nombre V_B.

7. Procédé permettant d'accélérer le calcul d'exponentiations en arithmétique modulo N, N désignant un entier comportant notamment plusieurs centaines de chiffres décimaux ; lesdites exponentiations intervenant notamment dans des protocoles de cryptographie mis en œuvre à l'aide de ressources informatiques ; ledit procédé ayant pour objet d'élever un nombre A, ci après dénommé la base, à une puissance p modulo N ; p étant un nombre entier supérieur ou égal à 2, ci-après dénommé l'exposant, et étant représenté sous forme binaire ; ledit procédé étant plus particulièrement conçu pour économiser lesdites ressources informatiques, telles que puissance de calcul et/ou espace mémoire ;

ledit procédé comprenant les étapes préliminaires suivantes :

- l'étape de stocker ledit nombre N sur q mots,
- l'étape de soustraire ledit nombre N d'une puissance de 2 supérieure à N, notée R, pour obtenir un nombre, noté d ;

ledit nombre R étant la première puissance de 2 ne pouvant pas être stockée sur q mots ;

- l'étape de stocker ledit nombre d dans une mémoire de ladite ressource informatique ;

ledit procédé comprenant en outre les étapes préliminaires suivantes :

- l'étape de stocker en mémoire le reste modulo N du produit par R de la base A, ledit reste stocké en mémoire étant dénommé le nombre F,

- l'étape d'initialiser une zone mémoire de 2q mots, ci-après dénommée la zone mémoire Y, avec une copie nombre F ;

ledit procédé comportant un premier algorithme ayant pour objet de remplacer un argument, stocké sur 2q mots, par un résultat qui est congru modulo N audit argument et dont les q mots de poids faibles sont nuls ; ledit argument étant ci-après dénommé l'argument dudit premier algorithme ; ledit résultat étant ci-après dénommé le résultat dudit premier algorithme ;

de sorte que le résultat dudit premier algorithme est un multiple de R ;

ledit procédé comprenant les étapes principales suivantes :

- l'étape de sélectionner, en commençant par les bits de poids forts, le premier bit suivant le premier bit non nul de l'exposant p,

- l'étape consistant à mettre en œuvre, selon le cas, les opérations ci-après définies :

dans le cas où le bit sélectionné est égal à zéro :

- (a) on calcule le carré du contenu de la zone mémoire Y ;
- (b) on stocke dans la zone mémoire Y, le résultat du calcul précédent ;
- (c) on met en œuvre ledit premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir comme résultat un nombre U_0 congru modulo N à cet argument et dont les q mots de poids faibles sont nuls ;
- (d) on stocke dans les q mots de poids faible de la zone mémoire Y, les q mots de poids forts du résultat U_0 du calcul précédent ;
- (e) on met à zéro le contenu des q mots de poids forts de la mémoire Y ;

dans le cas où le bit sélectionné est égal à un, on réalise les mêmes opérations (a) à (e) que dans le cas où ledit exposant sélectionné est égal à zéro et en outre :

- (f) on multiplie le contenu de la mémoire Y, au terme de l'opération (e), par le nombre F ;
- (g) on stocke dans la zone mémoire Y, le résultat du calcul précédent ;
- (h) on met en œuvre ledit premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir comme résultat un nombre U_1 congru modulo N à cet argument et dont les q mots de poids faibles sont nuls ;
- (i) on stocke dans les q mots de poids faible de la zone mémoire Y, les q mots de poids forts du résultat U_1 du calcul précédent ;
- (j) on met à zéro le contenu des q mots de poids forts de la mémoire Y ;

ledit procédé comprenant en outre l'étape principale suivante :

- l'étape de lire le bit suivant de l'exposant p, dans l'ordre des poids décroissants, et
- lorsque ledit bit est égal à zéro on itère les opérations (a) à (e) ;
- lorsque ledit bit est égal à un on itère les opérations (a) à (j) ;

lesdites itérations étant effectuées autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné tous les bits de l'exposant, la mémoire Y contient un nombre qui est congru modulo N au produit du nombre R par le résultat de l'exponentiation A puissance p ;

ledit procédé comprenant en outre l'étape de mettre en œuvre ledit premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir un nombre S congru modulo N à cet argument et dont les q mots de poids faibles sont nuls ;

de sorte que les q mots de poids fort du nombre S représentent un nombre compris entre zéro et 2N et congru modulo N au résultat de l'exponentiation A puissance p ;

ledit procédé comprenant en outre les étapes finales suivantes :

- l'étape de lire le nombre à q mots D constitué par les q mots de poids fort dudit nombre S,

- l'étape de comparer ce nombre au nombre N

- l'étape, lorsque D est supérieur à N, de retrancher le nombre N du nombre D ;

de sorte que le nombre D auquel on a éventuellement soustrait N est égal au résultat de l'exponentiation A puissance p modulo N qu'on souhaite calculer.

8. Procédé selon la revendication 7 ; ledit procédé comprenant les étapes préliminaires suivantes :

- l'étape de stocker en mémoire le reste modulo N du produit par R du nombre A, ledit reste stocké en mémoire étant dénommé F ;

- l'étape d'initialiser une zone mémoire de q mots, ci-après dénommée la zone mémoire Y, avec le nombre F ;

ledit procédé comportant un premier opérateur prenant en entrée deux nombres E1, E2 stockés chacun sur q mots et fournissant en sortie un nombre W, stocké sur q mots, tel que le produit par R dudit nombre W soit congru modulo N au produit desdites entrées E1, et E2 les dites entrées E1, E2 étant ci-après dénommées les entrées dudit premier opérateur, et le nombre W étant ci-après dénommé le résultat dudit premier opérateur ;

ledit procédé comprenant les étapes principales suivantes :

- l'étape de sélectionner, en commençant par les bits de poids forts, le premier bit suivant le premier bit non nul de l'exposant p,

- l'étape consistant à mettre en œuvre, selon le cas, les opérations ci-après définies :

dans le cas où le bit sélectionné est égal à zéro :

(a) l'étape de mettre en œuvre ledit premier opérateur les deux entrées E1 et E2 dudit premier opérateur prenant toutes deux une valeur égale au contenu de la zone mémoire Y,

(b) l'étape de stocker le résultat W fourni par ledit premier opérateur dans la zone mémoire Y,

dans le cas où le bit sélectionné est égal à un :

on réalise les mêmes opérations (a) et (b) que dans le cas où ledit exposant sélectionné est égal à zéro et en outre les deux étapes suivantes :

(c) l'étape de mettre en œuvre ledit premier opérateur les entrées E1 et E2 dudit opérateur prenant l'une pour valeur le nombre F et l'autre une valeur égale au contenu de la zone mémoire Y,

(d) l'étape de stocker le résultat W fourni par ledit premier opérateur dans la zone mémoire Y,

ledit procédé comprenant en outre l'étape principale suivante :

- l'étape de lire le bit suivant de l'exposant p, dans l'ordre des poids décroissants, et

• lorsque ledit bit est égal à zéro on itère les opérations (a) à (b) ;

• lorsque ledit bit est égal à un on itère les opérations (a) à (d) ;

lesdites itérations étant effectuées autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné tous les bits de l'exposant, la mémoire Y contient un nombre qui est congru modulo N au produit du nombre R par le résultat de l'exponentiation A puissance p ;

ledit procédé comprenant en outre l'étape finale de mettre en œuvre ledit premier algorithme avec comme argument le contenu de la mémoire Y pour obtenir un nombre S congru modulo N à cet argument et dont les q mots de poids faibles sont nuls ;

de sorte que les q mots de poids fort du nombre S représentent un nombre congru modulo N au résultat de l'exponentiation A puissance p ;

ledit procédé comprenant en outre les étapes finales suivantes :

- l'étape de lire le nombre à q mots D constitué par les q mots de poids fort dudit nombre S,

- l'étape de comparer ce nombre au nombre N

- l'étape, lorsque D est supérieur à N, de retrancher le nombre N du nombre D ;

de sorte que le nombre D auquel on a éventuellement soustrait N est égal au résultat de l'exponentiation A puissance p modulo N qu'on souhaite calculer.

9. Procédé selon la revendication 7 ou 8 ; ledit premier algorithme consistant à itérer q fois une opération ayant pour effet de remplacer le nombre obtenu au terme de l'itération précédente par un nombre qui lui est congru modulo N et tel que le nombre obtenu au terme de la ième itération ait tous ses mots de poids les plus faibles nuls jusqu'au ième ; ladite ième opération comprenant les quatre étapes suivantes :

(a) on multiplie le ième mot, dans l'ordre des poids croissants, du nombre issu de l'itération précédente, ou lors de la première itération le mot de poids faible de l'argument dudit premier algorithme, par une constante prédéfinie de manière à obtenir un résultat dont on ne conserve que le mot de poids faible,

(b) on obtient au terme de l'étape précédente (a) un résultat, dénommé x, qu'on multiplie par ledit nombre d,

(c) on décale le résultat x de la première étape (a) de q+i crans vers les poids forts et on ajoute le nombre ainsi décalé au nombre issu de l'itération précédente ou lors de la première itération à l'argument dudit premier algorithme,

(d) on obtient au terme de l'étape (b) un résultat, dénommé y, qu'on décale de i crans dans le sens des poids forts

puis qu'on soustrait au nombre issu du résultat de l'addition effectuée à l'étape (c) précédente,

le résultat de cette soustraction constitue le résultat de l'itération en cours,

de sorte qu'on obtient au terme desdites quatre étapes de la ième itération un nombre qui est congru modulo N audit argument dudit premier algorithme et dont tous les mots de poids les plus faibles sont nuls jusqu'au ième.

10. Procédé selon la revendication 8 ou 9 ; ledit premier opérateur comprend les étapes suivantes :

- l'étape de multiplier entre elles les entrées E1 et E2 dudit premier opérateur pour obtenir un nombre T stocké sur 2q mots,

- l'étape de mettre en œuvre ledit premier algorithme avec comme argument ledit nombre T pour fournir comme résultat un nombre dénommé U, stocké sur 2q mots,

- l'étape de lire le nombre à q mots constitué par les q mots de poids fort dudit nombre U, ledit nombre à q mots constituant le résultat W dudit premier opérateur ;

de sorte que les q mots de poids faible de U sont nuls ;

de sorte que ladite sortie W multipliée par R est congrue modulo N au produit desdites entrées E1 et E2.

11. Procédé la revendication 8 ou 9 ;

ledit premier opérateur comprenant les étapes préliminaires suivantes :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 dudit premier opérateur,

ledit premier opérateur comprenant l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier ledit mot ainsi sélectionné par l'entrée E2, le résultat de cette multiplication étant un nombre stocké sur q+1 mots et étant dénommé P,

- (b) la sous-étape d'additionner ledit nombre P, calculé à l'étape précédente, au contenu de ladite mémoire Z, ledit bit de retenue étant positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de ladite mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre N, le résultat de cette multiplication étant un nombre dénommé Q stocké sur q+1 mots,

- (f) la sous-étape d'additionner ledit nombre Q, calculé à l'étape (e) précédente, au contenu de la mémoire Z,

- (g) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente ;

de sorte que le mot de poids faible de ladite mémoire Z contient la valeur zéro ;

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de ladite mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de ladite mémoire Z le contenu dudit bit de retenue,

- (j) la sous-étape de mettre à zéro ledit bit de retenue ;

ledit premier opérateur comprenant en outre l'étape principale suivante :

- l'étape de lire le mot suivant de l'entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j), autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, ladite mémoire Z contient un nombre, compris entre zéro et 2N, dont le produit par R est égal, modulo N, au produit des nombres E1 et E2 ;

ledit premier opérateur comprenant en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de ladite mémoire Z,

- l'étape, lorsque ce contenu est supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction étant stocké dans la mémoire Z ;

le résultat W dudit premier opérateur étant égal au contenu de la mémoire Z.

12. Procédé selon la revendication 8 ou 9 ;

ledit premier opérateur comprenant l'étape préliminaire suivante :

- l'étape d'initialiser à zéro une zone mémoire Z formée de q+1 mots et d'un bit de retenue,

- l'étape de sélectionner, en commençant par les mots de poids faibles, le premier mot de l'entrée E1 dudit premier opérateur ;

ledit premier opérateur comprenant l'étape principale consistant à itérer les sous-étapes suivantes :

- (a) la sous-étape de multiplier ledit mot ainsi sélectionné par ladite entrée E2 dudit premier opérateur, le résultat de cette multiplication étant un nombre stocké sur q+1 mots et étant dénommé P,

- (b) la sous-étape d'additionner ledit nombre P, calculé à l'étape précédente, au contenu de ladite mémoire Z, ledit bit de retenue étant positionné à 1 ou à 0 selon qu'il y a ou non dépassement de capacité,

- (c) la sous-étape de stocker dans ladite mémoire Z le résultat de l'addition effectuée dans l'étape précédente,

- (d) la sous-étape de multiplier le contenu du mot de poids faible de ladite mémoire Z par une constante prédéfinie, de manière à obtenir un résultat dont on ne conservera que le mot de poids faible, dénommé m,

- (e) les sous-étapes :

• d'additionner le résultat m de l'étape précédente au contenu du mot de poids fort de ladite mémoire Z,

• de stocker le résultat de cette addition dans le même mot de poids fort,

• en cas de dépassement de capacité de cette addition, de mettre à 1 ledit bit de retenue,

- (f) la sous-étape de multiplier le résultat m de l'étape précédente par le nombre d, le résultat de cette multiplication étant un nombre dénommé Q,

- (g) la sous-étape de soustraire ledit nombre Q, calculé à l'étape précédente, au contenu de ladite mémoire Z, puis de stocker dans ladite mémoire Z le résultat de cette soustraction ;

de sorte que le mot de poids faible de ladite mémoire Z contient la valeur zéro ;

- (h) la sous-étape de décaler d'un mot dans la direction des mots de poids faibles le contenu des mots de ladite mémoire Z,

- (i) la sous-étape de stocker dans le mot de poids fort de ladite mémoire Z le contenu dudit bit de retenue,

- (j) la sous-étape de mettre à zéro ledit bit de retenue ;

ledit premier opérateur comprenant en outre l'étape principale suivante :

- l'étape de lire le mot suivant de ladite entrée E1, dans le sens des poids croissants et de réitérer les sous étapes précédentes (a) à (j) autant de fois que nécessaire ;

de sorte qu'au terme du processus, après avoir sélectionné, dans l'ordre des poids croissants, tous les mots constituant l'entrée E1, ladite mémoire Z contient un nombre H

dont le produit par R est égal, modulo N, au produit des nombres E1 et E2 ;

ledit premier opérateur comprenant en outre les deux étapes finales suivantes :

- l'étape de comparer au nombre N le contenu de ladite mémoire Z,

- l'étape, lorsque ce contenu supérieur à N, de lui soustraire le nombre N, le résultat de la soustraction étant stocké dans la mémoire Z ;

le résultat W dudit premier opérateur étant égal au contenu de la mémoire Z.

13. Procédé selon l'une quelconque des revendications 7 à 12 ; ledit procédé étant tel que pour calculer le reste modulo N du produit par R de la base A, dénommé le nombre F :

- on multiplie le nombre A par le reste modulo N du carré de R, pour obtenir un nombre dénommé T, stocké sur 2q mots,

- on applique ledit premier algorithme avec comme argument ledit nombre T, pour obtenir comme résultat un nombre V, qui lui est congru modulo N et dont les q mots de poids faibles sont nuls ;

ledit nombre F est alors le nombre à q mots formés des q mots de poids forts dudit nombre V.